FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Serverless multiplayer hra pro mobilní zařízení
Student:	Bc. Jakub Homolka
Vedoucí:	Ing. Josef Gattermayer
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Cílem práce je vytvořit mobilní hru pro Android pro dva hráče.

Pokyny:
1. Vytvořte kompletní specifikaci pravidel hry, včetně systému bodování pro více kol hry.
2. Analyzujte možné způsoby přímého propojení dvou mobilních zařízení (bluetooth, wifi, ...) a vytvořte prototypy těchto způsobů propojení. Vyberte z nich nejvhodnější pro budoucí hru.
3. Navrhněte, implementujte a otestujte mobilní aplikaci pro Android, která bude využívat nalezený nejvhodnější způsob propojení a implementovat pravidla hry.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 28. ledna 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Serverless multiplayer hra pro mobilní zařízení

Bc. Jakub Homolka

Katedra softwarového inženýrství
Vedoucí práce: Ing. Josef Gattermayer, Ph.D.

9. ledna 2019

Poděkování

Rád bych poděkoval svému vedoucímu práce ing. Josefu Gattermayerovi, dále pak své manželce Daniele Homolkové, která mě po celou dobu podporovala, provedla závěrečnou korekturu celé práce a také se mnou hru testovala. Poděkovat bych chtěl i všem dalším testerům, jmenovitě Janu Baštovi, Elišce Soukupové, Karolíně Soukupové, Matějovi Homolkovi, Pavlu Bohatému, Petře Bohaté a Lucii Bohaté.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

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

V Praze dne 9. ledna 2019 .....

Odkaz na tuto práci

Homolka, Jakub. *Serverless multiplayer hra pro mobilní zařízení*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Tato práce se zabývá vytvořením hry na mobilní zařízení pro dva hráče. Hra je implementována pro operační systém Android a nevyužívá žádný server, komunikace mezi zařízeními probíhá na přímo.

Pravidla hry vycházejí ze hry, která se původně hrála na papíře. Hráči si určí libovolné delší slovo a z písmen tohoto slova se pak v časovém limitu snaží vytvořit co nejvíce dalších slov.

V práci se nejprve zabývám podrobnou specifikací hry a definuji pravidla.

Následně provádím analýzu způsobů přímého propojení dvou mobilních zařízení. Po teoretické části vytvářím prototypy, na kterých zjišťuji, jaká technologie je pro mou práci vhodnější.

V další části se věnuji samotnému návrhu a implementaci aplikace, která využívá nalezený nejvhodnější způsob propojení. Mimo jiné se zde zabývám i grafickým designem aplikace.

V poslední kapitole popisuji uživatelské testování a řešení všech objevených problémů.

Klíčová slova hra na mobilní zařízení, slovní hra, pro dva hráče, serverless, OS Android

Abstract

The thesis deals with the creation of a two-player mobile game. The game is implemented for the Android operating system and does not use any server, the devices communicate directly.

The rules of the game are based on the game originally played on paper. The players choose any longer word and try to create as many words as possible within the time limit.

In my work, I first deal with the detailed game specification and define the rules.

I then analyze how to directly connect two mobile devices. After the theoretical part, I create prototypes to find out what technology is best for my work.

In the next part, I'm working on the design and implementation of the application that uses the most appropriate way to connect. Among other things I also deal with the graphic design of the application.

In the last chapter, I describe user testing and solving all the problems I have discovered.

Keywords mobile game, word game, multiplayer, serverless, OS Android

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Původ hry

Vždy jsem měl rád hry se slovy a písmeny. Obzvlášť oblíbená byla ta, které jsme říkali Slovo. Většinou jsme ji hráli ve dvou, ale bylo možné ji hrát i ve více lidech. Každý hráč potřeboval papír a tužku a také bylo potřeba na něčem měřit čas.

Hra začala tím, že se určilo libovolné delší slovo, které si hráči napsali na svůj papír. V časovém limitu se pak z písmen tohoto slova snažili vymyslet co nejvíce dalších slov. Čím originálnější a delší slovo, tím lépe. Po uplynutí limitu si hráči navzájem ukázali svá vymyšlená slova o obodovali si ji. Za každé písmeno byl jeden bod. Pokud ovšem slovo druhý hráč neměl, mělo písmeno dvojnásobnou hodnotu.

Často jsem tuto hru hrál s manželkou na cestách, ale nebylo vždy snadné sehnat papír a dvě tužky, ani si na telefonu nastavovat pokaždé stopky. Proto se zrodil nápad vytvořit aplikaci, která by tuto hru implementovala a všechny výše zmíněné problémy vyřešila.

Cíl práce

Cílem mé práce je vytvořit aplikaci na mobilní zařízení s operačním systémem Android pro dva hráče, která by implementovala hru na papíře popsanou v předchozí sekci.

Prvním dílčím cílem je kompletně specifikovat pravidla hry, včetně systému bodování pro více kol hry.

Druhý dílčí cíl mé práce je analyzovat možné způsoby přímého propojení dvou mobilních zařízení, vytvořit prototypy těchto způsobů propojení a z těchto způsobů vybrat nejvhodnější pro budoucí hru.

Třetím dílčím cílem je navrhnout, implementovat a otestovat mobilní aplikaci pro Android, která bude využívat nalezený nejvhodnější způsob propojení a implementovat pravidla hry.

Popis kapitol

Práce je rozdělena do čtyř hlavních kapitol. První kapitola má za cíl specifikovat hru. Definuji zde pravidla hry, která vycházejí z původní hry na papíře, a upřesňují, jak se bude hra chovat na mobilních zařízení. Detailně popisuji průběh celé hry s různými variantami. Na konci kapitoly připojuji také nefunkční požadavky na aplikaci, které by měly být součástí specifikace.

V nefunkčních požadavcích není specifikováno, jaká technologie má být použita pro komunikaci mezi zařízeními. Tomu se věnuji ve druhé kapitole, kde provádím analýzu způsobů přímého propojení dvou mobilních zařízení. Uvádím zde seznam technologií a dvě nejvhodnější rozpracovávám do větších detailů. Nejprve technologie krátce představím, a poté podrobně popíši, jak provést propojení dvou zařízení včetně přenosu dat.

