Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Mobilní aplikace pro sázkaře

Tomáš Neitzel

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Tomáš Neitzel

Informatika

Název práce

Mobilní aplikace pro sázkaře

Název anglicky

Mobile application for punters

Cíle práce

Cílem práce je naprogramování mobilní aplikace, která bude vycházet z UI specifikace vytvořené v mé bakalářské práci s názvem UI specifikace mobilní aplikace pro sázkaře.
Cílem teoretické části práce je vysvětlit sázkařské pojmy, které budou používány v práci a zároveň pojmy, které se týkají tvorby mobilních aplikací. Dílčím cílem teoretické části bude porovnání podobných sázkařských aplikací. Dalším cílem bude analýza technologií pro tvorbu mobilních aplikací a popis problematiky softwarového návrhu aplikací.
Cílem praktické části práce je vytvořit uživatelsky přívětivou aplikaci, která bude sloužit jak pro sázení, tak jako blog pro profesionální sázkaře. V průběhu vývoje aplikace bude prováděno testování funkčnosti, návrhu a designu aplikace potencionálními uživateli. Tato zpětná vazba bude mít výrazný vliv na další vývoj aplikace.

Metodika

Metodika řešené problematiky diplomové práce je založena na studiu a analýze odborných informačních zdrojů, dále na analýze stávajících aplikací a technik při sázení.

Hlavním parametrem pro vytvoření návrhu aplikace v praktické části jsou informace a data sázkařů. Dále analýza technologií, které jsou dostupné k programování mobilních aplikací a také analýza již stávajících aplikací, které jsou určené pro sázení. Primární budou fungující metodiky a jejich úspěšnost, nejpoužívanější sázkové kanceláře nebo například nejvýdělečnější sport. Aplikace bude průběžně upravována na základě odpovědí z dotazníků od sázkařů ve fázi programování i při finalizaci aplikace.

Návrh a vývoj aplikace bude vypracováván v několika fázích. Základem je definování požadavků od uživatelů na aplikaci společně s vizí designu aplikace, která by byla nejvíce přívětivá. K návrhu designu bude využit prototyp UI specifikace, která byla navržena v mé bakalářské práci. Další fází bude programovací část, tedy implementace technologií a jejich následná verifikace funkčnosti. Mezi předpokládané SW a systémy, které budou použity k vývoji patří například Git, Android IDE a také databázový server pro čtení a vyhodnocování výsledků zápasů a evidenci uživatelů.

Doporučený rozsah práce

60–80 stran

Klíčová slova

mobilní aplikace, OS Android, mobilní sázení, sázení na zápasy, profesionální sázkaři, Android IDE, C++, Android Studio, Visual Studio, NetBeans

---

Doporučené zdroje informací

Cohen, Ryan and Tao Wang: GUI Design for Android Apps 1st ed. Edition. US: Apress, 2014. ISBN: 978-1484203835

Darwin, F. Ian: Android Cookbook, 2nd Edition. Sebastopol, CA: O'Reilly Media, Inc., 2017. ISBN: 9781449374488

Cheng, Fu: Build Mobile Apps with Ionic 4 and Firebase. Berkeley, CA: Apress, 2018. ISBN: 978-1-4842-3774-8

McWherter, Jeff and Scott Gowell: Professional Mobile Application Development. Bengalúru, Indie: John Wiley & Sons, Inc., 2012. ISBN: 978-1-118-22842-5

Rogers, Rick, John Lombardo, Zigurd Mednieks and Blake Meike: Android Application Development. Sebastopol, CA: O'Reilly Media, Inc., 2009. ISBN: 978-0-596-52147-9

Předběžný termín obhajoby

2023/24 LS – PEF

Vedoucí práce

Ing. Dana Vynikarová, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 28. 11. 2023

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 9. 2. 2024

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 30. 03. 2024

Čestné prohlášení

Prohlašuji, že svou diplomovou práci „Mobilní aplikace pro sázkaře“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 31.3.2024 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Daně Vynikarové, Ph.D. za odborné rady při vypracovávání práce, cenné připomínky a za vedení mé práce.

.

Mobilní aplikace pro sázkaře

Abstrakt

Tato závěrečná je zaměřena na naprogramování mobilní aplikace, kterou lze využít pro sázky na sportovní zápasy a zároveň aplikaci lze využít pro profesionální sázkaře jako blog. V práci jsou vysvětleny základní sázkařské pojmy, které jsou používány v mobilní aplikaci a zároveň pojmy, které se týkají tvorby mobilních aplikací. Dále jsou zde porovnány podobné sázkařské aplikace a analyzovány technologie pro tvorbu mobilních aplikací spolu s popisem problematiky softwarového návrhu aplikací.

V praktické části práce je vytvořena mobilní aplikace společně s databázovým serverem, který je použit od Microsoft Azure s dočasným předplatným zdarma. Připojení mobilní aplikace na databázi poskytuje ručně nakonfigurované API. Aplikace disponuje evidencí uživatelů a jejich rolí, čímž rozlišuje rekreační a profesionální sázkaře. Dále umožňuje profesionálním sázkařům vytvořit článek, který je pouze zajímavostí nebo sepsat konkrétní strategii, kterou se mohou sázkaři inspirovat při své sázce. Uzavírání sázek je umožněné přímo v aplikaci na vybrané zápasy. Aplikace obsahuje panel nastavení pro administrátora, pomocí kterého může manuálně některému z uživatelů změnit roli. V průběhu vývoje je aplikace mnohokrát testována sázkařskou skupinou lidí, která poskytuje zpětnou vazbu k návrhu, designu a funkčnosti.

Klíčová slova: mobilní aplikace, OS Android, mobilní sázení, sázení na zápasy, profesionální sázkaři, Android IDE, C++, Android Studio, Visual Studio, NetBeans

Mobile application for punters

Abstract

The diploma thesis is focused on programming mobile application that can be used for betting on sports matches and at the same time the application can be used as a blog for professional bettors. The thesis explains the basic betting terms that are used in the mobile application and at the same time terms that relate to the creation of mobile applications. Furthermore, similar betting applications are compared and technologies for the creation of mobile applications are analysed together with a description of the issue of software application design.

In the next part, the focus is on the mobile application itself together with the database server, which is used from Microsoft Azure with a temporary subscription for free. Connecting the mobile application to the database provides a configured API. The application has registered users and their roles, thanks to which it distinguishes recreational and professional bettors. Furthermore, it allows professional bettors to write an article that is only of interest or to write a specific strategy that bettors can be inspired by in their bet. Placement of bets is possible directly in the application for selected matches. The application contains settings for the administrator, who can manually change the role of any of the users. In the process, the application is tested many times by friendly development people who provide feedback on design, design and functionality.

Keywords: mobile application, OS Android, mobile betting, betting on matches, professional bettors, Android IDE, C++, Android Studio, Visual Studio, NetBeans

Obsah

1 Úvod **11**

2 Cíl práce a metodika **12**

2.1 Cíl práce 12

2.2 Metodika 12

3 Teoretická východiska **13**

3.1 Kurzové sázení 13

3.1.1 Historie a evoluce 13

3.1.2 Sázkařské pojmy 14

3.1.2.1 Void 14

3.1.2.2 Cashout 14

3.1.2.3 Value 15

3.1.2.4 Návratnost investic 15

3.1.2.5 Dvojitá šance 15

3.1.3 Sázkové společnosti 16

3.1.4 Profesionální sázkaři 16

3.2 Tvorba aplikace 17

3.2.1 Uživatelské rozhraní 17

3.2.1.1 Fáze návrhu UI specifikace 17

3.2.1.2 Diagramy 20

3.2.2 Programovací jazyky 22

3.2.2.1 Kategorizace programovacích jazyků 22

3.2.2.2 XML 24

3.2.2.3 Java 28

3.2.2.4 C++ 32

3.2.3 Vývojové prostředí 32

3.2.3.1 Android studio 32

3.2.3.2 NetBeans 37

3.2.3.3 Visual Studio 37

3.2.4 Mobilní operační systémy 38

3.2.4.1 Android 38

3.2.5 Metodologie vývoje aplikací 39

3.2.5.1 Vodopádový model 39

3.2.5.2 Agilní model 41

3.3 Vybrané technologie pro tvorbu aplikace 42
3.3.1 Microsoft Azure databáze 42
3.3.2 Spring framework 42
3.3.3 Odds API 43
3.4 Analýza podobných sázkařských aplikací 44
3.4.1 Tipsport 44
3.4.1.1 Kurzová nabídka 44
3.4.1.2 Blog 44
3.4.2 Fortuna 45
3.4.2.1 Kurzová nabídka 45
3.4.2.2 Blog 45
3.4.3 Závěr analýzy 46
4 Vlastní práce 47
4.1 Požadavky na aplikaci 47
4.1.1 Funkční požadavky 47
4.1.2 Nefunkční požadavky 48
4.2 Návrh uživatelského rozhraní 49
4.2.1 Návrh domovské obrazovky 49
4.2.2 Návrh sekce strategie 50
4.2.3 Návrh sekce články 51
4.2.4 Návrh sázky na zápas 52
4.2.5 Návrh sekce vsazených tiketů 53
4.2.6 Use Case diagram 54
4.2.7 Diagram tříd 56
4.3 Struktura databáze 57
4.3.1 Evidování uživatelů 57
4.3.2 Definování příchozích zápasů z API 58
4.3.3 Stáhnutí příchozích zápasů z API 59
4.3.4 Evidování sázek 60
4.4 Realizace aplikace 61
4.4.1 Design mobilní aplikace 61
4.4.1.1 Design domovské obrazovky 62
4.4.1.2 Design sekce přihlášení a registrace 63
4.4.1.3 Design sekce strategie 64
4.4.1.4 Design sekce blog 65
4.4.1.5 Design sekce nadcházejících zápasů a sázek na zápas 66
4.4.1.6 Design profilu a vsazených tiketů 67

4.4.2 Logika zobrazování dynamického obsahu 68
4.4.3 Funkcionality aplikace 69
4.4.3.1 Registrace a přihlášení 69
4.4.3.2 Role v aplikaci 70
4.4.3.3 Blog 70
4.4.3.4 Sázení na zápasy 72
4.4.3.5 Profil 73
4.4.3.6 Upravení rolí uživatelům 74
4.4.3.7 Zobrazení nadcházejících zápasů 75
4.4.3.8 Zobrazení výsledků odehraných zápasů 75
4.4.3.9 Filtr vlastních strategií a článků 75
4.5 Testování aplikace 76
4.5.1 Emulace Android systému 76
4.6 Back end API endpoints 77
4.6.1 Fetchování dat z Odds API 81
4.6.2 Ukázka servisu strategií 82
4.7 Požadavky pro spuštění aplikace 83
4.7.1 Softwarové požadavky 83
4.7.1.1 Android Studio 83
4.7.1.2 Databázový server 83
4.7.1.3 IntelliJ Idea 83
4.7.2 Hardwarové požadavky 84
5 Výsledky a diskuse 85
5.1 Porovnání podobné sázkařské aplikace s řešením autora 85
5.1.1 Aplikace sázkové kanceláře Tipsport 85
5.1.2 Porovnání Tipsport a ThomasBet aplikace 86
5.2 Výsledky dotazníku z průběhu vývoje 87
6 Závěr 88
7 Seznam použitých zdrojů 89
8 Seznam obrázků, tabulek, grafů a zkratek 92
8.1 Seznam obrázků 92
8.2 Seznam tabulek 93
8.3 Seznam grafů 93
8.4 Seznam použitých zkratek 93
9 Přílohy 94

1 Úvod

Aktuálně v sázkařském světě začíná přibývat profesionálních sázkařů, kteří nabízí své placené tipy, avšak poskytují také strategie zdarma pro jejich sledovatele. Ačkoliv je tento koncept zdarma, uživatel musí neustále sledovat sociální sítě, aby některou z těchto strategií nezmeškal. Shromážděním všech veřejně dostupných strategií usnadní autor aplikace práci sázkaři, jelikož budou dostupné stále na stejném místě ve vypracované aplikaci. Zároveň sázkař nebude muset otevírat novou aplikaci, ale může si vsadit přímo v té, kde se o strategii dozvěděl.

Dostupné sázkařské aplikace českých sázkových kanceláří nenabízí možnost vytvářet analýzy nebo strategie s výjimkou Tipsportu, který ale umožňuje vytváření analýz pro všechny uživatele, čímž není zaručena důvěryhodnost.

Hlavní motivací bylo vytvořit alternativní aplikaci k aplikaci od sázkové kanceláře Tipsport, která je nejpopulárnější sázkařskou aplikací, avšak nespolupracuje s profesionálními sázkaři. Programování obecně je pro autora práce výzvou, nicméně by rád zlepšil své schopnosti v tomto směru, především kvůli obecnějšímu pohledu na informatiku a rozšíření obzorů v budoucnosti.

Diplomová práce se rozděluje na dva větší celky, teoretickou a praktickou část. V teoretické části jsou vysvětleny pouze nejdůležitější sázkařské pojmy, které se vyskytují v aplikaci. Dále byly porovnány podobné sázkařské aplikace s autorovou aplikací a analyzovány technologie pro tvorbu mobilních aplikací spolu s popisem problematiky softwarového návrhu aplikací.

Praktická část se věnuje mobilní aplikaci společně s databázovým serverem a ručně nakonfigurované API. Aplikace disponuje evidencí uživatelů včetně jejich rolí a umožňuje profesionálním sázkařům vytvořit článek nebo strategii. Dále je umožněno sázení přímo v aplikaci na vybrané zápasy pomocí vložení peněz na účet a aplikace obsahuje nastavení rolí pro administrátora.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem práce je naprogramování mobilní aplikace, která bude vycházet z UI specifikace vytvořené v mé bakalářské práci s názvem UI specifikace mobilní aplikace pro sázkaře.

Cílem teoretické části práce je vysvětlit sázkařské pojmy, které budou používány v práci a zároveň pojmy, které se týkají tvorby mobilních aplikací. Dílčím cílem teoretické části bude porovnání podobných sázkařských aplikací. Dalším cílem bude analýza technologií pro tvorbu mobilních aplikací a popis problematiky softwarového návrhu aplikací.

Cílem praktické části práce je vytvořit uživatelsky přívětivou aplikaci, která bude sloužit jak pro sázení, tak jako blog pro profesionální sázkaře. V průběhu vývoje aplikace bude prováděno testování funkčnosti, návrhu a designu aplikace potencionálními uživateli. Tato zpětná vazba bude mít výrazný vliv na další vývoj aplikace.

2.2 Metodika

Metodika řešené problematiky diplomové práce je založena na studiu a analýze odborných informačních zdrojů, dále na analýze stávajících aplikací a technik při sázení.

Hlavním parametrem pro vytvoření návrhu aplikace v praktické části jsou informace a data sázkařů. Dále analýza technologií, které jsou dostupné k programování mobilních aplikací a také analýza již stávajících aplikací, které jsou určené pro sázení. Primární budou fungující metodiky a jejich úspěšnost, nejpoužívanější sázkové kanceláře nebo například nejvýdělečnější sport. Aplikace bude průběžně upravována na základě odpovědí z dotazníků od sázkařů ve fázi programování i při finalizaci aplikace.

Návrh a vývoj aplikace bude vypracováván v několika fázích. Základem je definování požadavků od uživatelů na aplikaci společně s vizí designu aplikace, která by byla nejvíce přívětivá. K návrhu designu bude využit prototyp UI specifikace, která byla navržena v mé bakalářské práci. Další fází bude programovací část, tedy implementace technologií a jejich následná verifikace funkčnosti. Mezi předpokládané SW a systémy, které budou použity k vývoji patří například Git, Android IDE a také databázový server pro čtení a vyhodnocování výsledků zápasů a evidenci uživatelů.

3 Teoretická východiska

3.1 Kurzové sázení

Kurzové sázení je druh hazardní hry, při které uživatel vsadí určitý finanční obnos na předem vypsanou sázku s vidinou výdělku. Pokud bude předpověď naplněna, bude mu vyplacena výhra v závislosti na předepsaném kurzu sázkovou kanceláří. Čím je kurz vyšší, tím se také zvyšuje potencionální výhra, ale zároveň platí, že čím je vyšší kurz, tím je větší riziko prohry spolu s menší pravděpodobností úspěchu. Vsadit lze v aktuální době na široké spektrum událostí, které už dávno nemusí být spojené pouze se sportem. Kromě klasického sázení na fotbal, tenis, hokej a další sporty lze vsadit i na vítěze voleb do senátu, budoucího prezidenta, vítěznou politickou stranu a další. (1)

Jelikož při sázení dochází k výdělku, musí se odvádět daň příslušnému státu. Každý stát má daň nastavenou jinak a pro sázení platí ve většině zemí speciální podmínky. V České republice máme tyto podmínky regulované zákonem, konkrétně Zákon č. 186/2016 Sb. Tento zákon nahradil v roce 2017 jeho dřívější verzi z roku 1990. Odváděná daň státu v České republice je v něm stanovena tak, že pokud sázkař za kalendářní rok má čistý výdělek, tedy součet výher a proher pod 1 milion korun, nemusí odvádět státu daň žádnou. Pokud je jeho čistý zisk vyšší než 1 milion korun, tak odvádí 15 % srážkovou daň, tedy až do prosince 2023. Od 1.1.2024 přišla novela již zmiňovaného zákona z roku 2017, která stanovuje novou hranici na 50 000 korun. (2)

Pokud tedy roční čistý zisk sázkaře bude vyšší než tato částka, sázkař má povinnost odvést daň státu a uvést tuto skutečnost na daňovém prohlášení. Do této částky je nutno zahrnout všechnu aktivitu na hazardním hrách, ovšem každá kategorie se počítá samostatně. Tudíž kromě kurzového sázení se musí danit také loterie, losy, technické hry a živé hry, přičemž každá kategorie má svůj limit, který se nesčítá s ostatními limity jiných hazardních her.

3.1.1 Historie a evoluce

Sázky vznikly již před několika tisíci let, kdy první nedoložitelné zmínky o sázkách pocházejí z Mezopotámie, kdy lidé vsázeli, jaká hodnota padne na kostkách. (3)

První doložitelné zmínky týkající se sázení pochází z 15. století, kdy šlo konkrétně o loterii. (4)

Začátkem 20. století začínají vznikat první závody na koních, které začaly lákat sázení na vítěze. V této době to bylo stále na hranicích zákona. První sázkařský byznys vznikl kolem roku 1925 americkou mafií v době, kdy začínal být baseball obrovsky oblíbeným sportem ve Spojených státech. Samozřejmě tyto sázky byly za hranicí zákona a nebyly legální, což zde zůstalo dodnes až na pár výjimek jako například v Nevadě, konkrétně Las Vegas. Mezi lety 1970 a 1990 bylo již mnoho bookmakerů na trhu, kteří přijímali sázky na sportovní události. V těchto letech probíhal také rychlý vývoj internetu a začalo se uvažovat, zdali nebude výhodnější přesunout sázky tam. V roce 1994 přijal stát Antigua zákon, který umožnil zakládat internetové sázkové kanceláře. Tento krok inspiroval mnoho dalších zemí, v kterých začaly rovněž vznikat sázkové kanceláře s žádostí o vstup na internet. (5)

3.1.2 Sázkařské pojmy

V této kapitole budou popsány nejčastěji skloňované sázkařské pojmy, které související s praktickou částí, jelikož potencionální sázkařský slovník je mnohem rozsáhlejší. Jak již bylo zmíněno výše, tak kurzové sázení nabízí mnoho druhů sázek a vysvětlit všechny tyto sázky by vyžadovalo samostatnou práci na toto téma.

3.1.2.1 Void

Void (neboli vada) je neutrálním vyhodnocením sázky, jelikož sázkař při ní neztrácí vklad, avšak o něj ani nepřichází. Lze také tento stav vyjádřit jako výhra s kurzem 1, tedy sázkař vyhrál částku, která se rovná jeho vkladu.

Tento stav může nastat v mnoha případech, proto budou vyjmenovány jen ty nejčastější. Prvním a nejčastějším případem vady je zrušení, odložení nebo přesunutí zápasu na neurčeno. Druhým příkladem je druh sázky zvané výhra bez remízy. V takovém případě může dojít k vadě, pokud nastane v zápase remíza. Posledním uvedeným případem je chyba na straně provozovatele sázek, jelikož pokud by byla sázka vypsána v moment, kdy byl již její stav naplněn, sázka bude vyhodnocena jako vada. Tento scénář může nastat, pokud by sázkař vsadil, že první branku vstřelí domácí tým, který ji již vstřelil. (6)

3.1.2.2 Cashout

Cashout (neboli vyplacení sázky předčasně) nastává v moment, kdy zápas ještě nebyl ukončen a sázkař si přeje již výhru vyplatit. To je však v České republice možné pouze, pokud sázkař vsadil na více zápasů než jeden a zároveň platí, že byl alespoň jeden

zápas již ukončen a vyhodnocen. Při této příležitosti se mění kurz v závislosti na již odehraných a neodehraných příležitostí a zároveň podle změny kurzů neodehraných příležitostí.

Některé sázkové kanceláře nabízí také částečné vyplacení výhry, což umožňuje uzavření pouze jedné příležitosti a ponechání ostatních sázek. (7)

3.1.2.3 Value

Value (neboli hodnota sázky) v sázkařském pojetí označuje sázku, která má z pohledu sázkaře větší hodnotu, než ji dává provozovatel sázek. Tento pojem je často uváděn profesionálními sázkaři, jelikož ti vyhledávají pouze tyto sázky za účelem maximalizace čistého zisku. Hodnota sázky záleží na výši nastaveného kurzu. Pokud tento kurz bude příliš nízký, tak sázkař uváží, zdali pro něj má dostatečnou hodnotu na takovou příležitost vsadit. Pokud je kurz vysoký, zjišťuje se jeho hodnota v porovnání s rizikem sázky. Pokud je riziko razantně vyšší než hodnota sázky, není výhodné na danou příležitost vsadit. (8)

3.1.2.4 Návratnost investic

Pojem ROI (*return of investments*, česky návratnost sázek) je často mnoha sázkaři opomíjen, avšak každý sázkař by mu měl věnovat pozornost. Tímto pojmem lze zjisti, zdali lze sázkař z kurzových sázek získává zisk nebo naopak ztrátu. Výsledná hodnota se udává v procentech a označuje čistý zisk nebo čistou ztrátu v poměru všech vsazených příležitostí. Nejjednodušší interpretací výpočtu této hodnoty bude následující případ. Sázkař na začátku měsíce vložil na sázecí platformu 10 000 kč. Na konci měsíce má na svém hráčském kontě 20 000kč, a tudíž jeho ROI je 200 %. (9)

3.1.2.5 Dvojitá šance

Dvojitou šancí se označují takové příležitosti, při kterých nastává výhra v přesně dvou scénářích. Pokud je dvojitá šance vsazena ve prospěch domácího týmu, výhra nastává při výhře domácího týmu a zároveň při remíze. Pokud vyhraje hostující tým, sázka není výherní. (10)

3.1.3 Sázkové společnosti

Sázková společnost zprostředkovává kurzové sázení. V České republice má sázkař na výběr hned z několika společností, avšak musí volit takové společnosti, které mají aktivní smlouvu s Českou republikou. Na výběr má ze společností Tipsport, Fortuna, Betano, Chance, SYNOT TIP, Sazkabet a relativně nové společnosti MerkurXtip. (11)

Bohužel Česká republika neumožňuje sázení u nejznámějších celosvětových společností, kterými jsou například Bet365, Stake nebo 1xBet. (12)

Každá sázková kancelář nabízí různé vstupní bonusy, spektrum sázek, kurz, možnosti výběru peněz, nebo uživatelskou podporu. Všechny tyto faktory ovlivňují, kterou sázkovou společnost si uživatel vybere. Nejvíce sázkaře ovlivňuje spektrum sázek a kurz, jelikož to jsou hlavní faktory kurzového sázení.

3.1.4 Profesionální sázkaři

Profesionální sázkař je velice relativní pojem, jelikož každý se může považovat za profesionálního sázkaře. Většinou každý sázkař, který se veřejně prezentuje jako profesionální, tak poskytuje takzvanou verifikaci svých tipů, čímž dokládá, že sázky, které zveřejňuje jsou skutečně vsazené a nejedná se o podvod. Verifikace tipů je třetí strana, která má přístup do databáze všech sázkových kanceláří a může ověřit, zdali sázka byla opravdu vsazena u některé u sázkových společností. Pokud sázkař neposkytuje verifikaci svých tipů, ztrácí důvěryhodnost a nelze jej považovat za profesionálního sázkaře, jelikož nelze potvrdit, jestli byly tipy opravdu vsazeny. Hranice úspěšnosti, kdy lze považovat sázkaře za profesionála je opět relativní a záleží na subjektivním uvážení každého člověka. (13)

Obecně lze konstatovat, že pokud sázkaři prodávají své tipy obyčejným sázkařům s vysokou úspěšností, důvěrou a mají k dispozici verifikaci, můžeme je označovat za profesionálního sázkaře na základě jejich výsledků.

Jak již bylo zmíněno v předchozím odstavci, existují profesionální sázkaři, kteří prodávají své tipy za určitý finanční obnos. Tato nabídka má však zpravidla několik omezení, které musí sázkař dodržet, aby se dostal do stejného zisku jako profesionální sázkař. Nejčastějším omezením je základní kapitál, uživatelský účet u více sázkových společností z důvodu maximalizace zisku na základě nabízeného kurzu nebo komunikační kanál podle výběru profesionálního sázkaře. (13)

3.2 Tvorba aplikace

3.2.1 Uživatelské rozhraní

User interface neboli uživatelské rozhraní označuje veškerou část viditelnou uživatelem na první pohled. Uživatel pomocí uživatelského rozhraní interaguje mezi uživatelem a systémem. Pomocí prvků, které mu jsou zobrazeny odesílá požadavky na systém, který mu na základě programovacího kódu v pozadí generuje odpovědi. Zásadními prvky správného designu je jednoduchost a přehlednost, pomocí kterých lze hodnotit kvalitu uživatelského rozhraní. Uživatelské rozhraní se používá k manipulaci a kontrole stroje z pozice člověka, kdy stroj informuje člověka na základě informací, které uživatel poskytl a stroj pomocí kódu vyhodnotil. Toto rozhraní často rozhoduje o tom, zdali uživatel nadále bude systém využívat nebo zvolí jiný. (14)

3.2.1.1 Fáze návrhu UI specifikace

Každý projekt by se měl skládat z předem definovaného plánu práce, který definuje systém a kroky v kterých se bude projekt vypracovávat. Předem definované kroky vedou k úspěšnému dosažení cíle a kompletaci projektu. (15)

3.2.1.1.1 Uživatelské cíle, porozumění uživatelům

Základním prvkem je vždycky uživatel, pro kterého se systém navrhuje a vyvíjí, proto je potřeba porozumět, co vlastně uživatel od systému očekává a jaké má nároky. Současně s tím je potřeba si definovat, pro jakého uživatele je systém vyvíjen. Tudíž je potřeba si definovat ideálního uživatele systému a spolu s ním jak občasného uživatele, tak také uživatele, který systém nikdy nevyužije.

