FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

# Zadání bakalářské práce

| | |
|---|---|
| **Název:** | Aplikace rugby - modul organizace turnajů a zápasů |
| **Student:** | Matěj Ulman |
| **Vedoucí:** | Ing. Jiří Chludil |
| **Studijní program:** | Informatika |
| **Obor / specializace:** | Webové a softwarové inženýrství, zaměření Softwarové inženýrství |
| **Katedra:** | Katedra softwarového inženýrství |
| **Platnost zadání:** | do konce letního semestru 2022/2023 |

## Pokyny pro vypracování

Aplikace rugby je mobilní aplikace pro OS Android zaměřující se na organizaci rugbyových klubů (tréninky, turnaje, soupisky hráčů, výzvy, atd...).

1. Analyzujte požadavky na organizaci turnajů a zápasů, zapisování výsledků zápasů a jejich zpracování a zasílání notifikací uživatelům v rámci systému.
2. Prozkoumejte existující aplikace a jimi nabízené možnosti a nedostatky
3. Pomocí metod softwarového inženýrství navrhněte architekturu umožňující implementaci všech modulů v rámci jedné aplikace
4. Proveďte implementaci prototypu
5. Implementovaný prototyp modulu podrobte integračním, akceptačním a uživatelským testům, a to v rámci celé aplikace.

---

*Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 30. března 2021 v Praze.*

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

# Aplikace rugby - modul organizace turnajů a zápasů

***Matěj Ulman***

Katedra softwarového inženýrství
Vedoucí práce: Ing. Jiří Chludil

13. května 2021

Tuto práci věnuji své babičce Haně Veselé, která 6. 2. 2021 prohrála svůj boj s covidem.

# Poděkování

Děkuji Ing. Jiřímu Chludilovi za jeho vedení a rady při tvorbě této práce. Dále děkuji Martinu Paulovi, Danielovi Karlovskému a Janu Špetlovi za spolupráci na projektu.

# Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principu při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisu. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisu, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programu, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli zpusobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelum). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným zpusobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 13. května 2021 .....

### Odkaz na tuto práci

Ulman, Matěj. *Aplikace rugby - modul organizace turnajů a zápasů.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2021.

# Abstrakt

Tato práce se zabývá vývojem mobilní aplikace pro správu sportovních klubů hrajících rugby. Aplikace je vyvíjena v rámci společného projektu, s cílem zjednodušení organizace a zajištění různých funkcí v rámci jednoho systému. Tato část se konkrétně zabývá pořádáním turnajů a zápasů mezi kluby. V práci jsou popsány požadavky, na základě kterých je představen návrh systému, skládajícího se z aplikace pro operační systém Android a serverové části pro poskytování a ukládání dat. Jsou představeny použité technologie, především platforma Android, framework Spring a technologie Firebase, používaná pro zasílání notifikací v systému. Výsledný prototyp je podroben testování a jsou představena možná budoucí vylepšení.

**Klíčová slova** mobilní aplikace, informační systém, Android, MVVM, Spring, Firebase

# Abstract

This thesis deals with the development of a mobile application for managing rugby sport clubs. The application is a part of a joint project, with the goal of simplifying organization and handling various functions in a single system. This thesis in particular focuses on tournament and match management between clubs. Concrete requirements are analysed, and serve as a base for a later design of a system, comprised of an application for the Android operating system, and a server portion for providing and storing data. Used technologies are described, mainly the Android platform, the Spring framework and the Firebase technology, used for sending notifications inside the system. The end result is working prototype, which is then tested, and possible future improvements are presented.

**Keywords** mobile application, information system, Android, MVVM, Spring, Firebase

# Obsah

**Úvod** **1**

**1 Cíl práce** **3**

**2 Analýza** **5**
2.1 Představení existujícího systému . . . . . 5
2.1.1 Aktuální možnosti systému . . . . . 5
2.2 Požadavky . . . . . 6
2.2.1 Funkční požadavky . . . . . 6
2.2.2 Nefunkční požadavky . . . . . 9
2.3 Existující řešení . . . . . 9
2.3.1 Winner . . . . . 10
2.3.2 Tournament Manager . . . . . 10
2.3.3 Challenge Place . . . . . 10
2.3.4 Shrnutí . . . . . 11
2.4 Model případů užití . . . . . 12

**3 Návrh** **21**
3.1 Architektura systému . . . . . 21
3.1.1 Prezentační vrstva . . . . . 21
3.1.2 Aplikační vrstva . . . . . 21
3.1.3 Datová vrstva . . . . . 22
3.2 Architektura Android aplikace . . . . . 22
3.3 Doménový model . . . . . 22
3.4 Plánování zápasů . . . . . 26
3.4.1 Formáty . . . . . 27
3.4.2 Úpravy harmonogramu . . . . . 28
3.5 Návrh uživatelského rozhraní . . . . . 28

**4 Implementace** **43**
4.1 Použité technologie . . . . . 43
4.2 Představení Android platformy . . . . . 45
4.2.1 Prvky uživatelského rozhraní . . . . . 45
4.2.2 Aplikační manifest . . . . . 46
4.2.3 Resources, lokalizace . . . . . 46
4.2.4 Activity a Fragment . . . . . 47
4.3 Implementace Model-View-ViewModel . . . . . 47
4.4 Realizace notifikací . . . . . 48
4.4.1 Inicializace . . . . . 49
4.4.2 Zasílání zpráv . . . . . 49
4.4.3 Příjem zpráv . . . . . 49
4.4.4 Zobrazení notifikace . . . . . 49
4.5 Synchronizace aplikace s databází . . . . . 50
4.5.1 Reakce na databázové operace . . . . . 50
4.5.2 Implementace oboustranného připojení . . . . . 50
4.5.3 Zasílání upozornění na změny . . . . . 51

**5 Testování** **53**
5.1 Automatické testy . . . . . 53
5.1.1 Jednotkové testy . . . . . 53
5.1.2 Integrační testy . . . . . 54
5.1.2.1 Testy UI . . . . . 54
5.2 Uživatelské testování a akceptační testování . . . . . 55

**6 Budoucí rozšíření** **57**

**Závěr** **59**

**Literatura** **61**

**A Seznam použitých zkratek** **63**

**B Instalační příručka** **65**

**C Uživatelská příručka** **71**

**D Testovací scénáře** **73**

**E Obsah přiloženého média** **77**

# Seznam obrázků


2.1 Winner – seznam zápasů . . . . . 11
2.2 Winner – tabulka výsledků . . . . . 11
2.3 Tournament Manager – správa turnaje . . . . . 12
2.4 Tournament Manager – tabulka výsledků . . . . . 12
2.5 Challenge Place – správa turnaje . . . . . 13
2.6 Challenge Place – zapisování zápasu . . . . . 13
2.7 Zapisování hřiště . . . . . 14
2.8 Případy užití pro správa zápasů . . . . . 16
2.9 Případy užití pro zaznamenávání výsledků . . . . . 18

3.1 Doménový model . . . . . 23
3.2 Hlavní obrazovka . . . . . 29
3.3 Seznam veřejných turnajů . . . . . 29
3.4 Seznam spravovaných turnajů . . . . . 31
3.5 Vytvoření turnaje . . . . . 31
3.6 Detaily turnaje . . . . . 32
3.7 Zvolení správců . . . . . 32
3.8 Nominace týmů na turnaj . . . . . 33
3.9 Pozvání klubů na turnaj . . . . . 33
3.10 Správa hřišť . . . . . 34
3.11 Výběr týmů pro turnaj . . . . . 34
3.12 Seznam zápasů turnaje . . . . . 35
3.13 Seznam zápasů rozhodčího . . . . . 35
3.14 Diagram zápasů turnaje . . . . . 35
3.15 Automatické vytvoření zápasů . . . . . 37
3.16 Možnosti úprav harmonogramu . . . . . 37
3.17 Detaily zápasu . . . . . 38
3.18 Zapisování výsledků . . . . . 38
3.19 Zapisování hřiště . . . . . 39
3.20 Zapisování zápasu . . . . . 39

3.21 Nastavení notifikace zápasu . . . . . 41
3.22 Odpovědi na zaslanou notifikaci . . . . . 41
3.23 Přijatá notifikace . . . . . 42

B.1 Podoba po importování Android projektu . . . . . 67
B.2 Vytvoření apk aplikace . . . . . 67
B.3 Spuštění aplikace na fyzickém zařízení . . . . . 68
B.4 Spuštění serveru . . . . . 69
B.5 Stažení JDK . . . . . 69

C.1 Nastavení IP adresy . . . . . 72

# Úvod

Tato práce pokračuje v existujícím projektu, jehož cílem je vytvoření informačního systému pro správu rugby sportovních klubů. Systém je vytvářen podle požadavků a nápadů členů klubu *RC Říčany*, s cílem převést aktuálně decentralizovanou organizaci pod jeden systém.

Konkrétně bude v této práci systém rozšířen pro podporu pořádání turnajů a zápasů, spolu se souvisejícími funkcemi – nominování týmů, zaznamenávání výsledků, systémem vytvářený harmonogram zápasů a další. Dále bude v systému přidána podpora pro zasílání notifikací uživatelům systému.

Práce je rozdělena na čtyři kapitoly, věnující se analýze, návrhu, implementaci a testování systému.

V první kapitole bude představena aktuální podoba systému, budou analyzovány a podrobně popsány konkrétní požadavky na aplikaci a z těchto požadavků jsou následně sestaveny případy užití.

V kapitole návrhu bude popsána aktuální architektura systému a navrhované změny, bude představen doménový model a navrhované uživatelské prostředí.

Implementační kapitola se věnuje představení použitých technologií a popisu řešení klíčových problémů při realizaci.

V poslední kapitole budou popsány testy, kterým byl prototyp podroben, od vytvořených automatických testů po manuální uživatelské testy.

Rozšíření systému se věnují i další tři souběžné práce. Výsledný prototyp tak bude obsahovat všechny implementované funkce v rámci jedné aplikace.

Martin Paul, modul tréninky [1].

Daniel Karlovský, modul zákonného zástupce [2].

Jan Špetl, modul výzev.

Kapitola 1

# Cíl práce

Cílem práce je rozvinutí stávajícího systému pro podporu pořádání rugby turnajů a zápasů. Aplikace bude podporovat založení nových turnajů, nominace týmů, plánování harmonogramu turnaje a zaznamenávání a zpracování výsledků zápasů. Dále bude systém rozšířen o podporu zasílání notifikací uživatelům aplikace, které budou využívány pro upozornění na zahájení zápasů, s možností dalšího využití v budoucnu. Cílem jednotlivých částí práce je tak zjistit konkrétní požadavky na systém a prozkoumat existující řešení, s ohledem na celkové požadavky plynoucí z dalších prací. Z výsledků analýzy bude proveden návrh a následná implementace nových funkcí.

Výsledný prototyp, obsahující funkce jak z této, tak i z ostatních souvisejících prací, bude podroben akceptačním testům pro ověření naplnění požadavků a uživatelským testům, pro kontrolu přívětivosti samotné aplikace.

Kapitola 2

# Analýza

## 2.1 Představení existujícího systému

Tato práce vychází ze společného týmového projektu, ve kterém byl navržen a implementován základní prototyp pro správu rugby klubů. Některé části původního systému budou zachovány, případně upraveny. V případě využití existujících částí se jedná o mé vlastní návrhy. Výjimkou je později zmíněná obrazovka pro zaznamenávání výsledků, autorem které je Martin Paul.

### 2.1.1 Aktuální možnosti systému

Pro lepší přehled o celém systému zde budou stručně představeny některé jeho hlavní funkce, na kterých se bude v práci dále stavět.

**Správa klubů** V rámci systému je možno vytvářet a spravovat kluby. Ke konkrétním klubům jsou přiřazovány hráči, týmy, turnaje a zápasy.

**Správa hráčů a týmů** Každý klub spravuje své hráče. Z hráčů klubu jsou sestavovány týmy, které mohou být nominovány na turnaje a účastnit se zápasů.

**Uživatelské účty** Pro využití většiny funkcí bude uživatel aplikace muset být přihlášen svým uživatelským účtem. Účty jsou vždy vázány k jednomu konkrétnímu klubu.

**Uživatelské role** Účtům v systému mohou být přiřazeny role. Každý účet může mít libovolný počet rolí. V systému aktuálně existují následující role:

- Správce klubu je oprávněn spravovat svůj klub, přijímat nové členy, přidělovat jim role a sledovat klubové týmy a turnaje.
- Pořadatel se stará o organizaci turnajů ve svém klubu.

- Trenér ve svém klubu sestavuje týmy, které může nominovat na turnaje. Může si nechat zobrazit seznam zápasů, na kterých bude hrát některý z jeho týmů.
- Rozhodčí jsou v rámci turnajů přiřazení ke hřištím, kde mají dohlížet na probíhající zápasy.

Tato práce se bude zabývat především rolemi *Pořadatel*, *Rozhodčí* a částečně rolí *Trenér* (z hlediska účasti na zápasech a nominaci týmů).

## 2.2 Požadavky

V této sekci práce budou vymezeny požadavky kladené na tuto část aplikace, a tedy čím konkrétně se bude práce v dalších kapitolách zabývat.

### 2.2.1 Funkční požadavky

Funkční požadavky slouží ke stanovení konkrétních funkcí, které aplikace bude schopna vykonávat. V požadavcích se předpokládá využití funkcí již implementovaných v aktuálním systému, které byly popsány v předchozí sekci.

**F1 Správa turnajů**
V aplikaci bude možné v rámci klubů vytvářet turnaje. Ty mohou být vytvořeny členy klubu s rolí *Pořadatel*. Nově vytvořený turnaj bude moci spravovat pořadatel, který jej založil. K turnaji může přiřadit i další klubové pořadatele, kteří tak získají stejnou pravomoc v jeho správě (dále budou pořadatelé spravující klub označování jako *správci turnaje*). Pro každý turnaj bude možné zadat čas zahájení a ukončení, místo konání a věkové kategorie, pro které bude turnaj pořádán. Tyto údaje bude možné později upravovat.

**F1.1 Nominace týmů**
Trenéři budou moci nominovat své týmy pro hru na turnaji. Tým nemůže být nominován, pokud turnaj nebyl vyhlášen pro věkovou kategorii klubu. Každý klub může nominovat více týmů, počet není limitován.

**F1.2 Soukromé a veřejné turnaje**
Turnaje mohou být označeny jako *soukromé* nebo *veřejné*. Veřejné turnaje jsou dostupné všem přihlášeným uživatelům, kteří mohou sledovat jeho zápasy a výsledky. Každý klub na turnaj může nominovat své týmy. Oproti tomu soukromé turnaje jsou viditelné pouze pro pořádající klub a kluby, kterým zaslali správci turnaje pozvánku. Nově vytvořené turnaje jsou automaticky soukromé, s možností zveřejnění.

**F2 Tvorba harmonogramu turnajů**
V jednotlivých turnajích bude možné vytvořit harmonogram, tedy seznam zápasů, které budou během turnaje odehrány. Pro každou věkovou kategorii turnaje bude možné vytvořit svůj samostatný harmonogram. Pro zápasy bude možné zaznamenávat následující informace:

- Oba hrající týmy
- Hřiště turnaje, na kterém se bude zápas hrát
- Čas zahájení
- Délku zápasu
- Rozhodčí zápasu
- Výsledky zápasu

**F2.1 Manuální vytvoření zápasů**
Aplikace umožní správcům turnaje manuálně vytvořit nové zápasy a upravovat existující zápasy. V takovém případě uživatel musí sám vyplnit všechny požadované parametry zápasu. Uživatel může rovněž manuálně upravovat zápasy vytvořené generátorem nebo importované ze souboru.

**F2.2 Generování harmonogramu**
Správci turnaje si mohou nechat harmonogram automaticky vygenerovat. Uživatel může zvolit několik nastavení, podle kterých mu aplikace navrhne možný harmonogram. Uživatel může harmonogram přijmout nebo odmítnout. Možné nastavení pro generátor jsou:

- Délka zápasů
- Formát turnaje
- Časové prodlevy mezi zápasy

**F2.3 Formáty turnaje**
Aplikace bude podporovat různé formáty zápasů, ze kterých si budou moci pořadatelé zvolit při generování harmonogramu. Podporovány budou formáty každý s každým, při kterém každý tým hraje se všemi ostatními týmy, a vyřazovací formát, ve kterém v každém kole vítězný tým postupuje do dalšího kola a poražený tým je vyřazen z turnaje. Bude možné přidání dalších formátů.

**F2.4 Import harmonogramu**
Aplikace bude podporovat vytvoření harmonogramu nahráním souboru z lokálního zařízení. Uživateli budou nahrané zápasy před jejich uložením ukázány, aby si mohl ověřit správnost harmonogramu.

**F2.5 Úpravy harmonogramu**
Vytvořené zápasy je možné dodatečně upravovat, nebo zcela zrušit. Manuálně je možné upravit jednotlivé zápasy. Aplikace bude rovněž podporovat automatickou úpravu harmonogramu, v reakci na náhlé organizační změny. Správce turnaje bude moci nechat odebrat rozhodčího, hřiště nebo tým a aplikace mu nabídne úpravu budoucích zápasů.

**F3 Zasílání notifikací o zahájení zápasu**
Pro zápasy budou moci správci turnaje nechat nastavit notifikaci, která se odešle několik minut před zahájením zápasu jeho účastníkům (rozhodčímu a trenérům obou týmů). Konkrétní čas zaslání bude možné zvolit. Po odeslání budou moci účastníci odpovědět, zdali na začátek zápasu stihnou dorazit. Správci turnaje a samotní účastníci se budou moci podívat na výsledky notifikace (kdo potvrdil/nepotvrdil účast nebo neodpověděl). Před odesláním bude možné upravit čas odeslání notifikace nebo ji zrušit.

**F4 Zapisování zápasů**
Aplikace bude podporovat zaznamenávání výsledků zápasů během jejich průběhu. Zapisování provádí uživatel aplikace pověřený správcem turnaje (dále *zapisovatel*). Během zapisování bude mít zapisovatel přehled o informacích zápasu a za průběhu zápasu bude moci upravovat skóre týmů. Zapisovatel může být přiřazen k jednomu konkrétnímu zápasu, nebo může být přiřazen ke hřišti. V takovém případě bude zaznamenávat všechny zápasy odehrávající se na daném hřišti. Při zapisování hřiště může zapisovatel přenechat zapisování libovolnému dalšímu uživateli.

**F4.1 Zapisovatel bez systémového účtu**
Aplikace bude umožňovat, aby byl k zápisu zápasu přiřazen někdo, kdo v rámci aplikace nemá svůj vlastní uživatelský účet. Jediným požadavkem je nainstalovaná aktuální verze aplikace.

**F4.2 Průběžné odesílání výsledků**
V případě stálého internetového připojení bude aplikace pravidelně odesílat informace o aktuálním skóre zápasu, které bude možné během zapisování sledovat.

**F5 Zaznamenávání výsledků zápasů**
Aplikace bude pro každý zápas zaznamenávat výsledky. V průběhu turnaje si budou moci zobrazit výsledky zápasů jeho účastníci.

**F5.1 Ukončení turnaje a zveřejnění výsledků**
Po odehrání všech zápasů budou moci správci turnaje nechat turnaj ukončit. Po jeho ukončení nebude možné plánovat další zápasy a dojde ke zveřejnění výsledků všech odehraných zápasů.

**F5.2 Export výsledků do souboru**
Výsledky turnaje si bude moci nechat libovolný uživatel, který má k výsledkům přístup, nechat exportovat do souboru na svém zařízení. Systém bude podporovat přidání dalších formátů výsledků.

### 2.2.2 Nefunkční požadavky

Nefunkční požadavky určují obecné požadavky kladené na systém.

**N1 Podpora OS Android**
Aplikace bude podporovat zařízení s operačním systémem Android ve verzi 5.0 a vyšší. Podle statistik dostupných v oficiálním vývojovém prostředí *Android Studio* tento rozsah pokrývá 94.1 % zařízení [3].

**N2 Internetové připojení**
Aplikace bude pro většinu úkonů vyžadovat stálé internetové připojení. Výjimkou je možnost pokračování zápisu zápasu, pokud v jeho průběhu dojde k výpadku připojení. V takovém případě bude zapisování pokračovat a výsledky budou uloženy. Po znovuzískání připojení dojde k synchronizaci výsledků.

**N3 Notifikace uživatelů**
Systém bude podporovat zasílání notifikací uživatelům. Ty budou viditelné ve formě takzvaných *push* notifikací na úrovni OS Android. Na některé notifikace bude moci uživatel odpovědět.

## 2.3 Existující řešení

V této sekci budou prozkoumány a představeny některé existující aplikace, které splňují výše uvedené požadavky. Přestože aplikací zaměřujících se na pořádání turnajů existuje velké množství, nepodařilo se mi najít žádnou, která by zároveň splňovala požadavky implementované v souvisejících pracích (například pořádání tréninků, výzvy pro hráče, správa profilů dětí rodiči...). Pro porovnání tedy budou použity aplikace splňujících požadavky pouze této části.

Pro porovnání byly vybrány tři aplikace splňující následující kritéria:

- Existence mobilní aplikace na službě Google Play
- Dostupnost alespoň části funkcí zdarma
- Možnost vytvoření a sdílení turnajů, možnost přidání týmů k turnaji

- Možnost automatického naplánování zápasů
- Zaznamenávání a pozdější dostupnost výsledků

### 2.3.1 Winner

Aplikace nabízí možnost vytvoření turnajů pro velké množství sportů. K turnaji se poté přidělí hráči nebo týmy, ze kterých se poté vytvářejí zápasy. Rovněž je možné k zápasům přiřadit vytvořená hřiště. Aplikace podporuje několik formátů (každý s každým, rozdělení do skupin, vyřazovací turnaj) a u každého z nich je možné nastavit další upřesňující možnosti (například počet získaných bodů za vítězství/remízu). V závislosti na vybraném formátu je aplikace schopna sama vytvořit harmonogram. Výsledky je k zápasům možné kdykoli přidat a upravit. Po přihlášení je možné vytvořené turnaje sdílet s dalšími uživateli. Aplikace je dostupná zdarma, s možností měsíčního předplatného (*PRO* verze) pro získání dalších funkcí, ovšem i bez něj je aplikace dobře použitelná.

