

Implementace knihovny mobilní platformy pro multiplayerové hry

Bc. Jiří Dabberger

Diplomová práce
2025



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2024/2025

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	Bc. Jiří Dabberger
Osobní číslo:	A23306
Studijní program:	N0613A140022 Informační technologie
Specializace:	Softwarové inženýrství
Forma studia:	Prezenční
Téma práce:	Implementace knihovny mobilní platformy pro multiplayerové hry
Téma práce anglicky:	Mobile Platform Library Implementation for Multiplayer Games

Zásady pro vypracování

- Nastudujte a popište problematiku v kontextu tématu práce.
- Zvolte vhodné technologie a prostředky k implementaci.
- Navrhněte a implementujte mobilní knihovnu pro vývoj multiplayerových her.
- Navrhněte a implementujte hru Poker pro demonstraci vytvořené knihovny.
- Publikujte knihovnu.
- Popište dosažené výsledky.

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. Nearby Connection API. Online. Dostupné z: <https://developers.google.com/nearby/connections/overview>.
2. SPROVIERO, Fernando. Nearby Connection tutoriál. Online. 2022. Dostupné z: <https://www.kodeco.com/35461793-nearby-connections-for-android-getting-started>.
3. DIXIT, Prashant. Tvorba Android knihovny. Online. Dostupné z: https://medium.com/@prashantdixit_87612/creating-and-publishing-your-own-android-library-in-kotlin-2023-part-1-3e542dad4ca0.
4. BLINKA, Danielle. O hře Poker. Online. 2024. Dostupné z: <https://www.wikihow.com/Play-Poker>.
5. IDRIS, Nazmul. Publikace knihovny. Online. 2021. Dostupné z: <https://developerlife.com/2021/02/06/publish-kotlin-library-as-gradle-dep/>.

Vedoucí diplomové práce: **Ing. Petr Žáček, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **27. října 2024**

Termín odevzdání diplomové práce: **2. června 2025**



L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 17. listopadu 2024

PROHLÁŠENÍ AUTORA DIPLOMOVÉ PRÁCE

Beru na vědomí, že

- odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., v platném znění bez ohledu na výsledek obhajoby;
- diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému a bude dostupná k nahlédnutí;
- jedno vyhotovení diplomové práce v listinné podobě bude ponecháno Univerzitě Tomáše Bati ve Zlíně k uložení;
- na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- podle § 60 odst. 1 autorského zákona má Univerzita Tomáše Bati ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- podle § 60 odst. 2 a 3 mohu užít své dílo – diplomovou práci – nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tj. k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá; neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji, že

- jsem na diplomové práci pracoval(a) samostatně a použitou literaturu jsem řádně citoval(a); v případě publikace výsledků budu uveden(a) jako spoluautor;
- odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou obsahově totožné.
- při tvorbě této práce jsem použil/a nástroj generativního modelu AI ChatGPT za účelem přepisu vlastních vět do více srozumitelné podoby. Po použití tohoto nástroje jsem provedl/a kontrolu obsahu a přebírám za něj plnou zodpovědnost.

Ve Zlíně, dne

.....
podpis autora

ABSTRAKT

Tato diplomová práce popisuje vývoj a implementaci knihovny určené pro tvorbu multiplayerových her na mobilním zařízení. Cílem této práce bylo vytvořit nástroj pro vývojáře mobilních aplikací, který usnadní vývoj her pro více hráčů v lokálním okolí a zároveň tento nástroj použít pro vývoj hry Poker. Jádrem knihovny je vytvořeno moderním programovacím jazykem Kotlin s využitím Nearby Connections API pro zajištění plynulé a bezpečné komunikace mezi zařízeními. Hra Poker je také naprogramována jazykem Kotlin spolu s frameworkem Jetpack Compose pro tvorbu uživatelského prostředí. Výsledná práce klade důraz na zdokumentování vytvořené knihovny a popisu funkcionalit spojených se hrou poker.

Klíčová slova: Nearby Connections API, lokální multiplayer, android knihovna, Poker, Kotlin

ABSTRACT

This thesis describes the development and implementation of a library designed for creating multiplayer games on mobile devices. The main goal of this thesis was to develop a tool for mobile game developers that facilitates the creation of local multiplayer games while also using this tool to develop a poker game. The core of the library is built using the modern programming language Kotlin with use of Google's Nearby Connections API to ensure smooth and secure communication between devices. The poker game is also programmed in Kotlin, along with the Jetpack Compose framework for building the user interface. This thesis emphasizes the documentation of the developed library and the description of functionalities related to the poker game.

Keywords: Nearby Connections API, local multiplayer, android library, Poker, Kotlin

Na tomto místě bych rád poděkoval panu Ing. Petru Žáčkovi, Ph.D., za vedení této práce, cenné připomínky a podněty k rozšíření jejího tématu. Zároveň děkuji své rodině za podporu a poskytnutý čas na vypracování této práce.

OBSAH

ÚVOD.....	15
I TEORETICKÁ ČÁST	17
1 MOBILNÍ HRY	18
1.1 KRÁTKÁ HISTORIE MOBILNÍCH HER	18
1.1.1 Revoluce s příchodem chytrých telefonů	18
1.2 VÝVOJ MOBILNÍCH HER	18
1.3 DRUHY HER.....	19
1.3.1 Herní žánr	19
1.3.2 Ovládání her	21
1.4 HRY PRO JEDNOHO HRÁČE.....	21
1.5 HRY PRO VÍCE HRÁČŮ.....	22
1.5.1 Motivace hráčů.....	22
1.6 ONLINE MULTIPLAYER	23
1.6.1 Výzvy při vývoji.....	23
1.6.2 Synchronizace a komunikace mezi hráči	24
1.6.3 Připojování do her	25
2 LOKÁLNÍ MULTIPLAYER	27
2.1 ZPŮSOBY PŘIPOJENÍ.....	27
2.2 POROVNÁNÍ ONLINE VS LOKÁLNÍ MULTIPLAYER	28
2.2.1 Technické požadavky	28
2.2.2 Výhody a nevýhody	28
2.3 SONGPOP PARTY	29
3 NEARBY CONNECTION API.....	30
3.1 HLAVNÍ VLASTNOSTI.....	30
3.2 STRATEGIE PŘIPOJENÍ.....	30
3.2.1 Klastr	30
3.2.2 Klient-Klient.....	31
3.2.3 Hvězda.....	31
3.3 ZÁKLADY IMPLEMENTACE.....	31
3.3.1 Jednoduchá Kotlin implementace	32
3.3.2 ConnectionsClient	35
3.4 KONKURENCE.....	36
4 POKER.....	37
4.1 VYSVĚTLENÍ POJMŮ	37
4.1.1 Limity sázek	39
4.2 TEXAS HOLD'EM.....	40

4.3	KARETNÍ KOMBINACE.....	41
4.3.1	Vysoká karta (High Card):	41
4.3.2	Pár (One Pair).....	42
4.3.3	Dva páry (Two Pair).....	43
4.3.4	Trojice (Three of a Kind)	44
4.3.5	Postupka (Straight)	44
4.3.6	Barva (Flush).....	45
4.3.7	Full House	46
4.3.8	Čtveřice (Four of a Kind / Poker).....	47
4.3.9	Postupka v barvě (Straight Flush)	47
4.3.10	Královská postupka (Royal Flush)	48
5	ANDROID KNIHOVNA.....	49
5.1	VYTVOŘENÍ A KONFIGURACE PROJEKTU KNIHOVNY	49
5.2	PUBLIKOVÁNÍ KNIHOVNY	50
II	PRAKTICKÁ ČÁST	51
6	VZNIK NÁPADU	52
6.1	VOLBA TECHNOLOGIÍ	52
6.1.1	Kotlin a Jetpack Compose	52
6.2	POŽADAVKY NA KNIHOVNU.....	53
6.2.1	Funkcionální požadavky knihovny.....	53
6.3	FUNKCIONÁLNÍ POŽADAVKY HRY POKER.....	54
6.4	NEFUNKCIONÁLNÍ POŽADAVKY	56
6.4.1	Ukládání dat.....	56
7	VÝVOJ KNIHOVNY	58
7.1	VYBRANÁ STRATEGIE PRO KOMUNIKACI	58
7.2	VÝVOJ JEDNOTLIVÝCH ČÁSTÍ	59
7.2.1	Výzvy a komplikace během vývoje	59
7.3	STRUKTURA KNIHOVNY	60
7.4	IMPLEMENTACE SERVERSIDE Z POHLEDU VÝVOJÁŘE	60
7.4.1	Použití ServerSide z pohledu vývojáře.....	62
7.5	IMPLEMENTACE CLIENTSIDE Z POHLEDU VÝVOJÁŘE.....	63
7.5.1	Použití ClientSide z pohledu vývojáře	65
7.6	SYSTÉM PRO PŘENOS ZPRÁV	66
7.6.1	Definované typy zpráv	66
7.6.2	Definování vlastních typů zpráv.....	67
7.6.3	Princip fungování	67
7.7	BONUSOVÉ VLASTNOSTI KNIHOVNY	69
7.7.1	Kontrola potřebných oprávnění.....	69
7.7.2	Práce s textem v resources.....	70

7.8	VOLÁNÍ FUNKCÍ V PROJEKTU.....	70
7.8.1	Inicializace serveru.....	70
7.8.2	Inicializace klienta.....	71
7.9	VYDÁNÍ KNIHOVNY NA JITPACK	71
7.9.1	Dokumentace knihovny.....	72
7.9.2	Použití knihovny v projektu	72
7.10	BUDOUCÍ ROZŠÍŘENÍ KNIHOVNY	73
8	VÝVOJ HRY POKER.....	75
8.1	STAVY VE HŘE.....	76
8.1.1	Stav hráče	76
8.1.2	Stav hry	77
8.1.3	Stav hry – doplnění Zpráv a historie	78
8.2	IMPLEMENTACE VIEWMODELŮ Z KNIHOVNY EASYLOCALGAME.....	79
8.2.1	PlayerViewModel.....	79
8.2.2	ServerViewModel – pouze hostitel	80
8.2.3	ServerViewModel – hostitel jako hráč.....	82
8.2.4	PlayerCoreInterface.....	83
8.2.5	Dependency Injection – Dagger Hilt.....	84
8.3	DALŠÍ TECHNICKÉ ZÁLEŽITOSTI HRY	85
8.3.1	Manažer hry.....	85
8.3.2	Počítání karetních kombinací a vítězů	88
8.4	GRAFICKÉ PŘEDSTAVENÍ DALŠÍCH FUNKCÍ	90
8.4.1	Rozvržení hlavního menu.....	90
8.4.2	Vytvoření hry	92
8.4.3	Herní stůl.....	93
8.4.4	Efekty a animace ve hře	95
8.4.5	Centrum zpráv	96
8.4.6	Ostatní grafika ve hře	97
8.5	VYDÁNÍ HRY NA GOOGLE PLAY	98
	ZÁVĚR	99
	SEZNAM POUŽITÉ LITERATURY.....	101
	SEZNAM OBRÁZKŮ	103
	SEZNAM TABULEK.....	104
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	105
	SEZNAM PŘÍLOH.....	106

ÚVOD

Vývoj softwaru na mobilních platformách zaznamenal v posledních letech výrazný pokrok, a to jak po technické stránce, tak i z hlediska dostupnosti nástrojů pro vývojáře. Díky moderním API a podpoře bezdrátové komunikace mohou dnešní aplikace snadno navazovat spojení mezi zařízeními a sdílet data v reálném čase. To otevírá dveře celé řadě scénářů, ve kterých může více zařízení spolupracovat nebo si mezi sebou vyměňovat data lokálně – například při hraní her v rámci jedné místnosti.

Jednou z technologií, která tuto funkcionalitu na platformě Android umožňuje, je Nearby Connections API. Tato technologie využívá kombinaci Bluetooth a Wi-Fi k navazování spojení mezi zařízeními, aniž by bylo nutné složitě nastavovat síť. V kontextu herního vývoje to znamená možnost vytvořit zážitek pro více hráčů, kteří se mohou jednoduše připojit k sobě navzájem a hrát společně, například během výletu, na večírku nebo ve školním prostředí.

Tato práce se proto zaměřuje na návrh a implementaci knihovny, která usnadní vývoj lokálních her pro více hráčů na mobilní platformě Android. Součástí práce je také demonstrační aplikace v podobě jednoduché pokerové hry, která slouží jako praktická ukázka využití navrženého řešení v reálném scénáři. Výsledkem této práce je tedy kombinace dvou odlišných výstupů, přičemž jeden je přímo závislý na druhém.

Úvod teoretické části se věnuje stručné historii mobilních her a jejich variant pro více hráčů. Následně se zaměříme na již zmíněnou technologii Nearby Connections – její fungování, vlastnosti a základy implementace. V další kapitole si popíšeme hru Poker, která byla zvolena jako vhodná ukázka pro demonstraci funkcionalit vytvořené knihovny. Důvodem volby této konkrétní hry je především možnost jednoduše předvést konektivitu mezi hráči v jedné místnosti a jejich vzájemné hraní. Nejedná se tedy o podporu hazardních her, ale o simulaci společenské hry určené pro zábavu mezi přáteli. Teoretická část je následně zakončena kapitolou o vývoji a publikování Android knihoven, díky čemuž může být vytvořené řešení snadno zpřístupněno ostatním vývojářům.

Praktická část je pak věnována samotnému vývoji knihovny pro mobilní platformu a demonstrační pokerové hry. Nejprve se zaměříme na vznik nápadu, volbu technologií a definici vývojových požadavků. Následuje podrobný popis vývoje samotné knihovny, rozdělený do několika logicky navazujících částí. Ty se věnují zejména technickým aspektům implementace a mohou zároveň sloužit jako základní dokumentace pro její

budoucí využití. Od vytvořené knihovny se dále přesuneme k tvorbě mobilní hry Poker. Její vývoj je opět rozdělen na několik technických kroků, přičemž závěrečná část je ale věnována uživatelskému rozhraní a grafickému ztvárnění hry s bližším popisem jednotlivých obrazovek. Před samotným závěrem se budeme věnovat publikaci vytvořené hry do obchodu Google Play.

I TEORETICKÁ ČÁST

1 MOBILNÍ HRY

Mobilní hry se staly jedním z nejrychleji rostoucích segmentů herního průmyslu. Díky široké dostupnosti chytrých telefonů a tabletů má téměř každý přístup k velkému množství her různých žánrů. Vývoj mobilních her se neustále vyvíjí s novými technologiemi, herními mechanikami a obchodními modely.

1.1 Krátká historie mobilních her

První mobilní hry se na světě začaly objevovat v 90. letech 20. století v té době na jednoduchých tlačítkových telefonech.

Jednou z prvních a nejznámějších mobilních her byla hra **Snake** (Had), kterou společnost Nokia předinstalovala na své mobilní telefony v roce 1997. Hra byla jednoduchá, ale velmi návyková, a pomohla popularizovat myšlenku hraní na mobilních zařízeních. V té době byly možnosti omezené – hry měly pouze černobílou grafiku a ovládaly se fyzickými tlačítky telefonu.

S příchodem barevných displejů a prvního internetového protokolu WAP (Wireless Application Protocol) na přelomu 20. století na trh začaly vstupovat jednoduché hry využívající internetové služby v podobě herních obchodů. [1][2]

1.1.1 Revoluce s příchodem chytrých telefonů

Revoluce v mobilních hrách nastala s příchodem prvních chytrých telefonů. V roce 2007 vydal Apple svůj první iPhone, který představil revoluční kapacitní dotykový displej, jenž eliminoval potřebu fyzických tlačítek a umožnil zcela nový způsob ovládání her. O rok později, v roce 2008, Apple spustil **App Store**, který výrazně usnadnil distribuci mobilních aplikací a her. [1; 2]

1.2 Vývoj mobilních her

Během vývoje mobilních her se využívá široká škála nástrojů, programovacích jazyků a specializovaných **herních enginů**, které usnadňují tvorbu grafiky, fyziky a herní logiky. Mezi nejpoužívanější enginy patří **Unity**, **Unreal Engine** nebo **Godot**. Vývoj mobilních her však nevyžaduje pouze technické znalosti, ale i pochopení specifik mobilních zařízení, jako jsou **dotyková obrazovka**, **omezený výkon a baterie**, **různá rozlišení displejů** či způsob distribuce přes Google Play a App Store. [3]

Úspěšný vývoj hry navíc nespočívá pouze v technické stránce, ale vyžaduje také **kreativní přístup**. Důležité je navrhnout poutavý herní koncept, zajímavé mechaniky a atraktivní vizuální styl, který hráče vtáhne do hry. Klíčovou roli hraje také **herní design**, který definuje pravidla, cíle a způsob, jakým hráč interaguje s herním světem. [4] Všechny tyto prvky musí být navrženy s ohledem na intuitivní dotykové ovládání a zároveň optimalizovány tak, aby hra plynule běžela i na levnějších mobilních telefonech se slabším hardwarem, nebo telefonech se starším operačním systémem.

Kromě zábavy je často cílem hry i **monetizace**, tedy způsob, jak z hráčů získat příjmy. Mezi nejběžnější obchodní modely patří: [3]

- **Prémiové hry** – hráč si zaplatí za plnou verzi hry předem.
- **Free-to-play s mikrotransakcemi** – Nejběžnější způsob a u vývojářů velice oblíben. Hra je zdarma, ale obsahuje nákupy za reálné peníze (např. lepší předměty, vylepšení, měna, ...).
- **Reklamní model** – hráč sleduje reklamy výměnou za odměny nebo nápovědy ve hře.

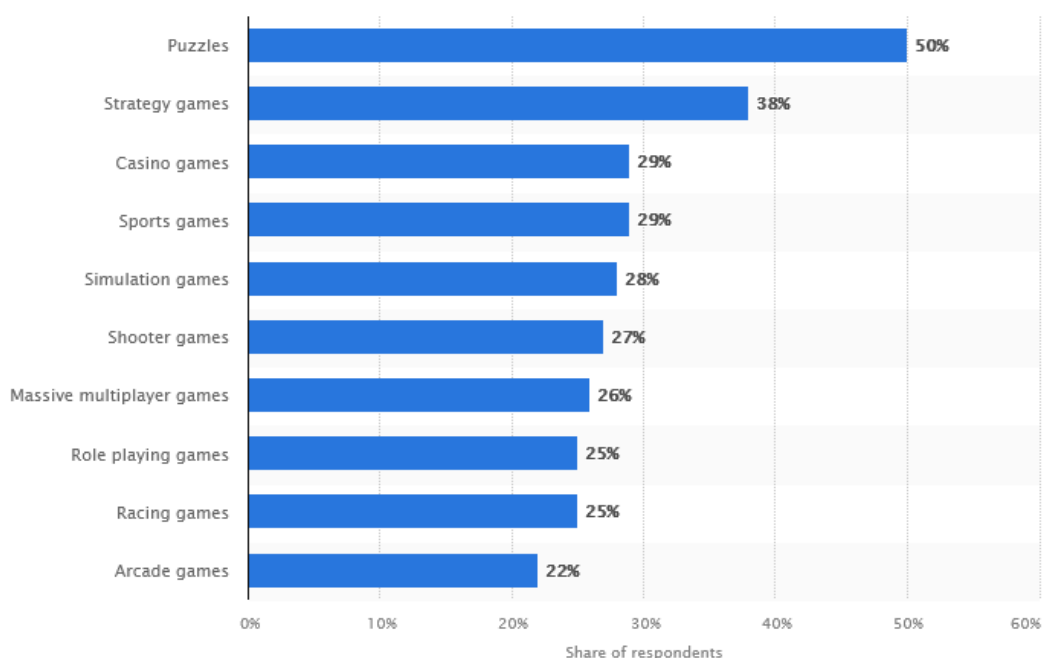
Vývoj her je tedy proces, který spojuje programování, grafiku, zvuk, psychologii hráče i marketing. Úspěšná hra musí nejen **bavit**, ale zároveň být dostatečně **atraktivní a udržitelná**, aby si udržela hráčskou základnu, a navíc přinášela vývojářům zisk.

1.3 Druhy her

Mobilní hry lze rozdělit podle různých kritérií, jako je **žánr, herní mechaniky, způsob ovládání nebo cílová skupina hráčů**. Zatímco některé hry napodobují tradiční herní žánry z PC a konzolí, jiné jsou navrženy speciálně pro mobilní zařízení a jejich unikátní vlastnosti, jako je hlavně dotykové ovládání.

1.3.1 Herní žánr

Jedním z nejdůležitějších kritérií pro rozdělení her je jejich žánr, který ovlivňuje nejen herní mechaniky, ale i způsob, jakým hráč s hrou interaguje. Každý žánr přináší odlišný herní zážitek – zatímco logické hry kladou důraz na řešení hádanek a strategické myšlení, akční hry vyžadují rychlé reflexy a přesné ovládání. Vývojáři her si proto pečlivě vybírají herní prvky a mechanismy tak, aby co nejlépe odpovídaly danému žánru a zároveň zaujaly cílovou skupinu hráčů.



Obrázek 1. Nejčastější žánry mobilních her za rok 2023.

(Zdroj: [5])

Podle statistik za rok 2023 z obrázku výše (Obrázek 1) jsou mezi hráči nejvíce oblíbené **logické hry (puzzle games)**, které hraje až **50 % mobilních hráčů**. Tento typ her je atraktivní svou jednoduchostí a nenáročností na výkon zařízení. Mezi typické zástupce patří hry jako **Candy Crush Saga** nebo **Sudoku** – Zdroj: obchod Google Play, kategorie vyhledání: Puzzle.

Dalším velmi populárním žánrem jsou **strategické hry (strategy games)**, které si oblíbilo **38 % hráčů**. Tyto hry často vyžadují promyšlené plánování a taktické rozhodování. Mobilní platforma nabízí jak tahové strategie typu **Clash of Clans**, tak i real-time strategie jako **Clash Royale**, které hráčům umožňují soupeřit s ostatními hráči v krátkých, ale za to dynamických soubojích – Zdroj: obchod Google Play, kategorie vyhledání: Strategy.

S **29% podílem** jsou na třetím místě **kasínové hry (casino games)** a **sportovní hry (sports games)**. Kasínové hry zahrnují jak **poker**, tak i různé automaty, které simulují skutečné herní prostředí. Sportovní hry zahrnují simulátory fotbalu, basketbalu nebo automobilových závodů, ale patří zde i různé golfové nebo kulečnickové hry. Mezi nejznámější patří série **EA SPORTS FC™ MOBILE SOCCER** (dříve FIFA mobile) – Zdroj: obchod Google Play, kategorie vyhledání: Sports.

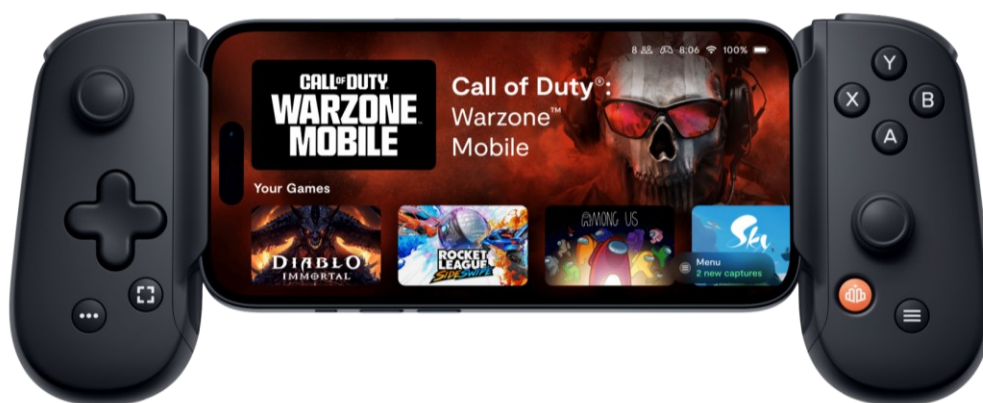
Mezi další populární žánry patří **simulační hry (28 %)**, **střílečky (27 %)**, **MMO (masivní online multiplayer) hry (26 %)**, **RPG (hra na hrdiny) hry (25 %)**, **závodní hry (25 %)** a **arkádové hry (22 %)**. Tyto kategorie pokrývají široké spektrum herních stylů –

od budovatelských a farmářských simulátorů až po intenzivní akční střílečky typu **Call of Duty Mobile**. RPG hry se soustředí na vývoj postavy a příběhové linie, zatímco závodní hry jako je série her **Asphalt** nabízejí realistické i arkádové zážitky. Arkádové hry pak obvykle spoléhají na rychlé reakce hráčů a jednoduchou, ale návykovou hratelnost, jako je tomu například u hry **Fruit Ninja**.