Definování ideálního uživatele je proces, při nímž si návrháři vymyslí fiktivní personu, které sestaví charakteristické rysy, dovednosti a životní styl. Tři persony jsou definovány, konkrétně primární uživatel, sekundární uživatel a antipersona, což je uživatel, který nikdy systém nebude využívat. Naopak primární persona je uživatel, který systém bude vyhledávat a bude pro něj nejlepší možnou variantou, jelikož bude poptávat přesně daný systém. Sekundární persona je uživatel, který systém sice nebude vyhledávat, ale pokud se mu dostatečně zalíbí, bude jej občasně využívat, avšak nikoliv na denní bázi. (16)

Nejlepší zpětnou vazbou od uživatelů jsou dotazníky, kdy po vyzkoušení systému nebo shlédnutí návrhu poskytne uživatel většinou jak pozitivní, tak i negativní zpětnou

vazbu. Každá zpětná vazba může systém posunout správným směrem a pomoci při jeho vývoji. (15)

3.2.1.1.2 Výzkum

Při výzkumu je nejdůležitější částí prozkoumat aktuální trh a analyzovat konkurenci, popřípadě zkusit jejich systémy a snažit se je vylepšit. Důležitou součástí je také originalita, pokud je systém dostatečně originální a jedinečný, uživatelé se budou vracet. Pokud je produkt dostatečně jedinečný, pak jej lze obohacovat o funkce, které budou jedinečné a uživatel je nikde jinde nedostane. (15)

3.2.1.1.3 Návrh UI specifikace

Další fází je náčrt systému, přičemž ideální forma je větší bíla pracovní plocha, na kterou se definují všechny shromážděné informace a poté se navrhuje systém. Rozmístění prvků, aby byly pro uživatele dostatečně intuitivní, jednoduchá navigace, jedinečný a jednoduchý systém. V této fázi se zapojují všichni členové týmu a prezentují své vlastní návrhy, které ostatní vyhodnocují, tuto situaci také nazvat anglickým pojmem *brainstorming*. (15)

Pro tvorbu UI specifikace se využívá několik druhů diagramů, například Use case, který zobrazuje funkcionality systému a definuje, kdo tyto funkcionality vyvolává. Dalšími možnostmi jsou diagram tříd, diagram chování a mnoho dalších. (17)

3.2.1.1.4 Design

Oživení projektu pomocí barev, druhů písma a dodávání unikátnosti z grafického pohledu. Grafický design má však svá pravidla, jako například zvolení správných barev, které jsou poutavé pro oči a nasměrují uživatele na nejdůležitější bod stránky, dále zvolení správného písma, velikosti a umístění. Grafický tým však musí spolupracovat s týmem, který následuje po nich, vývojářským týmem, jelikož ne všechny návrhy jsou lehce stvořitelné, někdy dokonce nemožné. A opět je přínosné, pokud design zhodnotí samotný uživatel, který poskytne zpětnou vazbu, co se mu na grafickém návrhu líbí a co nikoliv. (15)

3.2.1.1.5 Implementace

Poslední návrhovou částí je implementace, kdy se dodělávají poslední funkcionality a opravují se vzniklé chyby, před samotnou implementací na finální destinaci. Sestavují se poslední kódy a vzniká finální *user experience* před samotnou implementací. Po

implementaci je vhodné vyzkoušet znovu všechny funkcionality a snažit se přijít na potencionální nežádoucí chování před ostrým spuštěním mezi uživatele. (15)

3.2.1.1.6 Zhodnocení a úpravy

Posledním krokem, který je často nekonečný je opravování chyb a reakce na zpětnou vazbu uživatelů. Je vhodné přímo do systému implementovat dotazník, kde může uživatel poskytnout hodnocení produktu nebo navrhnout jeho vylepšení. Právě vylepšení je nekonečně dlouhá trať, jelikož se často objeví nápad, který systém nasměruje správným směrem. I po čase, kdy nedošlo k žádným chybám je vhodné, aby se vývojáři vrátili k systému a znovu zkontrolovali, zdali funguje správně a je stále stejně uživateli oblíben, jako při jeho spuštění. (15)

3.2.1.2 Diagramy

Diagramů, které se využívají ve fázi návrhu UI specifikace je velké množství. Z tohoto důvodu budou v této kapitole popsány jen vybrané diagramy.

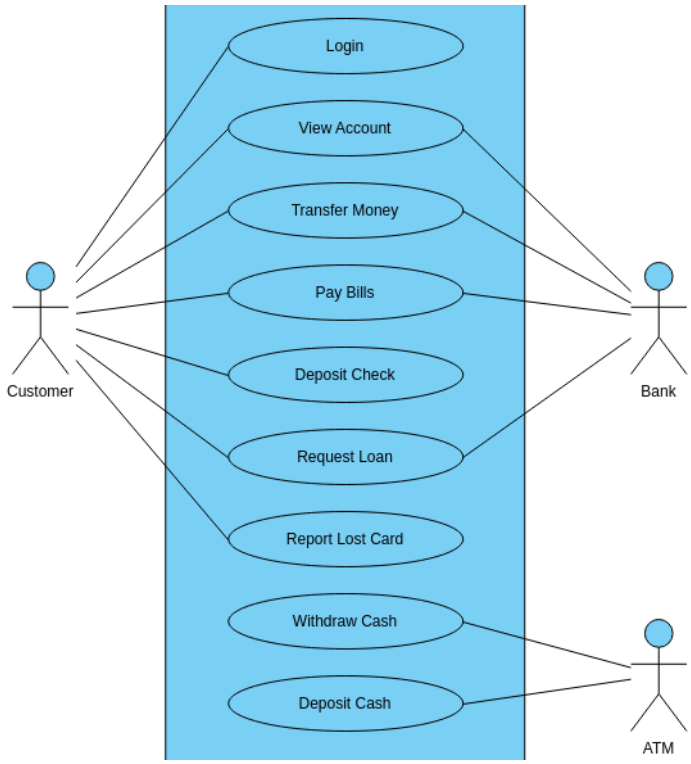
Diagramy umožňují zachycení určitých aspektů systému a zobrazují je v grafickém znázornění. Hlavním cílem diagramů je usnadnit porozumění a komunikaci složitých systémů. Diagramy mohou zahrnovat různé typy informací, jako jsou struktura, funkce, vztahy, procesy nebo datové modely.

3.2.1.2.1 Diagram případu užití

Diagram případu užití je znám také pod pojmem *Use Case* diagram, který zobrazuje chování systému. Hlavním účelem diagramu je popis funkcionalit systému, které uživatel od systému očekává. Diagram definuje, které funkcionality má systém obsahovat, avšak nedefinuje, jakým způsoben to bude provedeno, pouze aktéry. (17)

Diagram je z těchto důvodu často využíván jako první návrhový diagram, který se realizuje při návrhu UI specifikace.

Na níže přiloženém obrázku 1 je zobrazen Use Case diagram pro mobilní bankovní aplikaci. Aktéry jsou uživatel, banka a ATM. Uvedenými případy užití jsou přihlášení, zobrazení účtu, převod peněz, placení účtů, vložení šeku, žádost o půjčku, nahlášení ztracené karty, výběr hotovosti a vložení hotovosti.

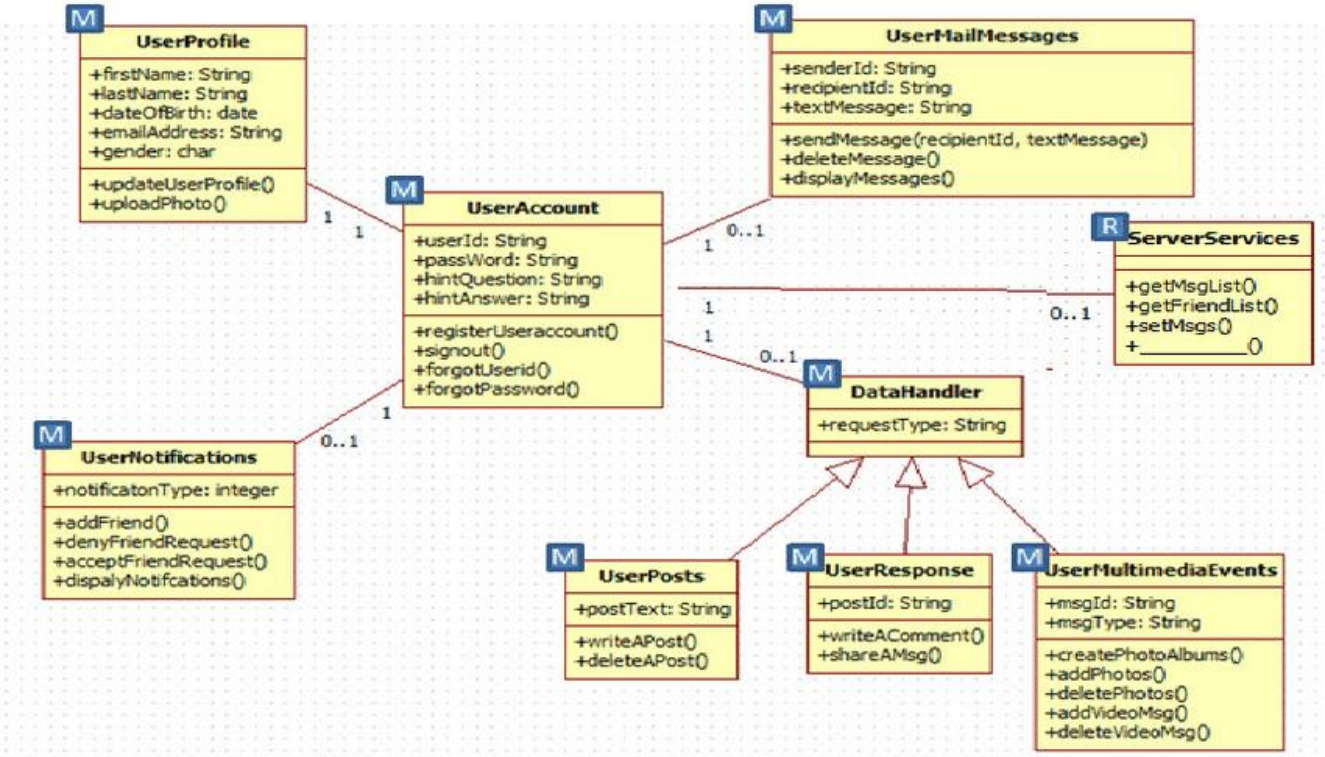


Obrázek 1 Use Case diagram (54)

3.2.1.2.2 Diagram tříd

Diagram tříd se využívá pro zachycení struktury tříd systému a vztahů mezi nimi. Na obrázku lze vidět diagram tříd zachycující strukturu mobilní aplikace GSN. Hlavním komponentem diagramu tříd jsou třídy, které obsahují název, atributy, operace a vazby s ostatními třídami. Vazby mohou být 0..1, 1..1, 0..* a 1..*.

Na obrázku níže lze vidět vazby 1:1, která symbolizuje, že právě jedna třída je v instanci k druhé třídy právě jednou. Příkladem z níže uvedeného obrázku je uživatelský účet, který obsahuje 1 uživatelský profil. Druhou používanou vazbou je 0..1, která je na obrázku mezi třídami uživatelského účtu a uživatelskou notifikací. Uživatelský účet nemusí mít žádnou notifikaci, ale může jich mít také více. (18)



Obrázek 2 Diagram tříd (55)

3.2.2 Programovací jazyky

Programovací jazyk je set instrukcí, které definují počítači činnost, kterou má vykonat. Používá se pro psaní programů nebo aplikací, které poskytují manipulaci s počítačovým systémem. Programovací jazyky mají svoji strukturu neboli syntaxi, tedy způsob a řád, kterým se programuje kód, který je následně odeslán počítači pro vypracování.

Programovací jazyky se v posledních letech neustále vyvíjejí rapidním krokem a přidávají se nové možnosti, funkcionality a rozšířené knihovny k implementaci. (18)

Programovací jazyky lze kategorizovat několika způsoby, proto bude níže popsáno jen nejčastěji používané dělen.

3.2.2.1 Kategorizace programovacích jazyků

Jak již bylo výše zmíněno, tato kapitola by mohla být mnohem obsáhlejší, jelikož způsobů a druhů kategorizace programovacích jazyků je několik. Nejvíce obecnou kategorizací je dělení na základě účelu kódu. V takovém případě dělíme programovací jazyky na *front end* a *back end*. Dále je můžeme dělit podle druhu programování na procedurální, funkční, objektové nebo logické. Podle účelů kódu na statické či dynamické programování a mnoho dalších rozdělení. (19)

3.2.2.1.1 Front end

Z hlediska programování *front end* označuje kód, který vykreslí uživateli interaktivní rozhraní, pokud si zobrazí webovou stránku nebo aplikaci. Tuto část kódu lze také nazvat klientskou částí, jelikož veškeré prvky kódu jsou zobrazeny uživateli, ať už pro interakci nebo pouze zobrazení. Hlavními požadavky pro vzhled jsou jednoduchost, rychlost a bezpečnost. Všechny tyto parametry uživatel od návštěvy stránky nebo aplikace očekává. Do této části se zahrnují tlačítka, navigace, obrázky, animace, vyhledávací pole a mnoho dalších prvků. (20)

Hlavními jazyky pro sestavení *front end* jsou značkovací jazyky HTML a XML, a programovací jazyk Java a stylistický jazyk CSS. Kombinací těchto jazyků lze zobrazit příjemné prostředí pro uživatele, které bude obsahovat intuitivně rozmístěné prvky a příjemný grafický design.

První jmenovaný programovací jazyk HTML je kostrou celého obsahu, pokud je tvořen webový obsah. Jedná se o prvotní vrstvu, kterou uživatel vidí. Má za úkol sestavit kostru celého projektu pomocí tagů (elementů) a prvků. Základní kostrou je hlavička, která

obsahuje znakové kódování a určuje světový jazyk, kterým bude výstup prezentován. Dále v ní můžeme najít jméno stránky nebo meta tag, který poskytuje responzivní design na mobilní zařízení. Poté následuje tělo stránky, které obsahuje veškeré elementy, které budou na stránce zobrazeny jako například nadpisy, menu, články a další. Posledním prvek v kostře stránky je patička, která většinou obsahuje údaje o copyrightu společně s užitečnými odkazy nebo sekundární navigační menu. (21)

Poslední zmíněný jazyk CSS se používá pro grafické zpracování webové stránky a její stylizaci. Používá se pro správné umístění prvků, definuje písmo a jeho styl, pozadí stránky a její části. CSS dodává stránce ten správný grafický vzhled, aby stránka nebyla pouze bílé pozadí s černým textem a odkazy. (22)

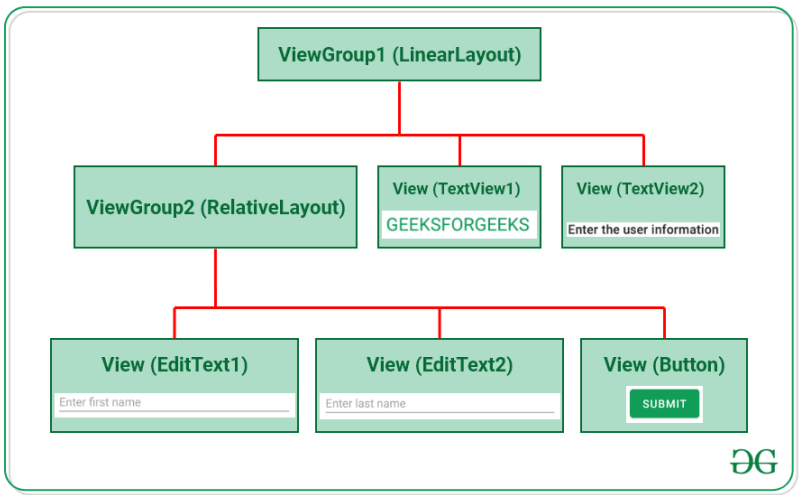
3.2.2.1.2 Back end

Z hlediska programování je *back end* přesným opakem *front end* kódu. Avšak ani jednu zmiňovanou část nelze opomenout, jelikož jsou na sobě závislé. *Back end* kód nikdy neuvidí uživatel, takže pro něj nemá žádný význam a *front end* kód nebude mít žádnou funkcionalitu bez *back end* kódu a budou to pouze rozmístěné prvky, které sice budou vypadat hezky, ale nebudou nic dělat. Tuto část kódu nemůže uživatel vidět, jelikož komunikace probíhá mezi aplikací nebo webem a serverem. Obsahuje například ukládání a procesování dat, komunikace s databází a logiku kódu. Jako příklad lze uvést systém evidence uživatelů, kdy uživatel na *front end* části vyplní formulář, který se následně pomocí *back end* kódu odešle na server, kde se účet zaeviduje do databáze a požadavek se vrací opět k uživateli, který je následně přihlášen do systému a má vytvořený účet pro pozdější přihlášení. (23)

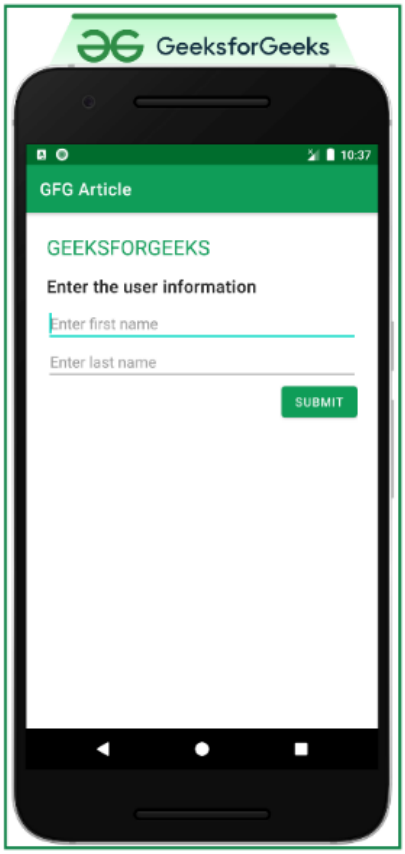
Do *back end* jazyků lze zařadit například JavaScript, SQL, Python, Java, PHP, C# a další méně populární programovací jazyky. Podle dat z průzkumu je nejpoužívanějším *back end* programovacím jazykem Javascript s 65,36 %, druhým SQL s 49,43 % a třetím Python s 48,07 %. Tato statistika je z 28.3.2023, tudíž žebříček se může v současnosti lišit, avšak Javascript si svoje prvenství obhajuje již delší časový úsek. (24)

3.2.2.2 XML

XML v překladu do češtiny znamená rozšiřitelný značkovací jazyk, podobně jako v případě webové stránky programovací jazyk HTML. XML jazyk je velmi dobře čitelný pro vývojáře i pro počítače. Hlavní výhodou je jeho lehkost z hlediska zátěži na jednotlivé stránky aplikace. XML nemá žádný předdefinovaný vzhled nebo kostru a je plně přizpůsobitelná pro uživatele. Hlavním účelem jazyku XML je navrhnout uživateli UI rozhraní, které se zobrazí při každém otevření aplikace. Tyto rozhraní jsou navrhovány hierarchicky a lze je znázornit následujícím diagramem z čehož vznikne následující výstup pro uživatele. (25)



Obrázek 3 Diagram struktury XML (25)



Obrázek 4 Výsledek diagramu v aplikaci (25)

3.2.2.2.1 Terminologie XML

Každý programovací jazyk má svou vlastní syntaxi neboli formu, kterou se píše. Syntaxe značkovacího jazyku XML obsahuje několik důležitých prvků, které budou popsány níže. Popsané prvky jsou pouze základní pro ukázku, samotný programovací jazyk jich obsahuje mnohem více.

3.2.2.2.1.1 Elementy

Elementy jsou stavebním kamenem každého XML dokumentu, jelikož definují jeho kostru, podle které navrhuje stránek aplikace. Elementy začínají pomocí levé lomené závorky (<) po které následuje název elementu který je v základním XML dokumentu ihned uzavřen pravou lomenou závorkou (>). Poté následuje text, který má element zobrazit a následně se element uzavře levou lomenou závorkou (<) lomítkem (/) názvem elementu a pravou lomenou závorkou (>) viz níže přiložený obrázek vlevo. (26)

```
<studentsList>
    <student id="1">
        <firstName>Greg</firstName>
        <lastName>Dean</lastName>
        <certificate>True</certificate>
        <scores>
            <module1>70</module1>
            <module12>80</module12>
            <module3>90</module3>
        </scores>
    </student>
```

Obrázek 5 Ukázka XML elementů (26)

3.2.2.2.1.2 Atributy

Atributy jsou rozšiřujícím prvkem pro elementy, kterým přidělují identifikátory, pozicování, odsazení a další vlastnosti. V níže přiloženém obrázku lze vidět, element *<TextView>* má několik atributů, které jej dále rozšiřují. Element má poziční atributy a textové atributy, jako je barva, velikost, styl a zarovnání textu.

```
<TextView
    android:id="@+id/register_redirect_text2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Registrovat se"
    android:layout_marginStart="5dp"
    android:layout_weight="1"
    android:textAlignment="center"
    android:textColor="#012760"
    android:textSize="27dp"
    android:background="@drawable/tiket_background"
    android:textStyle="italic" />
```

Obrázek 6 Ukázka XML atributů

3.2.2.2.1.3 Textová část

Textová část XML elementu obsahuje samotná data, která jsou uložena uvnitř určitého elementu. Textová část se nachází mezi vytvořením elementu a jeho ukončením a může obsahovat libovolný text, čísla nebo symboly. V XML dokumentu je příklad lépe pochopitelný, avšak i ve výše přiloženém obrázku z Android Studia lze vidět text, který bude zobrazen uživateli a je určen atributem *android:text="Registrovat se"*, tudíž uživateli bude zobrazeno textové pole s textem Registrovat se. V níže přiloženém obrázku z XML dokumentu bude zobrazovaným textem Pokusný nadpis, První odstavec, Druhý odstavec a Třetí odstavec. (26)

```
<článek>
  <nadpis>Pokusný nadpis</nadpis>
  <odstavec>První odstavec</odstavec>
  <odstavec>Druhý odstavec</odstavec>
  <odstavec>Třetí odstavec</odstavec>
</článek>
```

Obrázek 7 Ukázka XML textové části (26)

3.2.2.2.1.4 Komentáře

Komentáře se používají především u větších projektů i přes nepříliš velikou oblíbenost vývojářů. Komentáři lze označit jakoukoliv část nebo řádek kódu. Komentář nijak neovlivňuje funkčnost kódu a nebude na něj brán zřetel počítačem.

Prvním účelem komentářů je popsání funkcionality kódu a jeho podrobnější vysvětlení. Pokud je projekt rozsáhlejší, vývojář může zapomenout, jaká je funkcionalita určité části kódu, proto je lepší využít komentáře a popsat, co kód dělá a proč tomu tak je.