Hlavní nevýhodou aplikace je její celková jednoduchost v porovnání s výše uvedenými požadavky. Aplikace například nepočítá s přidělováním rozhodčích k zápasům, nepočítá se s průběžným zapisováním výsledků, automaticky vytvořený harmonogram neřeší časové kolize.

Mezi zajímavými možnostmi, které aplikace nabízí, je integrace s Google Maps, pomocí které je možné zvolit místo, kde se nachází hřiště. Dále je v *PRO* verzi možné přidělit logo pro jeho snazší identifikaci. Další výhodou je možnost používat aplikaci bez internetového připojení. Aplikace je dostupná zde [4].

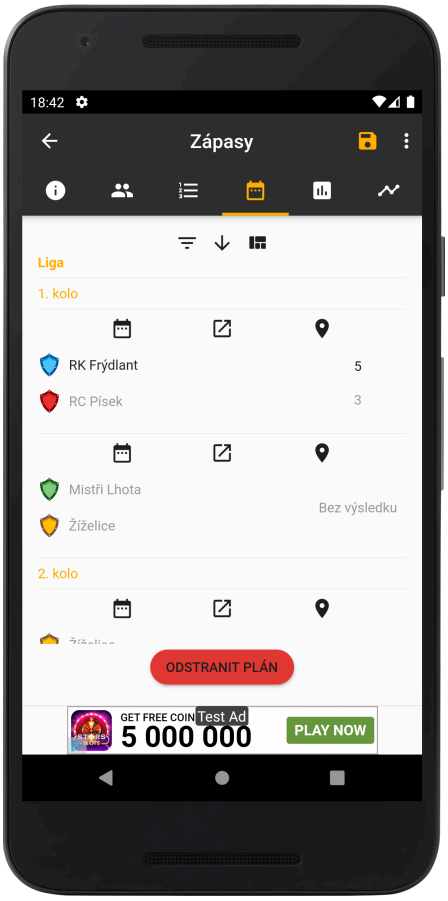
### 2.3.2 Tournament Manager

Podobně jako u předchozí aplikace je možné vytvářet turnaje a sdílet je s dalšími uživateli. Oproti předchozí aplikaci ovšem není možné zvolit sport a účastníci se musí pro každý turnaj vytvořit zvlášť. Opět podporuje automatické vytvoření zápasů, u kterých je možné si vybrat ze tří stejných formátů. K zápasům však není možné přiřadit čas konání ani hřiště. Aplikace, přestože je zcela zdarma, nabízí méně možností než neplacená část aplikace předchozí. Dostupná je zde [5].

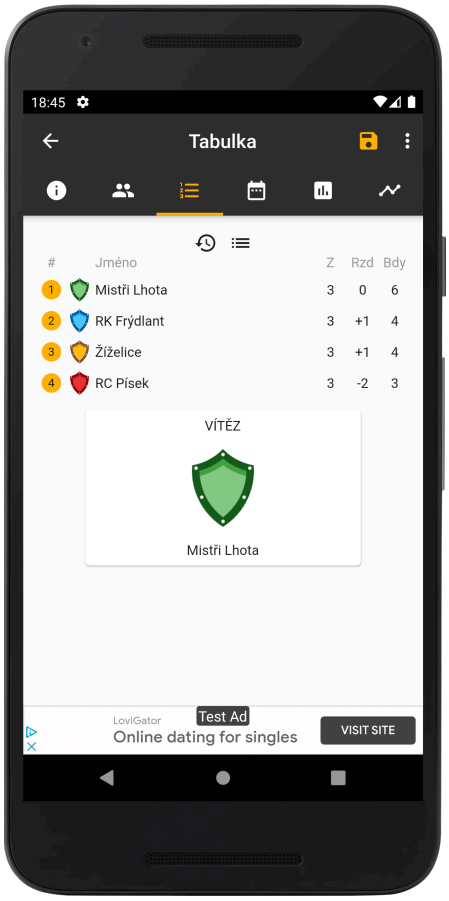
### 2.3.3 Challenge Place

Podobně jako u aplikace Winner je možné vytvářet turnaje z velkého výběru sportů a sdílet je s dalšími uživateli.

Navíc nabízí možnost určení stavu turnaje (zdali je ve fázi plánování, právě probíhá nebo byl ukončen) a nastavení času zahájení a ukončení. Aplikace rovněž počítá s průběžným zaznamenáváním zápasů, a kromě samotných vý-




Obrázek 2.1: Winner – seznam zápasů



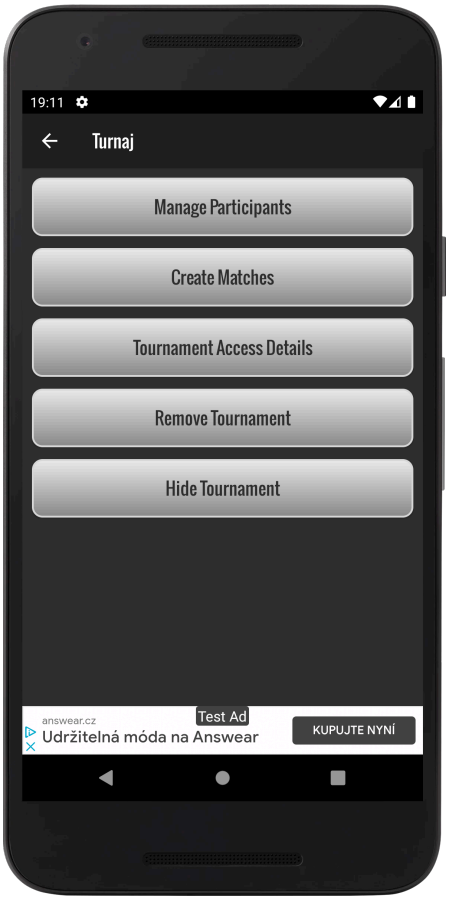

Obrázek 2.2: Winner – tabulka výsledků

sledků je možné na časové ose zaznamenávat další události (například fauly). Dostupná je na odkazu [6].

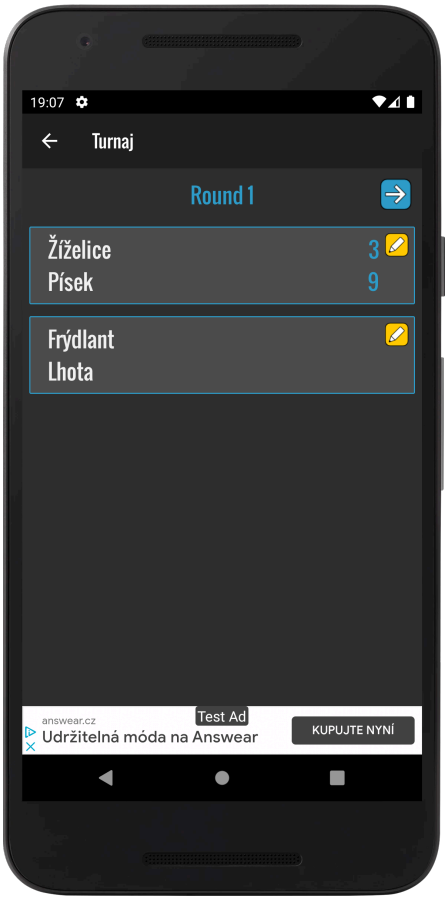
### 2.3.4 Shrnutí

Požadavky na celou aplikaci jsou poměrně rozsáhlé a zároveň specifické pro konkrétní potřeby klubu v rámci jednoho sportu. Implementace vlastního řešení tedy dává smysl. Přestože analyzovaná řešení nemohou být plnohodnotnou náhradou, poznatky z nich mohou posloužit pro budoucí zlepšení aplikace, mezi ně patří například:

- Zobrazení výsledků ve formě tabulek.
- Vizualizace statistik pomocí grafů (například počet získaných bodů).




Obrázek 2.3: Tournament Manager – správa turnaje

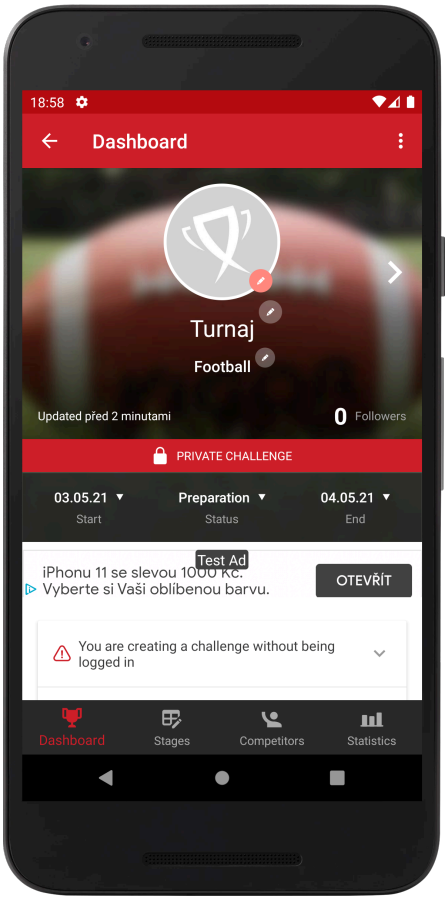



Obrázek 2.4: Tournament Manager – tabulka výsledků

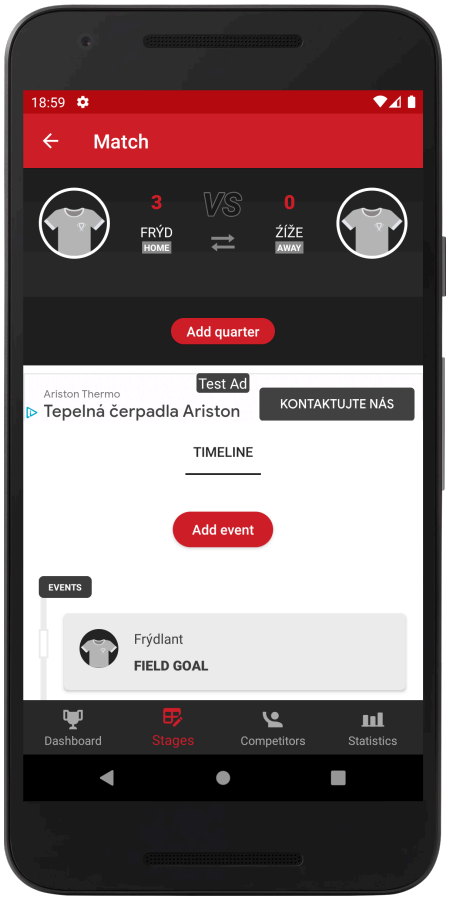
- Pokročilejší možnosti zapisování (zaznamenávání faulů, střídání hráčů, ...), dostupné i po ukončení zapisování na časové ose.
- Rozdělení harmonogramu na několik částí, s možnými různými formáty (například v první části odehrají týmy zápasy stylem každý s každým, následně v druhé části turnaje úspěšně týmy sehrají vyřazovací zápasy).
- Možnost přidělení turnajům/hřištím adresu zvolením místa konání pomocí aplikace s mapou.

## 2.4 Model případů užití

V této sekci budou popsány jednotlivé případy užití, které se budou v systému odehrávat. Jejich cílem je konkretizovat, jaké akce budou v systému moci




Obrázek 2.5: Challenge Place – správa turnaje

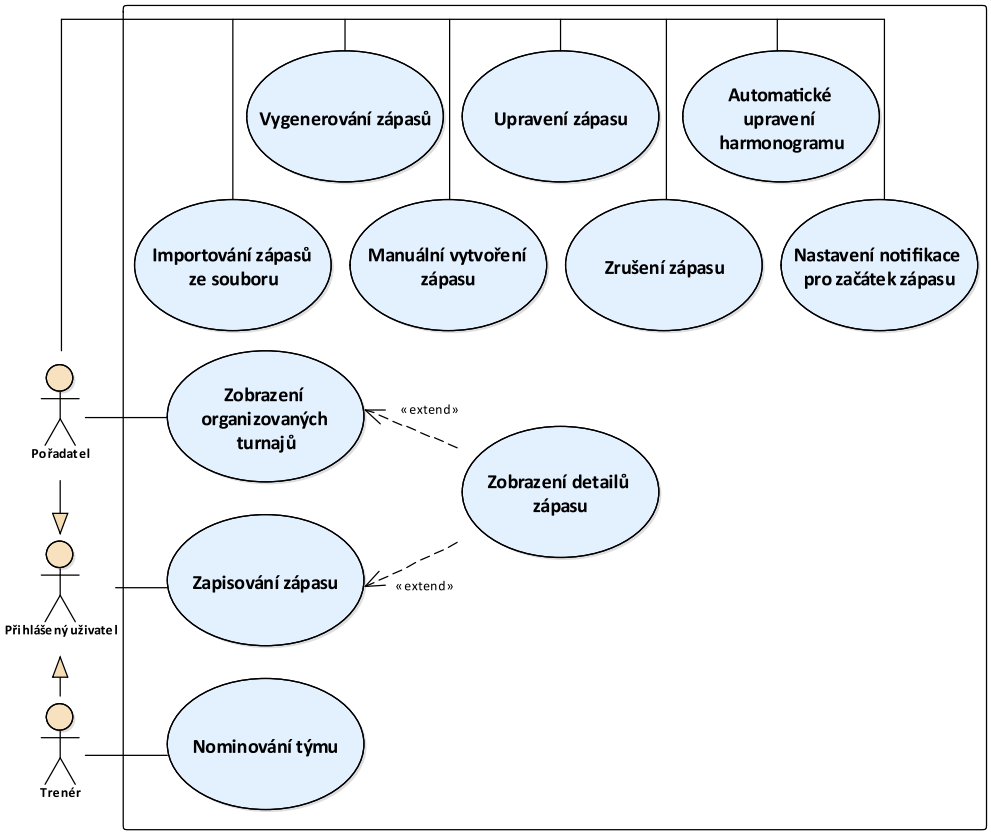



Obrázek 2.6: Challenge Place – zapisování zápasu

provádět uživatelé v závislosti na svých rolích. Případy užití by tak měli zcela pokrývat funkční požadavky kladené na systém, a jak budou naplněny.

**1 Správa turnajů**
Případy užití související s pořádáním turnajů, bez organizace zápasů (2.7).

**UC1.1 Zobrazení veřejných a zúčastněných turnajů**
Přihlášení uživatelé si budou moci volně zobrazit seznam turnajů, které byly pořadateli uveřejněny. Kromě veřejných turnajů je také možné si zobrazit turnaje, které jsou neveřejné, ale byl na ně klub pozván. Odtud si mohou zjistit informace o turnaji, zápasech a výsledcích.




Obrázek 2.7: Případy užití správu turnajů

**UC1.2 Zobrazení organizovaných turnajů**
Pořadatelé si mohou zobrazit turnaje, ke kterým jsou přiřazeny jako správci.

**UC1.3 Vytvoření nového turnaje**
Ze seznamu spravovaných turnajů může pořadatel založit turnaj nový. Při zakládání musí zvolit jméno turnaje, plánovaný čas zahájení a ukončení (je možné pořádat i turnaje trvající více dní) a místo konání. Také při zakládání zvolí věkové kategorie, pro které bude turnaj pořádán a u každé z nich zvolí počet hřišť, na kterých budou rozvrhovány zápasy. Zakládající pořadatel bude přiřazen jako jediný správce turnaje.

**UC1.4 Zobrazení detailů turnaje**
Ze seznamu turnajů si může uživatel zobrazit detaily o libovolném turnaji, o kterém jsou potom zobrazeny kompletní informace. V případě, že je uživatel správcem zvoleného turnaje, jej odtud může dále spravovat.

**UC1.5 Přiřazení pořadatelů k turnaji**
Z detailů turnaje k němu může správce turnaje může k turnaji přiřadit další pořadatele klubu jako správce. Ti pak získají kompletní pravomoci v rámci správy daného turnaje.

**UC1.6 Zveřejnění turnaje**
Nově vytvořený turnaj je automaticky neveřejný, je tak viditelný pouze pro svůj vlastní klub a pro případné pozvané kluby. Z detailu turnaje jej správce může nechat zveřejnit. Po zveřejnění turnaje je volně dohledatelný a kluby na něj mohou nominovat své týmy.

**UC1.7 Zaslání pozvánky na turnaj**
Pořadatel může pro neveřejný turnaj zaslat pozvánku libovolnému dalšímu klubu v aplikaci. Trenéři tohoto klubu poté budou moci na tento neveřejný turnaj nominovat své týmy. Zaslanou pozvánku je možné později stáhnout, klubem nominované týmy budou následně odebrány z turnaje.

**UC1.8 Nominování týmů**
Trenér klubu může na turnaj, po zobrazení jeho detailů, nominovat některý ze svých týmů. Může odtud případně nominaci také zrušit.

**UC1.9 Zobrazení seznamu týmů nominovaných na turnaj**
Pořadatelé turnaje si mohou z jeho detailů zobrazit seznam týmů, které byly na turnaj nominovány.

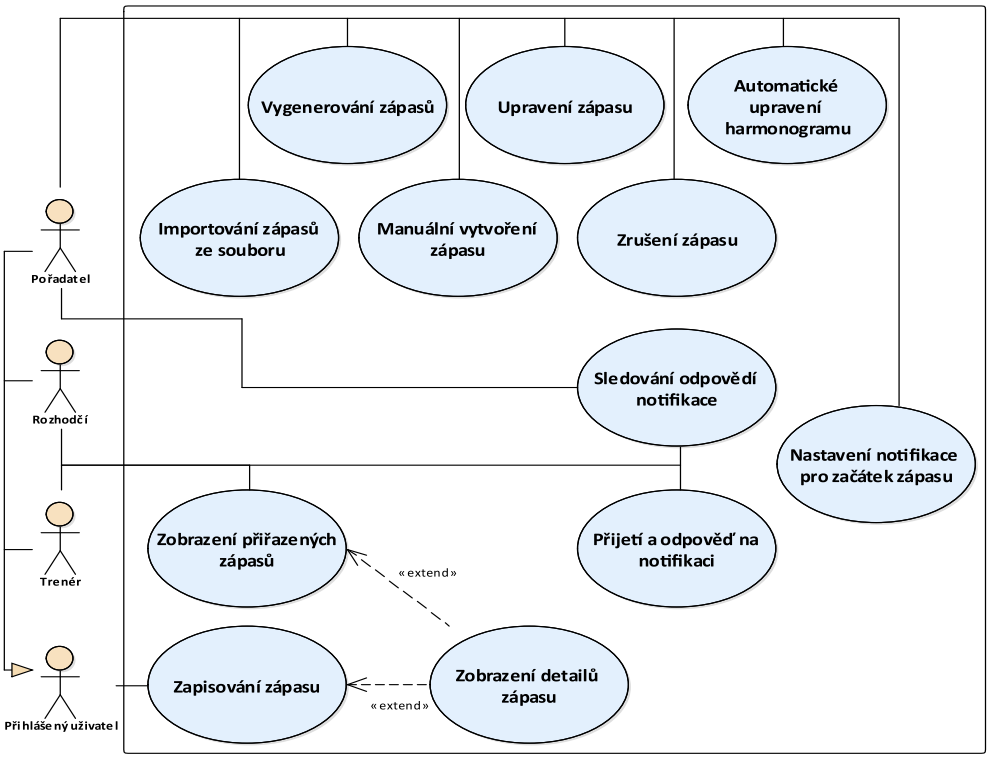
**UC1.10 Zvolení hrajících týmů**
Ze seznamu nominovaných týmů si mohou správci turnaje zvolit, které týmy budou na turnaji skutečně hrát. Skladbu hrajících týmů je možné později změnit.

**UC1.11 Ukončení turnaje**
Pořadatel může kdykoli ukončit turnaj. Ukončení turnaje by mělo proběhnout po odehrání plánovaných zápasů, je ale možné turnaj ukončit i předčasně. Ukončení turnaje je konečné, po ukončení není možné plánovat a zapisovat další zápasy, je pouze možné zobrazit výsledky odehraných zápasů.

**2 Správa zápasů**
Případy užití zaměřené na samotnou tvorbu a organizaci zápasů (2.8).

**UC2.1 Zobrazení zápasů turnaje**
Z detailů turnaje si může uživatel zobrazit seznam naplánovaných zápasů. Kromě seznamu je rovněž možné si nechat zápasy zobrazit ve formě diagramu, kde budou zápasy rozděleny podle kol a se zvýrazněnými postupy týmů napříč zápasy. Zápasy v obou způsobech zobrazení jsou rozděleny podle věkových kategorií.

**UC2.2 Zobrazení přiřazených zápasů**
Trenéři a rozhodčí mají možnost si zobrazit pro ně důležité zápasy.




Obrázek 2.8: Případy užití pro správu zápasů

Pro rozhodčího se jedná o zápasy, nad kterými dohlíží, pro trenéra se jedná o zápasy, na kterých hraje některý z jeho týmů.

**UC2.3 Zobrazení detailů zápasu**
Podobně jako u turnajů je možné si ze seznamu zápasů nechat zobrazit detaily o zvoleném zápase. Zde jsou uživateli zobrazeny plné informace o zápasu a správce turnaje odtud může zápas dále spravovat.

**UC2.4 Manuální vytvoření zápasu**
Pořadatel může vytvořit zápas pro turnaj manuálně. Při vytváření musí zvolit hrající týmy, hřiště, na kterém se bude zápas hrát, kolo, čas začátku zápasu a délku zápasu.