1.3.2 Ovládání her

Způsob ovládání hraje na mobilních zařízeních zásadní roli. Většina her je navržena tak, aby plně využívala dotykovou obrazovku, ať už jde o jednoduché kliknutí na objekt či tlačítko nebo složitější gesta a digitální joystick, například aktivní pohyb prstem pro ovládání postavy nebo auta. Některé tituly ale nabízejí i možnost pohybového ovládání pomocí gyroskopu, kdy hráč naklání telefon, aby například zatačel v závodní hře.

Dalším způsobem ovládání her jsou v dnešní době stále populárnější specializované herní kontroléry a držáky s přidanými tlačítky, které výrazně zlepšují herní zážitek. Tyto fyzické ovladače jsou oblíbené především mezi hráči akčních stříleček, závodních simulátorů a dalších her, kde přesné ovládání hraje klíčovou roli. K dispozici jsou jak plnohodnotné **Bluetooth gamepady**, které se podobají těm pro konzole PlayStation nebo Xbox, tak i **kompaktní mobilní ovladače**, které se přímo připojují k telefonu a rozšiřují jeho možnosti ovládání bez nutnosti dotýkat se displeje.



Obrázek 2. Backbone ovladač pro mobilní telefony.

(Zdroj: [6])

1.4 Hry pro jednoho hráče

Hlavním benefitem her pro jednoho hráče (singleplayerových her) je možnost hrát hru ve svém vlastním tempu bez nutnosti čekání na ostatní hráče nebo obav, že by mohl být herní

zážitek narušen chováním ostatních hráčů. Klíčovým aspektem těchto her je jejich **příběh**, který musí hráče zaujmout a motivovat je k delšímu hraní.

Další výraznou výhodou her pro jednoho hráče je jejich dostupnost. Většina těchto her nevyžaduje internetové připojení, což z nich činí ideální volbu pro hraní na cestách nebo v situacích, kdy hráč nemá stabilní připojení k internetu. Existují však i singleplayerové hry, které sice staví svůj hlavní obsah na offline zážitku, ale s dodatečným připojením k internetu nabízejí rozšířené funkce. Například ve hře **SimCity BuildIt** mohou hráči obchodovat s ostatními hráči po celém světě nebo synchronizovat svůj postup mezi zařízeními.

1.5 Hry pro více hráčů

Hry pro více hráčů, často označované jako **multiplayer**, patří mezi nejpobulárnější formy digitální zábavy. Na rozdíl od **singleplayerových** her, kde hráč hraje obecně sám nebo proti umělé inteligenci, multiplayer umožňuje soupeření (**kompetitivní režim**) nebo spolupráci (**kooperativní režim**) mezi reálnými hráči. Díky neustálému rozvoji internetu a mobilních technologií se multiplayerové hry staly nedílnou součástí mobilního hraní, přičemž některé tituly se na tento režim zaměřují výhradně.

Hry pro více hráčů můžeme dále rozdělit na 2 celky:

- **Online multiplayer** – neboli hra více hráčů využívající internetové připojení pro hraní hry.
- **Lokální multiplayer** – hry pro více hráčů, které naopak nevyžadují internetové připojení. Hráči jsou propojeni mezi svými zařízeními navzájem díky jejich fyzické blízkosti.

1.5.1 Motivace hráčů

Jako u většiny multiplayer her má každý hráč svou vlastní motivaci – často psychologickou. Ta se může lišit podle **herního žánru** nebo individuální preferencí. Například v logických hrách, jako jsou šachy, je hráč motivován k tomu, aby **porazil soupeře**. Aby toho dosáhl, musí se učit nové techniky, zdokonalovat ty stávající nebo si osvojit historicky úspěšné tahy.

Jednou z hlavních motivací v multiplayer hrách tedy může být **poražení soupeřů**. Hra tuto motivaci často posiluje dodatečným obsahem pro hráče, například speciálními odměnami nebo zobrazením jeho jména v globálním žebříčku nejlepších hráčů. [7] Jiné hry, jako například **Mobile Legends: Bang Bang** z prostředí online arény, umožňují hráčům

inspirovat se nastavením postav od nejlepších hráčů dané postavy. Dalším prvkem mohou být odznaky u jména hráče, které značí jeho herní úspěchy a mohou tak dále zvyšovat jeho motivaci.

Mezi další motivaci hráče můžeme zařadit jeho samotné **zlepšování v dané aktivitě** nebo hře. [7] Jedná se o obecnou schopnost každého člověka, který chce dosáhnout svých cílů. S pokrokem a učením zvyšujeme šanci na úspěch, který je doprovázen radostí a pocitem dobře odvedené práce, na který by hry neměli zapomínat a hráče by proto měli patřičně odměnit, čímž si hráče ve hře dýl udrží.

Další klíčovou motivací pro hraní multiplayer her je **sdílení zábavy s přáteli**. Hry by na to měly být připraveny například možnostmi zakládání cechů, seznamem přátel nebo speciálními herními módy určenými pro společné hraní. [7]

1.6 Online multiplayer

Online hry se staly fenoménem, který formuje celý mobilní herní průmysl. Mnoho populárních titulů je dnes navrženo primárně s důrazem na online interakci, ať už jde o akční hry, strategie nebo sociální simulace a hráči tak mohou mezi sebou soutěžit nebo sdílet herní zážitky s přáteli, ale i cizími lidmi z celého světa.

1.6.1 Výzvy při vývoji

Hraní her je zábavná činnost, jejich vývoj ale přináší mnoho výzev, které mohou být jak technického, tak i organizačního či hráčského charakteru. Vývojáři se musí vypořádat nejen s optimalizací síťové infrastruktury a serverového zázemí, ale také s chováním samotných hráčů, kteří často hledají způsoby, jak získat ne vždy legální výhodu proti ostatním hráčům.

Jednou z hlavních výzev je **zneužívání chyb ve hře**. Hráči se často snaží najít **exploity neboli chyby ve hře**, které jim poskytnou výhodu nad ostatními, a ne vždy na ně vývojáře upozorní. Závažnějším problémem je pak přímé **cheatování**, kdy hráči pomocí speciálních programů nebo úprav herního kódu manipulují s chováním hry. I když je tento problém častější u **počítačových her**, nevyhýbá se ani mobilním titulům. Ze sociálních aspektů jde pak také o interakci mezi hráči. Pokud hra obsahuje například herní chat, měl by obsahovat různé filtry pro kontrolu spamu či nadávek a možnost kontrolovaného nahlašování hráčů.

Problémy s podváděním hráčů lze vyřešit technikami jako je **ověřování stavu hry na serveru**, použití externích „anti-cheat“ softwarů pro analýzu chování hráče, nebo již zmíněný systém pro nahlašování samotnými hráči. [8]

Do technických výzev se pak musí počítat s tím, že online hry vyžadují **kvalitní a stabilní síťovou infrastrukturu**, která umožňuje rychlou komunikaci mezi hráči.

1.6.2 Synchronizace a komunikace mezi hráči

Jeden ze základních pilířů online her je synchronizace a komunikace mezi hráči. Oproti hrám pro jednoho hráče, kde probíhá veškerá herní logika lokálně, je v multiplayeru nutné neustále synchronizovat herní prvky mezi klienty a serverem v co nejkratším čase. To zahrnuje nejen samotné objekty a jejich pohyby, ale také herní tahy a události, kterými hráči interagují se hrou i mezi sebou navzájem.

Jedním z nejčastěji používaných modelů synchronizace je architektura **Client-Server**. V tomto modelu **klienti odesílají na server** zprávy obsahující jejich herní akce, které server zpracuje a validuje. Díky tomu má server plnou kontrolu nad herním stavem a logikou hry, čímž může nejen zajistit konzistenci mezi všemi hráči, ale také detekovat a předcházet podvádění. Jakmile **server akce ověří, rozešle změny** ostatním hráčům, aby všichni viděli aktuální stav hry. Tento model je oblíbený především díky své spolehlivosti a možnosti centrálního řízení hry díky **autoritativnímu režimu serveru**. [8] Z této architektury však vyplývá i jedna z jejích hlavních nevýhod – vysoké nároky na síťovou infrastrukturu i na server, který musí zpracovávat veškerou herní logiku, synchronizovat data mezi hráči a zároveň udržovat stabilní výkon i při velkém množství připojených klientů.

Častým problémem online her je navíc zpoždění neboli **latence**, která nastává v komunikaci se serverem a jejím zpracováním. Aby hra působila plynule i při vyšších hodnotách latence, využívají se různé techniky **predikce a interpolace**. Jednou z nejběžnějších metod je **client-side predikce**, kde klient na základě posledních známých informací předpokládá, co se stane dál, a zobrazí to hráči okamžitě. Pokud později server pošle jiný výsledek, hra provede korekci a upraví stav tak, aby odpovídal realitě. Tento přístup se často využívá například u pohybu postav nebo aut – místo čekání na potvrzení od serveru hra předpokládá, že pohyb bude pokračovat stejným směrem a rychlostí. Pokud dojde k rozdílu mezi predikcí a skutečným stavem, provede se **interpolace**, aby změna nepůsobila tak rušivě. [8]



Obrázek 4. Snímek zobrazující připojování do herní místnosti.

(Zdroj: vlastní, hra *Mobile Legends: Bang Bang*)

Obrázek výše (Obrázek 4) zobrazuje herní obrazovku v mobilní hře **Mobile Legends: Bang Bang**, která slouží k stabilizaci připojení 10 hráčů, rozdělených do dvou týmu po 5 hráčích. Po úspěšném spojení požadovaného počtu hráčů jsou hráči přesunuti do místnosti (**room**), kde si mohou vybrat herní postavu. Po uplynutí stanoveného času jsou následně přesunuti přímo do hry.

2 LOKÁLNÍ MULTIPLAYER

Lokální multiplayer je hlavním tématem této práce. Podobně jako **deskové hry**, které se hrají v jedné místnosti s přáteli či rodinou, probíhá i hraní her s **lokálním multiplayerem** obvykle na jednom místě. Hlavní rozdíl spočívá v tom, že namísto fyzických herních prvků se využívají **mobilní technologie** – hráči hrají na svých **telefonech, tabletech**, a v některých případech mohou ke sdílení jedné herní plochy využít i **televizi**.

2.1 Způsoby připojení

Pro správné fungování lokálního multiplayeru je klíčové, aby mezi sebou hráči mohli **spolehlivě komunikovat**. Mobilní hry využívají různé metody propojení, které se liší **rychlostí, dosahem a spolehlivostí**.

První 2 způsoby popisují propojení několika zařízení mezi sebou, čímž se vyznačují hry, které vyžadují, aby měl každý hráč své vlastní zařízení. Tímto způsobem se obvykle jedno zařízení nastaví jako **host/server**, ke kterému se připojí ostatní – **klienti**. Na základě druhu hry se pak může hostovací zařízení chovat jako další hráč, nebo jen jako spojovatel, zobrazující například herní plochu.

- **Wi-Fi:** Nejčastěji používaný způsob propojení v lokálním multiplayeru. Hráči mohou být připojeni ke **stejně Wi-Fi síti** nebo může jeden z hráčů vytvořit **hotspot**, ke kterému se ostatní připojí. Zajišťuje nejspolehlivější a nejrychlejší připojení.
- **Bluetooth:** Bluetooth umožňuje přímé spojení mezi zařízeními bez nutnosti Wi-Fi sítě. Je vhodné hlavně pro jednodušší hry, které nepotřebují vysokou přenosovou rychlost.

Na rozdíl od prvních dvou způsobů propojení, které vyžadují vlastní zařízení pro každého hráče, sdílená obrazovka a hot-seat režim umožňují hrát více hráčům na jednom zařízení. Tyto způsoby tedy nevyžadují žádné připojení ostatních zařízení. [9]

- **Hot-seat:** Tento režim spočívá v tom, že se **hráči postupně střídají u jednoho zařízení**, například u strategií a deskových hrách. Aktuální hráč odehraje svůj tah a pak předá zařízení dalšímu hráči v řadě.
- **Split-screen:** Sdílená obrazovka naopak rozděluje herní plochu mezi více hráčů, kteří tak hru hrají současně. Nevýhodou těchto her je poměrně omezený prostor jak na displeji, tak i kolem fyzického zařízení.

2.2 Porovnání online vs lokální multiplayer

Z předchozích kapitol jsme zjistili, jakým způsobem lze hry více hráčů rozdělit. Nyní se zaměříme na jejich přímé porovnání, ve kterém si popíšeme hlavní rozdíly technických požadavků a jednotlivé výhody a nevýhody.

2.2.1 Technické požadavky

Online multiplayer vyžaduje stabilní internetové připojení a přístup k herním serverům, které spravují herní stav a synchronizují data mezi hráči. Tato komunikace probíhá prostřednictvím síťových protokolů, jako jsou TCP a UDP, které zajišťují přenos dat mezi hráči v téměř reálném čase. [8] Pro zajištění plynulosti hry je důležitá nízká latence, avšak ta může být ovlivněna geografickou vzdáleností mezi hráči a servery, což může způsobit zpoždění a zhoršení herního zážitku.

Na druhé straně **lokální multiplayer** funguje bez potřeby připojení k externím serverům. Hráči jsou připojeni prostřednictvím technologií jako Bluetooth nebo Wi-Fi. Tato připojení probíhají v rámci lokální sítě, což minimalizuje latenci a zajišťuje **plynulý herní zážitek**, jelikož data nemusí cestovat na dlouhé vzdálenosti.

2.2.2 Výhody a nevýhody

Jednou z největších **výhod online her** je schopnost hráčů propojit se s ostatními hráči v globálním měřítku. Tento aspekt totiž umožňuje hráčům interagovat s širokou herní komunitou, čímž vznikají dynamické herní světy, které mohou hráče spojit klidně i na několik let. Online hry umožňují různé herní režimy, jako jsou kooperativní mise, soutěže nebo PvP (hráč vs hráč), což vytváří živé a neustále se vyvíjející herní prostředí. Hráči mohou komunikovat, spolupracovat nebo se navzájem vyzývat, což podporuje různé sociální interakce a vztahy. Populární a úspěšné hry jsou pravidelně aktualizovány a obohacovány o nový obsah, čímž udržují hráče aktivní po delší dobu.

I když **online hry** nabízí široké spektrum možností, jejich **implementace a provoz jsou technicky náročné a drahé**. Z hráčského hlediska je dále **nevýhod online her několik**. Často se v online hrách můžeme setkat s toxickými hráči, kteří kazí atmosféru ostatním hráčům, chovají se nevhodně, podvádějí, nadávají nebo dokonce šikanují. Častým problémem úspěšných her zdarma jsou i **mikrotransakce**. Tento model, označovaný jako **pay-to-win**, může vést k nevyváženosti hry, kde hráči ochotní utracet reálné peníze (běžně

tisíce korun za balíček) získávají nespravedlivou výhodu nad ostatními. To může navíc vést i k vybudování závislosti na dané hře, což může negativně ovlivnit osobní život hráče.

Na rozdíl od online her, **lokální multiplayer** nevyžaduje takřka žádnou síťovou infrastrukturu, čímž zaručuje okamžitou a stabilní synchronizaci mezi hráči. **Je vhodný** zejména pro rodiny a přátele, kteří si spolu můžou užít zábavné momenty při společném hraní.

Na druhou stranu, lokální multiplayer **přináší určité limity**. Jedná se především o to, že hráči musí být fyzicky přítomní ve stejné místnosti, což vyžaduje určitou organizaci a dostatečně velký prostor okolo. Lokální hry se zaměřují především na sociální interakce mezi kamarády a rodinou, což značně omezuje žánry takových her. Navíc je potřeba, aby měl každý člen hry své vlastní zařízení (záleží na typu hry).

2.3 SongPop Party

SongPop Party je jednou z klíčových her, která inspirovala autora této práce k vytvoření pokerové hry využívající prvky lokálního multiplayeru. Tato hra, vyvinutá primárně pro platformu Apple, je optimalizovaná pro zařízení Apple TV, které slouží jako centrální herní plátno. Ostatní hráči se připojují prostřednictvím svých iPhonů nebo iPadů, které fungují jako ovladače pro interakci se hrou.

Jak už název napovídá, SongPop Party je **hudební kvízová hra**, která staví na rychlém a soutěživém hádání písniček. Hráčům je během hry přehrána krátká ukázka náhodně vybrané skladby a jejich úkolem je co nejrychleji určit správný název písně nebo interpreta. K odpovídání používají jednoduché gesto – tažením prstu po displeji svého zařízení vybírají jednu ze čtyř nabízených možností.

Hra podporuje jak **lokální multiplayer**, tak **online režim**, kde hráči mohou soupeřit s přáteli nebo s náhodnými protivníky z celého světa. Různé herní módy a bohatá databáze skladeb rozdělených do tematických playlistů (např. rock, pop, 80. léta, filmová hudba) zajišťují velikou rozmanitost hry. [10]

Tento koncept kombinující **hudební hádanky, rychlé reakce a sociální interakce** představuje zajímavý příklad využití lokálního multiplayeru v moderních hrách. Zároveň ukazuje, jak lze efektivně propojit **mobilní zařízení s centrálním herním plátnem**, což může být inspirací i pro jiné herní žánry.

3 NEARBY CONNECTION API

Nearby Connections API je pokročilá knihovna umožňující bezdrátovou komunikaci mezi zařízeními bez nutnosti připojení k internetu. Vychází z konceptu původního Nearby Messages API, které bylo v roce 2023 označeno jako zastaralé (deprecated). Knihovnu je možno využít na širokém spektru zařízení a operačních systémech, což následně dovoluje vývojářům její plnohodnotné využití pro jejich potřeby. [11]

3.1 Hlavní vlastnosti

Díky různým možnostem připojení – Bluetooth, BLE (úsporná varianta Bluetooth), a Wi-Fi je tato knihovna vhodná pro široké spektrum aplikací, jako například: [11]

- **Sdílení souborů** – rychlý a bezpečný přenos souborů mezi zařízeními bez nutnosti internetu.
- **Sdílení plochy** – umožňuje vzdálené ovládání nebo sdílení obrazovky.
- **Dálkový ovladač** – použití jednoho zařízení k ovládání jiného (např. smart TV, prezentace).
- **Lokální hry více hráčů** – umožňuje vytváření her s více hráči.

3.2 Strategie připojení

Knihovnu jako takovou lze v závislosti na řešeném problému rozdělit do 3 kategorií, které se nazývají **Strategie**. S výběrem dané strategie se vývojářům odemknou různé možnosti v použití knihovny. [12]

3.2.1 Klastr

Cluster je strategie, která umožňuje komunikaci **M-to-N**. To znamená, že jedno zařízení se může připojit k několika dalším zařízením a naopak. Výsledkem tedy může být síť, kde mohou všechna zařízení komunikovat přímo mezi sebou. Díky tomu, že může tento typ strategie obsahovat velký počet spojů je rychlost přenosu pomalejší a více náročná na režii komunikace.

3.2.2 Klient-Klient

Strategie *Point-to-Point* je naopak strategií **1-to-1**, znamenající pouze jedno přímé spojení s druhým zařízením v jeden čas. Jedná se o rychlou a bezpečnou komunikaci mezi dvěma zařízeními, často vhodnou například pro přenos souborů.

3.2.3 Hvězda

Poslední strategie je typu *Star*. Její topologie je **1-to-N**, což znamená propojení jednoho centrálního zařízení s N dalšími. Díky tomu je komunikace v topologii více striktní, což vede k vyšší rychlosti.

3.3 Základy implementace

Práce s knihovnou je rozdělena do dvou hlavních bloků:

- Advertising (inzerování) – Zařízení aktivně vysílá svou dostupnost a čeká na spojení.
- Discovery (objevení) – Zařízení vyhledává inzerující zařízení a pokouší se k nim připojit.

a navazování spojení je ve dvou fázích: [11]

1) Připojování (pre-connection)

- Inzerující zařízení spustí režim **Advertising** a čeká na žádosti o připojení.
- Objevující zařízení spustí **Discovery** a hledá inzerující zařízení.
- Jakmile objevující zařízení najde inzerenta, odešle žádost o připojení.
- Inzerující zařízení může žádost přijmout nebo odmítnout.

2) Navázané spojení (post-connection)

- Jakmile je spojení schváleno oběma stranami, zařízení mohou začít komunikovat.
- Výměna dat probíhá pomocí tzv. Payload (byty, soubory, streamy).

Předtím, než ale můžeme navázat spojení, jsou na zařízení s OS Android potřebná povolení z následující tabulky (Tabulka 1).

Tabulka 1. Přehled oprávnění pro Nearby Connections API.

Název oprávnění	Min. SDK	Max. SDK	Popis
ACCESS_WIFI_STATE	-	31	Umožňuje aplikaci získat informace o Wi-Fi připojení.
CHANGE_WIFI_STATE	-	31	Umožňuje aplikaci měnit stav Wi-Fi (např. zapnout/vypnout).
BLUETOOTH	-	30	Umožňuje aplikaci používat Bluetooth pro spojení se zařízeními.
BLUETOOTH_ADMIN	-	30	Umožňuje správu Bluetooth připojení, včetně párování zařízení.
ACCESS_COARSE_LOCATION	-	28	Umožňuje získat přibližnou polohu zařízení.
ACCESS_FINE_LOCATION	29	31	Přesná poloha, nutná pro skenování zařízení v okolí.
BLUETOOTH_ADVERTISE	31	-	Umožňuje zařízení inzerovat svou dostupnost.
BLUETOOTH_CONNECT	31	-	Oprávnění pro připojení k jiným zařízením.
BLUETOOTH_SCAN	31	-	Potřebné pro vyhledání a připojení k zařízením.
NEARBY_WIFI_DEVICES	32	-	Potřebné pro přímou komunikaci s Wi-Fi zařízeními v okolí bez internetu.
READ_EXTERNAL_STORAGE	-	-	Potřebné pro čtení souborů. *nepovinné

(Zdroj: [13])

Jelikož jsou oprávnění ACCESS_FINE_LOCATION, BLUETOOTH_ADVERTISE, BLUETOOTH_CONNECT, BLUETOOTH_SCAN a READ_EXTERNAL_STORAGE považována za **nebezpečná**, musí být uživatelem schválena **za běhu aplikace** prostřednictvím systému oprávnění Androidu. [13]

3.3.1 Jednoduchá Kotlin implementace

V programovacím jazyce Kotlin je implementace této knihovny poměrně jednoduchá. Knihovna Nearby Connections funguje na principu event-driven programování pomocí

Listenerů (posluchačů), kteří aktivně naslouchají příchozím zprávám a událostem. Jak bylo zmíněno výše (3.3), knihovna pracuje s dvěma hlavními rolemi – inzerování a objevování. Na to je po celou dobu vývoje nutné myslet a přizpůsobit k tomu logiku svého programu.

Abychom knihovnu mohli ve svém projektu použít, je nutné přidat závislost do souboru **build.gradle.kts** následovně:

```
implementation 'com.google.android.gms:play-services-nearby:VERSION'
```

1) Inzerování (Advertising)

Inzerování lze chápat jako **serverovou část**, ke které se připojují další zařízení.

```
private val connectionsClient = Nearby.getConnectionsClient(context)
fun startAdvertising() {
    val advertisingOptions = AdvertisingOptions
        .Builder()
        .setStrategy(Strategy.P2P_STAR)
        .build()
    connectionsClient.startAdvertising("Server", SERVICE_ID,
        connectionLifecycleCallback, advertisingOptions)
}
```

Kód výše vytváří proměnnou **connectionsClient**, která v systému hraje důležitou roli, neboť v sobě udržuje veškerý stav. Dále základní funkce **startAdvertising** obsahuje kód pro definici inzerování a pomocí funkce *connectionsClient.startAdvertising()* zapíná hlavní jádro knihovny, které se stará o naslouchání.

2) Objevování (Discovery)

Objevování umožňuje **aktivně vyhledávat** dostupná inzerující zařízení a pokusit se k nim připojit.

```
fun startDiscovery() {
    val discoveryOptions = DiscoveryOptions
        .Builder()
        .setStrategy(Strategy.P2P_STAR)
        .build()
    connectionsClient.startDiscovery(SERVICE_ID,
        endpointDiscoveryCallback, discoveryOptions)
}
```

Funkce **startDiscovery** je vytvořena podobným způsobem jako funkce **startAdvertising**. Je důležité, aby obě funkce pracovaly se **stejnou strategií a SERVICE_ID** (pro tuto volbu se obvykle používá název aplikace). [14]

3) Popis objektů

EndpointDiscoveryCallback je objekt využívaný při hledání dostupných zařízení k připojení. Registruje se pouze ve funkci **startDiscovery()** a vyžaduje implementaci následujících funkcí: [15]

- **onEndpointFound(endpointId: String, info: DiscoveredEndpointInfo)**
 - Volá se, když je nalezen inzerující (advertising) endpoint.
 - V tomto místě žádáme o připojení – funkce `connectionsClient.requestConnection()`.
- **onEndpointLost(endpointID: String)**
 - Zavolá se, když se zařízení ztratí nebo přestane inzerovat.

