TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

SW řešení pro organizaci a správu badmintonových turnajů

Diplomová práce

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

Autor práce: **Bc. Michal Čapek**
Vedoucí práce: Ing. Igor Kopetschke

Liberec 2019

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

SW solutions for badminton tournaments organization and management

Master thesis

Study programme: N2612 – Electrotechnology and informatics
Study branch: 1802T007 – Information technology

Author: **Bc. Michal Čapek**
Supervisor: Ing. Igor Kopetschke

Liberec 2019

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání diplomové práce

SW řešení pro organizaci a správu badmintonových turnajů

Jméno a příjmení: **Bc. Michal Čapek**
Osobní číslo: M17000124
Studijní program: N2612 Elektrotechnika a informatika
Studijní obor: Informační technologie
Zadávající katedra: Ústav nových technologií a aplikované informatiky
Akademický rok: **2018/2019**

Zásady pro vypracování:

1. Proveďte analýzu aktuálně používaných softwarových řešení pro organizaci a správu turnajů v badmintonu.
2. Definujte sadu nezbytných funkcionalit pro Vaše řešení a to jak z pohledu správy (backend), tak z pohledu klientského rozhraní.
3. Navrhněte a implementujte mobilní klientskou aplikaci pro operační systém Android za použití NoSQL databáze Google Firebase.
4. Navrhněte a implementujte backendovou část řešení pro správu turnajů ve formě webové aplikace.
5. Otestujte výsledný systém v reálném nasazení. a pokud to bude možné, získejte zpětnou vazbu, případně navrhněte další rozšíření a vylepšení systému.

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

Seznam odborné literatury:

[1] LACKO, Ľuboslav. Vývoj aplikací pro Android. Brno: Computer Press, 2015. ISBN 978-80-251-4347-6.
[2] Documentation for app developers [online]. 2018 [cit. 2018-10-01]. Dostupné z: https://developer.android.com/docs/.
[3] ŽÁRA, Ondřej. JavaScript: programátorské techniky a webové technologie. Brno: Computer Press, 2015. ISBN 978-80-251-4573-9.

Vedoucí práce:	Ing. Igor Kopetschke Ústav nových technologií a aplikované informatiky
Datum zadání práce:	18. října 2018
Předpokládaný termín odevzdání:	30. dubna 2019

L. S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

Ing. Josef Novák, Ph.D.
vedoucí ústavu

V Liberci 18. října 2018

Prohlášení

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

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

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

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

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

Datum:

Podpis:

Abstrakt

Cílem práce je navrhnout a implementovat komplexní softwarové řešení pro správu a organizaci turnajů v badmintonu. V práci je provedena analýza aktuálně používaných softwarových řešení v prostředí organizace a správy badmintonových turnajů. Dále je definována struktura softwarového řešení a sada funkcionalit nezbytných pro toto řešení. Navržené softwarové řešení se skládá z klientské mobilní aplikace a administrační webové aplikace. V práci je popsána implementace obou těchto aplikací. V závěru je pak popsán systém testování aplikace včetně použitých nástrojů a také systém získávání zpětné vazby od uživatelů.

Klíčová slova: Android, badminton, mobilní aplikace, ReactJS, webová aplikace.

Abstract

The thesis objective is to design and implement a complete software solution for administrating and organizing badminton tournaments. The thesis presents an analysis of currently used software solutions within the framework of organization and administration of badminton tournaments. It further on introduces a defined structure of software solutions and a set of functionalities needed for this solution. The designed software solution consists of a client mobile application and administrative web-based application. The thesis illustrates the implementation of both of these applications. At the end this thesis presents the system of testing of these applications, including all the tools and means used as well as the system of getting the feedback from the users.

Keywords: Android, badminton, mobile application, ReactJS, web application.

Poděkování

Rád bych na tomto místě poděkoval vedoucímu mé diplomové práce, panu Ing. Igoru Kopetschkemu, za čas, který mi věnoval.

Obsah

Seznam zkratek

BUSK	Badminton Umpire Score Keeper
ČBaS	Český badmintonový svaz
FM	Fakulta mechatroniky, informatiky a mezioborových studií Technické univerzity v Liberci
ID	Identifikátor
KR	Komise rozhodčích, orgán ČBaS
POJO	Plain Old Java Object
SDK	Software development kit
TLS	Transport Layer Security
TP	Tournament Planner
TUL	Technická univerzita v Liberci
UC	UseCase - případ užití
UI	User Interface - uživatelské rozhraní

Seznam obrázků

Seznam tabulek

1 Úvod

Aplikace pro chytré mobilní telefony jsou jednoznačným fenoménem dneška. Na Google Play, oficiálním obchodě s aplikacemi pro operační systém Android, je podle [3] v současné době více než dva a půl milionu aplikací. Za posledních 6 let se jejich počet zdvojnásobil [13]. Aplikace pronikají do všech odvětví našeho každodenního života a sport rozhodně není výjimkou. Podle [4] je na Google Play k dispozici více než 45 tisíc různých aplikací, které jsou zařazeny do kategorie Sport. Nejstahovanější z nich mají více než 50 milionů stažení. Uživatelé mají možnost v těchto aplikacích kromě jiného sledovat výsledky sportovních akcí z celého světa a mít tak přehled o výkonech svých oblíbených sportovců a týmů.

Cílem této diplomové práce je navrhnout a implementovat komplexní softwarové řešení pro správu a organizaci turnajů v badmintonu. Motivací pro vytvoření tohoto softwarového řešení mi bylo mé dlouhodobé působení v realizačním týmu profesionálního badmintonového klubu. Často také zastávám roli vrchního rozhodčího na republikových turnajích a právě z této pozice vidím velký prostor pro další digitalizaci a zautomatizování procesu sběru a prezentace výsledků zapasů a turnajů. V současnosti v prostředí badmintonu aplikace tohoto typu chybí. Její vytvoření může pomoci nejen s propagací badmintonu mezi laickou veřejností, ale i optimalizovat procesy v rámci Českého badmintonového svazu a rozšířit hráčskou základnu. V první části práce je provedena analýza současně používaných softwarových řešení v českém badmintonovém prostředí. Celé navrhované řešení se sestává ze dvou částí - mobilní aplikace pro chytré telefony s operačním systémem Android a webové aplikace.

Mobilní aplikace slouží dvěma cílovým skupinám. První cílovou skupinou jsou fanoušci - laická veřejnost. Těmto uživatelům aplikace přináší možnost sledovat online výsledky z turnajů družstev, získávat aktuální informace o dění na badmintonové scéně a informovat se o průběhu celé sezóny. Druhou cílovou skupinou jsou hlavní rozhodčí. Těm aplikace slouží pro zaznamenávání výsledků zápasů. Tyto výsledky jsou v reálném čase přenášeny do databáze a dále zpracovávány. Cílem práce je stanovit sadu funkcionalit, kterou mobilní aplikace bude uživatelům nabízet, a dále pak tyto specifikované funkcionality implementovat. Neméně důležité je pak implementovat přívětivé uživatelské rozhraní.

Webová aplikace je navržena jako administrativní rozhraní a implementována za použití knihovny ReactJS. Slouží úzké skupině uživatelů (vrchním rozhodčím)

k vytváření a správě obsahu zobrazovaného v mobilní aplikaci. V práci jsou specifikovány potřebné funkcionality pro tuto aplikaci a také popsána jejich implementace. Součástí implementace je pak i návrh uživatelského rozhraní, ke kterému byla využita knihovna react-bootstrap. Celý systém je postaven nad platformou Firebase od firmy Google. V práci jsou popsány její funkcionality, které byly při implementaci použity.

V poslední části práce je popsán systém testování aplikace, postup sběru zpětné vazby a navrženy další možnosti budoucího rozšíření systému.

2 Vymezení pojmů a analýza současného stavu

Jak již z názvu této diplomové práce vyplývá, celé softwarové řešení bude zaměřeno na badminton. V České republice není profesionální badminton až tolik populární jako fotbal nebo hokej, a tak by si někdo mohl myslet, že vytvářet softwarové řešení pro takový sport je zbytečné. Opak je ale pravdou. V následujících kapitolách bude badminton představen jako moderní sport s bohatou historií a také velkými ambicemi do budoucna. Dále pak budou vymezeny základní pojmy, které jsou s badmintonem spjaty, a které je nutné znát ke správnému pochopení fungování navrhovaného softwarového řešení.

2.1 Badminton jako olympijský sport

Předchůdce nynějšího badmintonu má své kořeny hluboko v historii. Už před 2 000 lety se v kultuře jihoamerických Inků a středoamerických Aztéků setkáváme s náznaky hry s „opeřeným míčkem“. V 7. století našeho letopočtu se v Číně hovoří o hře „Di–Dšen–Dsi“, kdy míček je odbíjen rukou nebo nohou. V Japonskuse mluví o hře „Cibane“ (14. století), v Indii o hře „Poona“. Ve Francii v 16. století to byla hra zvaná „Jeu Volant“. Za přímého předchůdce dnes považujeme indickou hru „Poona“, kterou do Evropy přivezl koncem 19. století (1872) anglický důstojník, vévoda z Beaufortu. Mezinárodní badmintonová federace (WBF) vznikla v roce 1934, dnes sdružuje 156 národních svazů a badminton se hraje na všech pěti kontinentech. Evropská badmintonová unie (EBU) vznikla ve Frankfurtu nad Mohanem v roce 1967 a jedním ze zakládajících členů bylo také tehdejší Československo. Největším impulzem v novodobé historii bylo zařazení badmintonu do programu LOH. Svou premiéru zažil badminton na OH v Barceloně v roce 1992. Ke světovým velmocím patří asijské státy: Čína, Malajsie, Indonésie a Jižní Korea.V Evropě se badminton těší největší oblibě v Dánsku, Německu, Švédsku, Anglii a Holandsku.[12]

2.1.1 Pravidla badmintonu

Pro správné pochopení funkcionalit navrhovaného softwarového řešení je nutné se seznámit alespoň se základními pravidly badmintonu. Badminton je individuální raketový sport. K zápasu nastupují vždy dva protivníci (nebo dvě dvojice), kteří sehrají zápas na dva vítězné sety do 21 bodů. Hráči svými raketami odpalují opeřený

míček a snaží se přinutit soupeře chybovat. Chybou je zahrání míčku mimo kurt, nebo neodehrání míčku, který soupeř odehrál správně. Na průběh zápasu vždy dohlíží tzv. hlavní rozhodčí (umpire). Pokud se jedná o zápas ve vyšší lize či na vyšším turnaji, jsou na daný zápas delegováni ještě tzv. čároví rozhodčí, kteří pomáhají hlavnímu rozhodčímu s rozhodováním o dopadu míčku. Hlavní rozhodčí má na kurtě nejvyšší slovo a jeho rozhodnutí musí být respektováno každým z hráčů. Hlavní rozhodčí má také na starost zapisování stavu a po skončení zápasu má povinnost výsledek nahlásit vrchnímu rozhodčímu.

V badmintonu existuje celkem 5 disciplín - dvouhra mužů a žen, čtyřhra mužů a žen a smíšená čtyřhra (často označovaná jako mix).

Badmintonové zápasy se odehrávají v rámci tzv. turnajů. Pod pojmem turnaj se rozumí seskupení badmintonových zápasů. Existují turnaje jednotlivců a turnaje družstev. Turnaje jednotlivců se většinou odehrávají během krátkého časového intervalu, nejčastěji během tří dní. Oblastní turnaje jsou zpravidla jednodenní. Na turnajích jednotlivců se dle pravidel odehrává všech 5 disciplín, každá disciplína má na konci svého vítěze. Turnaj řídí vrchní rozhodčí, který musí mít platnou licenci. Turnaje jednotlivců jsou dále rozděleny do věkových kategorií a také do kategorií dle úrovně. Rozdělení se řídí platnými předpisy ČBaS.

Turnaje družstev jsou od turnajů jednotlivců odlišné. Hráči zde nenastupují za sebe, nýbrž za tým. Existují speciální typy turnajů družstev, jako je například Mistrovství Evropy družstev, které mají svá vlastní pravidla. Nejznámějšími turnaji družstev jsou ale bezesporu ligy. V České republice je nejvyšší ligou Extraliga smíšených družstev, nižší celostátní ligou je 1. liga, a dále pak regionální 2. liga a 3. liga. Všechny tyto ligy se řídí pravidly, které stanovuje Soutěžní řád vydávaný ČBaS. Ligy probíhají v rámci delšího časového období, zpravidla od září do dubna. Tato práce je zaměřena na nejvyšší ligu - tedy Extraligu smíšených družstev. Navržené softwarové řešení je ale natolik komplexní a variabilní, že není problém ho použít i pro další ligy.

2.2 Badminton v ČR

Profesionální badminton zastřešuje Český badmintonový svaz (ČBaS), který má své stanovy, orgány, atd. Tématu této práce se rozhodně nejvíce týká orgán zvaný Komise rozhodčích (KR), který má na starost nejen zajištění hladkého průběhu všech pořádaných badmintonových turnajů, ale i školení nových hlavních a vrchních rozhodčích a také dohled na jejich nestrannost.

2.2.1 Extraliga smíšených družstev dospělých a její organizace

Nejvyšší tuzemskou celostátní ligou družstev dospělých je Extraliga smíšených družstev dospělých (dále jen Extraliga). Startuje v ní 8 nejlepších týmů z České republiky. Extraliga se dělí na základní část a na play-off. V základní části sehraje

každý přihlášený tým soutěžní utkání s každým ze zbývajících týmů. Za výhru v utkání získá tým do tabulky 4 body, za výhru v prodloužení 3, za prohru v prodloužení 2 a za prohru 1 bod. Po odehrání všech utkání základní části se dva nejvýše umístěné týmy kvalifikují do semifinále play-off, týmy na třetím až šestém místě se kvalifikují do čtvrtfinále play-off a poslední 2 týmy musí bojovat o udržení se v nejvyšší soutěží v utkáních play-out. Semifinále a finále probíhají v rámci jednoho finálového víkendu - v sobotu se odehrají semifinálová utkání a v neděli utkání o 3. místo a finále. Vítěz finále je prohlášen mistrem republiky smíšených družstev.

Každé utkání, které je v rámci ligy odehráno, má svou jasně definovanou strukturu. Rozpis soutěží [6] a Soutěžní řád [8] stanoví na začátku sezóny hrací dny a místa pro jednotlivá utkání. Každé utkání se pak skládá z přesně definovaných zápasů - jejich přehled je v tabulce 2.1.

Zkratka	Název
1. MS	1. dvouhra mužů
2. MS	2. dvouhra mužů
3. MS	3. dvouhra mužů
1. WS	1. dvouhra žen
1. MD	1. čtyřhra mužů
1. WD	1. čtyřhra žen
2. MD	2. dvouhra mužů
XD	smíšená čtyřhra

Tabulka 2.1: Zápasy v utkání

Za každý vyhraný zápas získá tým 1 bod. Při rovnosti bodů po odehrání všech osmi zápasů se hraje ještě prodloužení, neboli tzv. ”zlatý zápas” (buď MS, WS, nebo XD), který rozhodne o vítězi utkání a rozdělení bodů do tabulky ligy.

Na každé utkání deleguje KR (po domluvě s pořádajícím klubem) vrchního rozhodčího. Vrchní rozhodčí zodpovídá za průběh utkání, spravuje zápis o utkání, deleguje hlavní rozhodčí na jednotlivé zápasy a zapisuje výsledky zápasů. Určuje také pořadí, v jakém budou jednotlivé zápasy odehrány.

2.3 Současný stav zaznamenávání a prezentace výsledků

Ještě před několika lety bylo naprosto běžné, že vrchní rozhodčí turnaje řídil s tužkou a papírem v ruce, výsledky zaznamenával ručně do papírových ”pavouků” a hlavní rozhodčí na umpirech otáčeli čísla na papírových počítadlech skóre. Postupnou modernizací se stav zlepšil natolik, že vrchní rozhodčí musí na turnajích používat automatizovaný losovací program a program na správu výsledků turnajů. Používání tohoto programu je zakontveno v Rozpisu soutěží [7]. Tento komerční software

jménem Tournament planner ČBaS nakupuje v podobě ročních licencí. Software však trpí několika nedostatky, které jsou rozebrány v dalších kapitolách. Na větších turnajích (jako je třeba Mistrovství ČR, Mezinárodní mistrovství ČR nebo finále Extraligy) jsou organizátoři nuceni zajistit digitální ukazatel skóre na jednotlivých kurtech. Tyto ukazatele jsou však dostupné pouze ve velmi omezeném množství a jejich pořízení je pro kluby finančně náročné. Navíc se jedná o zcela jednoduchá zařízení, která nemají žádné možnosti integrace do dalších systémů a ani není možné je rozšířit.

2.3.1 Tournamentsoftware a Tournament Planner

Tournamentsoftware a Tournament Planner jsou komerční software od Nizozemské firmy Visual Reality B.V., který ČBaS nakupuje pro zjednodušení a digitalizaci zaznamenávání výsledků z badmintonových turnajů. Podrobné informace o tomto software jsou k dispozici na webových stránkách www.tournamentsoftware.com. Pojmem tournamentsoftware se rozumí webová aplikace, která slouží pro zobrazení výsledků jednotlivých turnajů. Tournament Planner je desktopová aplikace pro systém Windows, která slouží pro offline správu turnaje a následnou publikaci online na zmíněném webu. Přístupy do obou těchto svázaných software jsou pod jednou licencí. Tournament Planner tedy slouží vrchním rozhodčím turnajů k přípravě startovací listiny, losování ”pavouků”, zaznamenávání výsledků jednotlivých zápasů a ke spoustě dalších činností, které s řízením turnaje souvisí. Jedná se o velice komplexní software, který disponuje obrovskou sadou funkcionalit.