Následně se zabývám vytvořením prototypů těchto dvou technologií, abych mohl lépe určit, jaká bude pro mou aplikaci vhodnější. Popisuji návrh prototypu, který bude implementovat připojení zařízení a jednoduchý přenos dat. Dále detailněji rozpracovávám prototypy jednotlivých technologií.

Na závěr kapitoly provádím porovnání technologií pomocí vytvořených prototypů a vybírám tu vhodnější, kterou použiji při implementaci hry.

Ve třetí kapitole se zabývám návrhem a implementací samotné hry. Začínám návrhem jednotlivých obrazovek a uspořádání prvků na nich pomocí jednoduchých papírových wireframes. Dále již popisuji návrh tříd a jejich implementaci. Základní část aplikace zajišťující přenos dat mezi zařízeními byla převzata z prototypu. Uvádím tedy nové části aplikace.

V poslední části této kapitoly se zabývám grafickou podobou hry. Zmiňuji i některé implementační detaily související se vzhledem aplikace.

Čtvrtá kapitola se věnuje testování hry. Následně popisuji opravu všech nalezených problémů i s implementačními detaily.

Kapitola 1

Specifikace hry

V této kapitole se zabývám specifikací pravidel hry – funkčních požadavků a nefunkčních požadavků, které určují omezení kladená na systém [1].

1.1 Pravidla hry

Před samotným návrhem a implementací aplikace je potřeba specifikovat pravidla hry a definovat tak funkční požadavky na aplikaci. Pravidla vycházejí z původní hry na papíře a upřesňují chování na mobilním zařízení. V následujících sekcích popíši, jak by měla hra probíhat.

1.1.1 Spuštění hry

Hráči se dohodnou, který z nich hru založí (bude zadávat parametry hry a spouštět kola). Tento hráč pak zvolí možnost *Založit hru* a druhý hráč se k němu připojí.

1.1.2 Průběh kola

Zakladatel hry vybere slovo (vlastní nebo náhodné z připraveného seznamu) a nastaví parametry kola. Tyto parametry budou zároveň výchozí pro následující kolo. Parametry kola jsou následující:

- slovo,
- časový limit kola,
- způsob kontroly správnosti vytvořených slov:
  - kontrola protihráčem až při vyhodnocení,
  - podle slovníku,
  - bez kontroly,

- minimální délka vytvořených slov.

Po nastavení parametrů zakladatel hry odstartuje první kolo. Hráčům se ukáže obrazovka se slovem a odpočítáváním. Hráči klikáním na písmena ze slova vytvářejí co nejvíce jiných slov.

Po kliknutí na písmeno se stane neaktivní (každé písmeno lze tudíž použít jen jednou) a připíše se k právě vytvářenému novému slovu. Dále bude na obrazovce tlačítko smazaní posledního písmene a potvrzení nově vytvořeného slova. Toto tlačítko přidá slovo na seznam již napsaných slov, pokud vyhovuje nastavené kontrole. V opačném případě aplikace zobrazí upozornění a slovo nepřidá. Po úspěšném potvrzení slova je možné začít vytvářet další slovo.

Hráč se snaží vytvořit co nejvíce slov, která jsou co nejdelší. Výhodou je, když se soupeři stejné slovo vymyslet nepodaří. Po vypršení časového limitu dojde k vyhodnocení kola.

1.1.3 Vyhodnocení kola

Pokud byla zvolena kontrola slov protihráčem, zobrazí se každému hráči seznam soupeřových slov. Před hrou se hráči dohodnou, jaká slova budou uznávat. Doporučeno je se řídit stejnými pravidly jako u hry Scrabble, viz dále. Hráč bude mít možnost tlačítkem u každého slova slovo vymazat. (Hra předpokládá, že jsou hráči u sebe a mohou spolu komunikovat. Typicky se jednomu hráči nelíbí nějaké slovo a řekne to druhému. Ten mu vysvětlí, co slovo znamená a po domluvě mu ho první hráč buď uzná, nebo ne a slovo smaže.)

V případě kontroly podle slovníku dojde k odstranění všech slov, která se ve slovníku nenacházejí. Slovník by se měl řídit stejnými pravidly jako pro hru Scrabble – varianta klasik [2]. Měl by obsahovat všechna česká slova kromě následujících:

- podstatná jména, přídavná jména, zájmena a číslovky v jiných pádech než v prvním,
- vlastní jména,
- citoslovce,
- přechodníky,
- zkratky a značky (např. km),
- slova zastaralá, zřídka používaná, nářeční, vulgární a argotická.

Poté dojde k vyhodnocení. Aplikace zobrazí slova hráče i protihráče s jejich bodovým ohodnocením. Také zobrazí celkové součty obou hráčů a označí vítěze.

1.1.3.1 Způsob bodování

Každé slovo je ohodnoceno podle své délky. Pokud má stejné slovo i soupeř, je ohodnoceno jedním bodem za každé písmeno. Když soupeř slovo nemá, je hodnota jednoho písmene slova dva body.

1.1.4 Další kola

Poté, co si hráči prohlédnou vyhodnocení předchozího kola, bude moci zakladatel hry spustit nové kolo. Opět vybere slovo, případně změní parametry kola. Takto probíhají jednotlivá kola za sebou, dokud hra hráče baví. Výsledky jednotlivých kol se navíc sčítají a hráči mohou průběžné skóre sledovat po celou dobu hry.

1.1.5 Ukončení hry

Hru je možné ukončit po skončení kola z obrazovky s výsledky, a to hráčem, který hru založil. Po ukončení se hráčům zobrazí úvodní obrazovka, ze které je možné s kýmkoliv začít hru novou.

1.2 Funkční požadavky

Pravidla hry jsou shrnuta do těchto funkčních požadavků:

F1 Spuštění hry Uživatel bude moci spustit hru jako zakladatel a zadávat parametry kola, nebo se bude moci ke hře připojit. Aplikace zajistí propojení zařízení.