**UC2.5 Importování zápasů ze souboru**
Ze seznamu zápasů může správce turnaje vytvořit harmonogram ze souboru. Zde vybere ze svého zařízení soubor v podporovaném formátu. Pokud soubor není validní, bude na to uživatel upozorněn. Importované zápasy je možné přidat k již existujícím zápasům nebo je možné existující zápasy jimi nahradit.

**UC2.6 Vygenerování zápasů**
Na obrazovce se seznamem zápasů může pořadatel kliknout na tla-

čítko pro vygenerování zápasů. Po kliknutí může zvolit parametry pro generování: délku zápasů, časové prodlevy mezi zápasy, delší časové pauzy během turnaje a formát turnaje. Vygenerované zápasy je možné kombinovat s importovanými a manuálně vytvořenými zápasy.

**UC2.7 Upravení zápasu**
Pořadatel může vytvořený zápas upravit. Je možné změnit hřiště, čas zahájení nebo jeho délku. Změna týmů u existujícího zápasu není možná, respektive už by se nejednalo o stejný zápas.

**UC2.8 Zrušení zápasu**
Naplánovaný zápas může pořadatel zcela zrušit. Není však možné zrušit zápas, kterému byly, nebo právě jsou, zapisovány výsledky.

**UC2.9 Automatické upravení harmonogramu**
Pořadatel si může ze seznamu zápasů nechat aplikací automaticky upravit plán následujících zápasů. Po zvolení některého z možných problémů (výpadek týmu, rozhodčího nebo hřiště) a strategii řešení (vynechání zápasů, výměna rozhodčích nebo hřišť...) je uživateli nabídnuta možná úprava harmonogramu.

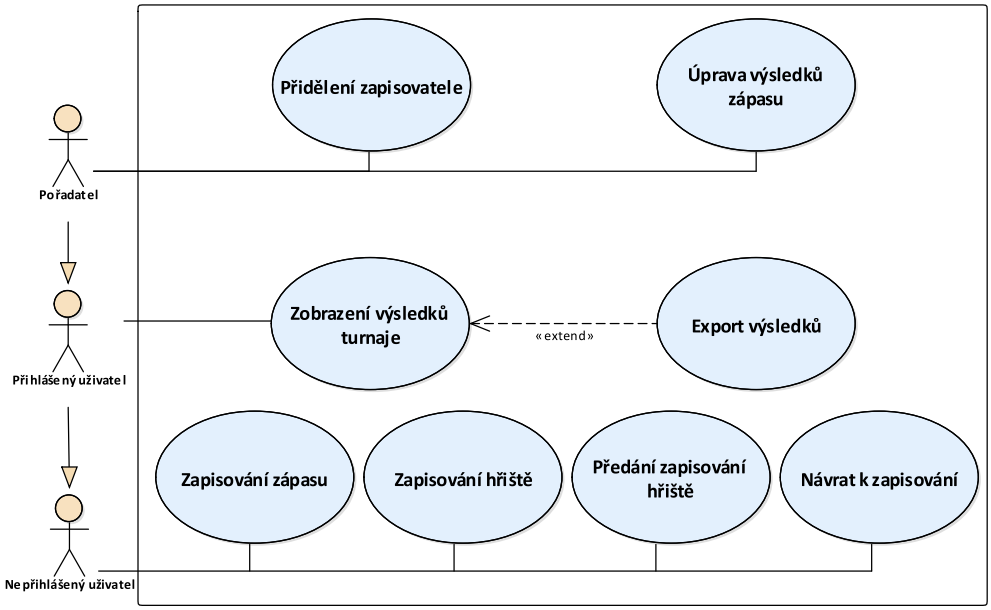
**UC2.10 Nastavení notifikace pro začátek zápasu**
Pro existující zápas může pořadatel nastavit notifikaci pro začátek zápasu. Pořadatel si zvolí čas, kdy systém odešle rozhodčímu zápasu a trenérům hrajících týmů v aplikaci notifikaci upozorňující na začátek zápasu. Notifikaci je před odesláním možné zrušit, nebo upravit její časovou prodlevu. Pokud již zápas měl začít, může pořadatel notifikaci odeslat okamžitě.

**UC2.11 Přijetí a odpověď na notifikaci**
Odeslaná notifikace bude uživateli zobrazena, i když aplikace není spuštěná. Pomocí notifikace budou uživateli zobrazeny základní informace o zápasu a bude vyzván k odpovědi. Uživatel tak může potvrdit nebo zamítnout, zdali stihne dorazit na zahájení zápasu.

**UC2.12 Sledování odpovědí notifikace**
U zápasu s odeslanou notifikací si v jeho detailech mohou správce turnaje i účastníci zobrazit aktuální stav odpovědí (kdo jak odpověděl, případně že neodpověděl vůbec).

**3 Zapisování výsledků zápasů**
Případy užití soustředěné na zaznamenávání a zpracování výsledků (2.9).

**UC3.1 Přidělení zapisovatele**
Pořadatel bude moci přidělit libovolného uživatele aplikace, bez nutnosti mít uživatelský účet, k zaznamenávání výsledků zápasu, nebo všech zápasů na hřišti.




Obrázek 2.9: Případy užití pro zaznamenávání výsledků

**UC3.2 Zapisování zápasu**
Při zapisování zápasu má uživatel dostupnou časomíru zobrazující plánovanou délku zápasu, kterou může spustit, případně pozastavit. Uživatel může přidat skóre týmu při bodování, případně libovolně upravit aktuální výsledek, pokud se dopustí chyby. Zapisování je možno ukončit kdykoli – je možné skončit předčasně, nebo přetáhnout plánovanou délku (funkce časomíry je tedy především orientační). Po ukončení zapisování již zapisovatel nemůže výsledek zápasu dále měnit.

**UC3.3 Zapisování hřiště**
Uživatel může být přiřazen k zapisování některého z hřišť turnaje. V takovém případě mu aplikace zobrazí plánovaný harmonogram zápasů pro dané hřiště, a s určitým časovým předstihem mu bude dovoleno zapisovat jednotlivé zápasy.

**UC3.4 Předání zapisování hřiště**
Při zapisování hřiště má zapisovatel možnost předat zapisování daného hřiště dalšímu uživateli. Ze seznamu zápasů turnaje si může zapisovatel nechat zobrazit kód pro zapisování, který může předat dalšímu uživateli.

**UC3.5 Návrat k zapisování**
Pokud zapisovatel opustí samotné zapisování, nebo i zcela ukončí aplikaci, bude mu nabídnuto se k zapisování vrátit.

**UC3.6 Zobrazení výsledků turnaje**
V průběhu turnaje je možné pomocí seznamu zápasů sledovat jejich výsledky. U každého zápasu je uvedeno, zda je právě zapisován, nebo už byl zapsán a je uvedeno zaznamenané skóre. Po ukončení turnaje je dostupný souhrn všech zapsaných zápasů s jejich výsledky. Výsledky jsou dostupné všem, kdo mají přístup k zápasům turnaje.

**UC3.7 Úprava výsledků zápasu**
Pořadatel turnaje má možnost u odehraných a zapsaných zápasů dodatečně upravit výsledky.

**UC3.8 Export výsledků**
Uživatelé, kteří mají přístup k celkovým výsledkům turnaje, si mohou nechat vyexportovat výsledky do souboru, který bude uložen v zařízení.

Kapitola 3

# Návrh

V následující kapitole bude představena aktuální architektura systému a plánované přepracování. Dále bude popsán doménový model pro přiblížení dat, se kterými se bude v systému pracovat. Část kapitoly je rovněž věnována navrhovanému způsobu plánovaní zápasů pro turnaje. Hlavní částí kapitoly pak je návrh uživatelského prostředí.

## 3.1 Architektura systému

Aktuální řešení, na které se bude navazovat, je realizováno jako třívrstvá architektura. Rozdělení na několik vrstev s jasně definovaným rozhraním pro komunikaci napříč vrstvami zjednodušuje budoucí správu systému a případnou budoucí náhradu stávajících vrstev.

### 3.1.1 Prezentační vrstva

Prezentační vrstva je nejvyšší vrstva systému, kterou používá samotný koncový uživatel. Získává data z aplikační vrstvy a zobrazuje je uživateli, zpracovává vstup od uživatele a zpětně zasílá požadavky aplikační vrstvě pro vykonání požadavků uživatele. V implementovaném systému se jedná o samotnou Android aplikaci.

### 3.1.2 Aplikační vrstva

Aplikační vrstva slouží jakožto prostředník mezi prezentační a datovou vrstvou. Přijímá a odpovídá na požadavky od prezentační vrstvy, kontroluje jejich validitu a komunikuje s datovou vrstvou pro získávání a ukládání dat.

V aktuálním systému se jedná o samostatný aplikační server, se kterým komunikují všechny Android klienti pomocí *HTTP* protokolu prostřednictvím REST rozhraní. Jedná se o jednorázové požadavky zahajované výhradně prezentační vrstvou. V rámci této práce bude serverová část rozšířena pro pod-

poru komunikace iniciovanou samotným serverem, hlavním důvodem je lepší podpora zasílání notifikací a možnost propagovat změny v persistentních datech okamžitě k uživateli. Server tak bude rozšířen o podporu protokolu *WebSocket*, umožňující oboustrannou komunikaci. Implementační detaily budou zmíněny v další kapitole.

### 3.1.3 Datová vrstva

Nejnižší datová vrstva slouží k trvalému uložení dat, slouží tak jako zdroj dat pro všechny koncové klienty. Komunikuje s datovou vrstvou pro přístup k uloženým datům a uložením dat nových. Tato vrstva je v systému reprezentována relační databází.

## 3.2 Architektura Android aplikace

V rámci této práce došlo k částečnému předělání Android aplikace podle vzoru *Model-View-ViewModel* (MVVM). Podle něj je aplikace rozdělená na tři části:

View představuje nejvyšší vrstvu aplikace, zajišťuje zobrazování obsahu a zpracovává vstup od uživatele.

Model reprezentuje samotný zdroj dat, v případě aplikace se jedná o část zajišťující komunikaci se serverem.

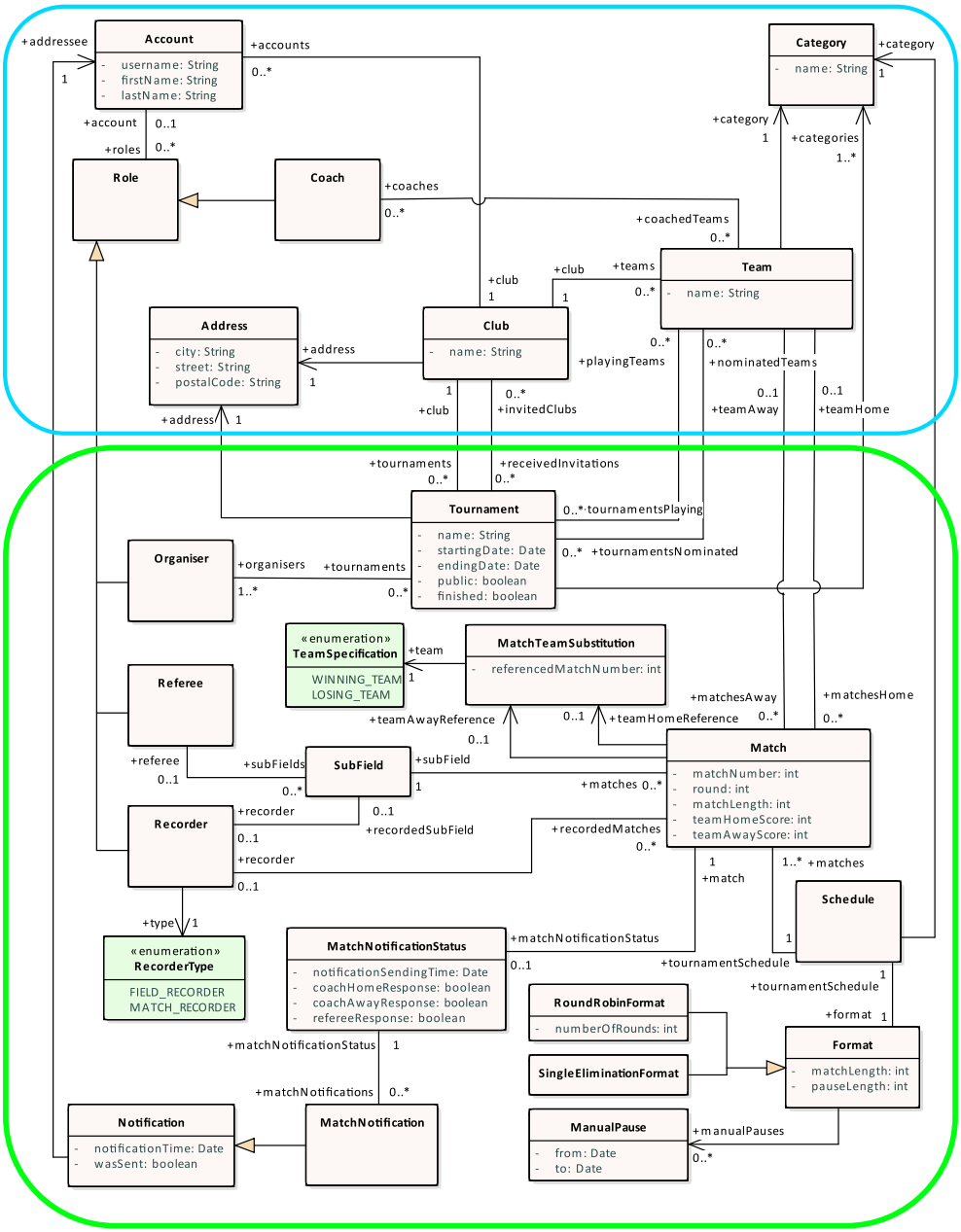
ViewModel je prostředník mezi *View* a *Modelem*. View požadovaná data získává od *ViewModelu*, který je zase získává od *Modelu*.

Hlavním cílem tohoto návrhu je oddělení logiky zajišťující zpracování dat (*ViewModel*) od logiky zajišťující uživatelské rozhraní (*View*). Tento vzor vychází ze vzoru *Model-View-Presenter* (MVP). Základní přístup je stejný, hlavní rozdí je ten, že na rozdíl od MVP komunikují pouze vyšší vrstvy s vrstvami nižšími, nikoli naopak. View tak může držet referenci k ViewModelu a může od něj vyžádat zdroj dat, ale samotný ViewModel nevlastní reference na žádný View a s žádným přímo nekomunikuje. Tento návrh může být realizován pomocí návrhového vzoru *Observer*, kdy View může sledovat data, která poskytuje ViewModel, a může automaticky upravit uživatelské rozhraní při změně dat.

Hlavním důvodem výběru tohoto vzoru je jeho dobrá podpora v Android ekosystému. Oficiálně je také doporučován pro návrh aplikací [7].

## 3.3 Doménový model

V této sekci bude se pomocí doménového modelu (na obrázku 3.1) představí třídy, se kterými se bude v systému pracovat, a vazby mezi nimi. V diagramu




Obrázek 3.1: Doménový model

jsou zobrazeny pouze části relevantní pro tuto práci, bez rozšíření z dalších souvisejících prací. Zeleně jsou označeny třídy používané pouze v této části, modře jsou pak označeny třídy, které jsou napříč pracemi sdíleny, a návrh kterých probíhal společně.

**Account** Reprezentace uživatelského účtu v systému. Pro každý účet se zaznamenává unikátní *username* (přihlašovací jméno) a hash hesla, které se dohromady používají pro ověření uživatele při přihlášení. Dále je u každého uživatele zaznamenáno jméno a příjmení pro jednodušší identifikaci ostatními uživateli v aplikaci. Každý účet je svázán se svým (jediným) klubem. Každý účet má seznam přiřazených rolí.

**Role** Abstraktní třída, ze které dědí konkrétní role v systému. U většiny rolí obsahuj referenci k účtu, ke kterému je role přiřazena.

**Referee** Třída reprezentující roli rozhodčího, který dohlíží na průběh zápasů. V systému jsou rozhodčí přiřazováni ke hřištím turnajů (ze stejného klubu jako klub vlastněného účtu), rozhodčí poté dohlížejí na všechny zápasy konající se na hřišti.

**Coach** Reprezentace role trenéra v systému. Mimo jiné jsou pro něj zaznamenávány trénované týmy, které může nominovat na turnaje.

**Organiser** Třída pro reprezentaci roli pořadatele, spravující turnaje v rámci svého klubu. Je zaznamenáván seznam turnajů, které pořadatel spravuje.

**Recorder** Role pro uživatele zaznamenávajícího výsledky zápasů. Protože zapisovat výsledky může i nepřihlášený uživatel, není tato podtřída vázána na uživatelský účet. Při každém zadání zapisovacího kódu je pro zapisování vytvořena nová instance. Každému zapisovateli je přidělen typ, podle toho, zdali byl přidělen k jednomu konkrétnímu zápasu nebo ke hřišti. V případě, že byl přidělen ke hřišti, instance povinně obsahuje referenci ke přiřazenému hřišti. Dále je zaznamenáván seznam zapsaných hřišť. V případě přiřazení k jednomu hřišti tedy seznam obsahuje jeden zápas, při přiřazení ke hřišti obsahuje všechny zápasy, které ze hřiště byly zapsány.

**Category** Třída reprezentující věkové kategorie. Pro kategorii je uloženo pouze unikátní „jméno“ podle kterého jsou jednotlivé kategorie rozlišeny.

**Team** Reprezentace týmů, které jsou nominovány na turnaje a přiřazovány k zápasům. Pro identifikaci je ke každému týmu přiřazeno jméno. U týmů je zaznamenán seznam hráčů, kteří za klub hrají. Každý tým má přiřazenou jednu věkovou kategorii, kterou musí mít i všichni hráči, kteří budou přiřazeny k týmu. Tým musí být přiřazen právě k jednomu klubu, ke kterému musí patřit i všichni jeho hráči. Každý tým má seznam trenérů, kteří jej spravují. Tým musí mít alespoň jednoho trenéra. Pro tým jsou dále zaznamenávány zápasy, na kterých hraje/hrál, které jsou rozděleny podle toho, zdali je tým přiřazen jako domácí (*matchesHome*) nebo hostí (*matchesAway*). Dále je zaznamenáván seznam turnajů, na

které je tým nominován (*tournamentsNominated*) a seznam turnajů, na kterých tým hraje (*tournamentsPlaying*).

**SubField** Turnajové hřiště, na které se budou rozvrhovat zápasy. Nereprezentují však skutečné, reálné hřiště. Ta jsou povětšinou rozdělena na vícero menších hřišť, pro každý turnaj může být rozděleno jinak. Hřiště jsou pro každý turnaj individuálně vytvářena jeho správci. Každé hřiště je tedy přiřazeno právě jednomu turnaji. Má přiřazenou právě jednu kategorii, která rozlišuje, které týmy na něm budou hrát. Dále je zaznamenán seznam naplánovaných a odehraných zápasů.

**Schedule** Reprezentace harmonogramu pro jednu věkovou kategorii v rámci turnaje. Obsahuje seznam zápasů, které jsou pro příslušnou kategorii naplánovány, a povinně obsahuje formát, který určuje, podle jakých kritérií a pravidel byly zápasy vytvořeny. Navíc ještě zaznamenáváme odebraná hřiště a odebrané týmy. Ty jsou harmonogramu přiděleny v okamžiku, kdy je hřiště nebo tým odebrán z turnaje, ale v harmonogramu existuje už odehraný zápas, ve kterém tým nebo zápas figuroval.

**Format** Abstraktní třída pro reprezentaci pravidel při automatickém vytvoření zápasů. Obsahuje některé obecné požadavky, které by měli být respektovány dalšími dědícími formáty, blíže popsány v sekci 3.4.

**RoundRobinFormat** Třída pro reprezentaci formátu „každý s každým“. Kromě zděděných parametrů pro formát je zaznamenáván počet kol.

**SingleEliminationFormat** Reprezentace vyřazovacího formátu.

**Match** Třída reprezentující zápas. Existuje v rámci jednoho konkrétního harmonogramu. Zápas má přiřazený čas zahájení a předpokládanou délku v minutách, časově zápas ovšem nesmí přesahovat časový rozsah turnaje. Dále má zápas povinně přiřazeno číslo zápasu, které je unikátní v rámci svého harmonogramu, a číslo kola, ve kterém se bude hrát. Rovněž je pro zápas zaznamenáván tým domácích (*teamHome*) a tým hostí (*teamAway*), nebo pokud některý z týmů závisí na výsledcích jiného zápasu, a není zatím známý, je relevantní tým reprezentován instancí MatchTeamSubstitution (*teamAwayReference* a *teamHomeReference*). Pro oba týmy je také uloženo skóre. Pomocí *finished* je rozlišováno, zdali byly úspěšně zapsány výsledky zápasu.

**MatchTeamSubstitution** Třída pro reprezentaci zatím neznámých týmů, které budou zápasu přiřazeny na základě výsledků jiného zápasu. Záznam *referencedMatchNumber* označuje číslo zápasu, ze kterého bude tým pocházet, *team* rozlišuje, zdali tým bude vítězný, nebo poražený tým zápasu.

**Tournament** Pro turnaj se sleduje adresa jeho konání, seznam věkových kategorií, pro které je turnaj pořádán, čas zahájení a ukončení. Turnaj musí být přiřazen k jednomu konkrétnímu klubu. Má přiřazen seznam týmů, které byly na turnaj nominovány a seznam týmů, které byly přiřazeny jako hrající. Pro každý turnaj je zaznamenáván seznam harmonogramů (vazba *schedules*). Ty reprezentují pořádané zápasy pro každou ze zvolených věkových kategorií. Pro každou věkovou kategorii tak turnaj obsahuje buď jeden příslušný harmonogram nebo žádný.

