FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	Implementace modulu vstupenky, forum a virtuální body pro aplikaci Elittero
Student:	Bc. Hanna Korniushyna
Vedoucí:	Ing. Josef Pavlíček, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2024/2025

Pokyny pro vypracování

Analyzujte a vytvořte moduly pro aplikaci Elittero. Postupujte dle následující metodiky:
Sesbírejte a analyzujte uživatelské požadavky na moduly vstupenky, Virtuální body a chat. U chatu zajistěte naplnění požadavku filtrace případných vulgarizmů.
Navrhněte architekturu modulů.
Moduly naprogramujte ve frameworku Flutter.
Vaše moduly otestujte formou uživatelského testování.
Výsledky testu popište, moduly opravte.
Diskutuje výsledky a formulujte závěr.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 31. října 2023 v Praze.

Diplomová práce

IMPLEMENTACE MODULŮ VSTUPENKY, FÓRUM A VIRTUÁLNÍ BODY PRO APLIKACI ELITERRO

Bc. Hanna Korniushyna

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Josef Pavlíček, Ph.D.
8. května 2024

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěla bych poděkovat vedoucímu mé práce Ing. Josefu Pavlíčkovi, Ph.D. za odborné vedení, přátelský přístup a cenné konzultace. Dále děkuji své rodině a přátelům za jejich neustálou podporu během mého studia.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 8. května 2024

Abstrakt

Tato diplomová práce se zaměřuje na analýzu, návrh a implementaci vybraných sekcí ”Virtuální body”, ”Fórum fanoušků” a ”Nákup a správa vstupenek” s využitím frameworku Flutter. Výsledek této práce bude sloužit jako základ pro obdobné moduly v aplikaci Eliterro.

Rešerše prezentovaná v této práci poskytuje zdůvodnění výběru mobilních platforem iOS a Android jako cílových platforem pro aplikaci. Dále obsahuje popis tří populárních technologií určených pro vývoj multiplatformních aplikací, jejich srovnání a odůvodnění volby Flutteru jako preferované technologie pro vývoj.

Analýza se soustředí na popis psychologie uživatele při interakci s virtuálními body, průzkum technologií používaných pro filtraci nevhodného jazyka na fórech a zkoumání existujících řešení pro implementaci platebních bran v mobilních aplikacích. Práce rovněž stručně představuje architekturu a koncepci frameworku Flutter a poskytuje úvod do aplikace Eliterro. V závěru analýzy jsou specifikovány funkční a nefunkční požadavky pro jednotlivé moduly.

Návrhová část se věnuje výběru technologií a metodik založených na provedené analýze. Obsahuje také návrh uživatelského rozhraní pro finální aplikaci a konceptuální model.

Realizační část popisuje implementaci modulů, jejich integraci do demonstrační aplikace a podrobně se věnuje jednotlivým realizovaným částem, včetně instalační, uživatelské a programátorské příruček.

Kromě toho diplomová práce zahrnuje automatizované a uživatelské testování, prezentuje výsledky uživatelského testování a případné úpravy modulů na základě získaných zpětných vazeb.

Klíčová slova mobilní aplikace, multiplatformní vývoj, Flutter, Dart, gamifikace, moderování textu, platební brána

Abstract

This diploma thesis focuses on the analysis, design, and implementation of selected sections: ’Virtual Points,’ ’Fan Forum,’ and ’Ticket Purchase and Management,’ using the Flutter framework. The result app serves as a basis for similar modules in the Eliterro application.

The research presented in this thesis provides a rationale for choosing iOS and Android mobile platforms as target platforms for the application. It also contains a description of three popular technologies intended for the development of cross-platform applications, their comparison and the justification for choosing Flutter as the preferred technology for development.

The analysis section focuses on describing the psychology of the user when interacting with virtual points, exploring the technologies used to filter inappropriate language in forums, and examining existing solutions for implementing payment gateways in mobile applications. This section also briefly presents the architecture and concept of the Flutter framework and provides

an introduction to the Eliterro application. At the end of the analysis, the functional and non-functional requirements for the individual modules are specified.

The design part is devoted to the selection of technologies and methodologies based on the performed analysis. It also includes the user interface design for the final application and a conceptual model.

The implementation section describes the implementation of the modules, their integration into the demonstration application, and details the individual implemented parts, including installation, user and programming manuals.

Additionally, the thesis includes automation and user testing, presents the results of user testing, and discusses possible modifications of the modules based on the feedback received.

Keywords mobile application, cross-platform development, Flutter, Dart, gamification, text moderation, payment gateway

Seznam zkratek

UI	User Interface(uživatelské rozhraní)
SDK	Software Development Kit(sada pro vývoj softwaru)
API	Application Programming Interface(aplikační programovací rozhraní)
IDE	Integrated Development Environment(integrované vývojové prostředí)
VM	Virtual Machine(virtuální stroj)
CPU	Central Processing Unit(centrální procesorová jednotka)
NLP	Natural Language Processing(zpracování přirozeného jazyka)
POS	Point-of-Sale(prodejní místo)
NFC	Near Field Communication(komunikace v blízkém poli)
PCI DSS	Payment Card Industry Data Security Standard(Standard zabezpečení dat odvětví platebních karet)
AI	Artificial Intelligence(umělá inteligence)
HTTP	Hypertext Transfer Protocol
APK	Android Application Package(balíček aplikací pro Android)

Úvod

S každým uplynulým rokem přibývá mobilních aplikací. Dle Forbes [1] se celkový počet mobilních aplikací na planetě momentálně odhaduje na přes 8,93 milionu. Tento počet každoročně narůstá. Nepatrná část těchto aplikací je spojena se sportem. Může se jednat o aplikace, jež podporují uživatele při udržení formy či získání sportovního zvyku. Nicméně mohou to být také aplikace pro věrné fanoušky některého sportovního týmu, kteří rádi sledují své oblíbené týmy, komunikují s ostatními fanoušky nebo si čtou články.

Aplikace Eliterro je jedním z těchto mobilních produktů. Jejím cílem je vytvořit univerzální aplikaci, která může být využita pro libovolný sport a jeho svaz. Propojuje nadšence stejného sportu a poskytuje uživatelům funkce jako prohlížení týmů, programů, novinek, komunikaci s ostatními fanoušky, personalizaci profilu, prodej vstupenek a mnoho dalších.

Tato diplomová práce se zaměřuje na tři klíčové části aplikace Eliterro - virtuální body, fórum nebo chat pro fanoušky a správu nákupu vstupenek. V rámci práce budou zahrnuty analýza, návrh, implementace a následné uživatelské testování těchto tří modulů. Vytvořená aplikace bude sloužit jako podklad nebo ”proof of concept”pro navržení těchto tří sekcí v aplikaci Eliterro.

Kromě toho bude aplikace vytvořena v rámci frameworku Flutter, který je jedním z nejpopulárnějších frameworků pro tvorbu multiplatformních aplikací. Tento framework umožňuje sjednotit aplikace pro obě mobilní platformy (iOS a Android) a vytvořit jediný společný kód pro všechny aplikace, což zaručuje rychlejší vývoj a snazší údržbu aplikace.

V rámci pracovní příležitosti jsem měla možnost přispět svým dílem k tvorbě aplikace Eliterro. Jak koncept, tak i myšlenka této aplikace mě velmi zaujaly, a proto jsem se rozhodla podrobněji se seznámit s touto problematikou, naučit se novým technologiím, se kterými jsem předtím neměla zkušenosti - jako například tvorba moderování obsahu, implementace platební brány do aplikace a celková konstrukce aplikace od základu. Další motivací bylo zdokonalení svých dovedností ve frameworku Flutter a využití získaných znalostí k vytvoření studijních materiálů pro ostatní nadšence tohoto frameworku.

Cíle

Hlavním cílem této diplomové práce je návrh a implementace tří klíčových modulů: virtuální body, fórum pro fanoušky a správa a prodej vstupenek. Součástí tohoto cíle je provedení rešerše, zaměřující se na evoluci mobilních aplikací a důvody výběru platforem iOS a Android pro vývoj. Navíc rešerše zahrnuje popis multiplatformního vývoje a technologií vhodných pro tento účel, s důrazem na výběr frameworku Flutter pro realizaci projektu. Další součástí tohoto cíle je analýza konceptu gamifikace, zkoumání jejích psychologických aspektů, posouzení různých možností odměňování (jaké odměny jsou nejvíce motivující pro uživatele a jak často by měly být udělovány) a následné navržení modulu „Virtuální body“, které zahrnuje výběr vhodných metod gamifikace a odměn pro aplikaci Eliterro. Další část práce zahrnuje analýzu existujících metod pro monitorování nevhodného jazyka v různých aplikacích a vývoj modulu „Fórum fanoušků“, včetně průzkumu technologií vhodných pro implementaci v rámci frameworku Flutter. Poslední část se věnuje zkoumání možností integrace online platebních systémů do mobilních aplikací, především z technologického a bezpečnostního hlediska, a průzkumu konkrétních možností pro framework Flutter a navržení modulu „Nákup a správa vstupenek“. Kromě toho budou pro každý modul specifikovány funkční a nefunkční požadavky, které se uplatní v procesu návrhu a implementace.

V praktické části bude implementován prototyp multiplatformní aplikace, využívající vybrané technologie. Výsledek bude sloužit jako podklad pro implementaci zmíněných tří modulů do aplikace Eliterro. Pro mobilní aplikaci budou vytvořeny instalační, uživatelská a programátorská příručky.

Výsledný prototyp aplikace bude podroben automatizovanému a uživatelskému testování, následně bude aplikace upravena na základě výsledků testování a bude formulován závěr.

Kapitola 1
Rešerše

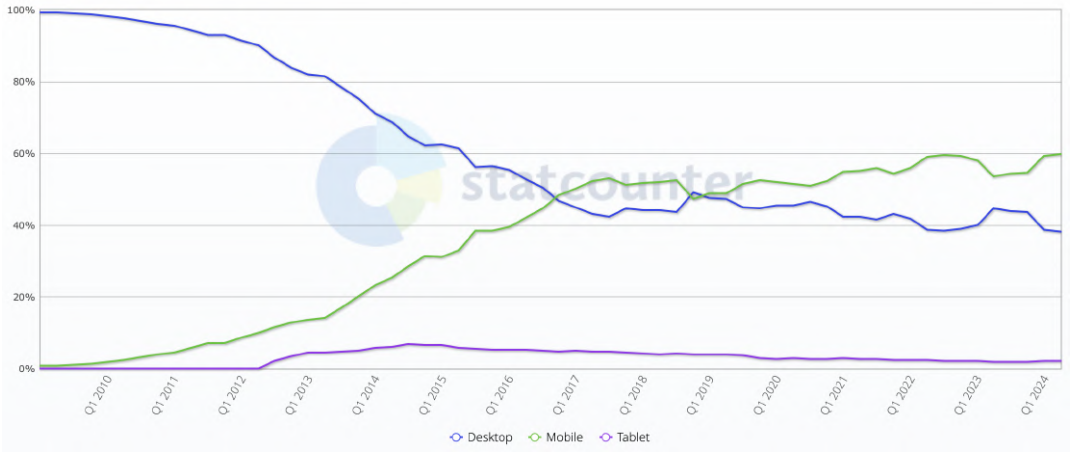
Rešerše této diplomové práce se zabývá odůvodněním výběru mobilních platforem iOS a Android jako primární cílové platformy pro vývoj aplikace. Součástí je přehled vývoje aplikací, od počátečních desktopových řešení až po současné trendy v oblasti mobilních aplikací, a analýza popularity mobilních aplikací. Dále se práce věnuje konceptu multiplatformního vývoje a obsahuje detailní popis tří klíčových a nejčastěji využívaných technologií pro multiplatformní vývoj, včetně výkladu jejich základních principů a souhrnu jejich předností a nedostatků. Dále bude vybrána jedna z těchto technologií, která bude použita pro implementaci praktické části této diplomové práce, a bude zdůvodněno, proč byla konkrétně tato technologie vybrána.

1.1 Evoluce mobilních aplikací

V průběhu posledního desetiletí jsme byli svědky prudkého nárůstu popularity mobilních aplikací, což lze doložit řadou statistik a grafů. Jeden takový graf od StatCounter[2], zobrazený na obrázku 1.1, ilustruje podíly počítačů, mobilních zařízení a tabletů na světovém trhu. Z něj je zřejmý výrazný vzestup mobilních zařízení v poslední dekádě a současný pokles počítačů, což dodatečně potvrzuje rostoucí zájem o mobilní aplikace. Podle údajů z webu Statista[3] celkový počet stažení mobilních aplikací po celém světě od roku 2016 do roku 2023 vzrostl z 140 na 257 milliard a tento trend stále pokračuje. Přesto je klíčové se seznámit s časovou osou evoluce a trendů aplikací, od prvních základních počinů, aby lépe porozumět důvodům obliby pravě mobilních aplikací a předpokládaným směrům vývoje v blízké budoucnosti.

První aplikace vznikaly současně s prvními počítači a s rozvojem nových technologií se objevovaly nové programy pro nové technologie a zařízení. Historie vývoje aplikací a jejich trendů se dosud vyvíjela následovně [4, 5]:

- První programy pro rané počítače byly vytvářeny převážně na zakázku a často přímo uživateli těchto strojů. Běžně se používaly pro matematické výpočty, vědecké simulace a zpracování obchodních informací. S nástupem osobních počítačů se software začal standardizovat a stal se komerčně dostupným, což položilo základ pro výrazný rozvoj vývoje desktopových aplikací v osmdesátých a devadesátých letech dvacátého století. První aplikace, dostupné pouze na stolních počítačích, byly instalovány přímo na pevný disk a primárně sloužily ke zvýšení produktivity (například Microsoft Office), podpoře kreativity (Adobe Photoshop) nebo zábavě (první počítačové hry). [6]
- Na konci devadesátých let se internet stává dostupnějším, což vede k nárůstu webových stránek a aplikací na trhu. Tyto aplikace se lišily od desktopových tím, že k nim lze přistupovat pomocí webových prohlížečů, což zjednodušuje proces instalace a aktualizace, který byl nutný



Obrázek 1.1 Podíl počítačů, mobilních zařízení a tabletů na celosvětovém trhu[2]

u stolních aplikací. To činí webové aplikace populárními v té době. S rozšířením popularity webových stránek vznikají také velké e-mailové aplikace jako Hotmail, vyhledávač Google a e-commerce společnosti jako Amazon a eBay.

- Ačkoliv první smartphone, model Simon od společnosti IBM, byl představen již v roce 1994, nezahrnoval funkce spojené s aplikacemi v moderním smyslu slova. Nabízel pouze několik základních předinstalovaných aplikací, jako byly kalendář, kalkulačka, poznámky a podobně. Opravdový zlom v popularitě mobilních aplikací nastal však až v roce 2007, kdy společnost Apple uvedla na trh první iPhone, a v roce 2008 App Store, platformu pro distribuci a monetizaci mobilních aplikací. Brzy po tomto kroku představuje společnost Google Android Market (nyní Google Play) pro Android zařízení, což zahajuje éru vývoje mobilních aplikací. Rychlé přijetí mobilních zařízení je dáno jejich pohodlím použití a intuitivním dotykovým rozhraním. [7] Do roku 2010 bylo v App Store společnosti Apple k dispozici téměř 300 000 aplikací a počet stažení přesáhl 3 miliardy[8].

- Od roku 2010 až po současnost drží mobilní aplikace svou celosvětovou popularitu. Statistiky[3] z roku 2023 ukazují, že uživatelé po celém světě stáhli 257 miliard aplikací a trend nevykazuje žádné známky zpomalení. Jako důvody této popularity lze uvést implementaci platforem sociálních médií jako Facebook, Instagram a Snapchat do mobilních aplikací, což přesunulo většinu uživatelů k používání mobilních aplikací místo webových a desktopových. Také široký výběr mobilních her přispěl k popularitě, některé se staly kulturními fenomény, jako například Angry Birds nebo Pokemon GO.

- Budoucí vývoj v oblasti mobilních aplikací se zaměřuje na stálé zapojení nejnovějších technologií, jako jsou rozšířená a virtuální realita či funkce založené na umělé inteligenci. Kromě toho dochází k rozšiřování využití mobilních technologií i mimo oblast smartphonů, což zahrnuje širší spektrum chytrých zařízení. Mezi ně patří nositelná elektronika, jako jsou inteligentní hodinky, nebo zařízení pro inteligentní domácnosti, jako jsou například smart televize. Vývoj aplikací pro různorodá chytrá zařízení, stejně jako vznik aplikací pro správu a ovládání všech chytrých zařízení z jednoho místa, se rovněž rýsuje jako významný trend budoucnosti. Ten je poháněn touhou po větším komfortu, personalizovaných zážitcích a vyšší efektivitě. [9, 10]

Popularita mobilních aplikací je dána několika klíčovými faktory [11, 12, 13]:

- Využití vestavěných funkcí chytrých telefonů, jako jsou GPS, fotoaparáty a senzory, rozšiřuje možnosti aplikací. Současně moderní aplikace integrují pokročilé technologie, jako je umělá

inteligence, rozšířená realita a strojové učení, což umožňuje přidání nových a zajímavých funkcí, jako je rozpoznávání obličejů nebo personalizované doporučení.

- Poskytování interaktivnějšího a personalizovanějšího uživatelského rozhraní a zážitku s pomocí intuitivního dotykového ovládání, využitím hardwaru a senzorů, zasíláním upozornění, aktualizací obsahu v reálném čase, přístupností a konzistencí zážitku napříč zařízeními. Tyto aspekty jsou často mimo dosah desktopových a webových aplikací s takovou mírou efektivity a adaptability.
- Používání mobilních aplikací je pohodlné, dostupné a rychlé, což znamená, že poskytují okamžitý přístup ke službám, které lze jednoduše využívat odkudkoliv a kdykoliv, což vyhovuje rychlému životnímu stylu moderních uživatelů.
- Rychlý vývoj v oblasti mobilních technologií umožňuje vybavit moderní smartfony výkonnými procesory, bohatou pamětí, displeji s vysokým rozlišením a pokročilými senzory, což podporuje plynulé spuštění a bezproblémový chod složitějších aplikací.
- Rozvoj širokopásmového připojení a mobilních dat umožňuje, aby internet na mobilních zařízeních byl všudypřítomný, spolehlivý a cenově dostupný, což otevírá dveře pro vývoj rozšířených funkcionalit mobilních aplikací, jako je streamování videa, online hraní her a zasílání rychlých zpráv, jejichž výkon může konkurovat stolním počítačům.
- Platformy jako Apple Store nebo Google Play zjednodušují uživatelům proces vyhledávání, instalace a aktualizace různých aplikací, jelikož všechny aplikace jsou dostupné na jednom místě, což vede k většímu zapojení uživatelů.

Vzhledem k široké oblíbenosti mobilních aplikací je zřetelné, že zacílení na mobilní zařízení jako hlavní platformu pro vývoj aplikací je zásadní k dosažení co největšího počtu uživatelů a k zajištění nejlepší možné uživatelské zkušenosti. Tato strategie rovněž poskytuje vývojářům možnost implementovat nejnovější technologie a inovace navržené specificky pro mobilní prostředí, což může vést k lepší angažovanosti a větší spokojenosti uživatelů. Je důležité poznamenat, že vývoj mobilní aplikace bude probíhat s využitím multiplatformních technologií, což zajišťuje vytváření jednotné základny kódu pro obě hlavní mobilní platformy – iOS a Android. Tato metodika také usnadňuje v budoucnosti přidávání podpory pro další platformy podle potřeby. Flexibilita multiplatformního vývoje může být prospěšná při integraci s již známými desktopovými a webovými rozhraními, a také při adaptaci na nové trendy, jako jsou chytré hodinky nebo televize, pokud vybraná technologie tyto možnosti podporuje.

1.2 Multiplatformní vývoj

Multiplatformní (někdy též označovaný jako meziplatformní) vývoj představuje metodiku tvorby softwarových aplikací, které jsou schopné fungovat a jsou kompatibilní s různými operačními systémy a platformami, přičemž vyžadují minimální, nebo žádné úpravy v základním kódu. V současnosti se tato strategie těší velké popularitě, zejména v oblasti mobilního vývoje. Většina mobilních aplikací totiž cílí hned na dvě hlavní platformy – Android a iOS. Díky multiplatformnímu přístupu je možné vyvíjet jednotný kódový základ pro obě platformy, což je výrazně efektivnější než tradiční nativní vývoj, který vyžaduje vytváření dvou oddělených kódových základen pro každou platformu zvlášť. [14]

Mezi hlavní výhody multiplatformního vývoje patří zejména efektivita, která se projevuje ve zkráceném čase a snížených nákladech na vývoj – stačí vytvořit jedinou kódovou základnu pro více platforem. Tento přístup rovněž usnadňuje údržbu a aktualizace aplikací, protože se opět týká pouze jedné kódové základny, na rozdíl od nativního vývoje.

Na druhou stranu, multiplatformní vývoj přináší i některé nevýhody a výzvy, které se týkají především výkonu (nativní aplikace jsou obvykle rychlejší kvůli absenci dodatečných abstrakčních

vrstev) a uživatelského zážitku. Rozhraní multiplatformních aplikací nemusí vždy dokonale odpovídat nativním prvkům, což může vést k rozporuplným reakcím uživatelů. Navíc ne všechny multiplatformní technologie plně využívají možnosti a API různých platforem, což může snížit kvalitu uživatelského zážitku. [15]

Existuje široká paleta technologií pro multiplatformní vývoj, z nichž každá nabízí specifický přístup k sjednocení kódovacího prostředí. Mezi nejznámější patří React Native, Flutter, Xamarin, Cordova a Ionic. V následujícím textu se podrobněji zaměříme pouze na první tři zmíněné technologie.

1.2.1 React Native

React Native je open-source framework vyvinutý společností Facebook v roce 2015, který umožňuje vytváření multiplatformních aplikací na základě programovacího jazyka JavaScript a knihovny React, určené pro design uživatelských rozhraní. Na rozdíl od standardního Reactu, který je orientován na webové prohlížeče, se React Native primárně zaměřuje na mobilní platformy. Toho dosahuje tak, že napodobuje tradiční mobilní vývoj. Hlavní koncepce React Native spočívá v tom, že komponenty uživatelského rozhraní jsou psány v JavaScriptu, přičemž napsaný kód interaguje s nativní platformou a renderuje komponenty jako widgety dané platformy.

Mezi hlavní přednosti použití technologie React Native pro multiplatformní vývoj patří [16, 17, 18]:

- Funkce „Fast Refresh“, která zvyšuje produktivitu a zrychluje vývojový proces aplikací. Umožňuje okamžitě vidět změny v aplikaci bez nutnosti její kompletní rekompilace. Při změně souboru funkce ”Fast Refresh”aktualizuje pouze ty části aplikace, které byly změněny.
- I když výkon React Native aplikací zcela nedosahuje úrovně výkonu nativních aplikací, je optimalizován pro mobilní prostředí tím, že využívá samostatné vlákno pro uživatelské rozhraní a nativní interakce. Toto zajišťuje, že kód běží asynchronně. React Native také vykresluje uživatelské rozhraní pomocí nativních rozhraní API, což přispívá k lepšímu výkonu.
- Jelikož React Native vyvíjí společnost Facebook, uživatelé mohou počítat s pravidelnými aktualizacemi, vylepšeními a podporou. Díky své popularitě a spojení s Facebookem je podporován širokou komunitou, která neustále přispívá k aktualizacím, opravám chyb a přidávání nových knihoven, nástrojů a pluginů, což usnadňuje vývojářům hledání řešení a získání pomoci při řešení problémů během vývoje.
- Na rozdíl od starších multiplatformních řešení, která poskytovala webové aplikace zabalené do nativního kontejneru, React Native vytváří uživatelské rozhraní pomocí nativních komponent poskytovaných platformami. To znamená, že rozdíl mezi uživatelským rozhraním nativní aplikace a aplikace v React Native je těžko rozpoznatelný.
- Pro vývoj aplikací v React Native se používá JavaScript, jeden z nejoblíbenějších a nejrozšířenějších programovacích jazyků. Vývojáři mohou snadno přejít z webového vývoje na mobilní s použitím React Native a JavaScriptu, aniž by se museli učit nové jazyky a technologie. Navíc, balíček React je jedním z nejběžněji používaných balíčků pro JavaScript, což činí React Native ještě atraktivnější a zajišťuje snadné nalézání vývojářů pro tvorbu aplikací v této technologii.

Co se týče slabých stránek, je možné uvést následující [14, 16, 17]:

- Využití pokročilých a specifických funkcí, které jsou přístupné pouze omezeně, vyžaduje tvorbu nativních modulů. Od vývojáře se pak očekává alespoň základní znalost programovacího jazyka příslušné platformy. V případě Androidu a iOS každá platforma používá svůj vlastní programovací jazyk, což vyžaduje od vývojáře orientaci ve dvou dodatečných jazycích.

- Ačkoliv výkon aplikací v React Native bývá často srovnatelný s nativními aplikacemi, existují situace, kdy výkon React Native výrazně klesá. K takovému poklesu může dojít při použití nativních modulů, které komunikují s JavaScriptem prostřednictvím asynchronních „mostů“. Čím více je takových „mostů“, tím více dochází k poklesu výkonu aplikace. V případě, že je třeba využít mnoho funkcí nedostupných v React Native a je nutné je implementovat pomocí nativního kódu, může být vhodnější zvolit jinou technologii nebo přistoupit k nativnímu vývoji.

- JavaScript je poměrně náchylný k chybám, což může vést k nižší úrovni zabezpečení vyvíjených aplikací. Z toho důvodu není React Native, který JavaScript používá, doporučen pro aplikace vyžadující vyšší úroveň zabezpečení, jako jsou například bankovní nebo finanční aplikace.

- Pro tvorbu složitějších uživatelských rozhraní, která vyžadují komplexní gesta, přechody na obrazovce, animace nebo mnoho interakcí, není React Native ideální volbou. Dotykové subsystémy iOS a Android se od sebe natolik liší, že použití jediného rozhraní pro obě platformy může být náročné a nevhodné.

1.2.2 Xamarin

Xamarin od Microsoftu je open-source soubor nástrojů a knihoven založený na frameworku .NET, určený pro vývoj mobilních aplikací pro různé platformy. Tato technologie je jednou z prvních v oblasti multiplatformního vývoje a umožňuje vývojářům psát kód v jazyce C#, který je populární, univerzální a výkonný, a následně tento kód spouštět na různých platformách, zejména na iOS a Androidu. Xamarin funguje tak, že poskytuje jednotný .NET framework, který spolupracuje s nativními rozhraními API jednotlivých platforem (nativní knihovny jsou integrované do .NET vrstvy). To znamená, že vývojáři píší kód v C# a Xamarin jej automaticky zpracovává, projekt se pak kompiluje na specifické nativní subsystémy a vykresluje uživatelské rozhraní jako nativní prvky na každé z platform. Pro komunikaci s nativními rozhraními API využívá Xamarin framework Mono. [19]

Aplikace v Xamarinu lze vytvářet dvěma způsoby. První možnost zahrnuje použití Xamarin.iOS a Xamarin.Android pro vytvoření aplikací specificky pro iOS a Android s možností sdílení kódu. Druhá možnost spočívá ve využití Xamarin.Forms, což umožňuje navrhovat uživatelská rozhraní v XAML (Extensible Application Markup Language) a sdílet až 100 procent kódu mezi platformami. Tento nástroj transformuje komponenty uživatelského rozhraní na prvky specifické pro daný operační systém. Toto řešení je ideální pro vývoj aplikací s minimálním počtem funkcí specifických pro jednotlivé platformy. [20]

Mezi hlavní výhody použití technologie Xamarin patří [14, 19, 20]:

- Vysoký výkon aplikací vytvořených pomocí Xamarinu se přibližuje výkonu nativních aplikací. Tento fakt je dán zejména dvěma optimalizačními prvky: použitím Xamarin.Forms a Xamarin.Essentials, které umožňují přímý přístup k API specifickým pro danou platformu, a také předběžnou kompilací, jež převádí kód Xamarinu přímo do nativního ARM assembly, čímž dochází k nárůstu výkonu. Navíc vývojové prostředí nabízí různé nástroje pro monitoring a testování, které zlepšují přehlednost a umožňují identifikovat slabá místa vývoje.

- Díky přístupu Xamarinu téměř ke všem nativním API rozhraním umožňuje tato technologie využívat hardwarové a softwarové funkce specifické pro každou platformu a také integrovat aplikace s nativními knihovnami. Příkladem může být přístup k ARKit na iOS nebo funkce Multi-Window na Android, což je něco, co mnoho multiplatformních technologií nenabízí. Tento přístup zaručuje, že aplikace vytvořené pomocí Xamarinu nabízí zcela nativní vzhled a chování, což výrazně zlepšuje uživatelskou zkušenost.

- Jazyk C# je považován za jeden z nejpopulárnějších programovacích jazyků, což zahrnuje i velkou komunitu vývojářů, kteří přispívají svými znalostmi k vylepšení Xamarinu. Micro-

soft navíc poskytuje rozsáhlou dokumentaci, vzorové programy a průběžné aktualizace pro Xamarin.

- Díky Xamarin.Forms lze vytvářet různé verze aplikací pro široké spektrum platforem, včetně iOS, Android, Windows Phone a Mac, ale také pro specializovanější zařízení jako jsou televize Samsung nebo tvOS a watchOS od společnosti Apple.

Jako každá technologie má i Xamarin své slabé stránky, které jsou uvedeny níže [19, 21, 20]:

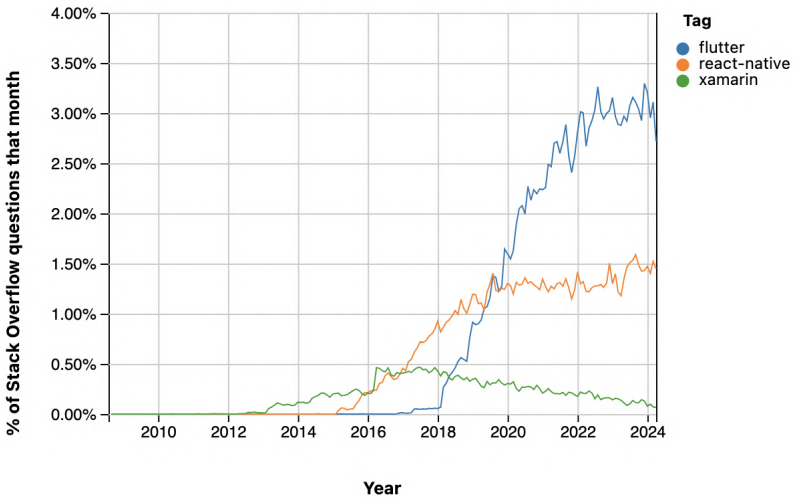
- Aplikace vyvinuté v Xamarinu mají značně větší velikost ve srovnání s nativními aplikacemi, což je dáno přítomností komponent frameworku Mono a přidružených knihoven, jež musí být součástí každé aplikace v Xamarinu. Jak lze vidět na obrázku 1.2, nejmenší možná základní aplikace "Hello World"zaujímá velkou část svého prostoru právě kvůli těmto podpůrným komponentám a knihovnám. To může způsobit, že uživatelé mohou být odrazeni od jejich používání kvůli delší době stahování a vyšší spotřebě dat.
- Proces vývoje v Xamarinu není tak uživatelsky přívětivý jako u jiných zkoumaných technologií, které nabízejí funkci okamžitého znovunačtení (hot reload), což může vést k pomalejšímu procesu vývoje a delšímu ladění kódu. Kromě toho může sestavování aplikací Xamarin trvat déle, neboť se aplikace kompilují pro všechny platformy najednou.
- Ačkoliv Xamarin umožňuje vytvořit uživatelské rozhraní zcela identické s nativním, realizace složitějších uživatelských rozhraní může vyžadovat více úsilí a času, zvláště při implementaci pomocí Xamarin.Forms, jelikož tato není tak přímočará jako v případě nativních nástrojů. To činí Xamarin méně vhodným pro aplikace s náročnou grafikou a složitým designem.
- Přestože je Xamarin open-source, pro větší podniky (pro jednotlivce a menší společnosti je zdarma) je nutné zakoupit licenci na Visual Studio, což je vývojové prostředí pro Xamarin (použití jiného IDE je možné, ale pro plné využití funkcí Xamarinu je doporučeno Visual Studio). To může být poměrně nákladné, ceny začínají na 500 dolarů za jednoho uživatele na rok, a náklady na licenci mohou rychle narůstat.

Obrázek 1.2 Velikost základní aplikace Xamarin [22]

1.2.3 Flutter

Flutter je open-source sada vývojových nástrojů od společnosti Google, určená pro tvorbu multiplatformních aplikací pro mobilní zařízení, webové platformy a počítače z jediné kódové základny. Klíčovým principem Flutteru je využití takzvaných widgetů, z nichž je sestaveno uživatelské rozhraní aplikací vyvíjených v tomto frameworku. Pro vývoj aplikací ve Flutteru se používá programovací jazyk Dart, který má syntaxi podobnou jazykům z rodiny C, což usnadňuje jeho osvojení zejména vývojářům obeznámeným s objektově orientovanými koncepty. Flutter umožňuje vytvářet vysoce efektivní aplikace s kvalitním uživatelským rozhraním a vynikajícím výkonem, což je dosaženo kompilací aplikace do nativního kódu. Od svého založení si Flutter vybudoval rostoucí popularitu mezi vývojáři, což dokládá graf na obrázku 1.3. Z něj je patrné, že Flutter téměř dvakrát převyšuje React Native a výrazně předčí Xamarin, který s každým rokem ztrácí na popularitě a v porovnání s ostatními zkoumanými technologiemi se nachází na posledním místě.

Důvodem velké popularity Flutteru jsou jeho silné stránky, mezi které patří [24, 25, 26]:



Obrázek 1.3 Porovnání počtů otázek ohledně Flutter, Xamarin a React Native na Stack Overflow [23]

- Jazyk Dart, používaný pro vývoj Flutter aplikací, je objektově orientovaný programovací jazyk a velmi se podobá populárním jazykům jako Java, Python, C++. Pro většinu vývojářů je tento jazyk snadno pochopitelný a naučitelný, protože strukturálně i syntakticky připomíná tyto jazyky. Kromě toho má Flutter vynikající dokumentaci, která napomáhá lepšímu porozumění. A navíc, Flutter má aktivní komunitou, která ochotně odpovídá na veškeré otázky.
- Jedním z klíčových a významných přínosů Flutteru je funkce živého překompilování, známá jako hot reload. Tento nástroj umožňuje zobrazovat změny v kódu bez nutnosti kompletní rekompilace, protože Flutter při živém překompilování vkládá změny přímo do běžícího prostředí Dart VM, aniž by ovlivnil aktuální stav. Díky hot reloadu lze okamžitě vidět účinky změn v reálném čase, což významně zkracuje dobu vývoje a urychluje implementaci změn a oprav chyb v kódu. Dalším užitečným nástrojem je inspektor widgetů, který usnadňuje vizualizaci a řešení problémů s rozvržením uživatelského rozhraní.
- Dalším pozitivním aspektem je vysoký výkon, který téměř dosahuje úrovně nativních aplikací. Flutter aplikace dosahují vyšší rychlosti díky přímé kompilaci do strojového kódu, který je optimalizován pro hostitelské zařízení, a vyvarují se tak použití jakýchkoli prostředníků(mostů), jež by mohly zpomalit výkon. Tuto skutečnost potvrzují různé výzkumy. Například podle studie provedené společností Thoughbot [27], Flutter aplikace téměř dosahují využití CPU nativní aplikace a je dokonce uvedeno, že Flutter může přinést skutečné zlepšení výkonu oproti React Native. Podle výzkumu společnosti InVerita [28], která porovnávala výkon Flutter aplikací, React Native a nativních aplikací, je výkon Flutteru blízký tomu nativnímu.
- Základní stavební kameny Flutteru, widgety, představují obsáhlou sbírku rozmanitých prvků uživatelského rozhraní. Ty zahrnují widgety pro různé účely, včetně některých nativních widgetů pro platformy Android a iOS. Kromě toho vývojářům umožňují vytvářet vlastní widgety. Tato variabilita umožňuje vytvářet vysoce přizpůsobitelná a vizuálně atraktivní uživatelská rozhraní, která se vyrovnají rozhraním nativních aplikací.

Samozřejmě, že Flutter má i slabé stránky, ty jsou uvedeny níže [24, 25, 26, 29]:

- Flutter může být méně vhodný pro projekty mobilních aplikací, které vyžadují specifickou optimalizaci pro jeden operační systém, zvláště pokud jde o iOS platformu. Vývoj jedné nativní aplikace pro vybranou platformu může být v takových případech efektivnější a rychlejší.

Rozdíly ve funkcionalitě mezi operačními systémy, jako je lepší podpora některých funkcí na Androidu oproti iOS, mohou ovlivnit rozhodování o platformě. Toto může souviset s tím, že Google, který Flutter vyvíjí, má blízký vztah k Androidu, což může vést k rychlejšímu přijímání nových funkcí a aktualizací Androidu ve Flutteru než u iOS.

- I přes to, že Flutter umožňuje vývoj pro šest různých platforem, stále má své omezení. Například není možné vytvářet aplikace pro watchOS, tvOS nebo Apple CarPlay pomocí Flutter. S ohledem na očekávaný nárůst prodeje chytrých hodinek na čtvrt miliardy kusů během následujících pěti let, je to významné omezení. Flutter se snaží řešit tuto situaci nabídnutím alternativních řešení, jako je možnost integrovat nativní rozšíření pro Apple Watch do Flutter aplikací.
- Při zvažování vhodnosti použití Flutteru je důležité mít na paměti, že pro problémy, které silně závisejí na specifických funkcích dané platformy nebo vyžadují hlubokou integraci s externími zařízeními, nemusí být Flutter nejoptimálnější volbou. Čím větší potřeba existuje po nativních funkcích, které nejsou součástí Flutteru, tím složitější může být zachování jednotného kódu pro různé platformy a tím méně vhodným se Flutter může byt.
- Velikost Flutter aplikací obvykle překračuje velikost nativních aplikací a to může být pro některé uživatele problematické, zejména pro ty, kteří mají omezený úložný prostor na svých zařízeních. Minimální velikost Flutter aplikace často přesahuje 4 MB, což je značně větší než u nativních Java (539 KB) a Kotlin (550 KB) aplikací. Avšak tuto velikost lze optimalizačními technikami snížit. Flutter má schopnost vytvářet aplikace, které mohou za běhu stahovat další kód a prostředky Dart. To umožňuje instalovat aplikace s menší velikostí a stahovat funkce a prostředky pouze tehdy, když jsou skutečně potřeba.