F2 Parametry kola Kolo hry bude mít následující parametry:

- slovo,
- časový limit kola,
- způsob kontroly správnosti vytvořených slov:
  - kontrola protihráčem až při vyhodnocení,
  - podle slovníku,
  - bez kontroly,
- minimální délka vytvořených slov.

Zakladatel hry bude moci před každým kolem tyto parametry nastavit. Slovo bude možné napsat vlastní nebo vybrat náhodně z připraveného seznamu.

F3 Kolo hry Uživatel bude moci vytvářet slova z písmen vybraného slova (každé písmeno lze použít jen jednou). Při psaní bude moci k vytvářenému slovu přidávat písmena nebo je mazat. Také bude možné smazat celé vytvářené slovo. Po potvrzení uvidí uživatel nově přidané slovo v seznamu již vytvořených slov. Uživateli se bude také zobrazovat počet zbývajících sekund do konce kola.

F4 Kontrola protihráčových slov Při zvolení typu kontroly slov protihráčem umožní aplikace uživateli si prohlédnout soupeřova slova a libovolná z nich vyškrtnout.

F5 Vyhodnocení kola Aplikace zobrazí uživateli výsledky kola na základě porovnání slov hráčů a typu kontroly slov. Při kontrole protihráčem se hodnotí pouze slova, která soupeř nevyškrtl, při kontrole slovníkem se hodnotí pouze slova v něm obsažená. Výsledky budou obsahovat:

- informaci, zda hráč vyhrál, či prohrál,
- skóre a celkové skóre,
- skóre a celkové skóre protihráče,
- vytvořená slova s jejich bodovým ohodnocením a zvýrazněním, pokud bylo slovo za dvojnásobek bodů,
- protihráčova slova s jejich bodovým ohodnocením a zvýrazněním, pokud bylo slovo za dvojnásobek bodů.

Skóre je spočítáno na základě délky vytvořených slov. Pokud má soupeř stejné slovo, je hodnota jednoho písmene jeden bod, jinak dva body.

F6 Začátek nového kola a ukončení hry Uživatel, který hru založil, bude moci spustit nové kolo, nebo hru ukončit.

1.3 Nefunkční požadavky

NF1 Od aplikace je vyžadováno, aby fungovala bez internetového připojení a komunikace mezi zařízeními probíhala přímo, bez potřeby třetího subjektu (například serveru). Zařízení od sebe nebudou vzdálená více jak 10 metrů.

NF2 Druhým nefunkčním požadavkem je, aby byla hra implementovaná pro mobilní operační systém Android.

Kapitola 2

Analýza způsobů přímého propojení dvou mobilních zařízení

2.1 Možné technologie

Pro propojení dvou telefonů bez použití serveru bylo potřeba provést analýzu možných technologií, které toto umožňují. Podmínkou je, že řešení musí podporovat přenos málo častých krátkých zpráv mezi dvěma telefony bez potřeby připojení k internetu, či k jinému dalšímu zařízení. V úvahu připadají následující tři technologie:

- Bluetooth,
- Wi-Fi,
- NFC.

Technologie NFC – Near Field Communication vyžaduje pro spojení dvou zařízení vzdálenost nejvýše 4 centimetry [3]. To je vhodné například pro jednorázový přenos dat, ne však pro dlouhodobější komunikaci, která je pro tuto hru potřeba. Toto řešení je tedy možné vyloučit.

Další dvě technologie nyní rozeberu podrobněji.

2.2 Bluetooth

2.2.1 Úvod

Bluetooth je nízkonákladová, výkonově nenáročná rádiová technologie na krátký dosah, původně vyvinutá jako náhrada za propojovací kabely mezi mobilním

zařízením a příslušenstvím, jako jsou například sluchátka. Firma Ericsson Mobile Communications začala s vývojem této technologie v roce 1994. [4]

V únoru 1998 vznikla skupina Bluetooth Special Interest Group (SIG) složená z několika společností za účelem definovat specifikaci Bluetooth. Mezi zakladatelské společnosti patřily: Ericsson Mobile Communications AB., Intel Corp., IBM Corp., Toshiba Corp. a Nokia Mobile Phones. První verze Bluetooth 1.0 vyšla následně v roce 1999. [4] Ke skupině se připojují další a další společnosti a nyní má již přes 30 000 členů. [5]

2.2.2 Zajímavost: Původ názvu

Technologie je pojmenována podle Haralda Blatanda (Blatand znamená Modrozub, anglicky Bluetooth). Harald byl dánský vikingský král vládnoucí v 10. století, kterému se podařilo ovládnout a sjednotit valčící kmeny v Dánsku a Norsku. Stejně tak má technologie Bluetooth sjednotit telekomunikační a výpočetní průmysl. [4]

2.2.3 Android

Operační systém Android technologii Bluetooth samozřejmě podporuje a nabízí její funkcionalitu prostřednictvím Android Bluetooth APIs. Aplikační rozhraní nabízí možnost:

- skenovat okolní zařízení s Bluetooth,
- dotázat se lokálního adaptéru Bluetooh na již spárovaná zařízení,
- zřídit RFCOMM kanál,
- připojit se k jinému zařízení,
- přenášet data z jiného zařízení a na něj,
- spravovat více připojení. [6]

Bluetooth nabízí dva režimy, které operační systém Android podporuje: Classic Bluetooth a Bluetooth Low Energy. Bluetooth Low Energy je dostupný od verze Android 4.3 (API level 18) a je určený převážně pro připojení jiných zařízení k telefonu. [6]. Pro svoji práci, kde je potřeba zajistit komunikaci mezi dvěma telefony, tedy využiji Classic Bluetooth.

2.2.4 Poznámka: Názvosloví

Pro zjednodušení popisu a pohodlnější čtení budu používat pro některé prvky (třídy) systému Android používané počeštělé názvy s malými písmeny a i v pádech. Jedná se o slova:

- aktivita – potomek třídy Activity respektive AppCompatActivity představující jednu obrazovku aplikace,
- fragment – potomek třídy Fragment,
- broadcast receiver – potomek třídy BroadcastReceiver,
- listener – potomek třídy typu listener,
- handler – potomek třídy Handler,
- dialog – dialogové okno.