Druhým účelem komentářů je vložení řádku nebo části kódu do komentáře, čímž se danému úseku odebere funkcionalita a bude se chovat, jako by byl smazaný. Touto formou se často provádí testování v případě, že je někde v kódu chyba a vývojář nezjistí kde přesně. Pak přistoupí k metodě postupnému vkládání úseků do komentáře, čímž si nesmaže kód a nebude jej muset formulovat znovu, ale zároveň kód nebude proveden a nemůže způsobit chybovost. (25)

Komentáře mají v každém programovacím jazyku svoji vlastní formu zápisu. V XML se komentáře vkládají do uzavřeného úseku, který začíná „<!--“ a končí „-->“. V níže přiloženém obrázku jsou v komentáři vložené celé dva elementy. Komentář lze také poznat podle barvy kódu, kdy ve většině vývojářských platformách jsou označovány šedým písmem. (25)

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/app_background">

   <!-- <ImageView
        android:layout_width="150dp"
        android:layout_height="150dp"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="0dp"
        android:contentDescription="@string/logo"
        android:src="@drawable/app_logo" />-->

    <!-- <androidx.appcompat.widget.Toolbar
         android:layout_width="match_parent"
         android:layout_height="160dp"
         android:elevation="5dp"
         android:background="#3C403E"
         android:id="@+id/toolbar"
         android:foregroundGravity="fill|center_vertical"
         android:foreground="@drawable/app_logo"/>-->
    <include
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        layout="@layout/toolbar_menu"
        android:id="@+id/toolbar"/>
```

Obrázek 8 Ukázka XML komentáře

3.2.2.2.1.5 Procesní instrukce a deklarace

Procesní instrukce poskytují informace pro počítač, nikoliv pro uživatele. Pomocí těchto instrukcí lze do XML kódu vložit různé soubory nebo příkazy značkovacího jazyka. Tyto instrukce se zapisují do špičatých závorek a otazníků, tudíž například *<?xml version="1.0" encoding="utf-8"?>*. Tímto počítači vývojář sděluje, že používá XML verzi 1.0. (26)

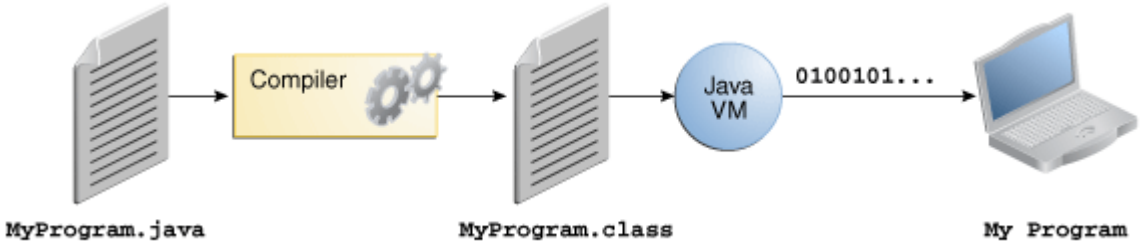
Deklarace ve většině případů specifikuje verzi a kódování, které bude v kódu použito. Procesní instrukce a jejich deklarace se vždy píše na začátek kódu, jelikož podrobněji popisuje XML dokument. Ve výše uvedeném příkladu je deklarací vše, co je zapsáno ve špičatých závorkách s otazníkem, tedy verze XML spolu s kódováním utf-8, které označuje použití českého jazyka. (26)

3.2.2.3 Java

Java je jedním z nejpoužívanějších programovacích jazyků určena pro vývoj serverové stránky, mobilních aplikaci a počítačových aplikací včetně podnikových webových aplikací. Java je objektově orientovaný jazyk s vysokou bezpečností. Hlavní předností jazyku Java je širokospektrální použití, což ho činí perfektní volbou pro projekty s vysokým výkonem a spolehlivostí. Java obsahuje obrovské množství knihoven, které mají předdefinované funkce, čímž vývojáři usnadní veliké množství času s vymýšlením nových funkcí od samého začátku. Společně s přehlednou a rozsáhlou dokumentací není porozumění tomuto jazyku příliš těžkým úkolem pro žádného vývojáře. Největší pozor si musí dát vývojáři na to, že Java je takzvaný *case-sensitive* jazyk, což znamená, že Projekt a projekt jsou dva odlišné pojmy. (28)

Java funguje stejně jako ostatní programovací jazyky jako spojovací vrstva mezi jazykem uživatele a jazykem počítače. Pro správné využívání Java je správné vědět, jakým způsobem funguje a komunikuje. Java pro komunikaci s *front end* částí komunikuje pomocí Java API, což je softwarový komponent Java, který je předdefinovaný a slouží pro čtení programů takzvaně *plug and play*. Jinými slovy, uživatel využije určitou část předdefinovaného kódu například pro získání aktuální datumu a času nebo vykonání matematických operací. *Java Virtual Machine* je další abstraktní vrstva, která se nachází mezi platformou a strojem uživatele, sloužící pro spuštění zdrojového kódu. (28)

Všechen zdrojový kód se ukládá do textového souboru s příponou „.java“. Následně se kód zkompiluje do třídy, který se pomocí bytového kódu virtuálního stroje přeformátuje do podoby programu, jak je vidět níže na obrázku.



Obrázek 9 Transformace Java programu (28)

Pomocí Java lze vytvářet různé typy aplikací, ať už webové, podnikové nebo mobilní. Webové aplikace jsou spuštěny na serveru, odkud se přenáší do dynamické webové aplikace. Nejznámější technologie, často označované jako *frameworky* umožňující tento proces jsou

Spring, Servlet, JSP, Struts, Hibernate a mnoho dalších. Framework lze definovat jako strukturu, která obsahuje řešení různých problémů, které po jeho implementaci zmizí a umožňuje rozšířit funkce, která budou v aplikaci využívány. (28)

3.2.2.3.1 Terminologie Java

Jak již bylo výše zmíněno, Java je objektově orientovaný jazyk, tedy pracuje s objekty, která vyvolává pomocí metod.

3.2.2.3.1.1 Třídy

Třída je logická entita, která je tvořena kolekcí objektů. Lze také konstatovat, že třída je plán všech objektů, který obsahuje vlastnosti a metody. Ve vývoji mobilních aplikací jsou třídy jednotlivé layouty, které si zobrazuje uživatel. Na přiloženém obrázku níže lze vidět, že je zde několik tříd fragmentů a hlavní aktivita. Třídou tedy nejsou jen fragmenty, ale také aktivity, které obsahují jednotlivé objekty. (29)

Obrázek 10 Ukázka Java tříd

3.2.2.3.1.2 Objekty

Jak již vyplývá z definice výše, objekty jsou instancí třídy, jejichž chování definuje entita. Entitou může být například kočka, pes nebo morče, kteří jsou objektem třídy zvířata. Objekt má tři základní charakteristické znaky, jimiž jsou stav, chování a identita. Stav je reprezentován datovou hodnotou objektu. Chování lze nazvat funkcionalitou objektu, například chování při výběru peněz. Posledním znakem je identita, lze označit také jako identifikátor, čímž lze jasně definovat daný objekt, typicky znám jako *ID*. Objekty mohou být vyjádřeny číselně hodnotami (*Number, Float, Long, Integer*) nebo slovně (*String, Boolean).* (29)

3.2.2.3.1.3 Metody

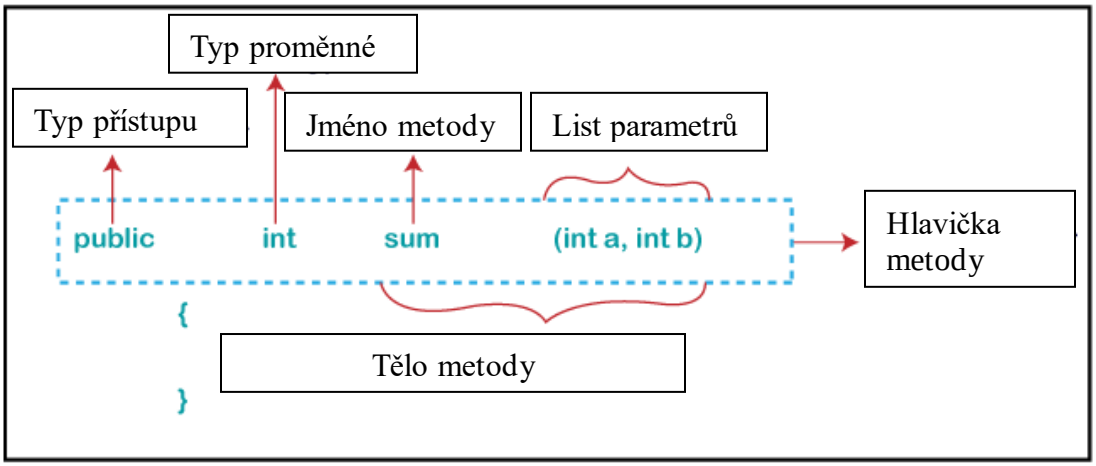
Metoda je část kódu, který je izolován od zbytku a nebude proveden, pokud nebude metoda vyvolána. Hlavním účelem metody je použití její části kódu na více místech najednou bez nutnosti vytvářet stejný kód znovu. Výhodou je přehlednost kódu, jednoduchá úprava a snadná implementace. Jak již bylo zmíněno, metody se neprovedou, pokud nebudou řádně zavolány anebo vyvolány. (30)

Na obrázku níže lze vidět detailněji rozepsanou a popsanou metodu. První deklarací je *public*, dále lze může být *private, protected* nebo *default*. Public metoda je dostupná pro všechny třídy v aplikaci, avšak jelikož je metoda v podstatě veřejná, nelze ji považovat za bezpečnou volbu. Metoda *private* je dostupná pouze třídě, v niž je definovaná. Stále ji lze použít na více místech, avšak pouze v rámci třídy a tato metoda je bezpečnější. *Protected* označuje metodu, která je dostupná pouze v izolovaném bloku nazývaný balíček nebo *package*, který obsahuje více metod. Poslední deklarací je *default* metoda, která je dostupná pouze ve stejném balíčku jako je metoda umístěna. (30)

Metoda musí vždy vrátit zpátky určitou hodnotu a podle toho se také metoda deklaruje. Deklarace „int“ symbolizuje, že vrácená hodnota bude číslo. Dalšími deklaracemi jsou například *String, boolean* nebo *void*, pokud metoda nevrací hodnotu žádnou. (30)

Jméno metody je libovolné a závisí na vývojáři, avšak jméno musí být unikátní a mělo by jasně charakterizovat metodu. Nepsaným pravidlem je, že pokud by měla být metoda složena z více slov, první písmeno každého dalšího slova je označeno velkým písmenem, například takto: „testovaciMetodaPriklad“. (30)

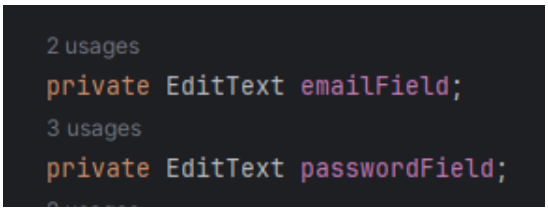
Poslední deklarací jsou parametry, které bude metoda používat. Jednotlivé parametry jsou definovány v kulatých závorkách a odděleny čárkou. Definují se datovým typem a názvem proměnné.



Obrázek 11 Ukázka Java syntaxe metody

3.2.2.3.1.4 Proměnné

Proměnné jsou používány ve všech programovacích jazycích a v Javě tomu není jinak. Do proměnných se převážně ukládá hodnota, s kterou se bude později manipulovat pro jednodušší a přehlednější zápis. Při používání proměnných je nutná jejich prvotní deklarace na začátku segmentu, v kterém působí, jak lze vidět na obrázku níže. Touto deklarací v aplikaci vývojář sděluje počítači, že bude používat privátní proměnnou, jejíž uživatelské rozhraní je *EditText* a proměnná je pojmenována *emailField* a *passwordField*. (31)



Obrázek 12 Ukázka Java proměnných a jejich deklarace

3.2.2.3.1.5 Komentáře

Stejně jako v každém programovacím jazyku i v Javě existuje možnost komentářů. Okomentovat lze pouze jeden řádek, pomocí použití dvou lomítek a následného sdělení komentáře. Druhou variantou je komentovat úsek kódu. Úsekové komentáře začínají lomítkem s hvězdičkou „/*“ a jsou ukončené přesně naopak, tedy v pořadí hvězdička a lomítko „*/. Mezi začátkem a koncem komentáře se nachází okomentovaný kód, který bude počítačem vynechán, jako by zde nebyl. (29)

i. Single line Comment