Mezi hlavní výhody patří:

- velké množství podporovaných sportů,
- software používaný velkým množstvím organizací a svazů,
- mezinárodní podpora,
- propracované zobrazování obsahu na webu.

Toto softwarové řešení ale není (právě z důvodu podpory velkého množství různých sportů) zcela optimalizováno pro zaznamenávání výsledků badmintonové Extraligy. Největším problém je datová struktura Tournament Planneru. Každý turnaj, který má být vidět na webu, musí být založen v desktopové aplikaci. Ta pro turnaj vytvoří tzv. TP soubor, což je specifický soubor pro použití právě v desktopové aplikaci. Tento TP soubor se pak uploaduje na server. Aplikace ale neumožňuje z webu TP soubor stahovat, a to je pro použití v Extralize největší problém. Tím totiž není dovoleno zapisovat do stejného turnaje z více míst v jeden čas najednou, což je základní požadavek (utkání v rámci jednotlivých kol se hrají na různých hracích místech souběžně). Není tak možné sdílet výsledky z jednotlivých hracích míst online do jednoho digitálního systému.

2.3.2 Průběh zveřejnění výsledků

Z výše uvedých důvodů musí zveřejňování výsledků probíhat následujícím způsobem:

1. Na začátku sezóny ČBaS stanoví datum a čas konání jednotlivých utkání a založí TP soubor pro aktuální sezónu.
2. Před každým kolem deleguje vrchní rozhodčí pro každé z utkání.
3. Vrchní rozhodčí v průběhu utkání zapisuje výsledky jednotlivých zápasů do tzv. Zápisu o utkání (viz. Obrázek 2.1), což je ve skutečnosti tabulka v MS Excel.
4. Po skončení utkání odešle vrchní rozhodčí tuto tabulku na ČBaS.
5. ČBaS počká, až dostane tabulky ze všech hracích míst, pak upraví TP soubor a uploaduje ho na server. Zároveň publikuje vyplněné tabulky na svém webu. Takto musí ČBaS učinit po každém hracím kole.

ZÁPIS O UTKÁNÍ SMÍŠENÝCH DRUŽSTEV

Název soutěže:												
Družstvo "A"								Datum:				
Družstvo "B"								Místo:				
Vrchní rozhodčí:												x kolo v turnaji
	"A"	"B"	Výsledky setů			Součet míčů		Sety		Body		Rozhodčí
			1	2	3							
1.dvouhra mužů			:	:	:	0	0	0	0	0	0	
2.dvouhra mužů			:	:	:	0	0	0	0	0	0	
3.dvouhra mužů			:	:	:	0	0	0	0	0	0	
dvouhra žen			:	:	:	0	0	0	0	0	0	
1.čtyřhra mužů			:	:	:	0	0	0	0	0	0	
čtyřhra žen			:	:	:	0	0	0	0	0	0	
2.čtyřhra mužů			:	:	:	0	0	0	0	0	0	
smíšená čtyřhra			:	:	:	0	0	0	0	0	0	
DMxDŽxXD			:	:	:	0	0	0	0	0	0	
VÍTĚZ:						**0**	**0**	**0**	**0**	**0**	**0**	

Podpis vrchního rozhodčího

Potvrzujeme, že utkání bylo sehráno podle platných pravidel a soutěžního řádu.

Námitky: .....

Podpis vedoucího družstva "A": ..... Podpis vedoucího družstva "B": .....

Obrázek 2.1: Vzor zápisu o utkání

Z výše popsaného postupu je zřejmé, že stávající postup trpí mnoha nedostatky. Těmi jsou především:

- obrovská režie;
- nemožnost jakéhokoliv online sdílení aktuálního stavu jednotlivých utkání;
- nemožnost napojení jiných systémů, které by umožnily digitalizaci procesu zaznamenávání skóre jednotlivých zápasů a jeho sdílení;
- časové prodlevy mezi ukončením utkání a zveřejněním výsledků.

Navrhovaný systém musí všechny tyto nedostatky řešit. Zvláštní důraz je kladen na sdílení dat v reálném čase tak, aby byl zajištěn co nejvyšší komfort pro koncového uživatele.

2.4 Existující konkurenční řešení

Na Google Play (obchod s aplikacemi pro zařízení s operačním systémem Android) je v současné době ke stažení aplikace ”Badminton Umpire Score Keeper” (dále jen BUSK), a to jak v neplacené verzi, tak ve verzi ”Pro”, která je zpoplatněná. V rámci průzkumu v současnosti používaných řešení bylo zjištěno, že v jednom z českých badmintonových klubů je tato aplikace aktivně využívána pro zaznamenávání a zobrazování skóre zápasů. Níže jsou popsány zjištěné výhody a nevýhody této aplikace, která může být označena jako konkurenční k navrhovanému řešení.

2.4.1 Výhody

Po provedené analýze by se výhody aplikace BUSK daly shrnout do následujících bodů:

- Aplikace nabízí nejen hotovou aplikaci pro Android, ale ve verzi Pro i aplikaci pro Windows, která po propojení dokáže zobrazit data z mobilní aplikace v podobě, která se dá např. promítnout na velkoplošné obrazovce.
- Na Google Play má více než 10000 stažení
- Uživatel mobilní aplikace má velké možnosti nastavení pro jednotlivé zápasy. Sám si může zvolit, zda se jedná o dvouhru, či čtyřhru, sám zadává jména hráčů a upravuje podávající stranu.
- Aplikace nabízí pro každý odehraný set přehledný graf s posloupností získávání bodů jednotlivými hráči.

2.4.2 Nevýhody

Aplikace ovšem oplývá mnoha nevýhodami. Mezi ně patří zejména níže uvedené body.

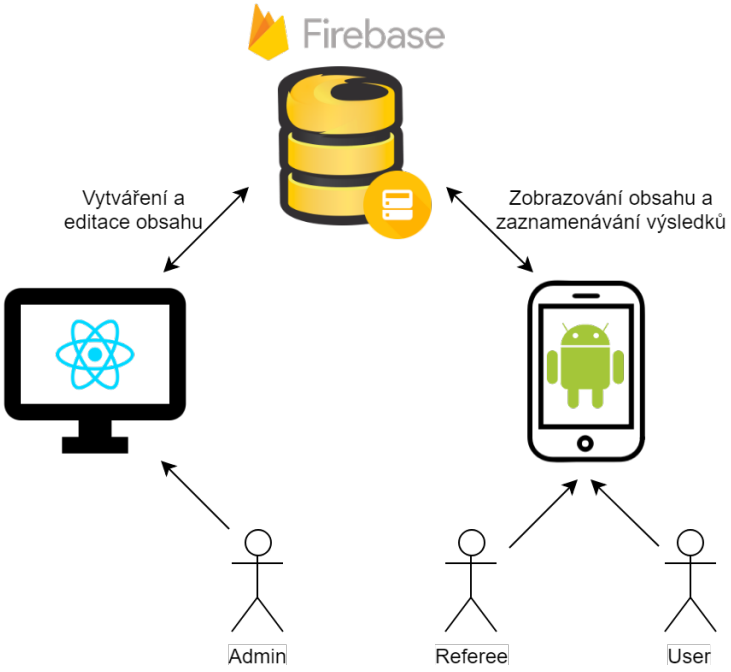
- Verze Pro je zpoplatněná. Licence jsou vázané na Google účet uživatele - tedy je nutné pro každého hlavního rozhodčího koupit jednu kopii.
- Ve volně přístupné verzi neumí mobilní aplikace komunikovat s aplikací pro Windows.
- Aplikace pro Windows je dostupná pouze z neoficiálních zdrojů, což s sebou může nést bezpečnostní riziko.
- Celé řešení vyvinul neznámý vývojář, takže se nedá vysledovat, zda data neodchází někam, kde by mohla být zneužita.
- Řešení funguje pouze lokálně. Není zde žádná možnost odesílání výsledků na autorizovaný server, kde by docházelo například k ukládání, nebo dalšímu zpracování.
- Veškerý obsah, který se v aplikaci vytváří, je uložen pouze lokálně na mobilních zařízeních jednotlivých hlavních rozhodčích. Data nejsou nijak centralizována. Navíc veškerý obsah je závislý pouze na tom, jaké údaje hlavní rozhodčí zadá. Není zde žádná možnost přípravy dat dopředu tak, aby hlavní rozhodčí nemusel již na kurtu zadávat jména hráčů a podobně.
- Výsledky umí aplikace sdílet pouze jako exportované do formátu PDF. Ten je pro další zpracování nevhodný.

Celkově se dá o aplikaci říct, že nabízí funkcionality, které stačí hlavnímu rozhodčímu k řízení zápasů. Nicméně tato práce si klade za cíl vytvořit systém, který bude umět integrovat zaznamenávání výsledků do jednoho centrálního systému, delegovat registrované rozhodčí k již připraveným zápasům a sbírat výsledky ve strojově čitelné formě tak, aby bylo možné tyto výsledky dále zpracovávat.

3 Definice sady funkcionalit navrhovaného systému

Navrhovaný systém se sestává ze dvou základních komponent - webové aplikace a mobilní aplikace pro operační systém Android. Webová aplikace slouží především pro vytváření a správu obsahu a je určena pro úzký okruh uživatelů, u kterých se předpokládá, že ovládají základní práci s počítačem a webovými stránkami. Před začátkem používání aplikace budou všichni uživatelé proškoleni. Mobilní aplikace je navržena tak, aby ji mohla používat široká veřejnost. Obsahuje v sobě ale i skrytou část pro rozhodčí, která je dostupná po přihlášení a přidělení příslušných oprávnění. V této kapitole budou popsány případy užití (dále jen UC z anglického UseCase) pro jednotlivé komponenty navrhovaného systému a definována sada potřebných funkcionalit systému.

Na obrázku 3.1 je zobrazeno celkové schéma navrhovaného systému, včetně předpokládaného použití jednotlivých částí systému uživateli.



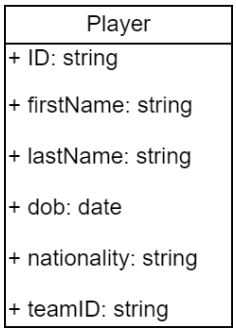
Obrázek 3.1: Celkové schéma navrhovaného systému

3.1 Datové entity

Nejprve je potřeba definovat, jaké datové entity se v aplikaci budou vyskytovat, jaká budou mít vzájemná propojení a jaké budou mít vlastnosti.

3.1.1 Hráč (Player)

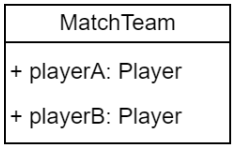
Datová struktura, která definuje hráče. Každý hráč je jednoznačně definován svým ID. Dále pak obsahuje vlastnosti jméno, příjmení, datum narození a národnost. Dále obahuje ID týmu, do kterého je zařazen.



Obrázek 3.2: Datová struktura - Player

3.1.2 Tým v zápase (MatchTeam)

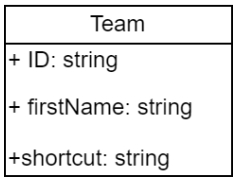
Protože k zápasům nastupují hráči buď sami (dvouhra), nebo s partnerem (čtyřhra, mix), je třeba to pro potřeby aplikace unifikovat. Datová struktura MatchTeam reprezentuje vždy jednoho z protivníků na kurtě. Skládá se ze dvou struktur Hráč. Pokud je zápas dvouhrou, je vyplněn pouze jeden z hráčů.



Obrázek 3.3: Datová struktura - MatchTeam

3.1.3 Tým (Team)

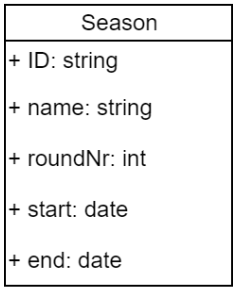
Datová struktura, která definuje tým ve smyslu soutěžního týmu. Každý tým je jednoznačně definován svým ID. Dále pak obsahuje vlastnosti jméno a zkratku.



Obrázek 3.4: Datová struktura - Team

3.1.4 Sezóna (Season)

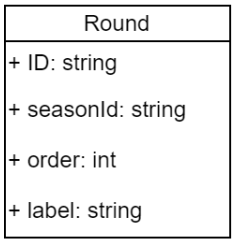
Datová struktura, která definuje sezónu. Sezóna je definovaná svým ID, které vychází z roku zahájení a z roku ukončení. Sezóna zastřešuje všechny další entity - ty jsou vždy přiřazeny ke konkrétní sezóně. Kromě ID obsahuje vlastnosti jméno, počet kol v sezóně, měsíc a rok zahájení a měsíc a rok ukončení. Ke každé sezóně je automaticky vytvořeno tolik kol, kolik uživatel zadá, plus čtvrtfinále, semifinále, finále a zápas o 3. místo.



Obrázek 3.5: Datová struktura - Season

3.1.5 Herní kolo (Round)

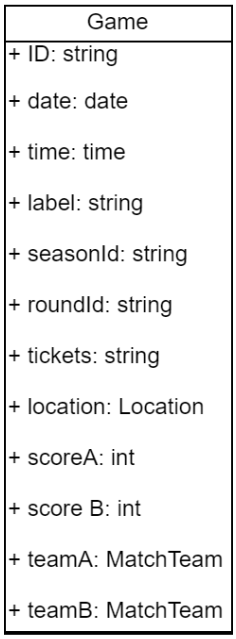
Datová struktura, která definuje herní kolo v sezóně. Kolo je definovaná svým ID, které vychází z ID sezóny, pod kterou je dané kolo přiřazeno, a z pořadí v této sezóně. Počet kol v sezóně je definován při vytváření sezóny a nelze změnit. Kolo jako takové slouží spíše k rozdělení jednotlivých utkání, proto obsahuje pouze ID sezóny, pod kterou spadá, svůj název a pořadí v sezóně.



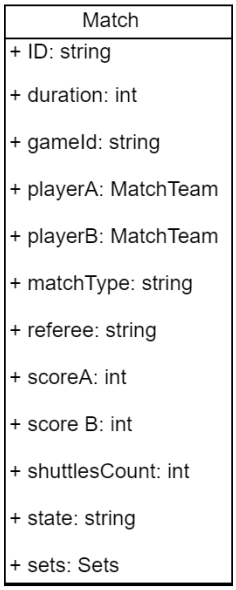
Obrázek 3.6: Datová struktura - Round

3.1.6 Utkání (Game)

Datová struktura, která definuje utkání mezi dvěma týmy v určitém kole. Každé utkání je jednoznačně definováno svým ID a dále zařazeno do sezóny pomocí `seasonId` a `roundId`. V rámci utkání jsou definováni soupeři - tedy dva týmy, které k utkání nastoupí. Týmy jsou určeny pomocí jejich ID. Dalšími duležitými vlastnostmi Utkání je čas a datum, kdy se dané kolo odehraje. Neméně důležité je i určení místa - tedy sportovní haly, kde je utkání sehráno. Pro diváky je pak důležitá i informace o ceně vstupenek. Výsledek utkání je pak uložen v polích scoreA a scoreB. Výsledné skóre utkání je určeno jako součet vyhraných zápasů jednotlivých týmů v daném utkání.



Obrázek 3.7: Datová struktura - Game



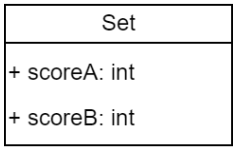
Obrázek 3.8: Datová struktura - Match

3.1.7 Zápas (Match)

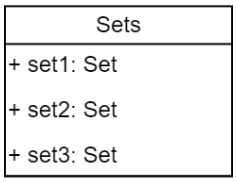
Zápas je základním stavebním kamenem celého řešení. Každý zápas je definován svým unikátním ID a přiřazen k utkání pomocí `gameId`. K zápasu nastupují vždy dva týmy - `playerA` a `playerB`. Tyto týmy jsou uloženy jako struktura `MatchTeam`. Výsledek jednotlivých setů zápasu je zaznamenáván do položky `sets`. Položky `scoreA` a `scoreB` pak informují o celkovém skóre zápasu. Typ zápasu je uložen v položce `matchType`. Zápas řídí hlavní rozhodčí, jehož ID je uloženo v položce `referee`. O zápase je zaznamenáváno i několik statistických informací - trvání zápasu a počet spotřebovaných míčků. Položka `state` potom vyjadřuje, v jakém stavu se zápas nachází. Stavy jsou definovány pomocí struktury `enum`.

3.1.8 Set a sety (Set, Sets)

Tyto dvě pomocné datové struktury umožňují strukturovaně ukládat výsledky z jednotlivých setů. Datová struktura Sets se skládá ze tří struktur Set (protože v jednom zápase je možné odehrát maximálně 3 sety). Struktura Set je pak jednoduchým úložištěm dvou číslených hodnot - bodů dosažených každým z protivníků.



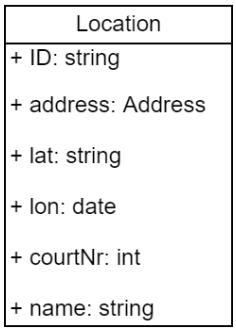
Obrázek 3.9: Datová struktura - Set



Obrázek 3.10: Datová struktura - Sets

3.1.9 Hrací místo (Location)

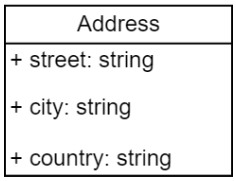
Datová struktura, která definuje místo, kde se odehrávají jednotlivá utkání. Každé hrací místo je jednoznačně definováno svým ID. Dalšími důležitými údaji jsou adresa (uložená jako pomocná datová struktura Address), GPS souřadnice, informace o počtu dostupných kurtů a název místa.



Obrázek 3.11: Datová struktura - Location

3.1.10 Adresa (Address)

Pomocná datová struktura, která slouží pro uložení adresy daného hracího místo. Skládá se z údajů o ulici, městě a státu, kde dané hrací místo leží.