Pomocí *public* se rozlišuje, zdali je turnaj veřejný, nebo soukromý. V případě soukromého turnaje je zaznamenáván seznam klubů, které byly na turnaj pozvány. Správce turnaje jej může ukončit, po čemž není možné turnaj dále upravovat a jsou pouze dostupné výsledky odehraných zápasů. Ukončení je rozlišeno příznakem *finished*.

**Notification** Abstraktní třída pro reprezentaci notifikací, které jsou zasílané uživatelům. U každé notifikace je zaznamenáván účet, kterému má být notifikace zaslána, čas, kdy má být zaslána, a označení, zda již byla odeslána.

**MatchNotification** Notifikace zasílaná uživateli pro upozornění o zahájení zápasu. Povinně má vazbu s *MatchNotificationStatus*, přes kterou rovněž získává informace o přiřazeném zápasu.

**MatchNotificationStatus** Třída agregující informace o nastavené notifikaci o zahájení zápasu. Povinně obsahuje referenci k přiřazenému zápasu, čas, kdy mají být samotné notifikace, odeslány a seznam přiřazených notifikací o zápasu. Dále obsahuje odpovědi na notifikace od rozhodčího a trenérů na zaslané notifikace.

**Address** Třída pro reprezentaci adresy turnaje a klubu.

## 3.4 Plánování zápasů

Hlavním způsobem vytváření harmonogramu bude automatické vytváření v rámci aplikace. Další možnosti (manuální vytvoření, definice ze souboru) nebudou zatím v prototypu implementovány, především z důvodu složitějšího zajištění validity zápasů.

Harmonogramy bude možné vytvářet zvlášť pro každou věkovou kategorii. Při vytváření jsou podle formátu rozplánovány zápasy pro všechny přiřazené hrající týmy dané kategorie, s použitím přiřazených hřišť. V systému se nepočítá se zaznamenáváním výkonnosti hráčů nebo týmů. Týmy tak proti sobě budou přiřazeny především náhodně.

Samotné vytváření zápasů může pořadatel ovlivnit nastavením několika parametrů:

- Čas zahájení, od kterého budou zápasy plánovány. V základu jsou zápasy vytvářeny v čas zahájení turnaje, je však možné zahájit hru později.
- Délku zápasů, která bude stejná pro všechny vytvořené zápasy.
- Časovou pauzu, která slouží jako minimální rozsah mezi dvěma zápasy. Týmy tak budou mít jistotu přestávky před dalším zápasem.
- Delší pauzy, manuálně zvolené časovými rozsahy. Je tak možné specifikovat, kdy se nemají zápasy hrát (například noční mezera pro turnaj odehrávající se během více dní.). V prototypu tato funkce zatím není implementována.

### 3.4.1 Formáty

Dále musí pořadatel zvolit jeden z dostupných formátů, podle kterého budou zápasy vytvářeny. Každý z formátů může mít vlastní parametry pro nastavení. V plánu je implementovat následující dva jednoduché formáty, s možností přidání dalších formátů v budoucnu.

**Každý s každým (Round robin)** Formát, ve kterém každý tým postupně sehraje jeden zápas s každým z ostatních týmů, případně je možné počet zápasů zvolit.

Pro vytváření zápasů je použit algoritmus „Circle method“ [8]. Pro sudý počet týmů jsou v prvním kole týmy rozpůleny do dvou řádků, v prvním řádku jsou týmy seřazeny zleva doprava, v druhém zprava doleva. V kole tak proti sobě hrají týmy na stejných pozicích v obou řádcích. Pro další kola jsou všechny týmy, kromě prvního týmu v prvním řádku, posunuty o jednu pozici ve směru hodinových ručiček. V prvním řádku se tedy týmy posunují o jednu pozici vpravo, ve druhém řádku se posouvají o jednu pozici vlevo. Poslední tým prvního řádku se stane posledním týmem řádku v druhém řádku, a první tým druhého řádku se dostane na druhou pozici prvního řádku. Zápasy jsou opět vytvořeny stejným způsobem. Tento proces se opakuje, dokud se každý tým nesetká se všemi ostatními týmy.

K lichému počtu týmů je možné přidat jeden dodatečný tým, a vytvořit zápasy stejně jako pro sudý počet. Tým, který by měl být přiřazen k dodatečnému týmu, v daném kole nebude hrát žádný zápas.

**Vyřazovací formát** Poražený tým v zápasu je okamžitě vyřazen z turnaje, zatímco výherce postupuje do dalšího kola, až do finále mezi posledním párem. V ideálním případě by tak měl turnaj mít přihlášený počet týmů odpovídající mocnině čísla 2, v opačném případě by některé týmy hráli menší počet zápasů než ostatní.

### 3.4.2 Úpravy harmonogramu

Již vytvořené harmonogramy bude možné dodatečně v průběhu turnaj upravit, bez nutnosti vytvářet harmonogramy nové. Změny se tak dotknou pouze budoucích zápasů, již odehrané zápasy zůstanou neupraveny. Pořadatel bude mít možnost upravit některé z parametrů (změny délky zápasů, přidání pauz) a odebrat tým nebo hřiště z turnaje. V případě odebrání týmu tak budou zápasy s tímto týmem zrušeny a ostatní zápasy budou časově přesunuty. V případě odebrání hřiště je toto hřiště v zápasech buď nahrazeno hřištěm jiným, bez dalších změn, nebo, pokud neexistuje volné hřiště, jsou ostatní zápasy přeplánovány pro uvolnění času na některém z obsazených hřišť.

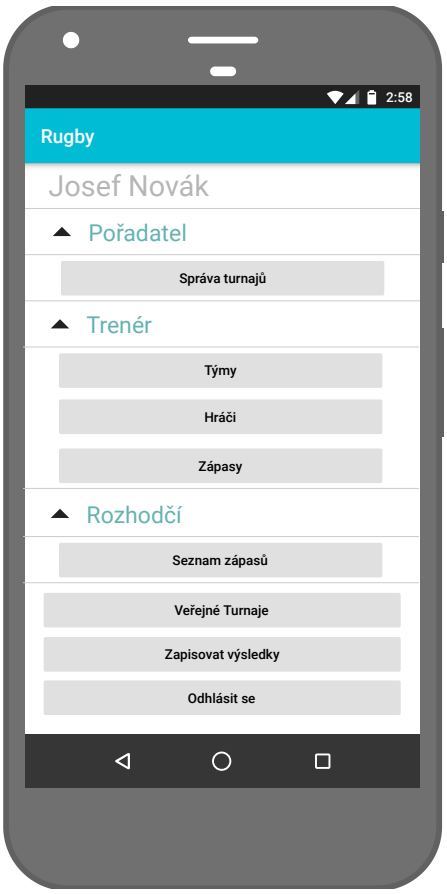
Ve výsledném prototypu jsou možnosti více omezené, s možností pouze odebrat hřiště nebo tým, bez možnosti úpravy nastavení.

## 3.5 Návrh uživatelského rozhraní

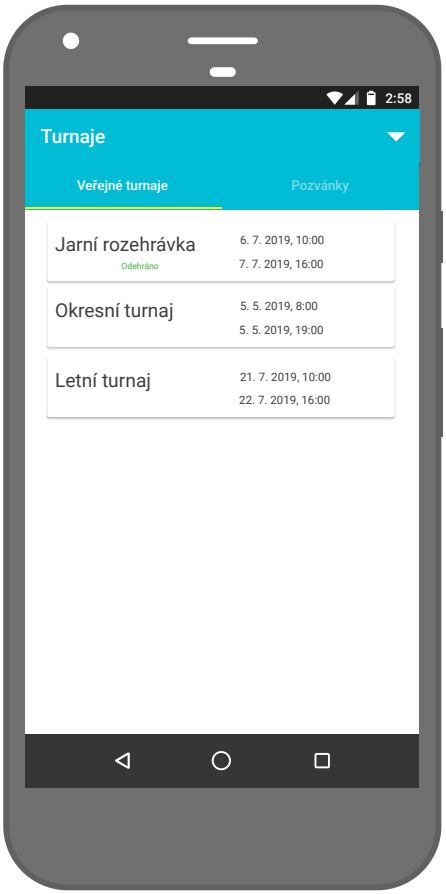
V této sekci budou představeny a popsány návrhy hlavních obrazovek aplikace. Cílem je konkrétně popsat, jak budou v implementované aplikaci naplněny jednotlivé modely užití, jak bude uživatel aplikaci používat a jak bude navigován napříč obrazovkami.

**Hlavní obrazovka**
: První obrazovka zobrazená po spuštění aplikace, navržená dohromady společně s kolegy (3.2). Nepřihlášený uživatel odsud může zapisovat zápasy a přihlašovat se. Po přihlášení se uživateli zobrazí seznam s jeho rolemi. Každou se zobrazených rolí může dále rozkliknout pro zobrazení možností, ke kterým ho daná role opravňuje. Rovněž si každý přihlášený uživatel může zobrazit seznam veřejných turnajů.

**Seznam turnajů**
: Všichni přihlášení uživatelé si mohou z hlavní obrazovky pomocí tlačítka *Veřejné turnaje* zobrazit seznam turnajů 3.3. Zde jsou uživateli zobrazeny v první skupině turnaje volně uveřejněné pořadateli a ve skupině druhé turnaje, na které byl pozván uživatelův klub. Turnaje jsou zobrazeny v projížděcím seznamu, počet turnajů tedy není limitován velikostí displeje. Turnaje v seznamu jsou seřazeny podle času zahájení, první jsou zobrazené stále probíhající turnaje, až poté jsou zobrazeny již ukončené turnaje. U každého turnaje jsou zobrazeny základní informace (název turnaje, čas zahájení a konce, zdali byl turnaj ukončen). Po kliknutí na turnaj jsou uživateli zobrazeny další informace. Turnaje se budou načítat po částech (stránkovat), uživatel si na konci seznamu bude moci vyžádat načtení dalších turnajů. Uživatel rovněž bude moci turnaje filtrovat pomocí tlačítka v horním panelu.




Obrázek 3.2: Hlavní obrazovka




Obrázek 3.3: Seznam veřejných turnajů

Pro pořadatele je rovněž dostupný seznam pořádaných turnajů, který je z hlavní obrazovky dostupný ze záložky *Pořadatel* 3.4. V seznamu jsou zobrazeny pouze turnaje, které uživatel aktuálně spravuje. Kromě toho zde může vytvořit nový turnaj, jinak je funkcionalita stejná jako u předchozího seznamu.

**Vytvoření nového turnaje**

Pořadatel si ze seznamu spravovaných turnajů může zobrazit obrazovku pro vytvoření nového turnaje 3.5. Pro jeho vytvoření musí povinně vyplnit název turnaje a po kliknutí na příslušná tlačítka zvolit adresu konání, čas zahájení a čas ukončení turnaje. Uživatel má také možnost zvolit, prostřednictvím zaškrtávacích políček, pro jaké věkové kategorie bude turnaj pořádán. Vybrat může libovolný počet kategorií, minimálně však jednu. Pomocí tlačítka v dolní části obrazovky může uživatel turnaj vytvořit, nebo se může vrátit zpět bez jakýchkoli změn. Pokud se uživatel

pokusí vytvořit turnaj s nevyplněnými nebo neplatnými údaji, aplikace při pokusu o vytvoření nevalidní pole barevně zvýrazní s popisem problému. Turnaj je po vytvoření soukromý, s vytvářejícím pořadatelem jako jediným správcem. Tomu je dostupný ze seznamu pořádaných turnajů, odkud jej může dále upravovat.

Obrazovka se používá i u budoucích úprav parametrů u existujících turnajů. V takovém případě jsou všechny pole předvyplněna existujícími údaji a tlačítko pro vytvoření, s pozměněným textem, místo vytvoření nového turnaje upraví turnaj existující. V případě upravení turnaje bude aplikace kontrolovat i další možné nevalidní změny, například zrušení kategorie, pro kterou jsou rozvrženy zápasy.

**Detaily turnaje**

Po kliknutí na turnaj ze seznamu se uživateli zobrazí okno s jeho detaily (3.6). Zde jsou uživateli zobrazeny kompletní informace o turnaji, který v závislosti na svých rolích může nad turnajem vykonávat další operace.

Všichni uživatelé si mohou zobrazit seznam zápasů turnaje. Případný správce turnaje odtud může zápasy vytvářet a upravovat.

Uživateli s rolí trenéra je dostupné tlačítko *Nominovat týmy*, pomocí kterého si může vybrat ze svých týmů pro hru na turnaji.

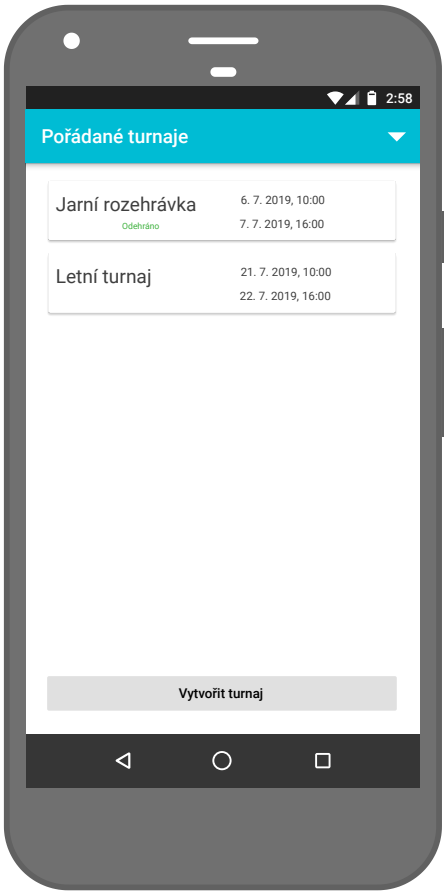
Správci turnaje jsou zobrazeny tlačítka pro jeho detailní úpravu. V případě, že je turnaj soukromý, je k dispozici dvojice tlačítek pro pozvání klubů na turnaj a ke zveřejnění turnaje. Zveřejnění turnaje je nevratné, proto zveřejnění musí uživatel dodatečně potvrdit. Dále má možnost spravovat hřiště turnaje, přiřadit k turnaji další správce, nechat si zobrazit týmy přihlášené na turnaj a upravit základní informace, zadané při vytváření turnaje. V případě odehrání všech zápasů má správce možnost turnaj uzavřít. Po uzavření není možné nad turnajem provádět žádné další operace, k dispozici jsou pouze informace a seznam odehraných zápasů s výsledky. Zatím neuzavřený turnaj může správce (po potvrzení) zrušit.

**Přiřazení správců**

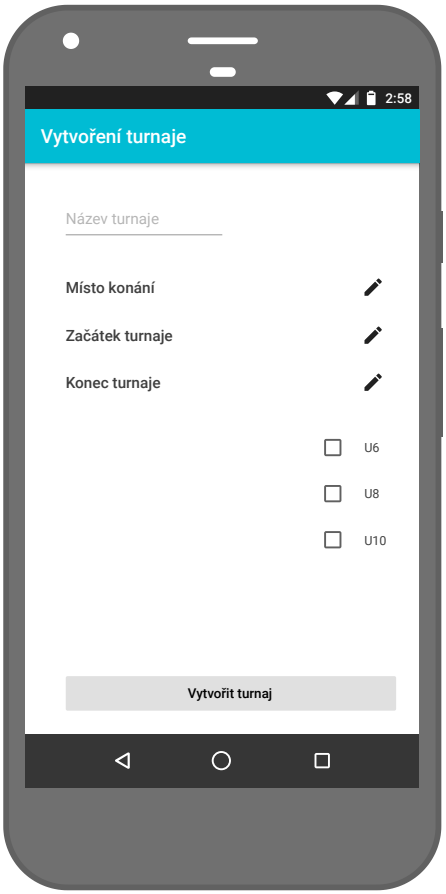
Na této obrazovce (3.7) může správce turnaje přiřadit (nebo odebrat) další klubové pořadatele k turnaji (sám sebe odebrat nemůže). Volbu může potvrdit pomocí tlačítka v dolní části.

**Pozvání klubů**

Z této obrazovky (3.9) může pořadatel pozvat kluby na jím spravovaný soukromý turnaj. Pod první záložkou je uživateli zobrazen projíždějící seznam všech zatím nepozvaných klubů, které může k turnaji pozvat. Členové pozvaného klubu poté mají k turnaji přístup ze seznamu turnajů (3.3) pod záložkou *Pozvánky*. Po kliknutí na druhou záložku jsou




Obrázek 3.4: Seznam pořádaných turnajů




Obrázek 3.5: Obrazovka vytvoření turnaje

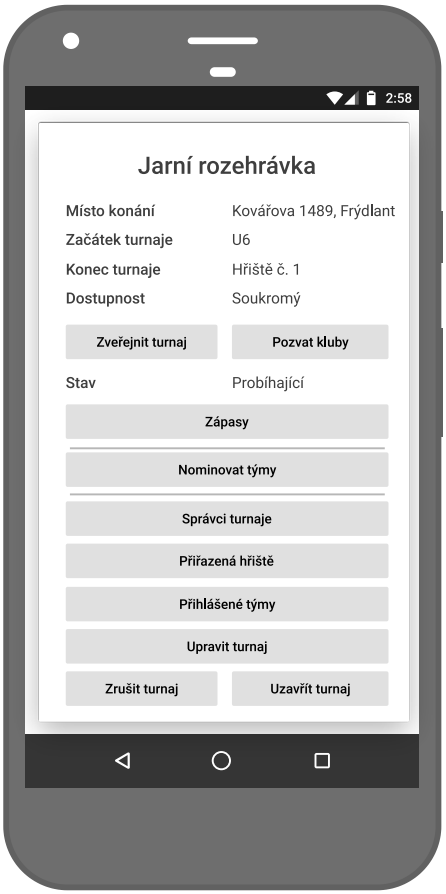
zobrazeny pozvané kluby, kterým naopak může pořadatel pozvánku zrušit. V případě zrušení pozvánky budou z turnaje odebrány všechny týmy nominované tímto klubem.

**Nominace týmů na turnaj**

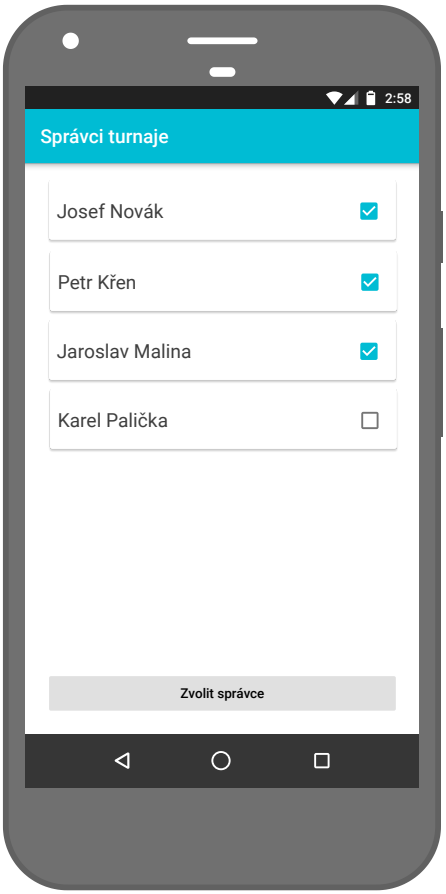
Z této obrazovky (3.8) může trenér nominovat své týmy na turnaj. Týmy jsou rozdělené do několika seznamů podle svých kategorií, mezi kterými lze přepínat pomocí tlačítek v horní části. Dostupné jsou pouze kategorie, pro které je turnaj pořádán. Nominovat tým je možné pomocí tlačítka u příslušného týmu, u týmů již nominovaných je možné nominaci zrušit, pokud zatím nebyl vybrán mezi hrající týmy.

**Správa hřišť**

Na této obrazovce (3.10) je správci zobrazen přehled o hřištích přiřazených k turnaji. Hřiště jsou přiřazována k vytvořeným zápasům, měli by tedy být přidány k turnaji předtím, než bude tvořen harmonogram.




Obrázek 3.6: Detaily turnaje

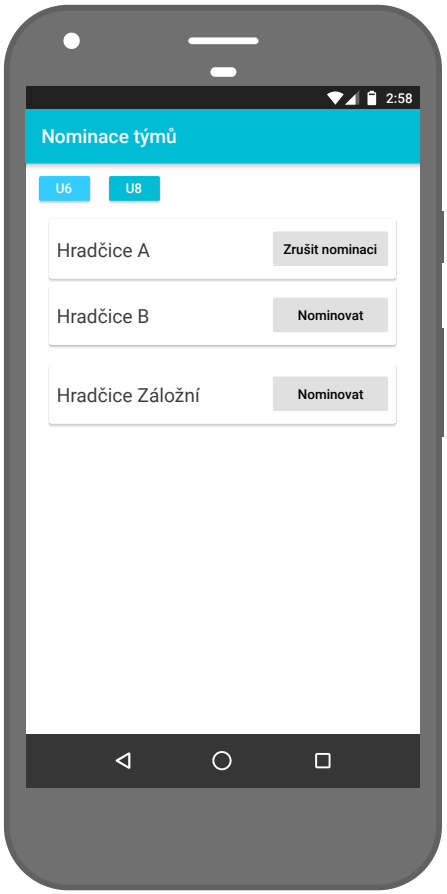



Obrázek 3.7: Zvolení správců

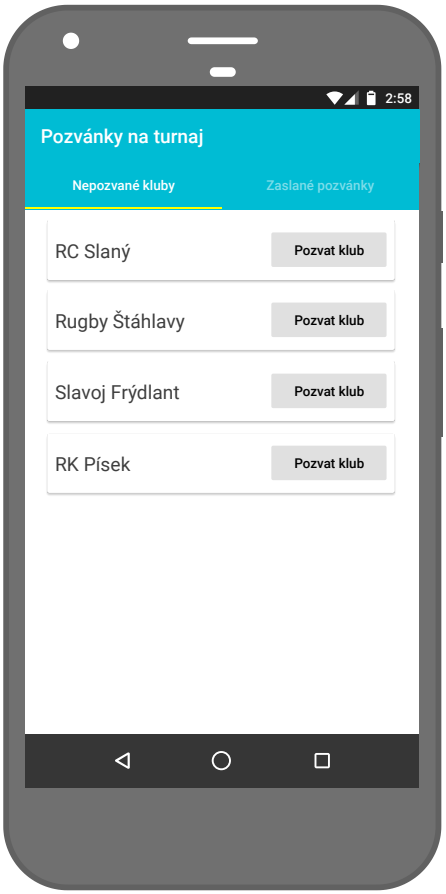
Pomocí tlačítka v dolní části má pořadatel možnost vytvořit nové hřiště po zadání popisu a přiřazení jedné z kategorií turnaje. Popis slouží k identifikaci hřiště a je zobrazován u zápasů odehrávajících se na daném hřišti. Vytvořené zápasy jsou zobrazeny v seznamech, rozdělených podle kategorie hřiště. U vytvořených hřišť si může pořadatel po kliknutí na tlačítko zobrazit menu s dalšími možnostmi pro hřiště. Zde může dodatečně změnit popis hřiště, nechat si zobrazit zapisovací kód pro zapisování zápasů hřiště a může hřiště z turnaje odstranit, pokud na hřiště nejsou rozvrženy žádné zápasy. Dále může ke hřišti přiřadit rozhodčího ze svého klubu. Pro jednoduchost jsou rozhodčí přiřazováni ke hřištím, místo k jednotlivým zápasům. Poté jsou automaticky přiřazeni k zápasům na těchto hřištích.