```
// System.out.println("This is an comment.");
```

ii. Multi-line Comment

```
/*
    System.out.println("This is the first line comment.");
    System.out.println("This is the second line comment.");
*/
```

Obrázek 13 Ukázka Java komentáře (29)

3.2.2.4 C++

C++ je objektově orientovaný programovací jazyk, podobně jako Java. Lze konstatovat, že se jedná o nadstavbu jazyka C, do kterého přidává objektové koncepty, jako jsou třídy, objekty, dědičnost a další. Pomocí tohoto jazyka lze vyvinout operační systémy, databázové systémy, hry a mnoho dalších typů aplikací. Hlavními pozitivními vlastnostmi jsou výkonnost, jelikož při vývoji aplikací je vyžadován vysoký výkon, dále flexibilita, jelikož C++ obsahuje kromě objektově orientovaného programování také procedurální či funkční programování a také rozšiřitelnost z hlediska počtu možných implementovaných knihoven. Stejně jako u Java je tento programovací jazyk tzv. *case-sensitive*, což znamená, že „Test" a „test" jsou dva rozdílné pojmy. (32)

V některých případech je C++ zastíněno Javou, jelikož vyžaduje větší část manuálního programování z čehož vyplývá, že je časově náročnější než programování v Javě nebo Pythonu například. C++ také nedisponuje vestavěnou podporou pro funkcionality jako přímý databázový přístup, tudíž je potřeba implementovat externí knihovny pro tento účel. Při vykreslování samotné aplikace z důvodu vyšší zátěže jazyka na paměť může být vykreslování pomalejší než například v Javě. (33)

3.2.3 Vývojové prostředí

Vývojové prostředí, často označované jako IDE neboli *Integrated Development Enviroment* je nástroj pro programátory, sloužící k vývoji softwaru. Tyto prostředí však nejsou vždy volně dostupné pro všechny uživatele, jelikož některé firmy mají pouze interní prostředí, které nechtějí expandovat volně komukoliv. Samozřejmě existují i takové prostředí, které jsou volně dostupné na internetu ke stažení kýmkoliv nebo s podmínkou registrace. Některé nejpoužívatelnější z těchto volně dostupných prostředí jsou zmíněny v této kapitole. (34)

3.2.3.1 Android studio

Toto vývojové prostředí je oficiálním IDE pro vývoj aplikací v Androidu. Prostředí má implementovaný editor kódu a vývojářské nástroje od IntelliJ IDEA, které programování výrazným způsobem usnadňuje. Další předností je emulátor, který dokáže nasimulovat prostředí Androidu a zároveň podporuje konektivitu s reálným Android zařízením, ale ten bude podrobněji rozebrán v kapitole níže.

Android studio je nejpoužívanějším nástrojem při tvorbě mobilních aplikací, které mají operační systém Android a poskytuje mnoho vestavěných funkcí optimalizovaných pro mobilní zařízení, narozdíl od vývojových prostředí Visual Studia nebo NetBeans.

Nejnovější verzí Android studia je verze s označením „Hedgehog 1.1“ z roku 2023 a „Gradle plugin 8.2.2“. Další verzí by měla být Iguana | 2023.2.1, která je v současnosti v beta verzi. Tato označení verze znamená, že verze sice prošla prvotní opravou ohlášených chyb, ale nemusí být stabilní a některé operace mohou skončit chybovou hláškou, která není nijak spojena s chybou uživatele, ale dodavatele. (35)

Avšak verze Iguana představuje progresivní vykreslování v grafickém náhledu a úmyslné snížení vykreslovací kvality, což by mělo ušetřit využití počítačové paměti. S tím souvisí další vylepšení, kterým by měla být možnost zobrazit si náhled více grafickým rozhraní naráz, a to i z jiného fragmentu nebo aktivity. Tudíž by si uživatel mohl zobrazit celý grafický vzhled aplikace, bez nutnosti jejího spuštění. (35)

Vývojové prostředí může být nainstalováno na operační systémy Windows, MacOS, Linux a ChromeOS. (36)

Android studio tedy podporuje všechny nejpoužívanější operační systémy a využívat jej může kdokoliv. Menší problém nastává v hardware požadavcích, kdy Android studio sestavuje z projektu pomocí emulace aplikaci, tak spotřebovává veliké množství výpočetní paměti, což některé starší zařízení nemusí výpočetně zvládnout a sestavování může skončit chybou. Celý proces prvotního spuštění aplikace může trvat i několik minut, jelikož se zde provádí veliké množství operací od sestavování projektu, sestavování pomocí Gradle pluginu, přes kód a grafickou vizualizaci až po spuštění emulace na Android zařízení a následné zobrazení aplikace. (37)

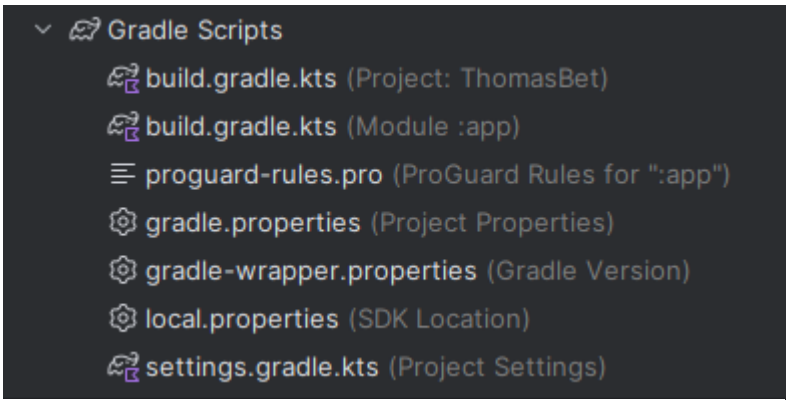
3.2.3.1.1 Gradle

Základním stavebním kamenem při spouštění aplikace je takzvaný Gradle, který je již při tvorbě nového projektu předdefinovaný, avšak uživatel má možnost si tento pokročilý nástroj upravit, podle svých parametrů. Na základě nastavení se pak sestaví všechny segmenty projektu do jednoho výstupu, kterým je aplikace. Při budování procesu se pomocí SDK programovací kód kompiluje, tedy překládá do spustitelné podoby. Tuto operaci má na starost Gradle plugin, který stejně jako Gradle funguje nezávisle na Android Studiu. Z tohoto vyplývá, že nijak neovlivňuje samotný projekt v Android Studiu a zároveň může být vyvolán i v zařízeních bez instalace Android Studia. Což znamená, že při každé

aktualizaci Android Studia, se musí zvlášť ještě aktualizovat Gradle, jelikož spolu prakticky nejeví žádnou součinnost. (38)

3.2.3.1.2 Prostředí

Android studio nabízí velké množství implementovaných funkcí, které jsou primárně skryty, což je přívětivější pro přehlednost a orientaci v prostředí. Prostředí lze jednoduše upravit podle potřeby programátora. Například postranní lišty se dají přemístit či minimalizovat na ikony, což rozšiřuje pracovní plochu.

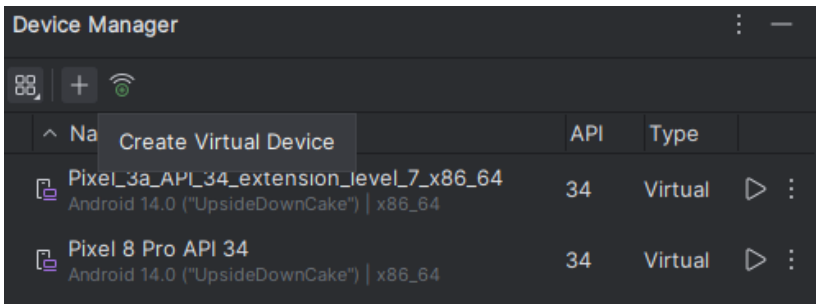


Obrázek 14 Ukázka Gradle souborů v Android Studiu

Při základním rozložení je na levém okraji lišta se spouštěcími prvky projektu. Lze zde naleznout Build, zobrazující výsledky Gradle pluginu, dále Run, který zobrazuje výsledek sestavení projektu spolu s emulací nebo Logcat, který zobrazuje detailnější popis všech operací společně s časovým razítkem a mnoho dalších užitečných záložek.

3.2.3.1.3 Fyzické zařízení a emulace

Projekt vyvíjený v Android studiu je vyvíjený v operačním prostředí Windows, Linux nebo MacOS, které neumí zobrazit aplikaci v operačním prostředí Android. Z tohoto důvodu se využívá emulace, která simuluje určitý operační systém. Android studio má vestavěnou funkci pro emulaci a vývojář si může definovat i verzi Androidu, na které bude aplikaci vyvíjet. (39)



Obrázek 15 Ukázka vytvoření nového emulovaného zařízení

Emulace inicializuje virtuální mobilní zařízení s vybranou verzí operačního systému, na kterém se bude aplikace spouštět. Ačkoliv je tato funkce velice užitečná, její spolehlivost již není tak vysoká, protože je omezena výpočetní paměť telefonů a celý proces prvotního spuštění emulace a načtení aplikace může trvat i několik minut. (39)

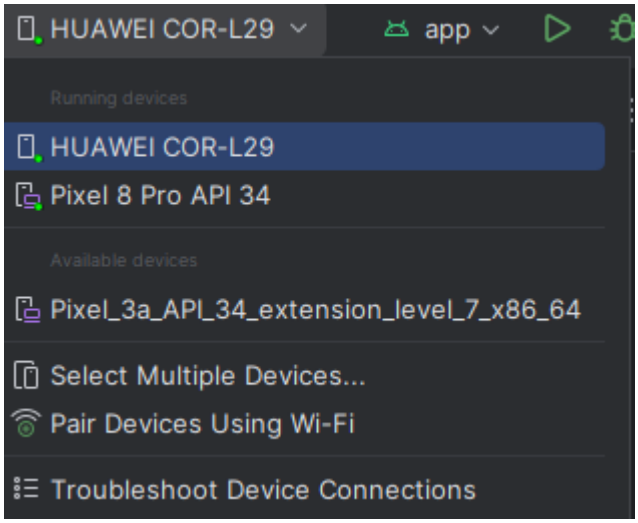
Obrázek 17 Ukázka připojování k emulátoru

Obrázek 16 Ukázka vytvořeného emulovaného mobilního zařízení

První obrázek ukazuje připojování se k emulátoru a vytváření rozhraní mobilního telefonu, který byl předdefinovaný v tomto případě Pixel 8 Pro. Druhý obrázek zobrazuje virtuální prostředí Android na již zapnutém telefonu a emulace je připravena k prvotnímu spuštění aplikace.

Pokud vývojář chce aplikaci testovat na fyzickém zařízení, je zde také tato možnost pomocí připojení fyzického zařízení do počítače pomocí USB nebo pomocí Wifi připojení. Ačkoliv připojení pomocí USB je spolehlivější, tak je nutností telefon spustit v režimu vývojáře čímž mobilní telefon ztrácí záruku. Tento režim totiž umožňuje povolit USB ladění což je nezbytné, aby Android studio načetlo mobilní zařízení. Připojení mobilního zařízení k Android studiu přes Wifi také není jednoduché, jelikož musí být počítač na které je nainstalované Android studio připojeno na stejné síti jako mobilní telefon. Avšak většina počítačů je dnes připojena přes Ethernet kabel, tudíž mobilní telefon se nezobrazí. (39)

Pokud jsou všechny výše uvedené kroky splněny, Android studio najde mobilní zařízení a rovnou bude přidáno k používání. Na výše přiloženém obrázku lze vidět, že fyzické zařízení je označené i jinou ikonou než zařízení emulované. Pokud se zařízení zobrazuje, po spuštění se aplikace načte přímo na fyzickém zařízení a rychlost aplikace je značně vyšší.



Obrázek 18 Ukázka připojeného fyzického zařízení

3.2.3.1.4 Aktivita a Fragment

Android studio rozlišuje dva hlavní segmenty, jimiž jsou Aktivity a Fragmenty, přičemž každý zprostředkovává něco jiného.

Aktivity se používají jako vstupní body do Android aplikace a reprezentují individuální obrazovky uživatelského rozhraní. Aktivity mají svůj vlastní životní cyklus a obsahují předdefinované metody jako „onCreate(), onStart(), onResume()“ a mnoho dalších. Aktivity jsou zodpovědně za navigaci mezi různými obrazovkami a nastavování jejich layoutů a životních cyklů.

Fragmenty jsou modulárním designem, které sestavují uživatelské rozhraní nebo popisují chování určité aktivity. Fragmenty se používají v rámci určité aktivity a jsou sekundárním elementem Aktivit. Životní cyklus Fragmentu záleží na jejich nadřazené aktivitě, která je může přidávat, odebírat nebo nahrazovat jiným Fragmentem. Právě nahrazování je nejčastějším používáním. Pokud uživatel je na domovské stránce a klikne na tlačítko nebo směrovací odkaz, Fragment domovské stránky se nahradí jiným Fragmentem podle definovaného chování tlačítka či odkazu. (40)

Nejčastější doporučení je používání architektury jedné Aktivity a více Fragmentů pro velké množství Android aplikací. Při tomto přístupu je jedna hlavní aktivita, která poskytuje

uživatelské rozhraní pomocí různých Fragmentů, které zobrazují jiné obrazovky a reprezentují odlišné funkcionality. Za tohoto předpokladu je kód více modulární a lehčeji udržovatelný. (41)

3.2.3.2 NetBeans

Vývojové studio NetBeans je další z mnoha IDE, které jsou dostupné pro vývojáře pro tvorbu mobilních aplikací nebo webů. Podpora je zde primárně určena pro programovací jazyk Java, ale podporuje také ostatní programovací jazyky jako HTML5, PHP, C, C++ a mnoho dalších. (42)

Po pročtení dokumentace vývojového prostředí autor usoudil, že výhodnější volbou pro vývoj aplikace bude Android Studio, které nabízí vestavěný emulátor pro simulaci mobilního zařízení s operačním systémem Android a celková příprava IDE není vyžadována. Není potřeba instalovat dodatečně balíčky pro vývoj ani připojovat fyzické zařízení pro testování funkcionalit. Z výše uvedených důvodů tato technologie nevyhovuje autorovi pro vývoj aplikace v tomto prostředí. (42)

3.2.3.3 Visual Studio

Visual studio je vývojové prostředí určené pro vývoj webových, mobilních, cloudových a herních aplikací. Nachází se zde podpora pro mnoho programovacích jazyků, například C#, Visual Basic, C++, Javascript, Python a mnoho dalších. Podporuje také mnoho typů projektu od konzolových aplikací, přes Windows Forms aplikace, přes ASP.NET webové aplikace až po Xamarin mobilní aplikace.

Ačkoliv podporuje i mobilní aplikace, po pročtení dokumentace autor usoudil, že výhodnější volbou pro vývoj mobilní aplikace bude vývojové prostředí Android Studia, které je oficiálním IDE pro vývoj Android aplikací a poskytuje mnoho implementovaných funkcí pro usnadnění tvorby. Dalším velikým rozdílem je podpora programovacích jazyků, kdy Android studio primárně používá programovací jazyky Java a Kotlin pro tvorbu mobilních aplikací, avšak Visual Studio používá Xamarin a C# spolu s .NET, s čímž autor práce nemá potřebné zkušenosti a znalosti.

3.2.4 Mobilní operační systémy

Operační systém je základní software, který řídí a spravuje hardware počítače a umožňuje spouštění aplikací. Operační systém v mobilních zařízeních je optimalizován pro mobilní zařízení omezeným výkonem, velikostí zařízení, dotykovým ovládáním a připojením k mobilnímu internetu z čehož vyplývá, že řídí spíše softwarové aplikace. Systém umožňuje uživatelům spouštět aplikace, komunikovat s hardwarovou částí zařízení jako například prohlížení internetu, psaní zpráv nebo instalování nových aplikací a her.

Oproti počítačovým operačním systémům jsou mobilní mnohem rychleji načítány a zabírají výrazně méně místa v zařízení. Jsou také nákladnější, co se týče výpočetní paměti, kterou vyžadují pro svůj běh. Avšak počítačové systémy jsou mnohem flexibilnější a nabízí větší množství funkcionalit. Mezi nejznámější mobilní operační systémy lze zařadit Android, iOS a Windows 10 Mobile. (43)

3.2.4.1 Android

Android je mobilní operační systém vyvinutý společností Google. Byl založen na Linuxovém jádře a je otevřeným operačním systémem. Android je aktuálně nejrozšířenějším mobilním operačním systémem na světě. Obecně platí, že mobilní telefony s operačním systémem Android jsou levnější oproti konkurenčním telefonům se systémem iOS. Android nabízí flexibilní operační systém s velikými možnostmi pro vývojáře si operační systém přizpůsobit podle sebe. (44)

Aplikace jsou volně dostupné z obchodu Google Play, který obsahuje údajně skoro 2,5 milionu aplikací. Statistika je z Března roku 2024. (45)

Nejčastějšími mobilními zařízeními s operačním systémem Android jsou telefony od HTC, Samsung, Xiaomi, Nokia, SONY a mnoho dalších. Operační systém Android je zde nastaven od výrobce, tudíž není potřeba manuální instalace jako v případě počítačových operačních systémů u některých počítačů.

Android je společností Google neustále vyvíjen a plně podporován, tudíž vychází stále nové verze jeho operačního systému. Nejnovější verzí je Android 14 s názvem „Upside Down Cake“, který byl nasazen v květnu roku 2023. (46)

S novou verzí přichází také nové funkcionality, například při příchozí zprávě lze nastavit problikávání svítilny, stejně jako u zařízeních s iOS, které tuto funkci mají již delší časový úsek. Přidala se také funkce vidět životnost baterie, avšak opět zařízení iOS tuto funkci mají již delší dobu. Oproti starší verzi Androidu se přidalo ještě mnoho dalších

funkcionalit, upravující vzhled zamykací obrazovky, vylepšená bezpečnost systému nebo podpora většího fontu. (47)

3.2.5 Metodologie vývoje aplikací

Při vytváření nové aplikace je nezbytné určit, který vývojový model bude využit pro její vytvoření. Každá metodologie vnímá vývoj jiným pohledem a vývoj je také ovlivněn mnoho různými faktory. I přes několik již známých metodologií si mnoho firem vytvořilo vlastní interní metodologie, pomocí nichž vyvíjí své vlastní aplikace. Metodologii lze také definovat jako způsob rozdělení práce, kdy se určuje postupné řešení vývoje v předem definovaných krocích.

3.2.5.1 Vodopádový model

Vodopádový model je lineární způsob vývoje softwaru, který definuje proces vyvíjení jako posloupné fáze, které jsou předem rozvrženy. Dříve byl více populární, než je tomu dnes, kdy se používá pouze jeho část. Avšak většina ostatních modelů je založena právě na klasickém vodopádovém modelu. Nejčastěji se využívá pro projekty, které jsou rozvrženy na delší časový úsek, avšak je zde minimální prostor na chyby, jelikož další fáze nebude započata, dokud bude současná fáze stále ve vývoji. Avšak toto pravidlo platí i naopak, pokud se započne nová fáze, nelze se vrátit k předchozí. Vodopádový model je skládá z šesti fází. (48)

3.2.5.1.1 Předpoklady a analýza

Hlavním krokem je předem stanovení požadavků, které systém musí zvládnout zpracovat včetně jeho funkcionalit, které budou sepsány, nejčastěji do formy dokumentu. Tyto požadavky vznáší zúčastněné strany a také potencionální uživatelé systému. Požadavky se poté musí analyzovat, zdali jsou splnitelné v definovaném časovém rozsahu. (48)

3.2.5.1.2 Design

Po splnění předchozího kroku a definování všech požadavků na funkcionality, začíná další fáze projektu, kterou je design. V této fázi se detailně připraví veškeré uživatelské rozhraní, systémové komponenty a funkcionality, které byly vyžadovány. Po navrhnutí celkové detailní struktury se začne v další fázi vyvíjet zdrojový kód. (48)

3.2.5.1.3 Vývoj

Na základě designu definovaného v předchozí fázi se začíná s vývojem zdrojového kódu, který zajistí funkcionalitu elementů v popředí. Po naprogramování všech uživatelských rozhraní se spouští drobné testy, které odhalují nedostatky v kódu, který se nachází na pozadí. Pokud se všechny modely chovají, jak bylo předem určeno, přechází se do další fáze vývoje. (48)

3.2.5.1.4 Testování

V této fázi se provádí hloubkové testování veškeré funkcionality. Veškeré prvky na uživatelském rozhraní musí odpovídat předpokladům z první fáze a funkčnost kódu rovněž tak. Po pečlivém vyzkoušení všech tlačítek a funkcionalit se přechází na implementaci. (48)

3.2.5.1.5 Implementace a nasazení

Testovací modul se smaže a aplikace se implementuje na výsledný modul, kde bude fungovat i po spuštění uživateli. Každý modul se postupně implementuje a průběžně testuje, zdali všechny komponenty stále fungují i při změně modulů. Po celkové úspěšné implementaci se provádí poslední testování před finálním spuštěním. Testování se provádí při dvou krocích, jimiž jsou alfa testování a beta testování. Při alfa testování je systém podroben různým testům od vývojářského týmu. Při testování beta je systém zpravidla rozšířen mezi přátelskou skupinu lidí, kteří poskytnou zpětnou vazbu ohledně aplikace a mohou se provést finální úpravy před spuštěním mezi všechny uživatele. (48)

3.2.5.1.6 Údržba

Údržba již vyvinutého systému je jedním z nejdůležitějších a nejdéle trvajících fází. Po úspěšném spuštění mezi všechny uživatele přichází fáze, kdy je potřeba sledovat veškeré aktivity uživatelů, zdali se zobrazují správně a nevzniká žádné nežádoucí chování. Chyby mohly být přehlédnuty v testování a odhalit je mohl až nečekaný uživatelský vstup. Proto je potřeba monitorovat aplikaci a snažit se chyby v co nejmenším časovém úseku odstranit bez ovlivnění uživatelů, kteří se nepotýkají s problémy. Některé chyby ovšem mohou vyžadovat například restartování serverů, což ovlivní veškeré uživatele právě využívající aplikaci. (48)

3.2.5.2 Agilní model

Agilní model přistupuje k vývoji pomocí takzvaných iterací, tedy opakování, dokud projekt nebude perfektně připravený před jeho spuštěním. Je zde také mnohem větší prostor na chyby, jelikož je projekt rozdělen do několika opakovacích cyklů, které nezahrnují dlouhodobé plánování projektu. Projekt je tedy vyvíjen postupně a fáze se neustále opakují, dokud nejsou splněny veškeré předpoklady projektu. Každé opakování zahrnuje krátký časový úsek, přičemž celý projekt je vyvinut obvykle do měsíce od jeho započnutí. Jelikož je projekt vyvíjen v těchto iteracích, množství rizik je minimální a časový úsek vývoje je také minimalizován. Každá iterace obsahuje fázi shromáždění požadavků, design požadavků, vývoj požadavků, často označovaný jako iterace, testování, nasazení a zpětná vazba. Každá fáze se podobá fázi vodopádového modelu, avšak největší rozdíl mezi nimi je hlavně v obecném pohledu na projekt a v počtu provedených fází. Hlavní nevýhodou může být nejasnost nebo chybná interpretace zadání což může vést k chybě v jednom z opakováních. Další nevýhodou je údržba projektu, jelikož byl vyvinut rychleji, neobsahuje tolik propracovanou a detailní dokumentaci, podle které by ho mohl vývojář udržovat. Z tohoto důvodu je agilní model nevýhodný pro větší projekty, které vyžadují stálou podporu vývojářů. (49)

Obrázek 19 Ukázka jedné iterace v agilním modelu (49)

3.3 Vybrané technologie pro tvorbu aplikace

Z výše zmiňovaných způsobů a technologií tvorby aplikace byl využit vodopádový vývoj aplikace, jelikož rozsah aplikace se předpokládá veliký, tak je důležité mít přesný plán práce. V této fázi byl také navrhnut vzhled mobilní aplikace, který byl inspirován vzhledem vytvořeným v bakalářské práci autora. Autorem byly zvoleny jazyky dva, konkrétně značkovací jazyk XML pro návrh uživatelského rozhraní a programovací jazyk Java pro splnění všech vyžadovaných funkcionalit aplikace. Autorem bylo zvoleno vývojové prostředí Android studio, které je oficiálním IDE doporučeným pro vývoj Android aplikací a obsahuje tak mnoho vestavěných funkcionalit, kterými ostatní IDE nedisponují.

Níže budou uvedeny vybrané technologie a knihovny, které mobilní aplikace využívá. Z důvodu rozsáhlosti budou uvedeny pouze nejhlavnější.

3.3.1 Microsoft Azure databáze

Microsoft Azure je platforma společnosti Microsoft, kterou je nabízena řada technologií, které poskytují evidenci dat, monitorování dat nebo také vytvoření dočasného zkušebního serveru na kterém lze monitorovat vytížení aplikací a spotřebované úložiště. Kromě základních statistik je nabízeno přizpůsobení SQL serveru včetně *firewall* pravidel. Je také nabízena zkušební verze SQL databáze, která umožňuje vytvoření a upravení databázového schématu.

3.3.2 Spring framework

Spring *framework* je jedním z nejpoužívanějších, jelikož podporuje implementaci dalších frameworků, které následně umožňuje využívat. Jeho největší výhodou je například šablona JPA, Hibernate nebo JDBC používána pro připojení a komunikaci s databází. Například pokud má vývojář databázi od společnosti Microsoft takzvanou Azure DB, je mu JDBC připojení poskytováno provozovatelem a stačí jej správně implementovat do zdrojového kódu v Javě. Tudíž není potřeba definovat připojení, argumenty, provádět transakci a podobně, ale stačí využít předdefinovanou šablonu. (50)

3.3.3 Odds API

Odds API je veřejně dostupná předdefinovaná služba, která pomocí API poskytuje JSON zobrazení následujících sportovních událostí, živých výsledků a také zobrazení výsledků sportovních událostí, které nejsou starší než 3 dny. Tyto údaje budou využity pro možnost vsadit na skutečné zápasy. (51)

Výše zmíněné API je jedinou bezplatnou verzí, kterou autor práce nalezl s omezením vznešení maximálního měsíčního počtu dotazů v hodnotě 500. Api poskytuje odpověď na požadavky ve formátu JSON.

3.4 Analýza podobných sázkařských aplikací

Nejpopulárnější české sázkařské aplikace jsou poskytovány sázkařskými společnostmi Tipsport a Fortuna, tudíž analýza se bude věnovat těmto dvěma aplikacím. (52)

3.4.1 Tipsport

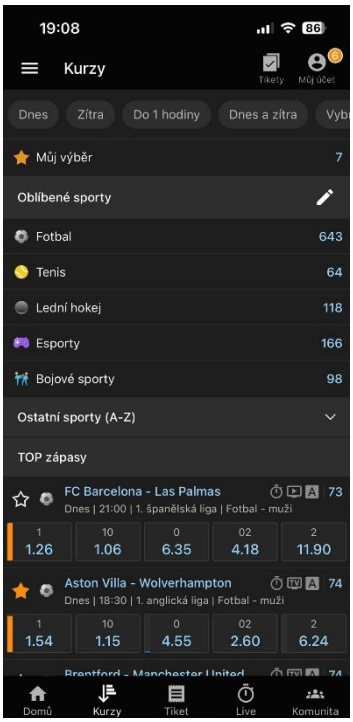
Aplikace Tipsport poskytuje mnoho uživatelsky přívětivých funkcionalit, například *darkmode*, který umožňuje uživateli vybrat si místo klasického designu s bílým pozadím design tmavší s šedivý pozadím, jelikož bílý design je pro lidské oko agresivní a příliš zářivý.

3.4.1.1 Kurzová nabídka

Kurzová nabídka obsahuje základní filtraci podle datumu, kdy se budou sportovní příležitosti hrát. Sportovní nabídka je velice rozsáhlá a obsahuje také unikátní kategorie, které nemají se sportem nic společného, například ve volebním období je zde možnost volit prezidenta nebo vítěznou politickou stranu.

3.4.1.2 Blog

Tipsport nabízí přihlášeným uživatelům možnost napsání krátké analýzy na zápas, čímž by mohl inspirovat ostatní sázkaře. Sázkařům je také zprostředkováváno fórum, v kterém lze komunikovat s ostatními sázkaři v libovolném rozpětí.



Obrázek 20 Kurzová nabídka aplikace Tipsport

3.4.2 Fortuna

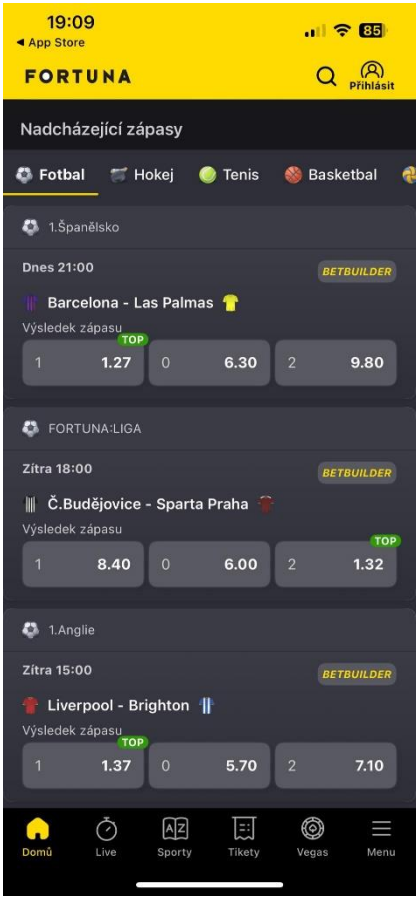
Aplikace Fortuna zvolila tmavší pozadí již od poskytovatele, avšak sázkař nemá možnost zvolit si pozadí světlé a funkce *darkmode* není sázkaři k dispozici.

3.4.2.1 Kurzová nabídka