1.2.4 Shrnutí

V závěru lze konstatovat, že tři analyzované technologie pro multiplatformní vývoj sdílejí řadu předností i nedostatků. Hlavním společným přínosem je nepochybně efektivita vývoje díky jediné kódové základně použitelné napříč různými platformami. Dále lze zmínit relativně dobrý výkon ve srovnání s výkonem nativních aplikací, stejně jako rozmanitost metod pro tvorbu uživatelského rozhraní, které umožňují maximální soulad s rozhraním nativních aplikací. Pokud jde o společné slabiny těchto technologií, patří mezi ně větší velikost výsledných aplikací a složitost při implementaci komplexnějších a pokročilejších funkcí nebo designových prvků nativních platforem.

Pro shrnutí rešerše týkající se různých technologií pro multiplatformní vývoj lze vytvořit následující tabulku 1.1, která zahrnuje všechny důležité aspekty a rozdíly těchto tří zmíněných technologií.

1.2.5 Vybraná technologie

Hlavním kritériem pro výběr použité technologie byla zkušenost autora této diplomové práce s vývojem aplikací v prostředí Flutter a s programovacím jazykem Dart, stejně jako jeho zájem o rozvoj a zdokonalení těchto dovedností. Dále byl tento výběr ovlivněn skutečností, že aplikace Eliterro, do které budou v budoucnu začleněny vyvíjené moduly, je také vytvářena pomocí Flutteru, což zjednodušuje integraci.

Flutter byl shledán jako nejvhodnější nejen z důvodu autorových předchozích zkušeností, ale také kvůli dalším faktorům, jako je vysoký výkon srovnatelný s nativními aplikacemi, příjemnější a rychlejší proces vývoje díky funkci hot reload. Tato technologie navíc disponuje širokou podporou a komunitou, která ochotně poskytuje odpovědi na různé otázky a nabízí množství balíčků pro urychlení a zlepšení vývoje. Oficiální dokumentace Flutteru je rovněž velmi dobře zpracovaná a obsahuje všechny nezbytné materiály pro tvorbu aplikací v tomto prostředí.

	React Native	**Xamarin**	**Flutter**
Jazyk a ekosystém	JavaScript a framework React	C# a ekosystém .NET	Dart ve stylu C
Výkon	JavaScriptový most pro komunikaci s nativními komponentami může vést k nižšímu výkonu	Kompiluje se do nativního kódu, čímž dosahuje výkonu srovnatelného s nativními aplikacemi	Díky kompilaci do nativního kódu a schopnosti kompilátoru Dart přímo kompilovat do nativních knihoven ARM nebo x86 je zajištěn vysoký výkon
Velikost aplikaci	Je větší než nativní aplikace kvůli obsahu JavaScriptového enginu a potřebným knihovnám pro běh aplikace, přičemž počáteční velikost jednoduché aplikace "Hello world"činí přibližně 7 MB pro Android a 15 MB pro iOS	Produkuje větší binární soubory a zahrnuje Mono (multiplatformní .NET framework) spolu s .NET Base Class Library, s počáteční velikostí základní aplikace od 15 MB pro Android a více pro iOS	Aplikace obsahuje prostředí Dart a engine Flutter, což má za následek větší velikost aplikace; pro Android je to 4-5 MB a mírně více pro iOS pro základní aplikaci
Proces vývoje	Dynamický vývojový proces je urychlen díky funkci fast refresh	Vývojové prostředí poskytované Visual Studio nabízí širokou škálu různých pomocných nástrojů pro vývoj	Zahrnuje funkci hot reload, která urychluje proces vývoje
Možnosti a konzistence uživatelského rozhraní	Poskytuje komponenty uživatelského rozhraní přímo prostřednictvím nativních komponent	Pro tvorbu uživatelského rozhraní přímo využívá nativní komponenty	Nabízí widgety, které zahrnují jak unikátní widgety, tak i widgety odpovídající prvkům uživatelského rozhraní nativních platforem
Komunita a podpora	Má rozsáhlou komunitu a poskytuje velké množství doplňujících knihoven a frameworků, které rozšiřují možnosti React Native	Nabízí širokou podporu, zvláště mezi vývojáři v ekosystému Microsoft, který zajišťuje dobrou dokumentaci a podporu	Rychle rostoucí komunita, podporovaná Googlem, nabízí rychle se rozvíjející doplňující balíčky a pluginy pro zjednodušení a zlepšení vývoje

Tabulka 1.1 Shrnutí rozdílů React Native, Xamarin a Flutter

Kapitola 2
Analýza

Tato část práce je především zaměřena na analýzu a průzkum problematiky související s každým modulem. Pro virtuální body je to psychologie uživatele a různé způsoby získaní virtuálních bodu v mobilních aplikaci, pro sekci fórum - existující způsoby hledaní a filtrování vulgarizmu, pro sekci vstupenky - analýza možností pro implementace platební brány a zabezpečení bezpečnosti. Navíc tato část zahrnuje rychle popis Flutter SDK a aplikace Eliterro.

2.1 Flutter SDK

Flutter je open-source software development kit (SDK), který je vyvinuty a podporovaný společnosti Google. Tento framework umožňuje vytvářet uživatelské rozhraní aplikace pro více platforem s jediným kódem. V současné době podporuje vývoj UI na šesti platformách takových jako iOS, Android, web, Windows, MacOS a Linux; i když původně byl zaměřen hlavně na mobilní aplikace. [24]

2.1.1 Architektura Flutteru

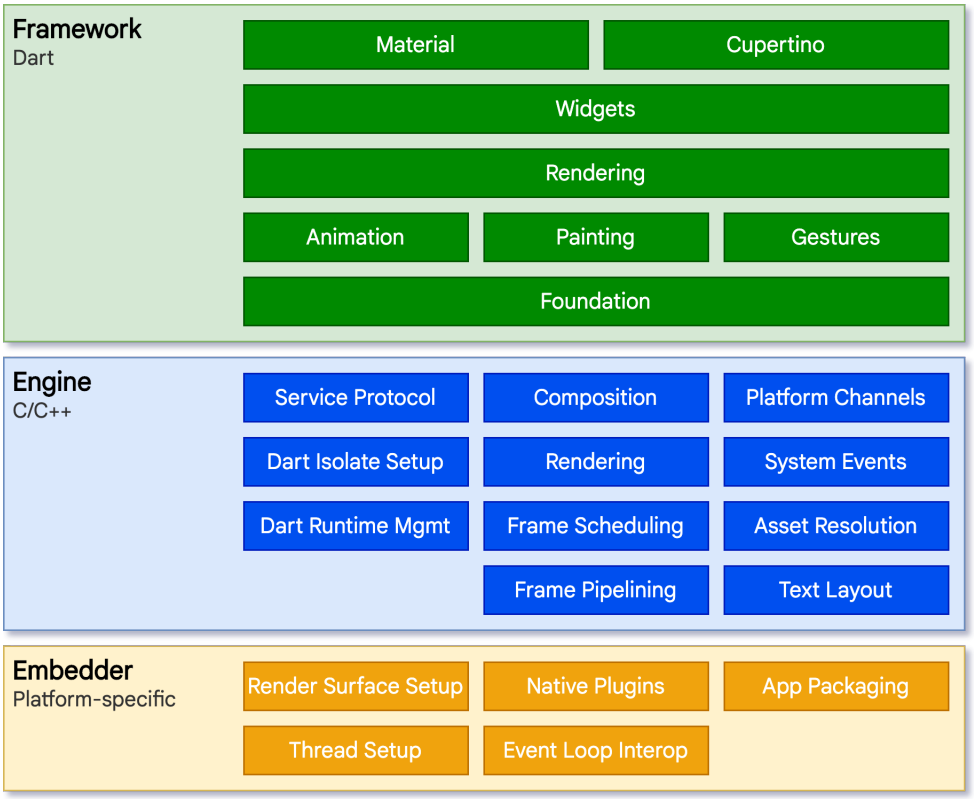
Architektura Flutteru je vrstveným systémem, který se skládá z následujících komponent: **embedder, engine a framework**. Na obrázku 2.1 jsou znázorněné tyto komponenty a jejich hlavni součásti a funkci. [30]

Hlavním účelem **embedderu** je spuštění aplikace na libovolné podporované platformě, funguje jako ”most”mezi nativním strojovým kódem, platformou a enginem Flutter a zajišťuje kompatibilitu a správné spuštění aplikace na různých operačních systémech. Stejně jako nativně aplikace, které jsou zabaleny pod konkretní operační systém, embedder Flutteru pro konkrétní platformu koordinuje s operačním systémem pro přístup do služeb jako jsou zobrazování věcí na obrazovce, interakce s uživateli, dostupnost atd. Jako programovací jazyk embedder používá jazyk specifický pro platformu (například Java a C++ pro Android nebo Objective-C pro iOS a MacOS platformy).

V ”srdce”Flutteru leží **engine**. Je to podobné jako tým v zákulisí divadla, který řídí veškerou základní práci pro každou aplikaci Flutter [31]. Takže odpovídá za takové důležité věcí jako rastrování scén (kdykoli je potřeba namalovat nový snímek), nízkoúrovňovou implementaci základního API Flutter(jako například grafika, layout textu, souborů a síťových vstupu/výstupu, architektury pluginů a také běh Dartu a sada nástrojů pro jeho kompilaci). Engine Flutteru je primárně napsán v programovacím jazyku C++.

A poslední, ale neméně důležitou částí Flutter SDK je **framework**. Je založeny na programovacím jazyce Dart a framework je tou částí architektury, ze kterou komunikuji vývojáře Flutter

aplikace, protože do teto častí patří kolekci nástrojů, knihoven a funkcí, které jsou používané pro implementace aplikace. Například to zahrnuje věci jako **widgety**, což jsou předpřipravené prvky, které se používají k sestavení aplikace(skládá se strom widgetů aplikace do scény), nebo také tady patři knihovny Material a Cupertino, které umožňuji využití UI komponenty ve stylu Google nebo Apple při vytvoření aplikace. Takže pravé tato vrstva zaručuje zjednodušeny proces vytváření uživatelského rozhraní, které umí dynamicky reagovat na různé události(jako detekce gest, zadávání vstupu) a udržovat jednotný vzhled napříč různými platformami. Poskytuje rozhraní API vyšší úrovně pro vytváření vysoce kvalitních aplikací (například widgety, testování přístupů, detekce gest, usnadnění, zadávání textu).

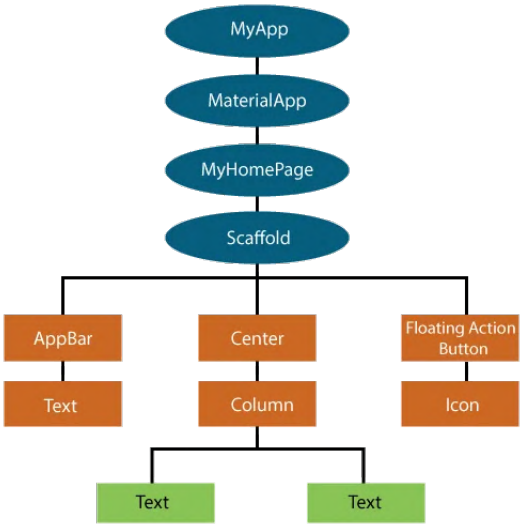


Obrázek 2.1 Architektura Flutteru [30]

2.1.2 Flutter widgety

Jak již bylo zmíněno v předchozí částí, **widgety** jsou základní stavební prvky pro tvorbu uživatelského rozhraní Flutter aplikace. Widgety se skládají do stromu a tvoří tak hierarchii, která se pak skládá do scény. Flutter nabízí širokou radu hotových widgetů, ale umožňuje je také přizpůsobit nebo vytvořit vlastní. Na obrázku 2.2 jde vidět příklad jednoduchého widget stromu, ktery pouziva predpripravene Flutter widgety. Aplikace obnoví uživatelské rozhraní, když dojde k nějaké události (napriklad akce uživatele) tím, že zadá pokyn frameworku na aktualizace jedeného widgeta ve struktuře za novy. Následně framework porovná nové a předchozí widgety a provede efektivní aktualizaci uživatelského rozhraní. [30]

Takže všechno co vidíme na obrazovce jsou widgety: od nejmenších tlačítek a textu do složitých animace - to všechno je různorodé widgety, vnořené jedna do druhé. Widgety ve Flutteru můžeme rozdělit podle dvou vlastnosti: podle jejich role v aplikace a podle jejich schopností



Obrázek 2.2 Příklad widget stromu [32]

řídit stav.

Podle oficiálního widget katalogu [33] lze nabízené Flutter widgety rozdělit do následujících 14 kategorií v závislosti na jejich roli:

- **Widget pro dostupnost** - widgety, které pomáhají implementovat uživatelské přívětivější aplikace, například takové widgety hraji klíčovou roli při vytváření aplikací pro uživatele s postižením.
- **Widgety pro animaci a pohyb** - widgety, které odpovídají za různorodé animace a různé dynamické prvky.
- **Widgety assetů, obrázků a ikon** - widgety pro správu vizuálních prostředků.
- **Async widgety** - widgety, které umožňují provedeni asynchronních operaci.
- **Základní widgety** - nezbytné widgety pro vývoj jakékoliv aplikace Flutter(například *Row, Column, ElevatedButton*).
- **Cupertino widgety** - widgety ve stylu iOS, vetšinou se používají pro tvorbu aplikace pro iOS zařízení.
- **Widgety pro vstup** - widgety pro zpracování uživatelského vstupu(například vyplňování textového pole)
- **Widgety pro modely interakce** - widgety pro zpracování dotykových událostí a navigaci mezi různými zobrazeními.
- **Widgety layoutu** - pomocné widgety pro uspořádání ostatních widget(například zarovnání nebo vycentrování widgetu).
- **Widgety Material komponent** - widgety, které jsou navržené podle pokyny společnosti Google pro návrh materiálů(jako materiály je tady myšleno různé prvky UI - tlačítka, vstupní poli, dialogy atd).
- **Widgety pro malování a efekty** - widgety, které zaručují vizuální efekty pro jiné widgety při zachovávání jejich velikost nebo polohy.

- **Widgety pro rolování** - pomocné widgety, které poskytuji funkci rolování(scrolling) nebo posunu widgetum, které tuto funkci neposkytuji v základním nastaveni.
- **Stylingové widgety** - widgety, které spravuji barvenu tému a typografický styl aplikace, nebo také přidávají mezery ostatním widgetum.
- **Textové widgety** - widgety, které dopovídají za zobrazeni a styl textu.

Dalším způsobem, jak můžeme klasifikovat widgety, je podle jejich schopnosti řídit stav: existují **stateful** a **stateless** widgety.

Stateless widgety jsou bezstavové widgety, jejichž obsah a stav nelze aktualizovat a menit po vytvoření. Bezstavové widgety se obvykle používají pro statické elementy jako například obrázek, ikonka, tlačítko, text apod.

Statefull widgety jsou naopak proměnlivé widgety, které mají stav a tento stav lze měnit v průběhu času a podle potřeby. Používají se pro komponenty uživatelského rozhraní, které vyžadují nějakou dynamickou aktualizace nebo zpracovávají uživatelské interakce.

2.2 Aplikace Eliterro

Aplikace Elittero, vytvořená společností DSparx, je sportovní platformou zaměřenou především na sportovní svazy a jejich fanoušky. Funguje jako ”konstruktor”, kde každý sportovní svaz může přizpůsobit aplikaci podle svého vlastního stylu, loga a barev. Dále mohou nastavit funkce, které aplikace nabízí, a definovat, jak s aplikací pracovat. Obsahuje širokou škálu základních modulů, jako jsou detaily zápasů, hráčů a týmů, ukázky statistik a výsledků, uživatelské profily (pro fanoušky), zpravodajství atd. Navíc umožňuje nastavení atraktivních prémiových modulů, jako jsou speciální notifikace, virtuální body, online streamování, minihry, chatování během zápasů/závodů a další. Na obrázku 2.3 jsou ukázky uživatelského rozhraní aplikace. [34]

V současné době se aplikace nachází ve fázi vývoje, ale většina základních modulů, stejně jako backendová část, je již implementována, a plánuje se vydání první verze ke konci jara.

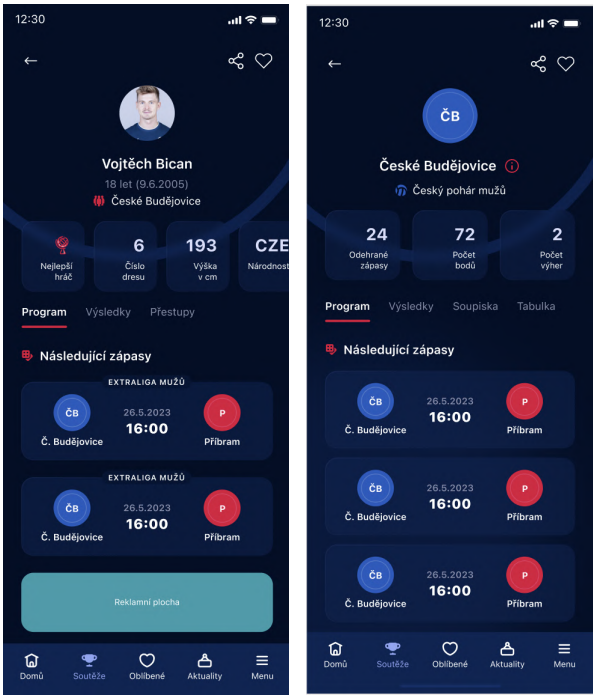
Tato diplomová práce se zaměřuje na analýzu návrhu, technická řešení a implementaci několika modulů: virtuální body (analýza získávání a využívání bodů z psychologického hlediska, implementace sekce virtuálních bodů uživatele podle návrhu a designu), nákup vstupenek (analýza existujících řešení pro propojení platební brány s Flutter aplikací a následná ukázka implementace s použitím vybraných technologií) a fórum/chat pro fanoušky (analýza možných řešení pro filtrování vulgarismů a použití vybrané technologie pro ukázkové řešení).

2.3 Virtuální body

V této části bude objasněn pojem gamifikace a uvedeny příklady jejího využití v různých aplikacích. Dále bude pojednáno o různých metodách gamifikace a jejich spojitost s psychologií uživatele. Následně se zaměříme přímo na odměny/ virtuální body a jejich psychologii (například jak udržet uživatele v aplikaci, jak je motivovat ke sbírání bodů a jak často by měli uživatelé získávat body). Tyto znalosti budou použité při návrhu sekci virtuálních bodů pro aplikace Eliterro.

2.3.1 Gamifikace

S každým rokem stoupá uznání gamifikačních systémů, které si kládají za cíl podnítit inovace, produktivitu a zapojení uživatelů. Tento trend je ovlivněn rostoucí popularitou chytrých telefonů, rozmanitostí trhu mobilních aplikací a silnou konkurencí na tomto poli. Navíc, v dnešní době se lidé, zejména nové generace - mileniálové a generace Z, rychle unavují a proto je neustále zaměstnává pouze něco vzrušujícího a náročného. Předpokládá se, že do roku 2029 dosáhne trh



Obrázek 2.3 Ukázka uživatelského rozhraní aplikace Eliterro [34]

gamifikace hodnoty 48,72 miliard USD. Během předpovídaného období (2024–2029) se očekává růst o 25,85 procenta! [35] Ale co vlastně znamená gamifikace?

Gamifikace je proces začleňování herních prvků a mechanismů do neherních kontextů, často aplikací, s cílem motivovat a zapojit uživatele a zajistit jejich pravidelnou návratnost do dané aplikace. Hlavním účelem gamifikace je vytvoření pohlcujícího zážitku, který udrží uživatele angažovaného a motivovaného při plnění úkolů, kde by jinak mohla chybět potřebná motivace. Tento přístup může být úspěšně využit například při učení nového jazyka nebo podporování pravidelného cvičení či běhání. Gamifikace využívá přirozenou lidskou touhu po úspěchu, sociální interakci a dosažení mistrovství, a proto mezi její prvky patří soutěže, odměny (jako trofeje, kupóny, body, mince a žebříčky) a výzvy, které uživatelům zajišťují trvalou motivaci. [36, 37]

2.3.2 Příklady gamifikace

Někteří čtenáři by mohli mít dojem, že gamifikace se omezuje na otravné vyskakovací reklamy s cenami nebo dokonce na plnohodnotné videohry, ale v současné době vývojáři aplikací implementují sofistikovanou a kvalitní gamifikaci do různých typů aplikací, včetně eCommerce, zdraví a fitness a dalších odvětví. Proto je důležité uvést příklady použití gamifikace ve světově známých aplikacích z různých oborů.

Je překvapivé, že gamifikace lze nalézt i v reálném, každodenním životě. Zde jsou uvedeny některé příklady [38]:

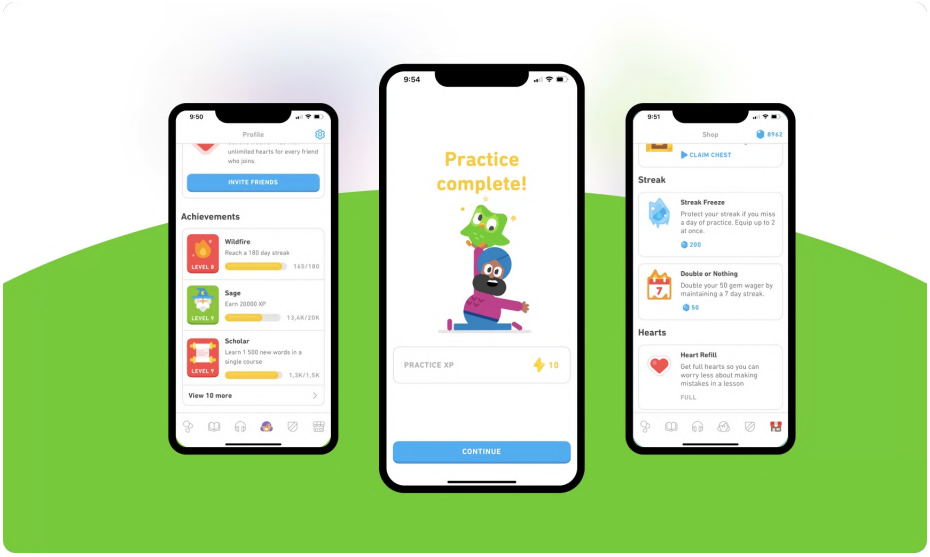
- Učitelé mohou například odměňovat své studenty nálepkami za dobře odvedenou práci.
- Některé obchody mohou svým zákazníkům nabídnout určitý počet nálepek v závislosti na utracené částce, takže zákazníci mohou později tyto nálepky vyměnit za propagovaný produkt.

- Kavárny nabízejí zákaznické kartičky, kde zákazníci sbírají razítka a nakonec mohou získat něco z nabídky zdarma.

Tyto triky motivují studenty k plnění úkolů a zákazníky k nákupu v daném obchodě nebo kavárně. Takto funguje gamifikace v reálném životě.

Pokud jde o využití gamifikace v mobilních aplikacích, je možné ji aplikovat téměř v jakémkoliv oboru, avšak nejpopulárnějšími jsou [38, 39]:

- **Vzdělávací aplikace a e-learning**. Většina uživatelů spojuje učení s nudou a rutinou, proto je klíčové přidat do vzdělávacích aplikací prvky gamifikace či zábavy. Technologie gamifikace zlepšuje angažovanost uživatelů v učebním procesu (učení prostřednictvím her stimuluje stejnou neurologickou aktivitu jako hraní videoher, čímž motivuje uživatele k intenzivnějšímu studiu), motivuje studenty, zlepšuje uchování znalostí a přispívá k větší dostupnosti vzdělání.[40] Jedním z nejznámějších příkladů e-learningové aplikace s prvky gamifikace je Duolingo - oblíbená platforma pro výuku jazyků s miliony uživatelů po celém světě. Výrazným gamifikačním prvkem Duolinga je jeho hlavní postava, sova Duo. Tento maskot provádí uživatele aplikací, což usnadňuje navigaci a dělá ji zábavnější. Dalšími používanými prvky gamifikace jsou sbírání bodů, získávání odznaků a odměn, virtuální měna - lingoty, personalizace, žebříček nejlepších, sociální interakce a notifikace. Na obrázku 2.4 je zobrazeno uživatelské rozhraní aplikace Duolingo spolu s některými příklady gamifikačních prvků. Existuje však desítky dalších vzdělávacích aplikací s prvky gamifikace, jako například Codecademy Go, Khan Academy, SoloLearn, atd.



Obrázek 2.4 Uživatelské rozhraní aplikace Duolingo [40]

- **Aplikace pro zvýšení produktivity**. Tyto aplikace obohacují každodenní rutinu a umožňují efektivnější život. Využívají gamifikaci k podpoře uživatelů při splňování důležitých úkolů, které často odkládáme. Tyto upomínky jsou navrženy tak, aby nebyly obtěžující, ale spíše motivující k dokončení úkolu. Mezi další oblíbené prvky gamifikace v těchto aplikacích patří přizpůsobitelná uživatelská rozhraní, animace, bodové systémy odměn, různé žebříčky a sociální interakce. Mezi příklady gamifikovaných produktivitních aplikací patří Todoist, Habitica nebo MagicTask.
- **Aplikace pro zdravý životní styl**. Tyto aplikace zahrnují různé oblasti zdravého životního stylu, jako je fitness, výživa, léčba, duševní zdraví, rehabilitace a další. Příklady takových

aplikací zahrnují Samsung Health, Google Fit, FitBit a MySugr. Tyto aplikace mohou obsahovat následující prvky gamifikace: odznaky a odměny (například za dosažení určitého počtu kroků), sociální interakce (sdílení výsledků na sociálních sítích nebo soutěže s přáteli), různé výzvy pro dosažení lepších výsledků nebo virtuální postavy, které pomáhají uživatelům dosáhnout svých cílů.

2.3.3 Způsoby gamifikace z psychologického hlediska

Některé metody gamifikace a příklady gamifikačních prvků již byly uvedeny v předchozí části. Nyní je důležité se na to podívat z jiného úhlu - z perspektivy psychologie uživatelského chování. Jádrem gamifikace jsou základní lidské potřeby pro motivaci, odměny, vyjádření sebe sama, úspěchy, soutěživost a postavení ve společenství. Ale především je postavena na motivaci uživatele. Motivace je vnitřní hnací síla, která pohání člověka vpřed, ovlivňuje naše činy, rozhodnutí a nakonec i náš úspěch; je to síla, která je základem našich úsilí a ambicí. V této části se zaměříme především na Octalysis Framework od guru gamifikace - Yu-kai Chou, který rozděluje gamifikační prvky na 8 hlavních hnacích sil (core drives) z pohledu psychologie uživatele. [41]

Behaviorální psychologie je odvětvím psychologie, které zkoumá lidské chování a motivaci, jež ho ovlivňují. Soustředí se na pochopení důvodů, proč lidé činí různá rozhodnutí, a způsobů, jak lze jejich chování ovlivnit. Tato znalost lidského chování může být pro designéry aplikací neocenitelná při lepším zvážení a navrhování uživatelského rozhraní. [37]

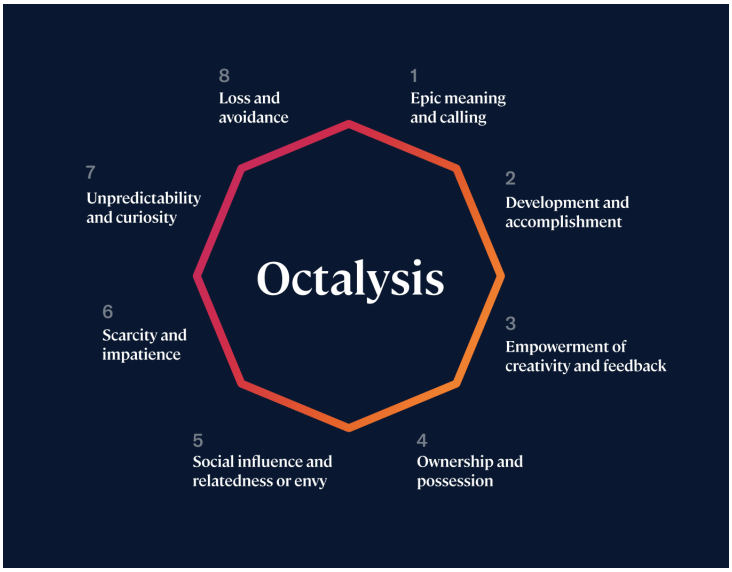
2.3.3.1 Octalysis Framework

Octalysis Framework, vytvořený odborníkem na gamifikaci, Yu-kai Chou, který se po dobu 17 let věnoval výzkumu a studiu, je nástrojem, který se stal nezbytnou literaturou ve výuce gamifikace po celém světě. Tento framework, přeložený do 16 jazyků, podrobně zkoumá 8 různých sil/-prvků(core drive), jež ovlivňují lidskou motivaci, a ukazuje, jak je možné je využít k vytvoření poutavých a odměňujících zážitků, jež posilují pozitivní uživatelské zkušenosti. Zjednodušený Octalysis Framework je znázorněn na obrázku 2.5 a je grafickým zobrazením osmiúhelníkového tvaru, kde každý úhel reprezentuje jednu ze sil. Je zajímavé, že když rozdělíme tento osmiúhelník vertikálně na dvě poloviny, tak můžeme říci, že vpravo jsou "Right Brain Core Drives", která více souvisí s tvůrčím myšlením, sebe projevem a sociálními interakcemi (není potřeba cíl ani odměna, abyste mohli využít svou kreativitu, povídat si s přáteli nebo pociťovat napětí nepředvídatelnosti – samotná činnost je odměnou sama o sobě), zatímco vlevo jsou "Left Brain Core Drives", která jsou více spojená s logikou, analýzou a vlastnictvím (jste motivováni, protože chcete něco dosáhnout, ať už je to cíl, odměna nebo cokoli, co nelze okamžitě získat).

Každá ze zmíněných sil je podrobně rozebrána s ukázkami a doplněna dalšími informacemi z ověřených zdrojů. Toto je pouze první úroveň Octalysis Framework; existují ještě další čtyři úrovně, které zahrnují pokročilejší principy designu a hloubkovou analýzu. Nicméně úroveň 1 obvykle postačuje pro většinu společností, které se snaží vytvořit lépe navržený gamifikovaný produkt a zážitek.

Vznešený Smysl a Povolání

Je to živoucí hnací síla, kdy hráč věří, že vykonává poslání nad rámec své osobnosti, či že byl pro toto poslání „vyvolen“. Ukazatelem této síly jsou jednotlivci, kteří aktivně přispívají do různých fór, přinášejí svůj díl k lepšímu chodu komunity (například vytvářejí open-source projekty, publikují články na Wikipedii). Ti sami vědí, že to činí nezištně, avšak věří, že tím dělají něco prospěšného pro náš svět. Jako prvky gamifikace, které naplňují tuto sílu, mohou sloužit různé maskoty, které provázejí uživatele přes aplikaci, vyprávějí svůj příběh a uživatel s nimi interaguje, čímž začíná pociťovat sebe jako „vyvoleného“ či hrdinu. Navíc začlenění prvků vyprávění do aplikace může vytvořit emocionální pouto a zvýšit zapojení uživatelů. Ti se pak



Obrázek 2.5 Octalysis Framework[42]

lépe identifikují s příběhem aplikace, což zvyšuje pravděpodobnost, že se k ní budou opakovaně vracet.

Růst a Úspěch

Tato moc představuje sílu dosahování pokroku a úspěchu, rozvíjení dovedností a konečného překonávání výzev. Slovo „výzva“ zde nese zvláštní význam, neboť odznak či trofej bez výzvy postrádá jakýkoliv smysl. Stará se o to, aby se uživatel po splnění výzvy cítil uspokojen a věděl, že svým úsilím postupuje vpřed a není to jen zbytečné zabíjení času. Podněcuje je k tomu, aby nadále využívali danou platformu. Tato moc je jednoduchá na implementaci a může zahrnovat různorodé ocenění, body, odznaky, odměny, značky, žebříčky, progress bary, výkonové grafy a další. Tato rozmanitá ocenění jsou většinou využívána uživatelem k získání statusu, hmatatelných odměn nebo k posunu v žebříčku lídrů, čímž slouží jako virtuální symboly postavení, rozvoje a úspěchu. Kromě toho různé odměny vyvolávají uvolňování dopaminu, neurotransmiteru spojeného s pocitem potěšení, což podporuje pozitivní chování. [37] Klíčové ale je, aby uživatelé nedostávali tyto pocty zdarma, nýbrž za dosažené úspěchy, například za získané body, postup a umístění v žebříčcích.

Posilování Kreativity a Zpětná Vazba

Tento faktor pohání využívání kreativity a utváření prostředí skrze zpětnou vazbu. Zahrnuje zapojení uživatele do kreativního procesu, kde experimentuje s různými kombinacemi, často opakovaně, aby vyjádřil svou kreativitu. Následně se dočká zpětné vazby, kterou touží přijmout, neboť potřebují vidět výsledky své tvorby, přijmout zpětnou vazbu a reagovat na ni. To vytváří pocit svobody a kontroly nad jejich činnosti. Jako prvky gamifikace lze zde využít okamžitou zpětnou vazbu, odemčení různých kombinací, sbírání posilovačů nebo možnost volby (smysluplně nebo intuitivně). Lze si to lépe představit na příkladu - uživatel aplikace Duolingo si během používání nevyhnutelně položí některé otázky špatně. I když nesprávnost může být někdy vnímána negativně, Duolingo to bere jako příležitost umožnit svým uživatelům učit se a rozvíjet se. Po nesprávné odpovědi na otázku se zobrazí červeně zbarvené vyskakovací okno, které označuje chybu. Ačkoli červená může evokovat negativní dojem, díky jednoduchému designu je o něco

méně zastrašující. Toto okno umožňuje uživateli pokračovat a také nahlásit chybu na Duolingo nebo kliknout na tlačítko „bublina“ a dozvědět se o slovu více.[42] Uživatel tak může ihned poskytnout zpětnou vazbu a reagovat na zpětnou vazbu týkající se své tvorby.

Vlastnictví a Majetek

Tato síla spočívá v principu, kdy jsou uživatelé motivováni tím, že něco vlastní, a touží po jeho zdokonalení, organizaci, ochraně a získání dalších přínosů. Jedná se zejména o gamifikační prvky jako virtuální zboží a měny, věrnostní body a také možnost přizpůsobení a úpravy profilu či avatara podle vlastních představ. Díky této hnací síle se pro uživatele stává sbírání odznaků či skládání puzzle zábavným zážitkem.

Sociální Vliv a Vztahy

Tato část zahrnuje veškeré sociální prvky, které mohou motivovat jednotlivce, včetně mentorství, uznání, sociální interakce, společenství, ale také soutěživosti a závisti. Sociální vliv a vzájemné vztahy představují jádro lidské psychiky, které svůj úspěch zakládá na přirozené a někdy neodolatelné lidské potřebě spojení a vzájemného porovnávání. Při správném využití může tato síla sloužit jako jedna z nejsilnějších a nejtrvalejších motivací pro zapojení jednotlivců do společného dění. To může zahrnovat navazování kontaktů s ostatními uživateli, přidávání přátel do aplikace, sdílení osobních zkušeností, skupinové úkoly a soutěže s přáteli. Nicméně může také vyvolávat negativní aspekty, jako je soudění, negativní srovnávání se s ostatními, které může snižovat sebevědomí, a nezdravé úrovně závisti. Tato oblast je relativně dobře prozkoumána a populární, protože mnoho společností v dnešní době kladě důraz na optimalizaci svých online sociálních strategií.

Omezenost a Netrpělivost

Popisují touhu po něčem, co vidíte a chcete mít, avšak není to okamžitě dostupné. Například je to běžná praxe v hrách, kde můžete získat odměnu, ale ne ihned, nýbrž až po určité době (například za hodinu nebo den). Tato situace motivuje uživatele k přemýšlení o odměně, dokud ji nedosáhnou. Mezi příklady herních prvků, které tuto sílu pokrývají, patří nedostupné prémiové funkce (které se stávají dostupnými, když máte dostatek bodů nebo zakoupíte prémiový účet), odměny, které jsou dostupné pouze po určité době, nebo úrovně, jež jsou zatím nedostupné. Facebook je krásným příkladem, jak tento princip funguje: na začátku byl přístupný pouze pro studenty prestižních amerických škol, později se otevřel pro všechny školy a když se nakonec stal dostupný pro každého, stal se velmi populárním, protože mnoho lidí se chtělo připojit kvůli tomu, že dříve nemohli.

Nepředvídatelnost a Zvědavost

Principy, které ovlivňují lidskou touhu zjistit, co se stane dál. Když uživatel neví, co přijde, jeho mysl je aktivní a často se tím zabývá. Nicméně tato síla má také negativní aspekt - je základním faktorem v závislosti na hazardních hrách. Příklady tohoto principu zahrnují náhodné a neočekávané odměny, náhodné mini-kvízy, kolo štěstí, tajemné krabice (uživatel neví, co se v nich skrývá - může to být něco jednoduchého nebo naopak velmi cenného), náhodné denní výzvy a další.