**Přihlášené týmy**

Zde (3.11) jsou uživateli zobrazeny seznamy týmů, které byly na turnaj nominovány, rozdělené podle věkových kategorií. U každého týmu může




Obrázek 3.8: Nominace týmů na turnaj

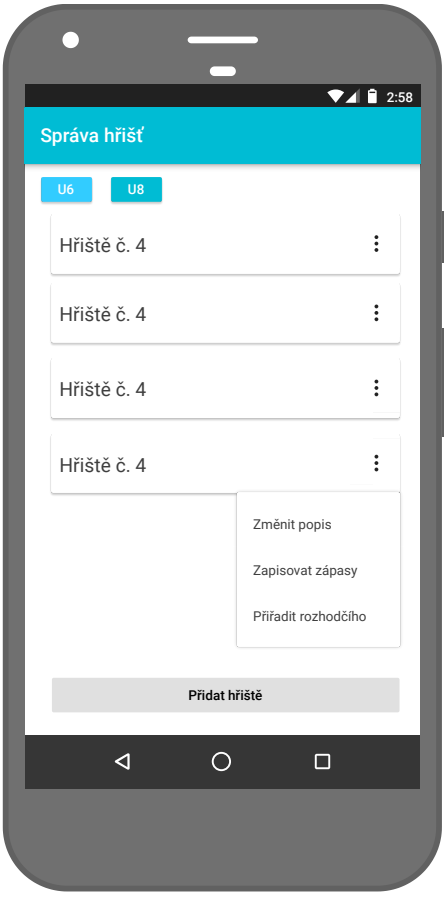



Obrázek 3.9: Pozvání klubů na turnaj

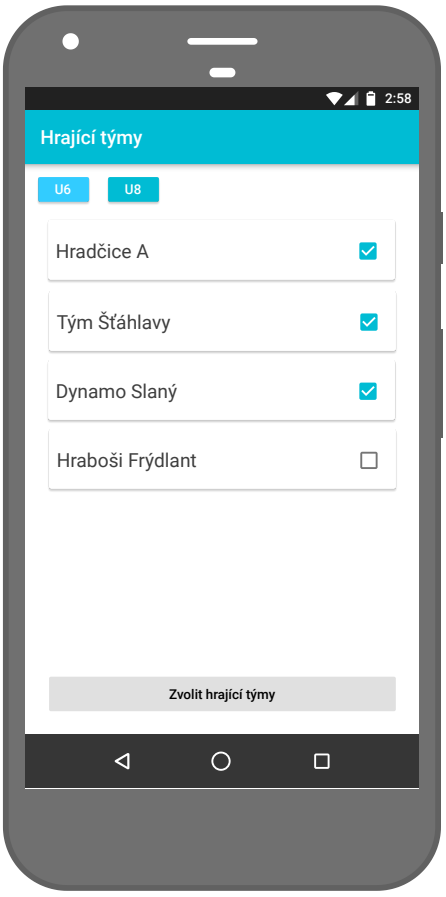
uživatel zaškrtnutím příslušného políčka zvolit tým pro hru na turnaji, případně po předchozím zvolení může odškrtnutím tlačítka účast týmu zrušit (pokud se tým již neúčastní nějakých zápasů). Volbu uživatel potvrdí pomocí tlačítka v dolní části obrazovky, jinak se může vrátit zpět bez uložení změn.

**Seznam zápasů turnaje**

Z detailů turnaje si může uživatel nechat zobrazit zápasy, které jsou naplánované pro turnaj (3.12). Zápasy jsou rozděleny do několika seznamů podle přiřazené věkové kategorie, seznamy lze přepínat pomocí tlačítek v horní části obrazovky. Zápasy jsou v seznamu seřazeny podle času zahájení. Pro každý zápas v seznamu jsou uvedeny některé základní údaje: čas zahájení, hrající týmy, číslo zápasu a výsledky, pokud již byly zaznamenány. Podobně jako u turnajů je možné si nechat zobrazit informace o zápasu po kliknutí na něj.




Obrázek 3.10: Správa hřišť turnaje



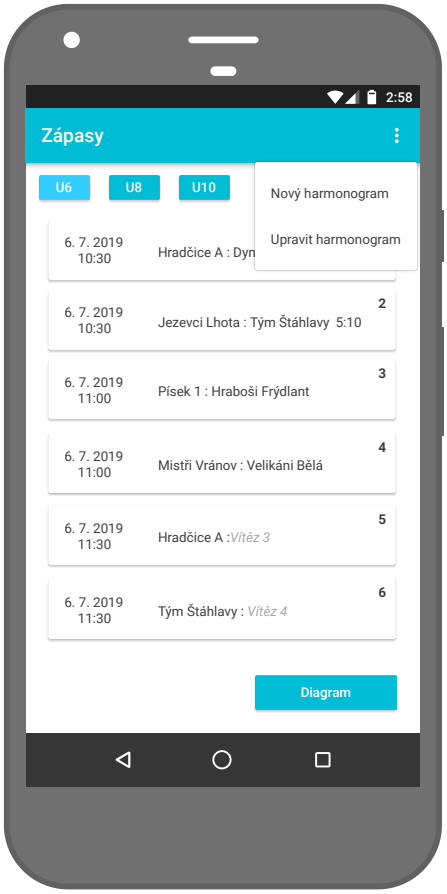

Obrázek 3.11: Výběr hrajících týmů turnaje

V případě, že je uživatel správcem turnaje, je pro něj dostupné tlačítko v pravém horním rohu pro vytvoření nového harmonogramu nebo úpravy stávajícího harmonogramu.

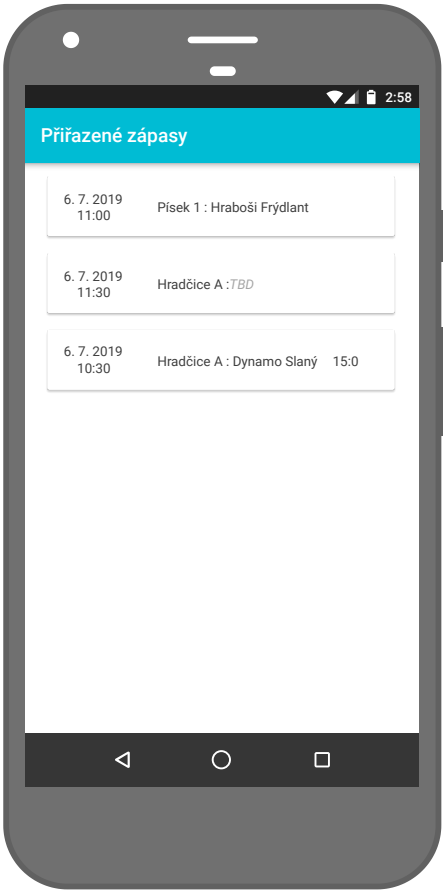
Kromě lineárního seznamu si může uživatel pomocí tlačítka v pravém dolním rohu nechat zobrazit zápasy v podobě diagramu (3.14). V něm jsou zápasy odděleny podle kol pro lepší orientaci. V případě existujících postupů mezi zápasy je tento vztah znázorněn.

**Seznam přiřazených zápasů**

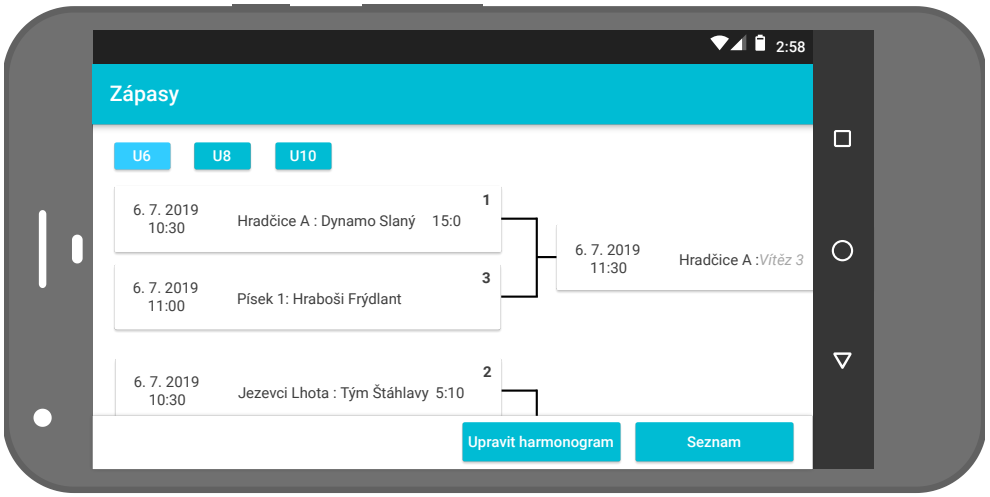
Rozhodčí a trenéři si kromě zápasů turnaje mohou z hlavní obrazovky zobrazit seznam pro ně relevantních zápasů, s cílem usnadnit orientaci v zápasech, kterých by se měl účastnit. Pro rozhodčího se jedná o seznam zápasů, ke kterým je přiřazen skrze hřiště (3.13), v případě trenéra se jedná o zápasy jeho týmů. Zápasy v seznamu jsou seřazeny podle času zahájení, již odehrané zápasy jsou umístěny na konec seznamu.




Obrázek 3.12: Seznam zápasů turnaje




Obrázek 3.13: Seznam zápasů rozhodčího




Obrázek 3.14: Diagram zápasů turnaje

**Vytvoření a úprava harmonogramu**

Pomocí této obrazovky (3.15) pořadatel nastavuje parametry pro automatické vytvoření zápasů. Detaily pro automatické vytváření jsou popsány v sekci 3.4. Uživatel tak zvolí formát zápasů, specifické nastavení pro formát a obecné parametry pro všechny formáty: čas zahájení, délku zápasů, délku pauz mezi koly a specifické přestávky.

Na základě nastavených parametrů může pořadatel vytvořit zápasy pomocí tlačítka v dolní části obrazovky. Před samotným uložením jsou zápasy prezentovány v podobě seznamu (a diagramu), podobné obrazovkám 3.12 a 3.14. Zde si může uživatel prohlédnou všechny navržené zápasy a pokud je s nimi spokojený, může harmonogram akceptovat, případně se může vrátit beze změn zpět a dodatečně upravit nastavení. V případě akceptace nové zápasy nahradí existující zápasy pro danou kategorii.

Ze seznamu vytvořených zápasů může uživatel upravit stávající harmonogram (3.16), podle popisu v sekci 3.4.2. Zde je možné upravit zatím neodehrané zápasy změnou některých parametrů použitých při generování, odebráním některého z hrajících týmů nebo odebráním hřiště, na kterém jsou naplánované zápasy.

**Detaily zápasu**

Ze seznamu zápasů si uživatel, podobně jako u turnajů, může po kliknutí na kartu zápasu nechat zobrazit dialog s detailními informacemi o zápasu. Uživateli, který je zároveň správcem turnaje, jsou navíc zobrazena další tlačítka pro správu zápasu. Pomocí nich může jednotlivé zápasy zrušit, nastavit pro zápasy odeslání notifikace o zahájení a nechat si vygenerovat zapisovací kód pro zaznamenávání výsledků zápasu.

**Zaznamenávání výsledků**

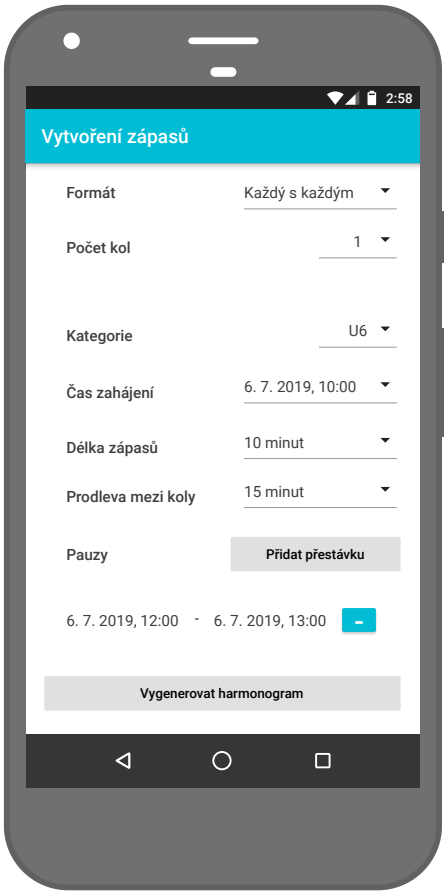
Zapisování výsledků zápasů může být svěřeno libovolnému uživateli, bez nutnosti, aby měl vlastní účet nebo byl přihlášen. Na hlavní obrazovce má uživatel k dispozici tlačítko *Zapisovat výsledky*, po kliknutí na něj je v dialogovém okně vyzván pro zadání zapisovacího kódu (3.18).

Zapisovací kód si může nechat systémem vytvořit pořadatel buď pro jeden konkrétní zápas z obrazovky detailů zápasu (3.17), nebo si ze seznamu hřišť (3.10) může nechat vytvořit kód pro zapisování hřiště. Vytvořené kódy jsou unikátně přiřazené k jednomu konkrétními zápasu nebo hřišti na dobu několika minut. Po vypršení času si musí pořadatel zažádat o kód nový.

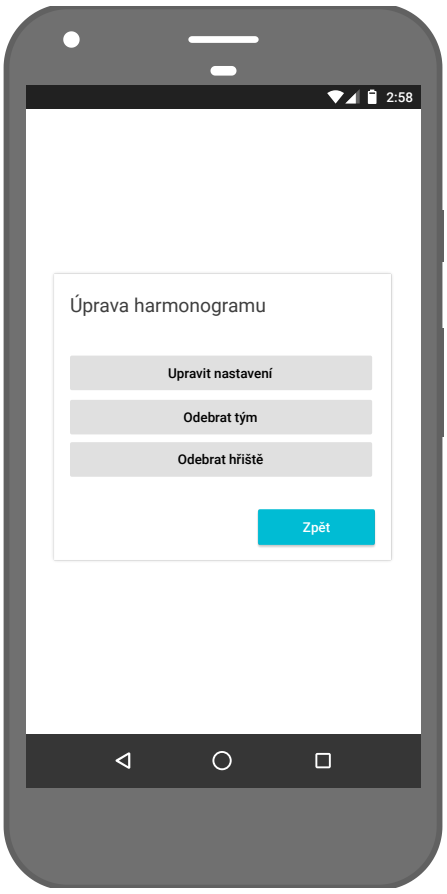
Získaný zapisovací kód předá zvolenému zapisovateli, který jej může zadat do dialogového okna pro zapisování příslušného zápasu/hřiště.

**Zapisování hřiště**

Při zapisování hřiště je uživateli zobrazen seznam zápasů naplánovaných




Obrázek 3.15: Automatického vytvoření zápasů




Obrázek 3.16: Úprava stávajícího harmonogramu

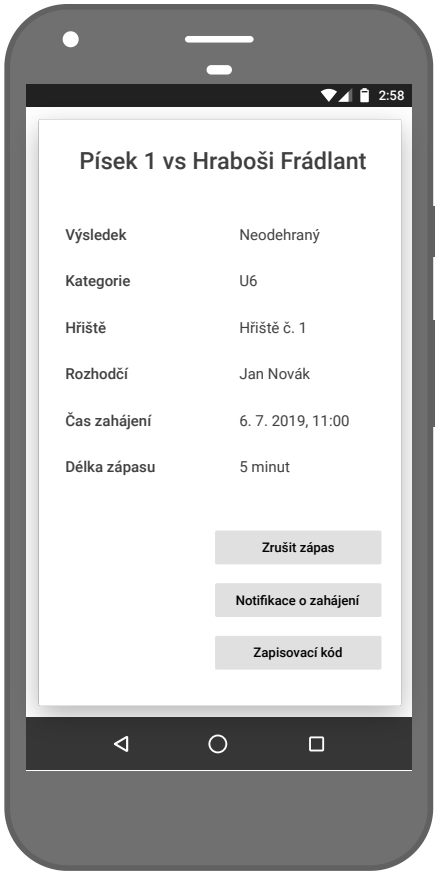
pro zapisované hřiště (3.19). Pro každý zápas je uživateli několik minut před plánovaných začátkem umožněno zahájit zapisování výsledků pomocí zobrazeného tlačítka o zápasu. Po zapsání zápasu je uživatel navrácen zpět na seznam zbylých zápasů.

Pomocí tlačítka *Ukončit zapisování* může zapisovatel po zapsání všech zápasů ukončit zapisování a vrátit se zpět na hlavní obrazovku. Poté se již k zapisování hřiště nemůže vrátit.

Zapisovatel hřiště může zaznamenávání neodehraných zápasů přenechat jinému uživateli pomocí tlačítka *Předat zapisování*. Princip je stejný jako při prvním přiřazení zapisovatele: uživateli se zobrazí dočasně přidělený kód, který může předat zvolenému následníkovi, který jej může zadat z hlavní obrazovky aplikace.

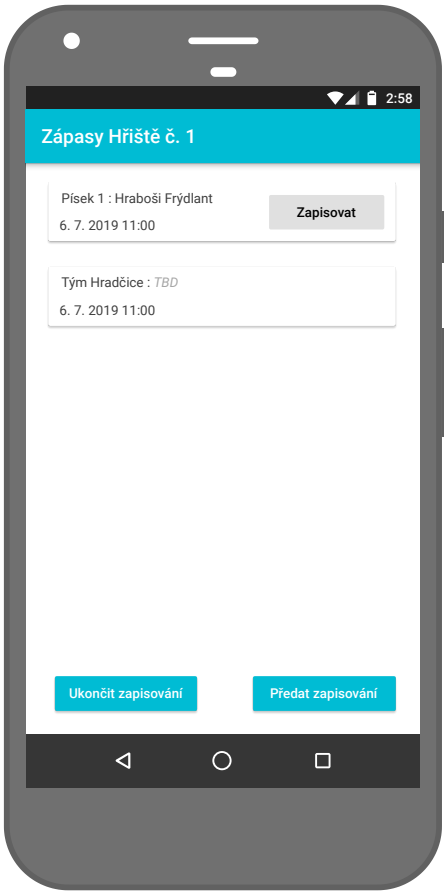
**Zapisování zápasu**

Pomocí této obrazovky (3.20) zapisovatel zaznamenává výsledky pro při-

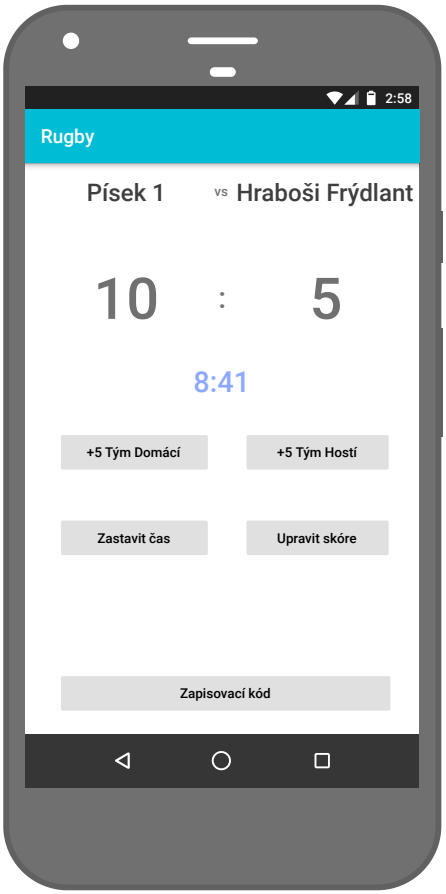



Obrázek 3.17: Detaily zápasu


Obrázek 3.18: Zapisování výsledků




Obrázek 3.19: Zapisování zápasů hřiště




Obrázek 3.20: Zaznamenávání výsledků zápasu

řazený zápas nebo pro zápasy přiřazeného hřiště. Pro zápasy se zaznamenává pouze skóre týmů v podobě „pětek“, tedy po 5 bodech. Uživatel má možnost přidávat každému z týmů po jedné „pětce“ (přidání pěti bodů), v případě chyby může pomocí tlačítka *Upravit skóre* konkrétně nastavit aktuální skóre. Před zahájením zapisování musí uživatel spustit časomíru, která je automaticky nastavená podle délky zápasu. Časomíru je následně možné pozastavit a znovu spustit, jinak ale slouží pouze pro orientaci zapisovatele a nijak neomezuje samotné zapisování.