Kurzová nabídka oproti aplikaci Tipsportu obsahuje více strohou filtraci datumu konání sportovních příležitostí. Sportovní nabídka je velice rozsáhlá, avšak neobsahuje unikátní kategorie, které nemají se sportem nic společného jako tomu je v aplikaci Tipsportu. Kurzová nabídka neobsahuje ani možnost filtrování zápasů podle datumu, kdy se budou odehrávat, což aplikace Tipsportu obsahuje.

3.4.2.2 Blog

Aplikace Fortuna neobsahuje možnost blogu, tudíž analýzy nemají možnost vytvářet analýzy na zápasy, mohou se k nim vyjadřovat pouze pomocí diskusního fóra přímo u zápasu.



Obrázek 21 Kursová nabídka aplikace Fortuna

3.4.3 Závěr analýzy

Porovnávanými kritérii byla kurzová nabídka, *darkmode* a blog. Aplikace Tipsportu nabízí přepínač mezi *darkmode* a *lightmode*, čímž dává sázkaři volbu designu aplikace. U aplikace Fortuna tato možnost chybí. Aplikace Fortuna nemá možnost filtrování zápasů podle datumu a ani neobsahuje speciální příležitosti, tudíž v této kategorii také prohrává. Poslední porovnávanou vlastností byl blog, který aplikace Fortuny neobsahuje. Aplikace Tipsport má k dispozici blog pro sázkaře, který umožňuje vytvořit sázkařům analýzy k zápasům.

Na základě porovnávaných kritérií je Tipsport rozvinutější aplikací s větším množstvím funkcionalit oproti aplikaci Fortuna.

Kritéria	Fortuna	Tipsport
Darkmode/lightmode	Ne	Ano
Kurzová nabídka	Ano, bez filtrování zápasů	Ano s filtrováním zápasů
Speciální příležitosti kurzové nabídky	Ne	Ano
Blog	Ne	Ano

4 Vlastní práce

Hlavním cílem praktické části je vytvoření mobilní aplikace, která bude sázkařům sloužit pro uzavření sázky na zápas a zároveň bude obsahovat blog na který budou moci přidávat profesionální sázkaři své strategie. Při tvorbě byl kladen veliký důraz na jednoduchost, intuitivnost a přívětivost aplikace pro všechny věkové kategorie potencionálních uživatelů. V této části práce jsou definovány funkční a nefunkční požadavky na aplikaci, dále návrh uživatelského rozhraní a struktura databáze. Poté následuje provedení aplikace, kde jsou popsány technologie a služby využité k realizaci aplikace a nejdůležitější funkcionality aplikace. Následuje testování aplikace a požadavky pro spuštění aplikace.

4.1 Požadavky na aplikaci

4.1.1 Funkční požadavky

- Obecné
    - Aplikace musí zobrazovat, že se jedná o diplomovou práci v sekci O nás spolu s krátkým představením
- Uživatelská správa
    - Aplikace musí umožnit uživatelům založení účtu a přihlášení se do něj pomocí uživatelské jména nebo emailu a hesla
    - Aplikace musí rozlišovat profesionální sázkaře a rekreační sázkaře
    - Aplikace musí umožňovat manuální přidělování rolí pomocí administrátorského panelu
- Blogová část
    - Aplikace musí umožnit rekreační sázkařům zobrazení strategií a článků
    - Aplikace musí povolovat psaní strategií a článků pouze přihlášeným uživatelům s rolí profesionální sázkař
    - Aplikace musí umožňovat editaci strategií i článků společně s funkcí mazání
- Výsledky zápasů
    - Aplikace musí obsahovat informace o sportovních zápasech, které se budou konat pro možnost uzavření sázky
    - Aplikace musí obsahovat informace o sportovních zápasech, které se již odehrály a zobrazovat jejich výsledky

- Sázecí část
    - Aplikace musí umožnit uživatelům vsadit na zápasy
    - Aplikace musí umožnit uživatelům zobrazit výsledky odehraných zápasů
    - Aplikace musí umožnit uživatelům vidět vlastní vsazené tikety

4.1.2 Nefunkční požadavky

- Bezpečnost
    - Bezpečnost uživatelských dat (sázkařská historie, přihlašovací údaje)
    - Komunikace s databází přes http/https požadavky
    - Šifrování uživatelských hesel v databázi
- Responzivita
    - Uživatelské rozhraní musí být co nejvíce responzivní pro širokou škálu mobilních telefonů a jejich rozlišení
    - Všechny texty musí být dostatečně čitelné i při menším displeji
- Ošetření chybových hlášení
    - Aplikace musí mít ošetřené neočekávané události
    - Aplikace musí mít ošetřené uživatelské vstupy vůči nesmyslnému zadávání
- Plynulost a rychlá odezva aplikace

4.2 Návrh uživatelského rozhraní

Vzhled mobilní aplikace bude vycházet z návrhů mobilní aplikace v bakalářské práci autora aplikace. Níže budou uvedeny pouze vybrané uživatelské rozhraní. Aplikace bude zahrnovat rozhraních mnohem více.

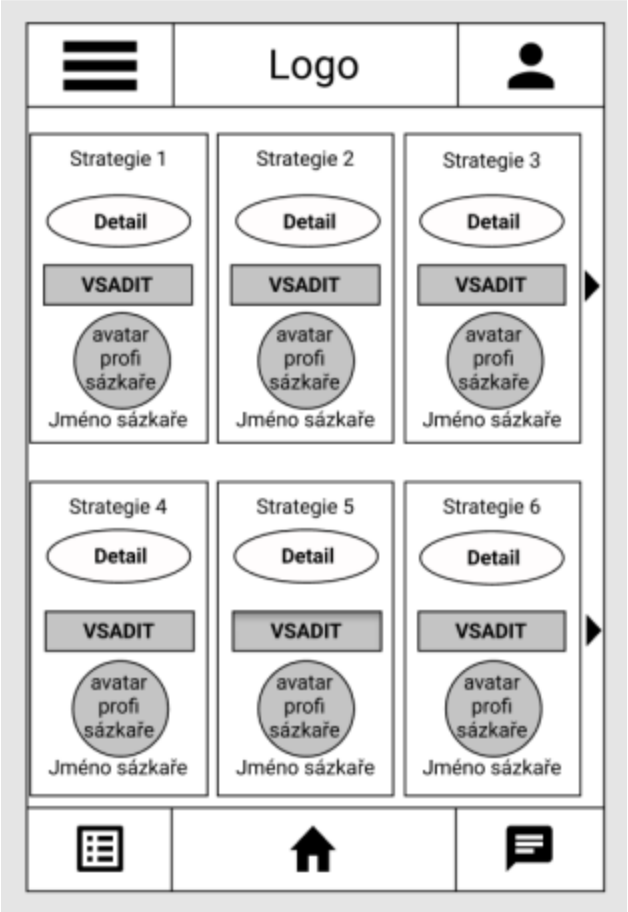
4.2.1 Návrh domovské obrazovky

Hlavním účelem této sekce je seznámit uživatele s vývojářem aplikace spolu se základními informacemi týkající se aplikace. Sekce zahrnuje také tlačítka pro registraci a přihlášení, jelikož bez nich nebude umožněno plné využívání aplikace.

Obrázek 22 Návrh uživatelského rozhraní domovské obrazovky

4.2.2 Návrh sekce strategie

Hlavním účelem této sekce je seznámit uživatele s dostupnými strategiemi, které jsou napsány profesionálními sázkaři. Rozhraní obsahuje veliké množství prvků a je pravděpodobné, že některé nebudou v provedení aplikace zahrnuty. Uživatel by musel obrazovku posouvat a rozhraní by vypadlo nedokončeně a příliš složitě. Některé strategie, které již poskytli profesionální sázkaři jsou obecnější a poskytují rady sázek na celou hokejovou ligu za určitých naplněných podmínek.



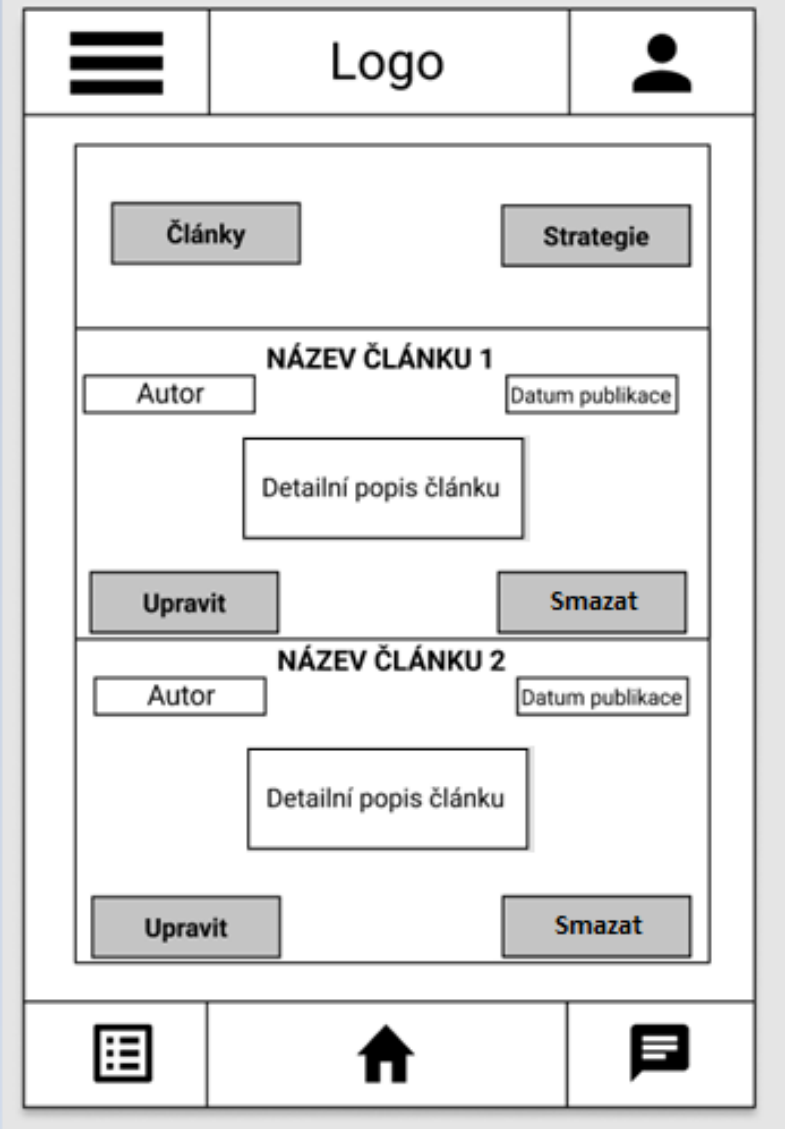
Obrázek 23 Návrh uživatelského rozhraní sekce strategie

4.2.3 Návrh sekce články

Hlavním účelem této sekce je seznámit uživatele se sekcí článků, kde lze naleznout pouze zajímavé články, nikoliv inspirativní. Sázkař se tak může dozvědět zajímavosti nebo pouze pročítat články jako zdroje odborných informací.

Profesionální sázkař bude mít k dispozici tlačítka upravit a smazat svůj vlastní článek. Administrátor bude mít možnost smazat a upravit všechny články pro případ, že by porušovaly pravidla.

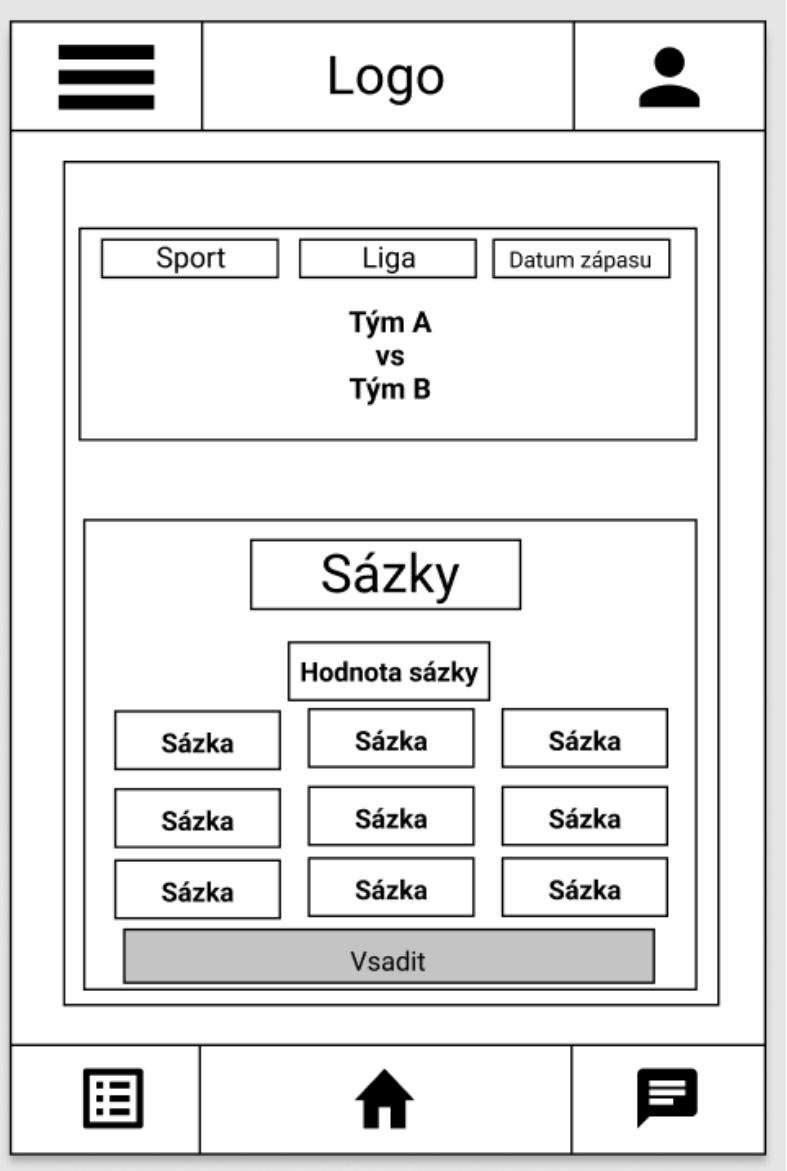
Návrh sekce články bude použit i u podobné kategorie strategií, kde budou pouze některá pole navíc.



Obrázek 24 Návrh uživatelského rozhraní sekce články

4.2.4 Návrh sázky na zápas

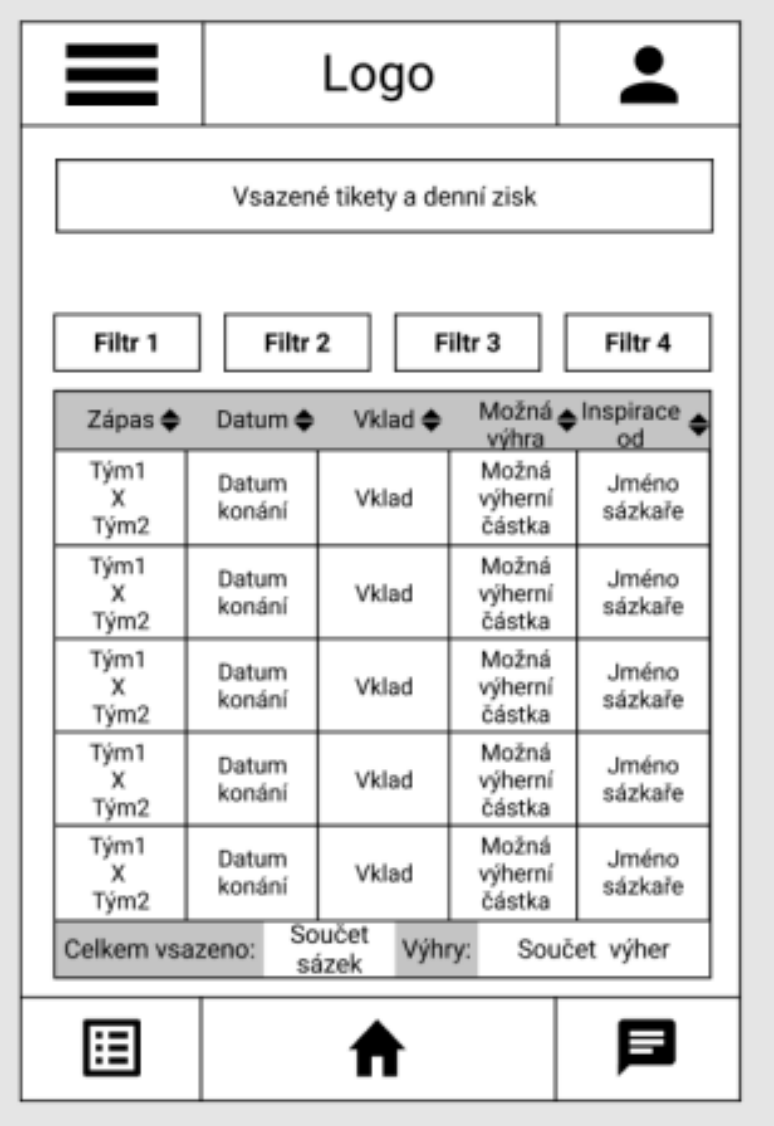
Návrh uživatelského rozhraní sázky na zápas obsahuje informace o zápase, který si uživatel vybral a vypsané sázky na něj. Nachází se zde také textové pole, do kterého sázkař napíše částku, kterou chce na zápas vsadit. Po kliknutí na tlačítko vsadit nebude možné akci vzít zpět a sázka bude uzavřena.



Obrázek 25 Návrh uživatelského rozhraní sázky na zápas

4.2.5 Návrh sekce vsazených tiketů

Návrh sekce vsazených tiketů musí zobrazit uživateli sázky, které byly uzavřeny a nebyly vyhodnoceny. Sekce musí zobrazovat informace o zápase, částku, kterou uživatel vsadil a nejlépe i částku, kterou může vyhrát. Sekce musí být dostatečně přehledná a zobrazit jen takové množství sázek, aby se v nich sázkař orientoval.



Obrázek 26 Návrh uživatelského rozhraní vsazených tiketů

4.2.6 Use Case diagram

Součástí návrhu je také Use Case diagram, který zobrazuje aktéry aplikace a jejich možnosti využívání funkcionalit aplikace.

Do diagramu byly přidáni 4 aktéři, reprezentující tři uživatelské role a systém. Role v aplikaci jsou administrátor, sázkař a profesionální sázkař. Administrátor reprezentuje osobu, která má možnost změnit role a upravit nebo smazat články. Sázkař je osoba, která aplikace využívá za účelem běžného používání, tedy zobrazování strategií a uzavírání sázek. Profesionální sázkař je osoba, která inspiruje ostatní svými napsanými strategiemi a články.

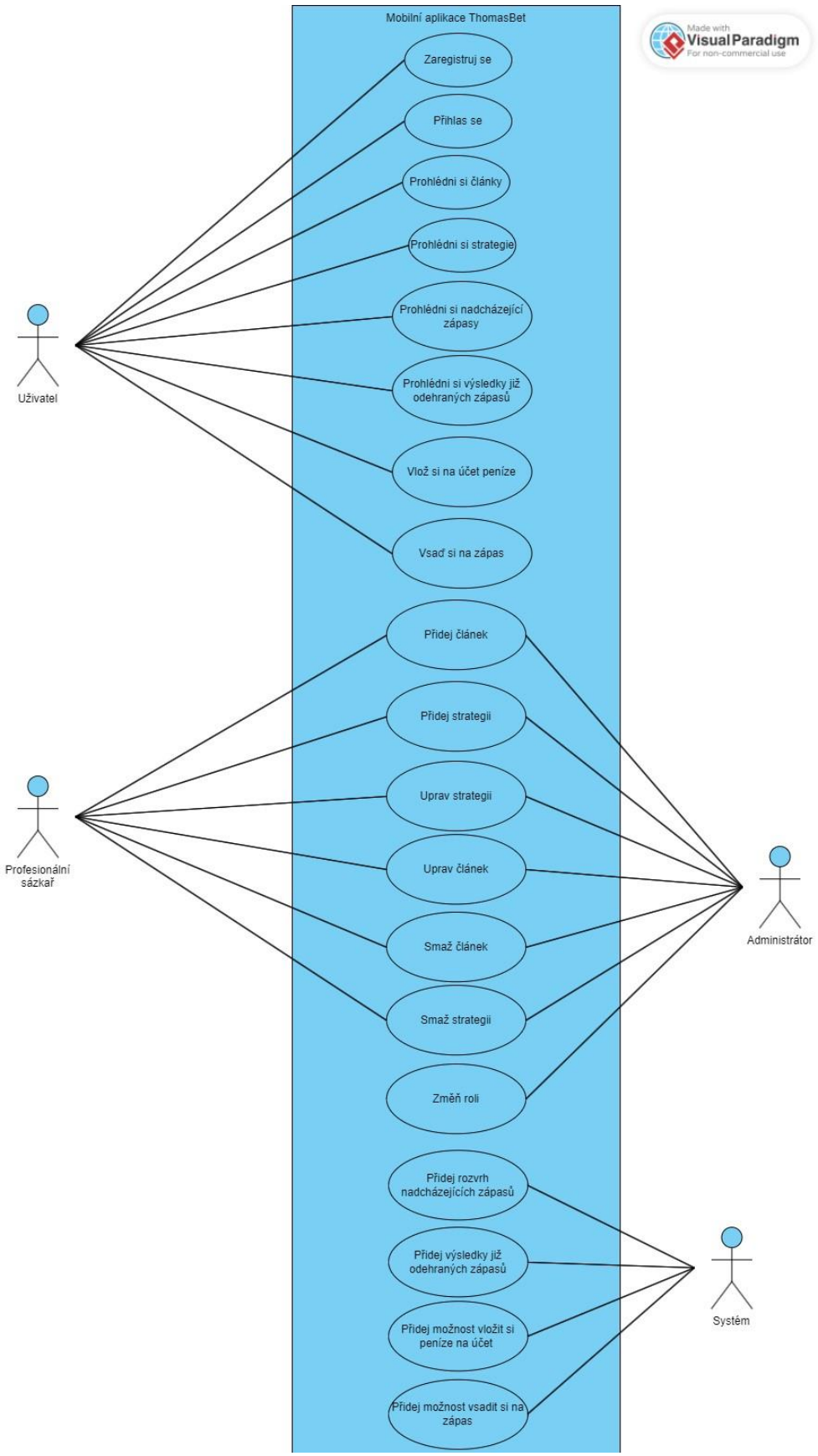
Administrátor má celkem 7 případu užití mobilní aplikace, přičemž přidání strategií a článků je zde pouze pro kontrolu správnosti funkcionalit. Tato funkce je poskytována primárně profesionálním sázkařům. Administrátor může článek nebo strategii kdykoliv upravit nebo odebrat. Má také jedinečnou funkci změnit jakémukoliv uživateli jeho roli.

Sázkař má celkem 8 případů užití mobilní aplikace. Registrace platí pouze pro nově příchozí sázkaře, ostatní využijí přihlášení do již existujícího účtu. Další případy jsou převážně prohlížení aplikace a poslední případ užití je uzavření sázky.

Profesionální sázkař má celkem 6 případů užití. Profesionální sázkaři převážně píše strategie a články, které mohou upravit nebo smazat.

Systém umožňuje určité případy užití a přizpůsobuje jej uživatelově volbě.

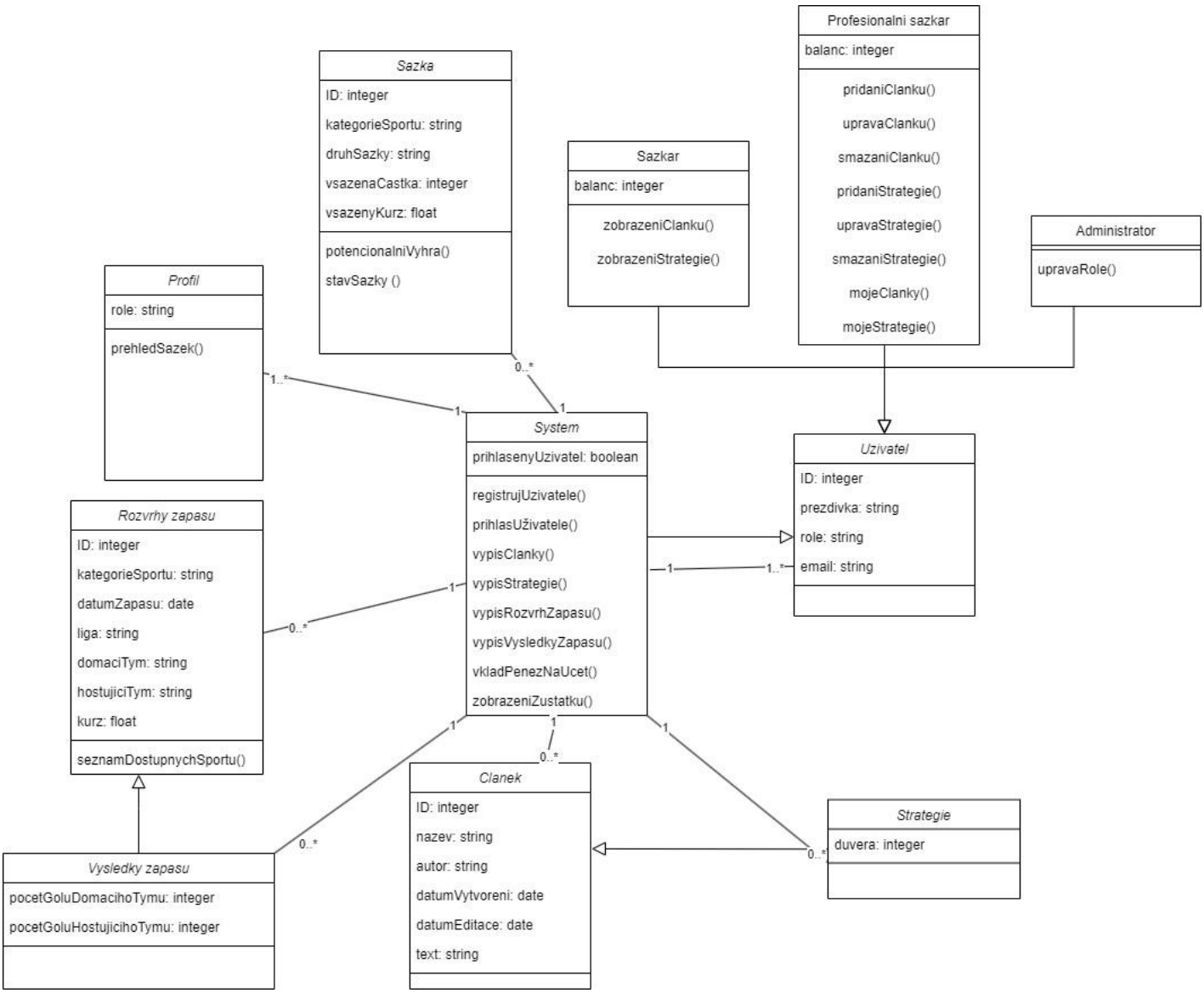
Na obrázku je Use Case diagram aplikace a jeho případy užití.



Obrázek 27 Use Case diagram

4.2.7 Diagram tříd

Součástí návrhu je také diagram tříd, který zobrazuje aktéry aplikace a jejich možnosti využívání funkcionalit aplikace. Pokud je přepsán diagram programátorem do kódu, musí znázornit všechny funkcionality aplikace.



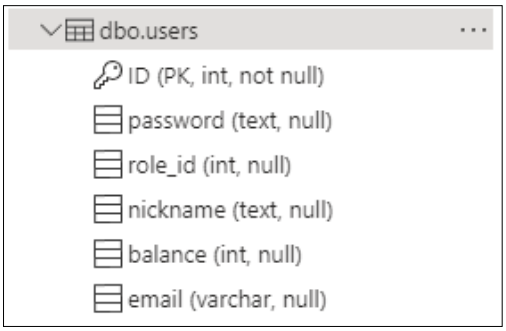
Obrázek 28 Diagram tříd

4.3 Struktura databáze

Veškeré data jsou uloženy v Microsoft Azure SQL databázi, ke které je zřízen přístup pouze vlastníkovi databáze. Ten se může přihlásit přímo do rozhraní uživatelským účtem nebo inicializovat přístup přes připojovací *string*, který musí obsahovat předdefinované SQL přihlášení administrátora.

4.3.1 Evidování uživatelů

Databáze obsahuje tabulku uživatelů s následující strukturou. První hodnota je primární klíč ID, který je unikátní pro každého uživatele identifikuje jej. Zároveň je sloupec omezen, aby hodnota nevznikla nulová, avšak takový scénář nikdy nemůže nastat, jelikož je zde aplikována funkce *Identity*, často označované jako *Auto Increment*, což znamená, že při každém novém záznamu se hodnota inicializuje a přičte k předchozí hodnotě číslo 1.



Obrázek 29 Databázová tabulka pro evidování uživatelů

V databázi jsou manuálně vytvořeny 3 základní uživatelské účty pro účely demonstrace. Vzorový účet administrátora s hodnotou role 1, která reprezentuje administrátora. Druhý účet náleží profesionálnímu sázkaři s rolí 2, která reprezentuje profesionálního sázkaře a jsou mu umožněny bonusové funkce, které klasický uživatelský účet nemá. Třetím účtem je již zmiňovaný klasický uživatel.

Uživatelská hesla jsou zašifrována pomocí funkce Bcrypt a při pokusu o nalezení některé dostupné dešifrovací stránky nebylo možné heslo rozšifrovat bez znalosti hesla, který *hash* reprezentuje.

ID	password	role_id	nickname	balance	email
1	$2a$10$o4867vsVA0zBfAMbi/F...	1	TomasAdmin	20000	tomas@admin.cz
2	$2a$10$VE.aOkaYsWcCucx/Cs...	2	TomasBetter	700	tomas@better.cz
3	$2a$10$s4kyCVQv0IIUnlk99Dd...	3	TomasUser	645	tomas@user.cz

Obrázek 30 Záznamy v databázové tabulce evidování uživatelů

4.3.2 Definování příchozích zápasů z API

Při řešení příchozích zápasů z API bylo hned několik omezení. Maximální měsíční počet požadavků 500 limitoval tolik, že byla nutnost přistupovat k datům jinak, než aby si jej každý uživatel načítal přes API.

Prvně návrh tabulky, která data se budou uchovávat z pole odpovědí API. Bylo potřeba uchovávat *sport_id* jako identifikátor, sport jako kategorii sportu, *league* jako ligu, *description* jako dodatečný popisek a *has_outrights* z důvodu ponechání NHL Stanley Cupu.

Obrázek 31 Databázová tabulka pro evidování sportů

Konkrétní data, která budou přijímána jsou zobrazeny na obrázku níže. Byly zde vybrány i jedinečné turnaje, jelikož v moment, kdy se hrají, neprobíhají většinou ostatní soutěžní ligy a aplikace by tak nevypisovala žádné zápasy. Byly zde zahrnuty nejvyšší fotbalové ligy a zároveň nejpopulárnější hokejová liga.

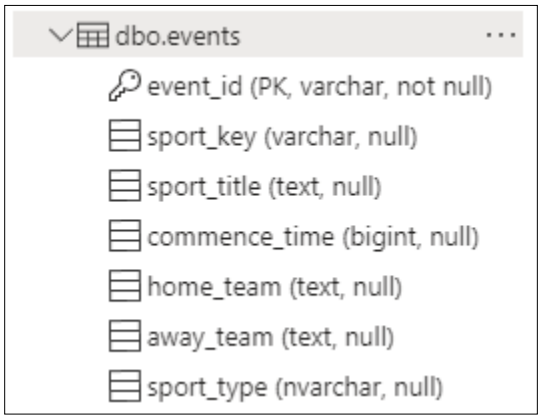
Tímto řešením se limituje počet požadavků, které budou na API posílány a nebudou vyvolávány uživatelem, jelikož mapování bude probíhat jednou denně, ideálně při prvním spuštění aplikace. Pokud by mapování zápasů nebylo omezeno, počet požadavků by mohl sázkař vypotřebovat během okamžiku.

sport_id	sport	league	description	has_outrights
icehockey_nhl	Ice Hockey	NHL	US Ice Hockey	False
icehockey_nhl_championship_winner	Ice Hockey	NHL Championship Winner	Stanley Cup Winner 2023/2024	True
soccer_france_ligue_one	Soccer	Ligue 1 - France	French Soccer	False
soccer_germany_bundesliga	Soccer	Bundesliga - Germany	German Soccer	False
soccer_italy_serie_a	Soccer	Serie A - Italy	Italian Soccer	False
soccer_spain_la_liga	Soccer	La Liga - Spain	Spanish Soccer	False
soccer_uefa_champs_league	Soccer	UEFA Champions League	European Champions League	False

Obrázek 32 Záznamy v databázové tabulce pro evidování sportů

4.3.3 Stáhnutí příchozích zápasů z API

Po nadefinování struktury přijímaných požadavků je potřeba přijmout od API samotné zápasy, které se uskuteční. Struktura byla zvolena podobná jako při definování zápasů. *Event_id* reprezentující identifikátor zápasu, *sport_key* jako cizí klíč propojující tabulku sportů a eventů, *sport_title* udávající ligu, *commence_time* je uložený čas zápasu, kdy se bude odehrávat ve formátu *unixtimestamp*, *home_team* a *away_team* reprezentující mužstva, která se spolu utkají a *sport_type*, pomocí kterého lze bude moct uživatel filtrovat zobrazované zápasy.



Obrázek 33 Databázová tabulka pro evidování zápasů

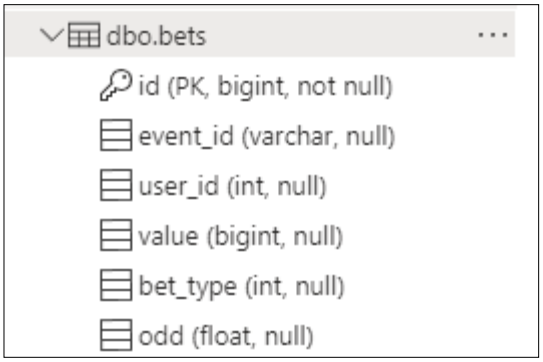
Níže je ukázka některých dat, která jsou aktuálně v databázi uložena. Nachází se zde reálná data zápasů, které se budou odehrávat v následujících dnech a sázkaři, který používá aplikaci tak bude umožněno si na tyto zápasy vsadit.

event_id	sport_key	sport_title	commence_time	home_team	away_team	sport_type
00b992e97755a9080affd...	soccer_germany_bundesl...	Bundesliga - Germany	1712341800000	Eintracht Frankfurt	Werder Bremen	Soccer
00e7d57d82b19500f7f26...	soccer_spain_la_liga	La Liga - Spain	1711894500000	Girona	Real Betis	Soccer
0125db269899a6fd7631...	soccer_uefa_champs_lea...	UEFA Champions League	1712689200000	Arsenal	Bayern München	Soccer
0158d5d1645e7bf43e5c0...	soccer_germany_bundesl...	Bundesliga - Germany	1711891800000	Augsburg	FC Koln	Soccer
036b5e2735b43b9248bd...	soccer_france_ligue_one	Ligue 1 - France	1712415600000	RC Lens	Le Havre	Soccer
03d62ba13460c56556a7...	soccer_spain_la_liga	La Liga - Spain	1713025800000	Mallorca	Real Madrid	Soccer
06b3d61db2f5d880daef1...	icehockey_nhl	NHL	1711677600000	Arizona Coyotes	Nashville Predators	Ice Hockey
0a994aac2e37342825b1c...	icehockey_nhl	NHL	1711674000000	Colorado Avalanche	New York Rangers	Ice Hockey
0f7c41bf84ba09760bd98...	soccer_france_ligue_one	Ligue 1 - France	1711814400000	Metz	AS Monaco	Soccer
0fa036b4b3bd50f8008f4...	soccer_germany_bundesl...	Bundesliga - Germany	1711809000000	RB Leipzig	FSV Mainz 05	Soccer

Obrázek 34 Záznamy v databázové tabulce pro evidování zápasů

4.3.4 Evidování sázek

Při ukládání sázek byly vzaty v potaz všechny požadavky, které by měl obsahovat výpis vsazených sázek. Z tohoto důvodu byla zvolena následující struktura. Id jako identifikátor sázky, *event_id* je cizí klíč definující zápas na který uživatel vsadil, *user_id* je cizí klíč definující uživatele, který na zápas vsadil, *value* je hodnota reprezentující výši vsazené částky, *bet_type* je předem definované číslo sázky, kdy 1 symbolizuje výhru domácího mužstva, a nakonec hodnota *odd*, která symbolizuje kurz, který byl na sázku vypsán.



Obrázek 35 Databázová tabulka pro evidenci sázek

V tabulce níže lze vidět některé z uskutečněných sázek a všechny údaje o nich. Sázky se budou vypisovat v sekci vsazených tiketů a hodnoty budou převedeny do formy, kde jim bude rozumět uživatel. Identifikátor zápasu se změní na jména týmů, kteří proti sobě zápas odehrály a typ sázky se změní na slovní vyjádření. Na základě vsazené částky a předdefinovaného kurzu se následně vypočítá potencionální výhra sázkaře.

id	event_id	user_id	value	bet_type	odd
1	00b992e97755a9080affd7f243f0fa5c	6	150	1	1,75
2	00e7d57d82b19500f7f2639309845494	4	150	1	1,70
4	0125db269899a6fd763134ea820abd68	3	150	2	3,07
5	00e7d57d82b19500f7f2639309845494	3	150	1	2,75
6	00b992e97755a9080affd7f243f0fa5c	2	100	6	2,51
7	00e7d57d82b19500f7f2639309845494	3	255	4	3,65

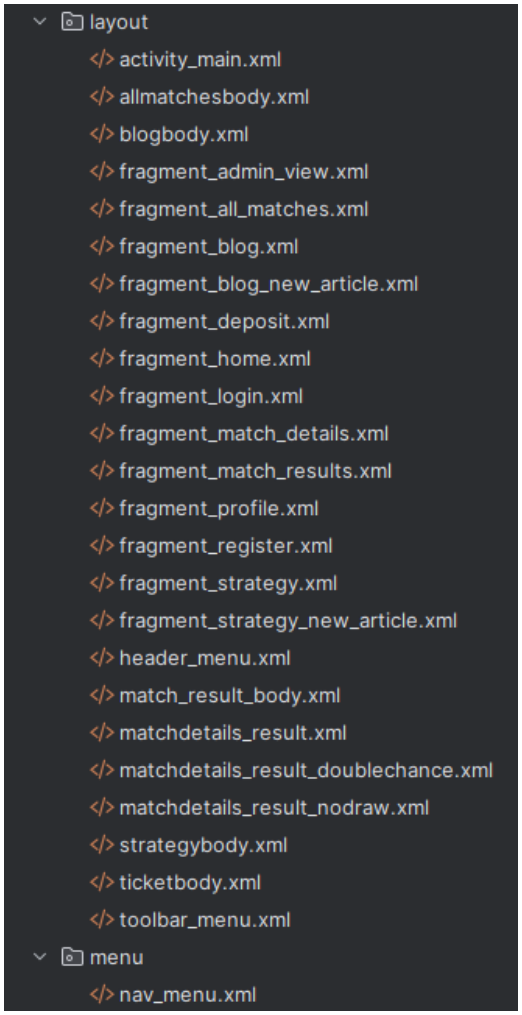
Obrázek 36 Záznamy v databázové tabulce pro evidenci sázek

4.4 Realizace aplikace

4.4.1 Design mobilní aplikace

Vzhled mobilní aplikace vychází z návrhů mobilní aplikace, avšak některé sekce byly upraveny pro zvýšení funkcionalit. Níže budou uvedeny pouze základní uživatelské rozhraní, avšak aplikace obsahuje mnohem více rozhraních, jak je vidět níže na obrázku.

Všechny sekce začínající slovem fragment reprezentují jednotlivé obrazovky, které uživatel může zobrazit. Ostatní rozhraní na obrázku jsou většinou předdefinované kostry dynamického obsahu, který se bude měnit v závislosti na akcích uživatele. Je zde také definováno rozpoložení prvků v navigačním menu, hlavička menu a také hlavička aplikace, která bude na každém uživatelském rozhraní stejná.



Obrázek 37 Všechny layouty použité v projektu

4.4.1.1 Design domovské obrazovky

Design domovské stránky byl navržen tak, aby obsahoval všechny důležité informace o aplikaci a zároveň ji představil uživateli. Je zde autor, logo firmy a základní tlačítka odkazující na registraci a přihlášení pro plné využívání aplikace. Pokud sázkař není přihlášen popřípadě ani zaregistrován, nebude v aplikaci zobrazován žádný obsah, jelikož menu nabízí pouze přesměrování na rozhraní domů,přihlášení a registraci.

Obrázek 38 Design domovské obrazovky

4.4.1.2 Design sekce přihlášení a registrace

Prvním krokem k vytvoření účtu je registrace, která vyžaduje vyplnění všech uvedených polí. V poli email je vyžadován znak „@“ reprezentující emailovou adresu. Pole přezdívky není nijak omezeno. První pole hesla se musí shodovat s druhým polem a bez splnění této podmínky není uživateli umožněna registrace. Pokud uživatel má již založený účet, pomocí odkazu pod tlačítkem se do něj může přihlásit.

Druhým krokem je přihlášení do již existujícího účtu, kdy uživatel musí správně vyplnit vyžadované údaje, které se budou shodovat s údaji uvedenými v databázi. Pokud se údaje shodují, bude přesměrován do aplikace a bude mu umožněn přístup k funkcím náležícím jeho roli.

Obrázek 39 Design sekce registrace

4.4.1.3 Design sekce strategie

Tato sekce je pro uživatele nejdůležitější, jelikož obsahuje důležité informace, které mohou sázkaře inspirovat při sázení. Dynamický obsah je vytvářen pomocí elementu *<RecyclerView>*. Strategie zobrazuje autora, datum vytvoření strategie a datum upravení strategie v případě, že byla strategie upravena. Strategie obsahuje textové pole, kde bude uživateli sděleno to nejdůležitější. Jsou zde dostupná tlačítka pro další stránku článků a předchozí stránku článků umožňující sázkaři zobrazovat další strategie.

Přihlášeným uživatelem je uživatel s přezdívkou „TomasBetter“, tudíž jsou zde ještě tlačítka pro úpravu nebo smazání u článku, který napsal. Na další stránce se zobrazí nové tlačítko na předchozí stránku, avšak na první stránce tlačítko zmizí.

Obrázek 40 Design sekce strategie první stránka

4.4.1.4 Design sekce blog

Sekce blogu je navržena podobně jako sekce strategií s rozdílem horní navigace. Horní 4 tlačíka tlačítka jsou statická, avšak sekci článků lze posunout dolů a přecíst si článek celý. Tato funkcionalita je umožněna opět pomocí kombinace *<ScrollView>*, který obsahuje dynamický kontent uložený v elementu *<RecyclerView>*.

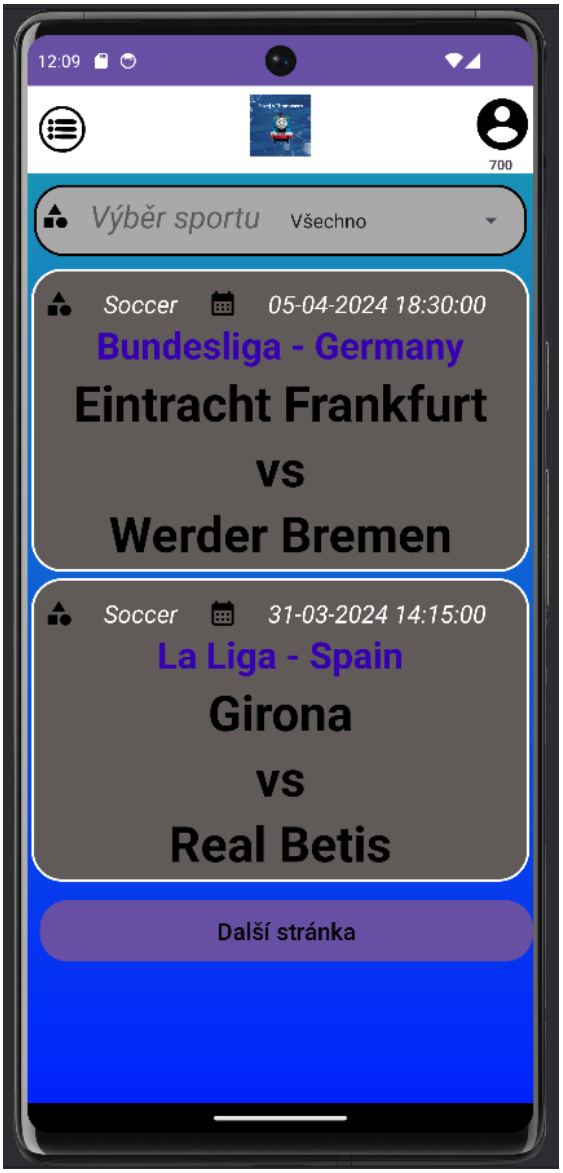
Přihlášeným uživatelem je uživatel s přezdívkou „TomasBetter“, tudíž jsou zde ještě tlačítka pro úpravu nebo smazání u článku, který napsal, jelikož se jedná o profesionálního sázkaře. Druhý článek je napsán jiným uživatelem a editace není tedy správně umožněna.

Obrázek 41 Design sekce blog

4.4.1.5 Design sekce nadcházejících zápasů a sázek na zápas

Sekce následujících zápasů obsahuje výběr sportu, které chce uživatel zobrazit. Základní hodnota jsou všechny sporty, avšak jsou zde dostupné ostatní sporty rozkliknutím rozbalovací nabídky. Po kliknutí jsou načteny a zobrazeny údaje z databáze.

Po kliknutí do pole zápasu je uživatel přesměrován na nabídku sázek na zápas. Na níže přiloženém obrázku vpravo lze vidět opět rozbalovací nabídku, kde jsou dostupné všechny sázky, které jsou na zápas nabízeny. Po vybrání sázky se změní její barva a sázkař může vepsat částku, kterou chce na zápas vsadit. Po vsazení tiketu je přesměrován do sekce profilu, kde se zobrazují zároveň vsazené tikety.



Obrázek 42 Design sekce nadcházejících zápasů

4.4.1.6 Design profilu a vsazených tiketů

Sekce profilu jako první poskytuje údaje o sázkaři, kde zobrazuje jeho roli spolu s uživatelským jménem. Dále je zobrazován finanční zůstatek na hráčském kontě a možnost přidat peníze na účet, pokud bude stav příliš nízký.

Profesionální sázkař, který je přihlášen na obrázku vlevo má možnost zobrazit všechny strategie a články, které jsou jím napsané. Sázkař tuto možnost nemá, jelikož mu není umožněno vytvořit článek nebo strategii.

Poslední sekce zobrazuje vsazené tikety, kde je přehled sázek uživatele. Je zde čas, kdy se zápas odehraje včetně týmů, které proti sobě hrají. Dále je zde sázka, kterou sázkař vybral a hodnota, kterou na ni vsadil. Posledním údajem je potencionální výhra, která odpovídá vynásobení vsazené částky spolu s kurzem, vypsaným na danou sázku. Pokud má sázkař vsazených více tiketů, zobrazuje se ještě tlačítko další stránky, které načte další vsazené tikety.

Obrázek 43 Design sekce profil

4.4.2 Logika zobrazování dynamického obsahu

Zobrazování dynamického obsahu je zajištěno následovně:

1. Obsah je uložen do modelu třídy, který obsahuje *raw values*, tedy nezpracovaná a nezformátovaná data. Tato třída obsahuje hodnoty nějaké entity, například strategie, reprezentující strukturu a obsah dat.
2. Tyto data jsou poskytovány určitému adaptéru, který obsahuje držitele (*view holdera)*, který je vytvořen pro každou položku, která bude později zobrazena v konečné fázi *RecyclerView*. Adaptér realizuje nové držitele pro každou jednotlivou položku, která mu přijde a naplňuje je daty z modelu.
3. Adaptér naplní data pomocí požadavku, který odešle na API endpoints, které odešlou požadavek na databázi. Databáze poskytne API data, která se vrátí a přeformátováním vrácených dat do adaptéru jsou uloženy reálná data z databáze.
4. Po naplnění všech dat do držitelů se data pošlou do konečné fáze *RecycleView*, který data oddělí od sebe a s dodatečnou funkcionalitou stránkování lze nahlížet na data jako na jednotlivé objekty, které jsou vykreslovány v uživatelském rozhraní.

4.4.3 Funkcionality aplikace

4.4.3.1 Registrace a přihlášení

Uživateli je umožněno vytvořit si účet v aplikaci, pomocí kterého bude moci využívat všechny funkcionality. Při vytvoření se jeho role automaticky nastaví na roli „Sázkař“, kterou lze rozdělovat profesionální sázkaře a rekreační sázkaře. Po vytvoření uživatelského účtu je účet uložen do databáze a je možné se k němu přihlásit. Data jsou odeslány do databáze pomocí API, avšak ještě před odesláním se zašifruje heslo a odesílá se již zašifrované.

Po správném vyplnění může uživatel pokračovat dále na kontrolu hesla. Heslo zadané v prvním poli se musí shodovat s heslem v druhém poli, aby se zabránilo tomu, že se uživatel překlepne a nebude tak své heslo znát. Pokud se tedy hesla neshodují, bude zobrazena hláška s vykřičníkem. Pod obrázkem je ukázka okomentovaného kódu, který tuto funkcionalitu zajišťuje.

Obrázek 44 Formulář registrace s vyskakovací chybovou hláškou