Obrázek 3.12: Datová struktura - Address

3.2 Role a oprávnění uživatelů

Navrhovaný systém cílí na širokou uživatelskou základnu. Jeden systém budou používat jak uživatelé bez speciálních oprávnění (diváci, fanoušci), tak uživatelé s dalšími úrovněmi oprávnění (rozhodčí, administrátoři, superadministrátoři). Proto je nezbytné, aby navrhované řešení nabízelo systém, který zajistí autentifikaci uživatelů a dovolí jim přiřazovat jejich role.

3.2.1 Nepřihlášený uživatel (Anonymous)

Nepřihlášený uživatel je takový uživatel, který neposkytne aplikaci své přihlašovací údaje. Takovému uživateli aplikace musí nabídnout možnost si vytvořit svůj uživatelský účet a následně se do něj přihlásit. Po vytvoření uživatelského účtu je tomuto uživateli přiřazena automaticky role User. Nepřihlášený uživatel nemá žádná další oprávnění.

3.2.2 Uživatel (User)

Uživatel, který si vytvoří uživatelský účet, získá automaticky roli User. Přihlášením se do aplikace může uživatel získat personifikované informace, speciální nabídky, atd (více v kapitole 7.3). V první verzi aplikace přihlášený uživatel nemá žádná další speciální oprávnění.

3.2.3 Rozhodčí (Referee)

Rozhodčím se stane uživatel ve chvíli, kdy Superadministrátor ověří jeho účet a nastaví mu v systému roli Referee. Tím Superdministrátor dává najevo, že je Rozhodčí schopen vykonávat funkci hlavního rozhodčího (vychází se stanov ČBaS) a zároveň že souhlasí s tím, aby Rozhodčí tuto funkci vykonával. Rozhodčí tím získá speciální opravnění zapisovat výsledky z zápasům, ke kterým ho Superadministrátor přiřadí. Rozhodčí si může zobrazit seznam zápasů, které jsou k jeho osobě přiřazeny. Rozhodčí nezískává právo delegovat jiné rozhodčí, ani zapisovat výsledky k zápasům, ke kterým nebyl přiřazen.

3.2.4 Administrátor (Admin)

Administrátor má všechna práva, která ma Rozhodčí. Administrátorem se stává ten uživatel, kterému tuto roli v systému přiřadí Superadministrátor. Administrátor tímto získává všechna práva na zásahy do databáze systému, zejména pak vytváření, modifikaci a mazání obsahu a přiřazování Rozhodčích k zápasům. Administrátor přiřazením funkce dává najevo, že si je vědom všech rizik spojených s užívání aplikace, zvláště pak s nevratností mazání obsahu. Administrátorem jsou většinou jmenování všichni vrchní rozhodčí jednotlivých utkání.

3.2.5 Superadministrátor (Superadmin)

Superadministrátor má veškerá práva spojená s užíváním aplikace. Jeho hlavní výsadou je možnost definování a přiřazování rolí ostatním uživatelům aplikace. Superadministrátor má právo na přímé zásahy do databáze.

3.3 Webová aplikace

Webovou aplikaci budou používat především vrchní rozhodčí jednotlivých utkání. Aplikace by měla být navržena tak, aby její ovládání bylo co nejvíce intuitivní a nezatěžovala vrchní rozhodčí zbytečnou administrativou. Typickým uživatelem této aplikace je člověk, který perfektně zná pravidla badmintonu, ovládá práci s počítačem a má dobré organizační schopnosti.

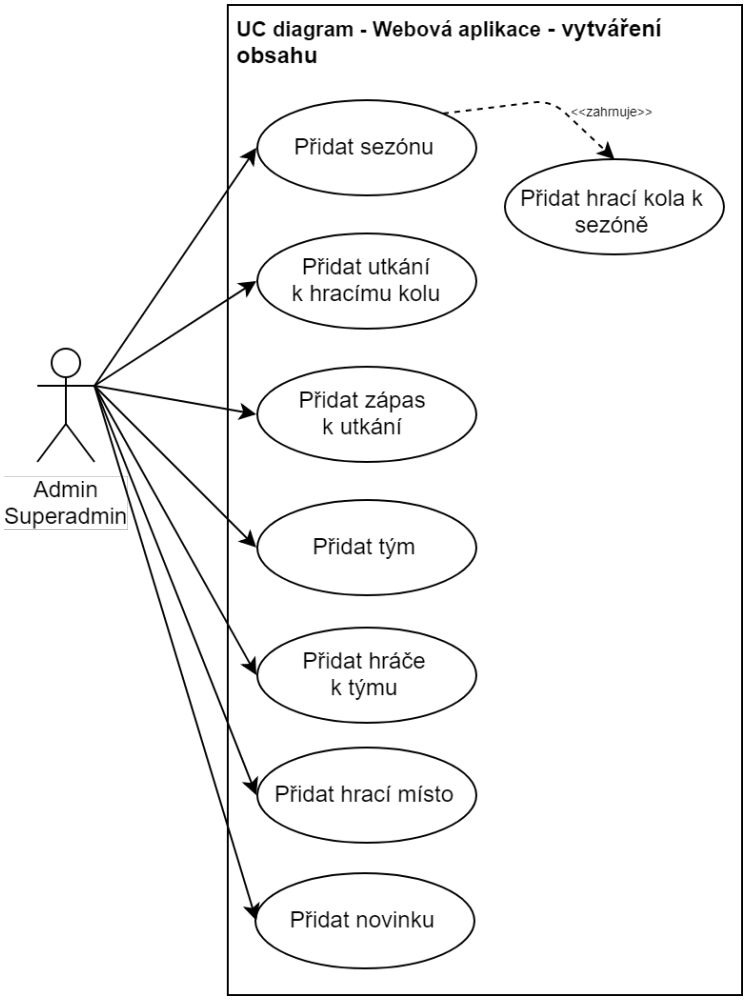
K používání webové aplikace je vyžadováno přihlášení. Bez přihlášení uživatel nemá možnost jakkoliv měnit obsah uložený v databázi. Nepřihlášeným uživatelům je zobrazena pouze sekce Novinky. Dále má takový uživatel možnost se přihlásit či zaregistrovat.

3.3.1 Vytváření a mazání obsahu

Základním požadavkem na webovou aplikaci je možnosti vytváření obsahu a jeho ukládání do databáze. Přihlášený uživatel, který má přiřazena práva ADMIN nebo SUPERADMIN, má možnost vytvářet a mazat následující entity:

- Hráč
- Tým
- Sezóna
- Utkání
- Zápas
- Hrací místo

Všechny datové entity, které mohou uživatelé dle obrázku 3.13 vytvořit, mohou také vymazat.



Obrázek 3.13: UC diagram - vytváření obsahu

3.3.2 Uživatelé bez speciálních oprávnění

Webová aplikace není primárně určená pro používání nepřihlášenými uživateli, nebo uživateli s oprávněním User. Tito uživatelé si po otevření aplikace ve webovém prohlížeči mohou zobrazit sekci Novinky. Nepřihlášeným uživatelům pak bude nabídnuta možnost se přihlásit, nebo si vytvořit nový uživatelský účet.

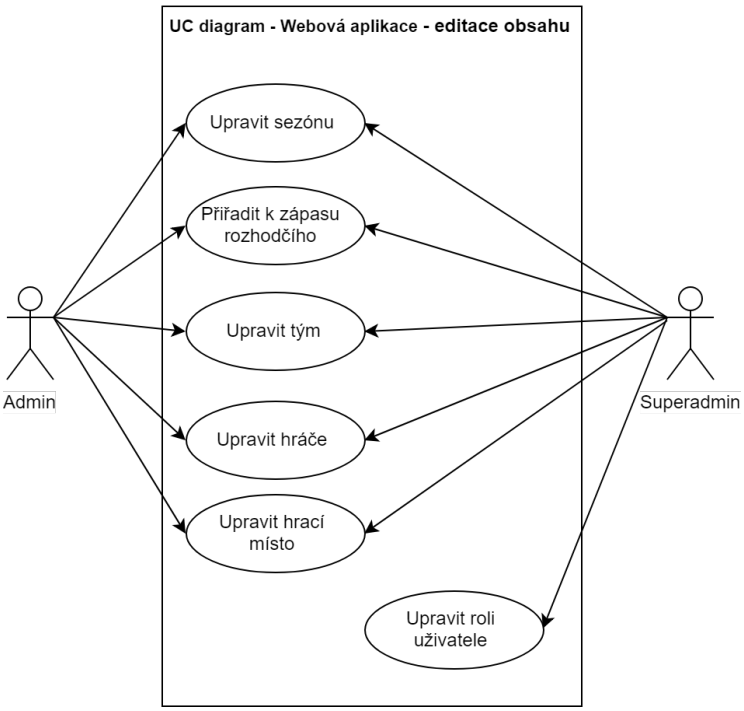
3.3.3 Editace obsahu

Editací obsahu se rozumí úprava stávajícího záznamu v databázi bez nutnosti smazání starých hodnot. Přihlášený uživatel, který má přiřazena práva ADMIN nebo SUPERADMIN, má možnost editovat následující entity:

- Hráč
- Tým

- Sezóna
- Zápas (pouze přiřazení rozhodčího)
- Hrací místo

Možnost editace záměrně není implementována u všech entit, které lze vytvořit. V některých případech je uživatelsky přívětivější danou entitu smazat a vytvořit znovu.



Obrázek 3.14: UC diagram - editace obsahu

3.4 Mobilní aplikace

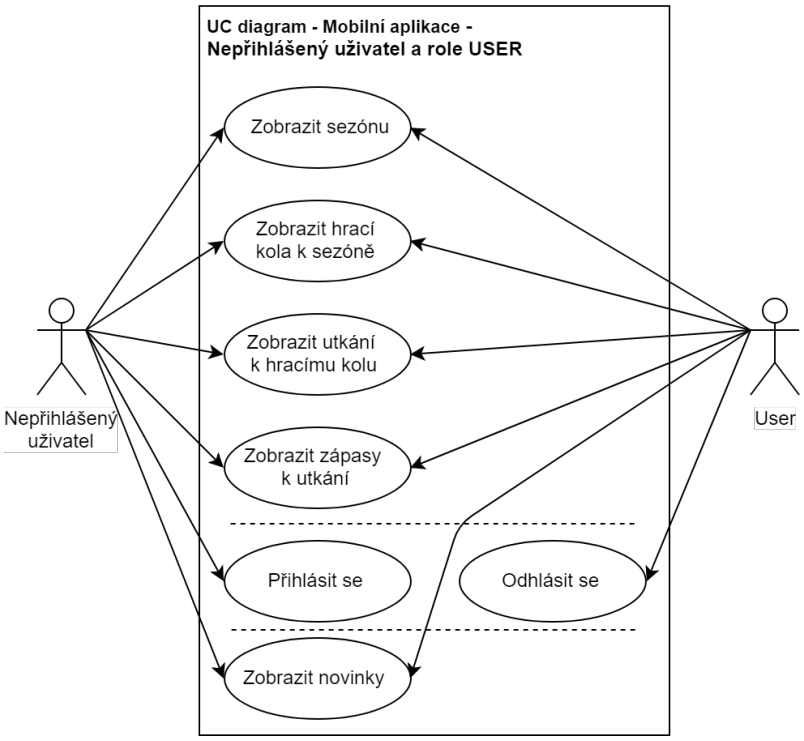
Mobilní aplikace bude navržena tak, aby ji mohla používat široká uživatelská základna - od uživatelů, kteří se pouze chtějí informovat o výsledcích Extraligy, přes fanoušky jednotlivých týmů, kteří chtějí být informováni o novinkách, až po rozhodčí, kteří aplikaci využijí pro zaznamenávání výsledků. Aplikace bude implementována pro operační systém Android.

3.4.1 Užívatelé bez speciálních oprávnění

Mnoho dnešních aplikací vyžaduje po uživateli již při prvním spuštění registraci. Častokrát ani není vysvětleno, proč by se uživatel registrovat měl a jaké tím získá výhody, a tak mnoho uživatelů aplikaci raději odinstaluje a nebude ji nadále používat.

Navrhovaná mobilní aplikace si dává za cíl mimo jiné zatraktivnit badminton v očích veřejnosti, a tak bude od samého začátku koncipována tak, aby nabídla maximum užitečných informací i pro uživatele, který se z nějakého důvodu nechce do aplikace registrovat. Veškeré veřejné sekce budou tedy dostupné v aplikaci i bez přihlášení.

Na obrázku 3.15 jsou zobrazeny případy užití mobilní aplikace nepřihlášeným uživatelem.

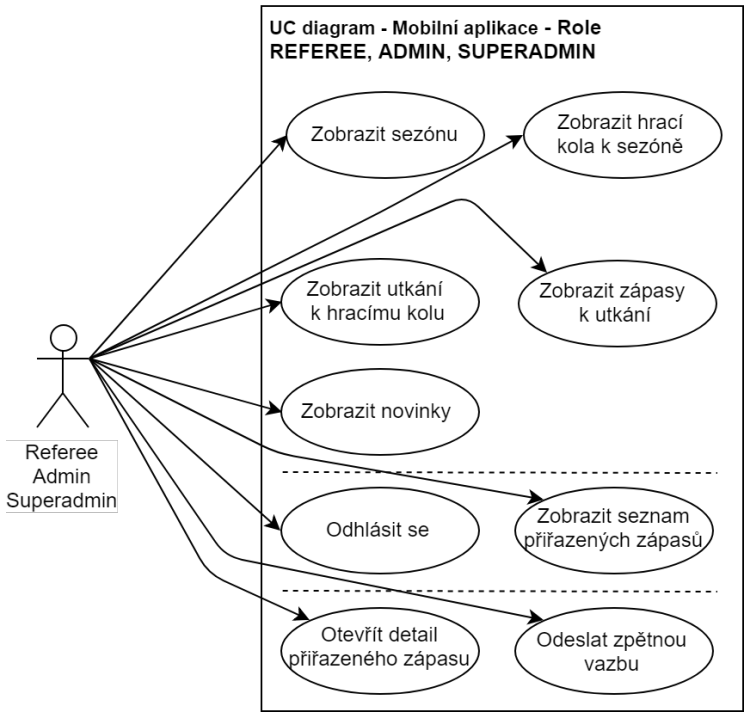


Obrázek 3.15: UC diagram - uživatelé bez zvláštních oprávnění

V první verzi mobilní aplikace nebude implementována funkce registrování nového uživatele. Každý uživatel si může vytvořit svůj účet ve webové aplikaci. Z obrázku 3.15 vyplývá, že přihlášený uživatel v první verzi aplikace nemá žádné výhody oproti uživatelovi nepřihlášenému. Přihlášení do aplikace bude nejprve sloužit převážně pro účely ověření uživatele za účelem pozdějšího přidělení vyšších práv (REFEREE). Později, až se aplikace rozšíří mezi uživatele, se mohou implementovat další funkcionality, které nabídnou přihlášenému uživateli výhody, jako jsou například personifikované informace, speciální nabídky, atd.

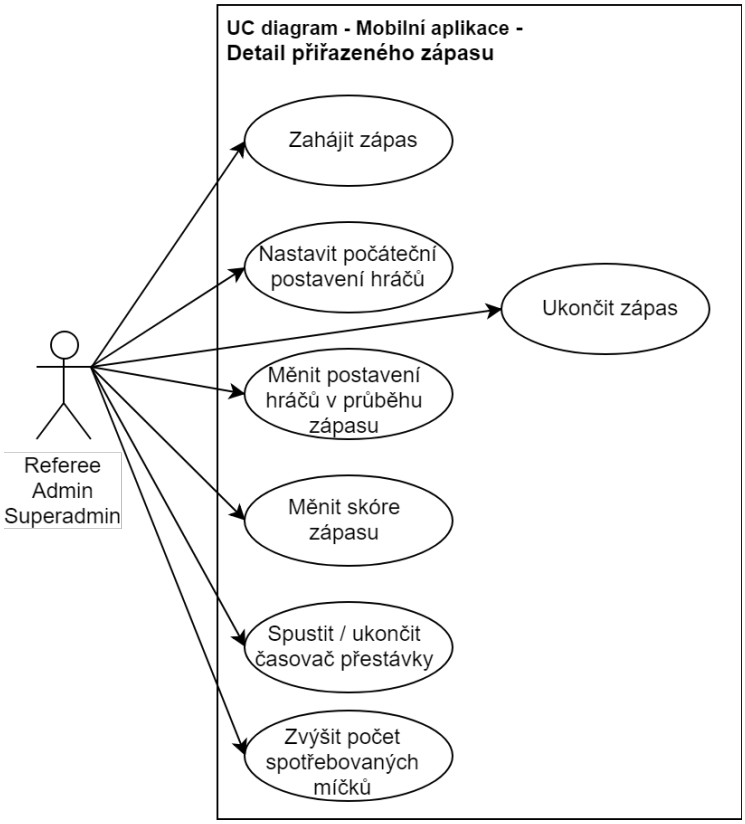
3.4.2 Užívatelé s rolí Referee a vyšší

Uživatelům, kteří mají přidělenou roli Referee, Admin nebo Superadmin, se v mobilní aplikaci po přihlášení zobrazí dosud skrytá sekce, která bude obsahovat seznam zápasů, ke kterým je daný uživatel přiřazen jako hlavní rozhodčí. Po vybrání zápasu ze seznamu se uživateli otevře obrazovka, která umožní zaznamenávat skóre tohoto zápasu. Zaznamenané výsledky se budou online zapisovat do databáze a zpětně promítat do aplikace. Dále bude uživateli zobrazena možnost odeslat zpětnou vazbu k aplikaci.