2.2.5 Postup navázání komunikace mezi dvěma zařízeními na OS Android

[6] popisuje, co vše je potřeba pro navázání komunikace mezi dvěma zařízeními a jak toho docílit.

Aby aplikace mohla přistupovat ke všem potřebným funkcím Bluetooth, musí požádat o těchto několik povolení:

- android.permission.BLUETOOTH,
- android.permission.BLUETOOTH_ADMIN,
- android.permission.ACCESS_COARSE_LOCATION.

Povolení je potřeba vložit do souboru Android manifest.

2.2.5.1 Nastavení Bluetooth

Předtím, než bude aplikace komunikovat přes Bluetooth, je nutné ověřit, zda je Bluetooth na zařízení podporován, a pokud ano, je potřeba se ujistit, jestli je povolen.

Pokud není Bluetooth podporován, měly by se všechny funkce využívající Bluetooth deaktivovat. V mé práci je však komunikace mezi zařízeními stěžejní, takže pokud zařízení Bluetooth nepodporuje, dojde k oznámení uživateli a aplikace se ukončí.

Že není Bluetooth podporován se zjistí tak, že metoda BluetoothAdapter.getDefaultAdapter() vrátí null.

```
BluetoothAdapter mBluetoothAdapter =
    BluetoothAdapter.getDefaultAdapter();
if (mBluetoothAdapter == null) {
    // Device doesn't support Bluetooth
}
```

Pokud Bluetooth podporován je, ale není zapnutý, je možné požádat uživatele o jeho zapnutí bez nutnosti opustit aplikaci. Uživateli se zobrazí dialogové okno s možností Bluetooth povolit.

Zobrazení dialogového okna se provede následovně:

```
if (!mBluetoothAdapter.isEnabled()) {
    Intent enableBtIntent = new
        Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
}
```

Odpověď od uživatele je zaslána do metody onActivityResult() s parametrem resultCode o hodnotě RESULT_OK nebo RESULT_CANCELED.

Alternativně lze odchytávat zprávu (Intent) ACTION_STATE_CHANGED pomocí třídy BroadcastReceiver. Zpráva obsahuje parametry EXTRA_STATE a EXTRA_PREVIOUS_STATE, které obsahují nový a starý stav Bluetooth.

2.2.5.2 Hledání okolních zařízení

Zjistit informace o vzdálených zařízeních potřebné k navázání spojení je možné dvěma způsoby, a to pomocí třídy BluetoothAdapter. Ta umožňuje vyhledat zařízení v okolí nebo na vyžádání poskytne seznam již spárovaných zařízení.

Spárovaná zařízení lze získat pomocí metody getBondedDevices().

```
Set<BluetoothDevice> pairedDevices =
    mBluetoothAdapter.getBondedDevices();

if (pairedDevices.size() > 0) {
    // There are paired devices. Get the name and address of each
    // paired device.
    for (BluetoothDevice device : pairedDevices) {
        String deviceName = device.getName();
        // MAC address
        String deviceHardwareAddress = device.getAddress();
    }
}
```

Hledání vzdálených zařízení (anglicky discovering) je proces, který vyhledává zařízení v blízkém okolí podporující Bluetooth a získává od nich informace, jako je jejich jméno, třída a MAC adresa. Na toto vyhledávání však odpovídají pouze ta zařízení, která jsou viditelná (anglicky discoverable).

Pro zahájení vyhledávání slouží metoda startDiscovery(). Proces vyhledávání je asynchronní. Metoda vrátí pouze informaci, zda vyhledávání započalo v pořádku, či nikoliv. Samotný proces trvá kolem 12 sekund a výsledky vyhledávání posílá systém pomocí zpráv (třída Intent) typu ACTION_FOUND. Ty je potřeba odchytit pomocí objektu třídy BroadcastReceiver, který bude zaregistrován právě na tuto akci.

Zde je ukázka kódu s vytvořením a registrací broadcast receiveru:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    ...
    // Register for broadcasts when a device is discovered.
    IntentFilter filter = new
        IntentFilter(BluetoothDevice.ACTION_FOUND);
    registerReceiver(mReceiver, filter);
}

// Create a BroadcastReceiver for ACTION_FOUND.
private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        if (BluetoothDevice.ACTION_FOUND.equals(action)) {
            // Discovery has found a device. Get the BluetoothDevice
            // object and its info from the Intent.
            BluetoothDevice device =
                intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
            String deviceName = device.getName();
            // MAC address
            String deviceHardwareAddress = device.getAddress();
        }
    }
};