```
// ověření, že hesla jsou shodná, pokud ne, vyskočí chybová hláška
2 usages
private boolean validateFields(boolean hasFocus){
    if (!hasFocus) {
        String password = passwordField.getText().toString();
        String confirmPassword = confirmPasswordField.getText().toString();

        // pokud se pole hesla neshoduje s polem potvrzení hesla
        if (!password.equals(confirmPassword)) {
            confirmPasswordField.setError("Hesla se neshodují");
            return false;
        }else {
            // pokud se pole hesla shoduje s polem potvrzení hesla, nastaví se na konec pole ikona
            confirmPasswordField.setError(null);
            passwordField.setCompoundDrawablesWithIntrinsicBounds( left: 0,  top: 0, R.drawable.baseline_done_24,  bottom: 0);
            confirmPasswordField.setCompoundDrawablesWithIntrinsicBounds( left: 0,  top: 0, R.drawable.baseline_done_24,  bottom: 0);
            return true;
        }
    }
    return true;
}
```

Obrázek 45 Ukázka metody validateFields

4.4.3.2 Role v aplikaci

Aplikace rozeznává tři základní role, které mají předem vymezené pravomoci. Sázkař je každý nově vytvořený uživatel a jediný kdo mu může upravit roli je administrátor aplikace. Profesionální sázkař musí být určen administrátorem a ten mu musí roli změnit. Administrátor je jenom jeden a tím je autor aplikace.

4.4.3.2.1 Sázkař

Sázkaři je umožněno v aplikaci uzavírat sázky, číst články a strategie a vložit si peníze na účet. Uživatel nemůže vytvořit článek ani strategii, tyto funkcionality mají pouze profesionální sázkaři.

4.4.3.2.2 Profesionální sázkař

Tuto roli dostanou sázkaři po pečlivém prověření jejich verifikace administrátorem aplikace. Administrátor aplikace zaručuje verifikaci této role a budou ji disponovat opravdu pouze verifikovaní sázkaři. Profesionální sázkaři mají navíc právo vytvářet články a strategie. Mají také právo článek nebo strategii kdykoliv upravit nebo smazat. U upravených článků je evidován datum a čas, kdy byli upraveny.

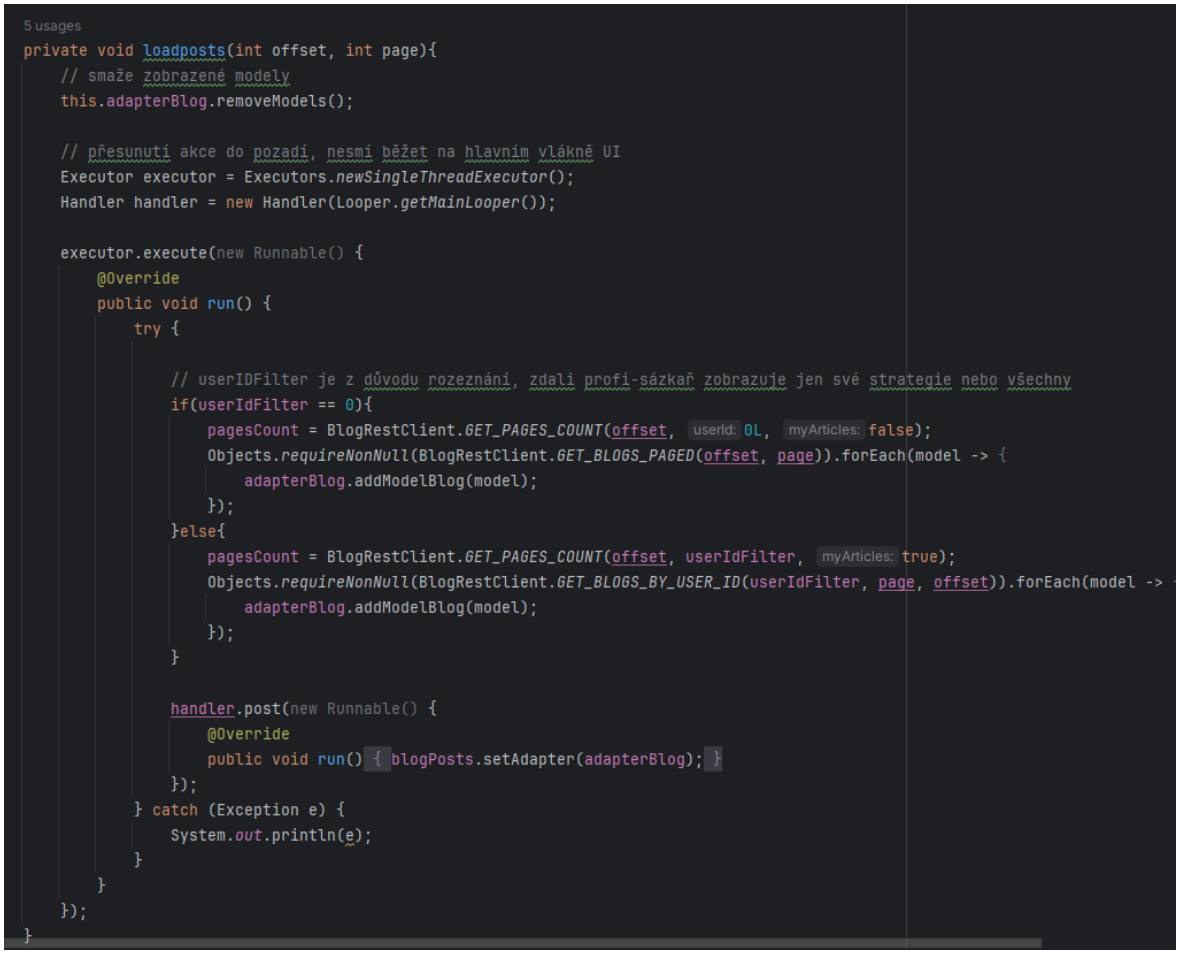
4.4.3.2.3 Administrátor

V aktuální době je pouze jeden administrátor, kterým je autor aplikace. Administrátor má všechny výše zmíněné funkce ohledně článků a strategií, avšak navíc má administrátorský panel, kde může měnit role uživatelům.

4.4.3.3 Blog

Články v blogu se načítají pomocí metody *loadposts(),* jejíž kód je zobrazený níže. Ve chvíli, kdy je zavolána tato metoda se smažou zobrazené modely a poté se na pozadí spustí nový proces.

Proces rozděluje, zdali se mají zobrazit články profesionálního sázkaře nebo všechny články pomocí podmínky *if – else*, která porovnává hodnotu proměnné *userIdFilter*, která je výše v kódu inicializována na počáteční hodnotu nula. Své vlastní články může zobrazit jen profesionální sázkař, což je ošetřeno pomocí *boolean* hodnoty *myArticles*. Poté je zavolán REST požadavek na server a po vrácení jsou hodnoty pomocí *forEach* uloženy do předdefinovaného modelu. Podrobnější popis modelu a této funkce se nachází v kapitole s názvem „Logika zobrazování dynamického obsahu“.



```
5 usages
private void loadposts(int offset, int page){
    // smaže zobrazené modely
    this.adapterBlog.removeModels();

    // přesunutí akce do pozadí, nesmí běžet na hlavním vlákně UI
    Executor executor = Executors.newSingleThreadExecutor();
    Handler handler = new Handler(Looper.getMainLooper());

    executor.execute(new Runnable() {
        @Override
        public void run() {
            try {

                // userIDFilter je z důvodu rozeznání, zdali profi-sázkař zobrazuje jen své strategie nebo všechny
                if(userIdFilter == 0){
                    pagesCount = BlogRestClient.GET_PAGES_COUNT(offset,  userId: 0L,  myArticles: false);
                    Objects.requireNonNull(BlogRestClient.GET_BLOGS_PAGED(offset, page)).forEach(model -> {
                        adapterBlog.addModelBlog(model);
                    });
                }else{
                    pagesCount = BlogRestClient.GET_PAGES_COUNT(offset, userIdFilter,  myArticles: true);
                    Objects.requireNonNull(BlogRestClient.GET_BLOGS_BY_USER_ID(userIdFilter, page, offset)).forEach(model ->
                        adapterBlog.addModelBlog(model);
                    });
                }

                handler.post(new Runnable() {
                    @Override
                    public void run() { blogPosts.setAdapter(adapterBlog); }
                });
            } catch (Exception e) {
                System.out.println(e);
            }
        }
    });
}
```

Obrázek 46 Ukázka metody loadposts

4.4.3.4 Sázení na zápasy

Sázení na zápasy je dostupné přes rozhraní rozvrh zápasů, kde si sázkař vybere zápas, na který chce vsadit. Po kliknutí na zápas je přenesen do rozhraní sázky. Zde jsou nabízeny předdefinované kategorie sázek pomocí voleb z elementu <*Spinner*>. Po vybrání kategorie sázek jsou zobrazeny vypsané sázky s kurzy.

Níže uvedený kód podle výběru kategorie sázky naplní tabulku předdefinovaným rozhraním sázek a uloží si hodnoty, které vybral uživatel do proměnných. Volají se zde také dvě další metody, které změní barvu pozadí vybrané sázky na zeleno a uloží si hodnotu kurzu ze zvolené sázky.

```
1 usage
private void showMatchResultBetType(){
    tableLayout.removeAllViews();
    textViews.clear();
    LayoutInflater inflater = LayoutInflater.from(getActivity());
    // layout kategorie sázek
    View tableRow = inflater.inflate(R.layout.matchdetails_result, root: null);

    //Inicializace a výpis tabulky dostupných sázek včetně kurzu
    TextView home, draw, away;
    home = tableRow.findViewById(R.id.result_homeTeam);
    home.setText(home.getText() + " " + String.valueOf(getRandomFloatNumber()));

    draw = tableRow.findViewById(R.id.result_Draw);
    draw.setText(draw.getText() + " " + String.valueOf(getRandomFloatNumber()));

    away = tableRow.findViewById(R.id.result_awayTeam);
    away.setText(away.getText() + " " + String.valueOf(getRandomFloatNumber()));

    textViews.add(home);
    textViews.add(draw);
    textViews.add(away);

    //listenery naslouchající, zdali na ně uživatel kliknul
    home.setOnClickListener(click -> {
        this.betTypes = BetTypes.HOME_WIN;
        markSelectedView(home);
        extractFloatValue(home);
    });
    draw.setOnClickListener(click -> {
        this.betTypes = BetTypes.DRAW;
        markSelectedView(draw);
        extractFloatValue(draw);
    });
    away.setOnClickListener(click -> {
        this.betTypes = BetTypes.AWAY_WIN;
        markSelectedView(away);
        extractFloatValue(away);
    });

    tableLayout.addView(tableRow);
```

Obrázek 47 Ukázka metody showMatchResultBetType

4.4.3.5 Profil

Na profilu jsou zobrazeny vsazené tikety, které jsou vypisovány podobně jako v případě blogu, konkrétně *REST callem*. Obsah je následně opět stránkován po dvojicích pro lepší zobrazení a čitelnost sázkařem. Pokud uživatel nemá žádné vsazené tikety, modely zůstanou prázdné a nebude zobrazeno nic. Kód metody je na obrázku níže.

```
3 usages
private void loadBets(int offset, int page){
    this.adapterMyBets.removeModels();

    Executor executor = Executors.newSingleThreadExecutor();
    Handler handler = new Handler(Looper.getMainLooper());

    executor.execute(new Runnable() {
        @Override
        public void run() {
            try {
                pagesCount = BetRestClient.GET_PAGES_COUNT(offset, (long) mainActivity.getUser().getId());
                BetRestClient.GET_MY_BETS(offset, page, (long) mainActivity.getUser().getId()).forEach(bet -> {
                    adapterMyBets.addModelBet(bet);
                });

                handler.post(new Runnable() {
                    @Override
                    public void run() {
                        bets.setAdapter(adapterMyBets);
                    }
                });
            } catch (Exception e) {
                System.out.println(e);
            }
        }
    });
}
```

Obrázek 48 Ukázka metody loadBets

Druhou metodou je metoda *updateButtons*, která se skládá ze dvou podmínek. První podmínka ošetřuje zmizení tlačítka „zobrazit předchozí stránku“, pokud se uživatel nachází na první stránce. Pokud se uživatel nachází na poslední stránce, zmizí tlačítko „zobrazit další stránku“. Tato funkce je využívána u blogů, strategií, rozvrhu zápasů i vypisování tiketů.

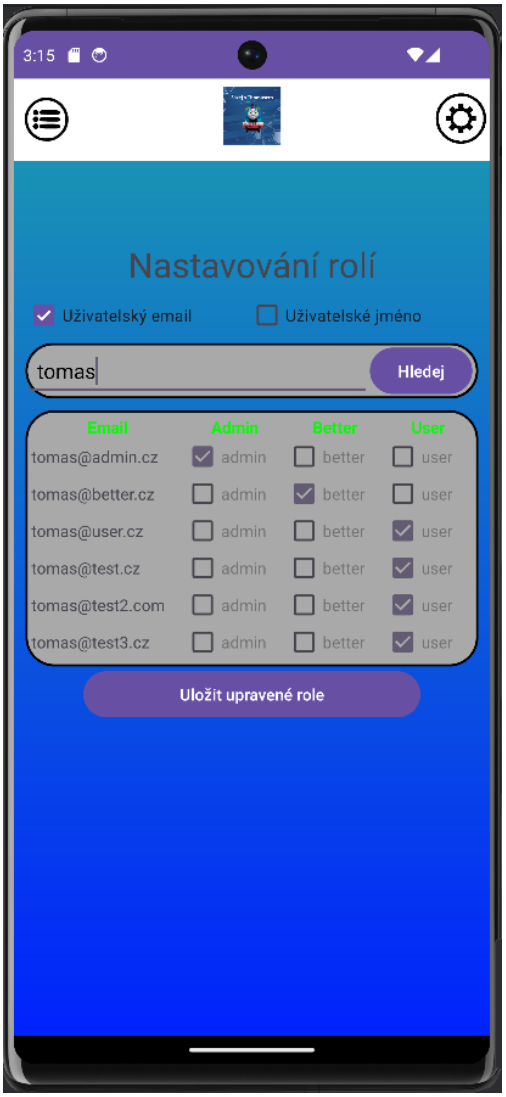
```
2 usages
private void updateButtons(){
    if(currentPage == 0){
        eventLoadless_btn.setVisibility(View.GONE);
    }else{
        eventLoadless_btn.setVisibility(View.VISIBLE);
    }