Obrázek 3.16: UC diagram - uživatelé s rolí Referee a vyšší

Zaznamenávání skóre zápasu bude probíhat na samostatné obrazovce, která se uživateli zobrazí po vybrání zápasu ze seznamu. Na obrazovce bude uživatelské rozhraní přizpůsobeno pohodlnému ovládání v průběhu zápasu. Jednotlivé funkcionality, které je nutné implementovat, jsou zobrazeny na obrázku 3.17. Všechny počáteční vlastnosti zápasu, jako jsou například jména hráčů, dostane hlavní rozhodčí už předvyplněné. Odpovědnost za správnost těchto údajů nese vrchní rozhodčí, který zápas vytvořil ve webové aplikaci.



Obrázek 3.17: UC diagram - zaznamenávání výsledků zápasu

4 Google Firebase

Firebase je platforma pro vývoj mobilních a webových aplikací, která vznikla v roce 2011 jako start-up, ale v roce 2014 ji odkoupila firma Google [14]. Pod křídly Google se velmi rychle rozrostla o mnoho produktů, v současné době jich nabízí necelou dvacítku [10]. Ty se zaměřují především na usnadnění vývoje mobilních a webových aplikací a vylepšování jejich kvality. Mezi nejznámější prodkuty platformy Firebase patří:

- Firebase Realtime Database - real-time NoSQL databáze, která dovoluje ukládat a synchronizovat data v rámci milisekund;
- Cloud Firestore - real-time dokumentově orientovaná databáze, nástupce Firebase Realtime Database. V průběhu zpracovávání této práce se Firestore posunul z verze beta do GA (General Availability);
- Authentication - služba pro ověřování uživatelů;
- Cloud storage - datové úložiště;
- Crashlytics - nástroj na sběr, prioritizaci a opravování chyb v aplikacích;
- Performance Monitoring - nástroj pro kontrolu výkonnosti aplikací a analýzu výkonnostních problémů;
- Cloud Messaging - nástroj pro odesílání cílených zpráv a notifikací.

Navrhované řešení se opírá především o real-time databázi Firebase Realtime Database a o systém pro ověřování uživatelů. Těmto produktům se detailně věnují následující kapitoly.

4.1 Firebase Realtime Database

Firebase Realtime Database je NoSQL databáze, která běží v cloudu společnosti Google. Umožňuje sdílení a synchronizaci dat mezi jednotlivými zařízeními v rámci milisekund, což umožňuje uživatelům přístup k nejnovějším datům ze všech zařízení. Je také optimalizovaná pro použití offline - pokud uživatel ztratí přístup k internetu, Realtime Database SDK použije lokální cache pro ukládání a servírování změn. Jakmile se uživatel opět připojí, jeho změny jsou automaticky synchronizovány. Firebase Realtime Database poskytuje SDK pro Android, iOS a Javascript, což hezky

koresponduje s požadavky této práce. Společně s Firebase Authentication umožňuje nastavit přístup konkrétního uživatele ke konkrétním datům v databázi, nebo naopak některým přístupům zabránit. Data jsou v databázi stromově strukturovaná, kořenovým prvkem je samotná URL databáze.

Real-time databáze je v rámci základního tarifu poskytována zdarma. Stačí se zaregistrovat do Firebase Console a spustit svou databázi. Omezením v základním tarifu je počet zároveň připojených uživatelů. Ten je omezen na 100, což pro první verzi navrhované aplikace stačí, nicméně do budoucna je potřeba počítat s nutností rozšířit základní tarif na placený. Ceny jednotlivých tafirů ukazuje tabulka 4.1.

Funkcionality Firebase Realtime Database	Základní tarif (Spark plan)	Flame plan	Blaze plan
	Zdarma	$25 měsíčně	Placený průběžně
Současně připojení uživatelé	100	100 000	100 000 na databázi
Maximální velikost databáze	1 GB	2,5 GB	$5 za GB
Maximální download	10 GB za měsíc	20 GB za měsíc	$1 za GB
Více databází v jednom projektu	NE	NE	ANO

Tabulka 4.1: Ceny tarifů Google Firebase [9]

4.2 Firebase Authentication

Firebase Authentication je nástrojem pro jednoduché a rychlé sestavení ověřovacího mechanismu. Sdružuje v sobě několik identity providerů, jako jsou např. Facebook, Google, nebo Twitter. Mimo to nabízí ověření uživatele pomocí e-mailu a hesla. Právě této možnosti využívá navrhovaný systém. Firebase nabízí Auth SDK pro Android, iOS a JavaScript. Základním objektem poskytovaným Firebase Auth je objekt User, který je vracen po úspěšném ověření, a obsahuje data o uživateli, jako jsou email, uid, způsob přihlášení nebo zobrazované jméno. Pro Android nabízí Firebase i předpřipravené uživatelské prostředí, kde jsou implementovány všechny metody ověření uživatele. Toto UI je ale až příliš striktní a proto nebude v práci použito. V první verzi aplikace bude implementováno pouze ověření uživatele pomocí emailu a hesla, ale řešení je připraveno i na implementaci ověření pomocí dalších podporovaných identity providerů.

4.3 Firebase Console

Firebase Console je webový nástroj pro nastavení a správu Firebase. Před použitím Firebase je potřeba zde založit nový projekt. V tomto projektu je pak možné zapínat a vypínat funkcionality Firebase, nastavovat povolené identity providery, manuálně zasahovat do databáze a tak dále.

4.4 Zabezpečení databáze

Komunikace mezi aplkací (Clientem) a databází (Hostem) je zabezpečená pomocí protokolu TLS (Transport Layer Security), což je protokol, který zabraňuje odposlouchávání či falšování zpráv. Kromě toho Firebase přichází k novým konceptem zabezpečení databáze, tzv. Firebase Rules. Na straně serveru uložen soubor, který pro každý uzel v databázi definuje, kdo s daným uzlem může jak nakládat. Firebase definuje čtyři základní typy pravidel:

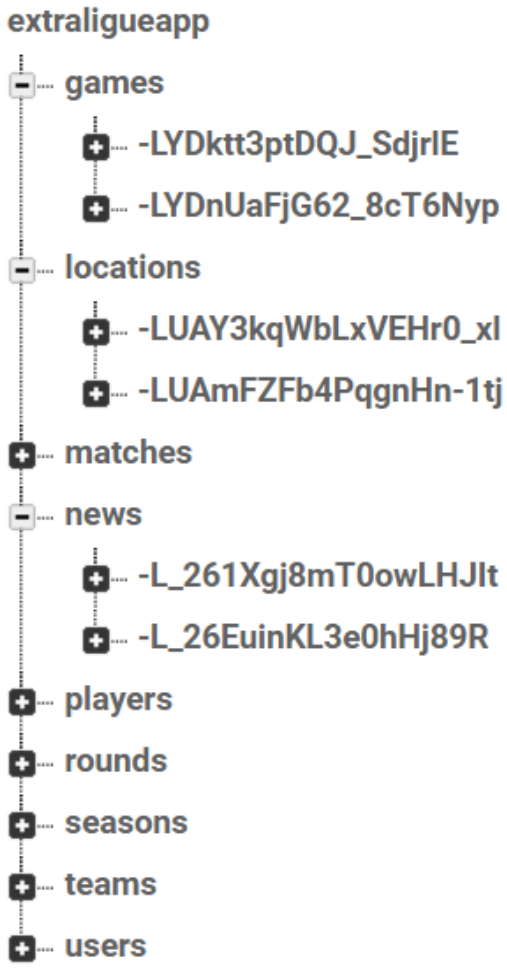
- .read - popisuje, zda a kdy je povoleno číst data uživateli,
- .write - popisuje, zda a kdy je povoleno data zapisovat,
- .validate - definuje, jaký je korektní formát zapisovaných dat,
- .indexOn - specifikuje index pro podporu třídění a dotazování.

Pokud do databáze přijde požadavek na čtení nebo zápis dat do konkrétního uzlu, databáze nejprve zkontroluje, zda daný dotaz odpovídá dříve definovaným bezpečnostním požadavkům. Pokud ne, není daný požadavek obsloužen. Bezpečnostní pravidla se definují přímo ve Firebase Console.

```
{
  "rules": {
    "matches": {
      ".indexOn": ["gameId","referee"],
      ".read" : "true",
      ".write": "auth.uid != null"
    },
    "news": {
      ".read" : "true",
      ".write": "auth.uid != null"
    },

    ...

    ".read" : "true",
    ".write" : "auth.uid != null"
  }
}
```

4.5 Struktura databáze

Firebase Realtime Database je NoSQL databáze, takže data v ní nejsou uložena ve formě relací, ale jsou uspořádána do stromové struktury. Jednotlivé záznamy se označují jako uzly. Kořenovým uzlem, nebo zkráceně kořenem, databáze je URL databáze, která je vygenerovaná na základě pojmenování projektu v Firebase Console. Na obrázku 4.1 je zobrazena navržená struktura databáze. Uzly první úrovně svým názvem reprezentují typ dat, který je v jejich poduzlech uložen. Uzly druhé úrovně jsou pak už klíče jednotlivých záznamů v databázi. Tyto klíče může vývojář definovat sám, nebo může nechat Firebase vytvořit klíč za něj. Výhodou automatického generování klíčů je to, že jsou generovány na základě time-stamp, a tudíž pak jejich lexikografické řazení odpovídá i řazení dle data vytvoření.



Obrázek 4.1: Stromová struktura navržené databáze

5 Webová aplikace

První částí navrhovaného softwarového řešení je webová aplikace, která slouží pro administraci a vytváření obsahu v Google Realtime Database. Následující kapitola krátce představuje použitou technologii, popisuje základní principy a strukturu řešení a seznamuje s navrženým uživatelským rozhraním webové aplikace.

5.1 ReactJS

React (také psáno jako ReactJS či React.js) je open-source JavaScriptová knihovna pro vytváření uživatelských rozhraní pro webové aplikace. React je postaven na používání tzv. "komponent" zapisovaných jako HTML tagy. Každá komponenta má vlastní stavové proměnné, což umožňuje vytváření komplexních uživatelských rozhraní (UI). Komponenty nemohou přímo ovlivňovat stav rodičovských komponent, do nichž jsou vnořeny. O projekt se stará firma Facebook, Instagram a komunita individuálních vývojářů [11]. Komponenty, které tvoří uživatelské rozhraní, jsou stromově strukturované. Tím je zajištěna přehlednost a také proudění stavových proměnných od kořene dále.

ReactJS používá JSX. JSX (Javascript Syntax Extension) je syntaxe textu ve formátu XML/ HTML. Za pomoci preprocesoru (Babel) dochází k transformaci textu v HTML formátu na standartní objekty jazyka Javascript. JSX umožnuje vkládání textu HTML do kódu jazyka Javascript, čímž vznikají tzv. React elementy.

Každá třída v ReactJS musí obsahovat minimálně metodu `render()`. Ta se stará o vykreslení uživatelského rozhraní. Základní struktura třídy vypadá následovně [5]:

```
1  import React, { Component } from 'react'
2  export default class Dashboard extends Component {
3    render () {
4      return (
5        <div>
6            Here comes the content.
7        </div>
8      )
9    }
10 }
```

Kromě metody `render()` se často používají metody životního cyklu ReactJS komponenty. V komponentách použitých v tomto projektu se užívají hlavně metody `componentDidMount()` a `componentWillUnmount()`.

Metoda `componentDidMount()` je volána hned po metodě `render()`, tedy po vykreslení uživatelského rozhraní [5]. V práci se používá především k inicializaci stavové proměnné komponenty `this.state` a také k volání Firebase Realtime Database - tedy k získání dat, která bude daná komponenta zobrazovat. Každou změnou `this.state` se vyvolá i překreslení uživatelského rozhraní. Stavová proměnná se nikdy nesmí nastavovat přímo - k její editaci slouží metoda `this.setState()`, která nastavení provede asynchronně a tedy neblokuje uživatelské rozhraní.

Metoda `componentDidUnmount()` je volána ve chvíli, kdy je daná komponenta odstraněna z uživatelského rozhraní [5]. V této metodě je dobré odhlásit všechny Observery tak, aby nedocházelo ke zbytečnému zahlcování paměti.

5.2 Struktura webové aplikace

Webová aplikace v této práci je implementována jako tzv. Single-page aplikace. Základní komponentou je třída `App`, která se stará o vykreslování dalších komponent podle volby uživatele. K navigaci je používána komponenta `BrowserRouter`. Třída `App` ve své `componentDidMount()` metodě kontroluje, zda je uživatel přihlášen. Na základě toho nastavuje svou stavovou proměnnou, kterou pak `BrowserRouter` používá k podmíněnému zobrazení komponent. Pomocí `Switch` komponenty pak kontroluje, jaká komponenta je právě uživatelem požadována a zobrazuje ji do elementu `container` třídy `App`. Třída `App` pak dále obsahuje element pro zobrazení vrchního navigačního menu. I zde se obsah vykresluje podmíněně podle toho, zda je uživatel přihlášen, či nikoliv. Ve chvíli, kdy uživatel požaduje přístup na stránku, která je pouze pro přihlášeného uživatele, je automaticky přesměrován na stránku přihlášení. Toto přesměrování je implementováno ve funkcích `PublicRoute` a `PrivateRoute`.

```
function PublicRoute ({component: Component,
authed, loggedin,...rest}) {
  return (
    <Route
      {...rest}
      render={(props) => loggedin === false
        ? <Component {...props} />
        : <Redirect to='/' />}
    />
  )
}
```

Funkce `PublicRoute` přijímá jako parametr `Component`, která má být zobrazena, a dále flagy `authed` a `loggedin`. Ty vyjadřují, v jaké roli uživatel ke komponentě přistupuje.

Flag `loggedin` nabývá hodnoty TRUE, pokud je uživatel přihlášen (nehledě na jeho roli). Flag `authed` potom říká, zda je uživatel oprávněn stránku zobrazit, tedy jestli má přiřazenou roli Admin a vyšší.

```
function PrivateRoute ({component: Component,
authed, loggedin,...rest}) {
  return (
    <Route
      {...rest}
      render={(props) => authed === true
        ? <Component {...props} />
        : loggedin === true
          ? <Redirect to={{pathname: '/',
                  state: {from: props.location}}} />
          : <Redirect to={{pathname: '/login',
                  state: {from: props.location}}} />}
    />
  )
}
```

V implementaci funkce `PrivateRoute` se nejprve kontroluje, zda má uživatel dostatečné oprávnění k zobrazení požadované stránky. Pokud ano, je požadovaná komponenta vrácena. Pokud ne, zkontroluje se, zda je uživatel alespoň přihlášen. Pokud přihlášen je, je přesměrován na úvodní stránku pro přihlášené uživatele, tedy stránku Home. Pokud přihlášen není, je přesměrován na stránku umožňující přihlášení.

Na obrázku 5.1 je zobrazen diagram všech komponent, které jsou dostupné přímo pomocí odkazů v navigační liště. Mimo tyto uvedené komponenty jsou v aplikaci použity i další, které jsou dostupné po vybrání konkrétní položky v zobrazených datech (např. seznam kol k vybrané sezóně). V levé části jsou zobrazeny odkazy, které jsou uživateli zobrazeny v navigační liště, vpravo pak konkrétní komponenty, které jsou uživateli v případě splnění podmínek zobrazeny.

5.3 Funkce pro manipulaci s databází

Pro zvýšení přehlednosti kódu jsou všechny funkce, které zapisují do Firebase Realtime Database, soustředěny do souboru `writeDatabase.js`. Všechny používají jako návratovou hodnotu objekt `Promise`, což je objekt reprezentující "příslib" budoucí hodnoty - výsledku asynchronní operace. Níže je uveden seznam všech implementovaných funkcí pro práci s databází. Další kapitoly, popisující implementaci jednotlivých komponent, se budou na tento seznam odvolávat.

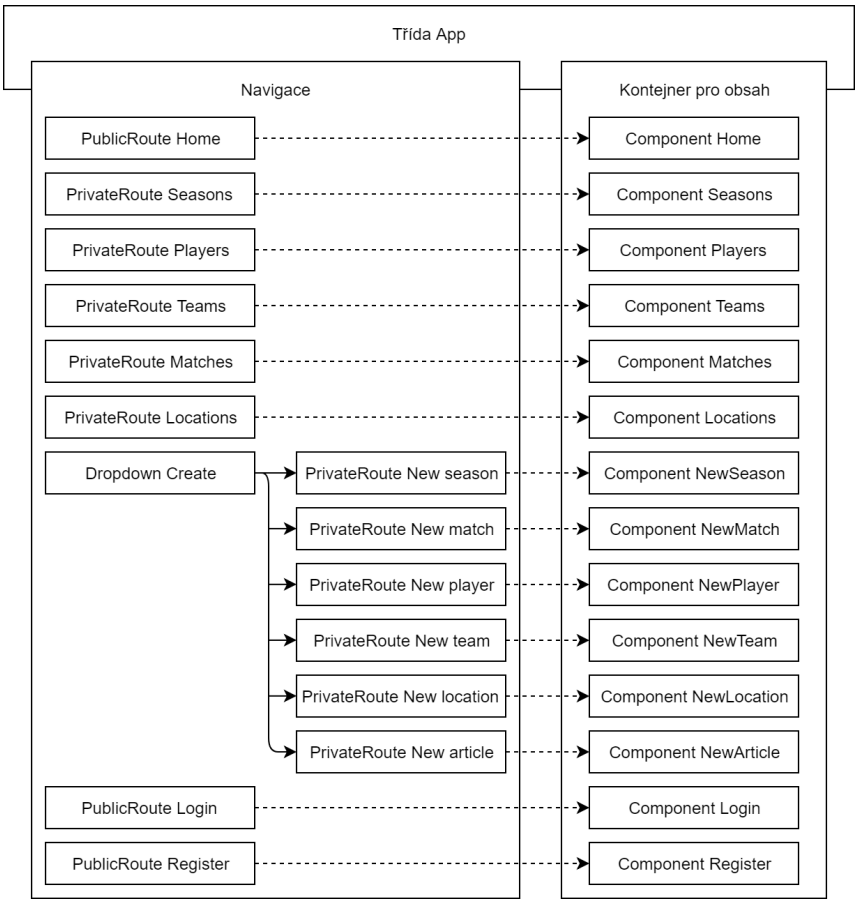
```
{
  function writeNewArticlePromise(article)

  function updateReferee(matchId, refereeId)

  function writeNewMatchPromise(match)
  function deleteMatchByIdPromise(matchId)

  function writeRoundsPromise(seasonId, rounds)
  function deleteRoundsBySeasonId(seasonId)
  function updateRoundsBySeasonId(seasonId, rounds)

  function writeNewSeasonPromise(season)
  function deleteSeasonById(id)
  function updateSeasonById(id, season)

  function writeNewPlayer(player)
  function deletePlayerById(id)
  function updatePlayerById(id, player)

  function writeNewTeam(team)
  function deleteTeamById(id)
  function updateTeamById(id, team)

  function writeNewLocation(location)
  function deleteLocationById(id)

  function writeNewGame(game)
  function deleteGameById(id)
}
```

5.4 Komponenty

V následujících podkapitolách budou popsány jednotlivé použité komponenty. Všechny dědí od třídy Component, která je součástí frameworku ReactJS. Zároveň jsou všechny tzv. stavové komponenty, tudíž si uchovávají svůj stav v `this.state`.