Zapisování je možné ukončit pomocí nejspodnějšího tlačítka, po kterém je ještě zapisovatel vyzván pro potvrzení. Následně se konečné výsledky zaznamenají a zápas je považován za zapsaný. Uživatel je navrácen na hlavní obrazovku, pokud zapisoval pouze tento zápas, nebo v případě, že zapisuje hřiště, na seznam zbylých zápasů.

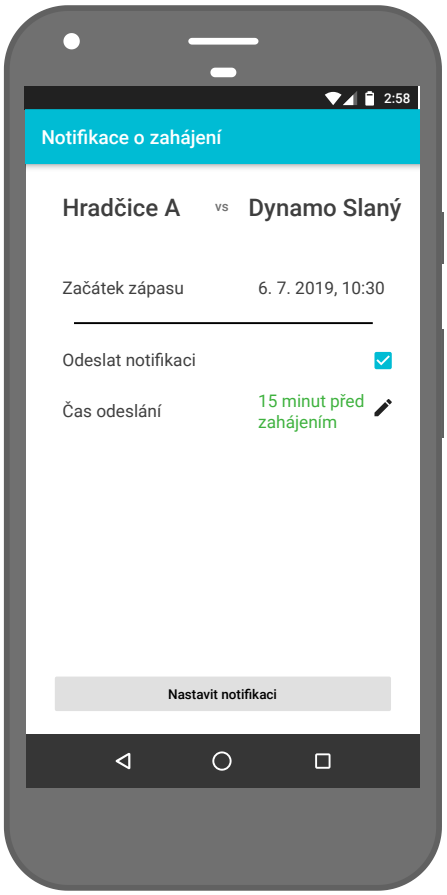
**Nastavení notifikace o zahájení zápasu**

Na této obrazovce (3.21) může správce turnaje nastavit zaslání notifikace o zahájení zápasu očekávaným účastníkům zápasu. Zaškrtnutím nebo odškrtnutím políčka může uživatel nastavit, zdali má být / nemá být notifikace zaslána, při zaškrtnutí má také možnost nastavit, kdy má být notifikace zaslána (pokud čas odeslání již proběhl, je notifikace odeslána ihned po nastavení). V případě, že zápas již měl začít, nemusí být zvolen čas, místo toho může být notifikace odeslána ihned.

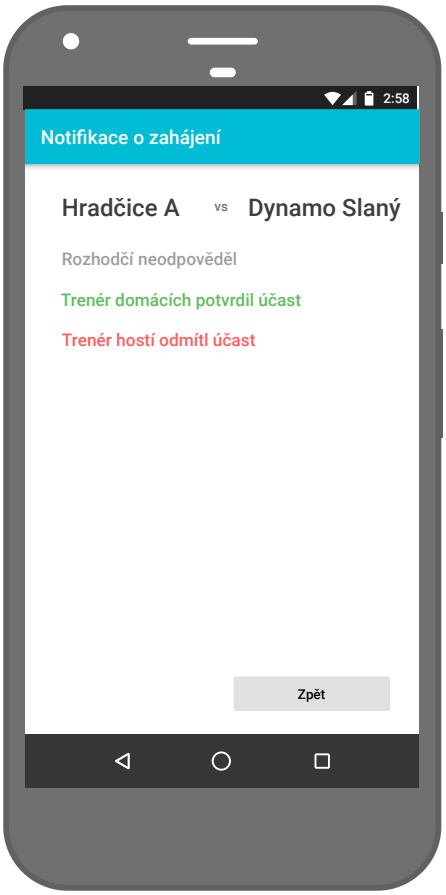
Pokud notifikace zatím nebyla nikomu odeslána, je možné změnit čas odeslání nebo odeslání zrušit. Jakmile byla odeslána, správce a účastníci zápasu si mohou zobrazit výsledky notifikace (3.22). Zde je uživateli zobrazeno, zdali rozhodčí a trenéři týmů potvrdili účast, zamítli účast nebo zatím neodpověděli.

**Odpovědi na notifikace**

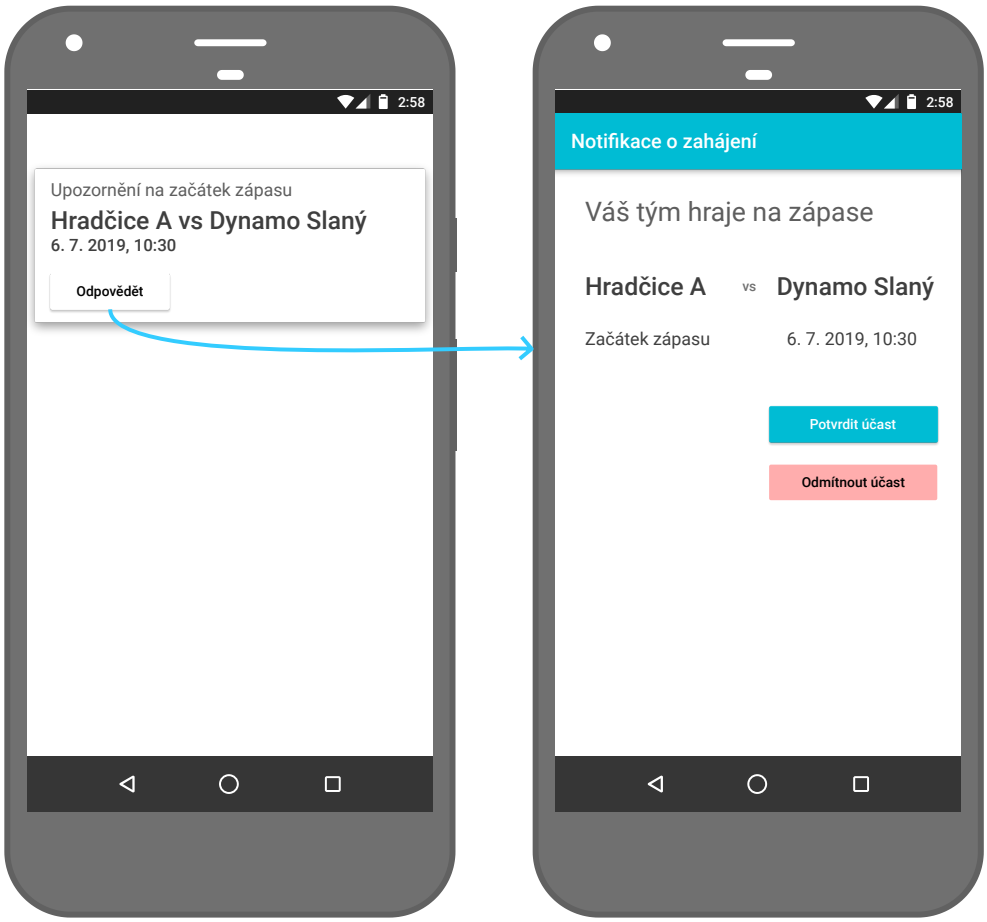
Uživatel může na přijatou notifikaci odpovědět pomocí tlačítka ze seznamu notifikací (3.23). Uživateli je poté zobrazena obrazovka s informacemi o jeho roli v zápasu, hrajících týmech a času zahájení. Pomocí tlačítek může potvrdit nebo odmítnout účast.




Obrázek 3.21: Nastavení notifikace o zahájení zápasu




Obrázek 3.22: Odpovědi na zaslanou notifikaci




Obrázek 3.23: Přijatá notifikace

Kapitola 4

# Implementace

V této kapitole jsou představeny použité technologie použité pro implementaci prototypu, je představena platforma operačního systému Android a je popsán postup řešení některých hlavních problémů. Instalaci a použití výsledného prototypu jsou věnovány přílohy B a C.

## 4.1 Použité technologie

V této části budou představeny některé technologie použité během implementace, a jakou práci konkrétně vykonávají.

**Spring**
Spring je obsáhlý open source framework pro jazyk *Java*, mimo jiné podporující i jazyk *Kotlin* [9]. Zajišťuje většinu hlavních funkcí na serveru.

- Zpracování HTTP požadavků od Android uživatelů. Automaticky zajišťuje konverzi parametrů a těl požadavků na objekty, a rovněž automaticky sestavuje odpovědi.
- Podpora dependency injection.
- Nabízí možnosti zabezpečení aplikace. Konkrétně v systému je tak zabezpečen přístup k určitým URI. Od požadavků na určité URI je možné vyžadovat autentizace, a zároveň je možné k nim přiřadit filtry, kterými projde požadavek před jeho vykonáním. Filtry mají přístup k informacím požadavku a mohou jej autorizovat.
- Značně zjednodušuje práci s relační databází pro trvalé ukládání dat. Zajišťuje samotné připojení k databázi, pro připojení stačí upravit konfigurační soubor pro specifikaci URI databáze a jaký ovladač má být pro připojení použit. Za předpokladu dostupnosti správného ovladače tak lze mezi různými databázovými řešeními měnit změnou jednoho řádku.

Dále Spring využívá JPA pro ORM. Umožňuje tak například automatické vytvoření databázového schématu z anotovaných tříd, a abstrahuje ukládání a načítání objektů do databáze. Lze se tak vyhnout přímé práci s databází, například pomocí SQL.

V systému je použita nadstavba Spring Boot, která má za cíl zjednodušit nasazení samotných aplikací. Kromě snížení množství nutné konfigurace například automaticky používá aplikační server Tomcat.

**Java JWT**
Knihovna implementující *JSON Web Token* [10]. Ty jsou v rámci systému používány pro ověřování identity uživatelů. Uživatelé se nejdříve přihlásí svým heslem, a následně jim je přidělen dočasný token pomocí kterého se autentizují v dalších požadavcích. V této části se rovněž tokeny přiřazují zapisovatelům po zadání zapisovacího kódu. Samotná knihovna se stará o vydávání nových tokenů a validaci tokenů přicházejících.

**Gradle a Maven**
Oba nástroje slouží k automatickému sestavování aplikací, zajišťují tak mimo jiné překlad zdrojového kódu, spravují závislosti na externím softwaru a spouštění automatických testů [11, 12]. Gradle je používán pro sestavení Android aplikace, kde se jedná o standard [13]. Maven byl z důvodu větších osobních zkušeností zvolen pro serverovou část.

**Gson**
Knihovna zajišťující převedení Javovských objektů do řetězce ve formátu JSON a zpětné převedení z řetězce na objekt. Je využívána v Android aplikaci pro obousměrné převedení těl HTTP požadavků. Její výhodou je snadné použití, kdy pro většinu tříd není třeba provádět žádnou další konfiguraci.

**Retrofit**
Knihovna využívající *OkHttp* pro vysokoúrovňovou abstrakci nad HTTP požadavky. V aplikaci je využíván pro zpracování požadavků na server.

**Espresso**
Oficiální framework pro automatické testování uživatelského prostředí Android aplikací [14]. Podporuje deklarativní vybírání prvků na obrazovce, se kterými je následně možné provádět akce, například kliknutí na tlačítko nebo vyplnění textového pole. Samotné testy probíhají na reálném nebo virtuálním stroji. Více se frameworku věnuje sekce 5.1.2.1 v následující kapitole.

**Firebase**
Platforma nabízející cloudové služby pro vývoj mobilních, webových a

desktopových aplikací. Mezi nabízené služby patří například sběr dat, autentizace uživatelů, databázové řešení a mnohé další. Některé nabízené služby jsou placené, zatímco jiné jsou dostupné zdarma.

V rámci aplikace je využívána funkce *Cloud Messaging*, kterou je možné používat zdarma. Tato služba umožňuje posílat koncovým zařízením zprávy, pomocí kterých je implementováno zasílání notifikací. Pro Android je zasílání zpráv závislé na *Google Play Services*. Hlavní výhodou tohoto řešení je snadné zasílání notifikací i uživatelům, kteří aplikaci aktuálně nepoužívají, nebo ji ani nemají zapnutou.

## 4.2 Představení Android platformy

V této sekci budou představeny některé základy vývoje pro platformu Android, s vývojovým prostředím Android Studio.

### 4.2.1 Prvky uživatelského rozhraní

Všechny prvky uživatelského prostředí mají společnou nadtřídu *View*. Objekty těchto tříd je možné vytvářet v samotném kódu, jednodušší je však vytvoření View pomocí *layout* souborů ve formátu XML. V nich lze deklarovat celou hierarchii, spolu s parametry pro jednotlivé prvky. Mezi nejpoužívanější podtřídy patří:

- *TextView* pro zobrazení textu s možností nastavení vlastností jako velikost a barva písma, font a mnohé další. Slouží jako hlavní nadtřída pro další prvky zobrazující text. Například *EditText* umožnuje uživateli zadat vlastní text. Tento vstup umožnuje automaticky omezit, například nastavením maximální délky vstupu nebo určením, jakého typu má vstup být (například při zvolení, že vstupem má být číslo, uživateli automaticky zobrazí klávesnici pouze s číslicemi).

- *Button* a *ImageButton* pro realizaci tlačítek (akci pro kliknutí je ale možné nastavit i pro základní View).

- *CheckBox* pro zaškrtávací tlačítka.

- *RadioGroup* a *RadioButton* pro skupinu tlačítek s výběrem jednoho z nich.

Důležitou skupinou jsou pak *ViewGroup* třídy, které sami mohou vlastnit další View objekty. Těchto kontejnerových tříd existuje velké množství s různými způsoby zobrazení přiřazených View.

- *LinearLayout* je v aplikaci nejpoužívanější třídou pro seskupení prvků, které jednoduše zobrazuje v jedné horizontální nebo vertikální řadě.

- *CardView* slouží pro zvýraznění položek v seznamech dodáním prostorového dojmu.
- *ScrollView*, *NestedScrollView* a *HorizontalScrollView* pro realizaci uživatelského prostředí nezávislého na velikosti displeje.
- *RecyclerView* se používá pro zobrazení seznamů s předem neznámým počtem prvků (například při zobrazení seznamu turnajů nebo zápasů). Seznamy jsou automaticky rolovací, je tak možné zobrazit libovolně dlouhý seznam bez ohledu na velikost obrazovky. Pro vytvoření jednotlivých prvků seznamu je třeba vytvořit vlastní implementaci třídy *Adapter*. Její metody jsou volány pro určení počtu prvků v seznamu a stará se o vytvoření a inicializaci *View* pro reprezentaci každého prvku.

Počítá se rovněž s možností zdědit některou existující třídu pro realizaci požadovaného chování. Toho se v aplikaci využívá například pro diagram zápasů.

### 4.2.2 Aplikační manifest

Každá aplikace musí obsahovat soubor *AndroidManifest.xml*, ve kterém jsou specifikovány důležité informace o aplikaci. Jedná se například o jméno aplikace, její ikonu, povolení vyžadovaná aplikací (připojení k internetu...). Dále zde také musí být definovány třídy *Activity* (popsané dále), *Service* a *BroadcastReceiver*, které jsou použity v aplikaci, spolu s filtry pro jejich spuštění.

### 4.2.3 Resources, lokalizace

V rámci vývoje jakékoli aplikace je vhodné vyhradit různé neměnné zdroje (numerické konstanty, text) mimo zdrojový kód na společné místo, na které se bude z kódu odkazovat. Výhodou jsou jednodušší budoucí úpravy, kde místo prohledávání celého kódu a změny při každém použití stačí změnu provést jednou na známém místě.

V Android projektu je pro tuto funkci vyhrazený adresář *res/*, do kterého je možné umístit další podadresáře pro oddělení různých skupin zdrojů [15]. Například v adresáři *res/values/* se nacházejí soubory agregující různé primitivní typy (*strings.xml* obsahující řetězce, *integers.xml* s konstantními celočíselnými hodnotami, *dimen.xml* pro číselné rozsahy UI prvků, ...) ke kterým je možné přistupovat pomocí unikátního klíče.

Kromě tohoto rozdělení lze k podadresářům přidat kvalifikátory, podle kterých je možné použít odlišné hodnoty v závislosti na konfiguraci zařízení. Jednou z možností je rozdělení podle používaného jazyka. Například je možné používat dva soubory pro řetězce, *res/values/strings.xml* a *res/values-en/strings.xml*. Primárně se tak budou používat řetězce z prvního souboru, pokud je ale aplikace spuštěna na zařízení používající anglický jazyk, budou

se používat řetězce ze souboru druhého. Přestože se v aplikaci počítá pouze s podporou češtiny, podpora dalších jazyků by tak znamenala pouze překlad jednoho souboru.

### 4.2.4 Activity a Fragment

*Activity* (dále aktivita) je základní třída pro zobrazení uživatelského prostředí. Samotnou podobu obrazovky lze nastavit pomocí vytvořené instance třídy View (s největší pravděpodobností vytvořené ze souboru). Jedná se o poměrně komplexní třídu nezávislou na běhu samotné aplikace. Například pro odpovědi na notifikace zápasů je zobrazena aktivita, která může být zobrazena i když samotná aplikace vůbec není spuštěna. Všechny používané aktivity musí být definovány v manifest souboru.

Přestože je možné uživatelské prostředí realizovat jen s použitím velkého množství aktivit, jednodušší je použití třídy *Fragment*, dále fragment. Ten obdobně jako aktivita může hostit přiřazené View pro interakci s uživatelem, musí však existovat v kontextu nějaké aktivity. Samotná aktivita může obsahovat několik fragmentů najednou, ať už zobrazených po jednom v hierarchii přes sebe, nebo zobrazených najednou na jedné obrazovce. V ideálním případě by tak uživatelské prostředí mělo být rozděleno na co nejmenší samostatně použitelné části, které budou realizovány pomocí fragmentů.

Aktivity a fragmenty se řídí vlastními životními cykly. Při změně stavu jsou zavolány odpovídající metody, kterými může uživatel řídit jejich chování. V případě aktivit tak lze použít vlastní implementaci metody *onCreate* pro inicializaci (zde je například zvolena podoba aktivity pomocí metody *setContentView*) a naopak v metodě *onPause* lze implementovat logiku pro ukončení aktivity.

Fragmenty i aktivity mohou být systémem zničeny systémem pro uvolnění systémových prostředků. K tomu může dojít především pokud je aplikace delší dobu na pozadí, ale i během používání mohou být cílené fragmenty a aktivity, které nejsou aktuálně viditelné. V takovém případě systém komponentu znovu zkonstruuje, jakmile bude potřeba.

## 4.3 Implementace Model-View-ViewModel

Jak již bylo zmíněno v kapitole návrhu, samotná Android platforma poskytuje komponenty pro implementaci vzoru MVVM. Jedná se především o třídy *ViewModel* a *LiveData*.

ViewModel, jak název napovídá, slouží pro realizaci střední vrstvy ViewModel. Uchovává tak data používaná View třídami (především aktivity a fragmenty). Každá instance je vázaná k životnímu cyklu nějaké komponenty. ViewModel je tak udržován v paměti, dokud není ukončena vázaná komponenta.

V UI třídách lze získat referenci na požadovaný ViewModel pomocí třídy *ViewModelProvider*. Ta spravuje všechny ViewModel instance pod konkrétní komponentou. Pomocí její metody *get* lze získat referenci na libovolný ViewModel. Pokud už existuje ViewModel vázaný na komponentu, je vrácena reference na něj, v opačném případě je vytvořena nová instance. Tímto způsobem je možné snadno sdílet stejná UI data napříč několika fragmenty, které existují v rámci jedné aktivity – mohou si požádat o ViewModely vázané na rodičovskou aktivitu a všechny fragmenty tak budou pracovat se stejnou instancí.

LiveData slouží pro implementaci návrhového vzoru *Observer*. Obaluje sledovaná data, která tak mohou být pozorována pomocí vlastní implementace rozhraní *androidx.lifecycle.Observer*. Výhodou tohoto řešení je jednodušší použití v UI třídách, kde může být problematický jejich životní cyklus. Při registraci pozorovatele je možné specifikovat instanci třídy se životním cyklem (typicky aktivity a fragmenty). Pozorovatel potom nebude upozorněn na změny, pokud je aktivita/fragment na pozadí (v případě, že se stane znovu aktivní, jsou pozorovateli předána aktuální data) a v případě, že je aktivita nebo fragment zcela odebrán, je automaticky odebrán ze seznamu pozorovatelů daného zdroje.

Třetí část vzoru, Model, je implementován formou *repozitářů*. Repozitáře nabízejí metody jakožto zdroj dat pro ViewModel, avšak abstrahují, odkud reálně data pocházejí.

V případě implementované aplikace jsou všechny požadavky řešeny výhradně získáním dat ze serveru, po vzoru výše uvedeného návrhu by bylo možné v budoucnu rozšířit aplikaci o lokální databázi, ze které by mohli být předávány stále aktuální data.

## 4.4 Realizace notifikací

Zasílání notifikací probíhá pomocí funkce *Cloud Messaging* z platformy *Firebase*. Hlavním důvodem použití této technologie je značné zjednodušení zasílání notifikací v případě, že je aplikace na pozadí, nebo neběží vůbec. Implementovat takové chování v samotné aplikaci by bylo výrazně složitější a samotný systém je poměrně restriktivní v možnostech vykonávání stálých operací na pozadí [16]. Pro Android tak pro zasílání zpráv Firebase využívá stále běžících *Google services*. Jistou nevýhodou je pak limitace aplikace pro zařízení využívajících Google služeb.

V případě implementovaného systému jsou využívány pouze datové zprávy, kterými je možné zasílat textové hodnoty mapované klíči. Kromě toho je ale možné posílat „notifikační“ zprávy, které obsahují informace pro automatické zobrazení notifikaci při přijetí. V systému jsou místo toho notifikace zobra-

zovány manuálně v rámci systému, mimo jiné proto, že je dodatečně ověřena jejich platnost.