@Override
protected void onDestroy() {
    super.onDestroy();
    ...
    // Don't forget to unregister the ACTION_FOUND receiver.
    unregisterReceiver(mReceiver);
}
```

Vyhledávání okolních zařízení spotřebovává mnoho zdrojů adaptéru Bluetooth a také výrazně snižuje šířku pásma dostupného pro všechna existující připojení. Je tedy vhodné po nalezení zařízení před zahájením připojení ukončit vyhledávání pomocí metody cancelDiscovery().

Jakmile iniciátor spojení zná MAC adresu druhého zařízení, může se k němu připojit. Pokud se k tomuto zařízení připojuje poprvé, ukáže se automaticky na obou zařízeních žádost o párování. V okamžiku, kdy je žádost potvrzena, dojde k uložení informací o druhém zařízení (jméno zařízení, třída a MAC adresa). Tyto informace je možné přes Bluetooth API kdykoliv vyžádat. Při opětovném připojení ke stejnému zařízení je tedy již MAC adresa známá a není potřeba znovu provádět vyhledávání.

Je tudíž rozdíl mezi tím, když jsou zařízení spárovaná a když jsou připojená. Spárovaná zařízení vědí navzájem o své existenci, mají sdílený klíč,

který je použit pro autentizaci, a jsou schopna navázat šifrované spojení s druhým zařízením. Připojená zařízení spolu aktuálně sdílí RFCOMM kanál a jsou schopna si přes něj navzájem posílat data. Systém Android vyžaduje, aby před vytvořením komunikačního kanálu byla zařízení spárována, proto při prvním připojení zobrazí žádost o párování.

Je nutno dodat, že zařízení Android není defaultně viditelné. Uživatel může v systémovém nastavení na určitý čas zařízení zviditelnit, nebo je možné uživatele požádat o zviditelnění zařízení přímo v aplikaci. Pro vyvolání dialogového okna je potřeba zavolat metodu startActivityForResult(Intent, int duration) s objektem třídy Intent typu ACTION_REQUEST_DISCOVERABLE. Zda uživatel viditelnost povolil, se zjistí v metodě onActivityResult() pomocí parametru resultCode. Ten bude v případě povolení obsahovat nastavenou dobu viditelnosti, v opačném případě hodnotu konstanty RESULT_CANCELED. Pokud nebyl Bluetooth na zařízení zapnutý, při zviditelnění zařízení dojde k jeho zapnutí automaticky.

2.2.5.3 Připojení se k zařízení

Pro vytvoření spojení mezi dvěma zařízeními je potřeba implementovat mechanizmus představující jak stranu serveru, tak stranu klienta. Pro každou stranu se vytvoří vlastní vlákno, pro server na čekání na klienta, pro klienta na připojení k serveru. Po navázání spojení je již pro obě strany část kódu implementující komunikaci stejná, tudíž je pro ni možné použít třetí společné vlákno.

Na straně serveru je nutné získat instanci třídy BluetoothServerSocket voláním metody listenUsingRfcommWithServiceRecord(). Tato metoda má dva parametry, jméno a UUID – Universally Unique Identifier.

Jméno identifikuje službu aplikace, kterou systém automaticky zapíše jako novou položku do databáze SDP (Service Discovery Protocol). Nejčastěji se používá přímo jméno aplikace.

UUID je také součástí položky SDP a je důležité pro připojení s klientským zařízením. Jednoznačně identifikuje službu, ke které se klient chce připojit. Pokud se UUID nebude shodovat, k připojení nedojde.

UUID má standardizovaný 128bitový formát, který je dostatečně velký na to, aby jeho náhodné zvolení nekolidovalo s jiným identifikátorem. UUID je doporučeno vygenerovat libovolným generátorem dostupným na internetu a inicializovat ho pomocí metody UUID.fromString(String).

Poté, co server získá objekt třídy BluetoothServerSocket, zahájí čekání na klienta metodou accept(). Jedná se o blokující volání, proto je tato část serveru prováděna ve vlastním vlákně, jak bylo popsáno výše. Pokud se klient pomocí stejného UUID připojí, vrátí metoda objekt třídy BluetoothSocket.

Po navázání spojení je možné soket serveru ukončit metodou close(). Komunikace bude probíhat pomocí nově získaného soketu. Ten je vhodné předat

do nového vlákna, které bude zajišťovat samotnou komunikaci mezi zařízeními. Implementace tohoto vlákna může být již společná pro server i pro klienta.

Ze strany klienta je potřeba navázat spojení se serverem. K tomu je zapotřebí mít instanci třídy BluetoothDevice reprezentující zařízení serveru. Tento objekt je možné získat přímo postupem popsaném v předchozí kapitole 2.2.5.2 nebo postačí tímto postupem získat pouze MAC adresu zařízení. Objekt lze pak získat metodou getRemoteDevice(address) volanou na instanci třídy BluetoothAdapter.

Na tomto objektu je následně potřeba zavolat metodu createRfcommSocketToServiceRecord(UUID), a získat tak instanci třídy BluetoothSocket. UUID musí být stejné jako při vytváření soketu serveru. Nejlepší je ho mít přímo v kódu aplikace, odkud si ho vezmou obě strany.

Pro připojení k serveru složí metoda connect() volána na získaném objektu třídy BluetoothSocket. Tato metoda je blokující, dokud nedojde k připojení k serveru (nebo nenastane výjimka třídy IOException). Po úspěšném připojení je možné předat soket, stejně jako v případě serveru, dalšímu vláknu, které se bude starat o komunikaci.

2.2.5.4 Přenos dat

Tato část je již stejná pro server i pro klienta. V novém komunikačním vlákně se z předaného soketu vytvoří dva proudy (objekty tříd InputStream a OutputStream metodami socket.getInputStream() a socket.getOutputStream()). Do vstupního proudu bude možné data zapisovat a ze vstupního data číst.