ConnectionLifecycleCallback je objekt, který spravuje životní cyklus spojení mezi zařízeními. Registruje se jak ve funkci **startAdvertising()** tak i ve funkci **requestConnection()** v části objevování, a proto si jsou ve velké části podobné. Tento objekt dědí od stejnojmenné abstraktní třídy a tím vyžaduje implementaci 3 hlavních funkcí: [15]

- **onConnectionInitiated(endpointId: String, conInfo: ConnectionInfo)**
 - Volá se, když zařízení zahájí připojení.
 - Zde můžeme **ručně přijmout nebo odmítnout** připojení.
 - V případě akceptování vytváříme další funkci: **payloadCallback**.
- **onConnectionResult(endpointId: String, result: ConnectionResolution)**
 - Volá se po potvrzení připojení.
 - Zde zjistíme, zda bylo spojení **úspěšné** nebo **neúspěšné** (`result.status.isSuccess`).
- **onDisconnected(endpointID: String)**
 - Zavolá se, když zařízení ztratí spojení nebo se odpojí.

PayloadCallback je posledním důležitým objektem, který zpracovává **příchozí** zprávy. Tento objekt je opět využitý v obou rolích, tzn., že je jeho implementace vyžadována jak u inzerenta, tak i u objevovatele. Payload objekt podporuje 3 typy data k přenosu:

1. **Byty** – malá data ve formě *ByteArray* (textové zprávy)

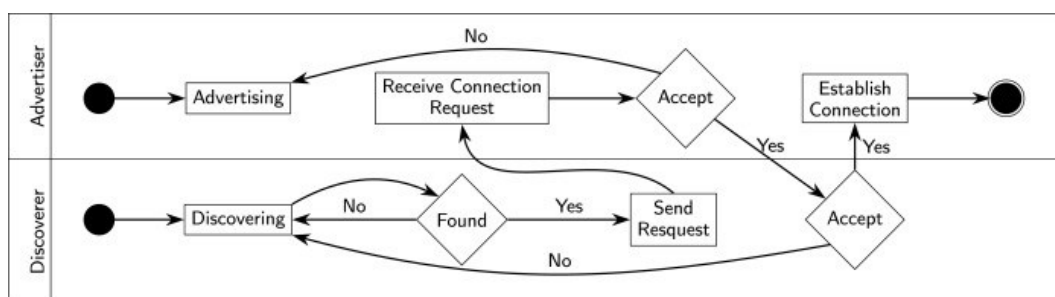
2. **Soubory** – vybrané soubory z úložiště zařízení (fotky, dokumenty, ...)

3. **Streamy** – datové streamy v podobě živého video či audio přenosu.

Ke své činnosti vyžaduje implementaci dvou klíčových funkcí: [16]

- **onPayloadReceived(endpointId: String, payload: Payload)**
 - Volá se v momentě, kdy zařízení přijme nový payload.
 - Pokud jde o **soubor** nebo **stream**, zařízení ho ještě **nemusí mít kompletně stažený**.
- **onPayloadTransferUpdate(endpointId: String, update: PayloadTransferUpdate)**
 - Lze využít ke sledování průběhu přenosu dat.
 - Hodnota *update.status* může obsahovat 4 stavy:
 - SUCCESS – přenos zprávy byl úspěšně vykonán.
 - FAILURE – přenos selhal.
 - IN_PROGRESS – přenos dat probíhá.
 - CANCELED – přenos byl předčasně ukončen a selhal.

Následující obrázek (Obrázek 5) nám zobrazuje zjednodušený tok a základní fáze knihovny Nearby Connections, které zde byly popsány.



Obrázek 5. Flow diagram navázání komunikace s Nearby Connections API.

(Zdroj: [17])

3.3.2 ConnectionsClient

Již zmíněná proměnná **connectionsClient** je úzce spojena s obsluhou knihovny a je důležité, aby během běhu aplikace existovala **pouze jedna instance**, se kterou se pracuje. Vytváření více instancí totiž může vést k nekonzistentnímu chování nebo neočekávaným odpojením, protože každá nová instance může narušit existující spojení.

Nejlepší praxí je zajistit, aby byl **connectionsClient** spravován centrálně a konzistentně po celou dobu běhu aplikace. Proto se doporučuje jeho správa pomocí **Dependency Injection**, což umožňuje jeho snadné poskytování do různých tříd bez nutnosti ruční inicializace a zároveň eliminuje riziko vytváření více instancí.

3.4 Konkurence

Knihovna jako Nearby Connections nemá příliš konkurence, jelikož je unikátní svou vysokou úrovní abstrakce, která značně usnadňuje proces komunikace mezi zařízeními. Většina konkurenčních řešení spadá do kategorie **low-level API**, což znamená, že vývojář musí implementovat větší část logiky sám.

Například již zmíněné **Bluetooth** nebo **Wi-Fi Direct**, které Nearby Connections API zabaluje, může vývojář implementovat přímo dle vlastních potřeb. Výhodou tohoto přístupu je větší flexibilita – například **BLE** je ideální pro nenáročnou komunikaci se senzory nebo fitness náramky. Na druhou stranu však vyžaduje složité nastavování, správu připojení a optimalizaci výkonu. [18]

V oblasti herního vývoje se jako alternativa může zmínit **Unity Netcode for GameObjects** nebo **Mirror**, které poskytují síťovou komunikaci pro multiplayer hry. [8] Nicméně tyto frameworky jsou navrženy primárně pro online komunikaci a nejsou optimalizovány pro offline spojení mezi mobilními zařízeními.

V praxi tak pro lokální komunikaci mezi mobilními zařízeními zatím Nearby Connections API nemá přímou konkurenci, protože žádné jiné řešení nenabízí stejně jednoduchou a výkonnou implementaci napříč různými síťovými technologiemi.

4 POKER

Poker je jednou z nejpopulárnějších **karetních** her na světě, která kombinuje **strategii, psychologii a pravděpodobnost**. Vzhledem ke svému charakteru je často spojován s hazardem a ve své sázkové formě je dostupný pouze zletilým osobám, které nesou odpovědnost za případnou ztrátu peněz.

Tato práce se však zaměřuje na poker výhradně jako formu zábavy, bez jakéhokoli spojení s hazardní činností. Autor si je vědom, že poker může být v některých případech vnímán jako hazardní hra, nicméně cílem této práce je představit jej jako společenskou hru určenou pro rekreační účely.

Jeho nejznámější varianta, **Texas Hold'em**, se hraje se standardním balíčkem čítající 52 **francouzských** karet, které se vyznačují čtyřmi druhy symbolů/barev (srdce, káry, piky, kříže) s hodnotami 2, 3, 4, 5, 6, 7, 8, 9, 10, J (kluk), Q (královna), K (král), A (eso). Cílem hráče je vyhrát sázky uložené během hry buď tím, že mají **silnější karetní kombinaci**, nebo že přimějí soupeře své karty složit tzv. **blafováním**.



Obrázek 6. Balíček francouzských karet.

(Zdroj: vlastní)

4.1 Vysvětlení pojmů

Poker je velmi rozmanitá hra s vlastní terminologií, která se často používá během hry i v diskuzích mezi hráči. Pro porozumění pravidlům a strategiím je důležité znát klíčové pojmy, které se ve hře nejčastěji vyskytují.

Následující seznam obsahuje především ty termíny, které hráči běžně používají při hře a které jsou zásadní pro správné pochopení herních mechanik. Všechny názvy jsou uvedeny v anglickém jazyce, protože v pokerové komunitě jsou standardem a mnohdy ani nemají český ekvivalent. [19]

- **Check** – Pokračování ve hře bez vsazení, pokud žádný hráč předtím nevsadil.
- **Bet** – První sázka v daném kole.
- **Call** – Dorovnání sázky jiného hráče vložením stejné částky do banku.
- **Raise** – Navýšení již existující sázky jiného hráče.
- **Fold** – Složení karet a vzdání se účasti v dané hře.
- **Pot (Bank)** – Součet všech žetonů vsazených v aktuální hře, o který hráči soutěží.
- **All-in** – Vsazení všech svých zbývajících žetonů.
- **Dealer** – Hráč, který rozdává karty (v kasinech tuto roli často zastává krupiéř). Pozice dealera se ve hře posouvá ve směru hodinových ručiček.
- **Small Blind & Big Blind** – Povinné sázky umístěné dvěma hráči sedícími po levé ruce dealera před začátkem hry. Small blind obvykle představuje polovinu big blindu.
- **Flop** – První tři společné karty vyložené na stůl.
- **Turn** – Čtvrtá společná karta vyložená na stůl.
- **River** – Pátá a poslední společná karta vyložená na stůl.
- **Community Cards** – Společné karty na stole, které mohou hráči použít k sestavení nejlepší možné kombinace.
- **Hole Cards** – 2 karty každého hráče, které schovává před ostatními hráči.

Znalost těchto pojmů je užitečná nejen při samotné hře, ale také při studiu strategií, sledování pokerových turnajů či čtení odborných článků o pokeru.

Kromě základních pokerových pojmů existují i další termíny, které se běžně používají během hry a mohou mít zásadní vliv na její průběh. Tyto termíny se často vztahují k pokročilým strategiím, rozhodujícím situacím nebo specifickým pravidlům, se kterými se hráči setkávají.

- **Kicker** – Doplnková karta, která rozhoduje při nerozhodném výsledku (např. pokud mají dva hráči stejný pár, vítězí hráč s vyšším kickerem).
- **Showdown** – Fáze, kdy hráči odhalují své karty a určuje se vítěz.
- **Bluff** – Strategie, při které hráč sází nebo navyšuje sázky s horší kombinací, aby donutil soupeře složit lepší karty.
- **Split Pot** – Situace, kdy dva nebo více hráčů mají stejnou výherní kombinaci a bank se mezi ně rozdělí.
- **Ante** – Povinná sázka, kterou **platí všichni hráči** před rozdáním karet (často se používá v turnajích).

4.1.1 Limity sázek

V pokerových hrách dále existují tzv. **limity sázek**, které určují, **kolikrát** a **o kolik** mohou hráči během jedno herní kolo navýšit sázky. Existuje několik variant, přičemž nejčastější jsou: [20]

- **No Limit** – Nejpopulárnější varianta, ve které mohou hráči vsadit libovolné množství žetonů kdykoli během hry. Minimální sázka je však obvykle nastavena na hodnotu **big blindu** a každé navýšení musí být alespoň dvojnásobkem předchozí sázky. Tento formát umožňuje také **All-in**, tedy vsazení všech dostupných žetonů. V některých případech může být No Limit doplněn omezením maximálního počtu navýšení během jednoho kola (Cap Limitem).
- **Fixed Limit** – V této variantě je pevně stanovena nejen výše sázek, ale také maximální počet možných navýšení během kola. Hráči mohou navyšovat pouze o předem určenou částku, která se po celou dobu hry nemění.
- **Pot Limit** – Maximální sázka, kterou hráč může vsadit nebo navýšit, je omezena velikostí aktuálního **banku** (potu). Pokud je v banku například \$20, hráč může navýšit maximálně o dalších \$20. Díky tomu tento limit obsahuje dynamické změny, které vyžadují promyšlené tahy hráčů.
- **Cap Limit** – Varianta, která stanovuje maximální počet navýšení během jednoho kola. Jakmile je dosažen limit, další navyšování již není možné, a hra pokračuje pouze dorovnáváním nebo složením karet.

4.2 Texas Hold'em

Varianta pokeru s názvem Texas Hold'em je jedna z nejznámějších variant kasinových her. Na začátku hry této varianty je vylosován jeden hráč, kterému se přiřadí žeton **dealer**, který označuje hráče rozdávajícího karty v dané hře. Dalšímu hráči po levici dealera je přiřazen žeton **small blind**, který udává, že musí hráč do hry vložit povinnou malou sázku. Druhému hráči nalevo (ve variantě dvou hráčů opět dealer) je přiřazen žeton **big blind**, který přikazuje hráči vložit velkou sázku do hry. Žetony se každou hrou přesouvají na další hráče.

Jakmile je malá a velká sázka vyložena, dealer rozdá **každému hráči 2 karty** a začíná **první sázkové kolo** zvané *pre-flop*, ve kterém začíná hrát hráč nalevo od **big blind**. V dané situaci má hráč na výběr **3 možnosti**:

- 1) **CALL** – Dorovnání aktuální sázky (v tomto případě big blind)
- 2) **RAISE** – Navýšení aktuální sázky (obvykle minimálně dvojnásobek)
- 3) **FOLD** – Zahození karet a ukončení aktuální hry.

Tímto způsobem se pokračuje až ke hráči s žetonem big blind. Pokud žádný z předchozích hráčů velkou sázku nenavýšil, má následující možnosti:

- 1) **CHECK** – Pokračování ve hře bez přihození peněz. Tímto tahem začne nové sázkové kolo.
- 2) **RAISE** – Navýšení velké sázky. Tímto tahem se spustí „nové“ kolo, ve kterém se akce jednotlivých hráčů opakují. Může se jednat o takticky výhodný tah.
- 3) **FOLD** – Nejméně pravděpodobná možnost výběru. Tímto tahem začne nové sázkové kolo.

Další **sázkové kolo flop** je doprovázeno rozdáním 3 společných karet na herní stůl a začíná prvním „nesloženým“ hráčem nalevo od dealera (obvykle hráč s žetonem small blind). V tomto kole není žádná aktuální sázka, takže má první hráč stejné možnosti, jako big blind v kole *pre-flop* (**check**, **raise** s minimální hodnotou velké sázky, **fold**). Tímto způsobem probíhá celé kolo opět až k prvnímu hráči. Pokud v daném kole nikdo nevsadil, popřípadě nenavýšil sázku prvního hráče, kolo končí a první hráč již nemá žádnou možnost.

Třetí sázkové kolo turn přinese na stůl 4. společnou kartu a pokračuje se stejným způsobem jako v předchozím kole.

Poslední 5. společná karta spouští **čtvrté sázkové kolo *river***, které je stejné jako předchozí. V jeho konci musí všichni hráči, kteří zůstali ve hře, vyložit své 2 karty z ruky a hra se na základě karetních kombinací (4.3) vyhodnotí. V případě jednoho výherce je vítěznému hráči předán **bank** celý, pokud je výherců více, tak se bank mezi výherce dělí rovným dílem. [21]

4.3 Karetní kombinace

Ve variantě Texas Hold'em je cílem hráče sestavit co nejsilnější **pěti-karetní kombinaci** z karet, které má k dispozici (2 v ruce a 5 na stole). Kombinace jsou podle pravděpodobnosti hierarchicky seřazeny od nejmenší po největší sílu. Každá karetní kombinace taky může mít více možných variant, přičemž vítězí ta, která obsahuje vyšší karty. Pokud mají hráči stejnou kombinaci se stejnou hodnotou karet, nastává remíza a peníze (bank/pot) se dělí rovným dílem. [21]

Jednotlivé pravděpodobnosti a šance výskytu jsou zdrojem následujících odkazů: [22][23].

4.3.1 Vysoká karta (High Card):

Nastává v situaci, kdy hráč nedokáže sestavit žádnou karetní kombinaci. Seznam 5 nejvyšších karet hráče.

Postup k vítězství: Pokud žádný ze soupeřů nemá lepší kombinaci, posuzuje se nejvyšší karta. V případě rovnosti se přechází na 2. nejvyšší kartu, to se opakuje až do poslední 5. karty. Pokud se rovnají všechny karty, dělí se bank.

Pravděpodobnost výskytu: 17,412 %

Šance výskytu: 1:4,74



Obrázek 7. Ukázka Vysoké karty.

(Zdroj: vlastní)

Na obrázku výše (Obrázek 7) vidíme situaci, kdy žádný ze tří hráčů nesestavil žádnou specifickou karetní kombinaci. O vítězi tedy rozhoduje nejvyšší karta. První dvě nejvyšší karty (K a J) jsou společné, takže rozhoduje třetí nejvyšší karta, kterou drží **Hráč 2** – karta 10. Díky tomu vyhrává bank.

4.3.2 Pár (One Pair)

Nastává v situaci, kdy má hráč dvě karty stejné hodnoty a tři nesouvisející karty.

Postup k vítězství: Pokud mají dva nebo více hráčů pár, vítězí vyšší pár. Při shodě párů se porovnává nejvyšší karta ze zbývajících tří, poté druhá nejvyšší a nakonec třetí. Pokud jsou všechny karty stejné, dělí se bank.

Pravděpodobnost výskytu: 43,822 %

Šance výskytu: 1:1,28

Následující obrázek (Obrázek 8) znázorňuje situaci, kdy dva hráči dosáhli stejné karetní kombinace – **pár desítek**. Jejich druhá držená karta je nižší než všechny společné karty na stole, což znamená, že o vítězi nerozhoduje. V důsledku toho dochází k remíze a bank se mezi oba hráče rovnoměrně rozdělí.



Obrázek 8. Ukázka Páru.

(Zdroj: vlastní)

4.3.3 Dva páry (Two Pair)

Nastává v situaci, kdy má hráč dvě dvojice karet stejné hodnoty a jednu nesouvisející kartu.

Postup k vítězství: Pokud mají dva hráči dva páry, vítězí hráč s vyšším párem. Pokud je shodný, rozhoduje druhý pár. Při rovnosti obou párů rozhoduje pátá karta. Pokud jsou všechny karty stejné, bank se dělí.

Pravděpodobnost výskytu: 23,496 %

Šance výskytu: 1:3,26

Obrázek níže (Obrázek 9) znázorňuje výhru Hráče 3, který sestavil **dva páry** (AA, 77). Ostatní hráči mají pouze společný pár 77 ze stolu, a proto Hráč 3 díky svému vyššímu druhému páru vítězí a bere celý bank.



Obrázek 9. Ukázkou Dvou párů.

(Zdroj: vlastní)

4.3.4 Trojice (Three of a Kind)

Nastává v situaci, kdy má hráč tři karty stejné hodnoty a dvě nesouvisející karty.

Postup k vítězství: Pokud mají dva hráči trojici, vítězí vyšší trojice. Při shodě rozhoduje vyšší ze zbývajících dvou karet, případně druhá zbývající karta. Pokud jsou všechny karty stejné, bank se dělí.

Pravděpodobnost výskytu: 4,829 %

Šance výskytu: 1:19,7



Obrázek 10. Ukázka Trojice.

(Zdroj: vlastní)

Na obrázku (Obrázek 10) vidíme situaci, kdy všichni tři hráči sestavili stejnou karetní kombinaci – **trojici es (AAA)**, která je vytvořena pomocí společných karet na stole. Jelikož žádný z hráčů nemá silnější doplňkovou kartu (kicker), dochází k remíze a bank je rovnoměrně rozdělen mezi všechny hráče.

4.3.5 Postupka (Straight)

Nastává v situaci, kdy má hráč pět po sobě jdoucích karet různých barev.

Postup k vítězství: Pokud mají dva hráči postupku, vítězí hráč s vyšší nejvyšší kartou postupky. Pokud mají shodnou postupku, bank se dělí.

Pravděpodobnost výskytu: 4,619 %

Šance výskytu: 1:20,65



Obrázek 11. Ukázka Postupky.
(Zdroj: vlastní)

Obrázek (Obrázek 11) znázorňuje situaci, ve které **Hráč 1 a Hráč 3 sestavili postupku**, zatímco Hráč 2 má pouze dvojici es (AA). Přestože mají dva hráči stejnou kombinaci, vítězem se stává **Hráč 1**, protože je jeho postupka složena z vyšších karet (2, 3, 4, 5, 6) než u Hráče 3 (eso, 2, 3, 4, 5). Díky tomu získává celý bank.

4.3.6 Barva (Flush)

Nastává v situaci, kdy má hráč pět karet stejné barvy, které netvoří postupku.

Postup k vítězství: Pokud mají dva hráči barvu, rozhoduje nejvyšší karta. Pokud je shodná, přechází se na druhou nejvyšší, případně třetí, čtvrtou a pátou kartu. Pokud jsou všechny karty stejné, bank se dělí.

Pravděpodobnost výskytu: 3,025 %

Šance výskytu: 1:32,05

Na obrázku (Obrázek 12) nastala situace, kdy Hráč 1 a Hráč 3 sestavili stejnou postupku jako v předchozím případě (Obrázek 11). Tentokrát však **Hráč 2** vítězí díky silnější kombinaci – **Barva** v srdcích, která postupku překonává.



Obrázek 12. Ukázka Barvy.

(Zdroj: vlastní)

4.3.7 Full House

Nastává v situaci, kdy má hráč kombinaci trojice a páru.

Postup k vítězství: Pokud mají dva hráči full house, vítězí hráč s vyšší trojicí. Při shodě trojice rozhoduje vyšší pár. Pokud je i ten shodný, bank se dělí.

Pravděpodobnost výskytu: 2,596 %

Šance výskytu: 1:37,52



Obrázek 13. Ukázka kombinace Full House.

(Zdroj: vlastní)

Tentokrát můžeme na obrázku výše (Obrázek 13) vidět stav, ve kterém **Hráč 1** překonává pár Hráče 2 i postupku Hráče 3, a to díky silnější kombinaci **Full House** (555AA). Díky tomu získává Hráč 1 celý bank.

4.3.8 Čtveřice (Four of a Kind / Poker)

Nastává v situaci, kdy má hráč čtyři karty stejné hodnoty a jednu nesouvisející kartu.

Postup k vítězství: Pokud mají dva hráči čtveřici, vítězí vyšší čtveřice. Pokud je stejná, rozhoduje pátá karta. Pokud je i ta shodná, bank se dělí.

Pravděpodobnost výskytu: 0,168 %

Šance výskytu: 1:594



Obrázek 14. Ukázka Čtveřice.

(Zdroj: vlastní)

Nyní obrázek (Obrázek 14) poskytuje pohled na hru, ve které mají Hráč 1 a Hráč 2 dvojici párů (22 a AA) ze společných karet. **Hráč 3** však drží v ruce pár dvojek, který je doplněn zbývajícím párem dvojek z karet na stole, čímž vytváří velice silnou kombinaci **Čtveřice** (2222) a bere tak celý bank.

4.3.9 Postupka v barvě (Straight Flush)

Nastává v situaci, kdy má hráč pět po sobě jdoucích karet stejné barvy.

Postup k vítězství: Pokud mají dva hráči postupku v barvě, vítězí hráč s vyšší nejvyšší kartou postupky. Pokud mají shodnou postupku, bank se dělí.

Pravděpodobnost výskytu: 0,028 %

Šance výskytu: 1:3589,57

Obrázek níže (Obrázek 15) nám ukazuje 2. nejlepší výherní kombinaci **Postupka** v pikové **barvě**, se kterou **Hráč 2** přebíjí barvu Hráče 1 a trojici hráče 3. Díky tomu Hráč 2 vyhrává celý bank.



Obrázek 15. Ukázka kombinace Postupka v barvě.

(Zdroj: vlastní)

4.3.10 Královská postupka (Royal Flush)

Jedná se o **nejsilnější** karetní kombinaci ve hře, která nastává v momentě, kdy má hráč postupku od desítky po eso v jedné barvě.

Postup k vítězství: Pokud má více hráčů královskou postupku (pouze v případě kombinace z 5 karet na stole = extrémně vzácné), bank se dělí.

Pravděpodobnost výskytu: 0,003 %

Šance výskytu: 1:30939



Obrázek 16. Ukázka Královské postupky.

(Zdroj: vlastní)

Poslední ukázková kombinace z obrázku výše (Obrázek 16) nám znázorňuje šťastného **Hráče 1**, kterému se povedla získat nejsilnější karetní kombinace – **Královská postupka** (10, J, Q, K, A, v kárové barvě).

5 ANDROID KNIHOVNA

Knihovny jsou nedílnou součástí moderního softwarového vývoje, protože umožňují sdílení a znovupoužití kódu mezi různými projekty. Umožňují vývojářům soustředit se na specifické funkcionality aplikace, aniž by museli znovu implementovat běžně používané komponenty. V ekosystému Androidu se knihovny často využívají k poskytování nástrojů pro práci s uživatelským rozhraním, správu síťových požadavků, databází, analytiky nebo jiných systémových služeb.