Ztráta a Vyhýbání se

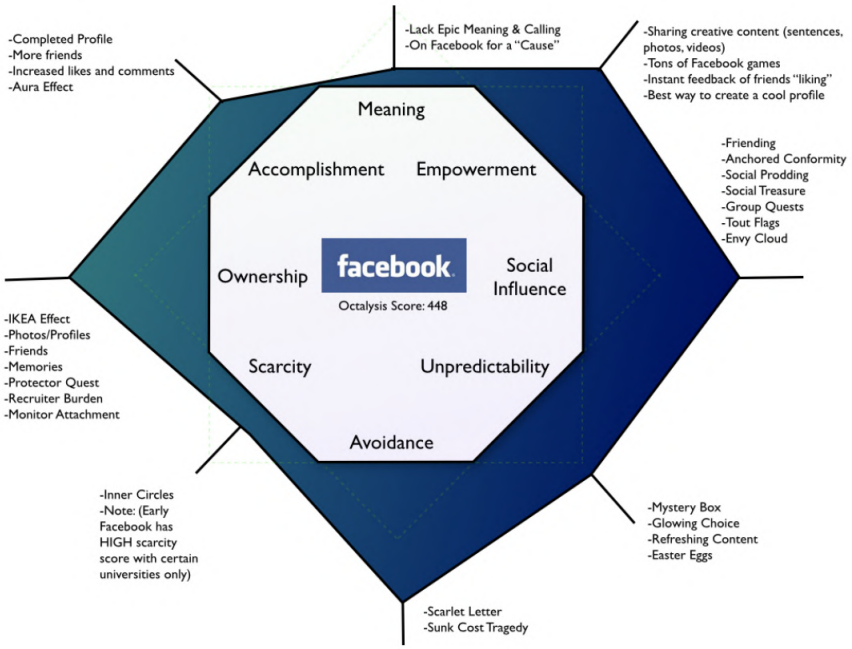
Motivy, které se zakládají na úniku před negativními situacemi. Tento pohon motivuje strachem z toho, že něco ztratíte, možná navždy, nebo že nastanou nežádoucí události. Je úzce spojen s principem ”Vlastnictví a Majetek”, kde vám může být odebráno něco, co jste si kdysi získali

a vypracovali. Mezi herní prvky spojené s tímto motivem patří ztráta pokroku, vyhýbání se neúspěchu, obava z úbytku sociálního zebříčku, strach ztráty bodů nebo statusu kvůli dlouhé neaktivnosti a další.

Příklad

Na obrázku 2.6 je zachycený Octalysis model aplikovaný na platformu Facebook. Graf ukazuje, že Facebook je silně orientován na "Right Brain Core Drives"(vnitřní motivace bez vnějších odměn). Dále je vidět, že aplikace aktivně podněcuje uživatele k opakované návštěvě každý den(hlavně síla "Unpredictability"). Další významnou silou je "Ownership", což znamená, že uživatelé jsou motivováni k větší aktivitě a angažovanosti, když něco sbírají, upravují a vylepšují to, co už je jejich (například kamarády, příspěvky, fotky atd).

Dále je patrné, že první síla - "Meaning"- je zde slabě zastoupena, což je dáno absencí jasně definovaného vyššího účelu používání Facebooku. Také je zde slabší prvek v šesté síle - "Scarcity", protože na Facebooku není takové množství obsahu, které by bylo omezené nebo zakázané, což by uživatele motivovalo.



Obrázek 2.6 Příklad Octalysisu pro Facebook[41]

2.3.3.2 Chyby při přidávání prvků gamifikace

Při implementaci herních prvků v mobilních aplikacích je třeba dbát na několik důležitých faktorů, které mohou ovlivnit výslednou kvalitu aplikace. Následující body popisují nejčastější chyby využívání gamifikace při vývoji aplikací [40]:

- **Přehnané množství herních prvků**. Je klíčové najít vyváženou míru a nedostat se do situace, kdy je aplikace přeplněná různými gamifikačními prvky. Přílišná komplexita může odvést pozornost uživatele od hlavního cíle aplikace.

- **Vytváření příliš konkurenčního prostředí**. Cílem není vytvářet prostředí plné soutěže, ale spíše motivovat uživatele k dosažení určitých cílů. Je důležité podporovat zdravou soutěživost mezi uživateli aplikace.
- **Nadměrné udělování odměn**. Časté a příliš hojné odměňování může vést k jejich devalvaci a ztrátě motivace uživatele. Je nutné pečlivě zvážit, kdy a za co jsou odměny udělovány, aby byly efektivní a udržely motivaci uživatele.
- **Nedostatečná variabilita odměn.**. Pokud jsou odměny po celou dobu stejné nebo opakující se, může to vést k úpadku zájmu a motivace uživatele. Je vhodné průběžně zvyšovat hodnotu odměn a přinášet nové varianty, aby uživatelé zůstali motivováni i po delší době používání aplikace.

2.3.4 Odměny

Jelikož odměny pro uživatele představují důležitou součást modulu "Virtuální body", je nutné se jim detailně věnovat, prozkoumat různé typy odměn a zhodnotit je z psychologického hlediska. Dále je třeba prozkoumat frekvenci získávání odměn, tj. jak často by měl uživatel dostávat odměny, aby stále měl motivaci k používání aplikace i v budoucnu.

2.3.4.1 Psychologie odměn

Odměny mají významnou roli při formování lidského chování. Z psychologického pohledu aktivují odměny centra mozku spojená s potěšením a uvolňují neurotransmitery, jako je dopamin, který vyvolává pocity štěstí a spokojenosti. Tato biologická reakce posiluje spojení mezi chováním a odměnou, čímž se zvyšuje pravděpodobnost opakovaného projevu požadovaného chování. Odměny mohou nabývat různých forem, včetně hmotných věcí, peněžních částek, ocenění, pochvaly nebo dokonce osobního uspokojení a radosti. [43]

V předchozí části věnované Octalysis Frameworku jsme již zmínili, že motivace uživatele lze rozdělit do dvou odlišných kategorií. Tyto kategorie motivace se nazývají vnitřní a vnější motivace a odkazují na sílu ovlivňující chování. Vnitřní motivace se projevuje, když uživatel vykonává určitou činnost z vlastního zájmu a považuje ji za obohacení nebo odměnu. Naopak vnější motivace nastává, když uživatel vykonává činnost za účelem získání vnější odměny nebo vyhnutí se trestu. [44] Tyto kategorie lze také aplikovat na odměny - vnitřní a vnější, a souvisejí s uvedenými typy motivace. [43]

Vnitřní odměny vycházejí z nitra uživatele. Příklady vnitřních odměn zahrnují pocit úspěchu, osobní růst, autonomii, vnitřní uspokojení a radost ze samotné činnosti. Tyto odměny často prožíváme, když se věnujeme činnostem, které rezonují s našimi vášněmi a zájmy. Vnitřní odměny mají dlouhodobý dopad na motivaci a spokojenost uživatelů. Když jsou jedinci vnitřně motivováni, pravděpodobněji se budou držet svých cílů i v obtížných situacích. Tato vnitřní síla podněcuje touhu zdokonalovat dovednosti, dosahovat osobních cílů a prožívat pocit naplnění.

Vnější odměny jsou naopak poskytovány někým jiným, například v případě mobilních aplikací jsou to odměny poskytované samotnou aplikací nebo jejími tvůrci. Ty mohou být v podobě bonusů, propagačních akcí, dárků nebo veřejného uznání. Jejich účelem je motivovat jednotlivce pomocí vnějších podnětů, často spojených s dosažením konkrétních výsledků nebo cílů. I když mohou být účinné jako krátkodobá strategie, dlouhodobě neposkytují trvalou motivaci a spokojenost. Naopak, s časem mohou snižovat motivaci, neboť uživatelé činnost vnímají pouze jako prostředek k dosažení odměny, nikoli jako samostatně uspokojující aktivitu. Dále mohou vnější odměny vést k vytváření konkurenčního prostředí, kde jednotlivci soutěží o odměny místo toho, aby se skutečně zapojovali do činnosti a rozvíjeli své zájmy, což může bránit kreativitě a vnitřní motivaci uživatelů.

Je podstatné, aby tyto obě formy odměn existovaly společně a navzájem se doplňovaly v mobilních aplikacích, protože každá z těchto forem má svůj vliv na motivaci a spokojenost

uživatele. Zatímco obě formy motivace mohou být účinné, vnitřní motivace je obecně považována za udržitelnější a vede k dlouhodobému úspěchu a motivaci, zatímco vnější odměny mohou sloužit jako silná krátkodobá strategie. Pro dosažení lepších výsledků je však důležité umět integrovat tyto dvě formy odměn dohromady: vnější odměny mohou sloužit jako katalyzátor pro jednotlivce, aby objevili vnitřní motivaci a rozvinuli skutečnou vášeň pro danou aktivitu, a naopak vnitřní odměny mohou zvýšit hodnotu a význam, který uživatelé získávají z vnějšího uznání a odměn. Autor Octalysis Framework, Yu-kai Chou, tuto myšlenku potvrzuje slovy: "je lepší přilákat lidi do zážitku pomocí vnějších odměn (dárkové karty, peníze, zboží, slevy), poté převést jejich zájem prostřednictvím vnitřních odměn (uznání, status, přístup)". [41]

Jako příklad můžeme uvést aplikaci od společnosti McDonald's. Zaznamenala úspěch s přístupem vnějších odměn a podpořila instalaci aplikace tím, že nabídla bezplatný sendvič jako odměnu. Lákadlo na bezplatný sendvič účinně zvýšilo motivaci těch, kteří by si jinak aplikaci nenainstalovali. [36] Ale tento úspěch je pouze krátkodobý, jelikož když přestanou nabízet slevy a kupony, uživatel rychle přestane být motivován a aplikaci přestane používat. Takže zde vidíme, jak je důležité zkombinovat obě formy odměn pro dlouhodobý zájem o aplikaci.

2.3.4.2 Typy odměn

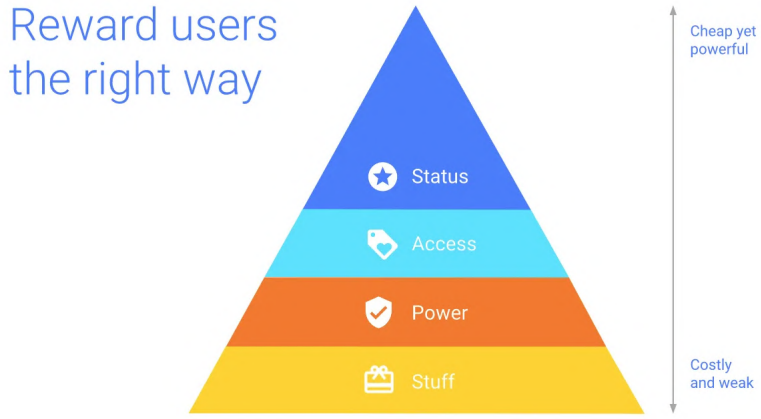
Z analýzy psychologie odměn vyplývá, že k přilákání uživatele je nezbytné pečlivě a ohleduplně navrhnout vnější i vnitřní odměny. Ty vnější slouží jako výchozí bod k objevení vnitřní motivace, zatímco ty vnitřní podporují a stimulují skutečný zájem o danou aplikaci.

Odborník na gamifikaci, Gabe Zichermann, vytvořil hierarchii odměn [36], která je založena na jejich relativní schopnosti vést k zapojení a udržení uživatelů, označovanou zkratkou SAPS - Status, Access (Přístup), Power (Moc), Stuff (Věci). Na obrázku 2.7 je zobrazena hierarchie SAPS. Z něj lze vyčíst, že odměny lze rozdělit do čtyř typů:

- **Statusy** - slouží jako indikátory moci (například VIP výhody, body, odznaky), jež lze snadno předvést ostatním uživatelům.
- **Přístup** - může zahrnovat exkluzivní obsah, k němuž ostatní uživatelé nemají přístup, stejně jako přednostní přístup k různým aktualizacím nebo nabídkám (například předprodej vstupenek).
- **Moc** - uživatel získává jako odměnu určitý status a práva, která může využívat, přičemž aplikace nabízí několik různých typů těchto práv (například hlasování o nové funkce nebo moderování ostatních uživatelů).
- **Věci** - mohou to být peněžní odměny, kupony, dárky apod.

V pyramidě SAPS je zřejmé, že vnitřní motivace narůstá při postupu zdola nahoru, zatímco významnost odměn roste sestupem shora dolů. Dolní odměny jsou ideální k přilákání uživatelů do aplikace, zatímco horní odměny slouží k udržení uživatelů v aplikaci. Z toho vyplývá, že mechanismus odměn by měl být pružný a aktualizovat se dle různých statistik aplikace, jako je uživatelská aktivita, výhrady vůči aplikaci, počet uživatelů, kteří aplikaci odinstalovali nebo opustili apod.

Další důležitou úvahou při vytváření mechanismů odměn je zajistit, aby odměny byly distribuovány náhodně a nečekaně. Náhodné odměny znamenají, že uživatel předem neví, jakou odměnu obdrží; může to být něco jednoduchého nebo naopak velmi cenného. Pravidelné náhodné udělování odměn udržuje hráče motivované a podněcuje je k návratu do aplikace každý den. Wendy Despain, autorka knihy "100 Principles of Game Design", uvádí: „Pokud hráč přesně ví, jakou odměnu dostane za konkrétní akci, odstraní to jakékoli překvapení [...] zážitek [odměna] zanechá pocit prázdnosti a nezaujatosti". [36] Neočekávaná odměna zajišťuje, že uživatel neví přesně, kdy odměnu obdrží, a tím je motivován k častějšímu používání aplikace.



Obrázek 2.7 Hierarchie SAPS[36]

2.3.4.3 Četnost získávání odměn

Při tvorbě aplikací s využitím odměn jako gamifikačních prvků je mimořádně důležité pečlivě zvážit, jak často a za jaké akce má uživatel získávat odměny, aby to nebylo příliš časté a odměny nebyly udělovány příliš snadno a "zdarma". Takový přístup by mohl uživatele odradit a snížit jeho motivaci. V důsledku by pak uživatel mohl přestat aplikaci používat. Proto je klíčové zvolit vhodný plán (někdy nazývaný i rozvrh) odměn.

V reálném životě není určité chování odměňováno pokaždé, kdy k němu dojde. Obvykle se postupuje podle specifického plánu posilování. Tento přístup lze využít i v případě mobilních aplikací. Plán odměn v aplikacích slouží k posílení specifického chování uživatele tím, že je odměňován za provedení určitých akcí. Tyto plány odměn lze rozdělit do dvou typů: Neustálé posilování (Continuous Reinforcement) a Částečné posilování (Partial Reinforcement).

Plán neustálého posilování znamená, že uživatel je odměňován za každou provedenou akci. Jedná se o nejčastěji používaný přístup, zejména při lákání uživatele na začátku, tedy v momentě, kdy se poprvé setkává s určitými akcemi. Tento přístup je nejúčinnější při formování nového chování. [45]

Jakmile požadované chování uživatele je stanoveno nebo uživatel aplikaci používá delší dobu, plán neustálého posilování se obvykle mění na plán částečného posilování. Částečné posilování lze rozdělit do 4 základních typů odměn [45, 46, 47]:

- **Fixed Ratio** nebo fixní poměr: odměna je udělena po stanoveném počtu opakování určité akce. Tento počet je přesně stanoven a uživatel ho zná. Například, uživatel ví, že dostane odměnu, když sdílí něco s kamarády 10krát. Tento plán odměn poskytuje uživateli pocit spolehlivosti, jelikož očekává stejný výsledek pokaždé a ví, kdy dostane odměnu.
- **Variable Ratio** nebo variabilní poměr: uživatel přesně neví, kolik akcí musí provést, aby dostal odměnu. Odměnu získá po náhodném počtu provedených akcí. Hazardní hry a loterijní hry jsou dobrými příklady odměn založených na rozvrhu s variabilním poměrem. Z těchto plánů je tento nejvýkonnější. Uživatel je si jist, že v nějaké určité fázi nakonec vyhraje, ale neví přesně, kdy, což povzbuzuje uživatele, aby neustále používali aplikaci.
- **Fixed Interval** nebo fixní interval: odměna je stanovena po stanovené časové době. Důležité upozornění je, že uživatel může dokončit akci jednou nebo stokrát, ale stále získá odměnu pouze jednou za časový interval. Často to vede k zvýšené aktivitě v bodě, kdy je udělena odměna, ale po odměně následuje pokles. Tato technika je skvělá pro povzbuzení uživatelů k účasti v aplikaci v určitou dobu, ale není vhodná pro posilování návyku.

- **Variable Interval** nebo variabilní interval: uživatel neví přesný čas, kdy dostane odměnu. Udělení této odměny však není závislé na jejich účasti, ale pouze na tom, zda uživatel dokončil akci alespoň jednou. Příkladem takového rozvrhu je, když je uživatel odměněn za přihlášení do aplikace po náhodném čase. Tento plán je používán velmi zřídka, protože reakce uživatelů na tento rozvrh se mohou pohybovat od nulového vlivu na chování až po odrazení od dalšího dokončení akce, nebo dokonce odradit od používání aplikace navždy.

Je klíčové kombinovat variabilní způsoby rozvrhu odměn a flexibilně je upravovat podle potřeby, aby bylo dosaženo různých efektů a zajištěno udržení dlouhodobé angažovanosti uživatele. Dalším důležitým aspektem je precizní specifikace konkrétního úkonu, za který bude uživatel odměněn, spolu s výběrem vhodného typu odměny (mohou to být různorodé hodnotné benefity). Účinnost zvolených plánů je pak závislá na tom, jaký typ odměn je vhodný. Testování odezvy uživatelů na dané plány odměn je dalším klíčovým faktorem, který umožňuje posoudit jejich reakce a případně upravit rozvrhy dle potřeby.

2.4 Fórum fanoušků nebo chat

Máme výsadu, že žijeme v éře internetu, což znamená, že máme přístup k prakticky veškerým informacím skrze síť a můžeme navazovat komunikaci s lidmi z jakéhokoli koutu světa, a to z pohodlí našich domovů. Nicméně tato epocha s sebou nese i své temné stránky a výzvy, jako je kybernetický útok, rasismus a virtuální šikana. Jedním z projevů takového chování je užívání nenávistných výrazů a hrubých slov v komentářích či diskuzních místnostech online. Proto je pro aplikace, které integrují takové prvky jako veřejné komentáře nebo diskuzní fóra, zásadní monitorovat užití těchto slov a nějakým způsobem provádět moderaci a filtraci. Aplikace Eliterro rovněž nabízí chat pro fanoušky, a proto je klíčové pečlivě prozkoumat různé nástroje a metody monitorování obsahu, ukázky použití těchto nástrojů a metod a vybrat tu nejvhodnější variantu.

2.4.1 Metody moderování obsahu

Je důležité vytvářet a udržovat bezpečné prostředí na internetu, a základním prvkem k dosažení tohoto cíle je moderace obsahu - zajišťuje, že veškerý zveřejněný materiál je kontrolován a vyhodnocen. Proces moderace zahrnuje uplatňování předem stanovených pravidel a omezení pro monitorování obsahu vytvořeného uživateli. Pokud materiál nesplňuje tato pravidla, označuje se jako nevhodný a je odstraněn. Moderace může být aplikována na všechny druhy obsahu v závislosti na zaměření platformy - text, obrázky, video a dokonce i živá vysílání. Vzhledem k tomu, že cílem této diplomové práce je vytvoření fóra, bude se analýza moderace zaměřovat především na textový obsah. Existuje šest základních metod moderace obsahu, přičemž každá metoda bude popsána samostatně.[48, 49, 50]

Před-moderování

Před moderováním se rozumí proces moderování obsahu, kdy moderátor (zde a dále se nemusí jednat pouze o lidského moderátora, ale také o umělou inteligenci pro moderování obsahu) prochází příspěvky ještě před jejich zveřejněním. Přijímá frontu odeslaných příspěvků a postupně je kontroluje, zda dodržují pravidla a neobsahují žádné nevhodné informace. Tento přístup zajišťuje vysokou úroveň kontroly nad obsahem a snižuje riziko, že uživatel se setká se škodlivým nebo urážlivým obsahem. Na druhou stranu to však zpomaluje publikování obsahu, což může být frustrující pro uživatele a odradit je od dalšího používání platformy. Takový způsob moderování je například využíván na webech a aplikacích, kde uživatelé chtějí něco inzerovat a vytvářet inzeráty. Dalším typem platforem, kde je tento přístup považován za oblíbený, jsou online komunity zaměřené na děti, kde je velmi důležité zachytit šikanu nebo sexuální manipulaci ještě před publikací obsahu.

Post-moderování

Post-moderování znamená, že obsah je zveřejněn pro veřejnost okamžitě poté, co je vytvořen. Až po jeho zveřejnění je zařazen do fronty ke kontrole moderátorem a pokud moderátor usoudí, že je tento obsah nevhodný, je odstraněn, aby byli chráněni ostatní uživatelé před tímto nevhodným obsahem. Většina platforem se snaží co nejvíce zkrátit dobu kontroly, aby nevhodný obsah byl online co nejkratší dobu. Hlavní výhodou takového způsobu moderování je rychlejší publikování obsahu. Nevýhodami jsou větší riziko, že uživatel narazí na nevhodný a škodlivý obsah, a navíc správa většího objemu obsahu může být velmi náročná.

Reaktivní moderování

Reaktivní moderování zahrnuje účast uživatelů platformy nebo aplikace na hodnocení obsahu společně s moderátory. To znamená, že uživatelé mají možnost rozhodnout, zda příspěvek schválit či nikoli. Tento způsob moderování je většinou realizován prostřednictvím tlačítka "Nahlásit", které upozorní moderátora a umožní mu prověřit daný obsah. Tato metoda snižuje zátěž pro moderátory a zároveň zvyšuje interakci a zapojení uživatelů. Avšak hlavní nevýhodou tohoto přístupu je riziko zneužití systému ze strany uživatelů, kteří mohou nahlašovat obsah z osobních důvodů, i když ten obsah splňuje pravidla. Navíc nevhodný obsah zůstane viditelný až do doby, než jej někdo nahlásí a moderátor ho zkontroluje a odstraní.

Reaktivní způsob moderování dosahuje optimálních výsledků ve spojení s post-moderováním nebo před-moderováním. Uživatelé mohou označit obsah za nevhodný i po jeho moderaci, což umožňuje dvojitou ochranu proti nevhodnému obsahu. Proto je zásadní poskytnout srozumitelné pokyny komunitě (v nichž bude jasně vymezeno, co je na dané platformě povoleno a zakázáno) a stanovit jasné a srozumitelné sankce za porušení pravidel. [51]

Distribuované moderování

Samotné distribuované moderování se podobá spíše reaktivnímu přístupu a vychází z zpětné vazby uživatelů. Tento způsob moderování využívá systém hlasování, který rozhoduje o tom, který obsah je vhodný a posouvá ho výše, a který není, a posouvá ho dolů. Obsah, který je posunut dolů, je poté buď skrytý, nebo smazán. Tento přístup je obvykle vhodný pro kontrolu komentářů a příspěvků na diskusních fórech, které se často nacházejí v rámci určité komunity, a jsou provozovány zkušenými "seniorními"uživateli (nebo pouze VIP uživateli, kteří mají právo hlasovat). Tento model moderace je vhodné propojit s odměnami pro uživatele: ti mohou získávat reputaci nebo body jako ocenění za jejich úsilí v moderaci. Hlavní výhodou tohoto systému je zapojení komunity do moderování a snížení potřeby externích moderátorů. Avšak tento způsob moderace může vést k neobjektivnímu hodnocení, může s sebou nést jak právní, tak obchodní rizika, a navíc nevhodný obsah bude viditelný, dokud nebude odhlasován.

Automatizované moderování

Automatizované moderování využívá nástroje umělé inteligence a algoritmy strojového učení k řízení obsahu. Tyto systémy detekují a odstraňují obsah na základě předem definovaných kritérií a vzorů. Jedná se o velmi efektivní řešení, neboť vyžaduje méně času na zpracování obsahu a menší lidskou námahu. Tento způsob moderování je zajímavý z technologického hlediska, a existuje několik typů automatizovaného moderování. Proto je vhodné detailněji prozkoumat tyto různé přístupy v následující sekci. Velkou výhodou tohoto přístupu je efektivní zpracování velkého objemu obsahu a kontinuální práce 24/7 bez nutnosti zásahu člověka. Pokud jde o nevýhody, hlavním problémem je možnost generování falešných výsledků a nižší efektivita v detekci nevhodného obsahu ve srovnání s lidskými moderátory, zejména v obtížných situacích, kde je potřeba porozumět kontextu.

Uživatelsky řízené moderování

Uživatelsky řízené moderování znamená, že samotní uživatelé přebírají roli moderátorů. Tento přístup je jedním z nejrizikovějších a nejméně používaných metod moderování, neboť nepředpokládá existenci externího moderátora, který by mohl řídit obsah. Doporučuje se pouze v případech, kdy platforma či aplikace má malý počet uživatelů. Absence externího moderátora může vést k rychlému pádu platformy do anarchie a atmosféra se pravděpodobně stane nepříjemnou, což může odrazovat potenciální nové uživatele. [48]

Jaky typ moderování vybrat?

Volba metody moderování závisí na typu platformy či aplikace, charakteru moderovaného obsahu a komunity, která danou platformu využívá. Některé vyžadují přísnější omezení a větší opatrnost při moderování, jiné naopak preferují menší nebo dokonce žádný zásah. Různé typy moderování lze také kombinovat a vytvářet tak rozmanité a efektivní systémy moderování. V dnešní době je velmi účinným a oblíbeným přístupem spojení automatizovaného moderování s lidskými moderátory (kteří mohou využívat různé formy moderování, většinou se jedná o reaktivní moderování), jelikož tento hybridní přístup zaručuje, že po kontrole obsahu automatizovaným moderátorem je vše, co nebylo zachyceno pomocí umělé inteligence, zpracováno lidským uživatelem. Takový hybridní přístup využívají různě známé aplikace jako například Instagram, Facebook apod.

Důležité je si uvědomit, že moderování nesmí sloužit k omezení kreativity, ale pouze k vytvoření a udržení bezpečného a respektujícího prostředí.

Když hovoříme o moderování textového obsahu, je třeba si uvědomit, že pouhé zachycení klíčových vulgarismů nestačí, protože nevhodný text může být vytvořen pomocí různých triků k vyhýbání se filtrům vulgarismů. Navíc je důležité zohlednit různé nuance a kulturní specifika.

2.4.2 Typy automatizovaného moderování

Automatizované moderování je poměrně nová technologie, která se však rychle rozvíjí, získává popularitu a je čím dál častěji využívána (avšak ve většině případů stále vyžaduje lidského moderátora). To zejména proto, že je schopna zpracovávat velké objemy dat, efektivně zvládá škálovatelnost systému a většinou nepotřebuje zásah člověka. Automatizované moderování je realizováno pomocí algoritmů umělé inteligence, které identifikují nevhodný obsah na základě předchozích dat. Existuje mnoho technik automatizovaného moderování obsahu, které lze použít, přičemž přesné řešení závisí na charakteru obsahu. [52, 53]

Text

Nejčastější metodou pro upravování textu jsou algoritmy zpracování přirozeného jazyka, zkráceně NLP (Natural Language Processing). Zpracování přirozeného jazyka je součástí umělé inteligence a v podstatě to znamená schopnost počítačových programů porozumět, rozpoznat a vytvářet lidský jazyk a lidskou řeč. Pro pochopení struktury a významu textu mohou být využity dva odlišné přístupy: strojové učení nebo pravidlové metody. Při diskusi o NLP se obvykle hovoří o dvou hlavních fázích: předzpracování dat a vývoj algoritmů. [54]

První fáze se zabývá přípravou a úpravou textového obsahu tak, aby byl srozumitelný pro stroje a aby mohly být tyto stroje schopny jej zpracovat (například převádí text do funkční formy a zdůrazňuje prvky v textu). Existuje několik metod předzpracování textových dat, mezi něž patří například ”lematizace a odvozování”- shlukuje slova se stejným kořenem nebo slova různého skloňování; nebo ”slovní značkování”- označuje slova podle slovního druhu, tedy například seskupuje podstatná jména, slovesa nebo přídavná jména.

Po předzpracování dat následuje druhá fáze - vývoj algoritmu pro zpracování těchto dat. Nejčastěji se využívají dva hlavní typy: systém založený na pravidlech nebo systém založený na

strojovém učení. První systémy využívají předem navržená lingvistická pravidla, zatímco druhé využívají statistické metody (učí se provádět úkoly na základě trénovacích dat, podle kterých upravují a zlepšují své algoritmy a metody).

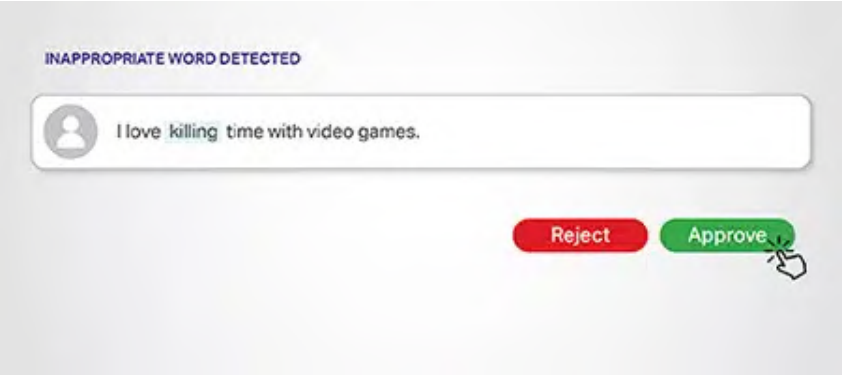
Když hovoříme o metodách, které využívají algoritmy zpracování přirozeného jazyka, hlavními jsou syntaktická a sémantická analýza. Syntaxe v NLP posuzuje význam textu na základě gramatických pravidel, zatímco sémantická analýza se zabývá porozuměním významu a struktury vět. Mezi syntaktické algoritmy patří například gramatický rozbor věty, rozdělení textu na menší části (například rozpoznání tečky, která odděluje věty) nebo morfologická segmentace, což znamená rozdělování slova na menší jednotky, nazývané morfémy. Mezi sémantické techniky patří například rozdělení významu slova - odvozování významu slova na základě kontextu nebo rozpoznání pojmenovaných entit (NER - Named Entity Recognition) - určuje slova, která lze seskupit do skupin (entit), a navíc dokáže rozlišit entity, které jsou vizuálně stejné, ale mají různý význam v kontextu.

Když se jedna o moderování textového obsahu, takové algoritmy se využívají k porozumění zamýšleného významu a kontextu textového obsahu, dekódování vložených emocí a přiřazování kategorií nebo emocí k textu na základě provedené analýzy.

Například, existuje Sentiment Analysis (analýza sentimentu), která dokáže identifikovat tón daného textového obsahu a odhalit emoce skryté za textem. Dokáže to rozdělit do negativních kategorií, jako jsou například šikana, hněv, sarkasmus, a pak označit text jako pozitivní, neutrální nebo negativní.

Další metody zpracování přirozeného jazyka mohou se zaměřit na detekci specifických klíčových slov v textu. Tato technika pracuje s předem definovaným seznamem slov, která jsou považována za problematická. Jestliže text obsahuje tato slova, systém může buď odmítnout jeho zveřejnění, nebo jej předložit k další kontrole (v závislosti na zvoleném přístupu k moderaci). [55]

Další metodou moderování textového obsahu je tzv. kontextové moderování. Tato strategie hodnotí obsah v rámci širšího kontextu. Na ilustraci 2.8 je patrné, že ve větě je použito slovo "killing", avšak v daném kontextu není škodlivé, neboť se odkazuje na uživatele, kteří rádi tráví čas hraním her. Tato forma moderace zajišťuje, že obsah obsahující zakázaná nebo škodlivá slova, ale ve vhodném kontextu, nebude neoprávněně blokován a bude povolen.



Obrázek 2.8 Příklad kontextového moderování[55]

Dále je důležité zdůraznit, že existuje alternativní přístup k monitorování vulgarismů v textovém obsahu, nazývaný "filtr vulgárních výrazů"(v angličtině profanity filters). Tyto filtry zabrání zveřejňování urážlivého textu tím, že automaticky prověřují text podle předem určeného seznamu urážlivých slov nebo frází, které jsou zakázány pro danou platformu nebo aplikaci. Navíc obvykle takové seznamy lze upravit pro každého uživatele individuálně, takže každý uživatel může přidat konkrétní slova do seznamu zakázaných slov, která budou "zablokována"pouze pro něj. Kromě toho algoritmus filtru vulgárních výrazů nejen identifikuje slova, která jsou uvedena v

seznamu, ale také slova, která se uživatel pokouší znejasnit použitím v zakázaném slově číslic nebo jiných znaků (přestože uživatelé občas tyto filtry obcházejí pomocí složitějších triků, jak je popsáno v této studii [56], která uvádí 6 různých způsobů, jak se těmto filtrům vyhnout - mezi ně patří opakování znaků, oddělení písmen ve slově, vkládání slov do jiných slov a podobně). Výsledky filtru vulgárních výrazů lze dále využít různými způsoby: předáním výsledků ke kontrole lidskému moderátorovi, zablokováním a odmítnutím celého textového obsahu obsahujícího urážlivá slova, nahrazením urážlivých výrazů hvězdičkami, nahrazením alternativními výrazy a dalšími. Filtr vulgárních výrazů je většinou implementován pomocí integrovatelného API rozhraní, které běží na pozadí, když uživatelé posílají zprávy nebo přidávají příspěvky. [57]

Hlavní rozdíl mezi filtrováním vulgárních výrazů a ostatními metodami automatizovaného moderování textu (často označovanými jako služby pro moderování textu) spočívá v tom, že ostatní metody využívají umělou inteligenci, která dokáže porozumět záměru a kontextu zveřejněného textu. To znamená, že na rozdíl od použití předem definovaného seznamu urážlivých slov se snaží najít jakékoliv úmysly nebo souvislosti, které mohou být urážlivé (může se stát, že obsah neobsahuje žádná zakázaná slova, ale celková myšlenka tohoto textu je urážlivá). Navíc filtry vulgárních výrazů vrátí pouze seznam slov nalezených v textu, zatímco služby pro moderování textu poskytnou podrobnosti o kontextu obsahu a vysvětlí, proč byl tento text označen jako urážlivý.

Hlas

Pokud se týká moderování hlasového obsahu, většinou se používá technologie známá jako analýza hlasu. Tato technologie spočívá v kombinaci dalších metod umělé inteligence, jako je převod hlasu na text, zpracování přirozeného jazyka nebo interpretace tónu hlasu.

Vizuální obsah

Pro automatizované moderování obrázků se využívají techniky založené na počítačovém vidění. Tyto techniky využívají různé algoritmy, které detekují škodlivé objekty nebo celé obrázky a navíc lokalizují konkrétní pozice škodlivého prvku na obrázku. Kromě toho existují algoritmy, které zpracovávají a rozpoznávají texty na obrázcích a dokážou moderovat i obsah tohoto textu! Mezi typy nevhodného vizuálního obsahu, které umí zachytit algoritmy automatizovaného moderování, již patří například nahota a pornografie, zbraně, alkohol a zakázané látky, slovní útoky a rasismus, propaganda terorismu atd.

Video

Automatizované moderování videa představuje nejkomplexnější proces, neboť vyžaduje pečlivou kontrolu celého záznamu snímek po snímku a využití kombinace různých technik, které byly již popsány pro jiné typy obsahu: počítačové vidění identifikuje nevhodné prvky ve vizuálních částech, zatímco pro kontrolu hlasového obsahu se používají algoritmy pro analýzu hlasu. Automatizované moderování lze aplikovat nejen na již natočená videa, ale i na živé vysílání, kde promítání probíhá v reálném čase.

2.4.3 Oblasti použití

Na internetu, kde lidé navzájem komunikují a interagují, může být šířen nevhodný obsah. Avšak v následujících oblastech je kvalitní moderace obsahu nezbytná [50, 57, 58]:

- **Sociální sítě či blogové platformy** slouží k sdílení různorodého obsahu mezi uživateli a často obsahují sekci komentářů, kde mohou ostatní uživatelé vyjádřit své názory. Nicméně tyto sdílené příspěvky nebo komentáře mohou být často negativní a urážlivé. Proto je velmi důležité používat správně nastavenou službu moderování obsahu na těchto platformách.

Moderování obsahu na takových platformách představuje výzvu kvůli velkému množství uživatelsky generovaného obsahu s různými kontexty a formáty (obrázky, text, videa, audio). Dobře nastavené moderování obsahu na těchto platformách zaručuje bezpečnost uživatelů, reputaci platformy, soulad s právními předpisy a aktivní zapojení uživatelů. Světově známé platformy jako Facebook, Twitter (známý také jako X), YouTube nebo Instagram využívají různé metody moderování, včetně automatizovaného moderování, moderování prováděného lidmi a nahlášení obsahu komunitou. Kromě toho mají přesně definované zásady a pokyny týkající se sdíleného obsahu.

- **Platformy a aplikace s chatem nebo fórem** jsou běžnou součástí online prostředí. Když hovoříme o aplikacích s online chatem, často se jedná o herní platformy nebo streamovací služby (například Twitch), které často využívají i mladí lidé. Proto je velmi důležité monitorovat chaty a chránit je před nenávistnými projevy. Navíc online hry často vyvolávají soutěživé prostředí, které může vést k konfliktům, šikaně nebo projevům nenávisti vůči ostatním hráčům. I když se nejedná o hry, ale o jiné aplikace s chatem či fórem, je stále důležité sledovat obsah, který uživatelé posílají, a zveřejňují pro širší publikum, aby bylo vytvořeno bezpečné a přátelské prostředí a podporovalo zdravou a bezkonfliktní interakci mezi uživateli. Aplikace jako Discord nebo Twitch například zaměstnávají lidské moderátory pro každý kanál nebo server a poskytují systém nahlášení pro uživatele, aby mohli upozornit na nevhodný obsah. Samozřejmě také využívají automatizované moderování, jako jsou například boti pro Discord, kteří automaticky filtrují spam nebo urážlivý jazyk, nebo automatizované systémy pro Twitch, které detekují a odstraňují spam, nenávistné projevy a další porušení pravidel komunity.

- **Online tržiště, či marketplace**, čelí řadě problémů, které je třeba řešit. Mezi ně patří nelegální nabídky, jako je prodej zbraní, drog nebo exotických zvířat, stejně jako podvodné praktiky. Proto je důležité, aby byly nabídky, které uživatelé chtějí zveřejnit, moderovány. To je zásadní k tomu, aby se zabránilo distribuci nelegálního zboží, chránilo uživatele před podvody a vytvářelo důvěryhodné prostředí, které podporuje vzájemnou důvěru. Například Amazon používá pro moderaci obsahu na svém tržišti filtrování klíčových slov (k automatickému označování nabídek obsahujících zakázaná slova), rozpoznávání obrázků k prohledávání přiložených fotografií a také moderuje recenze nabízených produktů, aby nedocházelo k spamu nebo obsahu nevhodného obsahu. Dalším příkladem je Facebook Marketplace, který využívá algoritmy umělé inteligence k analýze obrázků a textu, komunitní hlášení (uživatelé mohou nahlásit nevhodné nabídky) a také monitorování v reálném čase, které sleduje a odstraňuje nelegální a škodlivé položky.