Čtení je realizováno v metodě run(), kde je ve smyčce volána metoda inputStream.read(byte[]). Tato metoda je blokující a čeká, dokud druhé zařízení nepošle data. Pokud dojde k přerušení spojení, je vyhozena výjimka.

Zápis se provádí pomocí veřejné metody vlákna, která volá metodu outputStream.write(byte[]).

2.3 Wi-Fi

2.3.1 Úvod

[7] popisuje Wi-Fi takto: Velmi stručně lze říct, že Wi-Fi je způsob komunikace mezi bezdrátovými zařízeními. U zařízení Wi-Fi je zaručena spolupráce a běh v nějakém druhu standardu 802.11 bezdrátových sítí středního dosahu. Wi-Fi, což je zkratka *wireless fidelity* (bezdrátová věrnost), je vlastně certifikovaná verze variant bezdrátových standardů 802.11 vyvinutý organizací IEEE (`www.ieee.org`). Vybavení Wi-Fi certifikuje Wi-Fi Alliance (`www.wi-fi.org`) z hlediska kompatibility a naplňování standardů.

Na rozdíl od řady jiných bezdrátových standardů běží 802.11 na „volné“ části rádiového spektra. To znamená, že (např. oproti mobilním telefonům) pro vysílání a komunikaci pomocí 802.11 (neboli Wi-Fi) není zapotřebí žádná

licence. Bezdrátové standardy 802.11 jsou navrženy k využívání volných spekter, která nevyžadují specifické licencování. Spektry v současné době využívanými standardy 802.11 jsou 2,4 GHz a 5 GHz. Jak možná víte, tato volná rádiová spektra využívá i řada jiných domácích zařízení, jako např. mikrovlnné trouby a (především) bezdrátové domácí telefony.

Standard 802.11 je důležitý, protože jej využívají miliony lidí po celém světě. Používají ho pro připojení k internetu bez kabelů, když jsou na cestách a mimo svůj domov nebo kancelář. Standard 802.11 slouží rovněž k vytváření bezdrátových sítí v domácnostech i na pracovištích.

I když je technologie Wi-Fi teprve několik let, je důležitá, jelikož zaručuje vzájemnou kompatibilitu zařízení 802.11. Každé zařízení Wi-Fi dokáže „mluvit“ s každým jiným zařízením Wi-Fi – a organizace Wi-Fi Alliance certifikuje, že to tak skutečně je!

2.3.2 Android

Operační systém Android samozřejmě Wi-Fi podporuje a nabízí možnost připojení k Wi-Fi sítím. To ale pro tuto aplikaci není potřeba. Mobilní zařízení mají komunikovat na přímo a k tomu OS Android nabízí technologii Wi-Fi peer-to-peer (P2P).

Tato technologie je podporována od verze 4.0 (API level 14). Nabízí možnost komunikace mezi dvěma zařízeními podporujícími tuto technologii bez potřeby přístupového bodu. Framework Android Wi-Fi P2P vyhovuje certifikačnímu programu Wi-Fi Direct^{TM} společnosti Wi-Fi Alliance. [8]

Aplikační rozhraní Wi-Fi P2P nabízí následující:

- pomocí metod třídy WifiP2pManager objevovat další zařízení (anglicky peers), vyžádat si jejich seznam a připojit se k nim,
- pomocí tříd posluchačů (anglicky listeners) se nechat informovat o úspěchu nebo neúspěchu metod třídy WifiP2pManager. Když jsou volány metody této třídy, může každá metoda obdržet jako parametr specifický listener,
- pomocí speciálních tříd Intents se nechat informovat o specifických událostech detekovaných frameworkem Wi-Fi P2P, jako je ztracení spojení nebo objevení nového zařízení. [8]

2.3.3 Postup navázání komunikace přes Wi-Fi P2P

[8] uvádí také postup, jak navázat komunikaci přes Wi-Fi P2P.

2.3.3.1 Vytvoření třídy broadcast receiver

Jak už bylo popsáno výše, systém Android nabízí zasílání informací pomocí objektů třídy Intent. Aby bylo možné tyto informace získávat, je potřeba vy-

tvořit třídu dědící od třídy BroadcastReceiver. Do konstruktoru této třídy je vhodné předat parametry, jimiž jsou instance tříd WifiP2pManager, WifiP2pManager.Channel a Activity, ve které byl receiver zaregistrován. To umožní komunikaci s aktivitou a také, pokud je to potřeba, přístup k Wi-Fi hardwaru a komunikačnímu kanálu.

Receiver by měl být schopen přijímat následující čtyři typy akcí (událostí):

- WIFI_P2P_STATE_CHANGED_ACTION,
- WIFI_P2P_PEERS_CHANGED_ACTION,
- WIFI_P2P_CONNECTION_CHANGED_ACTION,
- WIFI_P2P_THIS_DEVICE_CHANGED_ACTION.

Událost WIFI_P2P_STATE_CHANGED_ACTION je zaslána, pokud dojde k povolení nebo zakázání Wi-Fi P2P.

Po zavolání metody discoverPeers() je při jakékoliv změně seznamu viditelných zařízení zaslaná událost WIFI_P2P_PEERS_CHANGED_ACTION. Po přijmutí této zprávy je zpravidla volána metoda requestPeers() pro získání seznamu zařízení.

Zaslání události WIFI_P2P_CONNECTION_CHANGED_ACTION značí změnu stavu připojení Wi-Fi na zařízení, například připojení k jinému zařízení nebo odpojení od něj.

Poslední událost WIFI_P2P_THIS_DEVICE_CHANGED_ACTION je zaslána, pokud došlo ke změně informací o zařízení, jako je například jméno zařízení.

2.3.3.2 Počáteční nastavení

Aby mohla aplikace komunikovat s dalším zařízením přes Wi-Fi, musí požádat o těchto několik povolení:

- android.permission.ACCESS_WIFI_STATE,
- android.permission.CHANGE_WIFI_STATE,
- android.permission.CHANGE_NETWORK_STATE,
- android.permission.INTERNET,
- android.permission.ACCESS_NETWORK_STATE.

Povolení je potřeba vložit do souboru Android manifest, stejně jako minimální SDK verzi – 14.

Poté je potřeba zkontrolovat, zda je na zařízení Wi-Fi P2P povoleno. To se provede v již vytvořeném broadcast receiveru v reakci na událost WIFI_P2P-_STATE_CHANGED_ACTION. Intent obsahuje parametr EXTRA_WIFI-_STATE, který může nabývat hodnot WIFI_P2P_STATE_ENABLED, nebo WIFI_P2P_STATE_DISABLED.