### 4.4.1 Inicializace

Před samotným použitím je třeba založit vlastní projekt ve Firebase konzoli. Dále je třeba přidat závislosti pro příjemce i pro zasílající server, závisející na konkrétních platformách a použitém sestavovacím systému. Pro každou stranu je rovněž třeba přidat konfigurační soubor, pro spojení s vytvořeným projektem. Pozornost si zaslouží serverová část, konfigurační soubor obsahuje privátní klíč pro autorizaci k zasílání zpráv pro daný projekt. Před zasílám zpráv musí být soubor manuálně použit k inicializaci.

### 4.4.2 Zasílání zpráv

Pro zaslání zprávy konkrétnímu příjemci je nutné znát jeho unikátní *token*, který se však může průběžně měnit. O aktuálně platný token si může příjemce sám zažádat a zároveň je upozorněn v případě, že mu byl přidělen token nový. Pro každý účet je tak v databázi uložen záznam o jeho aktuálním tokenu, který se aktualizuje při novém přihlášení a v případě že je uživateli přidělen nový.

Zasílaná zpráva je nejprve vytvořena pomocí návrhového vzoru *Builder*. Zde jsou zprávě předána data (v tomto případě reprezentace notifikace ve formátu JSON). Vytvořenou zprávu lze následně zaslat pomocí třídy *FirebaseMessaging*.

### 4.4.3 Příjem zpráv

Na straně Androidu lze přijímat zprávy pomocí vlastní implementace třídy *FirebaseMessagingService*. Ta zároveň implementuje třídu *Service*, která slouží pro vykonávání operací na pozadí, bez uživatelského prostředí. Musí tak být definována v manifestu, kde zároveň musí být zaregistrována s filtrem pro rozlišení Intentů, které ji mohou spustit.

V samotné třídě pak lze implementovat metody pro samotnou práci se zprávami. Základní metodou je *onMessageReceived*, ve které je předána samotná přijatá zpráva. Další zajímavou metodou je *onNewToken*, která je volána při přidělení nového tokenu uživateli.

### 4.4.4 Zobrazení notifikace

Z přijaté zprávy je deserializací vytvořena nová instance notifikace. Všechny třídy notifikací na straně Androidu implementují metodu pro sestavení instanci třídy Intent. Myšlenka je taková, že pro každý typ notifikaci bude existovat vlastní typ akce, kterou bude odebírat jeden konkrétní *BroadcastReceiver*. V samotné třídě přijímající zprávy tak dojde k sestavení notifikace,

zavolání její metody pro sestavení instance třídy Intent (do které notifikaci přidá potřebné informace pro její zobrazení) a vyslání této instance. BroadcastReceiver, který Intent přijme, se postará o zobrazení samotné notifikace.

Konkrétně o zobrazování notifikací o zahájení zápasu se stará *MatchNotificationReceiver*. Samotná instance systémové notifikace se konstruuje opět vzorem Builder. Mimo jiné lze nastavit ikonu, text nadpisu a text samotného těla. Pro odpověď je notifikaci přidáno tlačítko, které po stisknutí spustí předaný Intent. V tomto případě je k notifikacím přidáno tlačítko, které spustí aktivitu, pomocí které může uživatel na notifikaci odpovědět.

## 4.5 Synchronizace aplikace s databází

Ve výchozím systému nebyla nijak řešena synchronizace při současných změnách dat několika uživateli. Hlavním důvodem byla nemožnost kontaktování klientů samotným serverem. V této části tak došlo k rozšíření obou částí pro podporu oboustranné komunikace a zasílání zpráv o změnách v databázi.

### 4.5.1 Reakce na databázové operace

Prvním krokem je zajistit vykonání operací při změně objektu v databázi. Jednu z možností nabízí samotné rozhraní JPA. Pro databázové třídy (s anotací *@Entity*), je možné anotovat metody, které jsou poté zavolány v průběhu databázové transakce. Konkrétně jsou používány metody *@PostPersist*, *@PostUpdate* a *@PostRemove*, které jsou zavolány, jakmile je nějaká instance anotované třídy uložena, upravena nebo smazána v databázi.

Vzhledem k tomu, že tyto metody jsou volány během nedokončené transakce, se v dalších operacích pokračuje až poté, až se nad aktuální transakcí provede commit. K tomu je použita třída nabízená frameworkem Spring, *TransactionSynchronizationManager*, která nabízí statickou metodu pro registraci vlastní implementace rozhraní *TransactionSynchronization*, které definuje metody volané při různém stavu transakce. Pro požadované chování je implementována metoda *afterCommit*.

### 4.5.2 Implementace oboustranného připojení

Pro zasílání zpráv o změnách je využíván standard *WebSocket* [17]. Na serverové části je protokol implementován frameworkem Spring, v Android aplikaci pomocí OkHttp. K serveru se automaticky připojují přihlášení uživatelé nebo zapisovatelé. Inicializace připojení je prováděna HTTP požadavkem, autentizace uživatele tak probíhá stejně jako u jiných HTTP požadavků. V rámci připojení je poté možné snadno zasílat mimo binárních i textové zprávy.

### 4.5.3 Zasílání upozornění na změny

Před samotným zasláním změn jsou filtrováni uživatelé, kterým má být zasláno. Například pro zapisovatele hřiště jsou zasílány pouze upozornění pro změny v zápasech na jím zapisovaném hřišti, v případě změn v turnaji jsou zaslány upozornění uživateli, kteří jsou oprávněni k jeho sledování. Upozornění jsou poté zaslána aktuálně připojeným uživatelům.

Každý typ zprávy je reprezentován vlastní třídou, která dědí ze základní abstraktní nadtřídy. Samotné zprávy obsahují pouze základní informace, především zdali byla entity vytvořena/upravena/smazána a její identifikátor, pomocí kterého si potom může Android aplikace zažádat o aktuální podobu. Pro samotné doručení je v nadtřídě definována abstraktní metoda, které konkrétní zprávy implementují pro své doručení. Všechny typy zpráv jsou tak zpracovány deserializací z formátu JSON a zavolání této metody.

Samotná aktualizace uživatelského prostředí je poměrně jednoduchá díky použití MVVM vzoru. V repozitářích (implementovaných podle vzoru Singleton) nachází LiveData atributy obsahující identifikátory vytvořených, upravených nebo smazaných entit. Zpráva tak notifikuje příslušný repozitář o změně, který aktualizuje LiveData. Na změnu jsou následně upozorněny sledující ViewModely. Ty potom v závislosti na změnách mohou aktualizovat svá vlastní data, na změnu kterých jsou upozorněny UI třídy.

Celý systém může fungovat například následovně: uživatel aplikace se dívá na seznam turnajů, v tomto seznamu se nachází turnaj, kterému mezitím jiný uživatel změnil název. Při změně turnaje je tak prvnímu uživateli zaslána zpráva třídy *TournamentChangeMessage*. Ta při přijetí upozorní repozitář turnajů (*TournamentRepository*) na úpravu existujícího turnaje. Existující *TournamentsViewModel* zaregistroval pozorovatele pro sledování upravených turnajů. Ten je tedy upozorněn na id upraveného turnaje. ViewModel požádá repozitář o získání aktuální podoby turnaje, a poté aktualizuje svůj seznam turnajů s novou verzí turnaje. Na změnu seznamu turnajů je upozorněn fragment, který turnaje zobrazuje, a aktualizuje svou podobu.

Kapitola 5

# Testování

Testování je nedílnou součástí vývoje každého softwaru, od ověření funkčnosti konečného produktu po zjednodušení vývoje. V této kapitole bude popsáno, jakým způsobem byly aplikace a server testovány.

## 5.1 Automatické testy

Automatizované testy jsou velmi užitečné při samotném vývoji, kdy má vývojář možnost si ověřit, že po jeho zásazích do kódu je aplikace stále plně funkční.

Pro automatické testování je v Android aplikaci i na serveru použit framework JUnit. Na straně Androidu je situace mírně složitější. Testy jsou rozděleny na *local* testy, které využívají pouze v rámci *JVM* a nemají přístup k samotným Android třídám, a na instrumented testy pro testování Android částí, které využívají buď připojené fyzické zařízení nebo virtuální stroj [18].

### 5.1.1 Jednotkové testy

Jednotkové (Unit) testy představují nejmenší typy testů. Každý z testů ověřuje funkčnost jedné samostatné části, většinou třídy [19]. Uživateli tak dávají jistotu, že jednotlivé třídy fungují správně.

Na straně Androidu se testuje především správnost vytvořených zápasů a pozdějších úprav.

Na straně serveru jsou testovány *service* třídy, tedy třídy vykonávající samotnou hlavní logiku. Primárně se tedy testuje, zdali metody správně vrací nebo ukládají data. Pro zajištění izolace testů jsou závislosti testované třídy (především *repository* třídy pro přístup k datům a další service třídy) nahrazovány (mockovány) pomocí frameworku *Mockito* [20]. Pomocí něj je možné reálné implementace ostatních tříd nahradit objektem, u kterého je možné pro každý test specifikovat chování vybraných metod. Kontrolováno je tedy

pouze chování testované třídy. Pro použití v rámci Springu je instanční proměnné testovací třídy označit anotací *@MockBean*. Do proměnné poté bude automaticky přiřazena náhradní instance, která bude rovněž přiřazena k ostatním komponentám, které získávají instance stejného typu pomocí dependency injection.

### 5.1.2 Integrační testy

Integrační testy oproti Unit testům již testují širší části aplikace/systému, s komunikací mezi jednotlivými komponentami [19].

V serverové části jsou testovány *controller* třídy, která zpracovávají samotné HTTP požadavky, a volají různé service třídy, které obsahují samotnou logiku pro vyplnění požadavku. V testech jsou sestavovány samotné HTTP požadavky a následně se kontrolují odpovědi. Kontroluje se tak, zdali jsou metody dostupné na očekávaných adresách, správné zpracování těl požadavků, hlaviček požadavků a argumentů předaných v adrese, zavolání předpokládaných metod aplikační logiky a správnost odpovědí.

Při testech jsou kromě samotného controlleru rovněž testovány i používané filtry, kterými požadavky prochází před zpracováním ve třídě (především autentizační filtry), a převod z a do JSON formátu. Závislosti na service třídách jsou potom mockovány.

V budoucnu je možné testy rozšířit o celkové testy systému, bez mockování a s připojením ke skutečné databázi.

#### 5.1.2.1 Testy UI

Testy uživatelského prostředí se snaží plně testovat samotnou aplikaci, s rozdílem že odpovědi ze serveru jsou nahrazeny vlastní lokální třídou, která poskytuje předpřipravená data. Jsou tak testovány, mimo jiné, přechody mezi obrazovkami a celková navigace v aplikaci, zdali každá obrazovka správně vytváří a zobrazuje prvky na základě poskytnutých dat, zdali aplikace správně reaguje na neplatné požadavky, jestli jsou správně zpracovány validní požadavky, a další.

Testy uživatelského prostředí jsou (vzhledem k závislosti na samotných Android třídách) implementovány jako instrumented testy. Pro samotné testování uživatelského prostředí se používá výše zmíněný framework *Espresso*.

V každém testu je vždy nejdříve spuštěna vybraná aktivita, ze které je možné provádět akce na jednotlivých prvcích UI. Základní metodou je *onView*, která přijímá jako argument instanci třídy *Matcher* pro zvolení jednoho konkrétního *View* z aktuálně zobrazené hierarchie. Například je tak možné filtrovat prvky pomocí jejich id, přiřazeného tagu, zobrazeného textu, zdali jsou viditelné a mnohé další. Metody poté vrátí instanci třídy *ViewInteraction*, pomocí které lze nad nalezeným prvkem provádět akce, například provést kliknutí nebo vyplnit text.

## 5.2 Uživatelské testování a akceptační testování

V rámci uživatelských testů byli požádáni vybraní uživatelé ke splnění různých úkolů v aplikaci, kterou během testování používají poprvé. Cílem je zjistit, jakým způsobem je aplikace používána nezkušeným uživatelem, a odhalit možné nedostatky aktuálního návrhu (například neintuitivní uživatelské prostředí), s možnými nápady pro budoucí vylepšení. Pro uživatelské testování byly sestaveny testovací scénáře, v rámci kterých uživatel postupně vyzkouší většinu hlavních implementovaných funkcí. Scénáře jsou dostupné v příloze D.

Samotné testování bylo ztíženo pandemií covidu-19. Z toho důvodu proběhlo až 7. května na půdě Fakulty informačních technologií. V průběhu byli otestováni 3 uživatelé, kteří vyzkoušeli i části aplikace z ostatních prací. Každý z uživatelů tak byl testován přibližně 30 minut. Na testech byl osobně přítomen Martin Paul, moderování ostatních částí probíhalo vzdáleně. Testování této části bylo poněkud ztíženo dostupností pouze jednoho mobilního zařízení pro uživatele. Scénáře byly navrženy pro souběžné používání dvou zařízení. Na prvním zařízení by byl uživatel přihlášen jako pořadatel vytvořující nový turnaj, na druhém zařízení by uživatel zastával další účastníky systému (trenér nominující své týmy na vytvářený turnaj, rozhodčí odpovídající na notifikaci o zahájení zápasu, zapisovatel výsledků, ...). Některé části se tak testovali složitěji (nutnost častých změn přihlášeného účtu) a některé byly vypuštěny (přenechání zapisování dalšímu uživateli).

Během testování byla nalezena a později opravena občasná chyba při vytváření zápasů, které se někdy nevytvářely podle předpokladů. Také byl odhalen problém při zobrazování notifikací, způsobený rozdílným zobrazování notifikací na odlišných Android zařízeních, který vzhledem k pozdnímu termínu testování nebyl opraven. Jinak testování probíhalo bez problémů.

Dále byla během testování vypozorována možná budoucí vylepšení uživatelského prostředí:

- Zpřehlednění hlavní obrazovky, tlačítko pro zapisování vypadá stejně jako ostatní tlačítka, což dělalo problém s jeho vyhledáním.
- Upravení karet jednotlivých turnajů/zápasů v seznamech tak, aby bylo jasnější, že je možné na ně kliknout pro zobrazení dalších informací. Některým uživatelům chvilku trvalo, než na skutečnosti přišli.
- Na některých obrazovkách není jasné, že je třeba změny potvrdit (například při výběru hrajících týmů). Možné je lépe upozornit na nutnost potvrdit změny, například při pokusu o opuštění obrazovky bez uložení, nebo potvrzení nevyžadovat a změny provádět okamžitě.

Během uživatelského testování byl rovněž výsledný prototyp ukázán vedoucímu projektu, Ing. Jiřímu Chludilovi, pro demonstraci splnění požadavků. Demonstrace podobně jako testování musela probíhat vzdáleně, pomocí sdílení virtuálního stroje. Byly předvedeny všechny implementované požadavky a

zároveň byly zmíněny všechny odložené požadavky. Vedoucí byl s prototypem spokojen a akceptoval splnění realizovaných požadavků.

Kapitola 6

# Budoucí rozšíření

Z uvedených požadavků na aplikaci je v prototypu implementována většina, s výjimkou exportu výsledků a importu zápasů ze souboru. Hlavním důvodem je nedomluvení se na požadovaném formátu. Dále nebyly implementovány některé navržené funkce pro vytváření zápasů, především nemožnost nastavení specifických časových pauz mezi zápasy a omezené možnosti pozdější úpravy vytvořeného harmonogramu (limitována pouze na odebrání hřišť a týmů). V budoucnu by však bylo možné tyto funkce doimplementovat. Z nefunkčních požadavků se zatím nepočítá s použitím aplikace bez internetového připojení, aplikace tedy nezaznamenává výsledky zapsaných zápasů v případě výpadku připojení (kromě skóre jednoho zapisovaného zápasu).

Před reálným nasazením by bylo třeba vyřešit některé z nedostatků prototypu. Hlavním nedostatkem je absence stránkování a filtrování výsledků, což by představovalo velký problém při reálném nasazení s větším množstvím dat v systému. Pro řešení obou problémů je plně připraven framework Spring, pomocí kterého by mohlo být implementováno řešení po úpravách HTTP dotazů a uživatelského prostředí.

Dále by bylo vhodné sjednotit architekturu napříč všemi částmi aplikace. V rámci této práce byla stávající architektura přepracována na *Model-View-ViewModel* architekturu, ostatní části však vycházeli z předchozího návrhu (přímá komunikace napříč fragmenty a aktivitami) a aplikace tak není sjednocená.

Poněkud méně závažné, ale stále vhodné, by bylo sjednocení vzhledu uživatelského prostředí, které se napříč různými částmi mírně odlišuje.

V rámci projektu existuje rovněž několik nápadů pro možné nové funkce:

- Možnost komunikace mezi uživateli, například pomocí chatu v rámci klubu/turnaje nebo pomocí soukromých zpráv.
- Sdílení obsahu v rámci aplikace (fotky/videa ze zápasů, tréninků…).

- Větší integrace se systémem (propsání přiřazených zápasů do systémového kalendáře, použití map pro vizualizaci a přiřazení adres).
- Podpora dalších formátů pro zápasy (Turnaje se skupinami, vyřazovací formát s možnými prohrami...).

Dalším výrazným rozšířením by mohla být implementace uživatelské části systému na další platformy. Hlavním cílem by tak mohla být aplikace na operační systém iOS, případně by mohla být implementována alespoň část funkcí ve formě webové aplikace.

# Závěr

V rámci práce byla dle zadání rozšířena aplikace o podporu pořádání turnajů a zápasů a o zasílání notifikací. V rámci první kapitoly byly zpracovány požadavky na systém, podle kterých byly sestaveny a popsány modely případů užití, spolu s diagramem. Rovněž byly analyzovány tři existující aplikace s podobným zaměřením. V rámci návrhové kapitoly byla představena architektura aplikace, pomocí doménového modelu byla představena organizace dat v systému a návrhy uživatelského prostředí, byla představena podoba a ovládání aplikace. V implementační části byla představena platforma Android z hlediska vývoje aplikací a konkrétně byl popsán způsob realizace navrhované architektury, zasílání notifikací a synchronizace se serverem.

Výsledný prototyp implementuje, až ne některé malé nedostatky, všechny kladené požadavky. Těmto nedostatkům a dalším rozšířením pro reálné nasazení, se věnovala předchozí kapitola.

Cíl práce byl tedy splněn. Zdrojové kódy pro Android aplikaci i pro příslušný server jsou dostupné na dodaném fyzickém uložišti. Instalaci a zprovoznění obou částí se věnují příručky, dostupné v přílohách.

# Literatura


[1] Paul, M.: *Aplikace rugby – modul tréninky*. [bakalářská práce], 2021.

[2] Karlovský, D.: *Aplikace rugby - modul zákonného zástupce*. [bakalářská práce], 2021.

[3] Google: *Android Platform/API Version Distribution*. [software], verze 4.1.1. Viditelné při vytvoření nového projektu, po zvolení šablony pomocí tlačítka *Help me choose*. Dostupné z: `https://developer.android.com/studio`

[4] Apps, T.: *Winner*. [software], verze 9.12.3. Dostupné z: `https://play.google.com/store/apps/details?id=il.talent.winner`

[5] Poquesoft: *Tournament Manager*. [software], verze 1.28. Dostupné z: `https://play.google.com/store/apps/details?id=com.poquesoft.mistorneos`

[6] Place, C.: *Challenge Place*. [software], verze 2.41.3. Dostupné z: `https://play.google.com/store/apps/details?id=com.challengeplace.app`

[7] Google: *Guide to app architecture*. [online], 2021, [cit. 2021-04-13]. Dostupné z: `https://developer.android.com/jetpack/guide#recommended-app-arch`

[8] Suksompong, W.: Scheduling asynchronous round-robin tournaments. *Operations Research Letters*, ročník 44, č. 1, 2016: s. 96–100, ISSN 0167-6377, doi:https://doi.org/10.1016/j.orl.2015.12.008. Dostupné z: `https://www.sciencedirect.com/science/article/pii/S0167637715001674`

[9] VMware, I.: *Spring Framework*. [online], 2021, [cit. 2021-04-25]. Dostupné z: `https://https://spring.io/`

[10] M. Jones, M. a. d.: *JSON Web Token (JWT)*. [online], 2015, [cit. 2021-04-25]. Dostupné z: https://tools.ietf.org/html/rfc7519

[11] Inc., G.: *Gradle Build Tool*. [online], 2021, [cit. 2021-05-08]. Dostupné z: https://gradle.org/

[12] Foundation, A. S.: *Apache Maven Project*. [online], 2021, [cit. 2021-05-08]. Dostupné z: https://maven.apache.org/

[13] Google: *Configure your build*. [online], 2021, [cit. 2021-05-04]. Dostupné z: https://developer.android.com/studio/build

[14] Google: *Espresso*. [online], 2021, [cit. 2021-04-25]. Dostupné z: https://developer.android.com/training/testing/espresso

[15] Google: *App resources overview*. [online], [cit. 2021-04-08]. Dostupné z: https://developer.android.com/guide/topics/resources/providing-resources

[16] Google: *Guide to background processing*. [online], 2021, [cit. 2021-05-04]. Dostupné z: https://developer.android.com/guide/background

[17] Fette, I. a. A. M.: *The websocket protocol*. [online], 2011, [cit. 2021-04-17]. Dostupné z: https://www.hjp.at/doc/rfc/rfc6455.html

[18] LLC, G.: *Test your app*. [online], 2021, [cit. 2021-04-25]. Dostupné z: https://developer.android.com/studio/test/

[19] Mlejnek, J.: *Testování aplikací*. [přednáška], [cit. 2021-05-04]. Dostupné z: https://moodle-vyuka.cvut.cz/pluginfile.php/225520/mod_resource/content/2/09.prednaska.pdf

[20] *Mockito framework site*. [online], [cit. 2021-05-04]. Dostupné z: https://site.mockito.org/

[21] Google: *User opt-in for installing unknown apps*. [online], 2021, [cit. 2021-05-13]. Dostupné z: https://developer.android.com/distribute/marketing-tools/alternative-distribution#unknown-sources

[22] Google: *Download Android Studio and SDK Tools*. [online]. Dostupné z: https://developer.android.com/studio#downloads

[23] Google: *Enable developer options and USB debugging*. [online], 2021, [cit. 2021-05-04]. Dostupné z: https://developer.android.com/studio/debug/dev-options#enable

[24] s.r.o., J.: *Download IntelliJ IDEA*. [online]. Dostupné z: https://www.jetbrains.com/idea/download

Příloha A

# Seznam použitých zkratek

**ORM** Object relational mapping

**JVM** Java Virtual Machine

**HTTP** Hypertext Transfer Protocol

**JSON** JavaScript Object Notation

**OS** Operační systém

**MVVM** Model-View-ViewModel

**REST** Representational State Transfer

**AVD** Android Virtual Machine

Příloha B

# Instalační příručka

Tato část slouží jako návod pro instalaci a zprovoznění jak Android aplikace, tak její serverové části.