5.4.1 Komponenta Home

Komponenta Home slouží jako domovská stránka celé aplikace. Přístup k této komponentě je veřejný. Pokud se do webové aplikace přihlásí uživatel s právy nižšími než Admin, je toto jediná komponenta, kterou si může zobrazit. Ve svém těle zobrazuje seznam novinek. Tento seznam se do stavové proměnné ukládá v metodě `componentDidMount` voláním Firebase Realtime Database, konkrétně cesty `/news`. Novinky jsou na stránce seřazeny podle data jejich publikace.



Obrázek 5.1: Diagram komponent dostupných z navigační lišty

5.4.2 Komponenta Register

Důležitou komponentou je komponenta Register. Ta uživateli zprostředkovává možnost registrovat se do aplikace. Po registraci je uživateli automaticky přidělena role User. Komponenta ve svém těle obsahuje dvě vstupní textová pole - jedno pro registrační email a druhé pro heslo. Tlačítkem ”Register” pak uživatel potvrdí vložené údaje a jeho účet je vytvořen.

5.4.3 Komponenta Login

Tato komponenta slouží k přihlášení uživatele, který si dříve zřídil svůj uživatelský účet. Po úspěšném přihlášení je uživatel automaticky přesměrován na komponentu Home. Na základě jeho role mu jsou zobrazeny nebo skryty další komponenty, které jsou označeny jako `PrivateRoute`.

5.4.4 Komponenty Seasons, NewSeason a UpdateSeason

Komponenta Seasons zobrazuje seznam dostupných sezón uložených v databázi. Každá položka je reprezentována komponentou `Panel`, která je dostupná

z importovaného balíku `react-bootstrap`. Na Panelu je zobrazeno jméno a ročník sezóny, počet kol a odkaz na zobrazení všech herních kol, které jsou k této sezóně přiřazeny. Uživatel má možnost sezónu upravit pomocí tlačítka "Update", nebo vymazat tlačítkem "Delete". Po kliknutí na tlačítko "Update" je uživatel přesměrován na komponentu `UpdateSeason`. Kliknutím na tlačítko "Delete" se uživateli zobrazí dialog, ve kterém musí potvrdit, že danou sezónu chce opravdu vymazat. Po potvrzení dialogu je sezóna nevratně vymazána. K vymazání slouží funkce `deleteSeasonById(id)` dostupná z vytvořeného balíku funkcí pro manipulaci s databází.

K vytvoření nové sezóny slouží komponenta NewSeason. Uživatel definuje novou sezónu pomocí jejího názvu, měsíce a roku začátku, měsíce a roku konce a počtu kol. Počtem kol se rozumí počet základních kol, kola play-off se přidají k vytvořené sezóně automaticky. Po potvrzení zadaných údajů je v systému vytvořen záznam o nové sezóně a zároveň jsou vygenerovány záznamy pro každé kolo této sezóny. Do databáze se zápis provede voláním funkce `writeNewSeasonPromise(season)`.

Kliknutím na tlačítko "Update" na seznamu sezón je uživatel přesměrován na komponentu UpdateSeason. Ta slouží k upravení již existující sezóny v databázi. Má implementováno stejné uživatelské rozhraní jako komponenta NewSeason, ale v metodě `componentDidMount` se do polí načtou hodnoty uložené v databázi podle ID sezóny, které komponenta dostane v `this.props`. Po upravení hodnot a kliknutí na tlačítko "Update season" se změny uloží do databáze pomocí funkce `updateSeasonById(id, season)` a uživatel je automaticky přesměrován na komponentu Seasons.

5.4.5 Komponenty Players, NewPlayer a UpdatePlayer

Komponenta Players slouží k zobrazení všech hráčů, kteří jsou uloženi v databázi. Každý jednotlivý hráč je reprezentován komponentou `Panel`, na kterém je zobrazeno jméno hráče a klub, ke kterému přísluší. Stejně jako u Sezóny i zde má uživatel možnost daného hráče upravit, nebo smazat. Smazání se provede (po potvrzení dialogu) pomocí funkce `deletePlayerById(id)`

K vytvoření nového hráče slouží komponenta NewPlayer. Hráč je definován pomocí jména, data narození, národnosti a týmu, za který nastupuje. Kliknutím na tlačítko "Create player" se zavolá funkce `writeNewPlayer(player)` a tím je vytvořen nový záznam v databázi.

Stejně jako u komponenty Seasons má uživatel po klinutí na tlačítko "Update" u konkrétního hráče možnost aktualizovat jeho údaje v databázi. V metodě `componentDidMount` se z databáze načtou aktuální informace o hráči podle jeho ID. Po upravení a kliknutí na tlačítko "Update player" je zavolána funkce `updatePlayerById(id, player)`, která aktualizuje údaje v databázi a uživatel je přesměrován na komponentu Players.

5.4.6 Komponenty Matches, NewMatch

Komponenta Matches slouží k zobrazení zápasů jednotlivých utkání. Uživatel musí nejprve z rozbalovacího seznamu vybrat, ke kterému utkání chce zobrazit zápasy. Tento seznam se načítá z databáze v metodě `componentDidMount`. Po vybrání utkání se zavolá metoda `loadMatchesForGameId(gameId)`, která z databáze získá všechny zápasy k vybranému utkání, a zobrazí každý jako `Panel`. Kromě jmen hráčů a skóre je na Panelu zobrazen i aktuálně přiřazený rozhodčí a také rozbalovací seznam, kde je možné přiřadit k zápasu rozhodčího jiného, nebo zápas nepřiřadit žádnému z dostupných rozhodčích. Hodnota přiřazeného rozhodčího se v databázi mění ihned s vybráním rozhodčího v seznamu - volá se metoda `updateReferee(matchId, refereeId)`. Níže je uvedena ukázka volání Firebase databáze, konkrétně načítání zápasů pro zvolené utkání.

```
loadMatchesForGameId = (gameId) =>{
  this.setState({
    matches : [],
    loading: true,
  })
  var matchesRef = firebase.database().ref('/matches/')
      .orderByChild("gameId").startAt(gameId).endAt(gameId);
  matchesRef.on('value', data => {
    var newMatches = []
    data.forEach( child =>{
      newMatches = newMatches.concat([ {'key' : child.key,
                    'data' : child.val() } ]);
    })
    this.setState({
      matches : newMatches,
    });
  })
  this.setState({
    loading: false,
  })
}
```

Nový zápas lze v aplikaci založit dvěma způsoby. Oba způsoby používají komponentu NewMatch, rozdíl je ale v tom, zda vytváří nový zápas z menu, nebo z detailu konkrétního utkání. Pokud uživatel vytváří nový zápas z menu, musí nejprve vybrat z rozbalovacího seznamu sezónu a kolo. Poté se mu zobrazí další rozbalovací seznam, kde jsou vypsána jednotlivá utkání pro zvolené kolo. Povybrání utkání se zobrazí pole pro vybrání hráčů, kteří k zápasu nastoupí. Uživatel opět vybírá z rozbalovacího seznamu, kde jsou zobrazeni pouze hráči, kteří jsou přiřazeni k týmům, které nastupují v dříve vybraném utkání.

5.4.7 Komponenty Teams, NewTeam a UpdateTeam

Stejný princip, jako je popsán výše u komponent manipulujících s objekty sezóny a hráče, je použit i u komponent, které zobrazují informace o týmech. Komponenta Teams zobrazuje všechny týmy, které jsou zavedeny v databázi. Každý zobrazuje jako samostatný `Panel`, na kterém je zobrazeno jméno týmu a jeho zkratka. Každý tým je možné upravovat a mazat. K upravování slouží komponenta UpdateTeam, která funguje stejně jako dříve zmíněné "Update komponenty". K mazání týmu slouží metoda `deleteTeamById(id)`, k úpravě metoda `updateTeamById(id, team)`. Nový tým se vytváří v komponentě NewTeam, která volá funkci `writeNewTeam(team)`.

5.4.8 Komponenty Locations, NewLocation a UpdateLocation

Výše uvedený princip je uplatněn i u komponent, které se týkají hracích míst. Komponenta Locations zobrazuje všechna hrací místa z databáze, každé jako samostatný `Panel`. Na něm je zobrazeno jméno hracího místa, jeho adresa a počet dostupných kurtů. Každé hrací místo lze opět upravovat a mazat. Úpravu stávajících údajů uživatel provádí na komponentě UpdateLocation. K mazání slouží metoda `deleteLocationById(id)`. Nové hrací místo se vytváří pomocí komponenty NewLocation, kde se po vyplnění potřebných údajů zavolá metoda `writeNewLocation(location)`.

5.4.9 Komponenty Rounds a Games

Komponenty Rounds a Games slouží jako "listview" - tedy zobrazují seznam položek (kol, resp. utkání). Komponenty Seasons, Rounds, Games a Matches tak tvoří kaskádu, kdy obsah podřazené komponenty záleží na položce zvolené v nadřazené komponentě. Informace o vybrané položce si komponenty předávají v `this.props`. Hrací kola není možné upravovat a mazat. Utkání je možné smazat.

5.4.10 Komponenta NewArticle

Poslední komponentou implementovanou ve webové aplikaci je komponenta NewArticle. Slouží k vytváření nových článků - novinek, které jsou zobrazeny v komponentě Home. V těle komponenty jsou celekm tři prvky - pole pro nadpis, textarea pro obsah článku a tlačítko pro uložení. Uložení se provede zavoláním funkce `writeNewArticlePromise`. Při uložení do databáze se k článku automaticky přidá datum a čas jeho vytvoření.

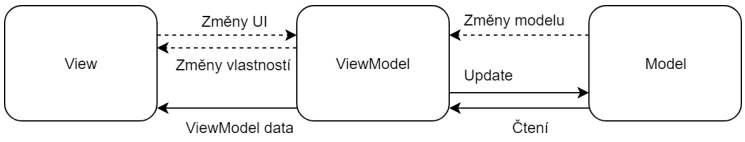
6 Klientská mobilní aplikace

Následující kapitola se podrobně zabývá implementací mobilní aplikace pro systém Android. Jsou v ní popsány nejdůležitější funkcionality a části kódu, naznačena struktura aplikace a popsáno uživatelské rozhraní.

6.1 Architektura aplikace - MVVM

MVVM je jeden z architektonických návrhových vzorů, který podporuje oddělení uživatelského rozhraní od business logiky aplikace. Dáva si za cíl zachovat uživatelské rozhraní tak jednoduché jak jen to jde, aby bylo snadnější aplikaci testovat a udržovat. MVVM se skládá ze tří částí:

- Model - obsahuje množinu tříd, která popisuje business logiku a data aplikace
- View - reprezentuje UI komponenty, sestává se z aktivit, fragmentů a jejich XML layoutů. Tato vrstva je zodpovědná za zobrazování dat získaných z ViewModel.
- ViewModel - spojovací vrstva mezi Modelem a View. Reprezentuje zobrazovací logiku, t.j. vystavuje metody, které pomáhají udržovat stav View a manipulovat s Modelem.



Obrázek 6.1: Model-View-ViewModel

6.2 Integrace Google Firebase

Aby bylo možné v aplikaci použít Google Firebase SDK, je třeba přidat do závislostí projektu následující řádky. Po synchronizaci Gradle jsou dostupné balíčky z Firebase SDK, jako `com.google.firebase.database`, nebo `com.google.firebase.auth`. Ty pak poskytují potřebné třídy.

```
dependencies {
  implementation 'com.google.firebase:firebase-core:16.0.7'
  implementation 'com.google.firebase:firebase-database:16.1.0'
  implementation 'com.google.firebase:firebase-auth:16.1.0'
}
```

6.3 Komunikace mezi aktivitami a fragmenty

Celá aplikace se skládá celkem ze tří aktivit. Aktivity pak v sobě obsahují fragmenty, které se zobrazují podle toho, jaký obsah si uživatel vyžádá. Jedním z důležitých bodů bylo implementovat komunikaci mezi mateřskou aktivitou a jejími fragmenty, protože akce vyvolaná z jednoho fragmentu může odkazovat na fragment jiný. Toho je dosaženo pomocí interface, které mateřská aktivita implementuje. Fragmentům se pak při vytváření předá reference na jejich mateřskou aktivitu. Ve chvíli, kdy na fragmentu dojde k události, o které by měla mateřská aktivita vědět, zavolá fragment přes tuto referenci mateřskou aktivitu. Ta provede danou akci.

Jako příklad může sloužit aktivita `MainActivity`, která implementuje rozhraní `OnItemSelectedListener`. Obsahuje spodní navigační menu, které obsahuje položky Dashboard, Actual season a My account. Podle toho, která položka v menu je zvolena, zobrazuje aktivita ve svém těle příslušné fragmenty.

Rozhraní `OnItemSelectedListener` je implementováno takto:

```
public interface OnItemSelectedListener {
  void onSeasonSelected(String seasonId);
  void onRoundSelected(String seasonId, String roundId);
  void onNewsSelected(String newsId);
  void onGameSelected(String gameId);
  void onLoginWanted();
  void onLoginClicked();
}
```

Hlavní aktivita toto rozhraní implementuje.

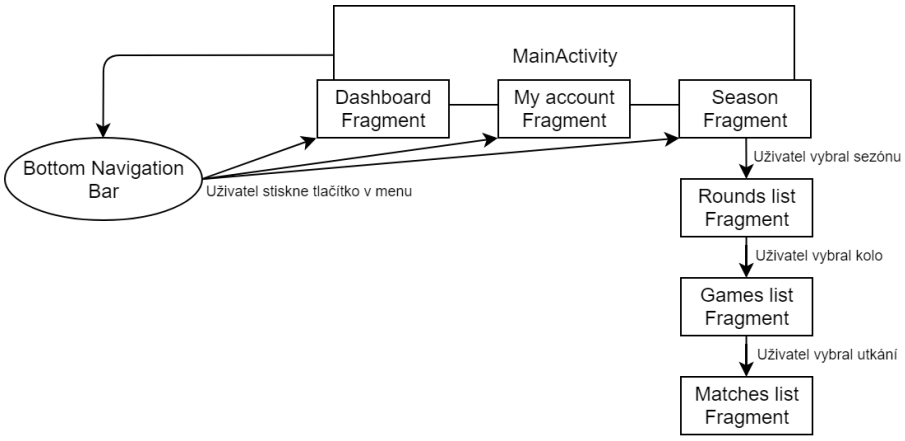
```
public class MainActivity extends AppCompatActivity
  implements OnItemSelectedListener {

  @Override
  public void onLoginClicked() {
      MyAccountFragment newFragment = new MyAccountFragment();
      switchFragments(newFragment);
      getSupportActionBar().setDisplayHomeAsUpEnabled(false);
  }


```

```
12      @Override
13      public void onSeasonSelected(String seasonId) {
14          RoundsListFragment newFragment = new RoundsListFragment();
15          Bundle bundle = new Bundle();
16          bundle.putString("seasonId", seasonId);
17          newFragment.setArguments(bundle);
18
19          switchFragments(newFragment);
20          getSupportActionBar().setDisplayHomeAsUpEnabled(true);
21      }
22
23      ...
24
25  }
```

Na každé volání hlavní aktivita reaguje přepnutím právě zobrazovaného fragmentu. O přepnutí se stará metoda `switchFragments()`, kde FragmentManager provede danou transakci. Aby mohl fragment volat mateřskou aktivitu, musí získat její referenci. Tu získá v metodě `onAttach()`, což je metoda životního cyklu fragmentu.

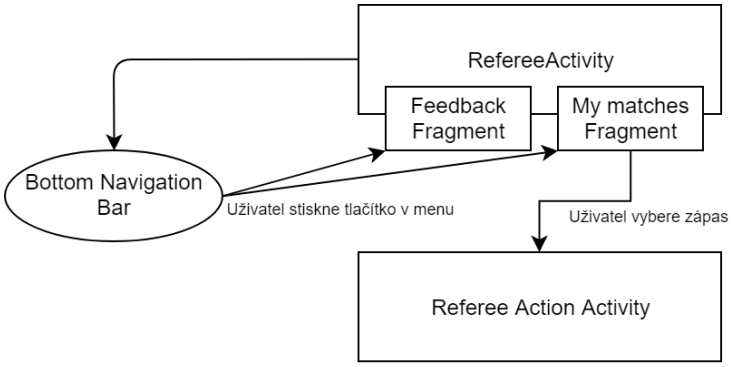
```
1   @Override
2   public void onAttach(Context context) {
3     super.onAttach(context);
4     try {
5         mCallback = (OnItemSelectedListener) context;
6     } catch (ClassCastException e){
7         throw new ClassCastException(getActivity().toString()
8         + " must implement OnItemSelectedListener");
9     }
10  }
```

Na obrázku 6.2 je vyobrazeno celkové flow fragmentů v hlavní aktivitě. Všechny zobrazené fragmenty volají hlavní aktivitu přes výše popsané rozhraní. Fragmenty, které obsahují listview, navíc aktivitě předávají informaci o tom, kterou položku z daného listview uživatel zvolil.