5.1 Vytvoření a konfigurace projektu knihovny

Vývoj Android knihovny zahrnuje několik klíčových kroků, od vytvoření projektu přes konfiguraci build systému až po její distribuci. Knihovny se obvykle vytvářejí jako samostatné **Android Library Module**, což umožňuje jejich snadnou integraci do různých aplikací pomocí vygenerovaných **AAR** souborů. [24]

Knihovnu lze vytvořit přímo v prostředí **Android Studio** přidáním nového modulu:

1. Otevřeme **Android Studio** a přejdeme do nabídky **File -> New -> New Module**.
2. Vybereme možnost **Android Library**.
3. Vyplníme potřebné údaje:
 - **Název knihovny**.
 - **Minimální verzi SDK** (např. 24).
4. Klikneme na **Finish** a počkáme na dokončení inicializace projektu.

Po vytvoření modulu knihovny bude v projektu dostupná složka odpovídající názvu knihovny. Tato složka obsahuje základní soubory, jako jsou **AndroidManifest.xml**, složka **res/** pro zdroje a adresář **kotlin+java/**, kde lze začít s vývojem knihovny.

Soubor **build.gradle.kts** pro modul knihovny by měl ve své hlavičce *plugins* obsahovat položku udávající, že se jedná o knihovnu a ne aplikaci:

```
id("com.android.library")
```

V rámci stejného projektu pak můžeme knihovnu lokálně spouštět a testovat v hlavní části naší aplikace. Přejdeme do projektového souboru **settings.gradle.kts** a na jeho konec přidáme řádek

```
include(":nazevknihovny")
```

Poté můžeme knihovnu zaregistrovat v aplikačním modulu `build.gradle.kts` v rámci *dependencies* bloku jako:

```
implementation(project(":nazevknihovny"))
```

Po synchronizaci projektu můžeme využít funkce knihovny v hlavní aplikaci stejně, jako kdyby byla již publikována a přidána přes externí závislost. [25]

5.2 Publikování knihovny

Po dokončení vývoje je důležité zajistit, aby knihovnu mohli snadno používat i ostatní vývojáři. Existuje několik způsobů distribuce, přičemž nejčastěji se využívají **JitPack** a **Maven Central**. Zatímco Maven Central je standardním repozitářem pro Java/Kotlin knihovny, jeho použití vyžaduje složitější proces registrace, konfigurace a schvalování. Proto je pro menší a nezávislé knihovny často výhodnější distribuovat je pomocí **JitPacku**, který umožňuje rychlou a jednoduchou integraci přímo z **GitHubu**.

II PRAKTICKÁ ČÁST

6 VZNIK NÁPADU

Ještě před samotným začátkem tvorby aplikace bylo potřeba mít jasnou představu o nápadu. Kapitola o hře *SongPop Party* (2.3) již naznačuje určitou inspiraci a směr, kterým se autor práce rozhodl vydat. Jelikož na trhu existuje jen omezený počet podobných her, rozhodl se pro vývoj pokerové hry postavené na vlastní knihovně, kterou mohou využít i jiní vývojáři a díky ní vytvářet podobné hry.

Tato práce tedy směřuje ke dvěma hlavním cílovým skupinám, které se od sebe výrazně liší. **První skupinou** jsou hráči, pro které byla vytvořena hra poker. Díky vysoké popularitě pokeru a podobných her byla aplikace stažena více než 2000krát již během vývojové fáze, a to i bez jakékoliv reklamy. To ukazovalo na velký zájem o takový typ zábavy a motivovala autora k dalšímu rozvoji. **Druhou skupinou** jsou vývojáři, pro které byla navržena knihovna, jež zjednodušuje tvorbu lokálních multiplayerových her. Tento nástroj by tak mohl nejen usnadnit jejich práci, ale také přispět k širšímu zájmu o vývoji her tohoto typu v budoucnu.

6.1 Volba technologií

Před vývojovou fází bylo nutné zvážit technické aspekty vývoje, především v podobě výběru platformy. Vzhledem k tomu, že takový typ aplikace vyžaduje přímou komunikaci mezi zařízeními, bylo obtížné najít vhodnou knihovnu, která by efektivně podporovala veškeré operační systémy a jejich požadavky. Knihovna **Nearby Connections** (3) se nakonec stala jasným vítězem, avšak bohužel s omezením, že bude hra dostupná pouze na zařízení s operačním systémem Android s budoucí motivací přechodu na multiplatformní vývoj, který je díky použitým technologiím velice slibný.

6.1.1 Kotlin a Jetpack Compose

Ačkoli se technologie Kotlin a Jetpack Compose primárně používají pro vývoj moderních aplikací, v tomto konkrétním případě se ukázaly jako ideální volba. Pro tvorbu her pro více hráčů, které obvykle nemají složité grafické efekty a nevyžadují fyzikální enginy, se tyto nástroje jeví jako vynikající volba, zejména pro jejich jednoduchost, efektivitu a neustálý vývoj, který se **postupně blíží k plné podpoře multiplatformního vývoje**. Kromě toho Jetpack Compose přináší výhody v podobě moderního přístupu k uživatelskému rozhraní a zjednodušuje kódování pro mobilní platformy. Kotlin, jako jazyk, snižuje složitost vývoje

a poskytuje silnou podporu pro tvorbu kvalitního kódu, což přispívá k efektivnímu vývoji hry.

6.2 Požadavky na knihovnu

Důležitým krokem bylo stanovení požadavků na vytvářenou knihovnu. Již od začátku bylo jasné, že se bude jednat o obal nad hlavním jádrem, které využívá Nearby Connections API od Googlu. Tento obal poskytuje přímý přístup k funkcionalitě této knihovny a současně zajišťuje vyšší úroveň abstrakce, čímž usnadňuje vývoj lokálních multiplayerových her. Architektura knihovny je navržena tak, aby umožňovala snadné rozšíření o další funkcionalitu v budoucnu.

6.2.1 Funkcionální požadavky knihovny

Jak již bylo zmíněno výše, knihovna má sloužit jiným vývojářům k tvorbě vlastních her založených na lokálním propojení hráčů a jejich zařízení. Z toho vychází následující set požadavků:

- Možnost vytvoření serveru a jeho konfigurace
 - Nastavení typu serveru (režim host nebo client).
 - Definování maximálního počtu připojených hráčů.
 - Možnost správy připojených klientů.
- Práce se stavem serveru
 - Udržování připojených hráčů a jejich synchronizace.
 - Naslouchání na jejich eventy (např. připojení, odpojení, odeslaná zpráva).
- Možnost vytvoření klienta a jeho konfigurace
 - Přiřazení základních vlastností (např. jméno hráče, avatar).
 - Možnost přizpůsobení dalších atributů podle potřeby vývojáře.
 - Zajištění jednoduchého připojení klienta k serveru.
- Zajištění komunikace mezi klientem a serverem
 - Navázání automatické komunikace.
 - Poskytnutí rozhraní pro odesílání a přijímání zpráv.

- Možnost definování vlastních zpráv a jejich formátu.
- Podpora rozšiřitelnosti
 - Poskytnutí generických funkcí, které si může vývojář upravit podle svých potřeb.
 - Možnost přidání vlastních typů zpráv a událostí.
- Bonusová funkcionalita
 - Podpora pro práci s *Android Resources* (např. překlady textů).
 - Do budoucna: Možnost integrace dalších UI prvků jako například menu obrazovka a podobně.

6.3 Funkcionální požadavky hry poker

V případě samotné pokerové hry byly funkcionální požadavky tvořeny na základě ovládání a možností vytvořené hry. Jedná se zkrátka o funkce, které se ve hře daného charakteru očekávají.

- Možnost zapnutí hry v režimu server
 - Umožnění vybrat herní režim serveru
 - Buď pouze jako herní plátno a zprostředkovatel hry.
 - Nebo zároveň hráč.
 - Systém hry umožňuje nastavit si vytvořenou hru
 - Název nastavení
 - Peníze hráče.
 - Malá a Velká sázka.
 - Časovač tahu.
 - Systém dovoluje si nastavení hry vytvořit, uložit a načíst.
 - Zařízení v režimu serveru má následně možnosti k hlavnímu ovládání hry.
- Možnost zapnutí hry v režimu klient (včetně režimu server-hráč)
 - Systém hry vyžaduje zápis herní přezdívky hráče.
 - Systém umožňuje výběr avatara.

- Systém umožňuje navázat spojení hráče se serverem a zobrazit herní plochu.
- Systém hráči umožňuje připravit se ke hraní hry po jejím připojení. Dále umožňuje se od připravit nebo se ze hry odpojit.
- Během hry má hráč následující možnosti herních tahů:
 - Check, Call, Raise, Bet, Fold.
 - Tyto možnosti se automaticky přizpůsobují aktuálnímu stavu herní hry.
 - Systém umožňuje vibrování telefonu jakožto signál značící, že je hráč na řadě.
- Systém hry obsahuje nastavení a manuál, který je dostupný i v aktivní hře
 - Součástí nastavení je možnost výběru jazyka (čeština, angličtina) a vibrování.
 - Manuál dále obsahuje informace o hře, pravidla, karetní kombinace, slovník pojmů a použité zdroje.
- Systém her je plně autonomní
 - Po zapnutí hry „serverem“ se kontroluje počet připravených hráčů, ten musí být větší nebo roven 2.
 - Jednotlivá kola hry se automaticky mění.
 - Server jakožto vlastník hry validuje veškeré dění ve hře.
 - Po posledním kole je hra automaticky vyhodnocena
 - Zobrazení karet nesložených hráčů.
 - Krátká animace.
 - Start nové hry.
- Systém hry obsahuje centrum zpráv
 - Zobrazuje připojení nebo odpojení hráče.
 - Zobrazuje informace o odehraných kolech včetně detailů
 - Kdo měl jaké karty a karetní kombinace (pouze v případě nesložení karet)
 - Jaké karty byly na stole

- Kdo byl v daném herním kole vítěz.
- Systém ve hře indikuje aktuálně hrajícího hráče barevnou časomírou kolem jeho avatara.
- Systém hry automaticky rozsazuje hráče kolem stolu a přizpůsobuje vzhled na každém zařízení.

6.4 Nefunkcionální požadavky

Z hlediska nefunkcionálních požadavků můžeme obě skupiny programů spojit, jelikož zde se jedná převážně o vlastnosti technického charakteru, které neovlivňují uživatele přímo, ale mají zásadní dopad na celkovou použitelnost a kvalitu aplikace. Tyto požadavky se vztahují jak na samotnou knihovnu, tak na hru, která ji využívá.

Knihovna musí být navržena tak, aby umožňovala snadné rozšíření funkcionality v budoucnu a poskytovala intuitivní rozhraní pro vývojáře, kteří ji chtějí integrovat do svých projektů. Důležitým faktorem je stabilita a spolehlivost komunikace mezi zařízeními. Zároveň je třeba dbát na bezpečnost přenosu dat mezi zařízeními, aby nedošlo k neoprávněnému přístupu třetí strany. Dalším důležitým aspektem je kompatibilita s různými verzemi Androidu, přičemž cílem je, aby knihovna fungovala na co nejširší škále zařízení a v budoucnu podporovala i zařízení s operačním systémem iOS.

Co se týče samotné hry, klíčovým požadavkem je plynulost uživatelského rozhraní, která zajistí stabilní chod aplikace bez znatelných lagů nebo zpomalení. Ovládání hry musí být intuitivní, aby bylo snadno pochopitelné i pro nové hráče, a zároveň přizpůsobitelné různým typům zařízení. Jelikož hra běží na mobilních telefonech s různými rozlišeními a poměry stran, je nezbytné zajistit její správné zobrazení ve všech případech.

6.4.1 Ukládání dat

Hra využívá několik různých metod ukládání dat v závislosti na typu informací, které je potřeba uchovat.

Pro uchování hráčských preferencí, jako je například nastavení **vibrování**, je použita technologie Jetpack **DataStore**. Tato moderní alternativa k SharedPreferences umožňuje efektivní a bezpečné ukládání jednoduchých nastavení aplikace. [26] Díky tomu si hra dokáže zapamatovat individuální preference hráčů i po restartu aplikace.

Změna jazyka aplikace pak probíhá dynamicky pomocí **AppCompatDelegate**, který upravuje jazykovou lokalizaci během běhu aplikace bez nutnosti jejího restartu. Veškeré texty jsou uloženy prostřednictvím systému **Android Resources**. Tento přístup umožňuje jednoduchou správu textů a podporu více jazykových verzí. Překlady jsou organizovány v samostatných souborech, například:

res/values/translations.xml (angličtina)

res/values-cs/translations.xml (čeština)

Díky tomu si hráč může v nastavení aplikace kdykoliv změnit jazyk, přičemž aplikace automaticky načte odpovídající jazykovou verzi z uložených zdrojů bez nutnosti opětovného spuštění aplikace.

Naopak pro složitější a strukturovanější data, jako je **herní konfigurace** při spouštění serveru – například počáteční částka hráče, velikost malých a velkých sázek nebo časový limit tahu, je využita databáze **Room**. Tato databázová vrstva poskytuje efektivní způsob ukládání a načítání těchto údajů, což zajišťuje, že hráči mohou kdykoli obnovit své předchozí nastavení a pokračovat ve hře podle svých preferencí.

Díky kombinaci těchto technologií je zajištěno, že jednoduchá uživatelská nastavení, tak i komplexní herní konfigurace jsou bezpečně a efektivně uloženy, což přispívá k lepšímu uživatelskému zážitku a vyšší spolehlivosti aplikace.

7 VÝVOJ KNIHOVNY

Knihovna vznikla **oddělením síťové logiky z původního kódu hry**, kde byla již od počátku součástí vývoje a umožňovala testování multiplayerové komunikace. Až v pozdější fázi vývoje se ukázalo, že **síťová logika je dostatečně oddělitelná**, aby mohla být zpřístupněna jako **samostatná knihovna**, kterou mohou využít i další vývojáři.

Hlavním cílem této knihovny je **usnadnit vývoj her s lokálním multiplayerem** s využitím technologie **Nearby Connections API** od Googlu. Díky tomu mohou ostatní vývojáři snadno vytvářet vlastní hry, aniž by se museli zabývat složitým procesem navazování komunikace mezi zařízeními.

Tato kapitola se zaměřuje na celý proces vývoje knihovny – od prvotního návrhu struktury, přes implementaci klíčových částí až po jejich použití v projektu. Nejprve se podíváme na návrh knihovny a popíšeme, jak byly řešeny hlavní výzvy spojené s komunikací mezi zařízeními. Následně se zaměříme na implementaci jednotlivých komponent a jejich možností integrace do různých her pro Android. V závěru se zmíníme o možnostem jejího budoucího rozšíření.

7.1 Vybraná strategie pro komunikaci

V teoretické kapitole o knihovně **Nearby Connections** (3) byly popsány **tři dostupné strategie** pro zprostředkování komunikace mezi zařízeními a formáty přenosu dat. Výběr vhodné strategie ovlivňuje nejen samotnou komunikaci, ale také **bezpečnost a synchronizaci dat mezi hráči**.

Z tohoto důvodu knihovna přešla z dynamického výběru strategie vývojářem na pevně stanovenou metodu **Star** (hvězda). Tato strategie umožňuje plné řízení hry výhradně ze zařízení nastaveného jako server, což zajišťuje konzistentní stav hry a minimalizuje riziko nesynchronizovaných dat mezi hráči.

Ačkoliv má vývojář stále možnost upravit interní logiku hry tak, aby simulovala jiné modely komunikace, přímá komunikace mezi klientskými zařízeními není v rámci této strategie podporována. Všechny zprávy tak **vždy prochází přes server**, což usnadňuje správu herního stavu a eliminuje potenciální konflikty v synchronizaci.

7.2 Vývoj jednotlivých částí

Jak již bylo zmíněno, **síťová logika byla jedním z prvních prvků celého projektu**, protože bez ní by nebylo možné testovat pokerovou část hry mezi více zařízeními. První verze této logiky byla **jednoduchá a pevně nastavená**, aby bylo možné rychle ověřit fungování propojení fyzických zařízení a jejich vzájemnou komunikaci.

Vývoj této části prakticky probíhal **paralelně s vývojem pokerové hry**, přičemž bylo nutné **neustále vylepšovat stabilitu připojení** a správně pracovat s instancí *ConnectionsClient*, aby nedocházelo k nežádoucím chybám nebo únikům paměti.

S postupem času a lepším pochopením fungování Nearby Connections API se síťová logika **stávala ucelenější a strukturovanější**. Původně jednoduchý kód se začal rozdělovat na **samostatné komponenty**, které začaly dávat smysl nejen v kontextu pokerové hry, ale i **potenciálního využití v jiných hrách**.

Jakmile byla komunikace **stabilní** a podporovala veškeré potřebné akce pro pokerovou hru, přišel klíčový moment – **oddělení síťové logiky do samostatné knihovny**. Tím vznikla univerzální **knihovna pro lokální multiplayer**, která byla pojmenována **EasyLocalGame**. Tento krok umožnil nejen **jednodušší údržbu kódu**, ale především **otevřel možnosti jejímu využití i v jiných projektech**.

7.2.1 Výzvy a komplikace během vývoje

Jak už tomu tak bývá, i během vývoje knihovny **EasyLocalGame** se objevilo několik zásadních výzev, které bylo nutné překonat. Tyto problémy souvisely především se správou síťového připojení, možnostem univerzálního použití, testováním na různých zařízeních a zajištěním celkové stability.

Jedním z klíčových problémů byla **správná správa instance ConnectionsClient**, která hraje zásadní roli v celé komunikaci mezi zařízeními. Bylo nutné zajistit, aby instance byla správně uchováвана a nezpůsobovala **chyby při opakovaném připojování** jako už bylo zmíněno dříve. Tento problém se ale netýká přímo knihovny samotné, jelikož je potřeba tuto instanci spravovat mimo ni a jen na tuto skutečnost upozorňuje vývojář.

Dalším výrazným problémem bylo **testování síťové komunikace**, jelikož autor ne vždy disponoval dvěma a více fyzickými zařízeními. Proto bylo často nutné využívat **Android emulátory** z Android Studia, jejichž fungování ale nebylo vždy ideální, jelikož:

- **Byly pomalé a zdlouhavé na spouštění**, což výrazně zpomalovalo vývoj a ladění.

- **Nefungovaly konzistentně** – mnohdy se stávalo, že při novém spuštění programu nebo emulátoru komunikace nefungovala, i když byla plně funkční.
- **Měly omezenou podporu Nearby Connections API** – komunikace mezi emulátory většinou fungovala pouze na zařízeních s emulovaným systémem "**Android VanillaIceCream Preview | x86_64**".

7.3 Struktura knihovny

Aby byla knihovna co nejlépe použitelná a umožňovala snadnou integraci do různých projektů ze strany ostatních vývojářů, byla rozdělena do několika samostatných částí, z nichž každá plní konkrétní úlohu v rámci síťové komunikace. Toto rozdělení zajišťuje přehlednost kódu, umožňuje lepší rozšiřitelnost a usnadňuje vývojářům implementaci do jejich vlastních aplikací. Knihovna je postavena na třech hlavních modulech:

- **ServerSide** – komponenta odpovědná za správu herního serveru, který vytváří herní místnost, přijímá připojení klientů a řídí průběh komunikace.
- **ClientSide** – část knihovny určená pro správu klientských zařízení, která se připojují k serveru a komunikují s ním.
- **PayloadUtils** – pomocná vrstva zajišťující strukturování, serializaci, přenos a deserializaci zpráv mezi klienty a serverem ve formátu JSON.

Každá z těchto částí je navržena tak, aby poskytovala vývojářům dostatečnou flexibilitu a umožňovala snadné přizpůsobení dle specifických potřeb jednotlivých her. V následujících podkapitolách budou jednotlivé komponenty popsány podrobněji včetně jejich klíčových funkcí a způsobu použití.

7.4 Implementace ServerSide z pohledu vývojáře

Jedná se o komponentu obsahující kód spojený se stranou serveru. Její hlavní třída **ServerManager** implementuje první část knihovny Nearby Connections v několika privátních funkcích, které jsou v teoretické části popsány v příslušné kapitole (3.3.1). **Z pohledu vývojáře** je hlavní především funkce **startServer**, která přebírá 2 hlavní parametry:

- **packageName** – Proměnná typu String udávající název balíčku aplikace.

- **serverConfiguration** – Předdefinovaná datová třída, díky které si může vývojář definovat chování zapnutého serveru na zařízení – např. kolik připojení může server přijmout, jak se bude server jmenovat, nebo zda bude použité zařízení sloužit i jako hráč.

Zavoláním této funkce se v aplikaci nastaví požadované vlastnosti a zařízení začne naslouchat pro připojení klientů.

Aby mohl vývojář reagovat na různé stavy knihovny, třída `ServerManager` obsahuje veřejnou proměnnou **serverState**, která je definována pomocí objektu `MutableStateFlow` a poskytuje přehled o aktuálním stavu serveru. Tento objekt je předdefinovaného typu `ServerState`, který obsahuje několik klíčových parametrů:

- **serverStatus** – Jeden z hlavních parametrů sloužící pro uchování a kontrolu stavu serveru. Tento parametr je typu **ServerStatusEnum**, což je *enum* typ, který může nabývat následujících hodnot:
 - **NONE**: Inzerce serveru nebyla inicializována. Tento stav znamená, že server ještě nezačal inzerovat svou přítomnost a není tedy připraven k připojení klientů.
 - **ADVERTISING_FAILED**: Inzerce serveru selhala. Tento stav je signalizován v případě, že došlo k chybě při zahájení inzerce, což může být způsobeno například problémy s dostupností zařízení, chybnou konfigurací nebo chybějícími oprávněními.
 - **ADVERTISING**: Server právě inzeruje svou přítomnost a klienti se mohou připojovat. Tento stav znamená, že server je připraven přijímat připojení od klientů.
 - **ACTIVE**: Server je aktivní, ale již neinzeruje svou přítomnost. Klienti se nemohou připojit, protože inzerce byla zastavena, ale **server stále běží** a může pokračovat ve hře nebo komunikaci.
 - **CLOSED**: Server byl uzavřen. Tento stav je často používán k ukončení hry nebo přechodu na jinou obrazovku (například přechod zpět na hlavní menu nebo úvodní obrazovku).
- **serverType** – Uložený stav z konfigurace serveru (zda je zařízení pouze server nebo zároveň i hráč).

- **connectedClients** – Seznam zařízení, která jsou aktuálně připojena k serveru. Tento seznam uchovává hodnoty typu *String*, přičemž každý prvek představuje **EndpointID** daného zařízení, což je unikátní identifikátor připojeného zařízení poskytovaného skrze Nearby Connections API.

Tento **serverState** objekt je užitečný nejen pro sledování aktuálního stavu serveru, ale i pro **přizpůsobení uživatelského rozhraní v reálném čase** (například přepínání mezi obrazovkami, zobrazení chybových zpráv nebo jiných informací podle aktuálního stavu serveru).

Dalšími dostupnými funkcemi pro vývojáře jsou funkce pro komunikaci s připojenými klienty. K dispozici jsou dvě varianty funkce **sendPayload**, které se liší v parametrech. První varianta přijímá pouze zprávu (typu **Payload**), kterou následně odešle všem připojeným klientům. Druhá varianta navíc specifikuje **EndpointID** cílového zařízení, což umožňuje odeslat zprávu pouze konkrétnímu připojenému zařízení.

Další dvě funkce slouží k řízení stavu serveru. Funkce **stopAdvertising** vypíná inzerci a přepíná server do stavu **ACTIVE**, čímž zastavuje vyhledávání dalších klientů a šetří tak baterii zařízení. Funkce **closeServer** zajistí odpojení všech připojených klientů, zastaví inzerci a přepne stav serveru do stavu **CLOSED**, což signalizuje ukončení provozu serveru.

7.4.1 Použití **ServerSide** z pohledu vývojáře

Pro usnadnění integrace knihovny do herních aplikací byl vytvořen **template ViewModelu**, který vývojářům umožňuje jednoduše spravovat serverovou logiku a funkce popsané v kapitole výš (7.4). Tento template je definován jako abstraktní třída **ServerViewmodelTemplate**, kterou je třeba zdědit v konkrétní implementaci ViewModelu.

```
abstract class ServerViewmodelTemplate(  
    private val connectionsClient: ConnectionsClient,  
): ViewModel() {  
    abstract fun clientAction(clientAction: ClientAction)  
    val serverManager: ServerManager by lazy {  
        ServerManager(connectionsClient, this::clientAction)  
    }  
}
```

Tato šablona tak poskytuje vývojářům:

- **Předpřipravenou logiku pro správu serveru** – obsahuje instanci třídy **ServerManager**, která zajišťuje práci s Nearby Connections API a přímé volání předešlých funkcí.
- **Abstraktní metodu clientAction**, kterou musí vývojář implementovat, aby mohl reagovat na akce klientů (například odpojení hráče, příjem zprávy apod.).