```
@Override
public void onReceive(Context context, Intent intent) {
    ...
    String action = intent.getAction();
    if (WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION.equals(action)) {
        int state =
            intent.getIntExtra(WifiP2pManager.EXTRA_WIFI_STATE, -1);
        if (state == WifiP2pManager.WIFI_P2P_STATE_ENABLED) {
            // Wifi P2P is enabled
        } else {
            // Wi-Fi P2P is not enabled
        }
    }
    ...
}
```

V metodě onCreate() v aktivitě je potřeba získat instanci třídy WifiP2p-Manager a registrovat aplikaci, aby mohla používat framework Wi-Fi P2P. To se provede metodou initialize(). Tato metoda vrací kanál – instanci třídy WifiP2pManager.Channel, který je použit k připojení aplikace k frameworku Wi-Fi P2P. Dále se vytvoří instance již vytvořeného broadcast receiveru a předají se mu potřebné parametry, jak již bylo zmíněno výše.

```
WifiP2pManager mManager;
Channel mChannel;
BroadcastReceiver mReceiver;
...
@Override
protected void onCreate(Bundle savedInstanceState){
    ...
    mManager = (WifiP2pManager)
        getSystemService(Context.WIFI_P2P_SERVICE);
    mChannel = mManager.initialize(this, getMainLooper(), null);
    mReceiver = new WiFiDirectBroadcastReceiver(mManager, mChannel,
        this);
    ...
}
```

Poté je potřeba broadcast receiver registrovat a nastavit mu pomocí třídy IntentFilter, při jakých událostech se nechá informovat. Jedná se o čtyři akce zmíněné v kapitole 2.3.3.1.

```
IntentFilter mIntentFilter;
```

```
...
@Override
protected void onCreate(Bundle savedInstanceState){
    ...
    mIntentFilter = new IntentFilter();
    mIntentFilter.addAction(
      WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION);
    mIntentFilter.addAction(
      WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION);
    mIntentFilter.addAction(
      WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION);
    mIntentFilter.addAction(
      WifiP2pManager.WIFI_P2P_THIS_DEVICE_CHANGED_ACTION);
    ...
}
```

Registraci receiveru je vhodné provést v metodě onResume() a zrušit ji v metodě onPause().

```
/* register the broadcast receiver with the intent values to be
   matched */
@Override
protected void onResume() {
    super.onResume();
    registerReceiver(mReceiver, mIntentFilter);
}
/* unregister the broadcast receiver */
@Override
protected void onPause() {
    super.onPause();
    unregisterReceiver(mReceiver);
}
```

2.3.3.3 Hledání okolních zařízení

Hledání okolních zařízení se zahájí voláním metody discoverPeers(). Volání této funkce je asynchronní, její výsledek lze získat pomocí instance třídy WifiP2pManager.ActionListener, která se metodě předá jako parametr. Listener má dvě metody onSuccess() a onFailure(), které jsou volány ve chvíli, kdy je znám výsledek metody discoverPeers(). Metoda onSuccess() pouze informuje, že proces hledání zařízení v pořádku započal, neposkytuje žádné informace o nalezených zařízení.

```
mManager.discoverPeers(channel, new WifiP2pManager.ActionListener() {
    @Override
    public void onSuccess() {
        ...
    }
```

```
    @Override
    public void onFailure(int reasonCode) {
        ...
    }
});
```

Ve chvíli, kdy je hledání úspěšné a byla nalezena nějaká zřízení, systém vyšle zprávu WIFI_P2P_PEERS_CHANGED_ACTION, která je odchycena v broadcast receiveru. Pro získání seznamu dostupných zařízení je potřeba zavolat metodu requestPeers(). Tato metoda je opět asynchronní a seznam zařízení je předán instanci třídy WifiP2pManager.PeerListListener v metodě onPeersAvailable(WifiP2pDeviceList peers).

2.3.3.4 Připojení se k zařízení

Pro připojení se k zařízení je potřeba znát pouze jeho MAC adresu. Seznam zařízení je reprezentován seznamem instancí třídy WifiP2pDevice, která má veřejnou členskou proměnnou deviceAddress obsahující MAC adresu daného zařízení. Jméno zařízení je uložené v proměnné deviceName.

MAC adresa zařízení se uloží do objektu třídy WifiP2pConfig. Tento objekt je pak parametrem metody connect(), která slouží pro samotné připojení se k zařízení. Tato metoda je opět asynchronní, takže jejím druhým parametrem je objekt třídy ActionListener(), který má, jak už bylo zmíněno, dvě metody onSuccess() a onFailure(). Metoda onSuccess() opět informuje pouze o úspěšném odeslání požadavku.

```
//obtain a peer from the WifiP2pDeviceList
WifiP2pDevice device;
WifiP2pConfig config = new WifiP2pConfig();
config.deviceAddress = device.deviceAddress;
mManager.connect(mChannel, config, new ActionListener() {

    @Override
    public void onSuccess() {
        // WiFiBroadcastReceiver will notify us. Ignore it here.
    }

    @Override
    public void onFailure(int reason) {
        // failure logic
    }
});
```

O tom, zda došlo skutečně ke spojení, informuje systém odesláním události WIFI_P2P_CONNECTION_CHANGED_ACTION. Tato událost je odchycena v receiveru, jen je potřeba zkontrolovat, zda bylo spojení skutečně navázáno. Zaslaná událost (intent) obsahuje objekt třídy NetworkInfo, který má

metodu isConnected(). Metodou requestConnectionInfo() se získají potřebné informace pro navázání komunikačního spojení.

```
    ...
    } else if (WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION
                .equals(action)) {
        ...
        NetworkInfo networkInfo = (NetworkInfo) intent
                .getParcelableExtra(WifiP2pManager.EXTRA_NETWORK_INFO);

        if (networkInfo.isConnected()) {
            // We are connected with the other device, request
            // connection info to find group owner IP
            mManager.requestConnectionInfo(mChannel,
                connectionListener);
        }
        ...
```

Tato metoda je opět asynchronní. Výsledek je předán do metody onConnectionInfoAvailable(WifiP2pInfo) třídy WifiP2pManager.ConnectionInfoListener, kterou vyžaduje metoda requestConnectionInfo() jako druhý parametr.