- **Seznamovací aplikace** stojí na komunikaci, a proto je zásadní zajistit bezpečnost a vytvořit pozitivní komunikační prostředí pro uživatele. Je důležité oddělit uživatele od různých forem nevhodného obsahu, jako jsou osobní útoky, fanatismus nebo sexuální obtěžování, což je běžné v aplikacích tohoto typu. Například aplikace Tinder využívá jak lidské moderování, tak i automatizované metody (jako je rozpoznávání nevhodných obrázků, které uživatelé nahrávají do svých uživatelských profilů, detekce falešných profilů nebo analýza textového obsahu). Uživatelé této aplikace mají navíc možnost nahlásit nevhodné profily nebo fotografie, které jsou poté kontrolovány lidskými moderátory.

- **Aplikace určené pro děti** vyžadují zvlášť pečlivou kontrolu a moderaci obsahu, aby se předešlo nevhodným situacím, jako je obsah nevhodný pro věk, sexuální obtěžování, kontakty pedofilů lákající děti na osobní setkání offline, kyberšikana a další nebezpečí. Jedním z příkladů takové dětské aplikace je YouTube Kids, který využívá algoritmus automatického filtrování na základě klíčových slov a metadat videí, lidské moderování a také rodičovskou kontrolu (rodiče mohou nastavit preference a omezení, včetně konkrétních kanálů a témat, ke kterým mají nebo nemají děti přístup). Dalším příkladem je oblíbená herní platforma pro děti, Roblox, která umožňuje vytváření vlastních her a také hraní her vytvořených ostatními

uživateli. Tato platforma využívá chatovací filtry pro automatickou detekci nevhodného obsahu v rámci chatovací funkce, kontrolu obsahu vytvořeného uživateli (lidskými moderátory, kteří ověřují soulad vytvořených her s komunitními standardy a pokyny) a navíc umožňuje uživatelům nahlásit nevhodný obsah či chování moderátorům.

2.5 Vstupenky

Pro modul "Správa a nákup vstupenek"je nejdůležitějším implementovaným prvkem samotný nákup vstupenek. Je zásadní zajistit bezpečný a spolehlivý proces nákupu vstupenek a následné platby prostřednictvím aplikace Eliterro. Při implementaci online zpracování plateb nelze obejít platební bránu, která je nedílnou součástí tohoto procesu a umožňuje uživatelům bezproblémově a bezpečně zadat své platební údaje a provést transakci. Je klíčové správně vybrat a integrovat platební bránu do vaší platformy, neboť 1 z 5 zákazníků opustí své online nákupy kvůli dlouhému a složitému procesu platby. [59] Proto je důležité zaručit rychlý, jednoduchý a bezpečný proces platby. Existuje několik typů platebních bran a správný výběr závisí na mnoha různých faktorech. Je nezbytné porozumět pojmu platební brána, fungování procesu provedení online platby, existujícím typům platebních bran, jejich rozdílům a poskytovaným bezpečnostním prvkům.

2.5.1 Platební brána

Platební brána je technologií, která slouží k autorizaci plateb kartou, tedy k ověření a bezpečnému přenosu citlivých platebních údajů mezi různými účastníky platebního procesu. Jedná se o technickou vrstvu, která ověřuje údaje kreditní nebo debetní karty uživatele a kontroluje, zda má uživatel dostatek peněz na účtu k provedení platby. Obvykle se považuje za "front-end", neboť uživatelé s ní přímo komunikují.

I když se tato diplomová práce zaměřuje na online platební brány, je důležité poznamenat, že existují i fyzické platební brány, které se používají v kamenných prodejnách. Fyzické platební brány využívají POS (point-of-sale) terminál, který přijímá informace o bankovních kartách pomocí fyzické karty nebo bezkontaktní platební metody, jako jsou chytré telefony s technologií NFC (Near Field Communication), a následně zpracovává tyto informace a platby elektronicky.

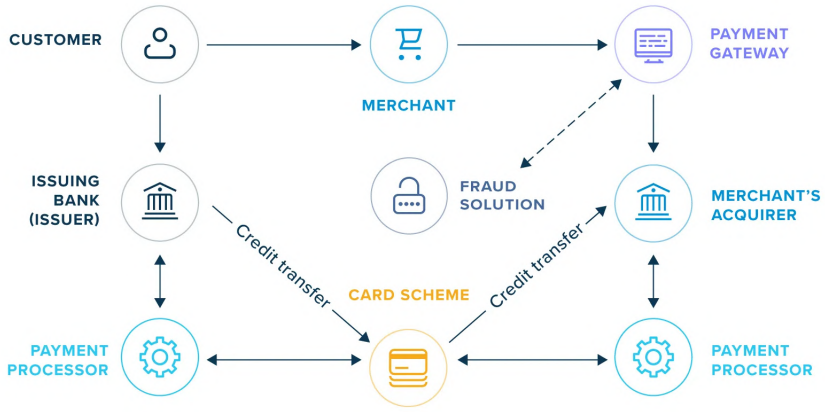
Pokud jde o online platební brány, ty přijímají informace prostřednictvím zadání karetních údajů uživatelem elektronicky (do textových polí) nebo pomocí přihlašovacích údajů k digitální peněžence (například peněženka Google, která ukládá zašifrovaná data karet do telefonu a zrychluje komunikaci uživatele s platebními branami).

2.5.1.1 Jak to funguje?

I přestože pro uživatele platba vypadá pouze jako jednoduché vyplnění několika kroků, ve skutečnosti se odehrává mnohem více procesů, než si možná uvědomují. Ze schématu na obrázku 2.9 je patrný velký počet "hráčů"a procesů, které jsou zapojeny do platebního systému, a je důležité probrat roli každého z nich [60, 61]:

- **Zákazník (Customer):** Uživatel aplikace nebo platformy, který zahajuje nákup zboží nebo služby a provádí platby online pomocí různých metod, jako jsou kreditní nebo debetní karty, online peněženky, internetové bankovnictví a podobně.
- **Obchodník (Merchant):** Buď podnik nebo jednotlivec, který prodává zboží nebo služby prostřednictvím online obchodu (aplikace nebo platformy). K přijímání online plateb obchodník potřebuje mít obchodní účet - bankovní účet, který umožňuje přijímat finanční prostředky z online transakcí, protože takový účet je integrován s platební branou.
- **Platební brána (Payment Gateway):** Technologie, která funguje jako "most"mezi platformou obchodníka a zúčtovací bankou, a usnadňuje bezpečný přenos platebních údajů.

- **Zúčtovací banka nebo nabyvatel (Merchant's Acquirer):** Banka nebo jiná finanční instituce, která zpracovává platby kartou jménem obchodníka (to znamená, že účet obchodníka je uložen v bance nabyvatele). Je důležité poznamenat rozdíl mezi platební branou a nabyvatelem. Klíčový rozdíl spočívá v tom, že nabyvatel pracuje na pozadí, autorizuje transakce a komunikuje mezi různými stranami, zatímco platební brána pouze bezpečně přenáší bankovní údaje zúčtovací bance a následně zobrazuje zákazníkovi výsledek transakce. Nabyvatel obvykle nabízí své vlastní interní platební brány, ale také může spolupracovat s nezávislými branami.
- **Zpracovatel plateb (Payment Processor):** Společnost třetí strany, která poskytuje obchodníkům různé služby zpracování plateb, jako je zpracování transakcí, různá řešení pro řízení rizik a podvodů, autorizace nebo usnadňování převodu prostředků mezi účtem zákazníka a obchodníkem během procesu zúčtování. Navíc zpracovatel plateb může být použit jako nezávislá služba pro zpracování (ale to stejně neplatí pro platební bránu, ta musí být integrována s platebním procesorem), jelikož poskytuje end-to-end online transakční službu.
- **Vydavatel nebo vydávající banka (Issuing Bank):** Banka nebo jiná finanční instituce, která vydává platební karty zákazníkům jménem kartových schémat.
- **Karetní systém (Card Scheme):** Strana, která spojuje zúčtovací banku a banku zákazníka, aby tyto dvě mohly mezi sebou předávat různé informace týkající se kartových transakcí. Mezi karetní systémy například patří Mastercard, Visa nebo American Express.



Obrázek 2.9 Schema fungování platební brány[62]

Proces platby s využitím platební brány probíhá následujícím způsobem [60, 62]:

- Při placení zadá zákazník své bankovní údaje, včetně jména držitele karty, čísla karty, data expirace a kontrolního kódu (CVV), na platební stránce. Tato stránka může být buď plně hostována platební branou, nebo jsou data citlivých údajů šifrována a bezpečně předávána platební bráně.
- Platební brána šifruje údaje o kartě (nebo přijímá již šifrované údaje). Šifrování zajišťuje ochranu citlivých informací během přenosu, aby nebyly zneužity. Poté jsou tyto údaje kontrolovány na podezřelé aktivity a informace o držiteli karty a transakci jsou předány nabyvateli.

- Nabyvatel bezpečně předává tyto informace karetním systémů, které provádějí další kontrolu na podezřelé aktivity a následně jsou údaje předány vydavateli k autorizaci.
- Vydavatel provádí další opatření k ověření transakce, včetně kontroly informací o transakci, dostatečnosti finančních prostředků na účtu a platnosti bankovního účtu. Poté vydavatel zasílá nabyvateli odpověď o schválení nebo zamítnutí transakce (přes karetní systém).
- Nabyvatel informuje platební bránu a uživatele o výsledku transakce. V případě schválení je zákazník přesměrován na stránku s potvrzením platby. V opačném případě je zákazník požádán o opětovné provedení platby nebo zadání jiného platebního způsobu.
- Pokud byla platba schválena, nabyvatel zahajuje proces zúčtování: inkasuje platbu od vydavatele banky a následně ji připisuje jako "odloženou"na obchodní účet (který musí být zřízen nabyvatelem pro zpracování online transakcí pomocí platební brány). Poté nabyvatel převádí finanční prostředky z obchodního účtu na podnikatelský účet v dohodnutém čase s poskytovatelem platebních služeb (může jít o pravidelné hromadné převody nebo jednotlivé platby). Je důležité zdůraznit rozdíl mezi obchodním a podnikatelským účtem, neboť slouží různým účelům. Obchodní účet umožňuje přijímání a zpracování plateb od zákazníků, zatímco podnikatelský účet slouží k běžným výdajům obchodníka.

2.5.2 Typy platebních brán

Existuje několik typů online platebních bran, z nichž každá poskytuje specifické funkce a vyžaduje odlišný přístup k integraci [63, 64]:

Hostovaná platební brána

Hostovaná platební brána představuje řešení, kdy poskytovatel služeb třetí strany hostuje platební systém pro obchodníky. Tento systém je následně integrován do platformy a funguje následovně: jakmile je uživatel připraven provést platbu za svůj nákup, je přesměrován ze stránky pokladny aplikace na stránku hostitele nebo poskytovatele platebních služeb, kde zadá své bankovní údaje. Externí platební brána následně bezpečně zpracuje platbu na svých serverech. Jestliže platba proběhla úspěšně, uživatel je vrácen zpět na platformu a může dokončit nákup. Hlavními výhodami tohoto řešení jsou skutečnosti, že obchodník nemusí řešit zpracování platby, kontrolu nad bezpečností provedení platby ani dodržování předpisů. Tyto aspekty přebírají na sebe poskytovatele platebních služeb, který je specializovaným poskytovatelem služeb. To zaručuje vysokou dostupnost a spolehlivost. Navíc takové brány jsou snadno a rychle nastavitelné, a to bez nutnosti přímé integrace se sítí pro zpracování plateb. Obvykle postačí pouze přidání kódu nebo pluginu pro danou platformu. Díky tomu není nutné ukládat a spravovat citlivá data, což značně snižuje administrativní zátěž platformy. Z negativních aspektů lze zmínit nedostatek kontroly nad platební branou a ztrátu kontroly nad cestou kupujícího, což v některých případech může mít zásadní vliv a zvyšovat provozní a bezpečnostní rizika. Navíc je platební proces komplikovanější a trvá déle, což může být frustrující pro uživatele a odradit je od nákupu. Dalším nedostatkem je omezená možnost přizpůsobení designu a brandingu obchodníka na externě hostovaných platebních branách.

Samohostovaná nebo samoobslužně platební brána

Samohostovaná nebo samoobslužně platební brána, která umožňuje platformám udržovat plnou kontrolu nad celým platebním procesem, jelikož spravují a provádějí platební transakce v rámci své infrastruktury. Fungují tak, že když uživatel zahájí platbu, tak vlastní platební brána zpracuje transakci v rámci obchodního prostředí (bez žádného přesměrování na externí stránky) a předá šifrované platební údaje zpracovateli plateb za účelem autorizace platby. Takové platební

brány mají větší kontrolu nad zpracováním a uchováním osobních a platebních údajů a také větší kontrolu nad uživatelskou zkušeností s procesem placení a nabízejí přizpůsobení a sladění s požadavky platformy (například, lze přizpůsobit design platební brány k designu aplikace), navíc cesta kupujícího je rychlejší bez žádného přesměrování, což také zlepšuje uživatelskou zkušenost. Takový typ platební brány je vhodné použít pro větší podniky, které upřednostňují udržování kontroly nad platebními údaji a snaží se minimalizovat únik dat. Ale když informace o kartách uživatelů ukládáte na vlastních serverech, je důležité odpovídat souladu s PCI (dodržování předpisů platebních karet, který se skládá z 12 bezpečnostních standardů) a ochrany údajů, jelikož celá odpovědnost nad ochranou a ukládáním citlivých dat leží na obchodníku, a potom samohostované platební brány mohou vyžadovat větší technické znalosti a úsilí a vyšší počáteční náklady na implementaci a udržení.

Platební brána hostovaná API

Platební brána hostovaná pomocí API je brána, které zprostředkovává platby prostřednictvím aplikačního programového rozhraní (API). Toto rozhraní může být poskytnuto jak obchodníkem, tak externí službou, čímž umožňuje uživatelům zůstat po celou dobu platby na platformě obchodníka bez přesměrování na další externí stránky. API působí jako spojovací můstek mezi platformou/aplikací a infrastrukturou pro zpracování plateb. Platební brána tohoto typu nabízí plně přizpůsobitelné prostředí pokladny – například obchodník si může vybrat, které platební metody jsou přijímány, či jaká zabezpečení mají být použita – a lze ji integrovat do různých platforem. Obchodník má plnou kontrolu nad designem pokladny, uživatelským zážitkem a je zodpovědný za celý platební proces. Platební brána funguje následovně: když zákazník zahájí platbu, brána zpracuje transakci bezpečným přenosem bankovních údajů k zpracovateli plateb, autorizuje platbu a následně vrací odpověď zpět do aplikace pro potvrzení stavu transakce. Jedná se tedy o ideální řešení pro platformy, které si přejí navrhnout každý aspekt platební brány podle své vlastní podoby. Mezi výhody patří kompletní kontrola nad zadáním platebních údajů a průběhem zpracování platby, přizpůsobení požadované funkcionalitě, snadná škálovatelnost díky integraci API, široká podpora různých měn a platebních metod, což je zásadní výhoda pro globální obchodníky, a také rychlé a bezproblémové platby, při kterých uživatelé nemusí opouštět platformu obchodníka. Mezi nevýhody lze uvést požadavek na vyšší technické znalosti kvůli integraci API mezi obchodníkem a platební branou, stejně jako potřebu řešit zabezpečení citlivých dat a dodržování přísných bezpečnostních standardů.

Integrační brána místní banky

Integrační brána místní banky představuje řešení, jež umožňuje obchodníkům integrovat se s místními bankami. Tím umožňuje přesměrování dat o transakcích k nejvhodnější místní bance, která tyto platby zpracovává. Tato implementace umožňuje obchodníkům snadno zajistit platby pro různé regiony, například tam, kde jsou specifické požadavky na zpracování transakcí, a rovněž uživatelům umožňuje provádět platby pomocí preferovaných místních platebních metod. Princip fungování této brány spočívá v tom, že když uživatel zahájí platbu, integrační brána místní banky přesměruje platební údaje a informace o transakci k příslušné místní bance, která poté platbu zpracuje a odpoví. Toto řešení je vhodné pro platformy obchodující v několika zemích, jelikož mohou využít integrační bránu místní banky pro každý region. Hlavní výhodou takové brány je zvýšení důvěry uživatelů, neboť v každém regionu je využita místní banka pro zpracování platby, kterou uživatel dobře zná. Nevýhodami tohoto řešení jsou složitější integrace s rozhraními API místních bank, které mohou vyžadovat vyšší úroveň technických znalostí, a rovněž takové řešení ve většině případů obsahuje pouze základní funkce zpracování plateb. Mohou tak chybět funkce jako opakované platby nebo vrácení peněz.

2.5.3 Bezpečnostní funkce platební brány

Zajištění bezpečnosti online transakcí obchodníků je prioritním zájmem a platební brána podniká různá bezpečnostní opatření. Mezi nejčastěji využívané patří [61]:

- **Tokenizace:** Toto opatření slouží k ochraně citlivých dat, jako jsou čísla karet nebo CVV kódy. Při tokenizaci se údaje o kartě nezpracovávají přímo v jejich původní podobě, ale jsou nejprve převedeny na síťové tokeny - unikátní digitální identifikátory, které slouží k dodání tokenizovaných hodnot místo citlivých dat. Takto zabezpečené informace nelze využít v případě útoku na data.
- **Prevence podvodů:** Platební brány využívají různorodé nástroje a funkce pro detekci podvodů, které analyzují vzory a chování transakcí v reálném čase. To umožňuje identifikovat a případně zabránit podvodům.
- **Peněženky dle standardu PCI DSS:** Jedná se o peněženky, které splňují standardy PCI DSS (Payment Card Industry Data Security Standard) - což jsou normy pro zabezpečení dat platebních karet vyvinuté Radou pro bezpečnostní standardy platebních karet, zřízenou mezinárodními platebními systémy Visa, MasterCard, American Express, JCB a Discover. Tyto peněženky zaručují bezpečné uložení platebních údajů držitelů karet pro opakované transakce.
- **White Label Wallet:** Tato bezpečnostní funkce je především určena pro mobilní peněženky. Většina platebních bran nabízí integrace white-label pro různé platební metody prostřednictvím mobilních peněženek, čímž se zajišťuje bezpečnější a pohodlnější provedení transakcí, jež jsou rovněž snadněji použitelné.
- **3D Secure autentikace:** Toto opatření znamená, že před dokončením transakce existuje ještě jedna vrstva zabezpečení. Obvykle to funguje tak, že uživatel obdrží jednorázový ověřovací kód na mobilní telefon nebo email, který zadá pro potvrzení platby.

2.6 Funkční a nefunkční požadavky

Tato sekce se zabývá analýzou funkčních a nefunkčních požadavků pro každý modul. Požadavky byly vytvořeny na základě provedené analýzy a dále budou použity pro tvorbu návrhu.

2.6.1 Požadavky pro sekce virtuálních bodů

Tato podsekce obsahuje popis funkčních a nefunkčních požadavků pro sekci ”Virtuální body”.

F1.1 Zobrazení profilu uživatele a jeho odznaků

- Zobrazení jednoduchého profilu uživatele: profilová fotografie a jméno. Zobrazení získaných odznaků a aktuálního počtu virtuálních bodů.

F1.2 Zobrazení přehledu odznaků

- Zobrazení kompletního přehledu odznaků: popis všech odznaků, jak je získat a kolik bodů za to uživatel může dostat. Zobrazení aktuálně získaných odznaků pro každý typ.

F1.3 Zobrazení tabulky všech profilů

- Zobrazení seřazené tabulky všech uživatelů závislé na celkovém počtu virtuálních bodů a počtu získaných odznaků.

F1.4 Zobrazení přehledu virtuálních bodů

- Zobrazení informace o virtuálních bodech: co to je, jak se sbírají, princip jejich utrácení apod. Zobrazení aktuálního počtu virtuálních bodů a aktuálního počtu aktivních odměn.

F1.5 Zobrazení historie použití virtuálních bodů

- Zobrazení historie použití virtuálních bodů: kdy a kolik bylo utraceno, za jakou odměnu, zda byla odměna již použita apod.

F1.6 Zobrazení možných odměn

- Zobrazení odměn, které uživatel může aktivovat, rozdělené do různých kategorií: novinky, vstupenky, doporučené apod. Zobrazení již aktivovaných odměn, a pokud nejsou vypršelé, zobrazení QR kódu nebo promo kódu pro jejich použití.

F1.7 Aktivování odměn

- Uživatel může aktivovat vybranou odměnu a tato odměna se přemístí do složky aktivních. Uživatel může rovnou zobrazit QR kód nebo promo kód aktivované odměny.

NF1.1 Implementace pro Flutter verzi 3.19

- Pro vývoj bude použit framework Flutter, verze 3.19, spolu s vývojovým jazykem Dart.

NF1.2 Implementace pro Android a iOS platformy

- Modul bude implementován pro dvě platformy: Android a iOS. I když framework Flutter podporuje i další platformy, ostatní však nebudou testovány.

NF1.3 Využití Firestore databáze

- Využití Firestore databáze pro ukládání a načítání virtuálních bodů uživatelů a také pro ukládání a načítání odměn, včetně aktivovaných a jejich historie.

2.6.2 Požadavky pro fórům fanoušků

Cílem této podsekce je sepsání funkčních a nefunkčních požadavků pro modul ”Fórum fanoušků”.

F2.1 Posílání textových zpráv

- Posílání jednoduchých textových zpráv.

F2.2 Úprava poslaných textových zpráv

- Možnost editace již odeslané zprávy nebo její úplné smazání.

F2.3 Zobrazení informací o poslané zprávě

- Zobrazení informací o odesílateli zprávy a samotné zprávy: profilová fotografie, uživatelské jméno a čas odeslání.

F2.4 Posílání obrázků

- Možnost posílání na fórum obrázků, které uživatel může nahrát z galerie (maximálně jeden pro jednu zprávu).

F2.5 Zobrazení zásad a pravidel k používání fóra

- Možnost prohlížení zásad a pravidel k používání platformy, to znamená, co lze a co nelze posílat a jaké sankce budou použity pro ty, kdo ta pravidla poruší.

NF2.1 Implementace pro Flutter verzi 3.19

- Pro vývoj bude použit framework Flutter, verze 3.19, spolu s vývojovým jazykem Dart.

NF2.2 Implementace pro Android a iOS platformy

- Modul bude implementován pro dvě platformy: Android a iOS. I když framework Flutter podporuje i další platformy, ostatní však nebudou testovány.

NF2.3 Moderování textového obsahu na předmět vulgarismů

- Moderování textového obsahu na předmět použití zakázaných nebo vulgárních výrazů. Do budoucna je třeba implementovat i moderování obsahu obrázků, ale tato diplomová práce se jím nezabývá.

NF2.4 Využití Firestore databáze

- Použití Firestore databáze pro ukládání zpráv a jejich načítání.

2.6.3 Požadavky pro modul nákupu vstupenek

Tato sekce se zabývá popisem a formulací funkčních a nefunkčních požadavků pro sekci ”Nákup a správa vstupenek”aplikace Eliterro.

F3.1 Nákup vstupenek

- Umožnění nákupu vstupenek na vybraný zápas pomocí různých platebních metod přímo v aplikaci. Uživatel může zakoupit více vstupenek najednou na tentýž zápas.

F3.2 Filtrování dostupných zápasů

- Možnost filtrování dostupných zápasů (na které lze zakoupit vstupenky) podle ligy a týmu-účastníka zápasu.

F3.3 Správa již zakoupených vstupenek

- Prohlížení již zakoupených vstupenek (hrající týmy, datum, čas a místo konání), možnost zobrazit QR kód zakoupené vstupenky nebo jej poslat na uvedený e-mail.

F3.4 Prohlížení již prošlých zakoupených vstupenek

- Prohlížení již prošlých zakoupených vstupenek. Musí se lišit od ještě aktivních, aby uživatel mohl je jednoduše rozlišit.

NF3.1 Implementace pro Flutter verzi 3.19

- Pro vývoj bude použit framework Flutter, verze 3.19, spolu s vývojovým jazykem Dart.

NF3.2 Implementace pro Android a iOS platformy

- Modul bude implementován pro dvě platformy: Android a iOS. I když framework Flutter podporuje i další platformy, ostatní však nebudou testovány.

NF3.3 Využití platební brány pro zpracování platby

- Pro zpracování platebních transakcí bude použit balíček, který podporuje zpracování plateb a obsahuje platební bránu.

NF3.4 Využití Firestore databáze

- Pro ukládání a zobrazení zakoupených vstupenek bude použita Firestore databáze. Informace o dostupných zápasech také budou brány z Firestore databáze.

2.7 Závěr analýzy

Analýza diplomové práce se zabývala popisem frameworku Flutter, popisem aplikace Eliterro a studiem problematiky pro každou sekci, která bude implementována v této diplomové práci. Pro sekci ”Virtuální body”byla provedena studie s cílem pochopit termín gamifikace, porozumět tomuto konceptu z psychologického hlediska, zkoumat psychologii a četnost získávání odměn. Analýza problematiky modulu ”Fórum fanoušků”je primárně zaměřena na studium různých typů moderování obsahu, oblastí, kde lze moderování využít, a specifikaci automatizovaného moderování. Pro modul ”Nákup vstupenek”byla provedena studie ohledně pojmu platební brána: její definice, role v zpracování online plateb, typy platebních bran a analýza bezpečnostních funkcí.

Všechny získané poznatky z výzkumu budou následně využity při návrhu a implementaci těchto tří modulů. Navíc, díky výzkumu, vznikly funkční i nefunkční požadavky, které budou použity při návrhu prototypu.

Kapitola 3
Návrh

Tato kapitola se věnuje návrhu prototypu aplikace, který zahrnuje moduly „Nákup a správa vstupenek“, „Virtuální body“ a „Fórum fanoušků“ na základě dříve zmíněných funkčních a nefunkčních požadavků.

Vzhledem k tomu, že výsledkem této diplomové práce je aplikace obsahující všechny tři klíčové moduly, bude v návrhu nejprve představena technologie použitá napříč celou aplikací, jako jsou například nastroj Firebase nebo Riverpod. Následně se návrh zaměří na konkrétní sekce, kde budou diskutovány specifika jednotlivých sekci, představeny vybrané řešení problémů a vytvořena uživatelská rozhraní pro dané sekce. V závěru bude formulováno shrnutí a vytvořen konceptuální model návrhu aplikace.

3.1 Použité technologie

V této části jsou představeny technologie, které se využijí pro vývoj výsledné aplikace a které najdou uplatnění napříč celým projektem. Jednou z klíčových technologií je Riverpod, knihovna pro správu stavu ve Flutter aplikacích. Ačkoliv existují další knihovny pro správu stavu, jako je Bloc nebo Provider, pro tuto aplikaci byl zvolen Riverpod. Tato volba byla učiněna z důvodu jeho již existující integrace do aplikace Eliterro. Použití Riverpodu je vhodné, jelikož to v budoucnu usnadní přidávání sekcí implementovaných v rámci této diplomové práce do reálného projektu.

Další významnou technologií jsou Firebase služby - soubor cloudových backendových služeb a platforem od společnosti Google, navržených pro vývoj aplikací na různých platformách. Pro tento projekt byl Firebase zvolen jako dočasné řešení. Plánován je totiž přechod na vlastní backendový server, který je již ve vývoji, ale zatím neposkytuje všechny nezbytné endpointy pro plnou funkčnost aplikace. Volba Firebase padla kvůli několika jeho výhodám, jako je jednoduchost nastavení, bezplatný základní tarif a nabídka kompletních technologických řešení potřebných pro vývoj mobilních aplikací, včetně autentizace a datových úložišť, což z něj činí ideální řešení pro vývoj menších aplikací, které zatím nevlastní kompletní backend.

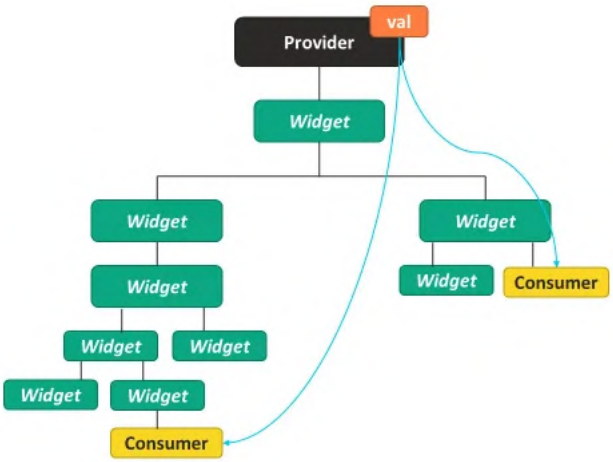
3.1.1 Riverpod

Správa stavu v aplikacích Flutter představuje klíčový prvek pro tvorbu odolných, škálovatelných a kvalitních aplikací. Flutter přímo nabízí tradiční metody pro správu stavu, jako jsou funkce setState, které umožňují překreslení widgetů v reakci na změny stavu a tím aktualizaci jejich vizuální prezentace, a InheritedWidget, což jsou widgety umožňující datový tok ve stromu widgetů, čímž zpřístupňují sdílená data ostatním widgetům ve stromu a umožňují jejich automa-

tické překreslení při změně těchto dat. Tyto základní techniky však mohou být omezené, zejména v případě větších a složitějších aplikací.

V reakci na tyto omezení vyvinula komunita Flutter různé sofistikovanější přístupy ke správě stavu, mezi které patří knihovna Riverpod. Riverpod staví na knihovně Provider, ale nabízí pokročilejší, flexibilnější a jemnější metody pro manipulaci se stavem aplikace.

Knihovna Provider, která je základem pro Riverpod, využívá třídy poskytovatelů (providers) a spotřebitelů (consumers) pro efektivní správu stavu. Provider skrývá stavové části aplikace a poskytuje je tam, kde jsou potřebné pro uživatelské rozhraní. Widgety závislé na stavu mohou sledovat změny ve stavu prostřednictvím Provideru a reagovat na ně obnovou a přestavbou, díky spotřebiteli (Consumer), který navíc umožňuje aktualizaci proměnných v Provideru. Zjednodušeně řečeno, balíček Provider rozšiřuje možnosti InheritedWidgetu tím, že usnadňuje aktualizace a konzumaci dat a umožňuje předávání stavu nebo jakýchkoliv sdílených dat napříč různými obrazovkami a widgety ve stromu widgetů, eliminující potřebu opětovného sdílení těchto dat při každém přechodu mezi nimi. Na obrázku 3.1 je ilustrován princip fungování Provideru, kde Provider funguje jako rodič všech ostatních widgetů, sdílí informace napříč všemi svými potomky, které to potřebují, a některé z těchto widgetů-potomků obsahují Consumer, který umožňuje aktualizovat proměnnou z Provideru. [65]



Obrázek 3.1 Graf fungování Provideru[65]

Riverpod přináší v porovnání s obvyklým Providerem několik zlepšení a změn [66, 67]:

- Na rozdíl od standardního Provideru, který vyžaduje obalení všech potomků do Provideru pro přístup k stavu a datům, Riverpod poskytuje providery deklarované globálně. Toto uspořádání umožňuje, že stav a logika mohou být udržovány nezávisle na stromu widgetů, což zajišťuje přístup k datům providera bez ohledu na jeho umístění ve stromu.
- Obvyklý provider je jednoduchý objekt, který obsahuje část stavu a umožňuje na něm naslouchat změnám. Riverpod provider naopak nabízí širší spektrum funkcí. Kromě uchování a přístupu ke stavu na různých místech (podobně jako běžný provider) také nahrazuje některé návrhové vzory, jako jsou singletony, dependency injections nebo služby lokátoru. Dále optimalizuje výkon tím, že minimalizuje opakované vybudování widgetů a ukládá výpočty stavu do mezipaměti, což zjednodušuje i proces testování. Každý provider lze během testování přepsat, aby se choval odlišně.
- Ve Flutteru je každému widgetu přiřazen objekt BuildContext, který slouží k přístupu k prvkům v rámci stromu widgetů. Na rozdíl od toho, provider z knihovny Riverpod se nachází

mimo tento strom, a proto pro jeho načítání využíváme objekt WidgetRef. WidgetRef umožňuje widgetům interakci s providery a poskytuje přístup k libovolnému provideru v aplikaci. Tento koncept je podobný objektu BuildContext, který umožňuje přístup k nadřazeným widgetům ve stromu prostřednictvím kontextu. Pro získání WidgetRef ve widgetu existují tři metody. První z nich spočívá v použití ConsumerWidget místo obvyklého StatelessWidget. Metoda build() ConsumerWidgetu tak získává nový parametr - objekt WidgetRef, který umožňuje sledování změn providera. Druhý způsob využívá zabalení widgetu do komponenty Consumer, která poskytuje builder s parametrem typu WidgetRef. Tento přístup je vhodný pro velké widgety, přičemž pokud je třeba sledovat stav pouze u malé části widgetu (například u jednoho textu), je efektivnější zabalení jen této menší části do Consumeru. Tím dojde k překreslení pouze malé části widgetu při změnách hodnot providera, což je výkonnější než použití ConsumerWidget pro celý widget. Poslední metodou je použití ConsumerStatefulWidget v případě, že potřebujeme přístup k providerům ve Stateful widgetu, který může dynamicky měnit svůj stav během běhu aplikace. Využití ConsumerStatefulWidget umožňuje přístup k referenci (WidgetRef) jak v metodě build() prostřednictvím ref.watch(), tak i v ostatních metodách životního cyklu widgetu prostřednictvím ref.read(). Toto je podobné deklaraci BuildContextu ve třídě Flutter State, který je dostupný přímo ze všech metod životního cyklu widgetu.

Riverpod poskytuje sedm různých typů poskytovatelů (providerů), z nichž každý je určen pro specifický případ užití [67, 68]:

- **Provider** - tento základní typ poskytovatele je nejčastěji využíván pro přístup k neměnným objektům a závislostem, jako jsou například úložiště, logovací služby nebo libovolné hodnoty, které neobsahují proměnlivý stav.
- **StateProvider** - tento poskytovatel se využívá pro správu a poskytování jednoduchých stavových objektů, které mohou podléhat změnám, například hodnota čítače nebo booleovská hodnota pro zaškrtávací políčka. Kdykoliv je tato hodnota sledována v metodě build(), widget se obnoví s každou její změnou.
- **StateNotifierProvider** - využívá se pro sledování a reakci na změny prostřednictvím StateNotifier, což je třída pozorovatel, která uchovává jeden neměnný stav. Tento poskytovatel je ideální pro správu stavů, které se mohou měnit v reakci na uživatelské interakce, jako je stisk tlačítka.
- **FutureProvider** - je variantou poskytovatele určenou pro asynchronní kódování. Jeho účelem je poskytovat výsledky operací, které vrací typ Future. Future je objekt představující hodnotu nebo chybu, která se objeví v budoucnu, a je typicky používán pro asynchronní úkony, jako je načítání dat nebo zpracování rozsáhlých výpočtů. Tento poskytovatel je využíván například pro vykonávání a uchovávání výsledků asynchronních operací, jako jsou volání API, stejně jako pro řešení chybových stavů nebo stavů při načítání a obnovování dat.
- **StreamProvider** - slouží pro sledování výsledků ze Streamu, což je asynchronní posloupnost datových událostí, jež lze postupně naslouchat a zpracovávat. Podobá se FutureProvideru, ale místo Future naslouchá Streamu. Je užitečný pro sledování například Firebase nebo webových socketů.
- **ChangeNotifierProvider** - tento poskytovatel slouží pro naslouchání a reagování na změny prostřednictvím ChangeNotifier, jednoduché třídy používané k upozorňování na změny v datech. Riverpod však nedoporučuje jeho používání, protože tento poskytovatel byl primárně navržen pro snadný přechod z klasického Provideru na Riverpod.
- **(Async)NotifierProvider** - je navržen pro naslouchání a odhalování změn prostřednictvím (Async)Notifier, komponenty, která aktualizuje posluchače při změně. Tento poskytovatel je

doporučeným řešením od Riverpodu pro správu stavů, které se mohou změnit v důsledku interakce s uživatelem a může také sloužit pro centralizaci logiky úpravy stavů.

Riverpod bude tedy využit pro řízení stavu ve výsledné aplikaci, což umožní například získání globálních hodnot napříč různými obrazovkami a widgety, jako je uživatelské jméno nebo ID uživatele, nebo pro získání a ukládání dat z/do databáze.

3.1.2 Nastroj Firebase

Firebase je kolekce cloudových vývojových nástrojů od společnosti Google, které usnadňují vývojářům práci na mobilních a webových aplikacích tím, že jim poskytují prostředky pro tvorbu, nasazení a škálování aplikací. Sada Firebase zjednodušuje vývojový proces nabídkou backendových nástrojů a služeb. [69]

Mezi nástroje, které Firebase nabízí, patří autentizační systémy, databáze v reálném čase, cloudové zasílání zpráv, sledování chyb, analýza výkonnosti a testovací laboratoře.

Ve výsledné aplikaci této diplomové práce budou použity dva nástroje: autentizace, která zajistí bezpečný a snadný způsob přihlášení do aplikace, a databáze v reálném čase, do které se budou ukládat všechny nezbytné data pro běh aplikace. Je třeba zdůraznit, že použití nástrojů Firebase je dočasným řešením, určeným pouze pro implementaci aplikace v rámci této diplomové práce. Pro celkovou aplikaci Eliterro se připravuje vlastní backendový server, který se bude starat o všechny tyto služby, a při migraci této ukázkové aplikace do aplikace Eliterro bude použití nástrojů Firebase nahrazeno vlastním API. Před začleněním jakékoliv služby Firebase do Flutter aplikace je nezbytné přidat Firebase do celého projektu, což zahrnuje inicializaci Firebase v dané aplikaci (pomocí přidání balíčku *firebase_core*) a následné aktivování a konfigurování FlutterFire CLI – nástroje příkazové řádky, který pomáhá efektivně konfigurovat a spravovat služby Firebase a automaticky dokončí všechna potřebná nastavení nutná pro konfiguraci Firebase služeb ve Flutter aplikaci. [70]

Firebase Authentication představuje komponentu platformy Firebase, která nabízí bezpečný a efektivní způsob, jak řídit autentizaci uživatelů. Tato služba podporuje různé metody přihlášení, včetně emailu a hesla, ověřování prostřednictvím telefonního čísla, nebo prostřednictvím externích poskytovatelů jako jsou Google nebo Facebook. Integrace této služby do aplikace či webové stránky je přímočará a zahrnuje kompletní správu bezpečnostních aspektů autentizace. Dále, autentikační služba umožňuje snadnou integraci s dalšími službami Firebase, což přispívá k robustnosti celé aplikace. [71] Pro vývojáře pracující s Flutterem je k dispozici balíček *firebase_auth* [72], který umožňuje interakci s backendovou částí této služby a jednoduché přidání požadovaných autentizačních metod.