    if(currentPage < pagesCount - 1){
        eventLoadMore_btn.setVisibility(View.VISIBLE);
    }else{
        eventLoadMore_btn.setVisibility(View.GONE);
    }
}
```

Obrázek 49 Ukázka metody updateButtons

4.4.3.6 Upravení rolí uživatelům

Administrátor má právo a možnost upravit kdykoliv roli sázkaře. K tomuto mu je poskytnuto vyhledávací pole, pomocí kterého si může vyhledat uživatele na základě emailu nebo jejich uživatelského jména. U každého uživatele jsou zobrazeny 3 *checkboxy*, reprezentující role, které lze přiřadit. Při vyhledání jsou zaškrtnuty aktuální role uživatele. Administrátor může kliknutím na jiný checkbox změnit roli více uživatelů naráz a uložit změny pomocí tlačítka pod tabulkou. Řádky jsou do tabulky vkládány dynamicky podle aktuálně nalezených záznamů v databázi, odpovídající hledané hodnotě. Po upravení rolí je zobrazena hláška „Role byly upraveny“ jak lze vidět na obrázku níže vpravo a administrátor má tak jistotu, že všechno proběhlo správně.



Obrázek 50 Design nastavení uživatelských rolí

4.4.3.7 Zobrazení nadcházejících zápasů

Aplikace obsahuje rozvrh fotbalových a hokejových utkání, které jsou dostupné v sekci s názvem „Rozvrh zápasů“. Sekce obsahuje kategorizaci, zdali se mají zobrazovat všechny zápasy, fotbalové zápasy nebo hokejové. Po kliknutí na název jednoho z týmů je sázkař přesměrován na možnosti sázky na daný zápas.

4.4.3.8 Zobrazení výsledků odehraných zápasů

Aplikace zobrazuje výsledky fotbalových a hokejových utkání, které si lze zobrazit v kategorii „Výsledky zápasů“. Sekce obsahuje stejně jako rozvrh zápasů možnost zvolit si kategorii a na výběr jsou opět všechny zápasy, fotbalové zápasy nebo hokejové zápasy.

Obrázek 51 Design výsledků odehraných zápasů

4.4.3.9 Filtr vlastních strategií a článků

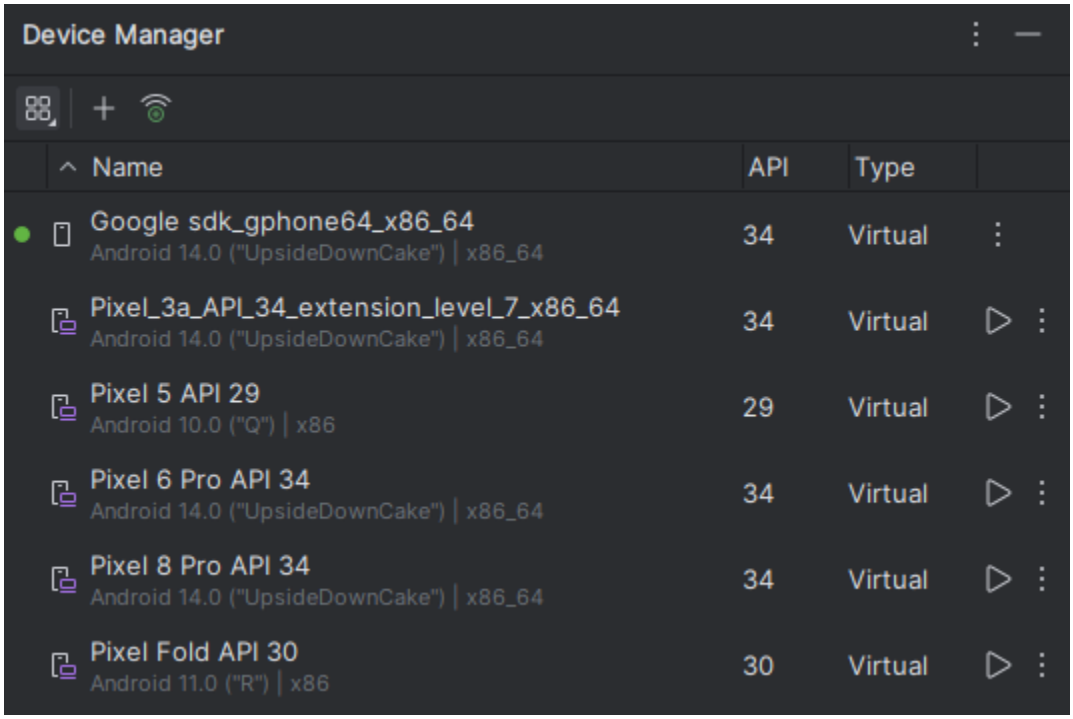
Jakýkoliv sázkař si může zobrazit všechny strategie a články. Profesionální sázkař může zobrazit pouze strategie a články, které napsal on a je mu tak usnadněno je spravovat, pokud by chtěl strategii změnit nebo smazat.

4.5 Testování aplikace

4.5.1 Emulace Android systému

Testování aplikace bylo prováděno pomocí emulovaných zařízení v Android Studiu. Vyzkoušena byla na mnoha mobilních telefonech s různými rozlišeními a verzemi operačního systému Android.

Zařízení jsou uvedena na obrázku níže.



Obrázek 52 Emulované zařízení v Android Studiu

Aplikace byla testována i na fyzickém mobilním zařízení, avšak musela být upravena z důvodu jiného prostředí. Emulované mobilní zařízení v Android Studiu využívají například jiné IP adresy pro připojení na localhost, kde bylo provozováno API endpoints, které se připojovalo na databázi.

Aplikace byla v průběhu vývoje testována skupinou sázkařů, kteří poskytovali zpětnou vazbu a testovali funkčnost aplikace. Aplikace po dokončení funguje stabilně a nebyl nalezen způsob, kdy by skončila chybovou hláškou a zavřela se.

4.6 Back end API endpoints

API *endpoints* je autorem vytvořený druhořadý projekt, který je přímo napojen na Azure databázi pomocí SQL přihlašovacích údajů. *Endpoints* fungují na adrese *localhostu*. Mobilní aplikace se připojuje na *endpointy* a komunikuje s nimi v případě, že vyžaduje údaje z databáze. Funguje tedy jako prostředník mezi databází a aplikací pro zvýšení bezpečnosti dat. Aplikace komunikuje s *endpointy* pomocí HTTP požadavků a ty odesílají požadavky na databázi. Toto řešení je zvoleno z důvodu bezpečnosti dat, jelikož databáze je stále chráněna vestavěným firewallem společnosti Microsoft Azure, kde je umožněna komunikace pouze autorovi aplikace. V budoucnu by se toto omezení muselo odstranit, aby aplikace fungovala i ostatním uživatelům.

Data jsou extrahovány z databáze pomocí tzv. *REST calls*, které obsahují HTTP požadavky GET, pro čtení dat a jejich vrácení do aplikace, POST pro vytváření nových dat v databázi, PATCH pro upravování dat v databázi a DELETE pro mazání dat v databázi. *REST call* se odesílá na *webserver*, který posílá SELECT dotazy na databázi a vrací data zpátky.

Připojení na databázi je nadefinováno v souboru vlastností a je poskytováno přímo od služby Microsoft Azure.

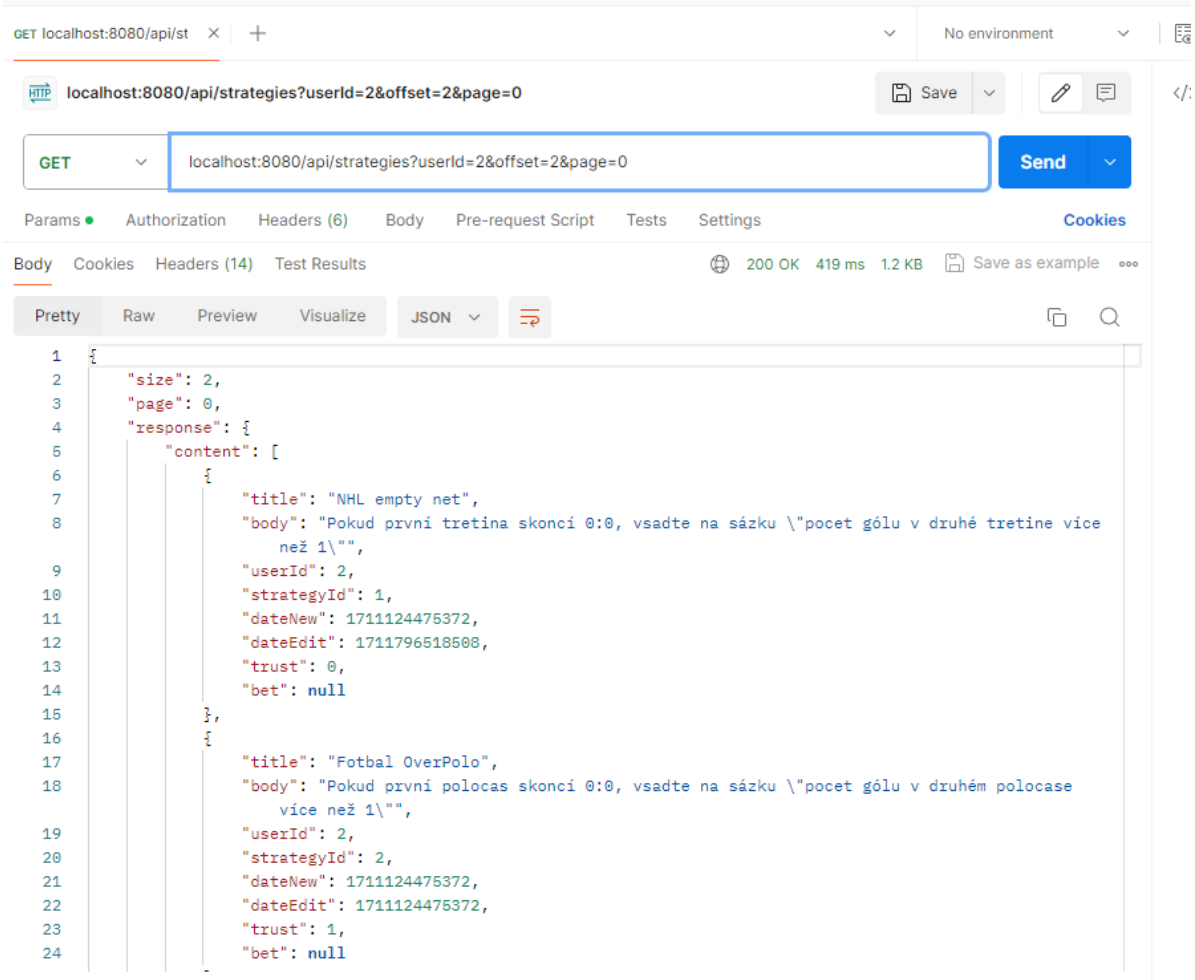
```
spring.application.name=endpoints
spring.datasource.url=jdbc:sqlserver://thomasbetsql.database.windows.net:1433;databas
spring.datasource.username=Thomas
spring.datasource.password=[redacted]
spring.datasource.driver-class-name=com.microsoft.sqlserver.jdbc.SQLServerDriver

odds.api.key=a58adb9d4cf563b95e31893a22a148ea
```

Obrázek 53 Konfigurace připojení na Azure databázi

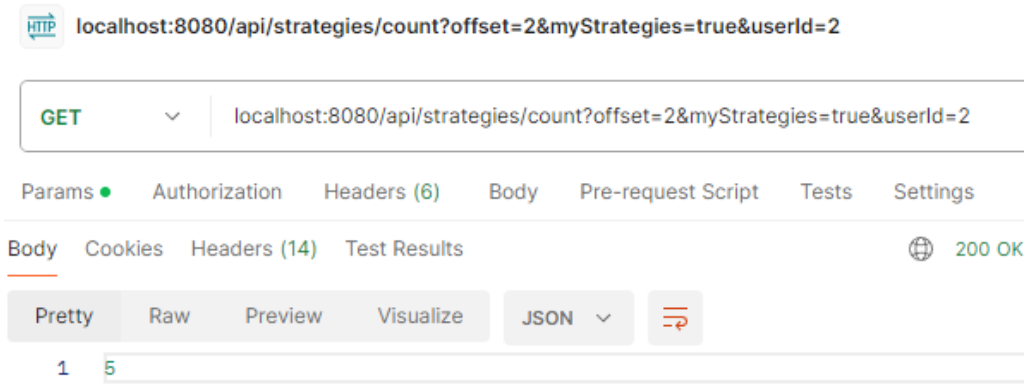
Ověřování funkčnosti nadefinovaných metod bylo prováděno přes software Postman, který umožňuje zobrazení dat přes http požadavky. Postman byl používán pro kontrolu funkčnosti serverové části, která se dotazuje přímo databáze.

Následující požadavek umožňuje vypsání článků, které napsal přihlášený uživatel. Pomocí definování parametru uživatelského ID lze vyhledat v databázi články shodující se s tímto ID a následně jej lze zobrazit v aplikaci uživateli. Parametry *offset* a *page* poskytují implementaci stránkování, aby nevzniklo nekončící posouvání, pokud bude k dispozici větší množství článků. Parametr *offset* tedy definuje počet článků, které se zobrazí na jedné stránce a pomocí parametru *page* je možné vypsat jednotlivé stránky.



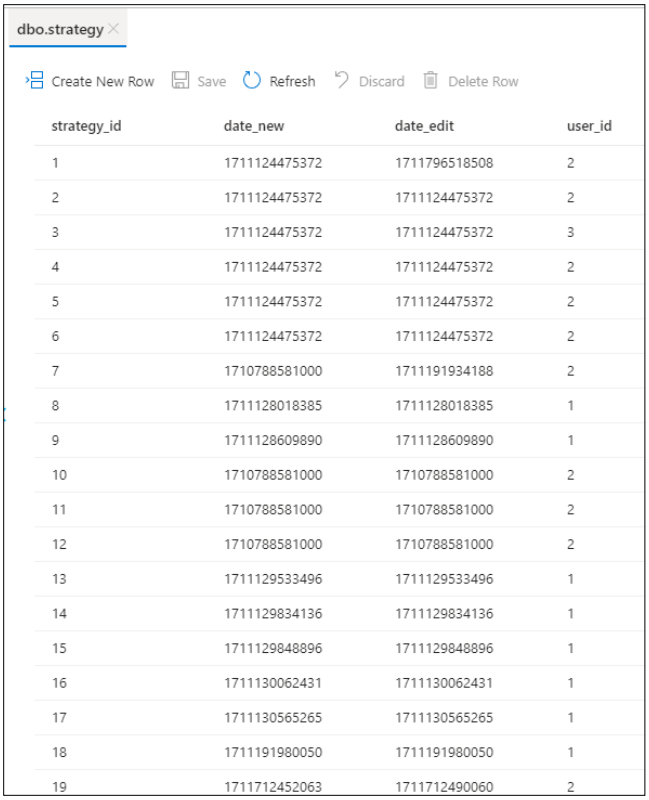
Obrázek 54 Ukázka JSON výpisu ze softwaru Postman

Tímto způsobem jsou zobrazovány strategie uživateli, avšak aby bylo možné skrývat tlačítka další a předchozí stránka, musí aplikace znát počet článků, které ji jsou poskytnuty. Tuto funkcionalitu zajišťuje metoda *count*, která počítá vrácené JSON pole. Z následujícího příkladu lze zjistit, že uživatel s identifikátorem 2 napsal 10 článků. Logika je taková, že vrácená hodnota je počet stránek, které musí být k dispozici za předpokladu, že se na jednu stránku vypíšou dva články.



Obrázek 55 Ukázka JSON výpisu ze softwaru Postman, metoda počítání článků

Tuto informaci lze ověřit v databázi pro ověření, že je funkcionalita naprogramována správně. Databáze skutečně obsahuje 10článků od uživatele s identifikátorem 2.



strategy_id	date_new	date_edit	user_id
1	1711124475372	1711796518508	2
2	1711124475372	1711124475372	2
3	1711124475372	1711124475372	3
4	1711124475372	1711124475372	2
5	1711124475372	1711124475372	2
6	1711124475372	1711124475372	2
7	1710788581000	1711191934188	2
8	1711128018385	1711128018385	1
9	1711128609890	1711128609890	1
10	1710788581000	1710788581000	2
11	1710788581000	1710788581000	2
12	1710788581000	1710788581000	2
13	1711129533496	1711129533496	1
14	1711129834136	1711129834136	1
15	1711129848896	1711129848896	1
16	1711130062431	1711130062431	1
17	1711130565265	1711130565265	1
18	1711191980050	1711191980050	1
19	1711712452063	1711712490060	2

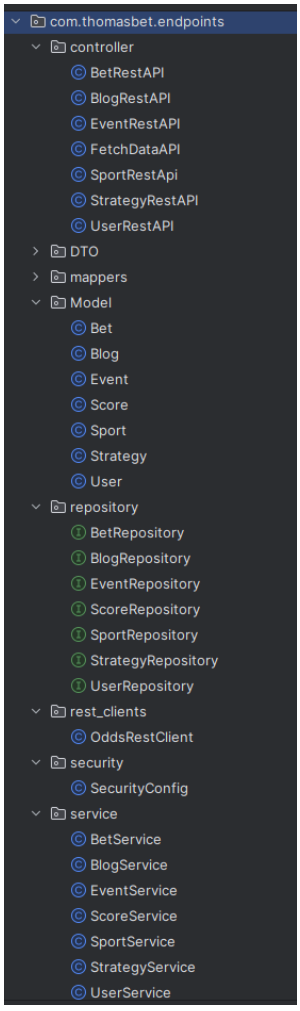
Obrázek 56 Záznamy v databázové tabulce strategií

Endpoins mají strukturu rozdělenou na několik *package*, neboli balíčků čímž je zajištěna přehlednost. Balíček controller obsahuje definování všech požadavků, které jsou aplikací vznášeny na databázi pomocí *REST calls*. *Endpoints* musí mít přirazeny entity, aby dokázali komunikovat s databází.

Tato funkcionalita je v balíčku Model, kde je definována struktura databáze a jsou zde vytvořeny Entity. Pro lepší pružnost dat jsou vytvořeny balíčky *mappers* a DTO, kdy se data pomocí *mapper* přemění v DTO a lze s nimi dále pracovat a lépe je číst.

Balíček *repository* obsahuje repositáře, které jsou dále využívány v serverové části. Obsahuje softwarové balíčky, které jsou využívány, konkrétně JPA repositář, který je *interface* pro Spring boot, který je využíván. Poskytuje CRUD metody, kterými jsou *Create, Read, Update* a *Delete*, tedy vytvoření, čtení, úprava a smazání pro správu entit.

Baliček *service* definuje konkrétní požadavky, které jsou aplikací dotazovány pro naplnění všech funkcionalit, které aplikace nabízí.



Obrázek 57 Struktura API endpoints

4.6.1 Načítání dat z Odds API

V aktuální verzi aplikace je potřeba načítat data z API ručně, a to kvůli omezenému měsíčnímu počtu dotazů, které lze vznést na externí službu *Odds API*. Pro stažení aktuálních zápasů je potřeba spustit adresu webového serveru a přidat za ni „/api/fetch/events/all“ pro načtení aktuálních dat a přemazání historických dat. Jsou také definované funkce pro načtení skóre u všech zápasu, načtení skóre jen u jednotlivé kategorie sportu a smazání všech načtených skóre.

Kód, který tyto funkcionality umožňuje je přiložen níže, avšak většina z nich volá opět další metody, které jsou implementovány v jiné části kódu.

V budoucích verzích aplikace by se tato událost dala nastavit tak, že při každém načtení aplikace se tento *fetch* provede automaticky, avšak načítání aplikace by bylo značně zpomaleno tímto procesem.

```
no usages
@GetMapping("/events/all")
public void fetchAllEvents(){
    removeEventData();
    sportService.getAllSports().forEach(sport -> {
        fetchEventsFromOddsBySportKey(sport.getSportId());
    });
}

1 usage
@GetMapping("/events/remove/all")
public void removeEventData() { eventService.removeAllData(); }


1 usage
@GetMapping("/scores/{sportKey}")
public void fetchScoresFromOddsBySportKey(@PathVariable String sportKey) throws MalformedURLException {
    oddsRestClient.getScoresBySportKey(sportKey).forEach(score -> scoreService.saveScore(score));
}

no usages
@GetMapping("/scores/all")
public void fetchAllScores(){
    removeScoresData();
    sportService.getAllSports().forEach(sport -> {
        try {
            fetchScoresFromOddsBySportKey(sport.getSportId());
        } catch (MalformedURLException e) {
            throw new RuntimeException(e);
        }
    });
}

1 usage
@GetMapping("/scores/remove/all")
public void removeScoresData() { scoreService.removeAllData(); }
```

Obrázek 58 Ukázka metod určených pro fetchování dat

4.6.2 Ukázka servisu strategií

Na obrázku níže lze vidět kód, který byl již demonstrován při ukázek pomocí softwaru Postman. Funkcionalitu, kterou Postman zobrazoval v kapitole 4.6 zajišťoval níže zobrazený kód, který specifikoval sestavení URL pomocí „@RequestParam“. Tato anotace *frameworku* Spring zajišťuje vyžadování parametrů pro správné zobrazení HTTP požadavku. Parametry jsou zadány jako nepovinné.

```
@RestController
@RequestMapping(path = "/api/strategies")
public class StrategyRestAPI {

    8 usages
    @Autowired
    private StrategyService strategyService;

    no usages
    public StrategyRestAPI(StrategyService strategyService) { this.strategyService = strategyService; }


    no usages
    @GetMapping
    public PageableResponse<Page<StrategyDTO>> getStrategies(@RequestParam(required = false) Long userId,
                                                             @RequestParam(required = false) int offset,
                                                             @RequestParam(required = false) int page){
        Page<StrategyDTO> strategiesList = strategyService.getStrategiesByUserIdPaged(userId, offset, page);


        return new PageableResponse<>(offset, page, strategiesList);
    }
```

Obrázek 59 Ukázka servisu strategií a metody getStrategies

Součástí kódu je také volání metody „*getStrategiesByUserIdPaged*“, která se nachází v balíčku *strategyService*. Na níže přiloženém obrázku lze vidět tuto metodu, která kromě definování počtu strategií na stránce a počtu stránek transformuje data pro přenos mezi rozdílnými vrstvami aplikace. Objekt strategie je transformován do formátu Strategie DTO, který je přenositelný přes HTTP požadavek.

```
1 usage
public Page<StrategyDTO> getStrategiesByUserIdPaged(Long id, int offset, int page){