## Android aplikace

Aplikace aktuálně není distribuovaná skrze službu *Google Play*, která je oficiální metodou distribuce softwaru pro systém Android. Jako náhradou je možné nainstalovat aplikaci pomocí speciálních souborů používajících příponu *apk* (dále *apk soubor*) přímo z fyzického uložiště zařízení. Případně je možné sestavení pomocí zdrojového kódu.

### Instalace pomocí apk souboru

Po přenesení apk souboru na cílové zařízení je možno aplikaci snadno nainstalovat. Po zvolení souboru by měl systém sám nabídnout instalaci aplikace. V rámci zabezpečení však systém v základu nedovoluje instalaci aplikací mimo *Google Play*. Proto je před instalací nutné povolit instalaci aplikací z neznámých zdrojů. Tato možnost se může v závislosti na konkrétním systému nacházet na jiném místě v nastavení [21], při pokusu o instalaci by však mělo být uživateli nabídnuto tuto možnost povolit. Samotná instalace pak proběhne automaticky po stisku tlačítka *Instalovat*.

### Sestavení ze zdrojového kódu

Pro sestavení aplikace je možné provést pomocí oficiálního vývojářského prostředí *Android Studio*, které je volně dostupné ke stažení na oficiálních stránkách [22] (dále v textu se bude vycházet z verze 4.1.3). Při instalaci Android Studia je možné doinstalovat podporu *Android Virtual Machine* (dále *AVD*). Následně bude možné aplikaci spouštět na virtuálním stroji, bez nutnosti používat vlastní zařízení.

Po spuštění je kliknutí na tlačítko *File* a následně na tlačítko *Open* možné otevřít přímo složku obsahující zdrojový kód aplikace (*rugbyAndroid*). Po otevření by se podle konfiguračních souborů měla automaticky stáhnout požadovaná verze sestavovacího nástroje *Gradle* a projekt by se měl připravit ke spuštění. Tento proces může trvat několik minut. Po jeho dokončení by měl projekt vypadat podobně jako na tomto obrázku B.1. Především by měla být připravena konfigurace, na obrázku označená červeně.

### Vytvoření vlastního apk

Pomocí Android Studia je možné si snadno nechat vytvořit apk soubor pro pozdější instalaci na zařízení. Postup vytvoření ilustruje následující obrázek B.2. Po vytvoření souboru se v levé dolní části aplikace zobrazí okno, pomocí kterého je možné otevřít umístění vytvořeného souboru.

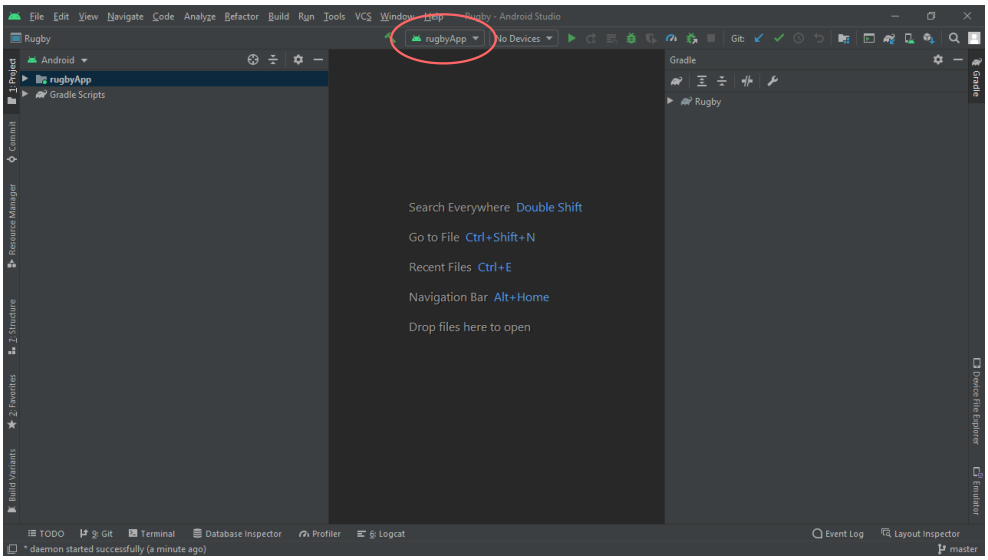
### Spuštění na fyzickém zařízení

Aplikaci je možné přímo z Android Studia spustit na vlastním zařízení, bez nutnosti instalace pomocí apk souboru. Pro to je nutné nejdříve na zařízení povolit *USB debugging*. Tato možnost se nachází ve skrytých vývojářských nastaveních. Pro jejich zobrazení je třeba v nastavení zařízení najít položku *Build Number* (*Číslo sestavení*), umístěnou v informacích o zařízení. Následně by měla být vývojářská nastavení dostupná v kořenu nastavení nebo pod systémovými nastaveními. Přesná umístění se liší v závislosti na verzích systému, pro konkrétnější informace je možné nahlédnout zde [23].

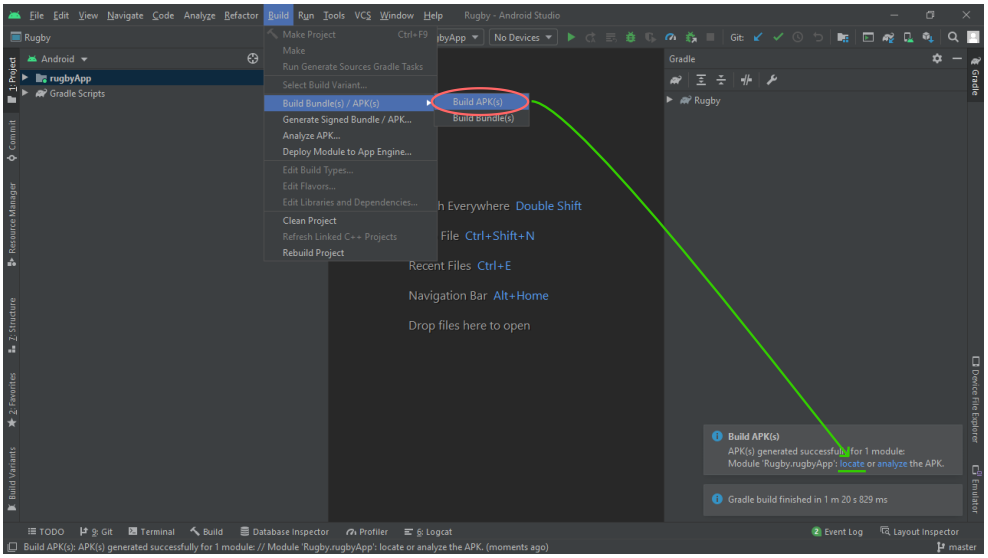
Po povolení tohoto nastavení by mělo být zařízení po připojení pomocí USB viditelné v Android Studiu v seznamu zařízení, napravo od konfigurace. Pokud se zde zařízení nenachází, je možné pokusit se problém vyřešit kliknutím na tlačítko *Troubleshoot Device Connections*. Po výběru zařízení je možné spustit aplikaci pomocí prvního tlačítka napravo od seznamu zařízení. Proces je ilustrován na následujícím obrázku B.3.

### Spuštění na virtuálním stroji

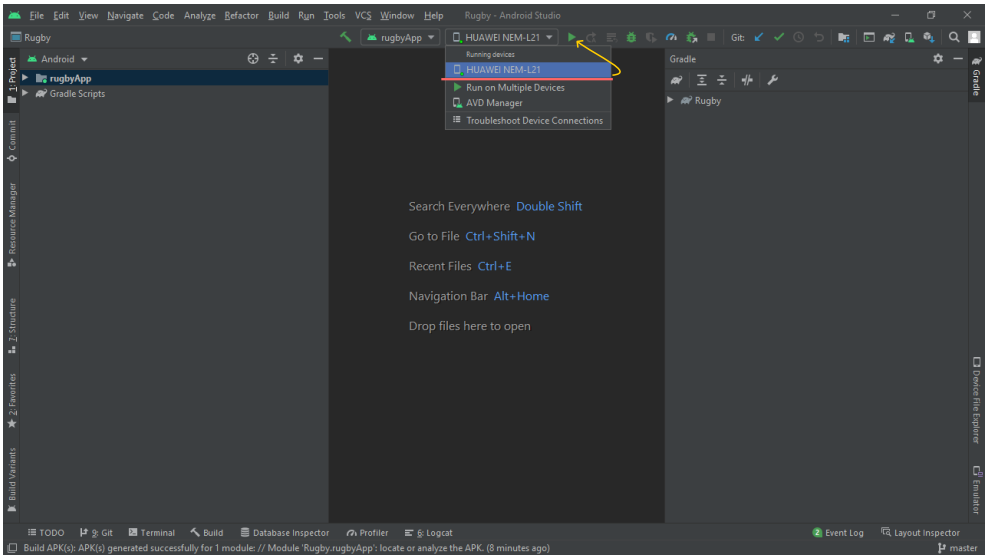
Pokud bylo nainstalováno AVD, je ze seznamu zařízení možné vytvořit nový virtuální stroj. Při vytvoření je možné zvolit typ zařízení (vhodné je zvolit mobilní zařízení) a obraz systému. Jednotlivé obrazy je nutné nejdříve stáhnout. Pro systém je třeba zvolit alespoň Android 5.0. Vytvořený virtuální stroj je poté možné vybrat ze seznamu zařízení a spustit aplikaci stejným způsobem jako u fyzického zařízení.




Obrázek B.1: Podoba po importování Android projektu




Obrázek B.2: Postup vytvoření vlastního apk souboru



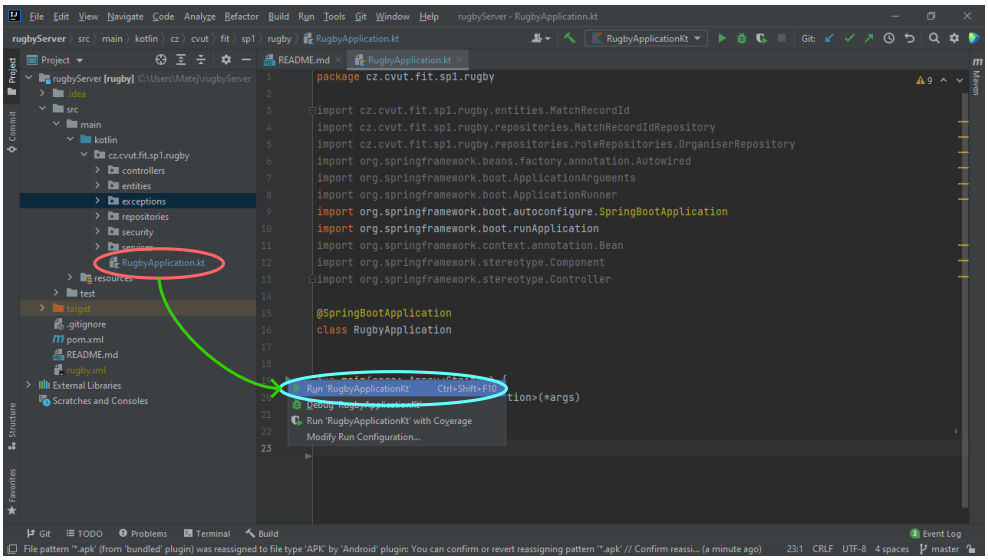

Obrázek B.3: Postup spuštění na připojeném zařízení

## Serverová část

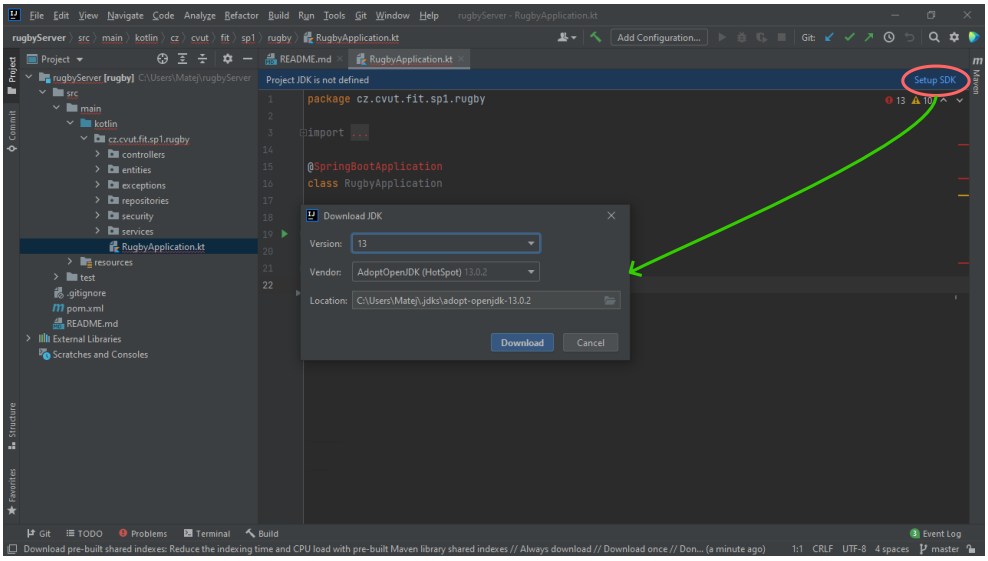
Pro spuštění samotného serveru je třeba sestavení ze zdrojového kódu. Pro práci se zdrojovým kódem je doporučeno použít vývojářské prostředí *IntelliJ IDEA*. Pro tyto účely plně postačuje verze Community, která je oficiálně volně dostupná na následujícím zdroji [24]. Další postup bude popisován podle verze 2021.1.

Během instalace není třeba věnovat pozornost žádnému speciálnímu nastavení. Po spuštění je možné přímo otevřít složku se zdrojovým kódem serveru (*rugbyServer*) pomocí možnosti *Open*. Automaticky by mělo dojít ke konfiguraci projektu. Po skončení procesu je možné server spustit a vypnout podle následujícího obrázku B.4.

Předtím bude ale s největší pravděpodobností nutné pro projekt nastavit vývojovou sadu Javy. Tuto možnost by mělo vývojové prostředí samo nabídnout při prohlížení zdrojového kódu. Podle následujícího obrázku B.5 je třeba stáhnout požadovanou verzi JDK, nebo projektu přiřadit již nainstalovanou verzi. Pro tento projekt je doporučeno použít verzi JDK 13, pomocí které byl vyvíjen. Po přiřazení JDK by již mělo být možné server spustit.




Obrázek B.4: Spuštění serveru




Obrázek B.5: Přiřazení JDK k projektu

Příloha C

# Uživatelská příručka

## Připojení k serveru

Pro vlastní použití je vhodné nainstalovat a spustit vlastní server a v aplikaci se připojit k němu. Aplikace se automaticky připojuje k vlastnímu serveru, je však možné manuálně nastavit vlastní IP adresu, ke které se aplikace pokusí připojit. Konkrétně je možné adresu nastavit z hlavního menu pomocí tlačítka na hlavním panelu v pravém horním rohu C.1.

## Přihlášení

Před použitím aplikace je třeba se přihlásit k uživatelskému účtu. Role přihlášeného účtu potom ovlivňují, jaké možnosti má uživatel k dispozici.

Při použité vlastního serveru jsou dostupné uživatelské účty dostupné uživatelské účty a jejich role definované ve výše zmíněném souboru *data.sql*. Mezi nejdůležitější účty v kontextu této práce patří:

ORGANISER, s rolí pořadatele

COACH, s rolí trenéra

REFEREE, s rolí rozhodčího

ALL_ROLES, se všemi rolemi v systému

Ke každému z účtů je možné se přihlásit pomocí jejich uživatelského jména a stejným heslem „password“.

## Použití aplikace

Použití funkcí implementovaných v aplikaci je popsáno v sekci návrhu uživatelského rozhraní 3.5. Aplikace obsahuje také obsah implementovaný v rámci


Obrázek C.1: Nastavení IP adresy

dalších rozšíření, popsaných v úvodu. Použití těchto částí je blíže popsáno v příslušných pracích.

Příloha D

# Testovací scénáře

Podle následujících scénářů probíhalo uživatelské testování této části aplikace.

**Přihlášení**

- Odhlaste se z aktuálně přihlášeného účtu
- Přihlaste se k účtu pořadatele, s uživatelským jménem „poradatel“ a heslem „password“

**Vytvoření turnaje**

- Vytvořte nový turnaj s následujícími informacemi
  - Název turnaje je „Turnaj“
  - Adresu zvolte Karlova 4, Praha, PSČ 1000
  - Turnaj bude začínat v aktuální čas a bude končit zítra ve stejnou hodinu
  - Turnaj bude pořádaný pro kategorie U6 a U8

**Přiřazení hřišť turnaji**

- Nově vytvořenému turnaji přidejte jedno hřiště pro kategorii U6 a jedno hřiště pro kategorii U8, s popisy „U6“ a „U8“
- Přiřaďte ke všem hřištím rozhodčího s přihlašovacím jménem „rozhodci1“

**Pozvání klubů a nominace týmů**

- První zařízení
  - Na nově vytvořený turnaj pozvěte kluby „Újezd“ a „Kostomlaty“
  - Po pozvání stáhněte pozvánku klubu „Kostomlaty“

  - Místo pozvánek nově vytvořený turnaj zveřejněte
- Druhé zařízení
  - Přihlaste se jako trenér klubu „Újezd“, uživatelské jméno „trener“, heslo „password“
  - Nominujte na nově vytvořený turnaj týmy „Újezd 1“, „Újezd 2“, „Újezd 3“, „Újezd 4“

**Zvolení hrajících týmů**

- Jako pořadatel pro nově vytvořený turnaj vyberte všechny nominované týmy, jako hrající týmy na turnaji a tuto volbu potvrďte

**Vytvoření harmonogramu**

- Pro nově vytvořený turnaj nechte aplikací automaticky vytvořit zápasy
- Pro automatické vytvoření zvolte následující parametry
  - Jako formát vyberte „Každý s každým“
  - Bude se hrát 1 kolo
  - Pro kategorii U6
  - Čas zahájení bude v čas zahájení turnaje
  - Délka zápasů bude 10 minut, mezi koly by měli být časové prodlevy 15 minut

**Zobrazení zápasů týmů**

- Na druhém zařízení, kde jste přihlášený jako trenér, si zobrazte zápasy svých týmů
- Zkontrolujte, zdali jsou všechny vytvořené zápasy zobrazeny

**Zobrazení zápasů rozhodčího**

- Na druhém zařízení se přihlašte jako jeden z rozhodčích, uživatelským jménem „rozhodci1“, heslo „password“
- Zobrazte si zápasy, na které je rozhodčí přiřazen a zkontrolujte, zdali jsou viditelné zápasy turnaje

**Odebrání týmu**

- Před začátkem turnaje jste zjistili, že tým „Újezd 4“ se nebude moci dostavit
- Odeberte tento tým z hermonogramu

**Zaslání notifikace**

- Na prvním zařízení vyberte jeden z prvních zápasů, a nastavte pro něj odeslání notifikace o zahájení
- Na druhém zařízení, kde jste nyní přihlášení jako rozhodčí zápasu, odpovězte na odeslanou notifikaci
- Na prvním zařízení si zobrazte odpovědi na odeslanou notifikaci

**Zapisování hřiště**

- Z vytvořených zápasů vyberte zápas „Újezd 1“ proti „Újezd 2“
- Zobrazte si zapisovací kód pro vybraný zápas
- Na druhém zařízení vložte zobrazený zapisovací kód pro zapisování zápasu

**Zaznamenávání výsledků**

- Začnete zapisovat zápas
- Odstartujte časomíru zápasu a přidejte 5 bodů týmu „Újezd 1“ a 15 bodů týmu „Újezd 2“
- Dopustili jste se chyby, tým „Újezd 2“ by měl mít pouze 10 bodů, tuto chybu opravte
- Výsledky odešlete
- Na prvním zařízení v seznamu zápasů zkontrolujte zaznamenání výsledků

**Zapisování hřiště**

- Vyberte hřiště „U6“ a zobrazte kód pro jeho zapisování
- Na druhém zařízení opět zadejte zapisovací kód
- Zapiště první ze zápasů na hřišti se s výsledkem 5:5
- Představte si, že po zápisu musíte odejít a chcete předat zapisování dalšímu zapisovateli
  - Nechte si zobrazit kód pro předání hřiště
  - Na prvním zařízení teď budete vystupovat v roli následujícího zapisovatele
  - Zadejte zobrazený předávací kód
  - Zapiště zbývající zápas

Příloha E

# Obsah přiloženého média

- `readme.txt` ..... popis obsahu
- `license.txt` ..... licence použitého kódu třetích stran
- `apk`
  - `rugby.apk` ..... APK soubor Android aplikace
- `rugby-android` ..... zdrojový kód Android aplikace
- `rugby-server` ..... zdrojový kód serverové části
- `text`
  - `src` ..... zdrojový kód práce ve formátu LaTeX
  - `text.pdf` ..... text práce ve formátu PDF