Firestore je cloudová NoSQL databáze pracující v reálném čase, která umožňuje ukládání a synchronizaci dat napříč různými zařízeními v reálném čase, čímž zajišťuje neustálou aktuálnost aplikace. Je navržena tak, aby byla flexibilní a snadno škálovatelná. Data jsou ukládána ve formě dokumentů, které jsou organizovány do kolekcí. Dokumenty mohou obsahovat různé datové typy (čísla, řetězce, booleovské hodnoty, data atd.) a jsou schopné ukládat i komplexní objekty. Databáze rovněž nabízí rozsáhlé možnosti dotazování, podporu offline přístupu k datům a jednoduchou integraci s ostatními Firebase službami. [73] Pro použití této databáze ve Flutter aplikacích stačí přidat balíček *cloud_firestore* [74] do závislostí projektu, který umožňuje provádění CRUD(create, read, update, delete) operací a synchronizaci dat v reálném čase skrze posluchače reagující na změny dat.

3.2 Virtuální body

Hlavním cílem sekce ”Virtuální body a odměny”je správné navržení způsobů, jak mohou uživatelé získávat body včetně frekvence jejich přidělování, a vytvoření systému, pro co budou tyto body uživateli sloužit. Cílem je, aby tyto prvky odpovídaly zaměření aplikace Eliterro a motivovaly

uživatele k častějšímu využívání aplikace. Nejprve byly na základě prováděného průzkumu a analýzy různých aplikací využívajících prvky gamifikace navrženy různorodé metody získávání bodů a odměn. Tyto metody byly zhodnoceny s ohledem na jejich výhody a nevýhody. Následně byly pomocí brainstormingu v pracovním týmu vybrány nejvhodnější z nich, které budou implementovány do aplikace, a bylo navrženo uživatelské rozhraní.

3.2.1 Jak body získávat?

Možné způsoby získávání virtuálních bodů vhodných pro použití v aplikaci Eliterro jsou následující:

- Sdílení informací (osobní úspěchy, zápasové statistiky, získávání odměn a odznaků apod.) na sociálních sítích za co uživatel získává body. Výhody této metody gamifikace zahrnují zvýšení viditelnosti a dosahu aplikace – jelikož uživatelé sdílí obrazovky z aplikace mezi své přátele na sociálních sítích, viditelnost a dosah aplikace se zvyšují; organický růst, což znamená, že noví uživatelé jsou přilákáni na doporučení svých přátel, což má větší sílu důvěryhodnosti než tradiční marketing; navíc aplikace může získávat dodatečnou zpětnou vazbu od uživatelů sociálních sítí, kteří komentují sdílené informace z aplikace. Mezi nevýhody patří možnost negativního vnímání, pokud se uživatelé cítí příliš tlačeni ke sdílení obsahu; a nadměrné spamování uživatelů na sociálních sítích může odradit potenciální nové uživatele od používání této aplikace. Tento způsob získávání bodů je běžnou praxí nejen v herních aplikacích, ale také například ve fitness aplikacích, kde uživatelé mohou sdílet své úspěchy, splněné cíle nebo statistiky, nebo také v aplikacích pro čtení knih – uživatelé mohou sdílet knihy, které čtou, recenze na přečtené knihy nebo také například statistiky, kolik knih přečetli za rok apod.
- Doporučení aplikace přátelům – uživatel může získat virtuální body za každého zaregistrovaného kamaráda, který aplikaci používá. Mezi klady patří virální růst – můžeme očekávat, že každý pozvaný kamarád pozve další, což může vést k exponenciálnímu počtu nových uživatelů; jde o marketing s minimálními výdaji, jelikož využívá stávající uživatele k přilákání nových; větší zapojení uživatelů – když přátelé využívají stejnou aplikaci, s větší pravděpodobností ji budou nadále používat. Mezi nevýhody patří obtěžování přátel spamováním pozvánek, což může vést k odrazení těchto přátel od vyzkoušení a používání aplikace; uživatelé se mohou zajímat pouze o odměny, nikoli o dlouhodobé používání aplikace, což pak může vést k nízkému počtu interakcí a zapojení uživatelů. Příklady aplikací, které používají takovou metodu, zahrnují různé finanční aplikace (například PayPal, Revolut), které nabízejí peněžní odměny za doporučení přátelům, nebo zábavní aplikace, které motivují uživatele pozvat své přátele ke získání herních odměn nebo nových úrovní.
- Každodenní aktivita, kde uživatel získává body za používání aplikace každý den, přičemž čím déle ji používá, tím více bodů získává. Tato metoda zvyšuje zapojení uživatelů, jelikož aplikaci používají každodenně; a vytváří návyk používání aplikace každý den. Nicméně tato metoda má i negativní stránky, jako jsou důraz na kvantitativní metriky – uživatel používá aplikaci každý den pouze pro získání bodů a ne proto, že by se v ní zajímal; vyčerpání uživatelů a demotivace z vynechání denní aktivity – tlak na udržování denní aktivity, monotónní úkoly a penalizace za zapomenutí denní aktivity, které vedou k ztrátě progrese, to vše může vést k demotivaci a odradit uživatele od aplikace; a snížená vnitřní motivace uživatelů, což znamená, že uživatel se především zabývá získáváním bodů, místo aby aplikaci využíval pro osobní cíle a motivaci. Příklady aplikací, které mohou použít takovou metodu získání bodů, zahrnují vzdělávací aplikace, kde uživatel dostává body za každodenní učení, nebo aplikace pro zdravý životní styl, kde uživatel dostává odměny za splnění každodenních cílů (například splnění počtu denních kroků nebo každodenní sportovní aktivita).
- Získávání virtuálních bodů za nákup vstupenek na zápasy prostřednictvím aplikací lze přirovnat k cashback systému, avšak ve formě virtuálních bodů. Tato metoda nabízí řadu možností pro marketingové aktivity, zejména pro propagační kampaně, jako jsou bonusové body za

první nákup nebo získání dvojnásobku bodů v určité dny. Mezi její pozitiva patří zvyšování věrnosti zákazníků, kdy získávání bodů za nákupy podněcuje zákazníka k dalším nákupům ve stejné aplikaci za účelem získání dalších bodů, a rovněž podpora prodeje vstupenek, neboť odměna za nákup motivuje uživatele k častějším nákupům. Mezi nevýhody patří vnímání virtuálních bodů jako méně hodnotných a možnost jejich použití v omezenějších podmínkách ve srovnání se skutečnými peněžními cashbacky, což může snížit jejich atraktivitu pro některé uživatele. Další potenciální nevýhodou je, že uživatel může platformu vnímat jako méně seriózní a profesionální kvůli využití různých gamifikačních prvků a může se obávat poskytnutí citlivých bankovních údajů, což by ho mohlo vést k přímému nákupu od ověřeného prodejce. Tato metoda se hodí do věrnostních aplikací pro různé obchodníky, kde mohou zákazníci získávat body za nákupy, které lze následně vyměnit za produkty nebo slevy.

- Denní kvíz, kde uživatel obdrží jednou denně náhodně generovanou otázku (přichází mu notifikace) a v případě správné odpovědi získá body. Čím déle uživatel odpovídá správně, tím více bodů získává. Tato metoda podporuje denní interakci uživatelů s aplikací, podněcuje zvykové chování a přispívá k rozšiřování vědomostí uživatelů. Mezi slabiny patří únava uživatelů z každodenních otázek (pokud nejsou dostatečně rozmanité a zajímavé), potenciál pro podvádění – uživatelé mohou hledat odpovědi na internetu a získávat body „zadarmo“. Navíc je stálé udržování čerstvých, zajímavých a kvalitních otázek finančně náročné. Tato metoda se hodí zejména do vzdělávacích aplikací nebo aplikací na trénink mozku a logiky, kde uživatelé odpovídají na otázky týkající se naučeného materiálu a tím si procvičují své znalosti, nebo do různých herních aplikací založených na kvízech.

- Tipování výsledků zápasů umožňuje uživatelům předpovídat výsledky a při správném tipu získat body. Tento systém motivuje uživatele k častějšímu sledování sportovních událostí prostřednictvím dané aplikace, jelikož mohou na konci zápasu získat odměnu. Zároveň podporuje návratnost uživatelů do aplikace za účelem tipování skóre a kontrolu výsledků, a posiluje komunitu, která o tipech a výsledcích diskutuje. Avšak metoda má i své nevýhody, jako jsou složitost vývoje a údržby, nutnost udržování spravedlnosti, riziko připomínání hazardních her a potenciální závislost. Tento přístup nachází uplatnění nejen v sportovních aplikacích, ale i ve finančních, kde uživatelé mohou předpovídat tržní trendy nebo ceny akcií, či v zábavních aplikacích, kde mohou uživatelé spekulovat o ději v televizních pořadech nebo knihách.

- Získávání různých úrovní ocenění a odznaků za opakované plnění různých úkolů a dosažení nové úrovně odznaků umožňuje uživatelům získávat virtuální body. Tato metoda motivuje uživatele k účasti na různých úkolech, přičemž je klíčové navrhnout a poskytnout pro uživatele srozumitelný systém odměn. Jasné cíle umožňují uživatelům vědět, co je potřeba udělat pro získání konkrétní odměny. Navíc odznaky lze často považovat za odměny, které se uživatelé mohou pochlubit před ostatními nebo sdílet na sociálních platformách. Slabé stránky této metody zahrnují nadměrný důraz na odznaky a ocenění, který může vést k zaměření se spíše na jejich získávání než na samotný obsah nebo aktivitu. Pokud je systém nastaven tak, že odznaky jsou snadno dostupné, vzrušení ze získání rychle klesá a ztrácí se zájem. To může vést k negativním zkušenostem a demotivaci uživatelů, kteří se mohou začít srovnávat s ostatními. Technologie ocenění se často využívá v různorodých aplikacích, jako jsou produktivní nástroje nebo fóra, kde uživatelé získávají ocenění za odpovědi na otázky a zapojení do komunity.

3.2.2 K čemu body využívat?

Protože hlavní odměnou pro uživatele jsou virtuální body, je nutné specifikovat, na co je mohou využít a co je motivuje k jejich sbírání:

- Body lze využít k nákupu různých odznaků, speciálních karet nebo ikon, které jsou následně viditelné pro ostatní uživatele. Tato forma odměny umožňuje uživatelům vyjádřit svou identitu

v rámci aplikace a stává se viditelným úspěchem pro ostatní. Může to vést k zvýšené sociální interakci a konkurenci, což na jednu stranu může zvyšovat zájem o aplikaci, na druhou stranu však může některé uživatele od používání aplikace odradit. Proto je zásadní stanovit správnou cenu pro odměny tohoto typu, aby podněcovaly zdravou konkurenci. Když uživatelé investují do takových odměn, je pravděpodobnější, že v aplikaci setrvají déle, protože nechtějí přijít o svůj dosažený pokrok.

- Další možností je utraceni bodů za odemknutí nových funkcí aplikace nebo zisk speciálního designu, například customizovaných avatarů. Taková odměna podporuje větší zapojení uživatelů a jejich loajalitu, stejně jako vytváří konkurenci a interakci mezi uživateli (podobně jako předchozí typ odměny). Navíc takový specificky navržený design může odlišit aplikaci od konkurence a přilákat nové uživatele, kteří hledají interaktivní a obohacující uživatelský zážitek. Mezi nevýhody takové odměny patří složitost implementace a designování, protože je nutné vytvořit široký spektrum designů a funkcí, které vyhovují potřebám co nejvíce uživatelů. Tato forma odměny je běžně využívána v různých aplikacích a obvykle souvisí s úpravami vzhledu, oblečení apod. maskota aplikace nebo avatara uživatele podle preferencí uživatele, které jsou poté viditelné ostatním.

- Slevy na vstupenky na zápasy nebo kupóny a promo kódy do sportovních obchodů (například sleva v Decathlonu). Tyto odměny zvyšují loajalitu zákazníků, podporují opakované používání aplikace, zvyšují zapojení uživatelů a jsou velmi atraktivní pro nové uživatele. Tato metoda je velmi efektivní, neboť odměny jsou přizpůsobeny zájmům uživatelů o sport. Existuje široké spektrum aplikací, které využívají kupóny a slevy jako primární odměny, jelikož jsou považovány za velmi lákavé a účinné.

- Zařazení do losování větší soutěže – uživatel může za virtuální body zakoupit místo v loterii, kde se losuje o větší a cennější ceny. Tato odměna zvyšuje vnímanou hodnotu používání aplikace, jelikož jde o větší ceny. Navíc může sloužit jako účinný marketingový nástroj: vítěz sdílí své zkušenosti a příběh o úspěchu na sociálních sítích, a tím láká své přátele k vyzkoušení aplikace. Existují však jistá rizika: riziko zklamání, pokud uživatel nevyhrává, což může od používání aplikace odradit; nevhodné řízení právních záležitostí spojených se soutěžemi a loteriemi. Tento druh odměny se často používá ve věrnostních aplikacích, které poskytují loterie a speciální soutěže pouze pro členy věrnostního programu.

- Voucher na něco velmi neobvyklého a vzácného, jako například setkání s oblíbeným hráčem, účast v tréninku oblíbeného týmu nebo tričko s podpisem. Takové spojení se známými sportovci může zlepšit celkový "brand"aplikace, odlišit ji od konkurence, a neopakovatelné zážitky uživatelů vylepší jejich věrnost k aplikaci, zvýší jejich zapojení a zvýší šance, že aplikaci doporučí přátelům buď osobně, nebo přes sociální sítě. Na druhou stranu taková odměna vyžaduje vysoké náklady na spolupráci se známými sportovci a je špatně skalovatelná, což znamená, že není možné nabídnout každému uživateli takovou exkluzivní odměnu, pokud je uživatelů mnoho.

Ačkoliv hlavní odměnou v aplikaci jsou virtuální body, které si uživatelé mohou vyměňovat za různé odměny, je důležité zmínit další dva typy odměn, které mohou být také vhodné pro použití v aplikaci Eliterro:

- Přehled všech uživatelů, kde si jednotlivci mohou porovnávat získané body s ostatními. Tato funkce sice není přímo spojena s možností utrácení bodů, ale působí jako interní uznání za nasbírané body a může motivovat uživatele k častějšímu a delšímu zapojení do aplikace. Žebříčky také podporují budování komunity, protože uživatelé mohou pociťovat sdílené cíle a cítit se jako součást větší skupiny. Konkurenční prostředí může zároveň zvyšovat motivaci, avšak je třeba dbát na to, aby aplikace nevytvářela příliš soutěživé nebo toxické prostředí.

- Náhodné odměny, které mohou být implementovány prostřednictvím ”kola štěstí”, jež se objevuje uživatelům náhodně a překvapivě. Uživatelé předem nevědí, jakou odměnu obdrží – může to být jak nepatrný počet bodů, tak i něco exkluzivního. Tento typ odměn zvyšuje angažovanost uživatelů, povzbuzuje je k častějšímu používání aplikace díky nepředvídatelnosti odměn a možnosti získat cenné ceny, což zvyšuje vzrušení a motivuje k návratům. Nicméně je třeba být opatrný, protože někteří uživatelé mohou považovat takové odměny za frustrující nebo manipulativní, zvláště pokud často získávají odměny nízké hodnoty. Také může být toto kolo vnímáno jako hazardní hra, což by mohlo vést k obavám z možné závislosti a následnému upuštění od používání aplikace. Tento typ odměny je často používán v herních aplikacích, ale lze se s ním setkat i v nákupních či vzdělávacích aplikacích.

3.2.3 Vybrané metody získávaní bodů a vybrané odměny

Nakonec se rozhodlo, že aplikace nebude přeplněna různými prvky gamifikace, aby si uchovala svůj hlavní účel. Proto byla zvolena jediná hlavní metoda získávání virtuálních bodů, která spojila většinu návrhů. Uživatelé získávají odznaky za plnění různých úkolů, jaké byly dříve navržené. Tyto odznaky mají šest úrovní, přičemž každá další úroveň umožňuje získat více bodů. Úkoly pro dosažení konkrétní úrovně jsou detailně popsány a jsou navrženy tak, aby nebylo možné je splnit v krátkém čase. Dosažení nejvyšší úrovně je téměř nemožné. Shrnutí všech úrovní, přibližný čas na jejich dosažení a počet bodů za dosažení konkrétní úrovně je uveden v tabulce 3.1. Všechny úrovně jsou stejné pro všechny odznaky a liší se pouze úkolem, který je potřeba splnit pro jejich získání.

Úroveň	Čas dosazeni	Počet získaných bodů
Bramborová medaile	den	10 bodů
Bronzová medaile	měsíc	20 bodů
Stříbrná medaile	3-6 měsíců	50 bodů
Zlatá medaile	1 rok	100 bodů
Patina	2 roky	200 bodů
Diamant	3-5 let	500 bodů

Tabulka 3.1 Shrnutí úrovní odznaků

Tabulka 3.2 popisuje každý z šesti navržených odznaků a výhody spojené s nimi. Uživatelé navíc budou moci sledovat svůj pokrok, zobrazit si, na jaké úrovni se nachází, co mají udělat a kolik jim zbývá do dosažení další úrovně. Součástí aplikace bude také tabulka všech uživatelů, z níž bude zřejmé, jaké odznaky a na jaké úrovni má který uživatel, a jaké má postavení mezi všemi uživateli.

V budoucnu se plánuje zavést další metodu získávání bodů, a to cashbacky za nákup vstupenek. Tato metoda však nebude implementována v rámci této diplomové práce a bude přidána později.

Pokud jde o odměny nabízené uživatelům, prvními jsou odznaky, které jsou viditelné ostatním a slouží jako vnitřní odměna. Další odměny budou prezentovány na samostatné obrazovce a zahrnují slevy na vstupenky, slevy u sponzorů atd. Hodnota jednoho virtuálního bodu odpovídá hodnotě koruny, tedy sto bodů je rovno sto korunám, a cena odměn bude stanovena na základě tohoto poměru.

3.2.4 Uživatelské rozhraní sekce

Pro tuto sekci je zásadní pečlivě navrhnout uživatelsky přívětivé, lákavé a funkční rozhraní. Proto bylo uživatelské rozhraní aplikace Eliterro vytvořeno designérem a je ilustrováno na obrázku 3.2.

Název odznaky	Popis	Co to přináší do aplikace
Nováček v aplikaci	Cílem tohoto odznaku je zpřístupnit uživatelům aplikaci a seznámit je s jejími funkcemi. Pro získání různých úrovní odznaku je nutné splnit úkoly, jako je vyplnění všech údajů v profilu, přidání oblíbených týmů a hráčů, aktivace odměn, dosažení první úrovně u ostatních odznaků atd.	Uživatelé se díky tomu lépe seznámí s aplikací a získají ucelený přehled o jejím používání
Věrný fanoušek	Úrovně odznaku se získávají na základě počtu dní, kdy byla aplikace navštívena. Například za první návštěvu získá uživatel první úroveň, a pro získání šestého úrovně je potřeba aplikaci navštívit tisíckrát (není nutná denní návštěva, počítají se pouze dny návštěvy)	Uživatel aplikace navštěvuje častěji
Týmový hráč	Uživatel postupuje na nové úrovně této odznaky tím, že zve přátele do aplikace, kteří pozvánku přijmou a zaregistrují se. Například za pozvání jednoho přítele získá uživatel jednu úroveň odměny. Pro dosažení nejvyšší úrovně je nutné pozvat 200 přátel	Díky tomu, že přátelé uživatele začnou aplikaci používat, roste celkový počet uživatelů
Milovník sportu	Uživatel dosahuje nových úrovní odznaku za počet sdílení obrazovek aplikace na sociálních sítích, přičemž každý den může získat maximálně jeden „bod“ za sdílení. Po prvním sdílení získá uživatel první úroveň, a aby dosáhl šesté úrovně, musí sdílet obrazovku 500krát	Sdílení aplikace na sociálních sítích představuje efektivní formu organického marketingu aplikace
Sportovní odborník	Úrovně této odznaky získává uživatel za počet správně zodpovězených otázek v denním kvízu. První úroveň dosáhne již po jedné správné odpovědi a nejvyšší, když správně zodpoví 500 otázek	Uživatelé se na aplikaci těší a snaží se ji navštěvovat pravidelně
Oddaný divák	Když je uživatel častým návštěvníkem různých zápasů, může snadno získat tuto odznaku. Za první návštěvu získá první úroveň a po dvousté návštěvě dosáhne poslední, šesté úrovně	Tento odznak slouží k přilákání uživatelů k návštěvě zápasů a k nákupu vstupenek

Tabulka 3.2 Popis odznaků, které budou implementovaný v aplikace Eliterro

Na tomto obrázku lze spatřit několik klíčových návrhů obrazovek, které ukazují, jak budou vypadat odznaky v profilu uživatelů, informační obrazovka umožňující uživatelům zjistit, kolik odznaků mají a jak mohou získat další, dále obrazovka poskytující kompletní přehled o virtuálních odměnách uživatele, a nakonec tabulka „vítězů“, kde si uživatelé mohou porovnat své umístění s ostatními.

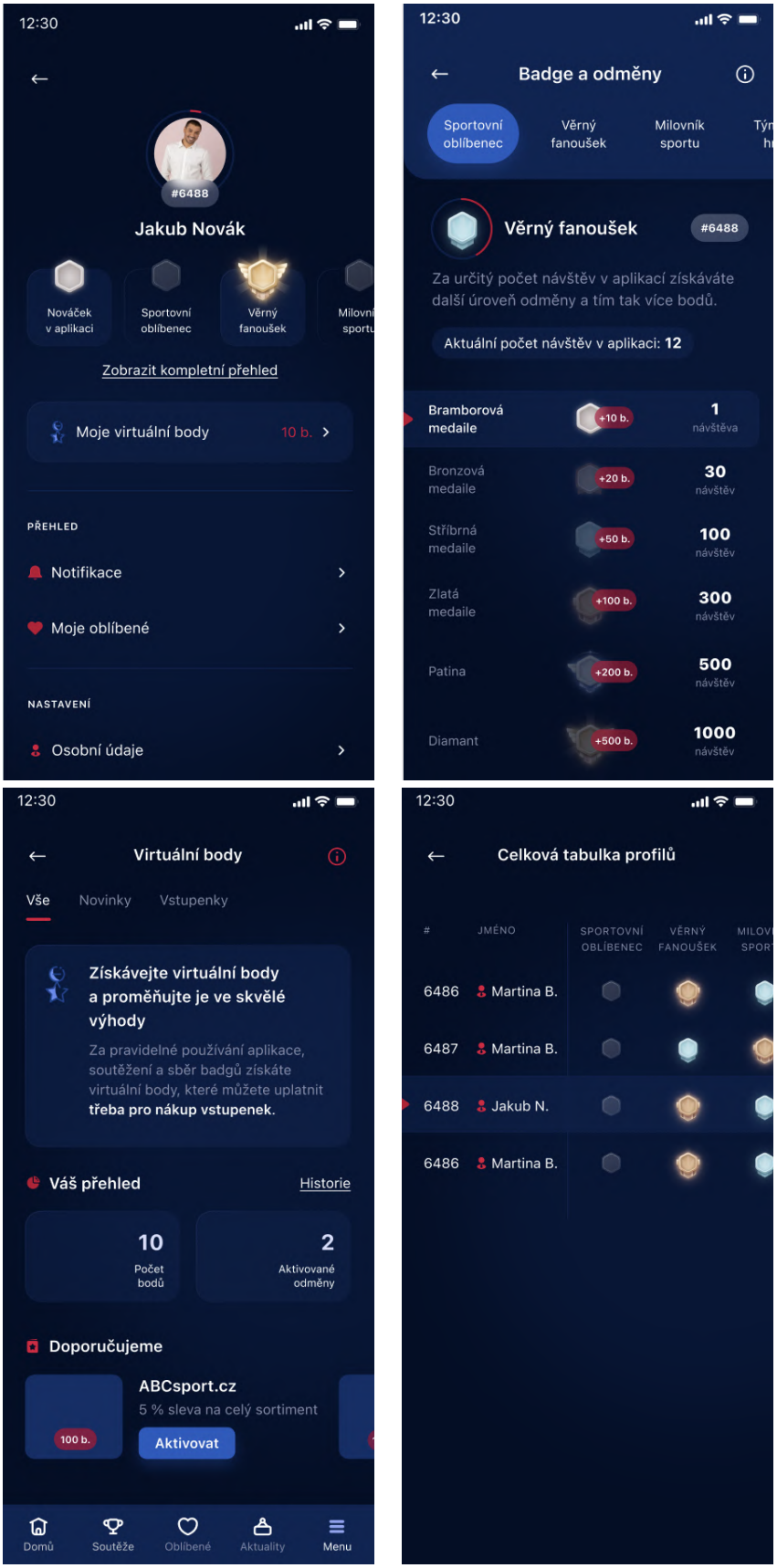
3.3 Chat

V této části se bude věnovat různým metodám moderace textového obsahu v aplikacích na platformě Flutter. Přestože chatovací aplikace obvykle vyžadují přítomnost lidského moderátora, aplikace Eliterro zatím bude využívat pouze automatické moderování. Lidský moderátor bude implementován později dle aktuálních potřeb, například pro řešení stížností uživatelů. V budoucnosti bude také nutné zahrnout moderování obrazového obsahu, což však není předmětem této diplomové práce. V následujících odstavcích budou popsány různé možnosti automatického moderování, jejich výhody a nevýhody, a bude vybrána nejvhodnější metoda pro aplikaci Eliterro. Navržený design dané sekce bude prezentován prostřednictvím lo-fi prototypů.

3.3.1 Způsoby pro moderování textového obsahu v Flutter aplikacích

Existuje množství metod pro moderování textového obsahu a detekci vulgárních výrazů, které lze snadno integrovat do aplikací vyvíjených ve Flutteru nebo je možné je v těchto aplikacích přímo využívat. Jednou z možností je využití následujících balíčků, které umožňují moderování obsahu přímo v aplikaci bez potřeby externích zdrojů:

- Balíček *profanity_filter* je jednoduchý nástroj pro vytváření filtrů, které kontrolují obsah textu na přítomnost vulgárních výrazů. Základní verze obsahuje seznam vulgárních výrazů v angličtině, který umožňuje zjistit, zda jsou tyto výrazy v textu obsaženy. Kromě toho lze k seznamu přidat další vlastní výrazy prostřednictvím konstruktoru *ProfanityFilter.filterAdditionally()* nebo vytvořit filtr založený pouze na vlastním seznamu pomocí *ProfanityFilter.filterOnly()*. Balíček nabízí několik metod, jako je *hasProfanity()*, která určí, zda text obsahuje vulgární výrazy, *getAllProfanity()*, která vrací všechna nalezená vulgární slova, a *censor()*, metoda pro nahrazení vulgárních výrazů bezpečným textem (ve výchozím nastavení hvězdičkami, ale lze nastavit i jiné náhrady). [75] Hlavními výhodami tohoto balíčku jsou jeho jednoduchost a možnost přizpůsobení, včetně možnosti přidat vulgární výrazy v češtině. Filtrování probíhá lokálně a nevyžaduje internetové připojení. Na druhou stranu, balíček neumí rozpoznat kontext zprávy, což může vést k chybným výsledkům. Také je nutné udržovat seznam urážlivých slov aktuální, což může být náročné, protože se vulgární slova rychle vyvíjejí.
- Balíček *badword_guard* je velmi podobný předchozímu, ale obsahuje dvojnásobně dlouhý základní seznam vulgárních výrazů, což může zvýšit jeho odolnost proti urážlivým slovům. Stejně jako první balíček, i tento je založen na anglickém jazyce, což může být omezení pro aplikace, které preferují jiné jazyky. [76] Balíček je novější a může proto využívat modernější technologie, nicméně disponuje pouze jednou stabilní verzí, což může znamenat přítomnost chyb a nedostatek aktualizací.
- Balíček *sentiment_dart* je softwarový modul, který využívá AFINN databázi (kolekce slov určených pro vyjádření sentimentu, přičemž každé slovo má přidělenou bodovou hodnotu v rozmezí od -5 do +5, kde -5 indikuje výrazně negativní sentiment a +5 naopak vyjadřuje pozitivní sentiment) a systém hodnocení sentimentu emoji, který je podobný hodnocení anglických slov, k analýze sentimentu libovolného textového obsahu. Tento balíček podporuje



Obrázek 3.2 Návrh uživatelského rozhraní "Virtuálních bodů"

čtyři jazyky – angličtinu, francouzštinu, italštinu a němčinu – a také emoji. Balíček obsahuje jedinou funkci *analysis()*, která přijímá jako vstup text a volitelně jazyk (výchozí je angličtina). Výstupem je skóre sentimentu textu spolu s popisem, která slova byla hodnocena jako negativní nebo pozitivní, a jejich hodnocením z databáze AFINN. [77] Mezi hlavní výhody tohoto balíčku patří pokročilé techniky moderace textového obsahu, což zvyšuje přesnost výsledků. Všechny výpočty se provádějí lokálně, což umožňuje moderaci obsahu offline, rychle a bezpečně. Na druhou stranu se tento typ analýzy považuje za relativně jednoduchý kvůli absenci pokročilých funkcí, jako je zpracování přirozeného jazyka, a značně závisí na správnosti a úplnosti databáze slov, což vyžaduje pravidelné aktualizace a údržbu. Navíc pro účely této diplomové práce je nezbytné najít nebo vytvořit podobnou databázi pro češtinu. Vilém Řezníček se pokusil vytvořit podobnou databázi českých slov [78], ale nelze se spolehnout na její aktuálnost a správnost, přičemž navíc používá zjednodušený bodovací systém pouze +1 a -1, což může vést k nesprávně pozitivním nebo negativním výsledkům pro tento balíček.

Dalším přístupem k moderování obsahu je využití technologií umělé inteligence, které obvykle zpracovávají požadavky na straně back-endu a výsledky poté vrací zpět do aplikace. Tyto systémy často implementují nástroje zpracování přirozeného jazyka (NLP) a strojového učení pro detekci škodlivého obsahu. Existuje množství různých řešení AI pro moderaci textového obsahu, které se liší v nastaveních, cenových modelích a uživatelských politikách. Mezi ně patří:

- **Perspective API Firebase Extension** – toto rozšíření nabízí efektivní řešení pro detekci urážlivého obsahu. Plugin Perspective API, stejně jako samotné rozhraní Perspective API, vyvinuté společností Jigsaw (součást Google), využívá technologie strojového učení k identifikaci úrovně toxického textu. Integrace tohoto API je provedena prostřednictvím Firebase, což značně zjednodušuje integraci a implementaci procesu. Pro zprovoznění tohoto rozšíření je nezbytné mít funkční a nastavený Firebase ve Flutter projektu, následně stáhnout rozšíření Perspective API Firebase a poté lze využívat funkce z tohoto rozšíření přímo v aplikacích Flutter. Toto rozšíření automaticky zpracovává interakci mezi Flutter aplikací a Perspective API – spravuje volání API a získávání odpovědí. [79] Řešení zaručuje snadnou integraci, moderování obsahu v reálném čase, jednoduchou škálovatelnost a je bezpečné a spolehlivé, podporované robustní infrastrukturou Google. Nicméně, může zahrnovat vyšší náklady (i když Firebase je zdarma, rozsáhlé používání Perspective API může být placené) a vede k závislosti na službách Firebase a konkrétním API.
- **Cloud Functions for Firebase** – prostředí, které poskytuje vývojářům možnost integrace cloudových funkcí do aplikací, umožňuje spouštět backendový kód jako reakci na události vyvolané funkcemi Firebase a požadavky HTTPS. Tyto cloudové funkce lze použít i pro vývoj funkce pro detekci urážlivého textu, kde lze například propojit s Perspective API, které hodnotí text na základě jeho úrovně toxicity. Uživatel odešle zprávu, Flutter tento text odešle na kontrolu do cloudové funkce pomocí HTTPS požadavku, funkce zpracuje požadavek prostřednictvím rozhraní API a pošle odpověď zpět do aplikace, kde se rozhodne, zda výsledné hodnocení dovoluje daný obsah či nikoli. [80] Toto řešení umožňuje vytvořit architekturu bez serveru (vývojář se nemusí starat o server, jeho škálování a údržbu), je škálovatelné a efektivní a jednoduše integrovatelné s celým ekosystémem Firebase (jako Firestore, Firebase Auth nebo Firebase Analytics). Na druhou stranu ale je to řešení, které vyžaduje robustní propojení s ekosystémem Firebase a migrace na jiné platformy a jiné backendové služby může být složitým problémem. Navíc testování a ladění cloud funkcí je velmi omezené a komplexní.
- **TensorFlow** – je open source knihovna vyvinutá společností Google Brain, se používá pro vytváření různých modelů neuronových sítí a je vhodná pro zpracování přirozeného jazyka (NLP), včetně detekce toxickeho nebo urážlivého textu. Pro Flutter existuje plugin *tflite_flutter* [81], který pomáhá zjednodušit integraci s TensorFlow a umožňuje implementaci a spuštění jejich modelů lokálně přímo v aplikaci(používá ale modely TensorFlow Lite, které

jsou speciálně optimalizované pro využití na mobilních zařízeních pro větší výkon a efektivitu). [82] Z výhod použití tohoto řešení plyne lokální zpracování moderace, což zajišťuje dobrý výkon a bezpečnost; existuje přímo balíček pro Flutter s dobrou dokumentací a proto napojení bude jednodušší; velký výběr různých trénovaných modelů; open source projekt s podpornou komunitou a dobře zpracovanou dokumentací. Z nevýhod lze uvést omezené možnosti modelů TensorFlow Lite z důvodu jejich optimalizace a zjednodušenosti pro použití na mobilních zařízeních (z čehož může plynout snížení přesnosti) a omezení modelů na jazyky a datové sady - může být těžko najít vhodné modely pro zjištění toxicity v češtině a možná bude třeba použít další technologie pro přípravu textu do kontroly (tzn. převod českého textu do angličtiny).

- **OpenAI** – laboratoř umělé inteligence, která vyvinula pokročilé modely známé svou dobrou schopností zpracování přirozeného jazyka. Jedním z takových modelů je GPT-3, který lze použít pro detekci urážlivého textu. Proces kontroly toxicity pomocí GPT-3 je založen na odesílání textových dat do GPT-3 API, které následně data zpracuje a vrátí hodnocení obsahu na základě trénovaných modelů. Pro použití GPT-3 API ve Flutteru stačí přidat balíček *http*, který pomáhá vytvářet požadavky HTTP (včetně zpracování požadavků API). Navíc existuje knihovna *chat_gpt_sdk*, která je vyvinuta pro zjednodušení integrace s OpenAI a abstrahuje programátora od mnoha složitostí, které souvisí s vytvářením požadavků HTTP a zpracováním odpovědí. [83] Pro zprovoznění tohoto řešení stačí se zaregistrovat na stránkách OpenAI, vygenerovat API klíč pro přístup a vytvořit službu pro volání GTP-3 API se zvolenými otázkami (v našem případě s otázkou týkající toxicity textu) a pak dostaneme zpátky odpověď, kterou je třeba zpracovat. [84] Výhodami použití API od OpenAI jsou vysoká přesnost odpovědí, snadná integrace s Flutter aplikací, neustálé zlepšování modelů na základě nových dat a výzkumu a z toho plynoucí přesnost výsledků. Z nevýhod lze zmínit, že přístup k GPT-3 API není zadarmo a může být nákladný pro velký objem textu; mohou nastavat problémy s latencí způsobené prováděním volání API přes síť a také možné obavy o soukromí - jelikož uživatelské data se posílají do externího rozhraní API. Je dobré uvést, že existuje podobné API od společnosti Google, které ale je zaměřeno přímo na zpracování přirozeného jazyka a analýzu textu - Google Natural Language API, ale integrace s Flutter je velmi podobná pro oba API.
- **Placená specialní API** – jako příklad lze uvést SightEngine nebo WebPurify. Tyto platformy nabízejí API, které umí rychle a efektivně moderovat různorodý obsah na předmět urážlivého obsahu pomocí technologií umělé inteligence. Když se bavíme o textovém obsahu, tak umí najít urážlivá slova a výrazy, vyfiltrovat nevhodný obsah, zabránit v obcházení filtru a umí detekovat URL odkazy. Navíc většinou podporují moderování různých jazyků. Hlavní nevýhodou takového řešení je, že se musí platit, cenová politika se hodně liší, ale ve většině případů se platí za počet provedených operací za měsíc, většinou cena základního balíčku startuje od 30 dolarů v měsíc pro nejzákladnější verzi. Další možnou problémovou může být to, že čeština není populárním jazykem a proto může se stát, že takové API ji nepodporují. Aplikace Eliterro je hlavně zatím zaměřena na česky mluvící, tak to může být velkým problémem.