Důležitým upozorněním opět je, že třída **ServerViewModelTemplate** vyžaduje jako parametr **connectionsClient**, což je instance **ConnectionsClient** poskytovaná knihovnou Nearby Connections. Tuto instanci nelze jednoduše vytvořit například ve **ViewModelu**, jelikož je **vyžadován kontext** aplikace, který **není vhodné uchovávat ve ViewModelu** kvůli rozdílným životním cyklům a riziku úniku paměti. [27] Proto je vývojářům doporučeno **ConnectionsClient** získat mimo **ViewModel**, například pomocí **dependency injection (DI)**, a následně jej předat do **ViewModelu** jako závislost.

Knihovna dále obsahuje definovanou třídu **ClientAction**, která obsahuje 3 různé typy akcí:

- **EstablishConnection** – Označuje úspěšné připojení klienta a navázání komunikace. Tato akce obsahuje dodatečné informace o připojeném hráči, které si vývojář může nadefinovat a následně využít například pro vytvoření herního objektu hráče. Mezi tyto informace může patřit například jméno hráče, obrázek avatara apod.
- **Disconnect** – Signalizuje odpojení klienta. Vývojář může na tuto změnu reagovat aktualizací herního stavu, například odstraněním hráče ze hry.
- **PayloadAction** – Obecná akce umožňující vývojáři **definovat vlastní chování** hráčů ve hře. Může se jednat například o akce jako "hráč je připraven ke hře", "hráč hodil kostkou" a jiné.

7.5 Implementace ClientSide z pohledu vývojáře

Podobně jako u serverové části knihovny je i klientská část spravována pomocí samostatné třídy – **ClientManager**. Tato třída obsahuje veškerou logiku potřebnou k navázání spojení se serverem a k jejich následné komunikaci. Její fungování je obdobné jako u třídy **ServerManager**, avšak s tím rozdílem, že zde jsou dostupné funkce určené pouze pro klienta.

Klíčovou funkcí této části je funkce **connect**, která umožňuje připojení ke zvolenému serveru. Tato funkce přijímá dva hlavní parametry:

- **packageName** – Proměnná typu String, která by měla odpovídat názvu balíčku aplikace. Tato hodnota **musí být stejná jako u funkce startServer**, aby bylo možné navázat správné spojení.
- **playerConnectionState** – Objekt implementující rozhraní `IPlayerConnectionState`. Vývojář si jej může definovat dle svých potřeb a využít ho k nastavení vlastností připojovaného hráče (např. jméno, avatar apod.). Tento objekt se následně odesílá serveru po úspěšném připojení, viz **EstablishConnection** akce výše.

Stejně jako serverová část obsahuje i ta klientská část proměnnou pro sledování aktuálního stavu spojení – **clientState**. Tento stav je reprezentován objektem datového typu **ClientState**, který zahrnuje následující parametry:

- **connectionStatus** – Enum typ **ConnectionStatusEnum**, který uchovává aktuální stav připojení klienta a může být opět použitý pro změnu obrazovky ve hře. Tento stav tak může nabývat jednu z následujících hodnot dle vývoje spojení s herním serverem:
 - **NONE** – Žádné připojení nebylo inicializováno.
 - **CONNECTING** – Uživatel zahájil připojování do hry.
 - **CONNECTING_FAILED** – Připojení k serveru se nezdařilo. Může značit chybějící povolení, chybnou konfiguraci nebo zkrátka vypršení pokusu pro vyhledání serveru.
 - **CONNECTING_REJECTED** – Připojení k serveru bylo odmítnuto.
 - **ROOM_IS_FULL** – Připojení bylo úspěšné, ale herní místnost je plná a nelze se připojit. Dochází k automatickému odpojení.
 - **ENDPOINT_LOST** – Klient ztratil spojení se serverem.
 - **CONNECTED** – Připojení bylo úspěšně navázáno.
 - **CONNECTION_ESTABLISHED** – Připojení bylo dokončeno a klient úspěšně odeslal své informace na server.
 - **DISCONNECTED** – Připojení bylo ukončeno.

- **serverID** – Identifikátor serveru, ke kterému je klient připojen.
- **serverType** – Typ serveru definovaný podle **ServerType**. Tento údaj umožňuje klientovi přizpůsobit zobrazení herní plochy v závislosti na tom, zda server zároveň slouží jako hráč nebo pouze jako hostitel.

Kromě funkce pro připojení obsahuje klientská část knihovny také dvě další veřejné funkce. První z nich, **disconnect**, umožňuje manuální odpojení klienta od serveru. Druhá, **sendPayload**, slouží k odesílání zpráv na server. V tomto případě není nutné specifikovat **EndpointID**, protože veškerá komunikace probíhá výhradně mezi klientem a serverem. Identifikátor serveru je automaticky uložen v parametru **serverID** objektu **clientState** při úspěšném připojení klienta.

7.5.1 Použití ClientSide z pohledu vývojáře

Aby bylo možné klientskou část knihovny efektivně využít, byla opět vytvořena šablona **ViewModelu**, která vývojáři usnadňuje práci a poskytuje přímý přístup k funkcím třídy **ClientManager**. Šablona ***PlayerViewModelTemplate*** tak obdobně poskytuje 2 hlavní komponenty:

- Instanci **clientManager**, díky čemuž získá vývojář přímý přístup k síťové části klienta.
- Abstraktní metodu **serverAction**, kterou si musí vývojář opět implementovat podle vlastních požadavků, aby mohl reagovat na akce přicházející ze serveru.

Tyto serverové akce jsou opět definovány vlastním typem **ServerAction**, který obsahuje 3 hlavní události:

- **UpdateGameState** – Slouží k aktualizaci herního stavu na straně klienta na základě informací od serveru.
- **UpdatePlayerState** – Umožňuje aktualizaci stavu hráče.
- **PayloadAction** – Univerzální akce, kterou si může vývojář přizpůsobit pro vlastní potřeb. Může ji například využít i jako alternativu k předchozím dvěma akcím.

7.6 Systém pro přenos zpráv

Komunikace mezi serverem a klientem v rámci knihovny probíhá prostřednictvím vytvořeného systému pro přenos zpráv v komponentě **PayloadUtils**. Tento systém využívá knihovnu Nearby Connections k formování **payload** objektu, který se jako JSON objekt posílá mezi zařízeními. Hlavním cílem této části knihovny je poskytnout vývojářům **jednoduchý a flexibilní způsob** přenosu strukturovaných dat, která mohou obsahovat jak systémové zprávy knihovny, tak vlastní herní události.

Přenos dat v knihovně využívá pouze metodu **Payload.Bytes**, což znamená, že veškeré zprávy se odesílají jako **bajtová data** (ByteArray). Tento přístup byl zvolen z důvodu efektivity a optimalizace pro real-time herní komunikaci. Jak již víme, knihovna Nearby Connections sice umožňuje i přenos **souborů (Payload.File)** a **streamů (Payload.Stream)**, avšak v kontextu herní komunikace by jejich využití bylo zbytečně náročné a neefektivní.

I když knihovna tyto funkce nabízí, jejich použití není pro vývojáře povinné. Pokud má aplikace specifické požadavky na serializaci objektů nebo jiný způsob zpracování zpráv, může vývojář implementovat své vlastní řešení. **Jedinou podmínkou pro zajištění komunikace mezi zařízeními je převod odesílaných zpráv na objekt Payload**, se kterým pracuje Nearby Connections API.

7.6.1 Definované typy zpráv

Pro rozlišení zpráv přenášovaných mezi serverem a klienty jsou v knihovně definovány **dvě enum třídy**, které slouží jako identifikátory jednotlivých typů zpráv převážně pro interní fungování knihovny:

- **ClientPayloadType** – obsahuje akce, které klient posílá serveru:
 - ESTABLISH_CONNECTION – Zpráva, že klient žádá o potvrzení připojení k serveru tím, že mu zasílá své herní údaje.
 - ACTION_DISCONNECTED – Hráč se odpojil od serveru.
- **ServerPayloadType** – obsahuje akce, které může server posílat klientům:
 - CLIENT_CONNECTED – Potvrzení úspěšného připojení klienta.
 - ROOM_IS_FULL – Informace, že místnost je plná a klient se nemůže připojit.
 - UPDATE_PLAYER_STATE – Aktualizace stavu konkrétního hráče.

- UPDATE_GAME_STATE – Aktualizace celkového stavu hry pro hráče.

V tomto momentě již lze vidět určité propojení s akcemi, které jsou popsány v kapitolách ohledně implementace ServerSide a ClientSide výše.

Například při změně herního stavu na serveru, server odešle svým připojeným klientům zprávu typu UPDATE_GAME_STATE, která se na klientské straně přemění na odpovídající akci ServerAction.UpdateGameState.

7.6.2 Definování vlastních typů zpráv

Ačkoliv knihovna poskytuje sadu výchozích *enum* typů (ClientPayloadType a ServerPayloadType), vývojář **má možnost používat i vlastní typy** zpráv přizpůsobené své konkrétní hře. Tato přidaná flexibilita umožňuje vývojářům **plně přizpůsobit komunikaci** mezi klienty a serverem podle potřeb jejich hry a dá se říct, že je to jeden z klíčových požadavků k libovolnému použití této knihovny.

A aby se nám to hezky uzavřelo, tak právě pro tyto typy zpráv slouží již zmíněná akce **PayloadAction** v sekcích ServerAction a ClientAction. Díky tomu si vývojář může **přizpůsobit herní události**, aniž by byl omezen pevně definovanými akcemi z knihovny. Pokud chce například přidat akci „hráč vsadil žetony“, nemusí měnit jádro knihovny, ale jednoduše si definuje vlastní typ zprávy a zpracuje ji v rámci události **PayloadAction**.

7.6.3 Princip fungování

PayloadUtils využívá **generické typy** (v kódu většinou značeny jako <T>), což umožňuje přenos **libovolných dat** v rámci jedné univerzální funkce. Díky tomu může vývojář posílat **různé typy objektů**, aniž by musel definovat samostatné metody pro každý případ.

Serializace a deserializace dat probíhá ve formátu JSON pomocí Kotlin knihovny **Gson**, přičemž je každá zpráva převedena na **pár hodnot** (Kotlin Pair), kde:

- **První** hodnota obsahuje typ zprávy (např. ServerPayloadType.*).
- **Druhá** hodnota obsahuje samotná data.

V rámci fungování této části knihovny pak můžeme definovat 3 základní bloky:

1) Tvorba Payload objektu pro odesílání zpráv

Odesílání zpráv pracuje se **třemi odlišnými funkcemi**, z nichž jsou 2 primárně určeny pro fungování uvnitř knihovny s cílem zajistit automatickou komunikaci například při připojování nového klienta k serveru. Všechny funkce ale pracují na stejném principu, kdy vstupem do funkce je **typ zprávy** a její **data**. Tyto informace se převedou do páru hodnot, který se následně převede na json objekt. Tento objekt je následně **převeden na objekt typu Payload**, se kterým může knihovna Nearby Connections pracovat a přenést jej tak mezi zařízeními.

- **toClientPayload<T>(payloadType: ClientPayloadType, data: T?)**
 - Pro označení typu zprávy pracuje s vytvořeným knihovním typem *ClientPayloadType*.
 - Uvnitř knihovny je automaticky volaná například když klient odesílá údaje o hráči během připojování (ESTABLISH_CONNECTION).
- **toServerPayload<T>(payloadType: ServerPayloadType, data: T)**
 - Prakticky totožná funkce jako toClientPayload, ale pro **odesílání zpráv ze serveru klientům**.
- **toPayload(payloadType: String, data: T?, typeAdapters: Map<Type, Any>?=null)**
 - **Obecná funkce** pro zasílání vlastních zpráv vývojáře.
 - Díky tomu může vývojář definovat **vlastní enum typ**, jehož prvek převede na String pomocí vestavěné funkce enum *.toString()*. (*MujTyp.READY.toString()*).
 - Tato funkce je **univerzální** a lze ji použít **jak pro klientskou, tak i pro serverovou část kódu**.

2) Přijímání zpráv

Při přijímání zpráv se využívá **univerzální funkce fromPayload**, která konvertuje přijatý objekt typu Payload na **pár hodnot (Pair<R, T?>)**, kde:

- **První hodnota (R)** představuje **typ zprávy** (např. *ServerPayloadType*, *ClientPayloadType* nebo vlastní Enum typ).

- **Druhá hodnota (T?)** obsahuje samotná **deserializovaná data**, pokud byla součástí zprávy.

Díky využití **generických typů** (<R, T>) umožňuje `fromPayload` vývojářům zpracovávat různé typy zpráv **v rámci jedné univerzální funkce**, bez nutnosti psát samostatné metody pro každý typ přenášených dat. Funkce dále poskytuje volitelné nastavení **TypeAdaptéru**, který je důležitý zejména u složitých objektů, které nelze jednoduše serializovat a deserializovat.

7.7 Bonusové vlastnosti knihovny

Aby byla knihovna `EasyLocalGame` pro vývojáře pohodlná a poskytovala funkce navíc, rozhodl se autor do knihovny implementovat i pomocné nástroje pro použití s frameworkem `Jetpack Compose`. Tyto nástroje **mohou** sloužit k pohodlnější implementaci uživatelského rozhraní samotné hry a zároveň poskytují užitečné zjednodušení některých procedur souvisejících s použitím knihovny.

7.7.1 Kontrola potřebných oprávnění

V pozdější fázi vývoje byla do knihovny přidána funkce, která **kontroluje povolení potřebných oprávnění**, jež jsou vyžadovány v `Nearby Connections API` (Tabulka 1). Za tímto účelem byl upraven soubor **AndroidManifest.xml** o potřebná oprávnění a v projektu knihovny vznikl modul **composeUtils** se souborem **PermissionsChecker.kt**, který obsahuje funkci **CheckPermissions**. Tato *@composable* funkce se stará o validní zajištění potřebných oprávnění na základě verze operačního systému Android, jelikož ne všechny verze vyžadují stejné oprávnění.

Tato funkce navíc přijímá tzv. **Slot API** nebo lambda parametr, pomocí kterého funkce vrátí list všech požadovaných oprávnění a jejich stav – zda je oprávnění schváleno uživatelem. Díky němu pak může vývojář tvořit další logiku, např. zobrazení obrazovky požadující od uživatele aplikace o udělení příslušného oprávnění.

Vývojář by pak měl tuto funkci zavolat ve chvíli, kdy jsou oprávnění skutečně potřeba (spouštění serveru / klienta), popř. při zapnutí hry, aby uživatel udělil oprávnění předem.

```
var permissions = rememberMultiplePermissionsState(emptyList())
CheckPermissions(permissionsState = { permissions = it })
```

7.7.2 Práce s textem v resources

Další přidaným nástrojem je *sealed* třída **UiTexts**, která usnadňuje načítání textových hodnot z Android **Resources** systému. *Sealed* třída poskytuje možnost definování předem známých možností, které mohou nastat a v případě **textových** Resources máme 3 možnosti, které třída UiTexts řeší:

- **StringResource** – Běžný textový řetězec odpovídající danému @StringRes ID.
- **PluralResource** – Množné číslo (@PluralsRes ID), které se automaticky přizpůsobuje na základě počtu (quantity).
- **ArrayResource** – Pole textových hodnot (@ArrayRes ID). Pomocí *indexu* lze navíc vybrat konkrétní prvek.

Díky této implementaci vývojář **nemusí volat nativní Android metody ručně**, ale může jednoduše pracovat s textovými hodnotami přímo pomocí UiTexts, čímž navíc získá možnost jednoduché implementace více jazyků ve hře.

7.8 Volání funkcí v projektu

Jakmile vývojář **implementuje knihovnu EasyLocalGame do svého projektu**, vytvoření **serveru** nebo **klienta** se na daném zařízení stává **jednoduchým procesem**. Stačí pouze vytvořit instanci příslušného **ViewModelu** a zavolat odpovídající metody pro inicializaci spojení.

7.8.1 Inicializace serveru

Pro spuštění serveru je potřeba zavolat metodu **startServer**, která nastaví parametry serveru a umožní připojení klientů.

```
val context = LocalContext.current
val serverViewModel = hiltViewModel<ServerViewModel>()
serverViewModel.serverManager.startServer(
    packageName = context.packageName,
    serverConfiguration = ServerConfiguration(
        serverType = ServerType.IS_TABLET,
        maximumConnections = 10
    )
)
```

```
// Odposlouchávání aktuálního stavu serveru
val serverState by serverViewModel.serverManager.serverState
    .collectAsStateWithLifecycle()
```

7.8.2 Inicializace klienta

Klient se k serveru připojuje metodou **connect**, kde zadává své údaje, jako je například **přezdívka a avatar**.

```
val context = LocalContext.current
val playerViewModel = hiltViewModel<PlayerViewModel>()
playerViewModel.clientManager.connect(
    packageName = context.packageName,
    playerConnectionState = PlayerConnectionState(nickname, avatarId)
)
// Odposlouchávání aktuálního stavu klienta
val clientState by playerViewModel.clientManager.clientState
    .collectAsStateWithLifecycle()
```

7.9 Vydání knihovny na JitPack

Z dřívější kapitoly (5.2) již víme, že jedním z nejjednodušších způsobů publikace knihovny je její **uložení do veřejného GitHub repozitáře** a využití služby **JitPack**. Tato služba umožňuje automatické sestavení knihovny a její zpřístupnění prostřednictvím jednoduché adresy, kterou lze použít v závislostech projektu.

Prvním krokem je nastavení knihovny pro její správné sestavení službou JitPack. V souboru **build.gradle.kts** je nutné do bloku *plugins* přidat *id(„maven-publish“)* a následně za tento blok přidat konfiguraci, která ale není zas tak důležitá: [28]

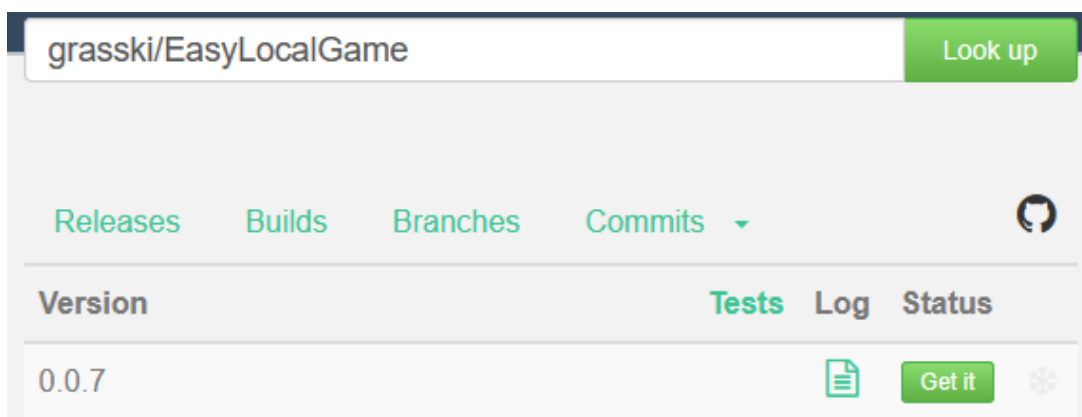
```
afterEvaluate{
    publishing{
        publications{
            create<MavenPublication>("release") {
                from(components["release"])
                groupId = "com.dabi"
                artifactId = "easylocalgame"
                version = "0.0.5"
            }
        }
    }
}
```

Dalším krokem je vytvoření **GitHub repozitáře**, do kterého se nahraje zdrojový kód knihovny. Jakmile je kód knihovny na GitHubu, je nutné vytvořit **tag**, který označuje konkrétní verzi knihovny. To lze udělat buď přímo ve webové aplikaci nebo pomocí příkazů v terminálovém okně:

```
git tag v1.0.0
git push origin v1.0.0
```

Tento krok je klíčový, protože JitPack používá **tagy** k identifikaci verzí knihovny. Jakmile máme tag vytvořený, v GitHub aplikaci vytvoříme **release** s příslušným **tagem**.

Po provedení uvedených bodů stačí přejít na stránku JitPack, kde lze vyhledat repozitář zadáním jeho GitHub URL adresy. Po vybrání správné verze (odpovídající tagu) spustí JitPack sestavení knihovny a vygeneruje adresu, kterou lze použít pro přidání knihovny do projektu. [29]



Obrázek 17. Úspěšné nahrání knihovny na JitPack.
(Zdroj: vlastní)

7.9.1 Dokumentace knihovny

Ještě před samotným vydáním však bylo nezbytné zajistit kvalitní dokumentaci klíčových prvků knihovny. K tomu byla využita struktura **KDoc**, která umožňuje v kódu přidávat formátované komentáře k jednotlivým třídám, funkcím a proměnným, což usnadňuje pochopení jejich účelu a správného použití.

Dalším důležitým krokem bylo vytvoření přehledného **README** souboru, který byl spolu s kódem knihovny umístěn na GitHubu. Tento soubor obsahuje základní informace o knihovně, její funkcionalitě a návod na integraci do projektu.

Kód knihovny a anglická verze dokumentace v podobě **README** soubor jsou dostupné na tomto odkazu: <https://github.com/grasski/EasyLocalGame>.

7.9.2 Použití knihovny v projektu

Jakmile je knihovna úspěšně publikována prostřednictvím JitPacku, lze ji jednoduše integrovat do jakéhokoli projektu, který ji potřebuje. Tento proces zahrnuje přidání **JitPack repozitáře**, nastavení **závislosti** a případné **konfigurace knihovny** v cílovém projektu. [28]

Nejprve je tedy nutné zajistit, aby projekt mohl získat knihovnu z JitPack serveru. To se provede přidáním **zvýrazněného** řádku do souboru **settings.gradle.kts**: [28]

```
dependencyResolutionManagement {  
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)  
    repositories {  
        google()  
        mavenCentral()  
        maven { url = uri("https://jitpack.io") }  
    }  
}
```

Po nastavení repozitáře je dalším krokem přidání samotné knihovny do souboru **build.gradle.kts** modulu aplikace:

```
implementation("com.github.grasski:EasyLocalGame:VERSION")
```

Po těchto krocích bude možné využívat funkcionality knihovny v libovolném projektu.

7.10 Budoucí rozšíření knihovny

Hlavním plánem rozšířením knihovny EasyLocalGame je její podpora multiplatformního použití prostřednictvím **Compose Multiplatform**. Tento krok by umožnil její využití nejen na zařízeních s operačním systémem Android, ale potenciálně i na dalších platformách, jako je iOS.

Jednou z hlavních změn bude přizpůsobení procesu serializace objektů, jelikož knihovna v současné době využívá Gson, který není v multiplatformní podobě podporován. Z tohoto důvodu bude nutné přejít například na knihovnu Kotlinx Serialization, která již multiplatformní podporu nabízí, avšak bohužel neposkytuje tak robustní řešení jako právě knihovna Gson.

Další podstatnou výzvou bude integrace technologie Nearby Connections API, která sice podporuje Android i iOS, ale vzájemná komunikace mezi těmito platformami může být problematická kvůli omezením uzavřeného ekosystému Apple. Jedním z hlavních problémů je skutečnost, že iOS nepodporuje Wi-Fi Direct, což může negativně ovlivnit způsob navazování přímých spojení mezi zařízeními. [30] Navzdory těmto omezením by i tak bylo možné vytvořit nativní implementaci pro iOS pomocí multiplatformního API s využitím expect/actual přístupu.

Přestože multiplatformní podpora představuje velkou příležitost pro rozšíření knihovny, jedním z hlavních důvodů jejího dosavadního nepoužití je i skutečnost, že Compose Multiplatform je stále v beta verzi pro systém iOS. Jeho současná nestabilita tak

může vést k častým změnám API a problémům s kompatibilitou, což by mohlo negativně ovlivnit vývoj a údržbu knihovny.

Další potenciální oblastí rozšíření knihovny by mohlo být vylepšené logovací prostředí, které by vývojářům poskytovalo lepší přehled o stavu knihovny během jejího běhu. To by umožnilo snadnější diagnostiku problémů a ověření, zda vše funguje podle očekávání.

8 VÝVOJ HRY POKER

Vývoj pokerové hry, které předcházela myšlenka vytvoření hry s prvky lokálního multiplayeru, započal kolem roku 2024. Jak už je zmíněno v úvodu, technologie použita k vývoji je již autorem známý programovací jazyk Kotlin s frameworkem Jetpack Compose.

K programování bylo zvolené vývojové prostředí Android Studio, které během vývoje prošlo několika aktualizacemi. Pro udržení čistoty a přehlednosti kódu byl zvolen architektonický model MVVM v kombinaci s prvky MVI, což umožňuje efektivní oddělení UI stavů od ViewModelu.

Během samotného vývoje pak byla hra vyvíjena a testována na několika fyzických zařízeních, včetně tabletu a Android TV pro zajištění správného zobrazení hry na větších displejích. Nejčastějším zařízením byl však mobilní telefon Xiaomi Mi 9T spolu s několika android emulátory pro řádné testování více propojených zařízení během hry.