    return strategyRepository.getStrateigesByUserId(id, PageRequest.of(page, offset))
            .map(strategy -> strategyMapper.strategyToStrategyDTO(strategy));
}
```

Obrázek 60 Ukázka metody getStrategiesByUserIdPaged

4.7 Požadavky pro spuštění aplikace

4.7.1 Softwarové požadavky

Mobilní aplikace je spustitelná pouze na mobilních zařízeních s operačním systémem Android. Mobilní aplikace byla vyvíjena na emulovaných mobilních zařízeních s nejnovější verzí operačního systému Android 14.0 („*UpsideDownCake*“), avšak je zaručena zpětná kompatibilita na emulovaných zařízeních do verze Android 10.0 („*Q*“). Aplikace byla testována i na fyzických mobilních zařízeních s nejstarším testovaným operačním systémem Android 9.0. Avšak pro funkcionalitu na fyzických mobilních zařízeních není aplikace uzpůsobena. Aplikace byla vyvinuta v prostředí Android studia, především díky vestavěné funkci emulace operačního systému.

4.7.1.1 Android Studio

Pro spuštění mobilní aplikace je vyžadována instalace vývojového prostředí Android Studio. Aplikace byla testována ve verzi Android Studia „*Hedgehog* | 2023.1.1 Patch 2“ a „*Iguana* | 2023.2.1 Patch 1“.

Dále je vyžadováno otevření projektové složky v Android Studiu, čímž se projekt implementuje. Poslední akcí v Android Studiu je definování emulovaného mobilního zařízení, na kterém se má aplikace spustit. Doporučena je nejnovější verze operačního systému Android 14.0 („*UpsideDownCake*“).

4.7.1.2 Databázový server

Pro správné fungování aplikace je nutné vytvořit vlastní databázi, protože autorova zkušební licence služby Microsoft Azure vypršela. V příloze bude uveden kompletní skript pro vytvoření stejné databáze.

4.7.1.3 IntelliJ Idea

Pro spuštění API endpoints je doporučen software IntelliJ Idea, konkrétně verze IntelliJ IDEA Community Edition 2023.3.5, která byla využívána.

4.7.2 Hardwarové požadavky

Po spuštění všech výše uvedených technologií může nastat situace, kdy počítač slabší třídy nezvládne takové množství vykreslování kvůli nedostatku RAM paměti. Spouštění bylo testováno s 12 GB RAM paměti obsažených ve stolním počítači a sestavování projektu nebylo vždy úspěšné, jelikož počítač nezvládal všechny operace naráz. Spouštění Endpoints v IntelliJ IDEA a vykreslování mobilní aplikace v emulovaném prostředí využívalo 95 % RAM paměti. Z výše uvedených důvodů je doporučeno testování aplikace na počítačích alespoň střední třídy hardwarových komponent.

5 Výsledky a diskuse

5.1 Porovnání podobné sázkařské aplikace s řešením autora

V České republice neexistuje žádná aplikace, která by umožňovala profesionálním sázkařům inspirovat svými analýzami ostatní sázkaře a zároveň umožňovala přímou možnost na zápas vsadit. Nejbližší podobnou aplikací je aplikace sázkařské společnosti Tipsport, který ovšem umožňuje vytvářet analýzy na zápasy komukoliv a nerozlišuje kvalifikované a rekreační sázkaře, čímž snižuje důvěrnost těchto analýz.

Ostatní aplikace sázkových kanceláří neumožňují sázkaři přidat jakoukoliv analýzu nebo poznatek k zápasu, a proto nebudou porovnávány. Jedna z hlavních funkcionalit aplikace ThomasBet je právě analyzování zápasu profesionálními sázkaři, a proto aplikace nejsou relevantní konkurencí.

Z výše uvedených důvodu bude aplikace porovnávána s aplikací Tipsportu, jelikož obsahuje analýzy k zápasům a možnost sázkařům vsadit si na ně.

5.1.1 Aplikace sázkové kanceláře Tipsport

Aplikace sázkové kanceláře Tipsport je nejpoužívanější českou sázkařskou aplikací, jelikož sázková kancelář Tipsport má největší počet aktuálních klientů.

Jedním z nejvyhledávanějších funkcí je funkce *darkmode*, která umožňuje grafické zobrazení aplikace změnit na tmavší odstíny barev, čímž působí šetrněji na oči a displej mobilního telefonu tolik nevyzařuje.

Aplikace vyžaduje přihlášení pro využití jejího maximálního potenciálu a následně umožňuje uživateli, aby si toto přihlášení uložil na aktuálním zařízení, což určitě usnadní uživateli přístup do aplikace.

Aplikace umožňuje sázkaři rozsáhlou sázkařskou nabídku na veškeré druhy sportů, včetně kuriozních sázek například při volebním období na vítěze voleb. Současně jsou sázkaři umožněny sázky přímo v již probíhajícím zápase.

5.1.2 Porovnání Tipsport a ThomasBet aplikace

V této kapitole bude krátké porovnání aplikace Tipsport s moji diplomovou aplikací, kterou jsem nazval ThomasBet.

Porovnání výsledků bylo zvoleno formou krátké tabulky, která zobrazuje výhody a nevýhody aplikace. Porovnány jsou na základě vybraných funkcionalit.

Předmět porovnávání	**Tipsport**		**ThomasBet**	
	Výhody	Nevýhody	Výhody	Nevýhody
Sportovní nabídka	Obsáhlá sportovní nabídka, která vyjde vstříc všem uživatelům a umožní jim vsadit na sport, který chtějí.	Příliš mnoho sportů, sázkař často dlouho hledá sport, na který chce vsadit.	Přehledná a jednoduchá sportovní nabídka pro nejpopulárnější sporty.	Některé sporty jsou vynechány z důvodu úspory a náročnosti aplikace.
Live sázky	Umožněny, široký výběr sázek.	Často nestabilní kurzy, výpadky služby.	Nedisponuje touto funkcí	Nedisponuje touto funkcí
Rozvrh zápasů	Umožňuje zobrazit rozvrh zápasů na několik dní dopředu.	Žádné.	Umožňuje zobrazit rozvrh zápasů na několik dní dopředu.	Žádné.
Evidence uživatelů včetně jejich rolí	Eviduje uživatele.	Není možnost role, existují pouze sázkaři a administrátoři.	Evidence uživatelů včetně rolí (profesionální sázkař, obyčejný uživatel a administrátor).	Žádné
Strategie	Nedisponuje touto funkcí.	Nedisponuje touto funkcí.	Strategie napsány pouze od profesionálních sázkařů.	Žádné

Tabulka 1 Porovnání aplikace Tipsport a ThomasBet

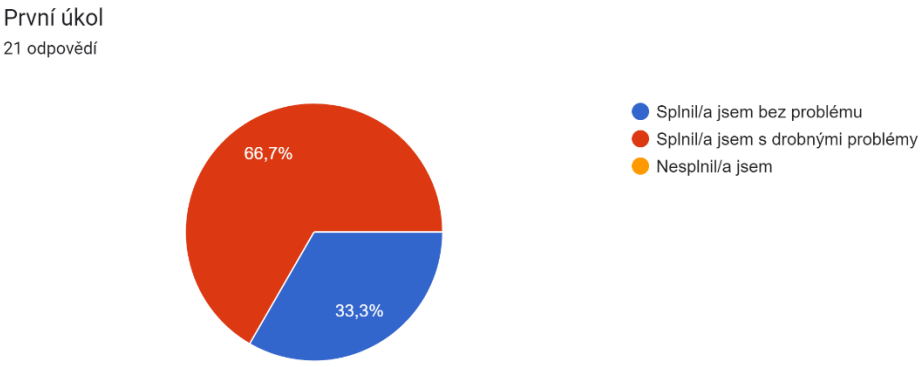
5.2 Výsledky dotazníku z průběhu vývoje

V průběhu vývoje byla aplikace testována skupinou sázkařů a následně byli požádáni o vyplnění dotazníku. Sázkaři si mohli aplikaci libovolně proklikat a zároveň jsem je požádal o splnění dvou úkolů. Dotazník byl vytvořen pomocí služby Google formuláře.

Jejich odpovědi jsou interpretovány formou grafů odpovědí z dotazníku. Celkový počet odpovědí na dotazník je 21.

1. Prvním úkolem bylo zaregistrování se a následné přihlášení do účtu.

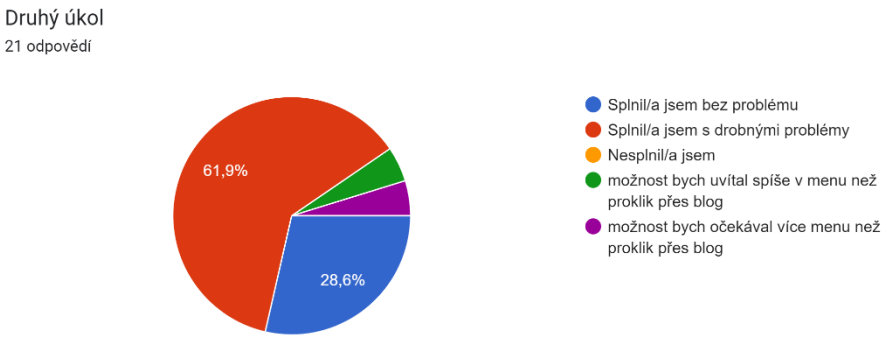
Podle odpovědí z dotazníku, které jsou vidět na níže přiloženém grafu zvládla tento úkol většina uživatelů bez problémů. Pár uživatelů s drobnými problémy a nebyl nikdo, kdo by úkol nesplnil



Graf 1 Graf interpretující odpovědi prvního úkolu

2. Druhým úkolem bylo zobrazení všech strategií v aplikaci.

V této fázi byli strategie dostupné pouze přes sekci blogu, což podle uživatelů nebylo dostatečně intuitivní. Z tohoto důvodu byla následně vytvořena v menu speciální sekce strategie.



Graf 2 Graf interpretující odpovědi druhého úkolu

6 Závěr

V rámci této diplomové práce byla vytvořena mobilní aplikaci s názvem ThomasBet, která umožňuje uživatelům sázet na zápasy anebo se inspirovat strategiemi, které jsou napsány do aplikace profesionálními sázkaři. Aplikace byla vytvořena pomocí vývojového prostředí Android Studio s použitím značkovacího jazyku XML a programovacího jazyku Java. Dále byla využita zkušební verze Azure databáze od společnosti Microsoft. Připojení na ni bylo zajištěno pomocí ručně nakonfigurovaných *API endpoints* pro zvýšení bezpečnosti.

Z analýzy dostupných aplikací vyplynulo, že žádná česká sázková kancelář nespolupracuje s profesionálními sázkaři a v tomto ohledu je vytvořená aplikace jedinečná. Nashromážděná data, která jsou interpretována v sekci strategie jsou volně dostupné informace profesionálních sázkařů.

Při tvorbě samotné aplikace byly představeny funkční a nefunkční požadavky na aplikaci. Dále byly vytvořeny návrhy uživatelského rozhraní, které vychází z návrhů v autorově bakalářské práci. Byly vytvořeny také diagramy, znázorňující chování aplikace. Poté byla popsána struktura databáze a fáze při realizaci aplikace. Kapitola je zakončena požadavky na spuštění aplikace. Za předpokladu, že budou splněny všechny požadavky je zaručena funkčnost aplikace.

Aplikace byla navržena a realizována podle předem definovaných cílů v zadání a všechny vytyčené cíle, které byly stanoveny, tak byly splněny.

7 Seznam použitých zdrojů

1. Sohail, Shehryar. How Do Odds Work in Betting? *Investopedia.* [Online] 2023. [Citace: 17. 01. 2024.] https://www.investopedia.com/articles/investing/042115/betting-basics-fractional-decimal-american-moneyline-odds.asp.
2. republika, Česká. Zákon č. 186/2016 Sb., Zákon o hazardních hrách. *Zákony pro lidi.* [Online] [Citace: 18. 01. 2024.] https://www.zakonyprolidi.cz/cs/2016-186/zneni-20240101.
3. Schwartz, David. *Roll the Bones: The History of Gambling.* : Winchester Books, 2013. ISBN 978-0-615-84778-8.
4. Glimne, Dan. Gambling History. *Britannica.* [Online] [Citace: 04. 03. 2024.] https://www.britannica.com/topic/gambling/History.
5. Williams, John. Betting with odds: History of Betting and how it came to czech republic. *Prague Post.* [Online] 2021. [Citace: 18. 01. 2024.] https://www.praguepost.com/blog/betting-odds-history-how-it-came-to-czech-republic.
6. Passion Predict. What is a Void Bet ? – Why a bet could be voided. *Passion Predict.* [Online] 2023. [Citace: 25. 01. 2024.] https://passionpredict.com/news/what-is-a-void-bet-why-a-bet-could-be-voided/.
7. Tipsport redakce. Částečný cashout je ve hře! *Tipsport.* [Online] 2022. [Citace: 16. 01. 2024.] https://www.tipsport.cz/blogy/castecny-cashout-je-ve-hre/43811.
8. Expert, Betting. Value Bet Explained. *Punch.* [Online] 2023. [Citace: 07. 02. 2024.] https://punchng.com/betting/how-to-bet/value-bet/.
9. Buchdahl, Joseph. What is a good ROI in sports betting? *Pinnacle.* [Online] 2023. [Citace: 17. 01. 2024.] https://www.pinnacle.com/betting-resources/en/educational/what-is-a-good-roi-in-sports-betting/lcf2wkdja6v3upnb.
10. Jurek, Kamil. Co je dvojtá šance? *BET ARENA.* [Online] 2020. [Citace: 04. 01. 2024.] https://www.betarena.cz/rubriky/skola-sazeni/co-je-dvojita-sance_1677.html.
11. Redakce. Seznam českých legálních online sázkových kanceláří. *Encyklopedie Hazardu.* [Online] 2024. [Citace: 19. 02. 2024.] https://www.encyklopediehazardu.cz/rubriky/kurzove-sazeni/seznam-ceskych-legalnich-online-sazkovych-kancelari_344.html.
12. Jurek, Kamil. Bet 365 sázení - společnost nemá v ČR licenci. *BET ARENA.* [Online] 2020. [Citace: 25. 01. 2024.] https://www.betarena.cz/rubriky/sazkove-kancelare/bet-365-sazeni-spolecnost-nema-v-cr-licenci_5096.html.
13. Prunner, Pavel. *Gamblerství aneb Ztráta svobody.* Plzeň : Aleš Čeněk s.r.o, 2013. ISBN 978-80-7380-452-7.
14. Indeed Editorial Team. What Is User Interface (UI)? *Indeed.* [Online] 2022. [Citace: 24. 03. 2024.] https://www.indeed.com/career-advice/career-development/user-interface.
15. Arkadiusz, Janik. 6 Steps Of UX/UI Design Process. *Selleo.* [Online] 2023. [Citace: 07. 01. 2024.] https://selleo.com/blog/6-steps-of-uxui-design-process.
16. Guimaraes e Equipe Aela, Felipe. Persona: Why Is It Essential for Any UX Design Project? *Aelaschool.* [Online] 2023. [Citace: 27. 02. 2024.] https://aelaschool.com/en/userexperience/persona-essential-ux-design-project/.
17. Hartinger, David. Lekce 3 - UML - Use Case Specifikace. *itnetwork.cz.* [Online] [Citace: 28. 02. 2024.] https://www.itnetwork.cz/navrh/uml/uml-use-case-specifikace-diagram.

18. UML Class Diagram Tutorial. *Visual Paradigm*. [Online] [Citace: 30. 03 2024.] https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/.
19. Sebesta W., Robert. *Concepts of programming languages — 10th ed.* : Electronic computers, 2012. ISBN 978-0-13-139531-2.
20. Banerjee, Pinaki. A Categorical List of programming languages. *geeksforgeeks.* [Online] 2023. [Citace: 16. 02. 2024.] https://www.geeksforgeeks.org/a-categorical-list-of-programming-languages/.
21. Cloudinary. Front-End Development: The Complete Guide. *Cloudinary.* [Online] 2024. [Citace: 21. 03. 2024.] https://cloudinary.com/guides/front-end-development/front-end-development-the-complete-guide.
22. HTML basics. *M mdn web docs.* [Online] 2024. [Citace: 19. 03. 2024.] https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics.
23. CSS: Cascading Style Sheets. *M mdn web docs.* [Online] 2024. [Citace: 21. 03. 2024.] https://developer.mozilla.org/en-US/docs/Web/CSS.
24. Coursera Staff. 5 Types of Programming Languages. *Coursera.* [Online] 2023. [Citace: 15. 01. 2024.] https://www.coursera.org/articles/types-programming-language.
25. Your Guide to The Top 15 Backend Languages For 2023. *Ishir.* [Online] 2023. [Citace: 11. 03. 2024.] https://www.ishir.com/blog/75047/your-guide-to-the-top-15-backend-languages-for-2023.htm.
26. Aditya. A Complete Guide to Learn XML For Android App Development. *geeksforgeeks.* [Online] 2022. [Citace: 08. 01. 2024.] https://www.geeksforgeeks.org/a-complete-guide-to-learn-xml-for-android-app-development/.
27. Kosek, Jiří. *XML pro každého, podrobný průvodce.* Praha : Grada Publishing, 2000. ISBN 80-7169-860-1.
28. Method in Java. *JavaTpoint.* [Online] [Citace: 08. 01. 2024.] https://www.javatpoint.com/method-in-java.
29. Java Tutorial. *JavaTpoint.* [Online] [Citace: 19. 01. 2024.] https://www.javatpoint.com/java-tutorial.
30. Mayur, Patil. Java Basic Syntax. *geeksforgeeks.* [Online] 2023. [Citace: 19. 03. 2024.] https://www.geeksforgeeks.org/java-basic-syntax/.
31. Instance Variable in Java. *JavaTpoint.* [Online] [Citace: 19. 01. 2024.] https://www.javatpoint.com/instance-variable-in-java.
32. Deronjic, Vojin. A Comprehensive Guide to C++: Advantages and Disadvantages. *Pangea.* [Online] 2023. [Citace: 26. 01. 2024.] https://pangea.ai/blog/languages/a-comprehensive-guide-to-c-advantages-and-disadvantages.
33. Davis, Ronald. Top Advantages of C and C++ Compared to Other Programming Languages. *Invensis.* [Online] 2022. [Citace: 19. 02. 2024.] https://www.invensis.net/blog/c-and-c-plus-plus-benefits-over-other-programming-languages.
34. Codecademy Team. What Is an IDE? *Code_cademy.* [Online] [Citace: 07. 03. 2024.] https://www.codecademy.com/article/what-is-an-ide.
35. Android Studio Iguana | 2023.2.1. *Developers Android.* [Online] [Citace: 12. 01. 2024.] https://developer.android.com/studio/releases.
36. Android studio. *Developer Android.* [Online] [Citace: 18. 01. 2024.] https://developer.android.com/studio.
37. Meet Android Studio. *Developers Android.* [Online] 2024. [Citace: 09. 02. 2024.] https://developer.android.com/studio/intro.

38. Configure your build. *Developers Android.* [Online] 2024. [Citace: 20. 03. 2024.] https://developer.android.com/build.
39. Run apps on the Android Emulator. *Developers Android.* [Online] [Citace: 16. 01. 2024.] https://developer.android.com/studio/run/emulator.
40. Vaibhav, Ahire. When to use Fragments VS Activities in Android App? *Tutorials point.* [Online] 2023. [Citace: 26. 01. 2024.] https://www.tutorialspoint.com/when-to-use-fragments-vs-activities-in-android-app.
41. Ramesh, Reza. Understanding the Differences Between Activities and Fragments in Kotlin. *Stackademic Blog.* [Online] 2023. [Citace: 25. 02. 2024.] https://blog.stackademic.com/understanding-the-differences-between-activities-and-fragments-in-kotlin-006605e9e2f2.
42. Mayukha, Lasya. An introduction to Java NetBeans. *Educative.* [Online] [Citace: 25. 02. 2024.] https://www.educative.io/answers/an-introduction-to-java-netbeans.
43. Pradeep, Kumar. Difference between Mobile and Desktop Operating System. *Tutorials point.* [Online] 2023. [Citace: 23. 01. 2024.] https://www.tutorialspoint.com/difference-between-mobile-and-desktop-operating-system.
44. Pavel, Junek. Operační systém Android. *It network.* [Online] [Citace: 08. 01. 2024.] https://www.itnetwork.cz/mobilni-zarizeni/android/android-operacni-system-google.
45. Number of Android apps on Google Play (Mar 2024). *AppBrain.* [Online] 2024. [Citace: 28. 03. 2024.] https://www.appbrain.com/stats/number-of-android-apps.
46. Check out these new Android 14 features. *Android.* [Online] [Citace: 24. 02. 2024.] https://www.android.com/android-14/.
47. Chakraborty, Sandip. Android 13 vs Android 14: Which one is better in 2024? *Sportskeeda.* [Online] 2024. [Citace: 18. 03. 2024.] https://www.sportskeeda.com/gaming-tech/android-13-vs-android-14.
48. Sayan. Waterfall Model – Software Engineering. *geeksforgeeks.* [Online] 2024. [Citace: 18. 03. 2024.] https://www.geeksforgeeks.org/waterfall-model/.
49. Agile Model. *JavaTpoint.* [Online] [Citace: 12. 01. 2024.] https://www.javatpoint.com/software-engineering-agile-model.
50. Tanzu, VMware. Spring Data. *Spring.* [Online] 2023. [Citace: 23. 02. 2024.] https://spring.io/projects/spring-data.
51. Sport betting API covering odds from bookmakers around the world. *Odds-api.* [Online] [Citace: 18. 01. 2024.] https://the-odds-api.com/.
52. Jurek, Kamil. Legální online kurzové sázení v Česku. *Betarena.* [Online] 2024. [Citace: 29. 3. 2024.] https://www.betarena.cz/rubriky/sportovni-clanky/legalni-online-kurzove-sazeni-v-cesku_961.html.
53. Oracle Java Documentation. Oracle Java Documentation. *The Java™ Tutorials.* [Online] [Citace: 19. 02. 2024.] https://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html.
54. [Online] [Citace: 30. 03 2024.] https://online.visual-paradigm.com/repository/images/7b0f5ce7-8799-4cb5-bc1f-cbd3e6aecf9b.png.
55. [Online] https://www.researchgate.net/profile/Divya-Keshamoni/publication/267042696/figure/fig2/AS:295690191294467@1447509443499/GSN-mobile-app-class-diagram.png.

8 Seznam obrázků, tabulek, grafů a zkratek

8.1 Seznam obrázků

Obrázek 1 Use Case diagram (54) 20
Obrázek 2 Diagram tříd (55) 21
Obrázek 3 Diagram struktury XML (25) 24
Obrázek 4 Výsledek diagramu v aplikaci (25) 24
Obrázek 5 Ukázka XML elementů (26) 25
Obrázek 6 Ukázka XML atributů 25
Obrázek 7 Ukázka XML textové části (26) 26
Obrázek 8 Ukázka XML komentáře 27
Obrázek 9 Transformace Java programu (28) 28
Obrázek 10 Ukázka Java tříd 29
Obrázek 11 Ukázka Java syntaxe metody 30
Obrázek 12 Ukázka Java proměnných a jejich deklarace 31
Obrázek 13 Ukázka Java komentáře (29) 31
Obrázek 14 Ukázka Gradle souborů v Android Studiu 34
Obrázek 15 Ukázka vytvoření nového emulovaného zařízení 34
Obrázek 16 Ukázka vytvořeného emulovaného mobilního zařízení 35
Obrázek 17 Ukázka připojování k emulátoru 35
Obrázek 18 Ukázka připojeného fyzického zařízení 36
Obrázek 19 Ukázka jedné iterace v agilním modelu (49) 41
Obrázek 20 Kurzová nabídka aplikace Tipsport 44
Obrázek 21 Kursová nabídka aplikace Fortuna 45
Obrázek 22 Návrh uživatelského rozhraní domovské obrazovky 49
Obrázek 23 Návrh uživatelského rozhraní sekce strategie 50
Obrázek 24 Návrh uživatelského rozhraní sekce články 51
Obrázek 25 Návrh uživatelského rozhraní sázky na zápas 52
Obrázek 26 Návrh uživatelského rozhraní vsazených tiketů 53
Obrázek 27 Use Case diagram 55
Obrázek 28 Diagram tříd 56
Obrázek 29 Databázová tabulka pro evidování uživatelů 57
Obrázek 30 Záznamy v databázové tabulce evidování uživatelů 57
Obrázek 31 Databázová tabulka pro evidování sportů 58
Obrázek 32 Záznamy v databázové tabulce pro evidování sportů 58
Obrázek 33 Databázová tabulka pro evidování zápasů 59
Obrázek 34 Záznamy v databázové tabulce pro evidování zápasů 59
Obrázek 35 Databázová tabulka pro evidenci sázek 60
Obrázek 36 Záznamy v databázové tabulce pro evidenci sázek 60
Obrázek 37 Všechny layouty použité v projektu 61
Obrázek 38 Design domovské obrazovky 62
Obrázek 39 Design sekce registrace 63
Obrázek 40 Design sekce strategie první stránka 64
Obrázek 41 Design sekce blog 65
Obrázek 42 Design sekce nadcházejících zápasů 66
Obrázek 43 Design sekce profil 67
Obrázek 44 Formulář registrace s vyskakovací chybovou hláškou 69

Obrázek 45 Ukázka metody validateFields 69
Obrázek 46 Ukázka metody loadposts 71
Obrázek 47 Ukázka metody showMatchResultBetType 72
Obrázek 48 Ukázka metody loadBets 73
Obrázek 49 Ukázka metody updateButtons 73
Obrázek 50 Design nastavení uživatelských rolí 74
Obrázek 51 Design výsledků odehraných zápasů 75
Obrázek 52 Emulované zařízení v Android Studiu 76
Obrázek 53 Konfigurace připojení na Azure databázi 77
Obrázek 54 Ukázka JSON výpisu ze softwaru Postman 78
Obrázek 55 Ukázka JSON výpisu ze softwaru Postman, metoda počítání článků 79
Obrázek 56 Záznamy v databázové tabulce strategií 79
Obrázek 57 Struktura API endpoints 80
Obrázek 58 Ukázka metod určených pro fetchování dat 81
Obrázek 59 Ukázka servisu strategií a metody getStrategies 82
Obrázek 60 Ukázka metody getStrategiesByUserIdPaged 82

8.2 Seznam tabulek

Tabulka 1 Porovnání aplikace Tipsport a ThomasBet 86

8.3 Seznam grafů

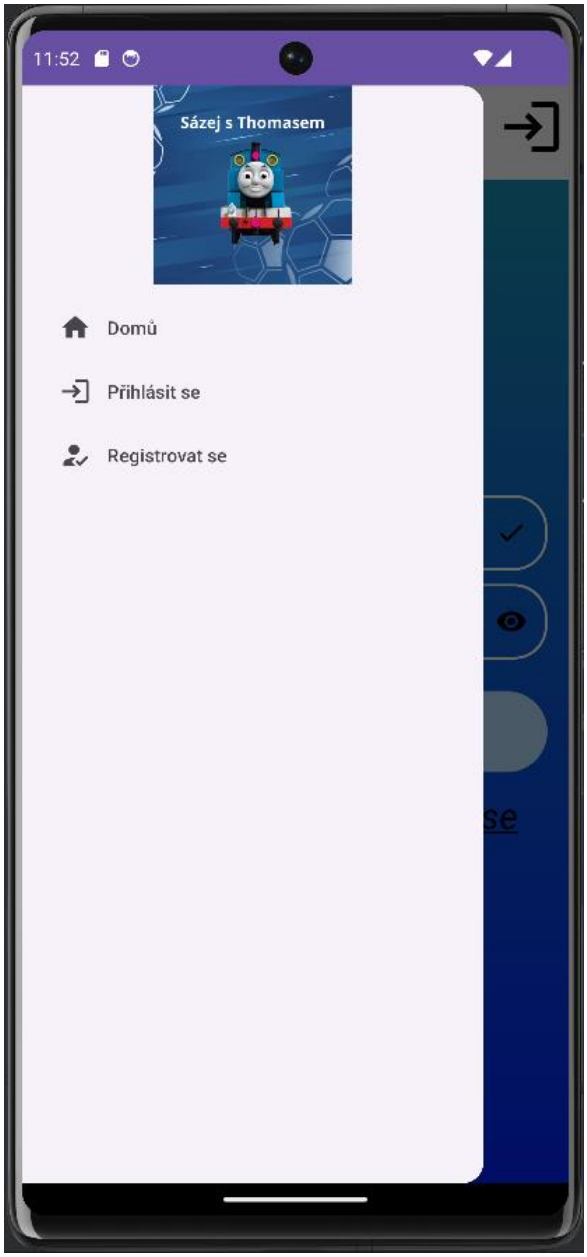
Graf 1 Graf interpretující odpovědi prvního úkolu 87
Graf 2 Graf interpretující odpovědi druhého úkolu 87

8.4 Seznam použitých zkratek

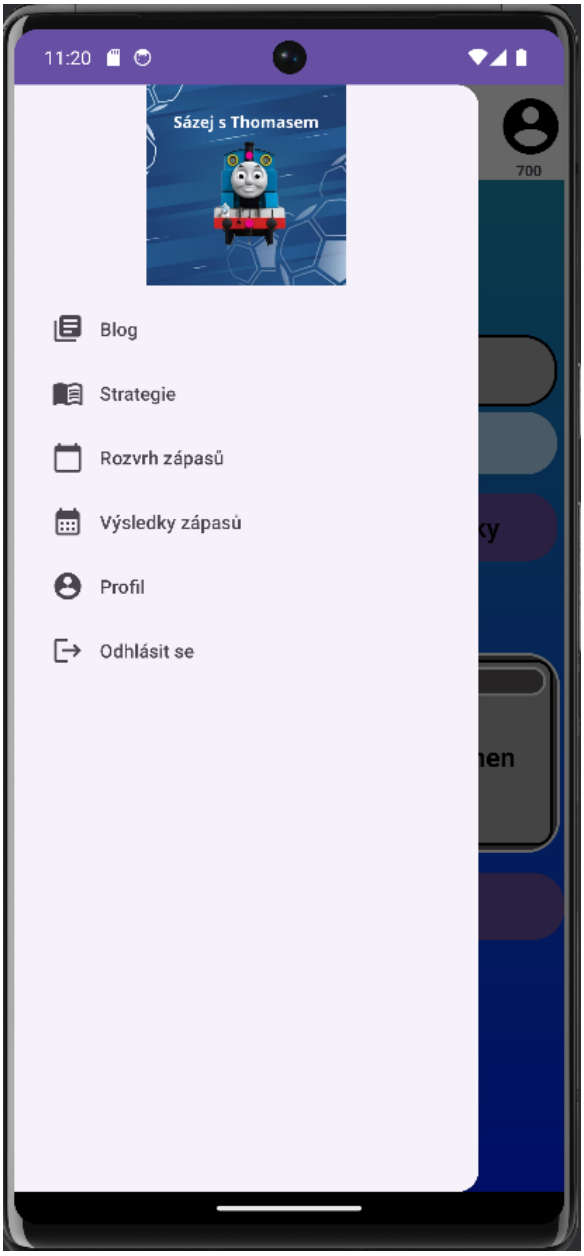
UI – User interface
ID – Identification
IDE – Integrated Drive Electronics
API – Application programming interface
URL – Uniform Resource Locator
DTO – Data Transfer Object

9 Přílohy

Příloha 1 Screenshot z aplikace – menu pro přihlášeného uživatele 95
Příloha 2 Screenshot z aplikace – menu pro nepřihlášeného uživatele 95
Příloha 3 Screenshot z aplikace – přihlašovací obrazovka 96
Příloha 4 Screenshot z aplikace – profil sázkaře.....97
Příloha 5 Screenshot z aplikace – nastavování rolí pro administrátora 98
Příloha 6 Screenshot z aplikace – sekce strategie s ukázkou stránkování 99
Příloha 7 Screenshot z aplikace – sekce strategie.....99
Příloha 8 Screenshot z aplikace – ukázka možnosti sázky na zápas.....100
Příloha 9 USB flash disk se zdrojovým kódem aplikace, zdrojovým kódem API Endpoints a skript pro vytvoření SQL databáze.



Příloha 1 Screenshot z aplikace pro nepřihlášeného uživatele

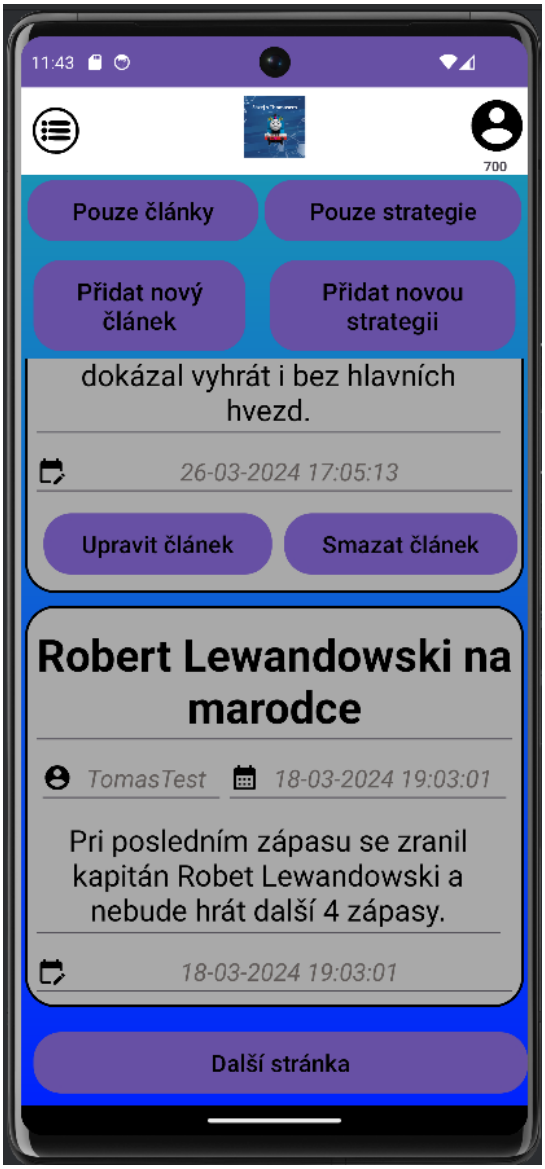


Příloha 2 Screenshot z aplikace — menu pro přihlášeného uživatele menu

Příloha 3 Screenshot z aplikace – přihlašovací obrazovka

Příloha 4 Screenshot z aplikace – profil sázkaře

Příloha 5 Screenshot z aplikace – nastavování rolí administrátorem



Příloha 7 Screenshot z aplikace – sekce strategie

Příloha 6 Screenshot z aplikace – sekce strategie s ukázkou stránkování

Příloha 8 Screenshot z aplikace – ukázka možnosti sázky na zápas