Obrázek 6.2: Flow fragmentů v hlavní aktivitě

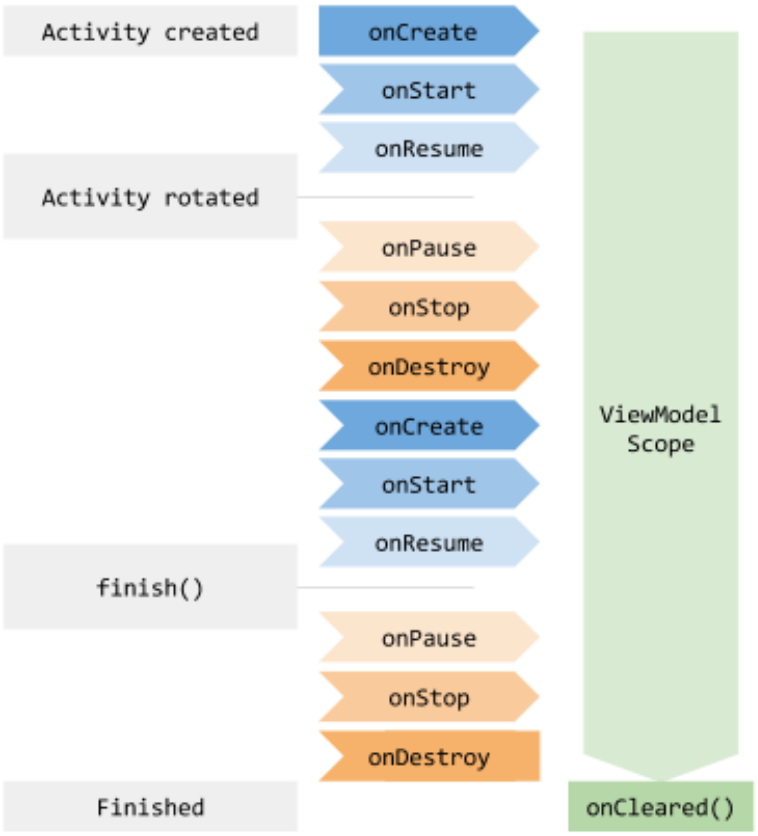
Podobně je implementována i `RefereeActivity`, která je dostupná přihlášeným uživatelům s rolí Referee a vyšší. Tato aktivita implementuje rozhraní `RefereeActionsListener`. Po vybrání zápasu ze seznamu je uživatel přeměrován do aktivity `RefereeActionActivity`, kde už dochází k samotnému zaznamenávání výsledků.



Obrázek 6.3: Flow fragmentů v aktivitě rozhodčího

6.4 Data a jejich sdílení mezi fragmenty

Data, která jsou v aplikaci zobrazována, jsou získávána voláním Firebase databáze. O jejich servírování prvkům UI se stará vrstva ViewModel, která je popsaná v kapitole 6.1. Každá aktivita v aplikaci si drží instanci ViewModelu. Ten je svázán s životním cyklem aktivity (scope) - to znamená, že se zachová tak dlouho, dokud bude scope aktivity "alive". Jinak řečeno, ViewModel nebude zničen (destroyed), pokud jeho vlastník bude zničen kvůli konfiguračním změnám (např. otočení). Nová instance vlastníka bude pouze přepojena na existující instanci ViewModelu. Celý proces popisuje obrázek 6.4.



Obrázek 6.4: Životní cyklus ViewModelu [1]

Každý fragment ve své `onCreate()` metodě získá referenci na ViewModel tzv. ve scope aktivity, což znamená, že mu bude k dispozici po celou dobu životního cyklu aktivity. Fragment tak má přístup ke všem metodám, které daný ViewModel vystavuje. Podle toho, která data má fragment zobrazovat, si ve své metodě `onActivityCreated` vytvoří tzv. Observery na vystavené metody. Při každé změně sledovaného objektu se vyvolá metoda `onChanged` a fragment si upraví svoje uživatelské rozhraní podle aktuálního stavu sledovaného objektu.

Jako příklad zde bude uveden fragment `DashboardFragment`. V metodě `onCreate()` získá instanci třídy `SharedViewModel`, která dědí od `ViewModel`.

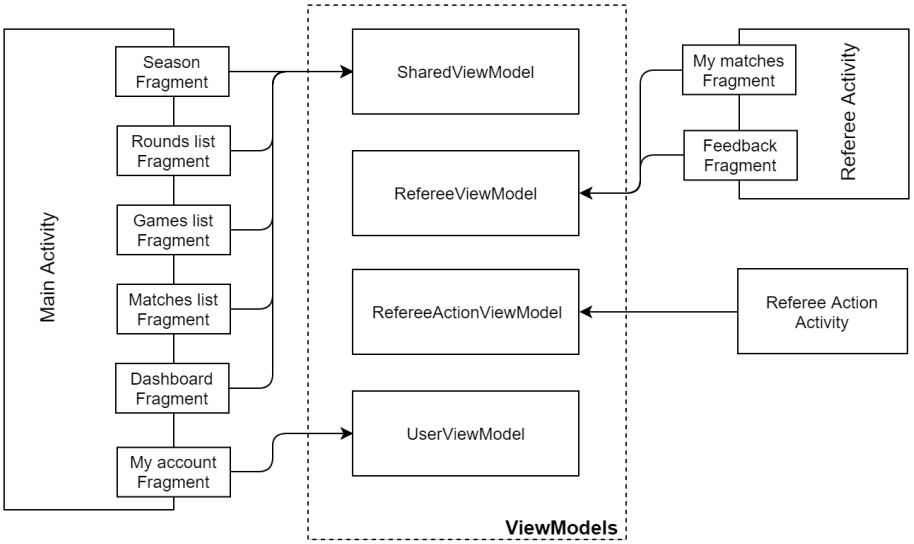
```
1 @Override
2 public void onCreate(@Nullable Bundle savedInstanceState) {
3   super.onCreate(savedInstanceState);
4   mModel = ViewModelProviders.of(getActivity())
5                              .get(SharedViewModel.class);
6
7   FirebaseApp.initializeApp(getActivity());
8 }
```

Poté v `onActivityCreated` nastaví Observer na metodu `getNewsListLiveData()`, kterou vystavuje právě zmíněný `SharedViewModel`.

```
@Override
public void onActivityCreated
(@Nullable Bundle savedInstanceState) {
  super.onActivityCreated(savedInstanceState);
  Log.d(TAG, "onActivityCreated");

  if (mModel != null) {
    LiveData<List<News>> liveData= mModel.getNewsListLiveData();

    liveData.observe(getActivity(),new Observer<List<News>>() {
      @Override
      public void onChanged(@Nullable List<News> news) {
        mDataList = news;
        mNewsAdapter.setNewsList(news);
        Log.d(TAG, "onChange: " + mDataList.size());

        if (mDataList.isEmpty()) {
          recyclerView.setVisibility(View.GONE);
          emptyImageView.setVisibility(View.VISIBLE);
          emptyTextView.setVisibility(View.VISIBLE);
        } else {
          recyclerView.setVisibility(View.VISIBLE);
          emptyImageView.setVisibility(View.GONE);
          emptyTextView.setVisibility(View.GONE);
        }
      }
    });
  }
}
```

Při každé změně dostane Observer do metody `onChanged()` aktuální list novinek, který si uloží a zároveň předá Adapteru, který ho pak dál zprostředkuje RecyclerView. Pokud je list prázdný, upraví své uživatelské rozhraní - skryje RecyclerView a zobrazí uživateli informaci o tom, že nebyla nalezena ani jedna položka. Více informací o použití RecyclerView je v kapitole 6.5.

Na obrázku 6.5 je vyobrazeno, jak jednotlivé fragmenty používají dostupné viewmodely. Viewmodel je vždy z fragmentu volán ve scope aktivity. Jednotlivé instance tříd ViewModel pak přímo čtou a zapisují do databáze Firebase Realtime Database.



Obrázek 6.5: Použití ViewModelů jednotlivými fragmenty

6.4.1 Čtení dat

Metody, které vystavuje ViewModel, mají za úkol jediné - získat aktuální data z databáze a poskytnout je View, které je přihlášeno k odběru (má nastavený Observer). Ke čtení dat se využívají třídy z Firebase SDK.

```
1  @NonNull
2  public LiveData<List<News>> getNewsListLiveData(){
3    Log.d(TAG, "Get news: ");
4    DatabaseReference news = FirebaseDatabase.getInstance()
5                             .getReference("news");
6    FirebaseQueryLiveData mLiveData =
7        new FirebaseQueryLiveData(news);
8
9    LiveData<List<News>> mNewsLiveData =
10       Transformations.map(mLiveData, new NewsDeserializer());
11
12   return mNewsLiveData;
13 }
```

Nejprve se získá reference na konkrétní uzel v databázi. Poté se tato reference předá třídě `FirebaseQueryLiveData`, která na danou referenci nastaví Listener. Tento Listener potom při každé změně sledovaného uzlu vrací aktuální `snapshot` dat. Ten je pomocí utility `Transformations.map()` rozparsován na jednotlivé potomky - instance tříd typu POJO. Zároveň, pro potřeby pozdějšího odkazování na konkrétní položku dat, se do POJO objektu navíc ukládá hodnota key - tedy klíč daného snapshotu v databázi. Této hodnoty je pak využito při zobrazování detailu dané položky.

```
private class NewsDeserializer
implements Function<DataSnapshot, List<News>> {
  @Override
  public List<News> apply(DataSnapshot dataSnapshot) {
    mNewsList.clear();
    for(DataSnapshot snap : dataSnapshot.getChildren()){
      News news = snap.getValue(News.class);
      news.setKey(snap.getKey());
      mNewsList.add(news);
    }
    return mNewsList;
  }
}
```

6.4.2 Zápis dat

RefereeActionViewModel se od ostaních implementovaných ViewModelů liší tím, že jako jediný do databáze zapisuje hodnoty. Těmito hodnotami jsou samozřejmě jednotlivé změny skóre zápasů a dalších vlastností zapasů, jako je počet spotřebovaných míčků nebo trvání zápasu. Jako příklad bude uvedeno zapisování hodnoty spotřebovaných míčků, nicméně implementace ostatních UseCase je až na malé odlišnosti stejná.

Pro zápis hodnoty se používá metoda `setValue()`, která je dostupná z Firebase SDK. Metoda se volá na instanci třídy DatabaseReference - na definovanou referenci zapíše předanou hodnotu. Pokud reference neexistuje, metoda ji v databázi vytvoří.

```
public void incrementShuttlesCount(){
  int actual = mMatchModel.getValue().getShuttlesCount();
  DatabaseReference ref = MATCH_REF.child("shuttlesCount");
  ref.setValue(++actual);
}
```

Aktuální hodnotu v databázi si ViewModel drží ve své vlastní instanci modelu `Match`. Není tedy nutné ve chvíli zapisování nových dat získávat aktuální hodnotu voláním databáze, ale stačí vzít hodnotu z lokálně uložené třídy.

6.5 Kolekce dat

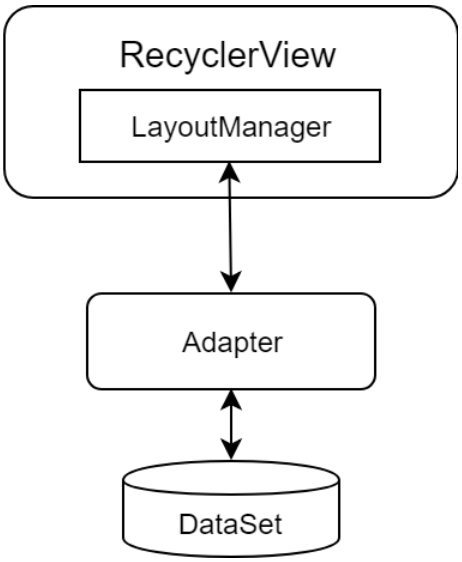
V aplikaci je často potřeba zobrazit scrollovací obrazovku se seznamem nějakých položek (novinek, kol, utkání, zápasů, ...). Velice dobrým a doporučovaným nástrojem pro zobrazení takových dat je `RecyclerView`. Jedná se o pokročilejší a flexibilnější verzi dříve používaného `ListView`. V RecyclerView pracuje několik komponent dohromady. Kontejnerem pro celé řešení je objekt `RecyclerView`, který se přidává

do layoutu. RecyclerView plní sám sebe objekty, které mu poskytuje `LayoutManager`. Položky v seznamu jsou reprezentovány tzv. view holder objekty. Tyto objekty jsou instance tříd, které jsou definovány rozšířením třídy `RecyclerView.ViewHolder`. Každý view holder zobrazuje právě jednu položku v seznamu. RecyclerView vytvoří právě tolik view holderů, kolik je potřeba pro zobrazení na jedné obrazovce a několik málo dalších (nevytváří tedy view holder ihned pro všechny položky seznamu, kterých může být veliké množství). Jakmile uživatel scrolluje seznamem, RecyclerView odebírá view holdery, které již nejsou na obrazovce vidět, a přepojuje je na data, které se se obrazovce zobrazují [2].

View holder objekty jsou spravovány Adapterem, který se vytváří rozšířením třídy `RecyclerView.Adapter`. Adapter mimo jiné váže view holdery k jejich datům pomocí přiřazení pořadí view holderu a zavoláním metody `onBindViewHolder()`.

Aby bylo možné RecyclerView v aplikaci použít, je nutné přidat do závislostí projektu následující řádek.

```
1 dependencies {
2   implementation 'com.android.support:recyclerview-v7:28.0.0'
3 }
```



Obrázek 6.6: Schéma fungování RecyclerView

6.6 Fragmenty

6.6.1 DashboardFragment

DashboardFragment obsahuje ve svém těle `RecyclerView`, ve kterém zobrazuje novinky, které získá voláním metody `getNewsListLiveData()` dostupné z `SharedViewModel`. Tento fragment náleží k hlavní aktivitě aplikace. Lze jej zobrazit tapnutím na první položku spodního navigačního menu hlavní aktivity. Fragment plní UC *Zobrazit novinky* z UC diagramu zobrazeného na obrázku 3.15 a 3.16.

6.6.2 SeasonFragment

SeasonFragment obsahuje ve svém těle `RecyclerView`, ve kterém zobrazuje sezóny uložené v databázi, které získá voláním metody `getSeasonsListLiveData()` dostupné z `SharedViewModel`. Tento fragment náleží k hlavní aktivitě aplikace. Lze jej zobrazit tapnutím na druhou položku spodního navigačního menu hlavní aktivity. Fragment plní UC *Zobrazit sezónu* z UC diagramu zobrazeného na obrázku 3.15 a 3.16. Tapnutím na položku v seznamu sezón se vyvolá Callback do mateřské aktivity a zobrazí se `RoundFragment` pro vybranou sezónu.

6.6.3 RoundFragment

RoundFragment obsahuje ve svém těle `RecyclerView`, ve kterém zobrazuje kola příslušná k vybrané sezóně, které získá voláním metody `getRoundsListLiveData(String seasonId)` dostupné z `SharedViewModel`. Tento fragment náleží k hlavní aktivitě aplikace. Je zobrazen po vybrání konkrétní sezóny uživatelem. Fragment plní UC *Zobrazit hrací kola k sezóně* z UC diagramu zobrazeného na obrázku 3.15 a 3.16. Tapnutím na položku v seznamu kol se vyvolá Callback do mateřské aktivity a zobrazí se `GameFragment` pro vybrané hrací kolo.

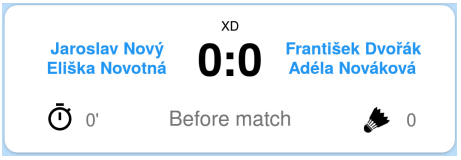
6.6.4 GameFragment

GameFragment obsahuje ve svém těle `RecyclerView`, ve kterém zobrazuje utkání příslušná k vybranému hracímu kolu, které získá voláním metody `getGamesListLiveData(String roundId)` dostupné z `SharedViewModel`. Tento fragment náleží k hlavní aktivitě aplikace. Je zobrazen po vybrání konkrétního hracího kola uživatelem. Fragment plní UC *Zobrazit utkání k hracímu kolu* z UC diagramu zobrazeného na obrázku 3.15 a 3.16. Tapnutím na položku v seznamu utkání se vyvolá Callback do mateřské aktivity a zobrazí se `MatchesFragment` pro vybrané utkání.

Obrázek 6.7: Detail položky utkání

6.6.5 MatchesFragment

MatchesFragment obsahuje ve svém těle `RecyclerView`, ve kterém zobrazuje zápasy příslušné k vybranému utkání, které získá voláním metody `getMatchesListLiveData(String gameId)` dostupné z `SharedViewModel`. Tento fragment náleží k hlavní aktivitě aplikace. Je zobrazen po vybrání konkrétního utkání uživatelem. Fragment plní UC *Zobrazit zápasy k utkání* z UC diagramu zobrazeného na obrázku 3.15 a 3.16.



Obrázek 6.8: Detail položky zápas

6.6.6 MyAccountFragment

MyAccountFragment zobrazuje ve svém těle základní údaje o uživateli, které získá voláním metod dostupných z `UserViewModel`. Tento fragment náleží k hlavní aktivitě aplikace. Lze jej zobrazit tapnutím na první položku spodního navigačního menu hlavní aktivity. Fragment plní UC *Přihlásit se* a *Odhlásit se* z UC diagramu zobrazeného na obrázku 3.15 a 3.16. Pokud uživatel není do aplikace přihlášen, je mu v těle fragmentu nabídnuto přejít na `LoginFragment`, kde se může přihlásit. Pokud přihlášen již je, tak na základě role, které je uživateli přiřazena, fragment zobrazí či nezobrazí možnost přechodu do části aplikace vyhrazené rozhodčím. Tapnutím na tlačítko ”Show my matches” se spustí aktivita `RefereeActivity`.