Již díky použité technologii a předchozím kapitolám je možné chápat, že je hra rozdělena na dvě základní role, které jsou ve hře dále rozšířeny:

- **Hostitel hry** – zařízení, jehož uživatel vytvořil pokerovou hru. Tento režim byl dále rozšířen na 2 varianty:
 - **Hostitel + hráč** – uživatel může hru nejen hostit, ale zároveň se jí i aktivně účastnit.
 - **Pouze hostitel** – zařízení slouží pouze k zobrazení pokerového stolu, bez možnosti zúčastnění se hry na daném zařízení.
- **Připojený hráč** – zařízení, jehož uživatel se připojil k již vytvořené pokerové hře.

Z těchto rolí pak vyplývá, že hra obsahuje 2 odlišná zobrazení pro hráče i pro hostitele a v následujících kapitolách se zaměříme nejen na jejich vývoj z hlediska UI, ale především na implementaci herní logiky, která se v pozadí stará o samotný průběh pokeru. Předtím se ale ještě zaměříme na vytvořené stavy související se hrou poker a na implementaci knihovny EasyLocalGame do hry.

8.1 Stavy ve hře

Tato kapitola popisuje dva základní stavy, která hra z pohledu pokeru obsahuje. Nejde tedy o konkrétní herní fáze, ale o parametry, se kterými se dále pracuje a které hrají klíčovou roli v dalších kapitolách.

Tyto stavy jsou reprezentovány pomocí datových tříd (**data class**), které primárně slouží pro uchování dat a zároveň umožňují jejich poměrně jednoduché kopírování a efektivní aktualizování. [31]

8.1.1 Stav hráče

Každý hráč v libovolné hře musí obsahovat nějaký stav a v této hře to není výjimka. Tento stav je realizován datovou třídou **PlayerState**, která implementuje rozhraní *IPlayerState* z knihovny EasyLocalGame. Pro hráče v pokeru byly definovány následující atributy:

- **nickname** – přezdívka hráče.
- **id** – EndpointID přiřazené po připojení k serveru.
- **avatarId** – vybraný obrázek hráče (referencovaný pomocí *@RawRes*).
- **isServer** – označuje, zda je hráč serverem.
- **isReadyToPlay** – stav hráče, zda je připraven ke hře.
- **isFolded** – označuje, zda hráč složil karty.
- **isPlayingNow** – určuje, zda je hráč aktuálně na tahu.
- **isBigBlind** – označuje, zda hráč platí velký blind.
- **isSmallBlind** – označuje, zda hráč platí malý blind.
- **isDealer** – označuje, zda je hráč aktuální rozdávající karet.
- **money** – množství peněz, které hráč má k dispozici.
- **holeCards** – seznam karet, které hráč drží v ruce.
- **called** – částka, kterou hráč dorovnal v aktuálním kole.
- **allIn** – označuje, zda hráč vsadil všechny své peníze.

Tato třída je **univerzální** pro běžného hráče, tak i pro hráče, který zároveň hostí hru. Rozlišení mezi nimi zajišťuje atribut **isServer**, který je u běžného hráče automaticky nastaven na hodnotu **false**, zatímco u „server hráče“ je nastaven na **true**.

8.1.2 Stav hry

Stav hry, reprezentován datovou třídou **GameState** je klíčovou součástí logiky pokerové aplikace, která uchovává aktuální informace o průběhu hry. Jeho implementace je podobná stavu hráče (**PlayerState**), ale zahrnuje mnohem více atributů, které jsou nezbytné pro řízení herních mechanik.

- **players** – Mapa všech hráčů připojených do hry, kde klíčem je jejich unikátní ID. Každý hráč je reprezentován instancí **PlayerState**.
- **gameReadyPlayers** – Mapa hráčů, kteří jsou připraveni ke hře, spolu s jejich pozicemi u stolu.
- **seatPositions** – Mapa přiřazující hráčům konkrétní místa u stolu.
 - Jedná se o přidanou logiku, která připojenému hráči automaticky nastaví náhodnou pozici u stolu.
- **gameSettings** – Nastavení aktuální hry, které vybírá hráč spravující server.
- **started** – Označuje, zda byla hra **úspěšně** spuštěna.
- **cardsDeck** – Balíček zbývajících karet v aktuální hře.
- **cardsTable** – Karty vyložené na stůl během jednotlivých kol sázení.
- **activeRaise** – Uchovává informaci o aktuálním navýšení sázky (Pár: hráč a částka).
- **bigBlindRaised** – Indikátor, zda hráč s velkým blindem navýšil sázku v prvním kole.
- **bank** – Celková aktuální suma peněz v banku, o kterou hráči hrají.
- **dealerId, smallBlindId, bigBlindId** – ID hráčů, kteří zastávají role dealera, malého a velkého blindu během aktuální hry.
- **roundStartedId** – Hráč, který zahájil aktuální **sázkové** kolo.
- **playingNow** – ID hráče, který je aktuálně na tahu.
- **completeAllIn** – Indikátor, který určuje automatické dokončení hry. K této situaci dochází například tehdy, když všichni aktivní hráči vsadili všechny své žetony (All-in), což znamená, že hra již nevyžaduje žádné další akce od hráčů a pouze postupně odhalí zbývající karty až do určení vítěze.
- **messages** – Seznam zpráv (data class **MessageData** - 8.1.3), které obsahují oznámení a historii ve hře.

- **round** – Číslo aktuálního kola jedné hry.
- **games** – Počet odehraných her.
- **gameOver** – Označuje, zda hra skončila. Slouží pro indikaci spuštění animace ve hře.
- **winningCards** – Sada karet, které byly v dané hře vítězné. Slouží pro indikaci spuštění animace ve hře.

Tento stav **se neustále aktualizuje podle dění ve hře** a umožňuje snadné řízení změn, jako jsou sázky, rozdávání karet, určování dealerů a správa herního banku. Klíčovou roli v tomto procesu hrají atributy `gameReadyPlayers` a `players`.

Atribut **`gameReadyPlayers`** je zvlášť důležitý v situaci, kdy se hráč odpojí během aktivní hry. I když je jeho instance okamžitě odstraněna z mapy **`players`**, v poloze `gameReadyPlayers` musí tento hráč zůstat až do konce aktuálního kola, aby bylo možné správně udržet pořadí hráčů a logiku aktivní hry. Tento hráč už sice na herním stole není přítomen, ale v pozadí hry se o něm jeho stav eviduje i nadále. V průběhu jeho odpojení je hráči automaticky nastaveno **`isFolded`** na hodnotu **`true`**, což znamená, že se stává složeným hráčem a už se tedy nemůže účastnit dalších akcí v aktuálním kole. Toto řešení, ač bylo na začátku složité vymyslet, se ukázalo jako velmi efektivní a spolehlivé pro další správu hry.

8.1.3 Stav hry – doplnění Zpráv a historie

Kromě základních atributů, které definují stav hry, obsahuje třída `GameState` i související data, která jsou nezbytná pro správné zobrazení zpráv a uchovávání historie kol. Jedním z těchto klíčových prvků je třída **`MessageData`**, která slouží pro zobrazení zpráv v průběhu hry a pro správu herní historie.

- **`sender`** – Jméno hráče, který zprávu odeslal (obvykle nenastaven).
- **`messages`** – Seznam textových zpráv (**`UITexts`** - 7.7.2).
- **`cards`** – Seznam karet spojených se zprávou.
- **`history`** – Detail konkrétní hry v podobě historie – třída **`GameHistoryState`**:
 - **`gameNumber`** – Číslo hry v rámci celé herní série.
 - **`tableCards`** – Seznam karet, které byly vyloženy na herní stůl během konkrétního kola.

- **players** – Historie hráčů včetně jejich karet a výsledných kombinací – třída

PlayerHistory:

- **nick** – Přezdívka hráče.
- **cards** – Karty, které měl hráč během konkrétní hry v ruce.
- **cardsCombination** – Výsledná kombinace karet daného hráče.

Typicky tato třída využívá pouze atribut **messages**, který obsahuje textové zprávy pro zobrazení ve hře. Když je však hra ukončena, naplní se i ostatní atributy (**cards** a **history**), což umožňuje uchovat detailní informace o průběhu dané hry.

Hra ale podporuje soukromí hráčů. To znamená, že pokud některý hráč vyhraje před koncem hry (například tím, že ostatní hráči složí své karty), jeho karty, ani karty ostatních hráčů, nebudou v historii zobrazeny. V opačném případě bude historie obsahovat karty výherce i všech **nesložených** hráčů, včetně jejich kombinace.

Třída `MessageData` rovněž obsahuje *@Composable* funkci **ShowMessage**, která je zodpovědná za zobrazení jednotlivých zpráv na obrazovce.

8.2 Implementace ViewModelů z knihovny EasyLocalGame

Jak už bylo několikrát zdůrazněno, hra je rozdělena na **hostitele hry** a **připojeného klienta**, přičemž jejich síťovou komunikaci zajišťuje vytvořená knihovna **EasyLocalGame**. Tato knihovna umožňuje efektivní správu připojení mezi hráči a hostitelem, ale její pouhé přidání do projektu (7.9.2) nestačí – je nutné implementovat **specifické ViewModely**, které zajistí správné fungování herní logiky. Tím jsme schopni rozdělit hru na 2 samostatné celky a vytvořit smysluplnou logiku pro jakoukoliv hru.

8.2.1 PlayerViewModel

Implementace logiky hráče je vytvořena ve třídě **PlayerViewModel**, která dědí ze šablony `PlayerViewModelTemplate` z knihovny **EasyLocalGame**. Na klientské straně se jedná o podstatně méně řádků kódu, jelikož je hlavní logika hry řešena na straně serveru a klient pouze zasílá provedené události.

Uvnitř ViewModelu je vytvořeno několik reaktivních State-Flow proměnných, z nichž první dvě odkazují na právě zmíněné stavy **PlayerState** a **GameState**. Další je proměnná pro volbu zobrazení herní plochy a poslední proměnná udržuje stav, který může hráč ve hře vykonat – **PlayerActionState**:

- **canCheck** – Označuje možnost pokračovat ve hře.
- **callAmount** – Označuje částku, se kterou může hráč dorovnat.
- **raiseAmount** – Označuje částku, se kterou může hráč navýšit bank.
- **canFold** – Označuje možnost složit karty.

Tento stav se aktualizuje v momentě, kdy hráč od serveru přijme podobu jeho nových dat (playerState a gameState). To je provedeno v *přepsané* funkci **serverAction** (7.5.1).

K provádění akcí, které hráč dále posílá na server se používá funkce **onPlayerEvent**, která je blíže popsána ve specifické kapitole níže (8.2.4).

8.2.2 ServerViewModel – pouze hostitel

Implementace serverové části se díky možnosti hraní i na hostovacím zařízení skládá ze dvou specifických ViewModelů, přičemž hlavní řešení logiky je vytvořeno v hlavním ViewModelu **ServerOwnerViewModel**, který pro fungování síťového propojení mezi zařízeními dědí šablonu ServerViewmodelTemplate z knihovny EasyLocalGame.

Z vývojového hlediska se jedná o složitější a komplexnější část, která musí řešit širokou škálu problémů souvisejících s řízením hry, síťovou komunikací a synchronizací herního stavu mezi všemi hráči. O tento stav se opět stará reaktivní proměnná **GameState**, pomocí které může server držet aktuální stav celé hry a řídit pomocí ní jednotlivé akce hráčů.

1. Část – inicializace serveru

Jakmile je server se hrou spuštěn, automaticky se provede jeho inicializace ve funkci **init**, při které se nastaví vybrané **nastavení hry** a spustí **automatické naslouchání změn proměnné _gameState**. Toto automatické naslouchání je důležité například pro **zasílání aktuálního stavu připojeným hráčům** a řešení situací spojených se hrou:

- Automatické složení karet hráčům, kteří se odpojili.
- Kontrola All-in stavu pro automatické zobrazení zbytku karet na stole.
- Časovač při konci hry pro zobrazení animace výsledku hry.

2. Část – akce pro obsluhu serveru.

Jedná se o jednoduchou funkci **onGameEvent**, která dovoluje osobě starající se o vytvořenou hru provádět základní akce. Mezi tyto akce patří:

- **Zapnutí hry** – slouží pro zapnutí pokerové hry (nikoliv pro vytvoření relace/místnosti).
- **Vypnutí serveru** – ukončí všechna spojení a vypne hru do Menu.
- **Vypnutí naslouchání** – ukončí naslouchání pro nová spojení pro šetření baterie.

3. Část – obsluha jednotlivých akcí hráčů

V tomto případě se jedná o *přepsanou* funkci **clientAction**, ve které se do hry promítají změny zasílané hráči, jak je popsáno v kapitole výše (7.4.1).

V bloku akce **ClientAction.EstablishConnection** je tedy postaráno o úspěšné propojení připojeného hráče do hry. Nastaví se mu jeho vybrané parametry (jméno, avatar), přiřadí se mu **id** a počet peněz (vychází z vybraného nastavení hry) a vygeneruje se mu náhodné místo u stolu. Do centra zpráv se následně vloží zpráva, že se připojil nový hráč.

Blok **ClientAction.Disconnect** pak řeší odpojení konkrétního hráče. Z herního stavu se odstraní jeho instance, uvolní se jeho místo u stolu a do **banku** se vloží peníze, pokud měl provedenou akci **call**. Do centra zpráv se pak napíše, že se hráč odpojil ze hry.

Poslední **nejdůležitější** blok je **ClientAction.PayloadAction**. V tomto bloku se zpracovávají akce, které hráč provedl během svého tahu ve hře. Tyto akce jsou definovány vlastním enum typem **MyClientPayloadType**, jehož vlastnosti jsou popsány v následující kapitole (8.2.4). Vstupem do tohoto bloku je odeslaný **payload**, který je nutné deserializovat a získat tak z něj data, které hráč zaslal na server. Příslušná akce je v tomto místě z hlediska autoritativního režimu serveru zkontrolována a v případě platnosti provedena. Pokud tedy například hráč odeslal akci **CALL**, tedy dorovnání aktuální sázky, server zde zkontroluje, zda byla akce odeslána správným hráčem, zda byla tato akce platná pro daného hráče (například dostatek peněz) a zda ji provedl v příslušném čase, který na provedení tahu měl. V případě, že všechno proběhlo tak, jak mělo, aktualizuje se stav hry a přepne se na dalšího hráče. Detailnější popis fungování herního kola však bude probrané později.

4. Část – kontrola tahu a časovač

V pozdější fázi vývoje byl server rozšířen o funkci časovače a automatického kontrolování tahu hráče, což umožnilo plynulý průběh hry i v situacích, kdy hráč zůstane neaktivní. Tento mechanismus je realizován pomocí čtyř klíčových funkcí, které spolupracují na udržení dynamiky hry.

Funkce **handlePlayingPlayer()** je zásadní pro řízení časového limitu pro tah aktuálního hráče. Vytváří časovač využívající proměnnou typu **Job**, která dovoluje běžet paralelně s hlavním programem a může tak sledovat zbývajících čas tahu hráče. Pokud hráč během svého intervalu neprovede žádnou akci, časovač se automaticky vypne a aktivuje se funkce **timeOutAutoMove**.

Funkce **timeOutAutoMove()** poté rozhoduje, jak se má hra zachovat v případě, že hráč neprovedl žádný tah. Pokud je ve hře **aktivní sázka**, je automaticky zavolána funkce **autoFold**, jelikož nelze dovolit, aby hra sama za hráče vsadila jeho peníze. Pokud však žádná sázka neprobíhá, je naopak zavolána funkce **autoCheck**, která hráči umožní pokračovat dál ve hře.

Funkce **autoFold()** se stará o automatické složení karet hráče, který buď překročil časový limit pro svůj tah, nebo se odpojil ze hry. V takovém případě je jeho účast v kole okamžitě ukončena, a hra pokračuje dál. Naopak funkce **autoCheck()** slouží k tomu, aby hráč mohl ve hře pokračovat, pokud jeho tah nevyžaduje žádnou akci. Tento systém tak zajišťuje, že hra nezůstane zablokována kvůli neaktivnímu hráči a plynule pokračuje dál.

8.2.3 ServerViewModel – hostitel jako hráč

Jak již bylo mnohokrát zmíněno, hra obsahuje režim, který umožňuje, aby se hostitel hry mohl sám aktivně zapojit jako hráč. Tuto funkcionalitu zajišťuje třída **ServerPlayerViewModel**, která dědí ze třídy **ServerOwnerViewModel** a tím přebírá její veškerou logiku potřebnou pro fungování hry.

Aby mohl hostitel udržovat svůj stav stejně jako běžný hráč, využívá proměnnou **PlayerState**, která reflektuje jeho aktuální situaci ve hře. Kromě toho implementuje rozhraní **PlayerCoreInterface** (8.2.4), ve kterém přepisuje metodu **onPlayerEvent**, čímž umožňuje zpracovávat herní akce stejně jako klientský hráč ve třídě **PlayerViewModel**.

Při inicializaci **ServerPlayerViewModel** se navíc automaticky nastaví atribut **isServer** na hodnotu **true**, čímž je jasně definováno, že je tento hráč zároveň i hostitelem hry. Ostatní parametry, jako je přezdívka a avatar, se přizpůsobí podle voleb hráče, aby měl stejnou uživatelskou zkušenost jako běžní klienti připojení ke hře.

8.2.4 PlayerCoreInterface

Pro jednotné naslouchání událostem, které hráč může během hry provést, bylo vytvořeno rozhraní **PlayerCoreInterface**, které slouží pro použití u **PlayerViewModel** a **ServerPlayerViewModel**. Toto rozhraní definuje základní funkce a poskytuje implementaci některých obecných metod, které usnadňují rozhodování o možných akcích hráče. Jeho hlavní výhodou je, že jej lze použít jak pro **běžného hráče**, tak i pro **hostitele hry**, který je v režimu hráče.

Rozhraní obsahuje tři povinné metody, které je nutné přepsat:

- **onPlayerEvent(event: PlayerEvents)** – Implementuje reakci na události hráče. Tuto metodu má každá strana (klient i hostitel) implementovanu pomocí *override*.
- **getPlayerState** – Vrací aktuální stav hráče.
- **getGameState** – Vrací aktuální stav hry.

Kromě těchto metod obsahuje rozhraní i několik **již implementovaných funkcí**, které usnadňují výpočty související s možnými herními akcemi:

- **checkEnabled** – Vrací true, pokud může hráč v daném okamžiku ve hře použít akci *Check*.
- **activeCallValue** – Vrací částku, kterou musí hráč dorovnat, aby mohl pokračovat ve hře. Pokud je například *activeRaise* null nebo hráč aktuálně nehraje (*isPlayingNow* == false), vrátí 0.
- **minimalRaise** – Vrací minimální částku, o kterou lze v daném okamžiku navýšit bank. Výpočet bere v úvahu aktuální sázku a pravidla hry (například hodnoty small/big blindů).

Díky této architektuře stačilo implementovat herní logiku pouze **jednou** v rámci rozhraní, což zajišťuje konzistentní chování hráčů bez ohledu na to, zda se jedná o běžného hráče nebo hostitele hry.

Oba ViewModely pak akorát **přepisují** metodu **onPlayerEvent(event: PlayerEvents)**, která definuje reakce na jednotlivé herní události provedené hráčem. Tyto události jsou reprezentovány v sealed třídě **PlayerEvents** a odpovídají klasickým akcím v pokerové hře:

- **Ready** – Označuje, že hráč je připraven ke hře. Používá se i pro „od připravení“.

- **Leave** – Hráč opouští hru. V případě klienta to znamená odpojení od serveru, zatímco u hostitele to znamená ukončení celé hry.
- **ChangeView** – Umožňuje změnit pohled hry pro běžného hráče.
- **Check** – Hráč neprovádí sázku a pokračuje ve hře, pokud je to podle pravidel možné.
- **Call(amount: Int)** – Hráč dorovná sázku ve výši amount, aby zůstal ve hře.
- **Raise(amount: Int)** – Hráč navyšuje sázku o danou částku amount.
- **Fold** – Hráč skládá karty a přestává být aktivním účastníkem daného kola.

Aby mohl klient odeslat svou akci na server a ten ji správně zpracoval, bylo nutné vytvořit **vlastní enum typ pro payload zprávy** (7.6.2). Tento typ (**MyClientPayloadType**) identifikuje jednotlivé akce a umožňuje tak serveru jejich správné rozpoznání a zpracování.

U běžného hráče se tedy **posílá**, například pro akci *Call*, **payload na server** v následující podobě:

```
override fun onPlayerEvent(event: PlayerEvents){
    when(event){
        is PlayerEvents.Call -> {
            val clientPayload = toPayload(
                payloadType = MyClientPayloadType.ACTION_CALL.toString(),
                data = event.amount
            )
            clientManager.sendPayload(clientPayload)
        }
        // Ostatní PlayerEvents akce
    }
}
```

U hráče, který **zároveň zastává roli serveru**, není nutné odesílat payload. Místo toho v bloku „is PlayerEvents.Call -> {}“ **přímo zpracovává** událost pomocí vlastní metody **clientAction** (8.2.2 - 3), protože se jedná o interní akci hráče – serveru.

8.2.5 Dependency Injection – Dagger Hilt

Všechny 3 zmíněné ViewModely jsou vytvořeny pomocí Dependency Injection (DI). Díky tomu lze při běhu aplikace snadno vytvořit jejich instance s proměnnou typu **ConnectionsClient**, která je v Hiltu definována jako singleton.

ViewModely pro **server** jsou navíc vytvořeny pomocí tzv. asistované injekce. Důvodem je, že oba ViewModely vyžadují při svém vytvoření **dynamickou proměnnou**, která se může za běhu aplikace měnit. Jedná se o ID herního nastavení, které vybírá osoba na hostujícím zařízení před začátkem vytvoření pokerové místnosti.

8.3 Další technické záležitosti hry

Po detailním rozboru síťové komunikace a propojení mezi zařízeními je nyní čas zaměřit se na samotnou herní logiku, která tvoří jádro celého **pokerového systému**. Zatímco síťová část se stará o správné doručování akcí hráčů mezi zařízeními, technická implementace samotné hry zajišťuje, že všechny herní mechanismy **probíhají podle pravidel pokeru**.

8.3.1 Manažer hry

Nejdůležitější součástí řízení průběhu pokerové partie je třída **GameManager**. Tato třída slouží jako hlavní řídicí jednotka hry, jelikož zajišťuje její spuštění, spravuje jednotlivé fáze hry a vyhodnocuje jejich výsledek. V rámci GameManageru se nachází klíčové funkce, které koordinují **celý herní cyklus**.

1) Spuštění hry

Funkce **startGame** je hlavním pilířem celého systému, protože je zodpovědná za inicializaci nové hry a její správný průběh. Při spuštění nové partie se provede několik důležitých kroků:

- **Příprava hráčů** – všichni hráči jsou resetováni do výchozího stavu (zruší se jejich předchozí sázky, odstraní se karty z ruky, nastaví se jejich základní atributy).
- **Filtrace hráčů** – do hry jsou zahrnuti pouze ti hráči, kteří jsou připraveni a mají dostatek peněz alespoň na big blind. Pokud nejsou k dispozici alespoň dva takoví hráči, **hra se neodstartuje**.
- **Určení herních rolí** – na základě pořadí hráčů se nastaví:
 - Dealer – hráč, který rozdává karty.
 - Small Blind – hráč, který umístí menší povinnou sázku.
 - Big Blind – hráč, který umístí větší povinnou sázku.
- **Generování karet** – vygeneruje se a zamíchá balíček karet.

- **Příprava na první kolo** – nastaví se první hráč, který začne kolo.

Tato funkce **není určena pouze pro první spuštění hry, ale také pro nastartování každé nové hry po skončení předchozí**. Při tomto opakovaném spuštění aktualizuje klíčové herní atributy, jako je počet odehraných kol, změna dealera a přenesení stavu hráčů mezi jednotlivými hrami.

Na konci této funkce se spouští **simulace prvního kola**, která uvede hru do pohybu.

2) Simulace prvního kola

Simulace „prvního“ kola (funkce **simulateRoundZero**) se stará o **automatické vložení malé a velké sázky** od vybraných hráčů do hry. Jelikož se podle pravidel jedná o povinnou činnost, byla tímto způsobem implementována do hry automaticky. Po umístění těchto sázek jsou hráčům rozdány dvě karty, které se následně odstraní z balíčku, aby nemohly být vylosovány znovu. Jakmile jsou karty rozdány, určí se hráč, který začne první sázkové kolo, čímž už získá možnost zaslat svou akci a hra se tak přepne do dynamického průběhu ovládaným hráči.