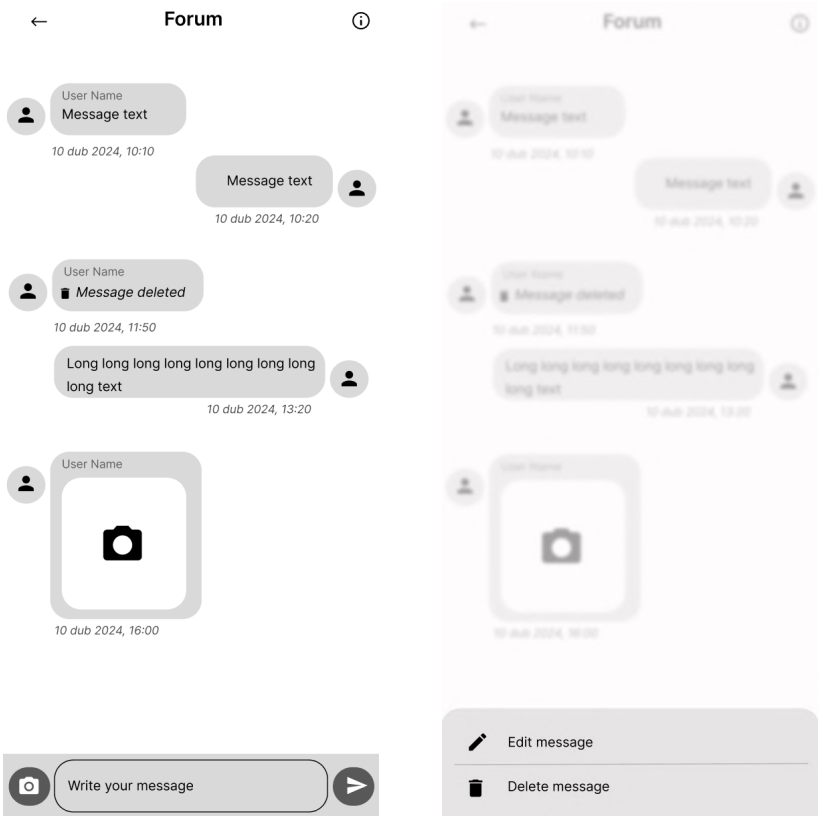
3.3.2 Vybrané řešení

Především bylo rozhodování o technologickém řešení ovlivněno cenovou politikou, neboť aplikace Eliterro zatím neplánuje využívat placené moderační API. S rostoucím počtem uživatelů se však možné zapojení takového API nevylučuje, což vedlo k vyřazení většiny technologií využívajících umělou inteligenci. Navíc, ačkoliv se v diplomové práci používá technologie Firebase, skutečná aplikace Eliterro disponuje vlastním backendem a nechce být závislá na zmíněné technologii. Proto bylo za vhodné považováno použití dvou technologií: TensorFlow, který umožňuje monitorovat toxické chování a další parametry na základě vybraného modelu – tento model je možné

v budoucnu nahradit jiným – a balíček *profanity_filter*, který zabraňuje použití velmi vulgárních slov. Tyto slova budou na fóru cenzurována na hvězdičky. Pokud se nepodaří najít vhodný model od TensorFlow pro češtinu, bude třeba řešit překlad zpráv do angličtiny s využitím příslušného balíčku a následné použití modelu trénovaného na detekci toxicity v angličtině.

3.3.3 Uživatelské rozhraní

Na základě provedené analýzy problému a definovaných funkčních požadavků byl vyvinut nízkofidelitní prototyp uživatelského rozhraní pro sekci ”Fórum fanoušků”. Tento prototyp je prezentován na přiloženém obrázku 3.3. Prototyp byl vytvořen pomocí nástroje Figma a skládá se ze dvou obrazovek. Hlavní obrazovka umožňuje uživatelům vzájemnou komunikaci, výměnu textových zpráv a obrázků. Druhá obrazovka, dialogové okno, zobrazuje možné akce s již odeslanými zprávami, které se aktivují dlouhým stiskem na zprávu v ”bublině”(editace je možná pouze u textových zpráv). Každá ”bublina”zprávy zahrnuje uživatelské jméno, datum a čas odeslání zprávy a profilovou fotografii. Navíc obsahuje hlavní obrazovka informační tlačítko v pravém horním rohu, které po stisknutí zobrazí pravidla a směrnice fóra a také sankce za jejich porušení.



Obrázek 3.3 Lo-fi prototyp fóra pro fanoušky

3.4 Nakup a správa vstupenek

Vzhledem k tomu, že vytvoření vlastní platební brány je složitý proces vyžadující rozsáhlé znalosti v oblastech backendového vývoje, databází a zabezpečení, bylo rozhodnuto v rámci této diplomové práce využít již existující platební bránu. V současnosti je na trhu dostupná široká

škála platebních bran, které odpovídají různým požadavkům a potřebám. Tato část práce poskytne přehled o různých platebních bránách, které je možné integrovat do aplikací vyvíjených v prostředí Flutter, přičemž se zaměří na jejich funkce, přednosti, omezení, a také na specifika jejich implementace a integrace. Následně bude vybráno nejvhodnější platební bránu a připravené lo-fi prototypy pro sekci "Nákup a správa vstupenek".

3.4.1 Různé možnosti pro implementaci platební brány

- **Stripe** je platforma, která nabízí mezinárodní platební brány a vyniká širokou nabídkou platebních metod a možností platby v různých měnách, spolehlivými bezpečnostními funkcemi a snadnou integrací. Její hlavní odlišností je design orientovaný přímo na vývojáře, což usnadňuje integraci na kterékoli z podporovaných platforem, díky čemuž je proces srozumitelný a dobře dokumentovaný. [85] Pro integraci do Flutter aplikací se využívá oficiálního pluginu *flutter_stripe* [86], který značně zjednodušuje proces integrace a umožňuje, aby se zpracování plateb odehrávalo přímo v aplikaci. Kroky pro integraci Stripe do Flutter aplikace zahrnují vytvoření a nastavení Stripe účtu, získání API klíče, přidání oficiálního pluginu do aplikace a správnou konfiguraci pro iOS a Android zařízení, nastavení Stripe přímo v aplikaci Flutter (inicializace a úprava uživatelského rozhraní podle potřeby) a nastavení správy plateb, což může být provedeno jak na straně backendového serveru, tak i přímo v aplikaci. Výhody používání Stripe zahrnují snadnou integraci, která umožňuje rychlé spuštění platební brány a zpracování plateb přímo v aplikaci, výborné zabezpečení, podporu více platebních metod a měn, a širokou škálu funkcí jako jsou nastavení předplatného a automatické opakování neúspěšných plateb. Hlavními nevýhodami jsou transakční poplatky (průměrně 1.5% + 6.50 Kč za jednotlivou transakci pro evropské karty), které mohou být vyšší než u konkurence, a také omezené možnosti pro přizpůsobení uživatelského rozhraní platební brány k rozhraní aplikace.

- **PayPal** je platební brána, která je považována za celosvětově uznávanou a používanou. PayPal nabízí různé platební možnosti, včetně možnosti platit přímo ze zůstatku na PayPal nebo použít jinou platební metodu, a umožňuje nakupování po celém světě bez obav z problémů s výměnou na jiné měny díky svým robustním bezpečnostním funkcím. [87] Pro integraci PayPal platební brány do Flutter aplikací je možné využít buď oficiální sadu PayPal SDK nebo pluginy třetí strany, jako je *paypal_flutter* [88], které zjednodušují proces integrace. Proces integrace bez doplňujících pluginů zahrnuje nastavení účtu PayPal a konfiguraci přihlašovacích údajů (tajný klíč, ID klienta), nastavení a konfiguraci backendového serveru, který zpracovává platby a volá PayPal API pro provedení plateb (nelze to dělat přímo v aplikaci, jelikož nelze sdílet citlivé přihlašovací údaje přímo v aplikaci), dále je třeba vytvořit koncové body, které se budou volat z klientské Flutter aplikace pro požadování o platbu nebo pro přijetí odpovědi o úspěšnosti platby. Z výhod použití PayPal jsou důvěra uživatelů (jelikož je to celosvětově známá značka), široký dosah a flexibilita (podporuje více než 100 různých měn a různé platební metody), kvalitní zabezpečení (je v souladu se standardy PCI). Z hlavních nevýhod jsou vysoké poplatky (ceny se začínají od 3$ plus fixovaný poplatek, který závisí na měně, za jednotlivou transakci) a složitost integrace způsobená neexistencí oficiálního pluginu pro Flutter.

- **Braintree** představuje globální řešení platební brány, které poskytuje společnost PayPal. Tato brána je široce využívána v oblíbených aplikacích jako Uber nebo Airbnb a umožňuje přijímat platby z různých zdrojů, včetně bankovních karet, účtu PayPal či bitcoinů, a podporuje také Google Pay a Apple Pay. [87] Pro snadnou integraci do aplikací Flutter je k dispozici plugin *flutter_braintree*. [89] Proces integrace zahrnuje registraci u Braintree, získání API klíče, nastavení serveru pro generování klientských tokenů a zpracování plateb, a konečně implementaci brány přímo v aplikaci. Mezi hlavní vlastnosti Braintree patří zabezpečení 3D dat, ochrana proti různým typům podvodů, dodržování standardů PCI, možnost sledování

transakcí v reálném čase a spolupráce s dalšími platebními stranami. Pozitiva zahrnují široké spektrum platebních možností, pokročilé zabezpečení, globální dostupnost a rozsáhlou podporu s dokumentací. Na straně negativ se nachází poměrně vysoké poplatky, které jsou stanoveny přibližně na 2,59 % + 0,49 USD za transakci, požadavek na backend server a omezenou kontrolu nad uživatelským rozhraním.

- **Razorpay** představuje jednu z předních platebních bran, která je široce užívána především v Indii. Tato platforma poskytuje komplexní řešení platebních služeb pro rozličné typy podniků. Umožňuje využití rozsáhlé palety platebních metod, včetně kreditních a debetních karet, mobilních peněženek a přes sto dalších možností. Razorpay také nabízí řadu funkcí pro efektivní správu plateb, jako jsou okamžité refundace a bankovní vyrovnání v reálném čase. [90] Integrace Razorpay s aplikacemi vyvinutými v Flutteru je zjednodušena díky oficiálnímu pluginu *razorpay_flutter* [91], který podstatně usnadňuje celý proces. Postup integrace zahrnuje vytvoření účtu Razorpay, přidání zmíněného pluginu do aplikace, nastavení platebních možností (například pomocí API klíče získaného od Razorpay, specifikace částky, popisu transakce apod.), přizpůsobení uživatelského rozhraní dle potřeb a implementaci callback funkcí pro zpracování platebních odpovědí. Mezi hlavní přednosti Razorpay patří snadná integrace podložená rozsáhlou dokumentací, široká nabídka různých platebních metod, vysoká úroveň zabezpečení splňující PCI standardy a pokročilé funkce pro reportování v reálném čase. Na druhou stranu, mezi nevýhody lze zařadit relativně vysoké náklady na jednotlivé transakce, které obvykle začínají na 2 % pro domácí platby, a zaměření této platební brány primárně na indický trh a spotřebitele.

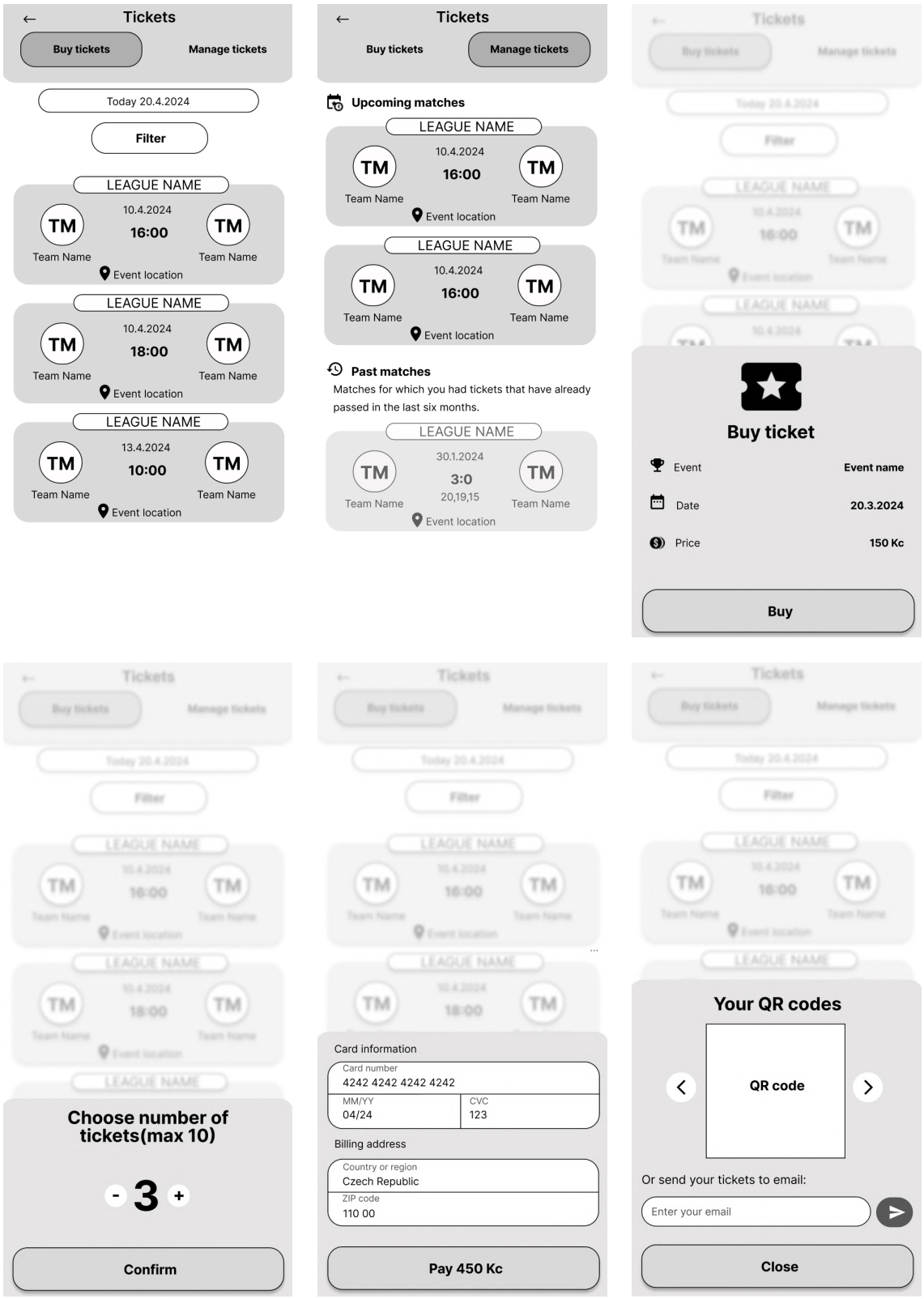
3.4.2 Vybrané řešení

Všechny zmiňované platební brány nabízejí bezpečné zpracování transakcí, možnost přizpůsobení uživatelského rozhraní aplikace, podporu různých platebních metod a měn a ve většině případů také jednoduchou a srozumitelnou integraci do aplikací Flutter. Přesto byla pro implementaci vybrána platební brána **Stripe**, především kvůli možnosti provádění plateb přímo v aplikaci. Toto řešení je momentálně klíčové, jelikož backendový server pro aplikaci zatím není připraven, a proto bude zpracování plateb dočasně prováděno přímo v aplikaci s možností pozdějšího přesunu na reálný server za účelem zvýšení bezpečnosti. Stripe se rovněž vyznačuje dobře dokumentovaným návodem k integraci, oficiálním pluginem pro Flutter a nejnižšími náklady mezi všemi posuzovanými bránami. Kromě toho Stripe podporuje platby prostřednictvím Apple Pay a Google Pay, což je zásadní pro implementaci platební brány v moderních aplikacích a výrazně zjednodušuje uživatelskou zkušenost. Ačkoli bude v rámci této diplomové práce implementována pouze jedna metoda platby – pomocí debetní nebo kreditní karty, budoucí přidání možnosti platby přes Apple Pay a Google Pay je nezbytné.

3.4.3 Uživatelské rozhraní

Na základě funkčních požadavků byl vytvořen nízkofidelitní prototyp uživatelského rozhraní pro sekci ”Nákup a správa vstupenek”, jehož náhled je uveden na obrázku 4.6. Tato sekce je navržena tak, aby obsahovala jedinou hlavní obrazovku rozdělenou na dvě záložky: ”Nákup vstupenek”a ”Správa vstupenek”. Obě záložky prezentují informace o různých sportovních utkáních, včetně názvu ligy, jména účastnících týmů, času, data konání a místa konání. Na záložce ”Nákup vstupenek”jsou zobrazena utkání s možností nákupu vstupenek, přičemž lze zápasy filtrovat podle týmu nebo ligy. Druhá záložka ”Správa vstupenek”zobrazuje jak vstupenky na nadcházející, tak i na minulé zápasy (do šesti měsíců zpět), u kterých je možné zhlédnout výsledky.

Navrhované uživatelské rozhraní zahrnuje dialogy pro nákup vstupenek, kde se uživatel stává kupujícím kliknutím na vybrané utkání v záložce ”Nákup vstupenek”. Po kliknutí je přesměrován do potvrzovacího dialogu, který obsahuje souhrnné informace o zápase a jeho cenu. Následně si

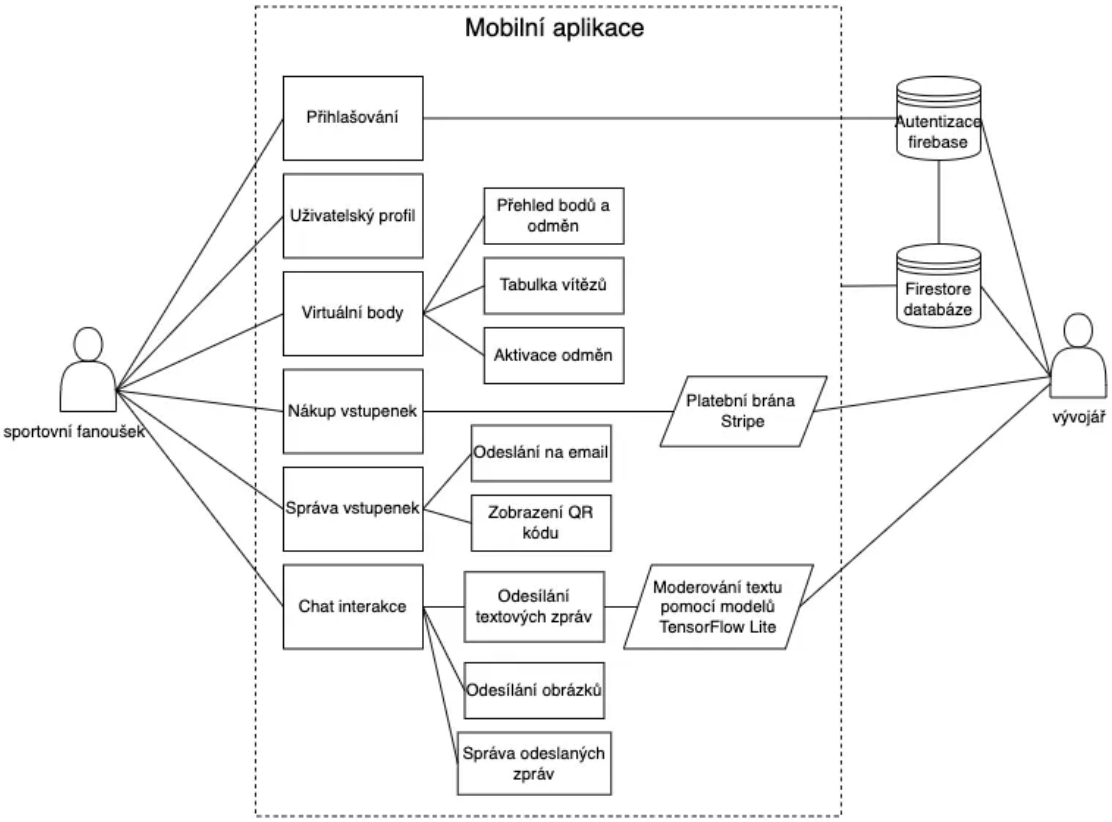


Obrázek 3.4 Lo-fi prototyp sekce ”Nakup a správa vstupenek”

uživatel vybírá počet vstupenek a platí prostřednictvím integrované platební brány v aplikaci. Po potvrzení platby se zobrazí QR kódy pro všechny zakoupené vstupenky, přičemž existuje také možnost poslat je e-mailem zadáním adresy a stisknutím tlačítka ”Poslat”. Podobný dialog se zobrazí, pokud uživatel klikne na aktivní zakoupenou vstupenku v záložce ”Správa vstupenek”, čímž je zajištěno, že o své vstupenky nepřijde.

3.5 Závěr

Návrh diplomové práce se věnoval podrobnému popisu technologií využitých v rámci celé aplikace, konkrétně Riverpodu a nástroji Firebase. Dále byla provedena analýza hlavních problémů, které je třeba vyřešit v jednotlivých sekcích. Pro každou sekci byly identifikovány a zdokumentovány různé možnosti řešení, z nichž byla pro mobilní aplikace ve Flutteru vybrána ta nejvhodnější. Pro každou sekci bylo také vytvořeno a popsáno uživatelské rozhraní, ať již ve formě lo-fi nebo hi-fi prototypů. Závěr práce obsahuje konceptuální diagram celé výsledné aplikace, viz obrázek 3.5. Na tomto diagramu jsou znázorněni primární uživatelé (v tomto případě sportovní fanoušci), klíčové funkcionality a akce, které lze v aplikaci provádět. Dále jsou zde uvedeny nezbytné externí technologie potřebné pro realizaci projektu, přičemž některé z nich se vztahují k celému projektu a jiné pouze k jednotlivým aktivitám.



Obrázek 3.5 Konceptuální diagram výsledné aplikace

Kapitola 4
Realizace

Tato část diplomové práce je věnována popisu praktického vývoje aplikace v rámci frameworku Flutter, který vyplývá z předchozího návrhu. Kapitola je strukturována do tří hlavních sekcí: instalační příručka, uživatelská příručka a programátorská příručka. Instalační příručka poskytuje podrobné instrukce pro úspěšné spuštění projektu. Uživatelská příručka se zabývá ovládáním aplikace, popisuje její funkcionalitu, možnosti, které aplikace nabízí, a obsahuje také ukázky z uživatelského rozhraní vyvinuté aplikace. Programátorská příručka je zaměřena na technické vysvětlení kódu, jeho klíčové a zajímavé části.

4.1 Instalační příručka

Pro spuštění aplikace v Flutter je nutné nainstalovat vývojové prostředí Flutter na počítači. Postup instalace pro různé platformy naleznete na oficiálních stránkách Flutteru [92]. Kromě toho je vyžadováno, aby na počítači bylo nainstalováno Android Studio nebo Visual Studio Code s pluginy pro Flutter a Dart. Po instalaci je vhodné spustit příkaz *flutter doctor* v terminálu, který prověří, zda je prostředí připraveno a zda jsou nainstalovány všechny nezbytné závislosti pro kompilaci a spuštění Flutter aplikací, přičemž uvede, zda něco chybí.

Dále je třeba zajistit, aby byl spuštěn emulátor nebo aby bylo k počítači připojeno reálné zařízení. Pro použití Android emulátoru je nezbytné mít nainstalované Android Studio, ve kterém lze emulátor vytvořit a konfigurovat. V případě simulátoru pro iOS zařízení, tyto jsou dostupné pouze na počítačích s operačním systémem macOS a pro jejich běh je třeba nainstalovat aplikaci XCode a nastavit v ní simulátory.

Pro spuštění aplikace použijte v terminálu příkaz *flutter run*. Tento příkaz spustí aplikaci v základním režimu ladění, který se během vývoje používá pro rychlé iterace a odstraňování chyb. Existují také další konfigurace pro spouštění Flutter aplikace, přičemž nejběžnější jsou režimy vydání a profilování. Pro spuštění aplikace v režimu vydání slouží příkaz *flutter run –release*, který je určen pro přípravu aplikace k nasazení do produkce a zahrnuje různé optimalizační techniky. Režim profilování, spouštěný příkazem *flutter run –profile*, je podobný režimu vydání, ale zahrnuje další nástroje pro analýzu výkonu aplikace, což je užitečné pro vyhodnocení výkonu.

Pro generování APK (Android application package), což je souborový formát používaný pro instalaci aplikací na zařízeních s Androidem, je třeba použít příkaz *flutter build apk*. Vygenerovaná verze aplikace je pak připravena k nahrání na Google Play nebo k distribuci jinými způsoby.

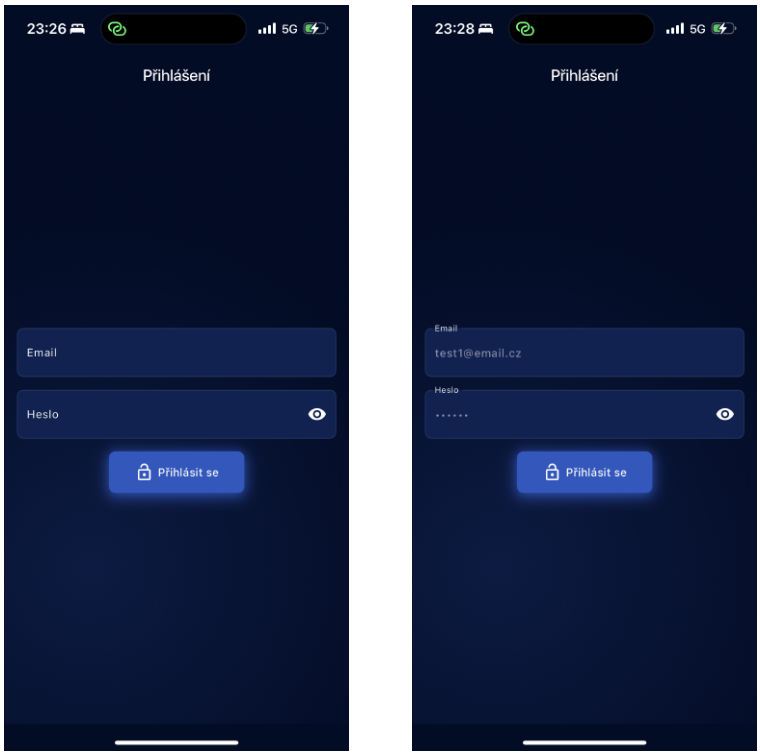
Pro vytvoření souboru .ipa, který obsahuje aplikace pro iOS, použijte příkaz *flutter build ipa*, který vytvoří archiv a poskytne soubor .ipa vhodný k distribuci prostřednictvím App Store nebo TestFlight, což je služba umožňující vývojářům distribuovat testovací verze aplikací. Je důležité poznamenat, že pro sestavení iOS aplikace je nezbytný počítač s operačním systémem macOS.

4.2 Uživatelská příručka

4.2.1 Přihlašovaní

Pro zobrazení virtuálních bodů a odměn, stejně jako pro možnost odesílání zpráv v chatu, je vyžadováno přihlášení uživatelů. Implementace přihlašovacího mechanismu je komplexní a časově náročná, a přesahuje rámec této diplomové práce. V rámci práce bylo proto zřízeno jednoduché přihlašovací rozhraní (bez možnosti registrace) a připraveno čtyři testovací účty.

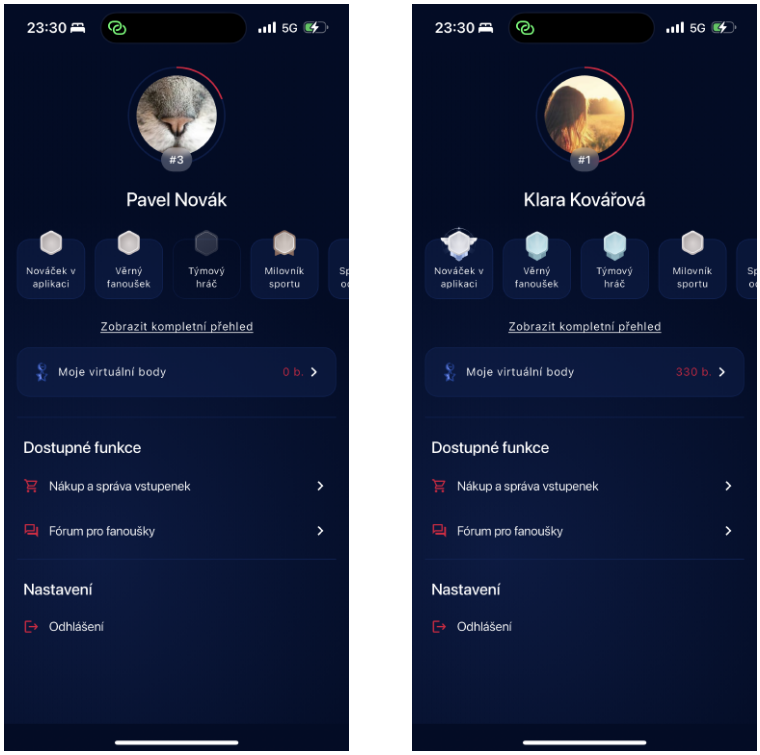
Vzhled přihlašovací obrazovky je zobrazen na obrázku 4.1. Přihlašování probíhá pomocí e-mailu a hesla. Po stisku tlačítka *Přihlásit se* je spuštěn autorizační proces prostřednictvím služby Firebase Auth a po úspěšném přihlášení do aplikace je uživatel automaticky přesměrován na obrazovku s profilem přihlášeného uživatele.



Obrázek 4.1 Obrazovka přihlášení

4.2.2 Uživatelský profil

Uživatelský profil slouží jako navigační rozhraní, které umožňuje přístup ke všem funkcím aplikace. Na obrázku 4.2 jsou zobrazeny rozhraní pro dva rozdílné testovací účty. Profil zahrnuje jméno uživatele, profilovou fotografii, krátký přehled získaných odznaků a počet virtuálních bodů, stejně jako jeho umístění v celkovém žebříčku uživatelů. Navíc nabízí přístup k dalším funkcím, jako je nákup a správa vstupenek nebo fórum pro fanoušky, a umožňuje také odhlášení z aplikace.

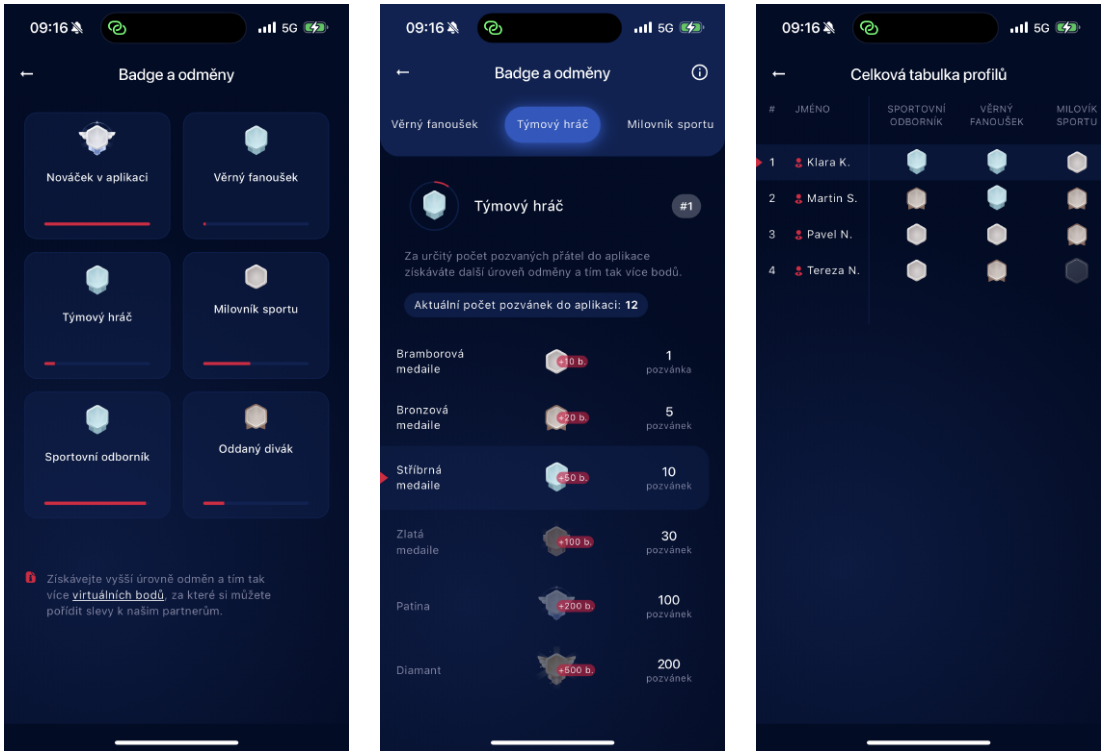


Obrázek 4.2 Obrazovka profil uživatele

4.2.3 Přehled odznaků

Pro zobrazení úplného přehledu odznaků je nutné kliknout na text *Zobrazit kompletní přehled*. Následně bude uživatel přesměrován na stránku, kde se zobrazí všechny dosažené úrovně odznaků společně s ukazatelem pokroku, který naznačuje, kolik zbývá k dosažení další úrovně. Uživatel má možnost vybrat si kterýkoliv z odznaků a po kliknutí na něj získat přístup k detailnímu přehledu. Alternativně může kliknout na zvýrazněný text *virtuální body*, což ho přímo přesměruje na sekci s přehledem a aktivací odměn. Tato sekce a její funkcionality budou podrobněji popsány v následující podsekci.

Po výběru konkrétního odznaku a přechodu na jeho detailní stránku se uživatel ocitne na obrazovce s podrobnými informacemi o vybraném odznaku. Tyto informace zahrnují název odznaku, pokrok, popis úkolů pro dosažení různých úrovní, jakož i přesný počet akcí potřebných k získání další úrovně. Zobrazuje se zde také současný počet splněných kroků a aktuálně dosažená úroveň. Na horní části obrazovky se nachází záložky s dalšími odznaky, což umožňuje uživatelům přecházet mezi detailními přehledy různých odznaků bez nutnosti se vracet zpět. Dále je na této stránce dostupné tlačítko s informacemi umístěné v pravém horním rohu, které poskytuje kompletní informace o odznacích a návod k používání této obrazovky. Uživatelé mají také možnost nahlédnout do žebříčku vítězů a zjistit, na jakém místě se nachází mezi ostatními hráči; k tomu slouží kliknutí na číslo u názvu odznaku na kterékoli záložce, což označuje jeho umístění v tabulce. Kompletní přehled odznaků, detailní popis vybraného odznaku a žebříček uživatelů jsou ilustrovány na obrázku 4.3.



Obrázek 4.3 Přehled obrazovek ze sekce detailů odznaků

4.2.4 Přehled odměn

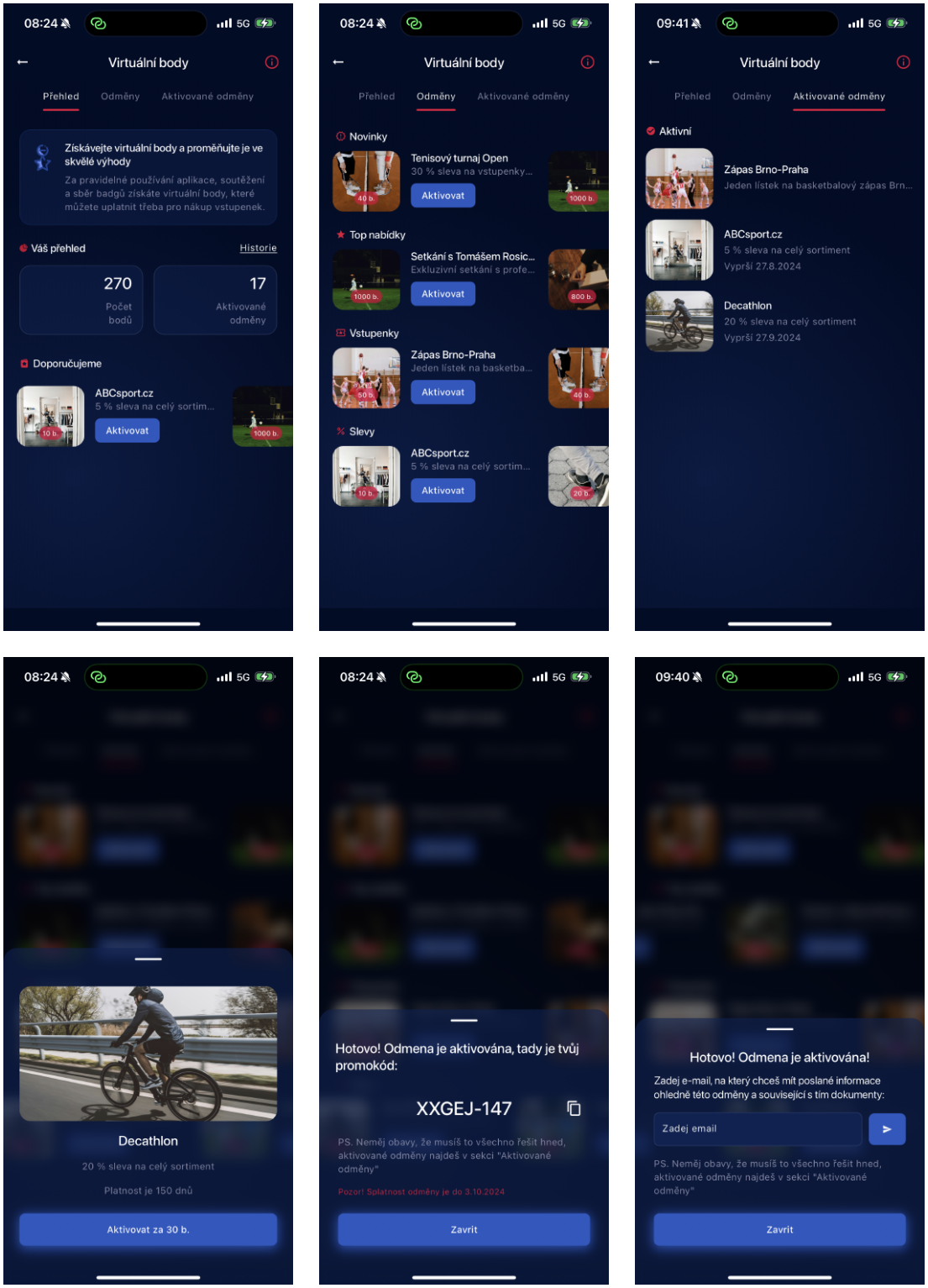
Pro přístup k sekci přehledu a aktivaci odměn je třeba, aby uživatel ve svém profilu klikl na tlačítko s nápisem *Moje virtuální body*, které jej přesměruje na novou obrazovku. Tato obrazovka nabízí tři záložky a v pravém horním rohu také informační tlačítko, jež poskytuje podrobnosti o virtuálních bodech, jejich získávání a principu získání odměn.

První záložka, nazvaná *Přehled*, zobrazuje celkový přehled virtuálních bodů, počet bodů a aktivovaných odměn. Z této záložky lze přejít na obrazovku s historií aktivovaných odměn kliknutím na zvýrazněný text *Historie*. Dále obsahuje seznam doporučených odměn.