6.6.7 LoginFragment

LoginFragment slouží k přihlášení uživatele do aplikace. V svém těle obsahuje dvě pole - jedno pro zadání uživatelského jména a druhé pro zadání hesla, a také tlačítko pro potvrzení. Stisknutím tlačítka se vyvolá metoda z `UserViewModel`, která se pomocí Firebase Auth SDK pokusí uživatele přihlásit. Při úspěchu se uživateli zobrazí `MyAccountFragment` s jeho aktuálními údaji, při neúspěchu je uživatel informován a k přesměrování nedojde. Fragment realizuje UC *Přihlásit se* z UC diagramu zobrazeného na obrázku 3.15.

```
public void doLoginWithEmailAndPassword(Activity activity,
String email, String password){
  mAuth.signInWithEmailAndPassword(email,password)
  .addOnCompleteListener(activity,
                      new OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
      if (task.isSuccessful()) {
        Log.d(TAG, "signInWithEmail:success");
        ((MutableLiveData<FirebaseUser>) mUser)
                          .setValue(mAuth.getCurrentUser());
        ((MutableLiveData<Boolean>) mErrorFlag).setValue(false);
        checkUserRole(mAuth.getCurrentUser().getUid());
      } else {
        ((MutableLiveData<Boolean>) mErrorFlag).setValue(true);
        mError = task.getException().getLocalizedMessage();
      }
    }
  });
}
```

6.6.8 MyMatchesFragment

MatchesFragment obsahuje ve svém těle `RecyclerView`, ve kterém zobrazuje zápasy, ke kterým byl přihlášený uživatel přiřazen Adminem jako rozhodčí. Data jsou získávána voláním metody `getRefereeMatchesListLiveData(String refereeUID)` dostupné z `RefereeViewModel`. Tento fragment náleží k aktivitě `RefereeActivity`. Fragment plní UC *Zobrazit seznam přiřazených zápasů* z UC diagramu zobrazeného na obrázku 3.16. Po tapnutí na konkrétní zápas se uživateli otevře aktivita `RefereeActionActivity`.

6.6.9 FeedbackFragment

FeedbackFragment slouží k informování přihlášeného uživatele o možnosti zanechat zpětnou vazbu k používání aplikace. Ve svém těle obsahuje informační `TextBox` a dále tlačítko, které uživatele po kliknutí přesměruje do webového prohlížeče, který automaticky přejde na webovou stránku s dotazníkem spokojenosti. Fragment plní UC *Odeslat zpětnou vazbu* z UC diagramu zobrazeného na obrázku 3.16. Tato zpětná vazba je určena konkrétně pro uživatele, kteří používají aplikaci s rolí Referee a vyšší. Pro uživatele s nižší rolí bude k zanechání zpětné vazby sloužit standardní nástroj poskytovaný Google Play (po vypublikování aplikace).

6.7 Aktivity

V celé aplikaci jsou implementovány celkem tři aktivity. Dvě z nich ve svých tělech zobrazují fragmenty, které jsou popsány v kapitole 6.6. Blíže se jednotlivými

aktivitami zabývají následující podkapitoly.

6.7.1 MainActivity

MainActivity je hlavní aktivitou aplikace, která je zobrazena vždy, když uživatel spustí aplikaci tapnutím na ikonu v menu telefonu. Obsahuje spodní navigační lištu, na které jsou tři tlačítka. Ve svém těle pak střídá fragmenty `DashboardFragment`, `SeasonFragment`, `RoundFragment`, `GameFragment`, `MatchesFragment`, `MyAccountFragment` a `LoginFragment`. Aktivita implementuje rozhraní `OnItemSelectedListener`.

6.7.2 RefereeActivity

RefereeActivity je druhou aktivitou aplikace, která ve svém těle pro zobrazení obsahu používá fragmenty. Je dostupná pouze přihlášeným uživatelům, kteří mají přiřazenou roli Referee a vyšší. Obsahuje spodní navigační menu, ve kterém jsou dvě tlačítka. V této aktivitě se zobrazují fragmenty `MyMatchesFragment` a `FeedbackFragment`. Aktivita implementuje rozhraní `RefereeActionsListener`.

6.7.3 RefereeActionActivity

RefereeActionActivity je jedinou aktivitou aplikace, která ve svém těle zobrazuje obsah přímo a ne pomocí fragmentů. Tato aktivita je stěžejní aktivitou pro rozhodčí, protože slouží právě k zaznamenávání výsledků zápasů. Je zobrazena vždy pro konkrétní zápas a tedy i používaný ViewModel je inicializován pomocí `matchId` tak, aby bylo možné nastavit správnou cestu do Firebase Realtime Database.

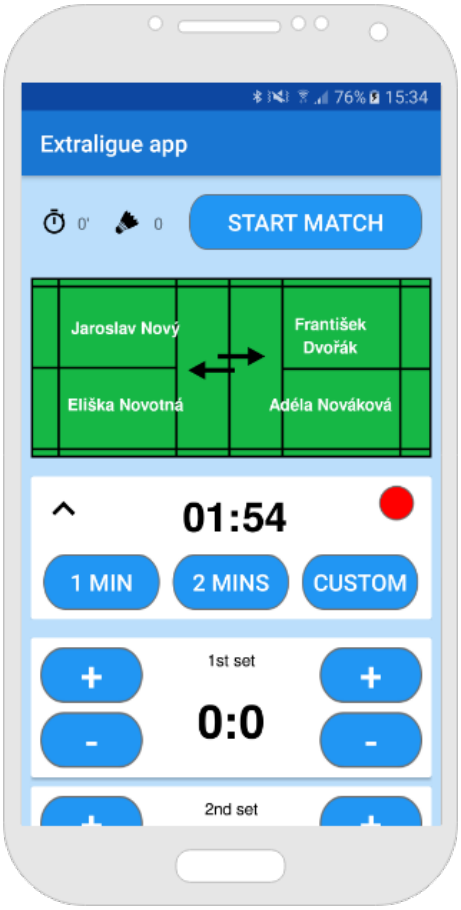
Aktivita disponuje velkým množstvím funkcionalit, které vycházejí z UC diagramu zobrazeného na obrázku 3.17. Proto i její uživatelské rozraní působí na první pohled velmi komplexně. Obrazovka se dá rozdělit do čtyř sekcí seřazených vertikálně.

- První sekce obsahuje ukazatel trvání zápasu, ukazatel spotřebovaných míčků a tlačítko pro zahájení/ukončení zápasu.
- Druhá sekce obsahuje vizualizaci badmintonového kurtu, na které jsou zobrazena jména hráčů, kteří v zápase hrají. Zároveň tato sekce slouží rozhodčím k určení a zaznamenání správného postavení hráčů při čtyřhře. Rozhodčí má možnost nejen zaměnit strany jednotlivých týmů na konci setu, ale i zaměňovat postavení hráčů v průběhu setu. Na změnu stran týmů reaguje celý zbytek UI tak, aby bylo ovládání co nejintuitivnější.
- Třetí sekce zobrazuje časovač, který je nezbytný pro řízení každého zápasu. Rozhodčí si ho může zobrazit/skrýt pomocí šipky v levé části tak, aby zbytečně nezabíral místo na obrazovce ve chvíli, kdy není potřeba. Rozhodčí má k dispozici dva předpřipravené intervaly odpočtu - 1 minutu a 2 minuty. Dále má možnost si definovat vlastní časový interval.

- Poslední sekce slouží k samotnému zaznamenávání bodů v jednotlivých setech. Obsahuje ukazatel stavu pro každý set a zároveň tlačítka pro přidání/odebrání bodu každému ze soupeřů.

Aktivita je spuštěna po vybranání zápasu z `MyMatchesFragment`. Při spuštění z `Bundle` získá `matchId`, kterým si hned inicializuje `RefereeActionViewModel`. Po spuštění uživatel musí nastavit počáteční postavení hráčů na kurtu. Dále pak musí tlačítkem "Zahájit zápas" potvrdit, že zápas začal. Tím se aktivují tlačítko pro zaznamenávání bodů a zároveň se spustí záznam trvání zápasu. Dále pak uživatel přiřazuje body jednotlivým hráčům podle dění na kurtu. Počet spotřebovaných míčků může zvýšit tapnutím na ikonu míčku. Dokončení setu je kontrolováno automaticky, další set není nutné zahajovat manuálně. Standardní konec zápasu (tedy jeho dohrání) je taktéž detekováno automaticky.

Aktivita je chráněna proti nechtěnému ukončení. Pokud by uživatel chtěl opustit aktivitu předčasně (tedy před řádným ukončením zápasu), musí potvrdit, že si je vědom, že zápas ještě nebyl dokončen. Další úpravy skóre pro daný zápas pak musí povolit Admin.



Obrázek 6.9: Uživatelské rozhraní - RefereeActionActivity

7 Testování aplikace a zpětná vazba

Nedílnou součástí vývoje softwarových řešení je testování a získávání zpětné vazby od uživatelů již od prvních použitelných verzí aplikací. Posledním bodem zadání této diplomové práce bylo otestovat aplikace v reálném nasazení a získat zpětnou vazbu, pokud to bude možné. V této kapitole je popsán proces prvotního testování aplikace a dále pak způsob získávání zpětné vazby od uživatelů.

7.1 Testování aplikace

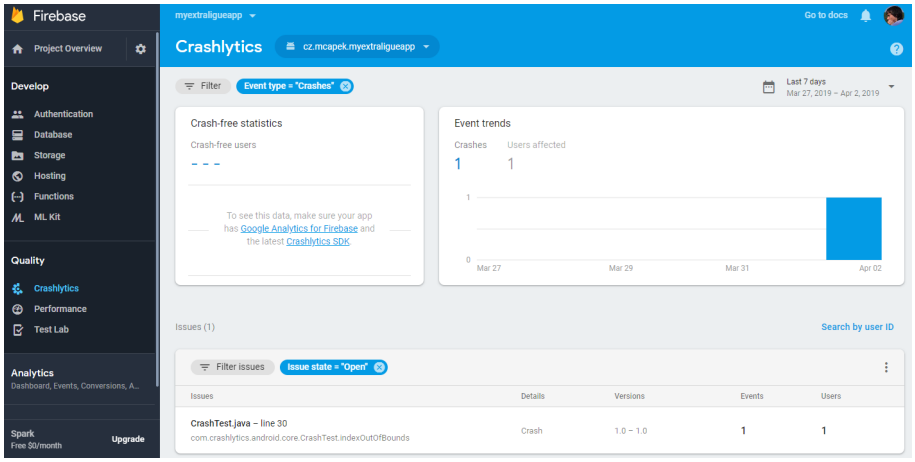
Testování aplikace probíhalo v několika vlnách podle toho, jak postupoval vývoj jednotlivých funkcionalit. Testování úplně první betaverze mobilní aplikace proběhlo v prosinci 2018 na turnaji Benátky Masters, který každoročně pořádá benátecký badmintonový klub. V této první fázi testování se mobilní aplikace distribuovala mezi hlavní rozhodčí, kteří na turnaji působili, pomocí přímé instalace testovacího buildu aplikace na jejich zařízení. Testování se zaměřovalo především na vhodnost a použitelnost vybrané technologie (Firebase Realitime Database). Dále pak byly s uživateli konzultovány prvky UI, které jsou nezbytné pro pohodlné ovládání aplikace během zápasů. Po skončení turnaje proběhl krátký workshop, ze kterého vzešel návrh na uživatelské rozhraní právě pro sekci aplikace, ve které pracují rozhodčí. Tato fáze testování ověřila, že vybraná technologie dokáže vyhovět požadavkům aplikace. Po ukončení této fáze testování byl vývoj testovací aplikace ukončen a začaly práce na nové verzi aplikace, která již korespondovala s požadavky definovanými v kapitole 3. Zároveň pokračovaly intenzivní práce na vývoji webové aplikace.

Druhá fáze testování se uskutečnila na konci února 2019. Na poslední únorový víkend bylo naplánováno 7. kolo extraligy smíšených družstev, ve kterém Benátky přivítaly ve své hale soupeře z Plzně. V rámci tohoto utkání probíhalo testování první verze mobilní aplikace a zároveň první verze webové aplikace. Webová aplikace byla testována ve spolupráci s vrchním rozhodčím tohoto kola. Mobilní aplikace byla otestována nejen hlavními rozhodčími, kteří zápasy v tomto utkání rozhodovali, ale i s dobrovolníky z řad fanoušků, kteří se přišli na průběh utkání podívat. Se všemi testery proběhl po ukončení utkání krátký rozhovor, kde testovanou aplikaci hodnotili a navrhovali možná vylepšení.

Další testování aplikace proběhne, pokud se podaří aplikaci prosadit u ČBaS

a dojde k nasazení aplikace na Google Play (obchod s aplikacemi pro Android). Tento bod již není součástí zadání práce.

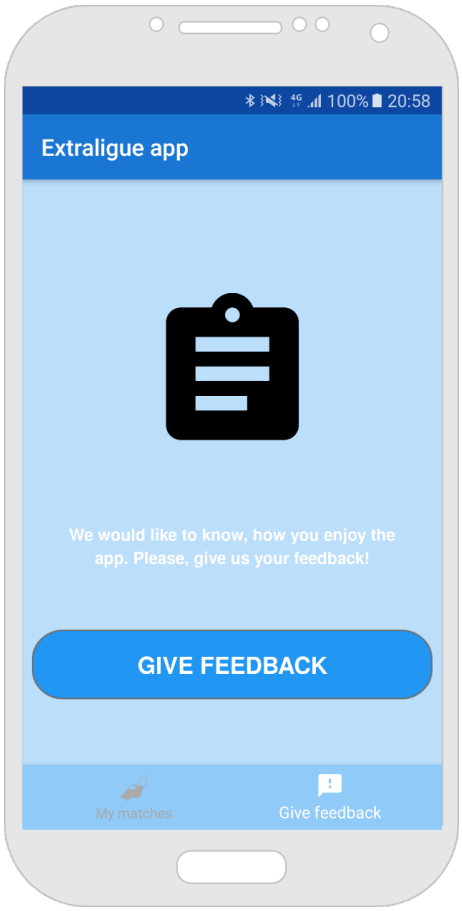
Pro sběr logů při pádu aplikaci při testování byl v aplikaci implementován nástroj Firebase Crashlytics. Jde o nástroj pro sběr, prioritizaci a sledování chyb v aplikaci. Ve Firebase Consoli je k dispozici přehledný dashboard, na kterém jsou v reálném čase zobrazovány pády aplikace i s přiloženými logy přímo ze zařízení. Nástroj tyto pády sám kategorizuje a prioritizuje, takže pro vývojáře je jednodušší se v reportovaných pádech vyznat a začít je opravovat. Na obrázku 7.1 je ukázka Crashlytics dashboard ve Firebase Consoli.



Obrázek 7.1: Google Firebase - Crashlytics dashboard

7.2 Získávání zpětné vazby

Zpětná vazba byla od uživatelů testovacích verzí aplikace získávána především přímým rozhovorem, a také pomocí dotazníkového šetření. Sloužila k vylepšení a úpravě funkcionalit aplikace. Testování se zúčastnila desítka testerů a jejich připomínky byly ihned zapracovávány. Pro další fázi testování je stejný dotazník, který byl předložen testerům, dostupný kliknutím na odkaz v aplikaci. Uživatel, který se rozhodne podělit se s vývojářem o svou zkušenost s aplikací, má možnost se v několika otázkách vyjádřit ke grafické stránce aplikace, k sadě nabízených funkcionalit, k použitelnosti a užitečnosti aplikace a také navrhnout další vylepšení a případně reportovat nedostatky. Pokud se aplikaci podaří nasadit na Google Play a rozšíří se mezi velké množství uživatelů, pak bude jistě tento dotazník důležitým nástrojem ke zjištění spokojenosti uživatelů a nabídne velké množství nápadů na další vylepšení aplikace.



Obrázek 7.2: Uživatelské rozhraní - FeedbackFragment

7.2.1 Chyby nalezené během testování

V tabulce 7.1 jsou uvedeny chyby, které se podařilo během testování odhalit. Zároveň je u každé zaznamenáno, zda tato chyba byla opravena, nebo zda je opravování irelevantní. Chyby uvedené v tabulce jsou sbírány již od prvních betaverzí aplikace.