3) Provedení tahu a kontrola nového kola

Jakmile hráč provede nějaký tah, jeho akce je, jak bylo vysvětleno v kapitole výš (8.2.2 - 3), nejprve validována přímo ve ViewModelu serveru. Pokud projde kontrolou, funkce **movePlayedCheckRound** ze třídy GameManager se postará o vyhodnocení situace, čímž nastaví nového aktivního hráče, který je na tahu a případně zahájí nové kolo.

Tato funkce tak hraje klíčovou roli ve správě průběhu herních kol. Jejím hlavním **úkolem je rozhodnout, zda je třeba začít nové kolo**, a pokud ano, provést potřebné změny ve stavu hry. V případě, že k novému kolu dojít nemá, funkce **pouze určí dalšího hráče**, který může provést svůj tah.

Funkce nejprve kontroluje, zda hra probíhá v režimu "all-in". V takovém případě se ihned přejde do dalšího kola. Pokud hra v tomto stavu není, funkce analyzuje, zda aktuálně zahraná akce neznamena konec sázkového kola. Pokud znamená (ve hře není aktivní sázka a další hráč na tahu odpovídá hráči, který kolo začínal), hra přechází do dalšího kola a na stůl jsou vyloženy nové karty:

- Druhé sázkové kolo – vyloží se tři karty.
- Třetí a čtvrté sázkové kolo – vyloží se jedna karta.

V případě, že se hra dostane až k „pátému“ kolu, nebo pokud ve čtvrtém kole již probíhá all-in situace, dojde k vyhodnocení hry funkcí **gameOver**.

Výjimečná situace dále nastává v prvním sázkovém kole, kde může dojít k problematickému případu, kdy má Big Blind možnost provést sázku a tím kolo formálně "restartovat". Tento případ byl poměrně složitý na implementaci, protože bylo třeba zajistit, aby se hra nepřerušila a správně proběhl „nový“ běh kola.

Funkce **movePlayedCheckRound** tak zásadním způsobem kontroluje přechod mezi jednotlivými koly. Správně nastavuje pořadí hráčů, přidává nové karty na stůl a rozhoduje o ukončení hry, čímž zajišťuje plynulý a bezchybný průběh hry.

4) Výběr dalšího hráče

K určení dalšího hráče na tahu slouží funkce **getPlayingNow**, která úzce spolupracuje s dříve popsanou funkcí **movePlayedCheckRound**. Funkce nejprve spočítá počet aktivních hráčů, kteří nesložili své karty. Pokud by měl ve hře zůstat pouze 1 hráč, funkce vrátí **null**, čímž signalizuje konec hry.

V opačném případě hledá ID hráče, který bude na tahu. K tomu využívá cyklický seznam hráčů, který je seřazený podle jejich pozic u stolu. Funkce navíc **podporuje 2 režimy** – buď vybírá **dalšího hráče po aktuálním**, nebo v případě zahájení nového kola hledá **prvního hráče po dealerovi**. Při výběru hráče kontroluje, zda ještě nesložil karty a pokud ano, přesune se na dalšího v pořadí.

5) Ukončení a vyhodnocení hry

Pro ukončení hry v případě, že je splněna některé z podmínek pro její konec, se stará funkce **gameOver**. Do banku se vloží všechny vsazené žetony a poté se zjistí, kolik hráčů zůstalo ve hře bez složených karet:

- Nezůstal žádný hráč – tato situace nikdy nenastane.
- **Zůstal jeden hráč** – takový hráč je automaticky vyhlášen za vítěze a získává celý bank. Do centra zpráv se zapíše zpráva, že hráč vyhrál hru před odhalením karet a automaticky se zapne nová hra zavoláním funkce **startGame**.
- **Zůstalo více hráčů** – v takovém případě se vyhodnotí jejich karetní kombinace pro určení **jednoho či více vítězů**, kteří si následně rozdělí bank. Pokud není bank rozdělen rovnoměrně, zbytek se přenesení do další hry. Stav hry se označí jako ukončený, do centra zpráv se uloží historie kola a všichni hráči jsou

nastavení jako neaktivní. Poté se zobrazí animace vítězů a po uplynutí nastaveného času se automaticky zavolá funkce **startGame** pro zahájení nové hry.

8.3.2 Počítání karetních kombinací a vítězů

Další klíčovou součástí pokerové logiky je správné určování vítěze na základě síly karetních kombinací. K tomuto účelu slouží soubory **CardsCalculation.kt** a **CardsUtils.kt**.

Soubor **CardsUtils.kt** obsahuje definice toho, jak si program představuje hrací karty a jak s nimi může pracovat. Konkrétně se jedná o datovou třídu **Card**, která reprezentuje kartu dvěma atributy:

- **Type** – typ karty vycházející z definovaného výčtového typu (enum) **CardType**:
 - HEART (srdce) 
 - DIAMOND (káry) 
 - SPADE (piky) 
 - CLUB (kříže) 
- **Value** – hodnota karty od 2 do 14 (eso)

Tato třída navíc implementuje metodu pro získání obrázku karty ze systémových prostředků **Resources**. Soubor dále obsahuje různé funkce, například pro **generování balíčku**, **odebírání karet z balíčku**, nebo jednoduchý **výpočet síly karetní kombinace**, která se zobrazuje hráčovi ve hře.

Podstatně složitější částí bylo vytvoření algoritmu pro kontrolu kombinací a jeho propojení s celkovým vyhodnocením všech hráčů pro určení vítěze/vítězů. Tento kód je umístěn v souboru **CardsCalculation.kt** a je prakticky rozdělen do 3 částí.

1. Část – Algoritmy pro kontrolu karetní kombinace

Tato část obsahuje několik algoritmů (funkcí), jejichž vstupem je seznam karet, ve kterém se kontrolují karetní kombinace. Například funkce **checkPoker** kontroluje výskyt **čtveřice** neboli čtyř stejných karet. Vzhledem k existenci několika kombinací založených na nalezení stejných karet byla vytvořena obecná funkce **checkNSame**. Tato funkce na základě vstupní hodnoty zjistí, zda se v seznamu karet vyskytuje **N** karet stejné hodnoty. Díky tomu ji lze využít i pro hledání **páru** a **trojice**, které jsou taky součástí kombinace **Full House**.

Funkce **checkEpicFlush** pak kontroluje barvu (pomocí funkce **checkFlush**) a postupku (**checkStraight**) současně, což umožňuje detekovat karetní kombinaci **Královská postupka** (Royal Flush) nebo **Postupka v barvě** (Straight Flush).

Každá kontrolní funkce pak vrací dvojici hodnot v následujícím formátu:

- Boolean – hodnota ANO/NE indikující výskyt kombinace.
- Seznam karet – v případě nalezení kombinace se vrací seznam těchto karet, v opačném případě je seznam prázdný.

2. Část – Vyhodnocení karet jednoho hráče

Pro snadné využití předchozích algoritmů byla vytvořena funkce **evaluatePlayerCards**. Tato funkce se stará o postupné volání funkcí pro kontrolu karetních kombinací. Začíná se nejsilnější kombinací – voláním funkce **checkEpicFlush**. Pokud tato funkce vrátí hodnotu **true**, zkontroluje se, zda se v seznamu karet nachází karta s hodnotou 14 (**nejsilnější** eso). Pokud ano, hráč má Královskou postupku. V opačném případě (eso chybí nebo má **nejslabší** hodnotu 1) má hráč Postupku v barvě. Tímto způsobem se postupně sestupuje až k nejslabší karetní kombinaci (4.3).

Tato funkce opět vrací pár hodnot, kdy první hodnota je **karetní kombinace** (enum **CardsCombination**) a druhá je **seznam karet** tvořících tuto kombinaci.

3. Část – Vyhodnocení karet všech hráčů s určením vítěze

Poslední a nejsložitější částí bylo naprogramování funkce pro nalezení vítěze ze všech hráčů ve hře. Tuto úlohu plní funkce **evaluateGame**, jejíž vstupem je seznamu hráčů (s jejich drženými kartami) a seznam karet na stole. Na základě těchto informací se vyhodnotí každý hráč zvlášť pomocí dříve popsané funkce. Poté se na základě nejlepší dosažené kombinace vyberou všichni hráči, kteří tuto karetní kombinaci mají. Funkce ale musí u každé kombinace počítat s tím, že ji mohou mít 2 nebo více hráčů a rozhodnout tak, kdo bude vítěz. Proto je nezbytné vždy najít tzv. **kicker karty**, které rozhodují o vítězi v případě, že mají hráči stejnou hlavní kombinaci. Pokud jsou i kicker karty stejné, znamená to, že hráči mají identické karty, nebo že karty na stole jsou lepší. V obou případech se ale jedná o remízu.

Funkce vrací seznam vítězů, výherní karetní kombinaci a seznam karet náležící vítězům, přičemž je autor přesvědčen, že by ji bylo možné přepsat mnohem jednodušeji, což by výrazně zlepšilo její čitelnost a usnadnilo pochopení její logiky.

Pro zajištění správného vyhodnocování vítězů byl dále pro kontrolu využit i online nástroj, který je dostupný na následující URL adrese: <https://www.cardplayer.com/poker-tools/odds-calculator/texas-holdem>.

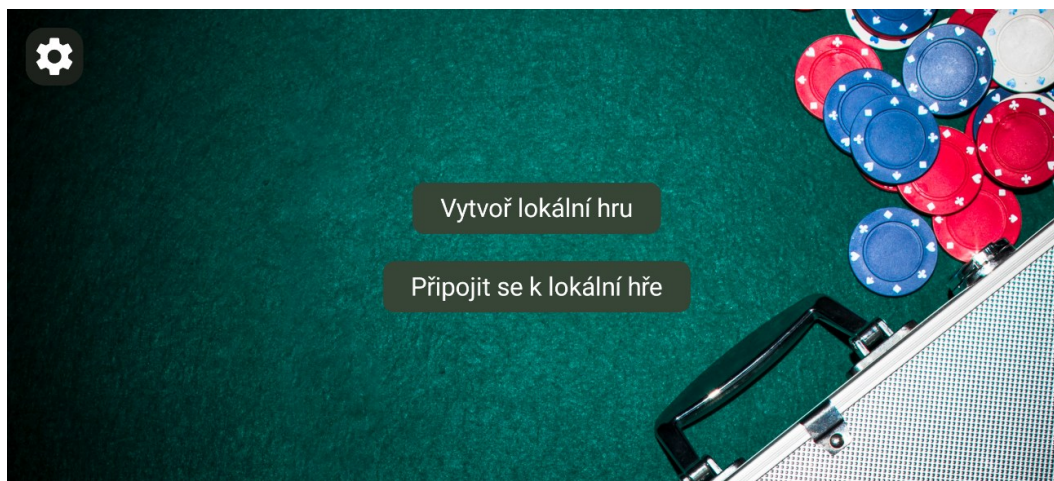
8.4 Grafické představení dalších funkcí

V této části si konečně představíme, jak se dříve popsané funkce promítají do uživatelského rozhraní samotné hry. Zatímco předchozí kapitoly se zaměřily na technickou stránku implementace, nyní se zaměříme na to, jak se tyto funkcionality projevují z pohledu hráče – tedy ve vizuálním a interaktivním prostředí aplikace.

Vzhled samotné pokerové hry, konkrétně hracího stolu, byl inspirován existujícími mobilními pokerovými aplikacemi, jako například *WSOP Poker Texas Holdem Game*, které sloužily jako základní vizuální vodítko pro uspořádání karet, hráčů a banku na stole. Ostatní části uživatelského rozhraní – například hlavní menu, vytvoření nebo připojení do hry, nastavení a další, byly navrženy zcela individuálně, a to s ohledem na specifickou technologii použitou při hraní – tedy hraní s přáteli v jedné místnosti bez potřeby internetu.

8.4.1 Rozvržení hlavního menu

První zobrazená obrazovka po otevření hry je hlavní menu, které obsahuje celkem 3 tlačítka.



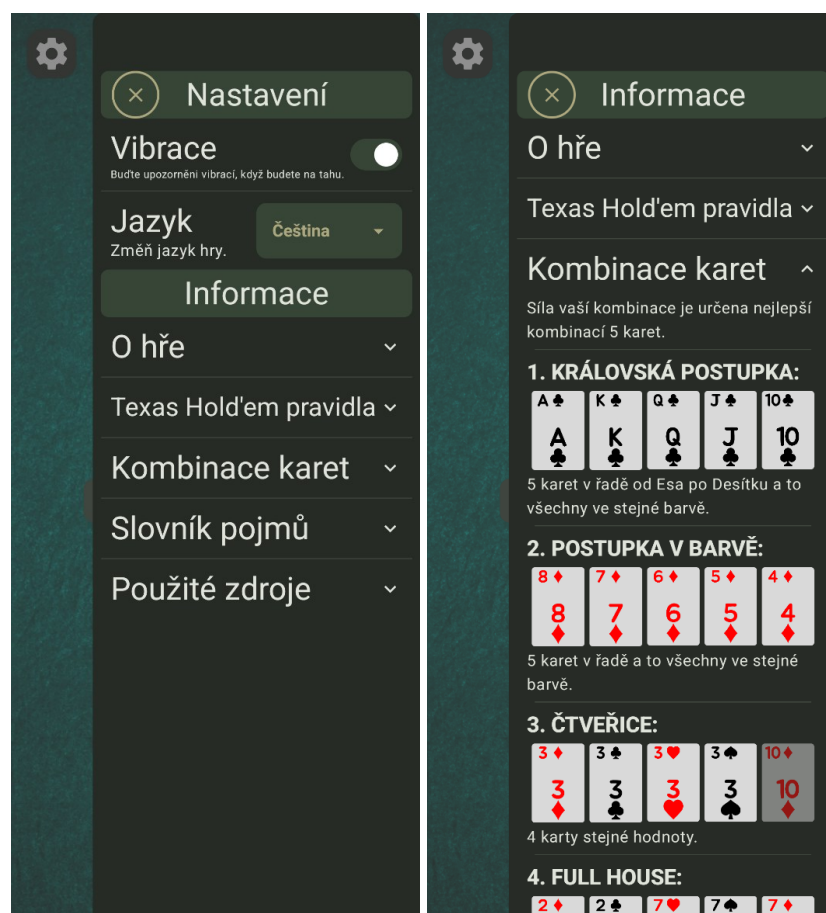
Obrázek 18. Menu hry v režimu na šířku.

(Zdroj: vlastní)

První tlačítko, vyskytující se v levém horním rohu, označuje nastavení, které vyjíždí z pravé strany obrazovky. V následujícím obrázku (Obrázek 19) lze vidět momentální podoba herního nastavení, ve kterém se vyskytuje povolení vibrací a změna jazyka, která se

provede i bez restartování hry. Dále zde existuje sekce **Informace**, která obsahuje tyto položky:

- O hře: Základní informace a účel vytvořené hry.
- Texas Hold'em pravidla: Pravidla, cíl, pojmy a průběh pokerové hry typu Texas Hold'em.
- Kombinace karet: Grafické znázornění 10 existujících kombinací karet.
- Slovník pojmů: Seznam často používaných pojmů v pokerovém světě.
- Použité zdroje: Odkazy na zdroje, které jsou ve hře použity – jedná se o obrázky, animace apod.



Obrázek 19. Nastavení a sekce Kombinace karet.

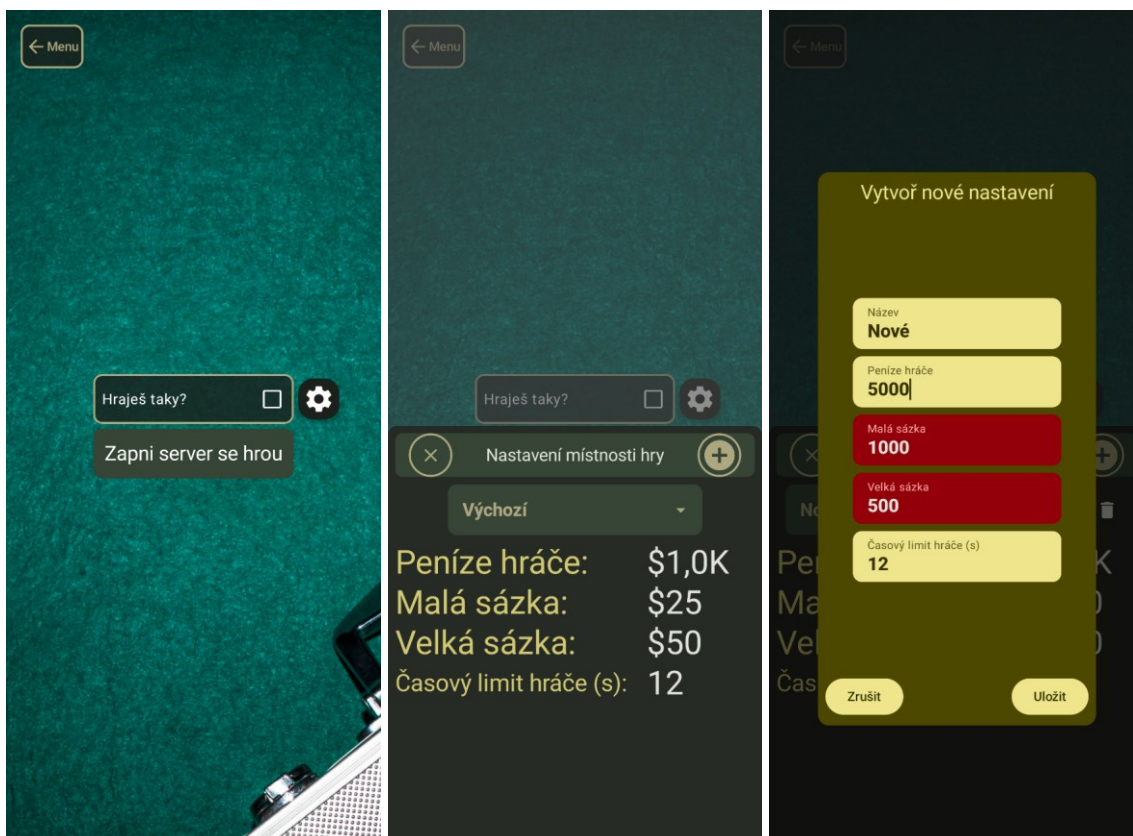
(Zdroj: vlastní)

Další 2 tlačítka pak odkazují na jednotlivé herní módy, které hra obsahuje. Jedná se o **Vytvoření lokální hry** a **Připojení se k lokální hře**.

8.4.2 Vytvoření hry

Po stisku tlačítka „Vytvoř lokální hru“ v Menu hry se přejde na další obrazovku. Z ní je možné se vrátit zpět do Menu buď systémovým gestem nebo tlačítkem v levém horním rohu.

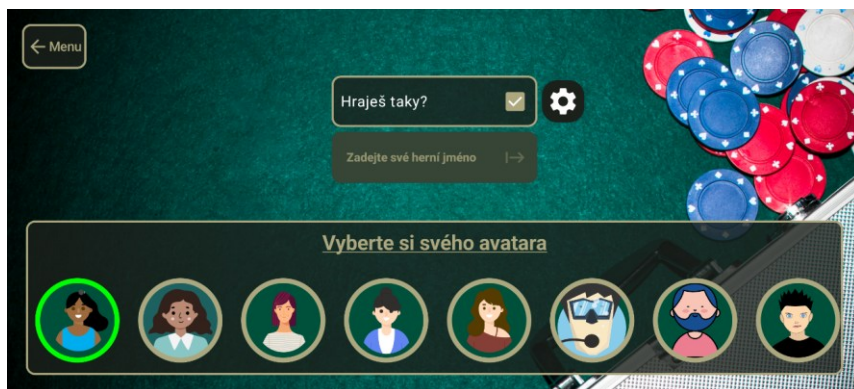
V obrázku níže (Obrázek 20) lze vidět základní strukturu obrazovky, ve které se vyskytuje tlačítko indikující **zapnutí serveru** v hostujícím režimu bez hráče. **Ozubené** kolečko pak značí herní nastavení, které může osoba zapínající server upravit tak, aby hra vyhovovala všem hráčům. Toto nastavení je závislé na orientaci zařízení – pokud je v portrétním režimu, tak nastavení vyjede ze spodní části obrazovky, v opačném případě přijíždí zleva. Tlačítkem „+“ lze vytvořit a uložit nové nastavení pro další použití. Formulář zároveň validuje jednotlivé položky, jako například to, že Malá sázka nemůže být větší než Velká sázka, nebo že kterákoliv sázka nemůže být větší než nastavené Peníze hráče.



Obrázek 20. Vytvoření hry a herní nastavení.

(Zdroj: vlastní)

Na obrazovce se dále vyskytuje kontrolní tlačítko, po jehož zaškrtnutí vpluje zprava výběr animovaných, herních avatarů a ve stejném momentě se tlačítko „Zapni server se hrou“ změní ve vstupní pole, do kterého zadá hráč svou přezdívku, čímž dá najevo, že dané zařízení bude bráno jako hráč zároveň hostující hru. Tento systém zobrazení je stejný jako po přechodu do sekce „Připojit se k lokální hře“ v Menu.



Obrázek 21. Výběr herních avatarů.

(Zdroj: vlastní)

8.4.3 Herní stůl

Nejobtížnější obrazovkou celé aplikace bylo bezpochyby vykreslení herního stolu, kolem kterého si hráči „sedají“. Každý hráč má na svém místě zobrazeného avatara, jméno, aktuální množství peněz, prostor pro odevzdané žetony a indikátory jako žeton rozdávajícího (dealer), malý a velký blind, či kontrolku připravenosti. Ve středu stolu se nacházejí sdílené herní karty, bank a jeden aktuální řádek z Centra zpráv.

Tato obrazovka si prošla rozsáhlým vývojem a bylo vytvořeno přibližně osm různých verzí, než vznikla finální podoba. Samotné rozložení a velikost stolu hrály klíčovou roli i pro zbytek vývoje. Bylo nutné navrhnout tzv. „zasedací“ **algoritmus**, který každého hráče přiřadí na předem definované místo u stolu. Umístění hráče se zároveň musí správně přepočítávat podle rozlišení displeje zařízení, aby bylo zajištěno, že nebude překrytý přední kamerou, nebude mít málo prostoru kolem sebe, ani se neocítne mimo obrazovku.

Finální rozložení hráčů, **kterých může být maximálně 10**, kolem stolu je znázorněno na obrázku níže (Obrázek 22). Na obrázku jsou žetony zobrazeny u všech hráčů záměrně, bez ohledu na herní situaci, aby bylo možné dobře ilustrovat jejich rozmístění.

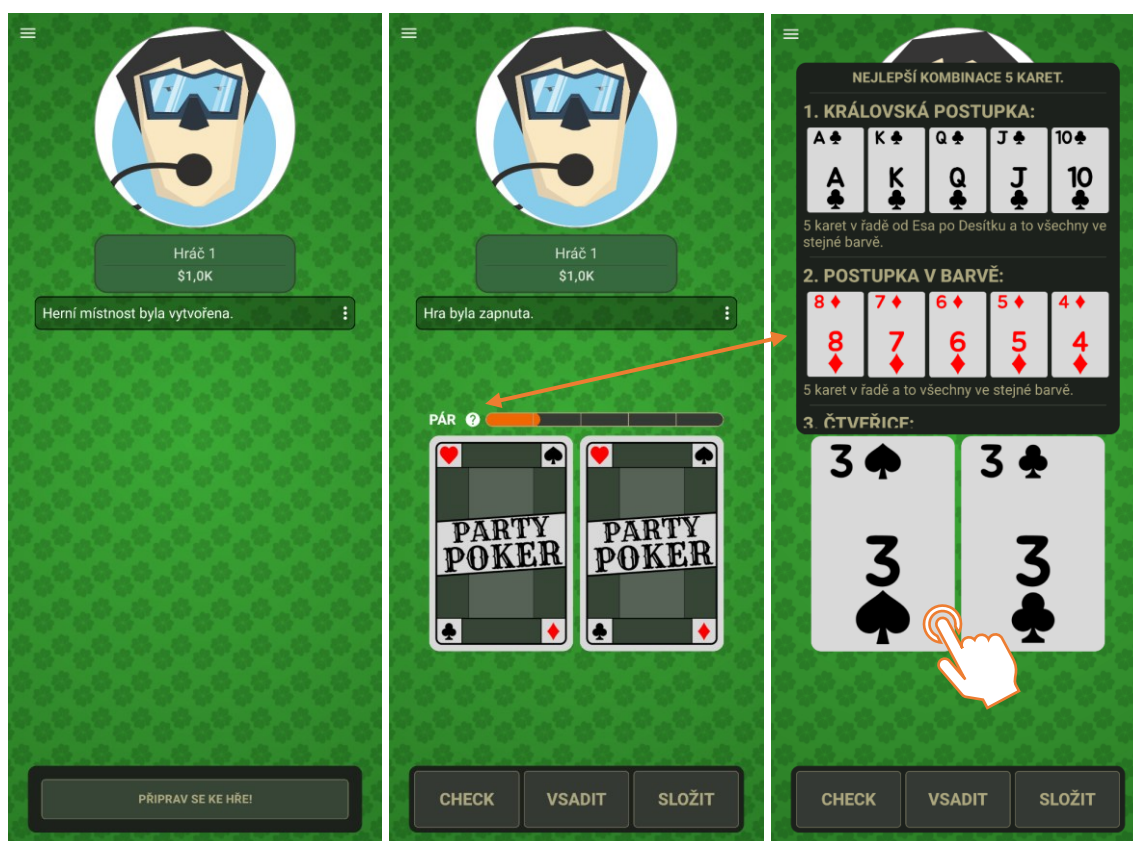


Obrázek 22. Rozložení hráčů kolem stolu z pohledu serveru.