Druhá záložka *Odměny* prezentuje všechny dostupné odměny rozdělené do různých kategorií. Po kliknutí na konkrétní odměnu se zobrazí dialogové okno s detailními informacemi o odměně (popis a platnost) a možností její aktivace, kterou lze provést i pomocí tlačítka *Aktivovat* přímo v seznamu odměn. Při aktivaci odměny, pokud uživatel disponuje dostatečným počtem bodů, buď aktivuje odměnu, nebo mu vyskočí dialogové okno upozorňující na nedostatek bodů. Po úspěšné aktivaci odměny může uživatel získat promo kód (v případě slevy do obchodu), který si může zkopírovat, nebo si nechat zaslat potvrzení a instrukce k aktivaci odměny e-mailem, který zadá do textového pole a následně stiskne tlačítko odeslat. Zobrazí se mu také datum, do kdy je odměna platná, pokud má omezenou dobu platnosti.

Uživatel nemusí řešit aktivované odměny ihned, protože na třetí záložce *Aktivované odměny* jsou zobrazeny všechny odměny, které kdy uživatel aktivoval. Zde jsou aktivní odměny, pro které může získat promo kód na slevu nebo poslat doplňující informace e-mailem, ale také odměny, které již vypršely nebo byly použity.

Na obrázku 4.4 jsou znázorněny několik příkladů sekce přehledu odměn, včetně ukázek všech tří záložek a tří příkladů dialogových oken: detailního přehledu odměny a dvou typů upozornění, které se objeví po aktivaci různých typů odměn.



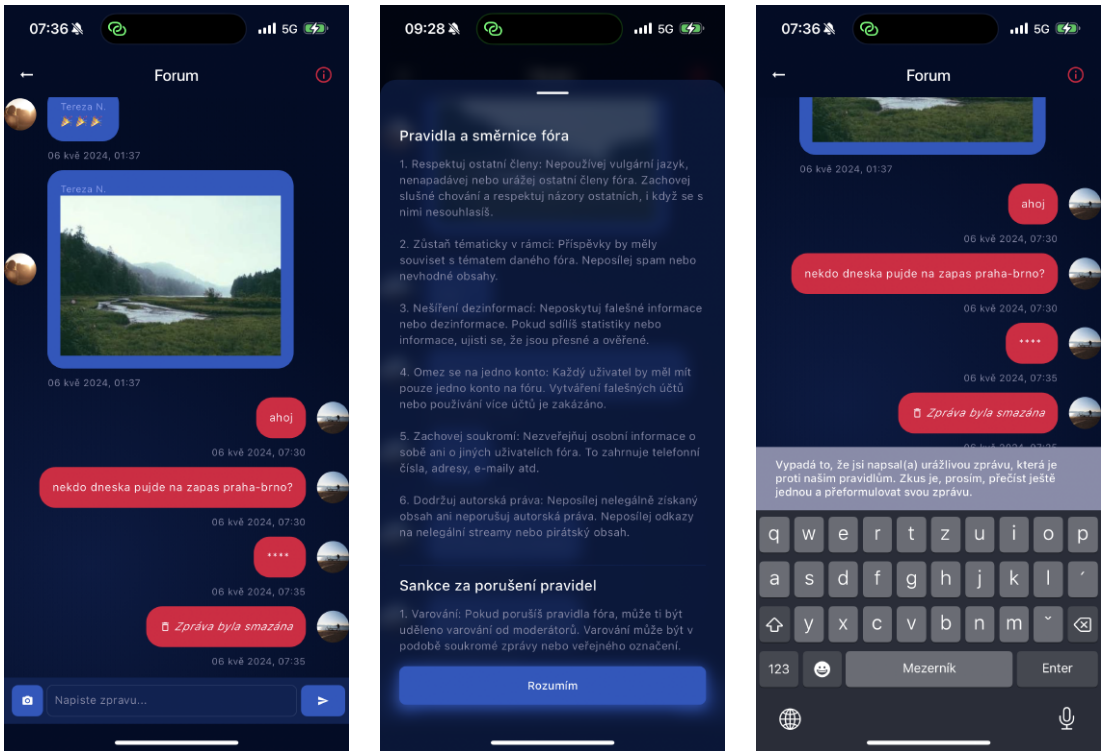
Obrázek 4.4 Přehled obrazovek ze sekce přehledu a aktivace odměn

4.2.5 Fórum fanoušků

Přechod do sekce fóra pro fanoušky je možné nalézt na profilové obrazovce pod odpovídajícím tlačítkem. Po vstupu do fóra se uživatel ihned dostane na stránku s chatem, kde jsou zobrazeny všechny zprávy, a to jak od samotného uživatele (označené červeně), tak od ostatních účastníků (označené modře). Uživatel má možnost seznámit se s pravidly a směrnicemi fóra, a to stiskem informačního tlačítka v pravém horním rohu. Okénko zprávy obsahuje text nebo obrázek, jméno odesílatele (pokud se nejedná o uživatele prohlížejícího), profilovou fotografii a také datum a čas odeslání zprávy.

K odeslání nové zprávy může uživatel využít dvě metody: napsání textu do textového pole a stiskem příslušného tlačítka nebo využití tlačítka s ikonou kamery, které uživatele přesune do galerie pro výběr fotografií k odeslání. Odeslané zprávy lze upravovat (pouze textové) a mazat dlouhým stiskem na požadované okénko se zprávou. V případě použití vulgárních výrazů ve zprávě bude uživatel upozorněn, že zpráva nebude odeslána kvůli obsahu nepřijatelného jazyka, nebo bude zpráva odeslána, ale nevhodné výrazy budou skryty za hvězdičkami.

Na obrázku 4.5 je zobrazena hlavní obrazovka fóra s ukázkami různých typů zpráv, dialogové okno s přehledem pravidel a sankcí fóra a obrazovka s upozorněním na zprávu, která nemohla být odeslána kvůli její vysoké negativitě.



Obrázek 4.5 Obrazovky ze sekce fóra fanoušků

4.2.6 Nakup a správa vstupenek

Pro nakupování a správu vstupenek je třeba stisknout tlačítko *Nákup a správa vstupenek* na profilové obrazovce. Na stránce pro nákup a správu uživatelé najdou dvě záložky: jedna slouží k nákupu vstupenek, druhá k jejich správě.

Na záložce pro nákup může uživatel vybírat zápas, který si přeje navštívit, buď z celkového seznamu dostupných zápasů nebo pomocí filtru. Filtr lze aktivovat tlačítkem *Filtrovat*, kde si uživatel může vybrat požadovanou ligu či tým ve filtrujícím dialogovém okně. Po výběru zápasu se objeví dialogové okno s podrobnostmi o zápase (liga, týmy, datum, čas a cena vstupenky). Po stisknutí tlačítka *Koupit* si uživatel zvolí počet vstupenek (maximálně 10) a spustí proces platby, přičemž je přesměrován do platebního dialogového okna, kde zadá své platební údaje. Úspěšná transakce vyústí v zobrazení dialogové okno s informacemi o úspěšném nákupu a vstupenky jsou uživateli poskytnuty ve formě QR kódů, které si může rovněž zaslat na e-mail zadáním e-mailové adresy do příslušného textového pole a stiskem tlačítka pro odeslání.

Při přechodu na záložku pro správu vstupenek uživatel uvidí seznam všech zakoupených vstupenek, seřazených podle data konání zápasu. Kliknutím na konkrétní kartičku se zápasem lze opět získat QR kódy k zakoupeným vstupenkám nebo je poslat na e-mail, podobně jako při dokončení nákupu. Na této záložce je také možné prohlížet přehled již odehraných zápasů z posledních šesti měsíců, pro které byly zakoupeny vstupenky, a to včetně zobrazení výsledků těchto zápasů. Aktivita kliknutí na tyto zápasy je však neaktivní.

Na obrázku 4.6 jsou prezentovány některé z obrazovek a dialogových oken této sekce. Zahrnuje výběr vyfiltrovaných zápasů na záložce pro nákup vstupenek, záložku správy vstupenek, stejně jako dialogová okna pro filtrování, kontrolu detailů zápasu, platební bránu a dialog potvrzující úspěšné dokončení nákupu.

4.3 Programátorská příručka

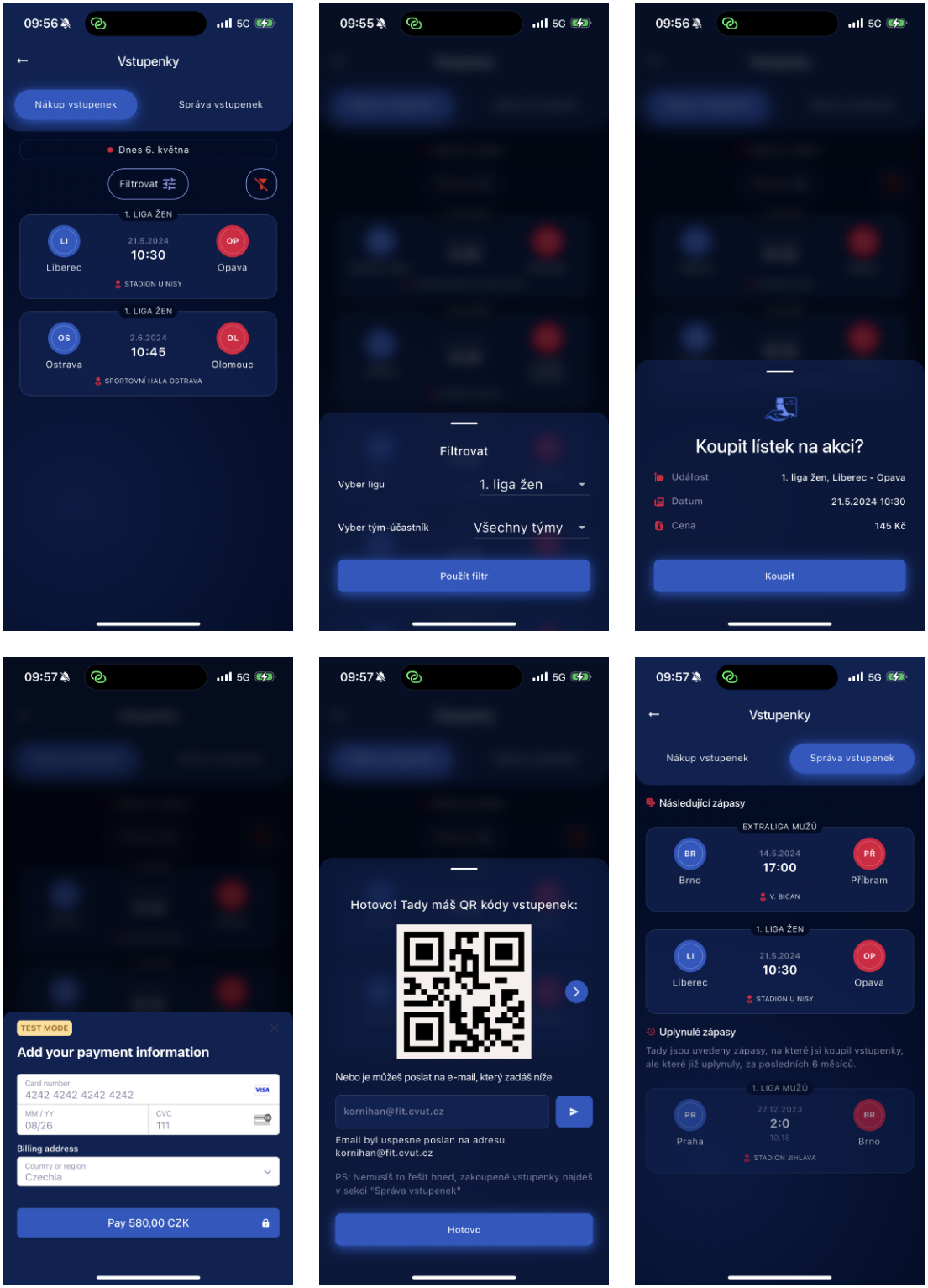
4.3.1 Verzování projektu

Během vývoje aplikace byl celý průběh projektu pečlivě dokumentován s využitím GitLabu, což je robustní systém pro správu verzí. Tento způsob práce umožnil detailní sledování všech změn a historie verzí. Aby bylo zajištěno, že základ kódu zůstane přehledný a spravovatelný, byla dodržována strategie větvení, při níž byla každá nová funkce vyvíjena v samostatné větvi. Tento přístup umožňoval oddělit vývojová prostředí pro jednotlivé funkce, což zajišťovalo, že integrace nových prvků mohla být před jejich začleněním testována nezávisle. Po úspěšném dokončení a ověření funkce byla příslušná větev sloučena do vývojové (dev) větve. Tento systematický proces větvení a slučování zajišťoval stabilitu vývojové linie a umožňoval bezproblémovou integraci a nasazování.

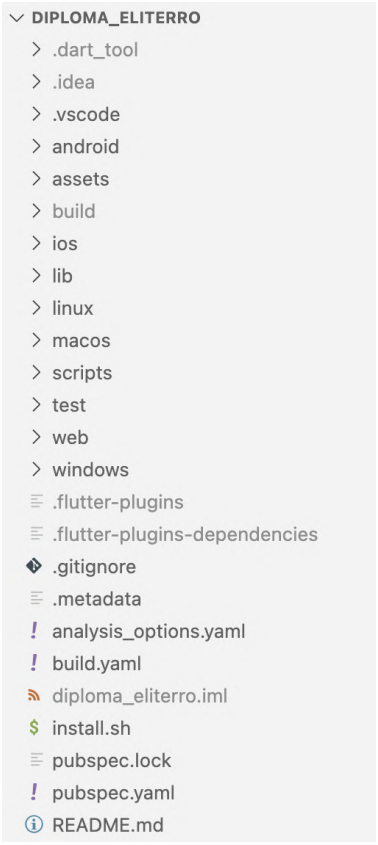
4.3.2 Struktura projektu

Struktura celého projektu je znázorněna na obrázku 4.7 a zahrnuje následující klíčové složky a komponenty:

- Složka **android** obsahuje všechny soubory specifické pro platformu Android, včetně manifestu Android, nativního kódu psaného v Java nebo Kotlin, a skriptů Gradle pro sestavení aplikace.
- Složka **assets** obsahuje různé typy prostředků, které lze v aplikaci použít, jako jsou obrázky, zvuky, videa a ikony. Pro jejich správné rozpoznání systémem Flutter je nutné na ně odkázat v souboru *pubspec.yaml*.
- Složka **build** je generována automaticky při sestavování aplikace a obsahuje výstupy build systému Flutter.
- Složka **ios** je podobná složce **android** a obsahuje specifické soubory a zdroje pro platformu iOS, včetně projektů XCode, různých konfigurací a nativního kódu psaného v Swift nebo Objective-C.



Obrázek 4.6 Obrazovky ze sekce pro nákup a správu vstupenek



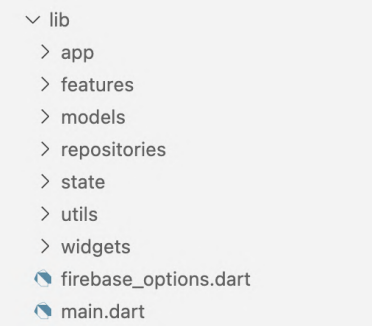
Obrázek 4.7 Struktura implementovaného projektu

- Složka **lib** obsahuje hlavní kód celého projektu napsaný v Dartu, včetně „vstupního bodu“ celé aplikace, který se nachází v souboru *main.dart*. Struktura této složky bude podrobně popsána později.
- Složky **linux, macos, web** slouží pro ukládání specifických souborů pro vývoj aplikací pro Linux, macOS a web. V této diplomové práci se však těmito platformami nezabýváme.
- Složka **scripts** obsahuje různé pomocné skripty pro urychlení vývoje, včetně skriptů pro generování překladů a různých generovaných souborů.
- Složka **test** obsahuje všechny testy napsané pro danou aplikaci, včetně unit a widget testů.
- Soubor **pubspec.yaml** je používán pro definici metadat projektu a správu závislostí. Zahrnuje odkazování na balíčky a pluginy od třetích stran, různé prostředky ze složky *assets* a různá nastavení projektu, včetně verze Flutter SDK nebo čísla verze aplikace.

4.3.3 Struktura složky lib

Hlavní složka pojmenovaná **lib** je základním úložištěm veškerého implementovaného kódu aplikace. Její struktura je ilustrována na obrázku 4.8 a zahrnuje následující komponenty:

- Složka **app** obsahuje základní prvky nezbytné pro fungování aplikace, včetně konfigurací navigace, tématu aplikace a aplikačních konstant.



Obrázek 4.8 Struktura složky lib

- Ve složce **features** se nachází všechny funkce dostupné v aplikaci, uspořádané podle jednotlivých funkcionalit. Každá z těchto podsložek může obsahovat obrazovky, widgety, modely, repositáře a poskytovatele specifické pro danou funkcionalitu.
- Složka **models** slouží pro ukládání společných datových modelů.
- Složka **repositories** zahrnuje repositáře, které se starají o komunikaci s databází Firebase a jsou využívány napříč celou aplikací, jako například repositář pro získaní uživatelských dat.
- Složka **state** obsahuje poskytovatele stavu pomocí Riverpod, které lze využít v celé aplikaci, například pro správu informací o virtuálních bodech nebo přihlášeném uživateli.
- Složka **utils** zahrnuje rozšíření některých tříd programovacího jazyka Dart, jako jsou DateTime nebo String. Nabízí také řadu užitečných nástrojů, jako jsou metody pro získávání různých atributů virtuálních bodů – například pokrok uživatele v získávání odznaků nebo aktuálně dosaženou úroveň. Tyto nástroje jsou systematicky organizovány do souborů podle svých funkcí.
- Složka **widgets** zahrnuje různé vlastní widgety, které jsou nebo mohou být v budoucnu použity napříč celou aplikací, včetně tlačítek, dialogů a textových polí. Widgety jsou uvnitř této složky organizovány do dalších podsložek podle typu.
- Soubor **firebase_options.dart**.dart představuje konfigurační soubor se specifickými nastaveními pro integraci služeb Firebase, včetně API klíčů a identifikátorů projektů.
- Soubor **main.dart** je vstupní bod aplikace, obsahuje funkci main(), která spouští aplikaci a inicializuje korenní widgety, stejně jako služby Firebase a Stripe pro platební bránu.

4.4 Práce s databází Firestore

Veškerá data potřebná pro tuto aplikaci jsou uložena v databázi Firestore. Pro přístup k ní a k dalším službám Firebase je nezbytné inicializovat Firebase ve funkci **main()** Flutter projektu pomocí následujícího příkazu:

```
await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
);
```

Abychom oddělili volání do databáze od widgetů, byly vytvořeny speciální soubory ***_repository.dart**, přičemž hvězdička značí účel daného repozitáře. Například **messages_repository.dart** je určen pro získávání a modifikaci dat v databázi souvisejících s chatem a zprávami, zatímco **virtual_points_repository.dart** se zabývá daty spojenými s virtuálními body, včetně získávání

virtuálních bodů, aktivovaných odměn konkrétního uživatele nebo získání odznaků uživatelů. Jako příklad lze uvést MessageRepository:

Výpis kódu 4.1 Implementace MessageRepository

```
class MessageRepository {
  final FirebaseFirestore _firestore = FirebaseFirestore.instance;

  Stream<List<ChatMessageModel>> getMessagesStream() {
    return _firestore.collection('messages')
    .orderBy('timestamp', descending: true).snapshots()
    .map(
        (snapshot) => snapshot.docs.map((doc) =>
        ChatMessageModel.fromDocumentSnapshot(doc.data(), doc.id))
        .toList()
    );
  }

  Future<void> updateMessage(String messageId, String content) async {
    DocumentReference messageRef = _firestore.collection('messages')
    .doc(messageId);
    await messageRef.update({'content': content});
  }

  Future<void> addMessage(ChatMessageModel messageToAdd) async {
    await _firestore.collection('messages').add(messageToAdd.toJson());
  }

  Future<void> deleteMessage(String messageId) async {
    DocumentReference messageRef = _firestore.collection('messages')
    .doc(messageId);
    await messageRef.update({'messageDeleted': true});
  }
}
```

Jak lze vidět na příkladovém kódu, **MessageRepository** obsahuje různé metody pro různé akce týkající se zpráv - získání všech zpráv, aktualizace, přidání nebo odstranění konkrétní zprávy. Přestože se data v databázi ukládají ve formátu JSON, při jejich získávání v aplikaci jsou ihned transformována do příslušného datového modelu. V případě **MessageRepository** se jedná o model **ChatMessageModel**, který uchovává informace o zprávě. Tento proces je zjednodušen díky tomu, že každý model má připravené metody **toJson()** a **fromJson(Map<String, Object?>json)**, generované pomocí balíčku *json_annotation*, který usnadňuje proces serializace a deserializace dat JSON. V případě **MessageRepository** se však používá metoda **fromDocumentSnapshot(Map<String, dynamic>json, String docId)**, která kromě metody **fromJson** přidává do datového modelu parametr **doc.id** – identifikátor dokumentu, jehož přidání do datového modelu je nezbytné.

Pro získání dat ve widgetech, zejména těch, které využívají metody načítající data z databáze a vrací objekty typu **Streamdynamic** , byl vytvořen widget **MyStreamBuilder**. Tento widget integruje **StreamBuilder** a navíc specifikuje chování při načítání dat a v případě výskytu chyby. Proces získání zpráv z databáze Firestore lze ilustrovat následovně:

Výpis kódu 4.2 Získání zpráv z databáze Firestore

```
MyStreamBuilder<List<ChatMessageModel>>(
    stream: _messageRepository.getMessagesStream(),
    builder: (messages) {
        return ...;
        },
```

```
);
```

Vidíme, že stream používá metodu **getMessagesStream()** z třídy **MessageRepository** jako parametr a v builderu vrací seznam objektů typu **ListChatMessageModel** , s nimiž je možné dále pracovat.

Pokud jde o ostatní metody, jako jsou přidání, úprava a mazání prvků v databázi, tyto jsou obvykle spouštěny reakcí na uživatelské akce (například stisknutí tlačítka). Tyto metody se často implementují v různých poskytovatelích, a jejich použití pak vypadá takto:

```
await _messageRepository.addMessage(messageToAdd);
```

Stačí tedy zavolat tuto metodu a přidat klíčové slovo **await**, které pozastaví vykonávání kódu, dokud se nesplní konkrétní akce (v tomto případě dokud není do databáze přidána nová zpráva).

4.5 Riverpod providery

Různé typy poskytovatelů Riverpod se využívají po celé aplikaci a lze je kategorizovat podle jejich využití.

4.5.1 Ukládání a aktualizace dat

Poskytovatelé, kteří mají za úkol uchovávat a zpřístupňovat proměnné a datové modely využívané na různých obrazovkách a widgetech, zahrnují například **userIdProvider**. Tento poskytovatel uchovává identifikátor aktuálně přihlášeného uživatele, stejně jako **virtualPointsProvider** a **activeRewardsProvider**, které jsou určeny pro správu dat souvisejících s virtuálními body a aktivovanými odměnami. Tyto poskytovatele se automaticky inicializují (načtou data z databáze), a následně je možné s těmito daty pracovat a aktualizovat je. Příklad implementace **virtualPointsProvider** ukazuje, že spolupracuje s **VirtualPointsNotifier**, kde dochází k inicializaci datového modelu virtuálních bodů a volání metody pro jejich aktualizaci.

Výpis kódu 4.3 Provider virtuálních bodů

```
class VirtualPointsNotifier extends StateNotifier<VirtualPointsModel> {
    final String userId;
    final VirtualPointsRepository _virtualPointsRepository =
    VirtualPointsRepository();

    VirtualPointsNotifier(this.userId) :
    super(VirtualPointsModel(pointsAmount: 0)) {
        _init();
    }

    void _init() {
        _virtualPointsRepository.getUserVirtualPoints(userId).first
        .then((points){
            state = points;
        }).catchError((error) {});
    }

    Future<void> updatePoints(int points) async {
        await _virtualPointsRepository.updateVirtualPoints(userId,
        points);
        _init();
    }
}
```

```
final virtualPointsProvider =
    StateNotifierProvider.family<VirtualPointsNotifier,
    VirtualPointsModel, String>((ref, userId) {
        return VirtualPointsNotifier(userId);
    });
```

Pro získání virtuálních bodů uživatele je možné využít následující příkaz (přístup k proměnné **ref** vyžaduje, aby widget byl typu ConsumerWidget nebo ConsumerStatefulWidget):

```
final virtualPoints = ref.watch(virtualPointsProvider(currentUserId));
```

Aktualizaci počtu virtuálních bodů pro konkrétního uživatele lze provést pomocí tohoto příkazu:

```
ref
    read(virtualPointsProvider(currentUserId).notifier)
    .updatePoints(virtualPoints.pointsAmount - reward.price);
```

4.5.2 Filtrace zápasů

Aplikace obsahuje komponentu s názvem **filterProvider**, která je implementována jako **StateNotifierProvider**. Tato komponenta uchovává stav filtrace, včetně vybrané ligy, týmu a informace, zda je proces filtrování aktivní. Stav je ukládán do struktury **FilterState**, která obsahuje uvedené parametry.

Dále **filterProvider** nabízí dvě metody: **setFilter(String? league, String? team)** a **resetFilter()**. Metoda **setFilter** aktivuje proces filtrování a aktualizuje stav s nově zvolenými filtry. Na druhou stranu, metoda **resetFilter** zruší veškeré nastavené filtry a obnoví výchozí stav.

Pro aplikaci filtrace po stisknutí tlačítka *"Filtrovat"* lze použít následující příkaz:

```
ref.read(filterProvider.notifier)
.setFilter(filterLeague.preselectedValue, filterTeam.preselectedValue);
```

4.5.3 Zpracování plateb

K optimalizaci zpracování plateb byli implementováni dva poskytovatelé. Prvním je **ticketsCountProvider**, což je jednoduchý provider, který uchovává informaci o počtu vybraných vstupenek. Tento počet se mění v reakci na uživatelovy změny požadovaného množství vstupenek v dialogovém okně. Druhým poskytovatelem je **paymentIntentProvider**, který je typu **FutureProvider**. Jeho hlavní funkcí je zajištění odeslání platebního požadavku prostřednictvím Stripe API přes HTTP protokol, čímž dochází k oddělení logiky zpracování API požadavků od widgetu. Implementace **paymentIntentProvider** je následující:

Výpis kódu 4.4 Provider pro zpracovani platby pomoci Stripe API

```
final paymentIntentProvider = FutureProvider.
family<Map<String, dynamic>, String>((ref, paymentDetails) async {
  var secretKey = dotenv.env['STRIPE_SECRET_KEY']!;
  var response = await http.post(
    Uri.parse('https://api.stripe.com/v1/payment_intents'),
    headers: {'Authorization': 'Bearer $secretKey',
    'Content-Type': 'application/x-www-form-urlencoded'},
    body: {
      'amount': paymentDetails.split(',')[0],
```

```
      'currency': paymentDetails.split(',')[1],
      'payment_method_types[]': 'card',
    },
  );
  return jsonDecode(response.body);
});
```

Pro zahájení procesu platby je potřeba použít následující příkaz:

```
final paymentIntent = await ref
    .read(paymentIntentProvider(paymentDetails).future);
```

4.5.4 Zpracování zpráv na fóru

V rámci aplikace je nezbytné efektivně spravovat různé akce spojené se zprávami, jako je jejich odesílání, úprava a mazání. Tyto činnosti musí být dostupné z různých widgetů. Pro tento účel byl vytvořen **messageProvider** – **StateNotifierProvider**, který uchovává informace o tom, zda probíhá úprava zprávy, o které konkrétní zprávě se jedná, nebo zda jde o novou zprávu. Stav tohoto provideru lze měnit pomocí metod **startEditing(ChatMessageModel message)** a **stopEditing()**. Struktura **MessageState** vypadá následovně:

```
class MessageState {
  final bool isEditing;
  final ChatMessageModel? editingMessage;

  MessageState({
    this.isEditing = false,
    this.editingMessage,
  });
}
```

Provider **messageProvider** nejenže řídí úpravy zpráv, ale také zahrnuje metody pro jejich přidávání a mazání. Metoda **sendMessage(String senderId, MessageType messageType, String content)** má na starost nejen moderaci zpráv, ale také jejich ukládání do databáze po přijetí. Na druhou stranu, metoda **deleteMessage(String messageId)** zabezpečuje vymazání zpráv z databáze. Typický příkaz pro smazání zprávy je:

```
await ref
    .read(messageProvider.notifier)
    .deleteMessage(messageId: widget.currentMessage.documentId);
```

4.6 Moderace textového obsahu na fóru fanoušků

Proces moderace textových zpráv na fóru fanoušků byl nakonec realizován ve dvou fázích.

V první fázi došlo k využití TensorFlow Lite modelu, natrénovaného pro klasifikaci zpráv. Model určuje, jaký procentuální podíl obsahu zprávy je pozitivní a jaký negativní. Jestliže negativita přesáhne 70%, zpráva není odeslána a uživatel obdrží upozornění. Model pro klasifikaci textu byl stažen z oficiálních stránek TensorFlow a integrován do složky **assets** projektu. Dále byla vytvořena třída **Classifier**, která text nejprve tokenizuje, převede na seznam ID slov (využívá soubor **text_classification_vocab.txt** jako slovník pro přiřazení slov k ID) a následně tento seznam zpracuje pomocí metody **run(Object input, Object output)** z balíčku **tflite_flutter**, aby získala pravděpodobnost pozitivity či negativity vstupu.

Během implementace se ukázalo, že najít TensorFlow Lite model schopný klasifikovat textový obsah v češtině je složité, a proto bylo třeba přidat fázi překladu textu do angličtiny pomocí balíčku **translator**.

Druhá fáze filtrace textu nastupuje, pouze pokud první metoda rozhodne, že text je možné zveřejnit. Tato metoda je inspirována balíčkem **profanity_filter**, avšak využívá odlišný přístup k identifikaci vulgárních slov. Existují dva seznamy vulgárních slov, jeden pro češtinu, druhý pro angličtinu, a třída **ProfanityFilter** zpracovává vstupní text a snaží se identifikovat přítomnost vulgárních slov. Pokud takové slovo najde, nahradí je hvězdičkami, v opačném případě vrátí text v původní podobě. Implementace třídy **ProfanityFilter** je založena na využití regulárních výrazů:

Výpis kódu 4.5 Implementace třídy ProfanityFilter

```
class ProfanityFilter {
  List<String> wordsToFilterOutList = [];

  ProfanityFilter() {
    wordsToFilterOutList = [...badEnglishWords, ...badCzechWords];
  }

  String censor(String inputString, {String? replaceWith}) {
    // Create a regex pattern to match words from the filter list with
    word boundaries
    String pattern = wordsToFilterOutList
        .map((word) => '\\b${RegExp.escape(word.toLowerCase()
        .withoutDiacritics())}[a-z]*\\b')
        .join('|');

    RegExp regExp = RegExp(pattern, caseSensitive: false);

    return inputString.replaceAllMapped(regExp, (match) {
      String replacement = replaceWith ?? '*' * match[0]!.length;
      return replacement;
    });
  }
}
```

Celý proces moderování textu a následného odesílání zprávy je strukturován následovně:

Výpis kódu 4.6 Celý proces moderování textové zprávy

```
//translate to english
final englishTranslation = await translator.translate(content, to: 'en');
//run first stage of moderation(tensorflow)
final classifierResult = classifier.classify(englishTranslation.text);
if (classifierResult[0] > 0.7) {
    onOffensiveContent?.call();
    return;
} else {
    //secon stage of moderation
    String cleanMessage = filter.censor(content);
    messageContent = cleanMessage;
}
await _messageRepository.addMessage(ChatMessageModel(
        senderId: senderId, timestamp: DateTime.now().toString(),
        content: messageContent, type: messageType.index));
```

4.7 Zpracování a odesílání obrázků ve fóru

Vzhledem k tomu, že chatovací sekce umožňuje odesílání obrázků (zatím pouze výběrem z galerie zařízení), bylo nutné zabezpečit správnou implementaci funkce pro výběr obrázku. Tato funkce zahrnuje také získání souhlasu uživatele pro přístup k galerii a následné úpravy kvality obrázku. Důvodem je limit Firestore databáze, která podporuje maximální velikost 1MB pro jeden dokument, což odpovídá jednomu datovému modelu zprávy včetně obrázku (pokud obsahuje).

Pro výběr fotografie z galerie se využívá balíček **image_picker**, přesněji metoda **pickImage(source: ImageSource.gallery)**. Otevření galerie a zpracování souhlasu s přístupem je řešeno interně v rámci balíčku. Pro správné fungování baličku na zařízeních s iOS je navíc nutné do souboru **Info.plist** přidat klíč **NSPhotoLibraryUsageDescription**, aby se zobrazilo dialogové okno s žádostí o přístup k galerii.

Po získání požadované fotografie začíná proces snižování její kvality. Metoda pro tento účel je uvedena níže:

Výpis kódu 4.7 Metoda pro snížení kvality fotografií

```
Future<String?> reduceQuality(File imageFile) async {
    Uint8List imageBytes = await imageFile.readAsBytes();

    img.Image? image = img.decodeImage(imageBytes);

    if (image != null) {
      int quality = 100;
      List<int> compressedImage = img.encodeJpg(image, quality: quality);
      while (compressedImage.length > 90 * 1024 && quality > 10) {
        quality -= 5;
        compressedImage = img.encodeJpg(image, quality: quality);
      }
      String base64Image = base64Encode(compressedImage);
      return base64Image;
    }
    return null;
}
```

Tato metoda snižuje kvalitu obrázku, dokud jeho velikost není dostatečná pro odesílání. Jakmile je dosaženo požadované kvality, obrázek je zakódován do formátu base64, a v takovém kódování je odeslán do databáze Firestore.

4.8 Integrace platebni brany

Platby v konečné aplikaci jsou zprostředkovány službou Stripe, která pro uživatelsky přívětivou integraci s Flutterem nabízí balíček **flutter_stripe**. Prvním krokem k integraci Stripe je registrace na jejich webu a získání veřejného a tajného API klíče. Z bezpečnostních důvodů je vhodné tajný klíč uchovávat na serverové straně, avšak vzhledem k absenci back-endu je tento dočasně uložen přímo v aplikaci. Inicializace Stripe probíhá v metodě **main()** následujícím způsobem:

```
Stripe.publishableKey = dotenv.env['STRIPE_PUBLISHABLE_KEY']!;
await Stripe.instance.applySettings();
```

Proces platby se skládá ze dvou klíčových metod: **makePayment()** a **displayPaymentSheet()**, které jsou uvedeny níže ve zkrácené formě:

Výpis kódu 4.8 Metody pro realizaci platby

```
Future<void> makePayment() async {
```

```
    final paymentSheetAppearance = PaymentSheetAppearance(
      colors: PaymentSheetAppearanceColors(
        primary: context.colorScheme.primary,
        background: context.colorScheme.primaryContainer,
        error: context.colorScheme.error,
        componentText: context.colorScheme.outline,
        placeholderText: context.colorScheme.outline,
        primaryText: context.colorScheme.onPrimary,
        secondaryText: context.colorScheme.onPrimary,
      ),
    );
    try {
      final paymentDetails = '${(ref.read(ticketsCountProvider) *
      widget.price * 100).toString()},CZK';
      final paymentIntent = await ref
      .read(paymentIntentProvider(paymentDetails).future);

      await Stripe.instance.initPaymentSheet(
        paymentSheetParameters: SetupPaymentSheetParameters(
          paymentIntentClientSecret: paymentIntent['client_secret'],
          merchantDisplayName: 'Elittero shop',
          appearance: paymentSheetAppearance,
        ),
      );
      await displayPaymentSheet(paymentIntent);
    } catch (e) {
      //error handling
    }
  }

Future<void> displayPaymentSheet(Map<String, dynamic> paymentIntent)
async {
    try {
      await Stripe.instance.presentPaymentSheet();
      //actions after success operation
    } on StripeException catch (e) {
      //error handling
    }
  }
}
```

Poté, co uživatel stiskne tlačítko *"Zaplatit"*, dojde k aktivaci metody **makePayment()**. Tato metoda nejprve definuje design platebního okna, které se následně zobrazí. Stripe nabízí rozsáhlé možnosti pro úpravu vzhledu téměř jakéhokoli prvku, což značně zvyšuje flexibilitu při designování. Proces platby se pak rozběhne odesláním požadavku na zpracování platby prostřednictvím paymentIntentProvider, který byl podrobněji popsán v předchozí sekci. Následuje inicializace platebního okna pomocí Stripe API, což může zahrnovat nastavení vzhledu a jména obchodníka. Jakmile je platební okno připraveno, zobrazí se uživateli. Veškerá logika spojená se zpracováním platby a zadáváním platebních údajů je delegována na Stripe. Po úspěšném dokončení platby, které nastane zavřením platebního okna, lze implementovat další funkce pro potvrzení úspěšné transakce.

Kapitola 5
Testování

Před uvedením aplikace na trh je nezbytné ji důkladně otestovat. Proto se tato kapitola věnuje testování aplikace, přičemž zahrnuje jak automatizované testy, tak testování uživatelské.

5.1 Automatizované testování

Před zahájením uživatelského testování je zásadní provést automatizované testování, aby byly odhaleny a odstraněny všechny základní chyby a překlepy. Pro tyto účely nabízí Flutter rozsáhlou sadu testovacích nástrojů, mezi které patří [93]:

- **Unit testy** - jedná se o nejmenší a nejvíce detailně zaměřené testy, které zkoumají chování jednotlivých částí kódu, jako jsou specifické funkce či třídy. Primárně jsou zaměřeny na ověření správnosti logiky, která je izolovaná od ostatních částí aplikace. Pro psaní a spouštění unit testů se používá balíček Dart *test*, který poskytuje standardizovaný způsob psaní testů a tvrzení (assertions).
- **Widget testy** - tento typ testů umožňuje zkoumat jednotlivé widgety nebo malé skupiny widgetů a ověřit, zda jejich chování odpovídá očekávání. Kontroluje, zda se widgety správně vykreslují v různých stavech a zda adekvátně reagují na uživatelský vstup. Tyto testy jsou složitější než unit testy, protože běží v kontextu prostředí Flutter Test Environment pomocí balíčku *flutter_test*, který rozšiřuje možnosti balíčku *test* Dart, ale nepotřebují pro běh skutečné zařízení nebo emulátor. Balíček *flutter_test* nabízí různé nástroje a widgety, které pomáhají simulovat uživatelské interakce a provádět kontroly ve stromu widgetů.
- **Integrační testy** - tyto testy zkoumají kompletní aplikace nebo jejich rozsáhlejší části za účelem zajištění celkové správné funkčnosti. Simulují skutečné uživatelské scénáře a pokrývají interakce napříč různými vrstvami aplikace. Jsou složitější než předchozí typy testů, jelikož interagují s aplikací stejně jako uživatel a vyžadují použití skutečného zařízení nebo emulátoru. Flutter poskytuje balíček *integration_test*, který umožňuje spouštění těchto rozsáhlejších testů na zařízeních nebo emulátorech a napodobuje chování uživatelů.