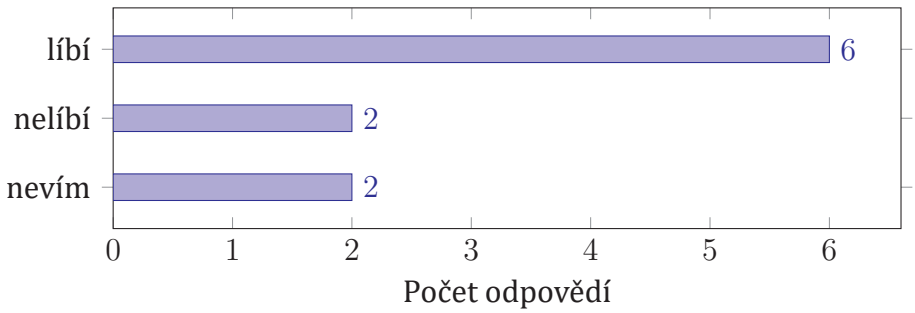
Popis chyby	Opraveno	Komentář
Kola jsou zobrazena ve špatném pořadí	ano	
Crash aplikace při přihlašování - nevyplnění jména a hesla	ano	
Zamrznutí aplikace při přihlášení bez přístupu k internetu	ano	
Crash aplikace při ruční změně role uživatele	ne	nebude měněno ruční změnou v databázi
Při sbalení timeru se timer vypne	ano	
Text novinky přetéká mimo vyhrazené místo	ano	
Zobrazen email namísto jména	ne	v první verzi aplikace není možnost ukládat jméno uživatele
Po ukončení zápasu se dále měnila doba trvání	ano	

Tabulka 7.1: Chyby nalezené během testování

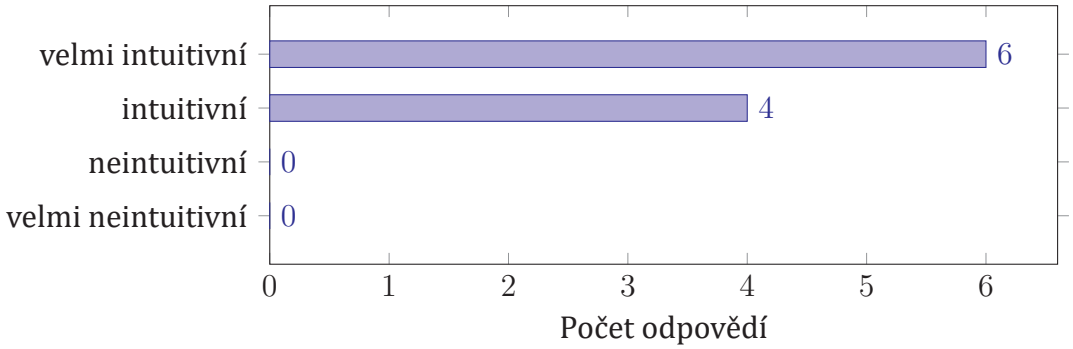
7.2.2 Výsledky dotazníku mezi testery

Níže jsou graficky zobrazeny výsledky dotazníkového šetření mezi testery aplikace. Celkem se šetření zúčastnilo 10 respondentů.

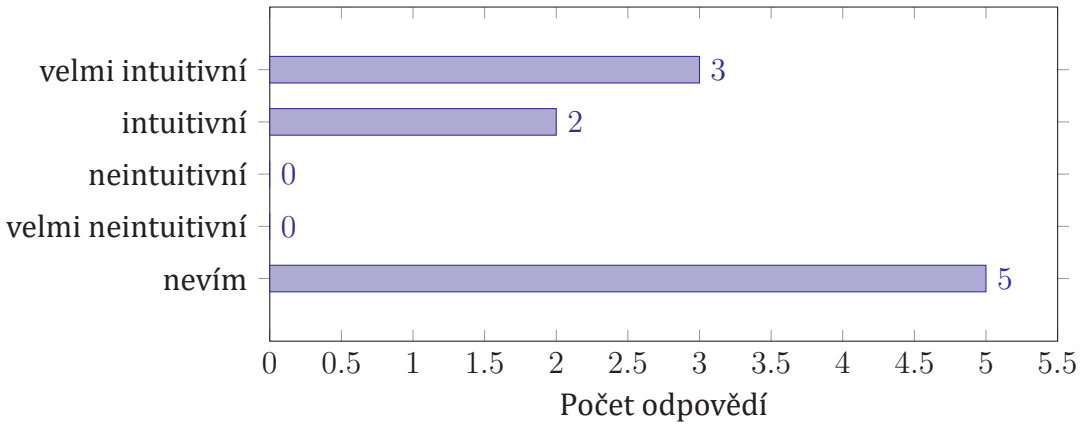
Otázka: Grafické uživatelské rozhraní se mi:



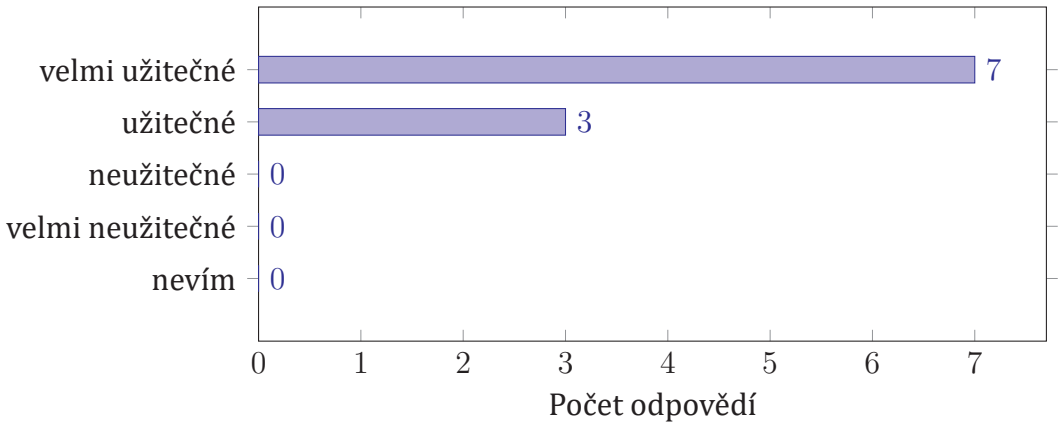
Otázka: Grafické uživatelské rozhraní aplikace hodnotím jako:



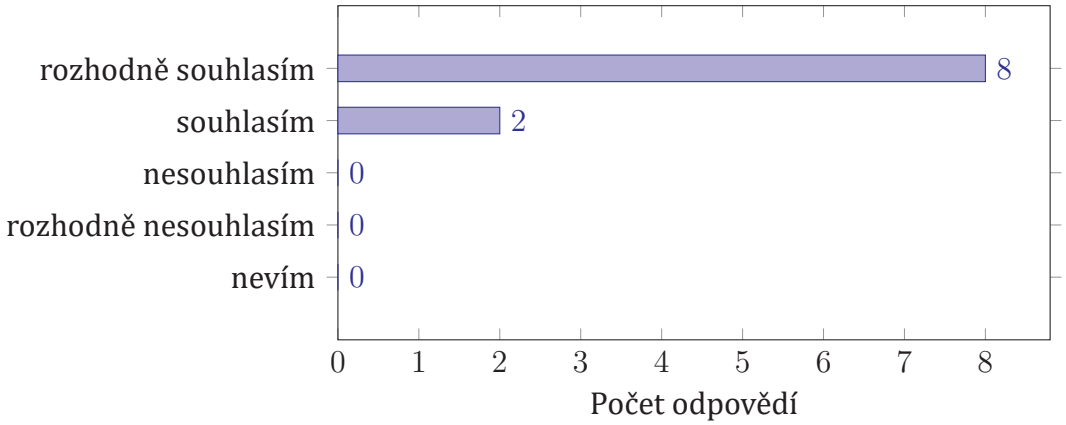
Otázka: Grafické uživatelské rozhraní sekce pro rozhodčí hodnotím jako:



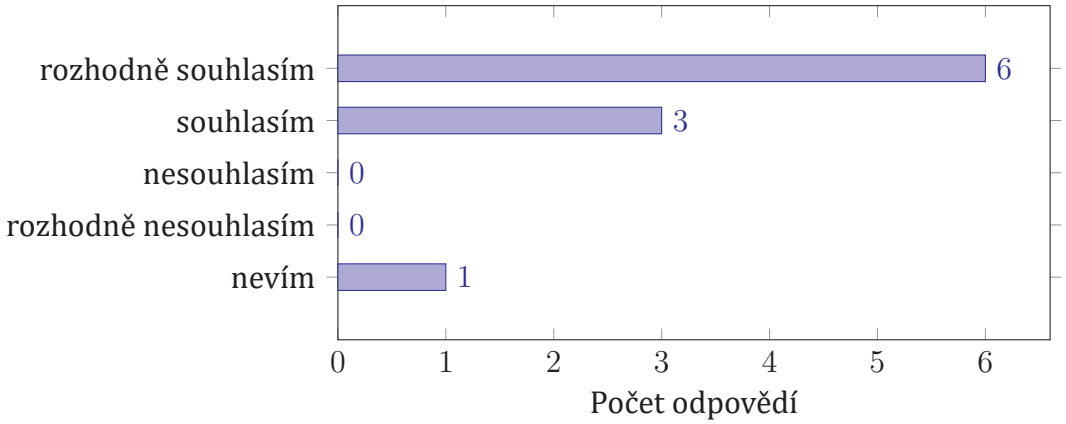
Otázka: Funkce aplikace hodnotím jako:



Otázka: Aplikace přináší nové funkce a budu ji rád(a) používat.



Otázka: Sada funkcionalit aplikace mi přijde zajímavá a dostatečná.



7.3 Budoucí možnosti vylepšení aplikace

Aplikace je v první verzi plně použitelná. Díky použité architektuře a vybraným technologiím je zde však velký prostor pro budoucí rozšíření aplikace o nové funkcionality, které přilákají ještě více uživatelů a zároveň zlepší celkový pocit zpoužívaní aplikace. Níže je uvedeno několik nápadů na nové funkcionality, které je možné do aplikace bez větších potíží implementovat. Některé z těchto nápadů pocházejí i ze zpětné vazby od uživatelů testovací verze.

- **Rozšíření datových struktur o nové položky**
  Většina použitých datových struktur by mohla být rozšířena o nové, zajímavé položky. Například u hráče je možné sledovat jeho výšku, váhu, dále pak pořadí v českém i světovém žebříčku, historii odehraných zápasů, historii týmů, a tak dále. Díky vlastnostem Firebase Realtime Database není přidání nových vlastností žádným problémem, na straně aplikace je pak potřeba pouze pozměnit třídy, které slouží jako modely pro datové struktury tak,

aby s novými položkami počítaly při parsování.

- **Implementace push notifikací**
  Push notifikace je skvělý nástroj, jak uživatele upozornit, že se v aplikaci něco děje, i ve chvíli, kdy uživatel aplikaci aktivně nepoužívá. Rozhodčí by tak mohli například dostávat upozornění o tom, že byli přiřazeni k zápasu a mají se tedy dostavit na kurt. Fanoušci by mohli být informováni o výsledcích jejich oblíbeného týmu, o nových zajímavých článcích, nebo o blížícím se dalším utkání. Push notifikace je možné implementovat do aplikace díky nástroji Firebase Cloud Messaging.

- **Implementace systému volání vrchního rozhodčího**
  Často se v průběhu zápasu stane, že hlavní rozhodčí musí na kurt přivolat vrchního rozhodčího. Většinou se tak děje pomocí zvednutí ruky. V aplikaci by však v budoucnu mohl být implementovaný systém, který by po stisku tlačítka na mobilním zařízení hlavního rozhodčího zobrazil vrchnímu rozhodčímu ve webové aplikaci notifikaci o tom, že je volán na kurt. Ve velkých halách by to jistě usnadnilo komunikaci mezi rozhodčími. Tato funkcionalita by se dala implementovat například pomocí nového uzlu ve Firebase Realtime Database.

- **Navigování uživatele**
  V mobilní aplikaci by bylo možné implementovat obrazovku s informacemi o jednotlivých halách, která by umožňovala mimo jiné i navigování uživatele k hale. Dále by pak bylo možné zobrazit aktuálně hrané zápasy podle dané haly, nikoliv podle utkání.

- **Divácké akce**
  Ve chvíli, kdy aplikaci bude používat velké množství uživatelů, by jistě bylo vhodné implementovat systém pro zobrazování různých diváckých akcí. Uživatel aplikace by byl informován pomocí výše zmíněných push notifikací. Jednalo by se například o různé akce na slevy na vstupné, soutěže o ceny, a tak dále. Pořadatelé jednotlivých utkání by tak mohli do svých hal přilákat větší množství fanoušků. Základy implementace by opět stály na Firebase Realtime Database, doplněné o Firebase Cloud Messaging.

- **Začlenění sponzorských aktivit**
  Naprostá většina týmů se neobejde bez sponzorů. Rozšířením aplikace mezi velké množství uživatelů se otevírá další kanál, kde je možné například zobrazovat loga sponzorů jednotlivých týmů a tím vytvářet reklamu. Na tento způsob je aplikace ve své podstatě připravena a bylo by pouze nutné konzultovat úpravy uživatelského rozhraní s grafikem tak, aby byl výsledek uživatelsky přívětivý. Navázání spolupráce se sponzory klubů by mohlo vést k budoucí přímé finanční podpoře aplikace.

8 Závěr

V první části práce byla provedena analýza aktuálně používaných softwarových řešení pro organizaci a správu turnajů v badmintonu. Z analýzy vyplynulo, že v prostředí badmintonu se používá centrálně pouze jedno softwarové řešení, které ale nesplňuje požadované funkční požadavky. Lokálně jsou v badmintonovém prostředí používany aplikace, které pocházejí od cizích autorů a nejsou připraveny na centralizaci sběru dat. Dále byl představen aktuálně používaný postup při zveřejňování výsledků utkání. Na základě této analýzy bylo rozhodnuto o vybudování kompletně nového softwarového řešení.

V druhé kapitole byly definovány funkční a technické požadavky na nově vznikající softwarové řešení. Byly specifikovány použité technologie a jednotlivé komponenty celého řešení. Dále byly definovány společné datové entity pro všechny komponenty, a také role jednotlivých uživatelů aplikace. Pomocí UseCase diagramů byly specifikovány funkcionality, které jednotlivé komponenty musí implementovat. V práci byla také představena platforma Google Firebase, jejíž produkty Firebase Realtime Database, Firebase Authentication a Firebase Crashlytics byly v práci použity.

První částí navrženého softwarového řešení je klientská mobilní aplikace pro operační systém Android. V rámci práce byl proveden návrh aplikace včetně její architektury a integrace s platformou Google Firebase. Aplikace byla dle požadovaných funkčních specifikací implementována. V práci jsou popsány její jednotlivé komponenty a také klíčové části kódu. Kromě implementace všech požadovaných funkcionalit bylo navrženo a implementováno grafické uživatelské rozhraní.

Pro správu turnajů a vytváření obsahu byla navržena a implementována webová aplikace. K její implementaci byla použita knihovna ReactJS, pro uživatelské rozhraní pak knihovna Bootstrap. V práci je popsána struktura webové aplikace, implementované komponenty a také seznam implementovaných funkcí pro manipulaci s obsahem databáze Firebase Realtime Database.

Navrhovaný systém byl průběžně testován, a to jak v kancelářských podmínkách, tak v reálném nasazení. Pro lepší sledování kvality aplikace byl implementován nástroj Firebase Crashlytics. Mobilní aplikace byla testována nejen uživateli z řad proškolených rozhodčích, ale i veřejností. Nalezené chyby byly ihned

opravovány. V průběhu vývoje byly od uživatelů také sbírány připomínky, které byly vyhodnocovány a relevatní byly v aplikaci zohledněny. Pro další sběr zpětné vazby byl vytvořen dotazník, který je dostupný přihlášeným uživatelům aplikace. V práci jsou také navržena další možná rozšíření aplikace, a to nejen z hlediska rozšíření obsahu, ale i funkcionalit. Aplikace v době odevzdání nebyla zatím z organizačních důvodů publikována, nicméně je na publikaci zcela připravena.

Literatura

[1] ANDROID DEVELOPERS: *ViewModel Overview* [online]. Dostupné z: https://developer.android.com/topic/libraries/architecture/viewmodel, [Cit. 11.3.2019].

[2] ANDROID DEVELOPERS: *Create a List with RecyclerView* [online]. Dostupné z: https://developer.android.com/guide/topics/ui/layout/recyclerview, 2019, [Cit. 1.4.2019].

[3] APPTORNADO GmbH: *Number of Android apps on Google Play | AppBrain* [online]. Dostupné z: https://www.appbrain.com/stats/number-of-android-apps, 2019, [Cit. 1.4.2019].

[4] APPTORNADO GmbH: *Top categories on Google Play | AppBrain* [online]. Dostupné z: https://www.appbrain.com/stats/android-market-app-categories, 2019, [Cit. 1.4.2019].

[5] BANKS, A.; PORCELLO, E.: *Learning React: functional web development with React and Redux*. Sebastopol, CA: O'Reilly Media, 2017, ISBN 978-1-491-95462-1.

[6] ČESKÝ BADMINTONOVÝ SVAZ: *Rozpis soutěží družstev dospělých* [online]. Dostupné z: https://czechbadminton.cz/files/autor/rozpis_soutezi_druzstev_dospelych_18-19_-v2.pdf, 2018, [Cit. 1.4.2019].

[7] ČESKÝ BADMINTONOVÝ SVAZ: *Rozpis soutěží jednotlivců - dospělí* [online]. Dostupné z: https://czechbadminton.cz/files/autor/rozpis_soutezi_dospelych_-_jednotlivci_18-19_v1_0.pdf, 2018, [Cit. 1.4.2019].

[8] ČESKÝ BADMINTONOVÝ SVAZ: *Soutěžní řád* [online]. Dostupné z: https://czechbadminton.cz/files/autor/soutezni_rad_upraveny_14032019.pdf, 2019, [Cit. 1.4.2019].

[9] GOOGLE: *Firebase pricing* [online]. Dostupné z: https://firebase.google.com/pricing/, 2019, [Cit. 6.3.2019].

[10] GOOGLE Inc.: *Firebase products* [online]. Dostupné z: https://firebase.google.com/products/, 2019, [Cit. 1.4.2019].

[11] IT slovník: *JavaScriptová knihovna React (React.js)* [online]. Dostupné z: https://it-slovnik.cz/pojem/react, [Cit. 16.3.2019].

[12] MENDREK, T.; NOVOTNÁ, M.: *Badminton*. Praha: Grada Publishing, a.s., druhé, upravené vyd□n□, 2007, ISBN 978-80-247-2004-3.

[13] STATISTA, Inc.: *Google Play Store: number of apps 2018 | Statista* [online]. Dostupné z: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/, 2018, [Cit. 1.4.2019].

[14] Tamplin, J.: *Firebase is joining Google* [online]. Dostupné z: https://firebase.googleblog.com/2014/10/firebase-is-joining-google.html, 2014, [Cit 1.4.2019].