(Zdroj: vlastní)

Pokud je hra spuštěna v režimu bez server hráče, připojeným klientům se zobrazí odlišná obrazovka. Na následujícím obrázku (Obrázek 23) je vidět výrazně jednodušší rozhraní, které obsahuje základní informace o hráči, Centrum zpráv, 2 skryté karty (zobrazí se po dotyku – ale až po jejich skutečném rozdání během hry) a sekci s ovládacími tlačítky pro vykonávání pokerových akcí.

Tento mód je optimalizován pro portrétní režim (na výšku) a **poskytuje hráči soukromý prostor** pro sledování svých karet. Je tedy ideální pro případy, kdy je herní stůl sdílen mezi všichni hráče například na televizi, tabletu nebo jiném větším zařízení uprostřed místnosti.



Obrázek 23. Soukromý pohled hráče.

(Zdroj: vlastní)

Pokud se do hry zapojí také serverový hráč, nebo se hráč ručně přepne z privátního režimu do klasického, zobrazí se jiné rozložení herní obrazovky. Toto rozvržení se z velké části podobá již zmíněnému zobrazení herního stolu (Obrázek 22). Hra se přepne do režimu na šířku a zobrazí se kompletní stůl včetně ostatních hráčů. Informace o daném hráči a ovládací tlačítka se však vždy nacházejí ve spodní části obrazovky.

Z tohoto důvodu byl implementován **algoritmus, který dynamicky „otáčí“ pořadí hráčů** tak, aby je měl každý hráč na svém zařízení ve správném kruhovém pořadí. Díky

tomu má každý hráč pocit, že sedí vždy na stejném místě u stolu, a zároveň vidí ostatní hráče v intuitivním rozmístění kolem něj. Zároveň dochází k drobné změně v rozmístění míst u stolu, aby se i nadále vešlo maximálních 10 hráčů. Na pravé i levé straně obrazovky jsou nyní umístěna dvě herní místa, a jedno další se nachází ve spodní části obrazovky nalevo od aktuálního hráče.



Obrázek 24. Rozložení hráčů u stolu z pohledu **Hráč 1**.

(Zdroj: vlastní)



Obrázek 25. Rozložení hráčů u stolu z pohledu **Hráč 2**.

(Zdroj: vlastní)

8.4.4 Efekty a animace ve hře

Hra obsahuje i řadu různých animací a efektů, které ji vizuálně oživují. K již zmíněnému „vplynutí“ různých sekcí nastavení nebo avatarů přibývají animace přímo ve hře během hraní, které jsou znázorněny na přiloženém obrázku (Obrázek 26). Jelikož je obrázek pouze statický, je možné efekty ukázat pouze symbolicky – jejich výskyt na obrázku je označen čísly v závorkách.

Jedním z nejvýraznějších prvků je **časovač tahu hráče**, který se zobrazuje v kruhu kolem avatara. Barva postupně přechází ze zelené do červené a po uplynutí času se efekt zcela vytratí a hra přechází na dalšího hráče (1).

Další výraznou animací je **vyhodnocení kola**. Pokud v závěru kola zůstane více hráčů, hry vyhodnotí karetní kombinace a zobrazí karty všech nesložených hráčů. Výherní kombinace je zvýrazněna **zářivým efektem** (2). Navíc se nad kartami na stole objeví čára s časovačem, která odpočítává start nové hry (3) a do Centra zpráv se napíše zpráva (4).

Animaci má i **indikátor síly kombinace**, který se zobrazuje jako posuvník nad ovládacími tlačítky (v režimu klasickém), nebo nad kartami (v režimu privátním). S rostoucí kvalitou kombinace se mění jeho barva i délka (5).



Obrázek 26. Obrázek znázorňující animace ve hře.
(Zdroj: vlastní)

Menší, ale přesto příjemný efekt tvoří **přeskakující číslice u částek** v banku a u jednotlivých hráčů – tento efekt však na obrázku vidět nelze.

8.4.5 Centrum zpráv

Jednou z důležitých součástí herního rozhraní je také **Centrum zpráv**, které se zobrazuje formou vyskakovacího okna. Toto centrum slouží ke sdílení veškerých herních informací mezi všemi hráči.

Zobrazují se zde systémové zprávy jako např. připojení nebo odpojení hráče, změna jeho stavu připravenosti, spuštění hry nebo také automatická oznámení v průběhu herního kola. Velkým přínosem bylo vytvoření detailní **historie každého kola**, která informuje o tom, kdo kolo vyhrál, jaké byly karty na stole, a jaké karty měli jednotliví hráči – samozřejmě pouze v případě, že své karty nesložili.

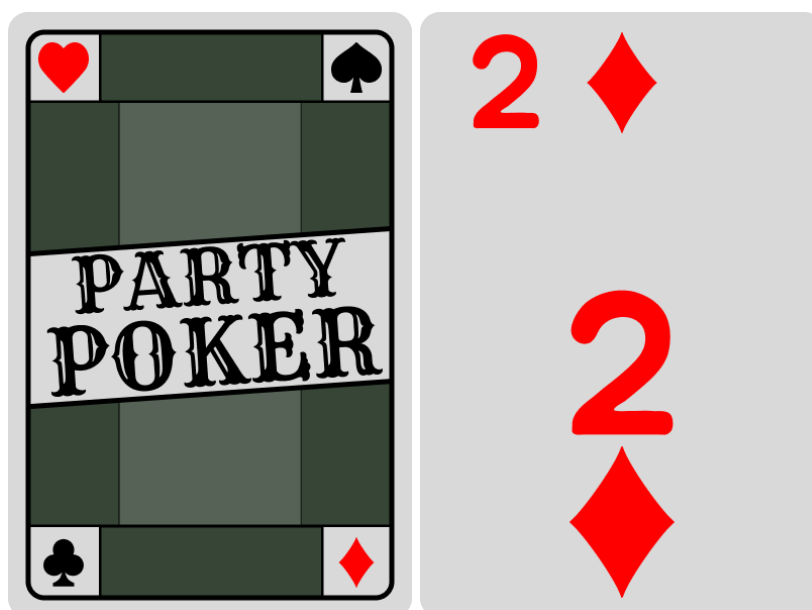


Obrázek 27. Centrum zpráv zobrazující různé pohledy na historii kol.
(Zdroj: vlastní)

8.4.6 Ostatní grafika ve hře

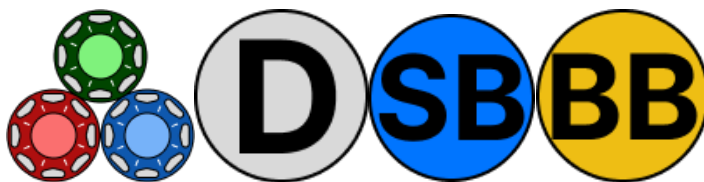
Součástí hry je také jednoduchá tvorba grafických prvků, jako jsou žetony, karty nebo postavy avatárů.

Veškeré animované obrázky použité ve hře (například avataři a chybová hlášení) jsou formátu **Lottie**. Odkazy na tyto animace jsou uvedeny v příloženém seznamu (příloha p II). Ve stejné příloze jsou dále také odkazy na 4 karetní symboly, které byly použity při tvorbě herních karet v online nástroji **Figma**. V něm bylo vytvořeno všech 52 herních karet + 1 zadní karta s názvem hry.



Obrázek 28. Vytvořený vzhled zadní a přední karty.
(Zdroj: vlastní – nástroj Figma)

Kromě karet zde vznikly i žetony pro bank a speciální žetony označující dealera, malý a velký blind.



Obrázek 29. Vytvořené herní žetony. Zleva: Bank, Dealer, Malý blind, Velký blind.

(Zdroj: vlastní – nástroj Figma)

8.5 Vydání hry na Google Play

Jak již bylo naznačeno v úvodu, hra byla již během vývoje nasazena do obchodu Google Play. První **interní testovací verze** byla vydána 8. června 2024 a díky ní bylo možné hru testovat na více zařízeních v domácnosti a okolí.

Od 13. verze aplikace následovalo spuštění **otevřeného testování**, které hru pomalu vypouští do světa a dovoluje různým testerům si ji nainstalovat a vyzkoušet.

Produkční vydání přišlo s 18. verzí aplikace 16. srpna 2024. V tomto momentu bylo nutné aplikaci řádně nastavit v obchodu Google Play. Základem bylo vyplnění různých formulářů, např. zda aplikace obsahuje reklamy, zda ukládá data na vzdálený server, zda obsahuje nákupy v aplikaci anebo bylo nutné vytvořit dokument **Zásady ochrany osobních údajů**, který je dostupný na následujícím odkazu: <https://github.com/grasski/PartyPokerPolicy/blob/main/README.md>.

Jelikož se jedná o hru, která má prvky simulovaného gamblingu, bylo navíc nutné projít schvalovacím procesem, který se liší v závislosti na zemi – většinou byl získán titulek 18+, ačkoliv v Jižní Koreji hra neprošla vůbec.



Obrázek 30. QR kód pro Google Play.

(Zdroj: vlastní - <https://www.qrcode-monkey.com>)

ZÁVĚR

Cílem této práce bylo vytvořit knihovnu pro mobilní platformu, která by umožňovala snadnou tvorbu multiplayerových her v lokálním prostředí s využitím frameworku Nearby Connections. Společně s knihovnou byla vyvíjena i první hra, která demonstruje její funkčnost a zároveň slouží jako odrazový můstek pro budoucí projekty.

Prvním krokem k realizaci tohoto cíle bylo studium vybrané technologie Nearby Connections a její integrace do Android aplikace pomocí programovacího jazyka Kotlin. Vývoj knihovny byl doprovázen několika technickými problémy, zejména spojenými s nestabilitou této technologie v externích projektech. Tyto potíže se však podařilo překonat a knihovna tak mohla být publikována do online úložiště JitPack, které umožňuje její snadné využití ostatními vývojáři formou závislosti v jejich projektech.

Po zajištění spolehlivé komunikace mezi blízkými zařízeními bylo nutné navrhnout hru, která by vhodně prezentovala veškerou funkcionalitu knihovny. Výsledkem je pokerová hra, jejíž vývoj probíhal souběžně s vývojem samotné knihovny.

Výsledná hra obsahuje celkem 2 možnosti, jak zapnout pokerovou hru a sdílet ji mezi přáteli. První možností je hra s hostujícím zařízením, které slouží pouze jako sprostředkovatel herního stolu ostatním hráčům. Tento režim se nazývá privátní a poskytuje každému hráči soukromý prostor pro své vlastní karty. Zároveň ale existuje i možnost, ve které se do hry může zapojit i zařízení hostující hru. K tomuto režimu slouží jiné, poměrně komplexnější zobrazení herní plochy, jelikož v něm není možné centrálně sdílet herní plochu, a proto vidí každé připojené zařízení celý herní stůl s vlastními kartami a okolními hráči.

Samotná hra byla vytvořena pomocí frameworku Jetpack Compose, který je primárně určen pro tvorbu mobilních aplikací. Nicméně pro jednodušší hry bez fyzikálních enginů, jako je právě tato pokerová hra, se ukázal jako zcela dostačující.

Celkově tedy byla vytvořena mobilní knihovna na platformu Android, která poskytuje ostatním vývojářům nástroj pro tvorbu her, zaměřených na lokální multiplayer. Spolu s tím vznikla hra Poker, která vytvořenou knihovnu implementuje a dovoluje tak hráčům zahrát si poker s přáteli v jedné místnosti, bez nutnosti vlastnění fyzické hry.

Do budoucna je, pokud to bude možné, plánováno rozšíření knihovny pro podporu multiplatformního vývoje pomocí Kotlin Multiplatform, ale také nové funkce ve hře. Mezi plánované funkce patří například možnost uložení a načtení rozehrané hry, výukové režimy

pro zlepšení herních dovedností (např. výběr nejlepší karetní kombinace), nebo režim pro jednoho hráče proti „botům“.

SEZNAM POUŽITÉ LITERATURY

- [1] KARTHIKEYAN, Kaavya. The History of Mobile Gaming. *Gameopedia* [online]. 2022 [cit. 2025-02-01]. Dostupné z: <https://www.gameopedia.com/the-history-evolution-and-future-of-mobile-gaming/>
- [2] MÄYRÄ, Frans a Kati ALHA. Mobile Gaming. *ResearchGate* [online]. 2020, 107-120 [cit. 2025-02-01]. Dostupné z: doi:10.4324/9780429351815-9
- [3] VOGELTANZ, Tomáš. *Vývoj mobilních her a monetizace*. 12. přednáška předmětu Vývoj počítačových her. Zlín: Univerzita Tomáše Bati., 2024.
- [4] VOGELTANZ, Tomáš. *Herní design*. 2. přednáška předmětu Vývoj počítačových her. Zlín: Univerzita Tomáše Bati, 2024.
- [5] Nejčastější žánry mobilních her za rok 2023. *Statista* [online]. 2024 [cit. 2025-02-04]. Dostupné z: <https://www.statista.com/statistics/1446849/top-gaming-genres-mobile-worldwide/#statisticContainer>
- [6] Backbone ovladač. In: *Backbone.com* [online]. [cit. 2025-02-04]. Dostupné z: <https://backbone.com/cz/products/backbone-one-usb-c/>
- [7] FREEMAN, Will. Motivace hráčů. *GameAnalytics* [online]. 2022 [cit. 2025-02-12]. Dostupné z: <https://gameanalytics.com/blog/getting-mobile-multiplayer-right-part-1-history-genres-and-motivations/>
- [8] VOGELTANZ, Tomáš. *Síťová komunikace a multiplayer ve hrách*. 11. přednáška předmětu Vývoj počítačových her. Zlín: Univerzita Tomáše Bati, 2024.
- [9] Hot-seat a Split-screen. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2024 [cit. 2025-02-25]. Dostupné z: https://cs.wikipedia.org/wiki/Videohra_pro_v%C3%ADce_hr%C3%A1%C4%8D%C5%AF
- [10] SongPop Party. *Apple* [online]. 2021 [cit. 2025-02-28]. Dostupné z: <https://apps.apple.com/us/app/songpop-party/id1551639567?itscg=30800&itsct=songpopparty>
- [11] Přehled NearbyConnectionAPI. *Google Developers* [online]. 2023 [cit. 2025-02-25]. Dostupné z: <https://developers.google.com/nearby/connections/overview>
- [12] Strategie NearbyConnectionAPI. *Google Developers* [online]. 2023 [cit. 2025-02-25]. Dostupné z: <https://developers.google.com/nearby/connections/strategies>
- [13] Android oprávnění pro práci s NearbyConnectionAPI. *Google Developers* [online]. 2023 [cit. 2025-02-25]. Dostupné z: https://developers.google.com/nearby/connections/android/get-started#request_permissions
- [14] Inzerování a Objevování. *Google Developers* [online]. 2023 [cit. 2025-03-08]. Dostupné z: <https://developers.google.com/nearby/connections/android/discover-devices>
- [15] Správa připojení s NearbyConnectionAPI. *Google Developers* [online]. 2018 [cit. 2025-03-08]. Dostupné z: <https://developers.google.com/nearby/connections/android/manage-connections>
- [16] Výměna dat s NearbyConnectionAPI. *Google Developers* [online]. 2020 [cit. 2025-03-08]. Dostupné z: <https://developers.google.com/nearby/connections/android/exchange-data>

- [17] JENSCHKE, Tomas Lagos, Marcelo DIAS DE AMORIM a Serge FDIDA. Obrázek zobrazující Flow-diagram pro NearbyConnectionAPI. In: *ScienceDirect* [online, obrázek]. 2023 [cit. 2025-03-08]. Dostupné z: <https://ars.els-cdn.com/content/image/1-s2.0-S2542660523002184-gr1.jpg>
- [18] Bluetooth Low Energy. *Android Developers* [online]. 2024 [cit. 2025-04-01]. Dostupné z: <https://developer.android.com/develop/connectivity/bluetooth/ble/ble-overview>
- [19] *Pojmy v pokerových hrách.* [online]. [cit. 2025-03-01]. Dostupné z: <https://iprc.soest.hawaii.edu/users/jimp/personal/poker/glossary.html>
- [20] Limity sázek v pokeru. *Poker.md* [online]. 2021 [cit. 2025-03-01]. Dostupné z: <https://poker.md/cs/s%C3%A1zkov%C3%A9-limity-v-pokeru/>
- [21] Pravidla Texas Hold'em pokeru. *Heltakoop* [online]. 2014 [cit. 2025-02-26]. Dostupné z: <https://www.heltakoop.cz/pravidla/pravidla-poker>
- [22] Pravděpodobnosti v Texas Hold'em. *PokerStrategy* [online]. 2009 [cit. 2025-02-26]. Dostupné z: <https://www.pokerstrategy.com/strategy/various-poker/texas-holdem-probabilities/>
- [23] Pravděpodobnosti v Texas Hold'em 2. *Wikipedia* [online]. 2025 [cit. 2025-02-26]. Dostupné z: https://en.wikipedia.org/wiki/Poker_probability#7-card_poker_hands
- [24] Android knihovna. *Android Developers* [online]. 2025 [cit. 2025-03-11]. Dostupné z: <https://developer.android.com/studio/projects/android-library>
- [25] DIXIT, Prashant. Tvorba Android knihovny. *Medium* [online]. 2023 [cit. 2025-03-11]. Dostupné z: https://medium.com/@prashantdixit_87612/creating-and-publishing-your-own-android-library-in-kotlin-2023-part-1-3e542dad4ca0
- [26] Ukládání dat pomocí Jetpack DataStore. *Android Developers* [online]. 2025 [cit. 2025-03-07]. Dostupné z: <https://developer.android.com/topic/libraries/architecture/datastore>
- [27] SPRYN, Oliver. *Kontext vs ViewModel* [online]. 2022 [cit. 2025-05-07]. Dostupné z: <https://www.youtube.com/shorts/kU9YzhovIvU>
- [28] Nastavení Jitpack. *Jitpack.io* [online]. [cit. 2025-03-11]. Dostupné z: <https://docs.jitpack.io/android/>
- [29] Publikování knihovny na server JitPack. *Medium* [online]. 2022 [cit. 2025-03-11]. Dostupné z: <https://medium.com/@theAndroidDeveloper/creating-and-publishing-your-own-android-library-an-in-depth-guide-26de164c96c2>
- [30] BOURDAKOS, Nick. Android <> iOS komunikace. *Github* [online]. 2023-2024 [cit. 2025-03-28]. Dostupné z: <https://github.com/google/nearby/issues/1720>
- [31] KOTLINLANG. Data class. *Kotlinlang.org* [online]. 2024 [cit. 2025-03-29]. Dostupné z: <https://kotlinlang.org/docs/data-classes.html>

SEZNAM OBRÁZKŮ

Obrázek 1. Nejčastější žánry mobilních her za rok 2023.	20
Obrázek 2. Backbone ovladač pro mobilní telefony.....	21
Obrázek 3. Predikce stavů v online hře.	25
Obrázek 4. Snímek zobrazující připojování do herní místnosti.....	26
Obrázek 5. Flow diagram navázání komunikace s Nearby Connections API.	35
Obrázek 6. Balíček francouzských karet.	37
Obrázek 7. Ukázka Vysoké karty.	42
Obrázek 8. Ukázka Páru.	43
Obrázek 9. Ukázkou Dvou párů.	43
Obrázek 10. Ukázka Trojice.	44
Obrázek 11. Ukázka Postupky.....	45
Obrázek 12. Ukázka Barvy.	46
Obrázek 13. Ukázka kombinace Full House.	46
Obrázek 14. Ukázka Čtveřice.	47
Obrázek 15. Ukázka kombinace Postupka v barvě.....	48
Obrázek 16. Ukázka Královské postupky.....	48
Obrázek 17. Úspěšné nahrání knihovny na JitPack.....	72
Obrázek 18. Menu hry v režimu na šířku.	90
Obrázek 19. Nastavení a sekce Kombinace karet.....	91
Obrázek 20. Vytvoření hry a herní nastavení.	92
Obrázek 21. Výběr herních avatarů.	93
Obrázek 22. Rozložení hráčů kolem stolu z pohledu serveru.....	93
Obrázek 23. Soukromý pohled hráče.....	94
Obrázek 24. Rozložení hráčů u stolu z pohledu Hráč 1	95
Obrázek 25. Rozložení hráčů u stolu z pohledu Hráč 2	95
Obrázek 26. Obrázek znázorňující animace ve hře.	96
Obrázek 27. Centrum zpráv zobrazující různé pohledy na historii kol.	97
Obrázek 28. Vytvořený vzhled zadní a přední karty.	97
Obrázek 29. Vytvořené herní žetony. Zleva: Bank, Dealer, Malý blind, Velký blind.	98
Obrázek 30. QR kód pro Google Play.	98

SEZNAM TABULEK

Tabulka 1. Přehled oprávnění pro Nearby Connections API.....	32
--	----

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

<T>	Značení obecné třídy
AAR	Android Archive
API	Application Programming Interface
BLE	Bluetooth Low Energy
ID	Identifikátor
iOS	iPhone Operating System
MVI	Model View Intent
MVVM	Model View ViewModel
Např.	Například
Popř.	Popřípadě
PvP	Player versus Player
SDK	Software Development Kit
TCP	Transmission Control Protocol
TV	Televize
UDP	User Datagram Protocol
UI	User Interface

SEZNAM PŘÍLOH

Příloha P I: Struktura odevzdaných dat

Příloha P II: Stažené grafické materiály

PŘÍLOHA P I: STRUKTURA ODEVZDANÝCH DAT



*Vytvořeno webovým nástrojem <https://tree.nathanfriend.io>.

PŘÍLOHA P II: STAŽENÉ GRAFICKÉ MATERIÁLY

Vektory karetních symbolů:

- <https://www.svgrepo.com/svg/399971/blackclubsuit>
- <https://www.svgrepo.com/svg/479183/diamond-mark-3>
- <https://www.svgrepo.com/svg/532473/heart>
- <https://www.svgrepo.com/svg/479189/spade-1>

Lottie animovaní avataři:

- <https://lottiefiles.com/free-animation/profile-avatar-of-young-woman-WIppZWULvi?from=search>
- <https://lottiefiles.com/free-animation/female-05-4Mz17Snijt?from=search>
- <https://lottiefiles.com/free-animation/avatar-female-vys3zMXt1F?from=search>
- <https://lottiefiles.com/free-animation/woman-avatar-xF8u2MuJJV?from=search>
- <https://lottiefiles.com/free-animation/avatar-woman-short-hair-ZbixzqM6fP?from=search>
- <https://lottiefiles.com/free-animation/avatar-2K8fgWn5xp?from=search>
- <https://lottiefiles.com/free-animation/avatar-animation-CcVadKDg5s?from=search>
- <https://lottiefiles.com/free-animation/avatar-M96ZOrdG4C?from=search>
- <https://lottiefiles.com/free-animation/avatar-beard-man-blink-wQ12gw4iUt?from=search>
- <https://lottiefiles.com/free-animation/diver-avatar-3JjbOPJD6B?from=search>

Lottie animované obrázky:

- <https://lottiefiles.com/free-animation/error-404-AQdcQKvXl1?from=search>
- <https://lottiefiles.com/free-animation/access-location-Um17zCqVsH?from=search>
- <https://lottiefiles.com/free-animation/error-occurred-iMz20W0ZOp?from=search>
- <https://lottiefiles.com/free-animation/reject-2-aXBZeSOXoi>

Ostatní grafika:

- https://www.freepik.com/free-vector/green-casino-poker-table-texture-game-background_48197468.htm
- https://www.freepik.com/free-vector/green-geometric-background-patricks-day-colors_5685390.htm
- https://www.freepik.com/free-photo/aluminium-suitcase-with-casino-chips-green-background_2900729.htm