Tato diplomová práce se zaměřuje na unit testy a widget testy. Unit testy jsou určeny k ověření rozšíření funkcionalit různých tříd (například String nebo DateTime) a k testování logických funkcí a metod používaných v rámci aplikace. Widget testy zkoumají základní widgety, přičemž klade důraz na kontrolu správného použití prvků, barev, reakcí na stisknutí tlačítek, vyplňování textových polí a podobně. Všechny testy jsou umístěny ve složce *test* v kořenovém adresáři aplikace. Podsložka *unit* obsahuje unit testy a podsložka *widget* widget testy. Spuštění všech unit a widget testů lze provést pomocí příkazu *flutter run* v terminálu ve složce projektu.

Je důležité zdůraznit, že pro testování byly využity mock objekty, které imitují chování skutečných objektů v předem definovaném režimu. Tyto objekty jsou užívány k simulaci složitějších služeb, modelů, komponent apod. Mocky jsou obzvláště užitečné, pokud simulované objekty nejsou spolehlivé nebo jsou obtížně konfigurovatelné. Zajišťují, že testy selžou pouze v případě skutečného problému s testovanou funkcionalitou, a ne kvůli potížím s některými objekty. Pro použití mocků ve Flutteru se využívá balíček *mockito* [94].

5.2 Uživatelské testování

Po dokončení automatizovaného testování následuje testování uživatelské. Toto vyžaduje pečlivou přípravu: je nutné sestavit dotazníky pro období před a po testování, stejně jako testovací scénáře, které určují postup při samotném testování. Je důležité zdůraznit, že se jedná o uživatelské testování, nikoli o testování použitelnosti. Zaměřuje se primárně na ověření správnosti implementace funkcionalit a kontrolu, zda aplikace funguje bezchybně a v souladu s plánem.

5.2.1 Průběh testování

V této části jsou popsány postupy, které byly následovány při realizaci testování:

- Před zahájením testování je nezbytné získat souhlas testerů s nahráváním, neboť testování zahrnuje záznam činnosti.
- Je důležité informovat testery, že hodnotíme aplikaci, nikoli jejich schopnosti. Testerům by mělo být zdůrazněno, že dělání chyb je přirozené a vítané, a to i v případě, že se s testováním setkávají poprvé.
- Před samotným testem tester vyplní dotazník, který slouží k získání informací o jeho zkušenostech s testováním a zájmech souvisejících se sportem a sportovními aplikacemi.
- Úvod do testování zahrnuje krátké představení aplikace a vysvětlení testovacích instrukcí, včetně cílů, kterých má tester dosáhnout a scénářů, které mají být otestovány.
- Během samotného testování je testerovi poskytnut papír s krátkými instrukcemi a úkoly, včetně specifických detailů jako je například číslo platební karty pro nákup vstupenek.
- Po dokončení testování tester vyplní dotazník, který se týká jeho celkových zkušeností a obsahuje otázky na možná zlepšení aplikace.

5.2.2 Dotazníky

Testování zahrnuje vyplnění dvou dotazníků. První z nich je určen k vyplnění před zahájením testování a druhý po jeho dokončení.

5.2.2.1 Před-testový dotazník

- Jaké jsou vaše pocity ohledně testování nové mobilní aplikace dnes?
- Měli jste již někdy možnost účastnit se testování nějaké aplikace?
- Zajímáte se o sport? Sledujete pravidelně nějaké sportovní disciplíny?
- Máte nějaké zkušenosti se sportovními aplikacemi? Kterými?

5.2.2.2 Post-testový dotazník

- Jak byste hodnotili své celkové zkušenosti s používáním této aplikace?
- Je podle Vás uživatelské rozhraní intuitivní a je jeho používání snadné pro cíle a úkoly, které byly stanoveny?
- Měli byste zájem o pravidelné používání této aplikace?
- Máte nějaké konkrétní návrhy, jak by šlo aplikaci vylepšit?

5.2.3 Testovací scénáře

Účastníci testování obdrží testovací instrukce, které propojí všechny tři testovací scénáře. Každý scénář se bude věnovat konkrétnímu modulu aplikace a provedení specifických akcí v rámci tohoto modulu.

5.2.4 Orientace ve virtuálních bodech

Účel testování
Prověřujeme schopnost uživatele orientovat se v sekci „Virtuální body“. Zjišťujeme, zda je schopen najít souhrn svých bodů a odznaků, informace o způsobech jejich získání, seznam vítězů a aktivované odměny. Zároveň testujeme, zda dokáže odměnu aktivovat a poté si ji zaslat e-mailem nebo zkopírovat promo kód.

Počáteční bod
Uživatel je přihlášen pod jedním z testovacích účtů a nachází se na obrazovce s uživatelským profilem.

Koncový bod
Seznámí se s funkcemi sekce „Virtuální body“ a aktivuje si odměnu.

Instrukce pro testujícího
Na obrazovce s profilem uživatele naleznete přehled vašich odznaků a počet virtuálních bodů. Dále si prohlédněte seznam odznaků, zkontrolujte jej a pokuste se najít tabulku vítězů, aby jste mohli porovnat své odznaky s odznaky ostatních uživatelů. Následně vyhledejte přehled získaných bodů a dostupné možnosti odměn. Poté si aktivujte nějakou odměnu (můžete vyzkoušet i takovou, na kterou vám body nestačí) a v závislosti na zvolené odměně buď odměnu pošlete na e-mail, nebo si vyžádejte promo kód odměny.

Očekávané kroky

1. Uživatel prochází svůj uživatelský profil a seznamuje se s aktuálně získanými odznaky.
2. Přechází k přehledu odznaků, zjišťuje, jaké odznaky může získat a kolik bodů mu ještě chybí k získání dalších odznaků.
3. Najde cestu k tabulce vítězů, rozhlédne se a zjistí, na kterém místě se nachází.
4. Vrací se zpět na obrazovku svého profilu.
5. Přechází na obrazovku virtuálních bodů a prozkoumává přehled odměn.
6. Seznamuje se s různými typy odměn a prohlíží si jejich detaily.

7. Zkouší aktivovat odměnu a po úspěšné aktivaci si odměnu odesílá e-mailem nebo kopíruje promo kód odměny.

5.2.5 Seznámení s fórem fanoušků

Účel testování
Provádíme testování úloh testera v rámci modulu "Fórum fanoušků". To zahrnuje pokusy o odeslání textových zpráv či obrázků, úpravy či mazání zpráv, a rovněž zkoumání používání vulgárních výrazů v komunikaci.

Počáteční bod
Uživatel je přihlášen pod jedním z testovacích účtů a nachází se na obrazovce uživatelského profilu.

Koncový bod
Uživatel je seznámen s fórem a jeho základními funkcionalitami.

Instrukce pro testujícího
V uživatelském profilu vyhledejte sekci „Fórum fanoušků“, přejděte do této sekce a seznámte se s pravidly používání chatu. Následně si vyzkoušejte odesílání a úpravy zpráv: zkuste posílat různé typy zpráv, provádět jejich mazání a úpravu, odesílání nevhodných zpráv a podobně.

Očekávané kroky

1. Uživatel prochází svůj uživatelský profil a přejde do sekce "Fórum fanoušků".
2. Uživatel se seznámí s obsahem této sekce a prostuduje pravidla diskuse.
3. Uživatel opakovaně pokusí odeslat zprávu, ať už textovou nebo formou obrázku.
4. Uživatel se pokusí použít nepovolená slova a výrazy ve své textové zprávě.
5. Uživatel se seznámí s nástroji pro editaci zpráv a vyzkouší jejich použití.

5.2.6 Nákup a správa vstupenek

Účel testování
Hlavním účelem je prověření modulu "Nákup a správa vstupenek", což zahrnuje seznámení se s touto sekci, nákup a správu vstupenek.

Počáteční bod
Uživatel je přihlášen pod jedním z testovacích účtů a nachází se na obrazovce uživatelského profilu.

Koncový bod
Uživatel má zakoupenou vstupenku a umí zobrazit její QR kód nebo si ji poslat na e-mail.

Instrukce pro testujícího
Naleznete sekci pro nákup a správu vstupenek, zakupte vstupenku na jeden z nabízených zápasů, při nákupu použijte testovací platební údaje a po úspěšném dokončení transakce zkuste vstupenky odeslat na e-mail.

Očekávané kroky

1. Uživatel prochází svůj uživatelský profil a přejde do sekce ”Nákup a správa vstupenek”.
2. Uživatel se seznámí s obsahem této sekce a s nabídkou sportovních utkání, na která si může zakoupit vstupenky.
3. Uživatel vybere konkrétní zápas, na který má zájem koupit vstupenky, a určí počet požadovaných vstupenek.
4. Uživatel vyplní testovací údaje platební karty a úspěšně dokončí nákup.
5. Uživatel obdrží QR kód všech zakoupených vstupenek a podle svého přání si může tyto vstupenky zaslat na svůj e-mail.
6. Uživatel přejde na záložku správy vstupenek, kde si prohlédne své zakoupené vstupenky a opět si zobrazí QR kódy k těmto vstupenkám.

5.2.7 Výsledky testování

Testování se zúčastnilo pět osob ve věkovém rozmezí 20 až 30 let z různých profesních oborů a s rozdílnými zkušenostmi v testování. Pro zajištění kvalitnějšího testovacího zážitku, probíhalo testování osobně a s předchozím souhlasem účastníků bylo každé setkání nahráváno. Záznamy z testování jsou dostupné v příloze této diplomové práce.

Celkově testování proběhlo úspěšně a většina testerů splnila požadované úkoly. Zde jsou klíčové poznatky z testování:

- Všichni účastníci pozitivně hodnotili estetiku uživatelského rozhraní a celkově měli dobrý dojem z aplikace. Po testování vyjádřili zájem o její další používání, pokud by obsahovala sporty, které sledují. Dále uvedli, že s podobnými aplikacemi se dosud nesetkali.
- Většina testerů měla obtíže rozlišit mezi sekce „Přehled odznaků“ a „Přehled odměn“, jelikož kontejner „Moje virtuální body“ vede na sekci odměn a tlačítko „Zobrazit kompletní přehled“ není jasné, co obsahuje. Přesto se většině testerů podařilo dostat se k přehledu odznaků, nicméně následně měli problémy najít tabulku všech uživatelů, která byla skrytá za pozicemi uživatelů v přehledu odznaků.
- Většina testerů nenašla problém s nalezením sekce odměn. Při aktivaci odměn nebyly zaznamenány obtíže, až na jednu chybu: při zadávání e-mailu pro zasílání instrukcí ohledně odměn bylo textové pole zakryto klávesnicí, což ztěžovalo viditelnost zadaného textu. Jednomu testerovi se také podařilo aktivovat odměnu vícekrát, což vedlo k jejímu opakovanému zobrazení v sekci aktivovaných odměn. Bylo by vhodné přidat upozornění, pokud se uživatel pokouší znovu aktivovat již aktivovanou odměnu.
- S funkcí „fórum fanoušků“ měli účastníci pozitivní zkušenosti. Pravidla chatu byla snadno dostupná, a funkce monitoringu vulgarismů byla velmi oceňována. Avšak někteří testerů doporučili přidat možnost zvětšení odeslaných obrázků na celou obrazovku a zjistili, že velká písmena nejsou v chatu podporována a je možné odeslat zprávu složenou pouze z mezer, což by mělo být opraveno.
- V sekci „nákup a správa vstupenek“ nebyly zaznamenány žádné problémy. Filtrování a proces platby byly srozumitelné a správa vstupenek probíhala hladce.

Závěrem lze říci, že testování dopadlo úspěšně a drobné nalezené chyby budou v blízké budoucnosti opraveny. Co se týče navigace v sekci virtuálních bodů, doporučuje se přehodnotit její uspořádání, konzultovat možné změny s designérem a alespoň přidat jasné popisky a upravit texty, jelikož tato sekce je pro uživatele stále matoucí a nepřehledná.

Závěr

Tato diplomová práce se zabývala návrhem a implementací tří modulů – virtuální body, fórum fanoušků a nákup a správa vstupenek – s cílem ověřit jejich koncepty a v budoucnu je začlenit do aplikace Eliterro.

V úvodu byla provedena rešerše, která se věnovala různým typům aplikací a jejich vývoji v čase. Byla zdůrazněna výhodnost mobilních platforem jako hlavních platforem pro vývoj. Rešerše dále obsahovala popis technologií umožňujících multiplatformní vývoj a zdůvodnění výběru technologie Flutter.

Následovala analýza, zahrnující popis konceptů gamifikace a zkoumání různých možností odměňování uživatelů. Proběhla také analýza metod pro moderování nevhodného a vulgárního obsahu, především textového, a byly popsány různé možnosti integrace platebních systémů do aplikací, včetně popisu fungování platební brány. Závěrem analýzy byly sepsány funkční a nefunkční požadavky.

V návrhové části byly popsány klíčové technologie, které byly použity při vývoji aplikace. Byly navrženy vhodné virtuální odznaky a odměny pro sekci virtuálních bodů, zkoumány technologie pro implementaci textové moderace ve Flutteru (pro sekci fórum fanoušků) a pro integraci platební brány do Flutter aplikací (pro sekci nákupu a správy vstupenek). Následně byly vybrány nejvhodnější technologie a byl zdůvodněn jejich výběr. Dále bylo navrženo uživatelské rozhraní těchto tří sekcí ve formě hi-fi prototypu pro sekci virtuálních bodů a lo-fi prototypů pro sekci fóra fanoušků a nákupu a správy vstupenek. Nakonec byl představen konceptuální model celé aplikace.

Implementace obsahuje finální aplikaci ve Flutteru, která splňuje veškeré funkční a nefunkční požadavky. Aplikace je kompatibilní s mobilními platformami iOS a Android, testování na dalších platformách podporovaných Flutterem nebylo provedeno.

Testování zahrnovalo popis automatizovaného a uživatelského testování. Uživatelské testování bylo skutečně provedeno a po získání zpětné vazby byly opraveny drobné chyby. Problémy s navigací v sekci virtuálních bodů byly probrány s designérem aplikace Eliterro a v budoucnu bude design přehodnocen a provedeny korektury.

Závěrem lze říci, že všechny cíle této diplomové práce byly splněny a výsledky práce budou v budoucnosti vyhodnoceny a poslouží jako podklad pro implementaci těchto tří sekcí v aplikaci Eliterro.

Bibliografie

1. KOETSIER, John. *There Are Now 8.9 Million Mobile Apps* [online]. Forbes, Feb 28, 2020 [cit. 2024-03-11]. Dostupné z: https://www.forbes.com/sites/johnkoetsier/2020/02/28/there-are-now-89-million-mobile-apps-and-china-is-40-of-mobile-app-spending/?sh=4b1fe76621dd.
2. *Desktop vs Mobile vs Tablet Market Share Worldwide* [online]. StatsCounter GlobalStats, [b.r.] [cit. 2024-04-30]. Dostupné z: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide/#quarterly-200901-202402.
3. *Number of mobile app downloads worldwide from 2016 to 2023* [online]. Statista, [b.r.] [cit. 2024-04-30]. Dostupné z: https://www.statista.com/statistics/271644/worldwide-free-and-paid-mobile-app-store-downloads/.
4. *Evolution of Software Development — History, Phases and Future Trends* [online]. Geeks for geeks, Nov 29, 2023 [cit. 2024-05-02]. Dostupné z: https://www.geeksforgeeks.org/evolution-of-software-development-history-phases-and-future-trends/.
5. VOLLE, Adam. *App* [online]. Encyclopedia Britannica, Mar 1, 2024 [cit. 2024-05-02]. Dostupné z: https://www.britannica.com/technology/mobile-app.
6. POLICKY, Bruce. *The Evolution of Desktop Software* [online]. Tabs3, Dec 22, 2022 [cit. 2024-05-02]. Dostupné z: https://www.tabs3.com/the-evolution-of-desktop-software.html.
7. *A Complete Evolution of Mobile and Mobile Apps* [online]. Color Whistle, Mar 29, 2024 [cit. 2024-05-02]. Dostupné z: https://colorwhistle.com/evolution-of-mobile-apps/.
8. ANKENY, Jason. *Ten billion downloads and counting: The history of Apple's App Store, and its all-time top apps* [online]. Fierce Network, Jan 22, 2011 [cit. 2024-04-30]. Dostupné z: https://www.fierce-network.com/special-report/ten-billion-downloads-and-counting-history-apple-s-app-store-and-its-all-time-top.
9. FROMENT, Liz. *What is the Future of Mobile App Development? 5 Trends to Watch* [online]. Mendix, May 2, 2024 [cit. 2024-05-02]. Dostupné z: https://www.mendix.com/blog/what-is-the-future-of-mobile-app-development/.
10. MITROFANSKIY, KOSTA. *20 Main Trends in the Development of Mobile Applications* [online]. IntelliSoft, Jan 30, 2024 [cit. 2024-05-02]. Dostupné z: https://intellisoft.io/20-main-trends-in-the-development-of-mobile-applications-in-2022/.
11. TERRELL HANNA, KATIE. *7 Reasons Why We Love Mobile Apps (and you should, too!)* [Online]. Yapp, May 23, 2023 [cit. 2024-05-02]. Dostupné z: https://blog.yapp.us/7-reasons-we-love-mobile-apps/.

12. BEHANI, Nitesh. *10 Reasons Why Mobile Apps are Better Than Mobile Websites* [online]. Magneto IT Solution, [b.r.] [cit. 2024-05-02]. Dostupné z: https://magnetoitsolutions.com/blog/mobile-apps-vs-mobile-websites.
13. TECHNOLOGIES, DianApps. *10 Reasons Why Mobile Apps Are Better than Websites* [online]. Medium, Sep 29, 2023 [cit. 2024-05-02]. Dostupné z: https://medium.com/@marketing_96275/10-reasons-why-mobile-apps-are-better-than-websites-beb58ea22f2.
14. SOLUTIONS, Grow. *Cross-Platform Mobile Development Frameworks: Build Native-Quality Mobile Apps* [online]. Medium, Oct 3, 2023 [cit. 2024-04-20]. Dostupné z: https://medium.com/@growsolutions/cross-platform-mobile-development-frameworks-build-native-quality-mobile-apps-60fd978e0d84.
15. AWATI, Rahul. *Definition cross-platform mobile development* [online]. TechTarget, May, 2023 [cit. 2024-04-20]. Dostupné z: https://www.techtarget.com/searchmobilecomputing/definition/cross-platform-mobile-development.
16. CHURYLOV, Maksym. *PROS AND CONS OF REACT NATIVE FOR CROSS-PLATFORM APP DEVELOPMENT* [online]. Mindk, [b.r.] [cit. 2024-04-20]. Dostupné z: https://www.mindk.com/blog/react-native-pros-and-cons/.
17. AGARWAL, Hitesh. *Advantages and Disadvantages of Using React Native* [online]. Tech Exactly, Jun 21, 2023 [cit. 2024-04-20]. Dostupné z: https://techexactly.com/blogs/advantages-and-disadvantages-of-using-react-native.
18. *The Good and the Bad of React Native App Development* [online]. AltexSoft, May 24, 2021 [cit. 2024-04-24]. Dostupné z: https://www.altexsoft.com/blog/react-native-pros-and-cons/.
19. P., Vinay. *Pros and Cons Xamarin Development: A Detailed Analysis of the Frameworks* [online]. Mindpool Technologies, May 18, 2023 [cit. 2024-04-22]. Dostupné z: https://mindpooltech.com/pros-and-cons-xamarin-development/.
20. *The Good and the Bad of Xamarin Mobile Development* [online]. AltexSoft, Nov 13, 2020 [cit. 2024-04-24]. Dostupné z: https://www.altexsoft.com/blog/pros-and-cons-of-xamarin-vs-native/.
21. ALMEIDA, Joana. *Pros and Cons of Xamarin App Development* [online]. Distant Job, Jun 27, 2022 [cit. 2024-04-22]. Dostupné z: https://distantjob.com/blog/xamarin-app-development/.
22. *Xamarin - Application Package Size* [online]. Microsoft, Feb 5, 2018 [cit. 2024-04-24]. Dostupné z: https://learn.microsoft.com/en-gb/previous-versions/xamarin/android/deploy-test/app-package-size.
23. *Stack Overflow Trends* [online]. Stack Overflow, 2024 [cit. 2024-02-24]. Dostupné z: https://insights.stackoverflow.com/trends?tags=flutter%2Creact-native.
24. *What is Flutter?* [Online]. Amazon Web Services, 2024 [cit. 2024-02-20]. Dostupné z: https://aws.amazon.com/what-is/flutter/.
25. KARASAVVAS, Theodoros. *Why Flutter is the most popular cross-platform mobile SDK* [online]. Stack Overflow, Feb 21, 2022 [cit. 2024-02-23]. Dostupné z: https://stackoverflow.blog/2022/02/21/why-flutter-is-the-most-popular-cross-platform-mobile-sdk/.
26. *The Good and the Bad of Flutter App Development* [online]. AltexSoft, Aug 4, 2022 [cit. 2024-02-24]. Dostupné z: https://www.altexsoft.com/blog/pros-and-cons-of-flutter-app-development/.

27. SULLIVAN, Alex. *Examining performance differences between Native, Flutter, and React Native mobile development* [online]. thoughtbot, Aug 4, 2023 [cit. 2024-02-23]. Dostupné z: https://thoughtbot.com/blog/examining-performance-differences-between-native-flutter-and-react-native-mobile-development#conclusion.
28. INVERITA. *The Good and the Bad of Flutter App Development* [online]. Medium, Mar 10, 2020 [cit. 2024-02-24]. Dostupné z: https://medium.com/swlh/flutter-vs-native-vs-react-native-examining-performance-31338f081980.
29. BUCZKOWSKI, Tymoteusz. *Pros and Cons of Flutter App Development* [online]. Droids On Roids, Jul 6, 2023 [cit. 2024-02-25]. Dostupné z: https://www.thedroidsonroids.com/blog/flutter-pros-and-cons.
30. *Flutter architectural overview* [online]. Flutter, 2024 [cit. 2024-02-20]. Dostupné z: https://docs.flutter.dev/resources/architectural-overview.
31. DEMBNY, Michał. *What is Flutter App Development and How Can It Benefit Your Business?* [Online]. Droids On Roids, Aug 7, 2023 [cit. 2024-02-21]. Dostupné z: https://www.thedroidsonroids.com/blog/what-is-flutter-app-development.
32. ÜNSAL, Günseli. *Stateless vs Stateful Widgets in Flutter* [online]. Medium, Jul 2, 2023 [cit. 2024-02-23]. Dostupné z: https://medium.com/@gunseliunsal/stateless-vs-stateful-widgets-in-flutter-852741b6046e.
33. *Widget catalog* [online]. Flutter, 2024 [cit. 2024-02-21]. Dostupné z: https://docs.flutter.dev/ui/widgets.
34. *Co je Eliterro?* [Online]. Eliterro, 2024 [cit. 2024-02-26]. Dostupné z: https://www.eliterro.com/.
35. *Gamification Market Size Share Analysis - Growth Trends Forecasts (2024 - 2029)* [online]. Mordor Intelligence, 2024 [cit. 2024-03-05]. Dostupné z: https://www.mordorintelligence.com/industry-reports/gamification-market.
36. MILES, Jeni. *The right app rewards to boost motivation* [online]. Medium, Jul 11, 2017 [cit. 2024-03-05]. Dostupné z: https://medium.com/googleplaydev/the-right-app-rewards-to-boost-motivation-c1ec86390450.
37. VIVEK, Shukla. *Mastering User Motivation: How Gamification and Behavioral Psychology Enhance Mobile App Design* [online]. Medium, Aug 10, 2023 [cit. 2024-03-05]. Dostupné z: https://medium.com/appfoster/mastering-user-motivation-how-gamification-and-behavioral-psychology-enhance-mobile-app-design-68b49282141b.
38. *App Gamification: 9 Examples of Mobile Apps Using Gamification* [online]. Helpshift, Feb 28, 2024 [cit. 2024-03-06]. Dostupné z: https://www.helpshift.com/app-gamification-9-examples-of-mobile-apps-using-gamification/.
39. GEORGIOU, Michael. *The Psychology Behind Effective Gamification in Apps* [online]. Imaginovation, Oct 2, 2023 [cit. 2024-03-07]. Dostupné z: https://imaginovation.net/blog/gamification-in-apps/.
40. GOLOVNYA, Sofiya. *Learning Apps That Use Gamification: Examples and Features Overview* [online]. Riseapps, Aug 2, 2023 [cit. 2024-03-07]. Dostupné z: https://riseapps.co/gamification-in-learning-apps/.
41. CHOU, Yu-kai. *Actionable Gamification*. Createspace Independent Publishing Platform, 2015. ISBN 978-1511744041.
42. BILHAM, Jasmine. *Case study: How Duolingo Utilises Gamification to Increase User Interest* [online]. Raw Studio, Jul 2, 2021 [cit. 2024-03-09]. Dostupné z: https://raw.studio/blog/how-duolingo-utilises-gamification/.

43. IYER, Karthik. *Differences Between Intrinsic And Extrinsic Rewards* [online]. Hub engage, Aug 28, 2023 [cit. 2024-03-12]. Dostupné z: https://www.hubengage.com/employee-recognition/exploring-the-differences-between-intrinsic-and-extrinsic-rewards/.
44. CHERRY, Kendra. *Intrinsic Motivation vs. Extrinsic Motivation: What's the Difference?* [Online]. Verywell mind, Dec 13, 2023 [cit. 2024-03-12]. Dostupné z: https://www.verywellmind.com/differences-between-extrinsic-and-intrinsic-motivation-2795384.
45. CHERRY, Kendra. *How Schedules of Reinforcement Work* [online]. Verywell mind, Mar 13, 2023 [cit. 2024-03-12]. Dostupné z: https://www.verywellmind.com/what-is-a-schedule-of-reinforcement-2794864.
46. REJMER, Ben. *5 Important Approaches For Gamifying Mobile Apps* [online]. Buzinga, Jun 4, 2015 [cit. 2024-03-15]. Dostupné z: https://www.buzinga.com.au/buzz/5-important-approaches-for-gamifying-mobile-apps/.
47. WANG, William. *Reward Schedules and When to Use Them* [online]. Game developer, Mar 29, 2017 [cit. 2024-03-15]. Dostupné z: https://www.gamedeveloper.com/business/reward-schedules-and-when-to-use-them.
48. MILLISCENT, Lucio. *Understanding Content Moderation Types* [online]. New Media Services, May 31, 2023 [cit. 2024-03-20]. Dostupné z: https://newmediaservices.com.au/understanding-content-moderation-types/.
49. *What are the different types of Content Moderation* [online]. Sight Engine, 2024 [cit. 2024-03-20]. Dostupné z: https://sightengine.com/knowledge-center/what-are-the-different-types-of-content-moderation.
50. GOLEMANOVA, RALITSA. *WHAT IS CONTENT MODERATION? — TYPES OF CONTENT MODERATION, TOOLS, AND MORE* [online]. Imagga, Sep 8, 2021 [cit. 2024-03-21]. Dostupné z: https://imagga.com/blog/what-is-content-moderation/.
51. HETLER, Amanda. *6 content moderation guidelines to consider* [online]. TechTarget, Feb 8, 2024 [cit. 2024-03-21]. Dostupné z: https://www.techtarget.com/whatis/feature/Content-moderation-guidelines-to-consider.
52. SHERIDAN, Sorcha. *5 Types of Content Moderation and How to Scale Using AI* [online]. Levity, Nov 16, 2022 [cit. 2024-03-24]. Dostupné z: https://levity.ai/blog/5-types-of-content-moderation-and-how-to-scale-using-ai.
53. GOLEMANOVA, Ralitsa. *AUTOMATED CONTENT MODERATION — WHAT IS IT? BENEFITS, TOOLS AND MORE* [online]. Imagga, Sep 29, 2021 [cit. 2024-03-24]. Dostupné z: https://imagga.com/blog/automated-content-moderation/.
54. GILLIS, Alexander S. *DEFINITION natural language processing (NLP)* [online]. TechTarget, Feb, 2024 [cit. 2024-03-26]. Dostupné z: https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP.
55. *Content Moderation with HITL: Top Benefits and Types* [online]. Shaip, Sep 26, 2023 [cit. 2024-03-25]. Dostupné z: https://www.shaip.com/blog/automated-content-moderation-benefits-and-types/.
56. WALKER, Stephanie. *6 Common Tricks Used to Avoid Profanity Filters* [online]. Chekkee, Sep 18, 2020 [cit. 2024-03-27]. Dostupné z: https://chekkee.com/6-common-tricks-used-to-avoid-profanity-filters/.
57. *Are Text Moderation Services Better Than Profanity Filters?* [Online]. WebPurify, Mar 6, 2024 [cit. 2024-03-28]. Dostupné z: https://www.webpurify.com/blog/are-text-moderation-services-better-than-profanity-filters/.
58. *Profanity Filters: Advanced Detection Filtering Solutions* [online]. Spectrum Lab, 2024 [cit. 2024-03-28]. Dostupné z: https://www.spectrumlabsai.com/profanity-filters/.

59. DATSYUK, Yuliya. *Mobile Payments Gateway Integration: How to Choose and Integrate Payments Into a Mobile App* [online]. Kindgeek, Oct 12, 2023 [cit. 2024-04-07]. Dostupné z: https://kindgeek.com/blog/post/mobile-payments-gateway-integration-how-to-choose-and-integrate-payments-into-a-mobile-app.
60. SHISHKOV, Lyubomir. *Payment gateway – what is it and how does it work?* [Online]. emerchantpay, Apr 4, 2024 [cit. 2024-04-05]. Dostupné z: https://www.emerchantpay.com/insights/what-is-a-payment-gateway-and-how-does-it-work/.
61. PARIKH, Saurin. *What is a Payment Gateway? Meaning and How Does it Work?* [Online]. Razorpay, Sep 18, 2023 [cit. 2024-04-07]. Dostupné z: https://razorpay.com/blog/payment-gateway-101/.
62. BENNETT, Nadja. *What are payment gateways and how do they work?* [Online]. TrueLayer, Oct 28, 2022 [cit. 2024-03-30]. Dostupné z: https://truelayer.com/blog/payments/what-are-payment-gateways-how-do-they-work/.
63. *WHAT IS A PAYMENT GATEWAY AND HOW DOES IT WORK?* [Online]. Checkout.com, Dec 18, 2023 [cit. 2024-04-06]. Dostupné z: https://www.checkout.com/blog/what-is-a-payment-gateway-and-how-does-it-work.
64. *What are the Different Types of Payment Gateway?* [Online]. Razorpay, Sep 19, 2023 [cit. 2024-04-06]. Dostupné z: https://razorpay.com/blog/different-types-of-payment-gateway/.
65. SÁNCHEZ, Daniel Herrera. *Flutter Provider: What is it, what is it for, and how to use it?* [Online]. Medium, Aug 16, 2022 [cit. 2024-04-16]. Dostupné z: https://medium.com/bancolombia-tech/flutter-provider-what-is-it-what-is-it-for-and-how-to-use-it-47d6941860d7.
66. TIWARI, Ankit. *Riverpod: The Future of State Management in Flutter* [online]. Medium, Jun 30, 2023 [cit. 2024-04-16]. Dostupné z: https://medium.com/@im_AnkitTiwari/riverpod-the-future-of-state-management-in-flutter-98e43003a930.
67. BIZZOTTO, Andrea. *Flutter Riverpod 2.0: The Ultimate Guide* [online]. Code with Andrea, Oct 28, 2022 [cit. 2024-04-16]. Dostupné z: https://codewithandrea.com/articles/flutter-state-management-riverpod/.
68. *Riverpod Docs - All providers* [online]. Riverpod, 2023 [cit. 2024-04-16]. Dostupné z: https://riverpod.dev/docs/providers/provider.
69. TERRELL HANNA, Katie. *Definition Google Firebase* [online]. TechTarget, May, 2023 [cit. 2024-04-16]. Dostupné z: https://www.techtarget.com/searchmobilecomputing/definition/Google-Firebase.
70. *FlutterFire Overview* [online]. FlutterFire, [b.r.] [cit. 2024-04-17]. Dostupné z: https://firebase.flutter.dev/docs/overview.
71. *Firebase Authentication* [online]. Firebase Authentication, [b.r.] [cit. 2024-04-17]. Dostupné z: https://firebase.google.com/docs/auth.
72. *firebase_auth* [online]. Pub.dev - The official package repository for Dart a Flutter apps, [b.r.] [cit. 2024-04-17]. Dostupné z: https://pub.dev/packages/firebase_auth.
73. *Cloud Firestore* [online]. Firebase Documentation, [b.r.] [cit. 2024-04-17]. Dostupné z: https://firebase.google.com/docs/firestore.
74. *cloud_firestore* [online]. Pub.dev - The official package repository for Dart a Flutter apps, [b.r.] [cit. 2024-04-17]. Dostupné z: https://pub.dev/packages/cloud_firestore.
75. *profanity_filter* [online]. Pub.dev - The official package repository for Dart a Flutter apps, [b.r.] [cit. 2024-04-10]. Dostupné z: https://pub.dev/packages/profanity_filter.

76. *badword_guard* [online]. Pub.dev - The official package repository for Dart a Flutter apps, [b.r.] [cit. 2024-04-10]. Dostupné z: https://pub.dev/packages/badword_guard.

77. *sentiment_dart* [online]. Pub.dev - The official package repository for Dart a Flutter apps, [b.r.] [cit. 2024-04-10]. Dostupné z: https://pub.dev/packages/sentiment_dart.

78. ŘEZNÍČEK, Vilém. *Analýza sentimentu - databáze českých slov s polaritou (AFINN.CZ)* [online]. Root.cz, Jul 14, 2018 [cit. 2024-04-11]. Dostupné z: https://blog.root.cz/hadoop-kdy-uz-ma-cenu-o-nem-uvazovat-a-kdy-jeste-n/analyza-sentimentu-databaze-ceskych-slov-s-polaritou/.

79. ACKERS, Darren. *Moderating comments through the Perspective API Firebase Extension* [online]. Invertase, Jan 19, 2023 [cit. 2024-04-13]. Dostupné z: https://invertase.io/blog/moderating-comments-through-the-perspective-api-firebase-extension.

80. GARNIER, Nicolas. *Content Moderation with Cloud Functions for Firebase* [online]. The Firebase Blog, Jun 2, 2017 [cit. 2024-04-13]. Dostupné z: https://firebase.blog/posts/2017/06/content-moderation-with-cloud-functions/.

81. *tflite_flutter* [online]. Pub.dev - The official package repository for Dart a Flutter apps, [b.r.] [cit. 2024-04-12]. Dostupné z: https://pub.dev/packages/tflite_flutter.

82. *An end-to-end platform for machine learning* [online]. Tensorflow, [b.r.] [cit. 2024-04-12]. Dostupné z: https://www.tensorflow.org/.

83. *chat_gpt_sdk* [online]. Pub.dev - The official package repository for Dart a Flutter apps, [b.r.] [cit. 2024-04-13]. Dostupné z: https://pub.dev/packages/chat_gpt_sdk.

84. HATFIELD, Scott. *Use OpenAI's APIs in Flutter without Plugins* [online]. Medium, Jul 30, 2022 [cit. 2024-04-11]. Dostupné z: https://medium.com/@Toglefritz/use-openais-apis-in-flutter-without-plugins-3d183baa0ff0.

85. *Payments* [online]. Stripe Docs, [b.r.] [cit. 2024-04-14]. Dostupné z: https://docs.stripe.com/payments.

86. *flutter_stripe* [online]. Pub.dev - The official package repository for Dart a Flutter apps, [b.r.] [cit. 2024-04-14]. Dostupné z: https://pub.dev/packages/flutter_stripe.

87. PATEL, Manish. *Which are the Top 5 Flutter Payment Gateways?* [Online]. Invertase, Dec 2, 2022 [cit. 2024-04-13]. Dostupné z: https://www.concettolabs.com/blog/payment-gateway-integration-in-flutter/.

88. *flutter_paypal* [online]. Pub.dev - The official package repository for Dart a Flutter apps, [b.r.] [cit. 2024-04-14]. Dostupné z: https://pub.dev/packages/flutter_paypal.

89. *flutter_braintree* [online]. Pub.dev - The official package repository for Dart a Flutter apps, [b.r.] [cit. 2024-04-14]. Dostupné z: https://pub.dev/packages/flutter_braintree.

90. *Payment Gateway* [online]. Razorpay Docs, [b.r.] [cit. 2024-04-14]. Dostupné z: https://razorpay.com/docs/payments/payment-gateway/.

91. *razorpay_flutter* [online]. Pub.dev - The official package repository for Dart a Flutter apps, [b.r.] [cit. 2024-04-14]. Dostupné z: https://pub.dev/packages/razorpay_flutter.

92. *Install Flutter* [online]. Flutter Documentation, [b.r.] [cit. 2024-04-28]. Dostupné z: https://docs.flutter.dev/get-started/install.

93. *Testing Flutter apps* [online]. Flutter documentation, [b.r.] [cit. 2024-04-26]. Dostupné z: https://docs.flutter.dev/testing/overview.

94. *mockito* [online]. Pub.dev - The official package repository for Dart a Flutter apps, [b.r.] [cit. 2024-04-26]. Dostupné z: https://pub.dev/packages/mockito.

Obsah příloh

- `readme.txt` ..... stručný popis obsahu média
- `apk` ..... adresář se spustitelnou formou implementace pro Android
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
- `testRecords` ..... záznamy z uživatelského testování