```
@Override
public void onConnectionInfoAvailable(final WifiP2pInfo info) {

    // InetAddress from WifiP2pInfo struct.
    InetAddress groupOwnerAddress =
        info.groupOwnerAddress.getHostAddress();

    // After the group negotiation, we can determine the group owner
    // (server).
    if (info.groupFormed && info.isGroupOwner) {
        // Do whatever tasks are specific to the group owner.
        // One common case is creating a group owner thread and
        // accepting incoming connections.
    } else if (info.groupFormed) {
        // The other device acts as the peer (client). In this case,
        // you'll want to create a peer thread that connects
        // to the group owner.
    }
}
```

If-větev v kódu výše je místo pro spuštění části kódu serveru, else-if-větev pro části kódu klienta. IP adresu serveru je možné získat zavoláním metody info.groupOwnerAddress.getHostAddress().

2.3.3.5 Přenos dat

Po vytvoření komunikačního spojení je možné přenášet pomocí soketu data mezi zařízeními. K tomu jsou potřeba následující kroky, podobné jako při připojení přes Bluetooth:

1. Vytvořit objekt třídy ServerSocket. Tento soket čeká na připojení od klienta na specifickém portu. Čekání je blokující, takže je potřeba tuto činnost dělat ve vláknu na pozadí.

2. Z pozice klienta vytvořit objekt třídy Socket. Klient pro připojení k serveru použije IP adresu a port server soketu.

3. Server soket čeká v metodě accept() na připojení od klienta. Jakmile dojde k připojení, je možné pro server i pro klienta zapisovat do soketu data a zaslaná data ze soketu číst. (Podrobněji popsáno v kapitole 2.2.5.4 věnující se propojení přes Bluetooth.)

2.4 Vytvoření prototypů

Jak bylo zmíněno na začátku kapitoly 2, je pro implementaci aplikace možné použít dva způsoby propojení dvou mobilních zařízení – Bluetooth a Wi-Fi. Pro vybraní nejlepšího způsobu jsem se rozhodl vytvořit dva funkční prototypy, na kterých jednotlivé technologie otestuji. Na základě tohoto porovnání vyberu vhodnější technologii a použijí při implementaci hry.

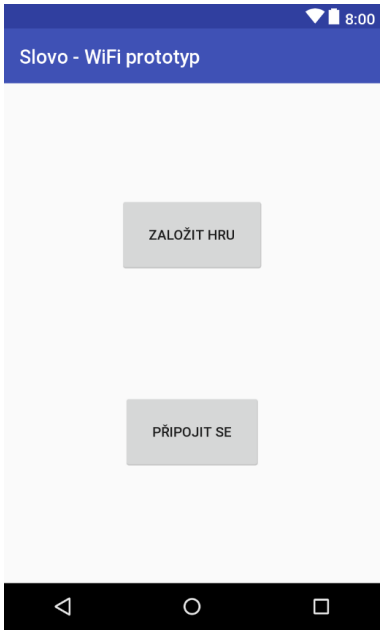
2.4.1 Návrh prototypu

Prototyp jsem navrhl tak, aby byl co nejjednodušší, ale zároveň vhodný jako základ pro implementaci hry. Obsahuje všechny obrazovky, které budou ve hře potřeba a implementuje potřebnou komunikaci mezi zařízeními.

Pro prototyp jsem zvolil následující velmi jednoduchou hru. V daném časovém limitu se hráči snaží co nejvícekrát kliknout na tlačítko na obrazovce. Vyhrává ten, kdo má větší počet kliknutí.

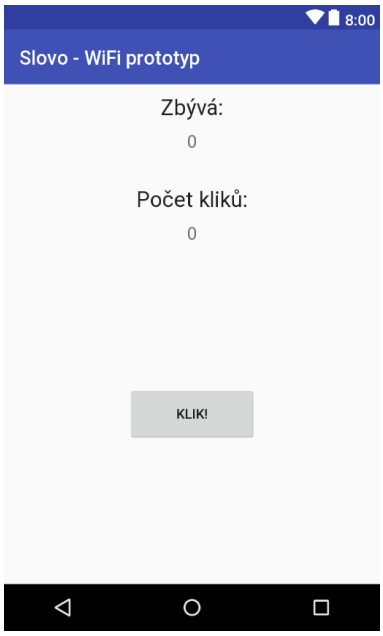
Na první obrazovce jsou umístěny pouze dvě tlačítka: Založit hru a Připojit se, viz obrázek 2.1 (na obrázcích je v záhlaví aplikace uveden WiFi prototyp, obrazovky pro Bluetooth prototyp jsou ale totožné). Hráči, který zvolil možnost Připojit se, se zobrazí dialog se seznamem zařízení. Tento dialog se v závislosti na typu prototypu liší, viz následující sekce. Z tohoto seznamu vybere protihráčovo zařízení a dojde k navázání spojení. Po úspěšném připojení se zakladateli hry objeví obrazovka s nastavením kola hry, viz obrázek 2.2. Zde nastaví časový limit pro jedno kolo a tlačítkem odstartuje hru.

Hráčům se následně objeví obrazovka s odpočtem a herním tlačítkem (obrázek 2.3). Po časovém limitu se hráčům ukáže obrazovka s vyhodnocením



Obrázek 2.1: Úvodní obrazovka prototypu

Obrázek 2.2: Obrazovka nastavení kola



Obrázek 2.3: Obrazovka, kde probíhá hra

Obrázek 2.4: Obrazovka s výsledky

(obrázek 2.4), kde zjistí zda vyhráli, či prohráli a kolik kliknutí se jim a soupeři povedlo udělat. Zakladateli hry se v dolní části obrazovky objeví dvě tlačítka, jedno pro ukončení hry, druhé pro začátek dalšího kola.

2.4.2 Bluetooth prototyp

Nejprve jsem podle návrhu z předchozí kapitoly 2.4.1 a postupu popsaném v kapitole 2.2 vytvořil Bluetooth prototyp.

Jelikož Bluetooth API na systému Android nabízí možnost zobrazit již spárovaná zařízení, skládá se dialog pro připojení ze dvou seznamů. První seznam obsahuje spárovaná zařízení uložená v telefonu, v druhém seznamu se objevují nově nalezená viditelná Bluetooth zařízení.

Při zvolení doposud nespárovaného zařízení se objeví na obou zařízení výzva ke spárování. Po úspěšném spárování dojde k propojení telefonů.

2.4.2.1 Návrh tříd a popis implementace

Základní třídy aplikace jsou takzvané aktivity (objekty třídy Activity, respektive AppCompatActivity), které představují jednotlivé obrazovky. Bluetooth prototyp se skládá ze tří základních obrazovek a jedné obrazovky, kde se provádí nastavení kola. Jedná se o třídy StartScreenActivity, RoundActivity, EndOfRoundActivity a SetRoundActivity. V mé aplikaci jsem si vystačil pouze s konceptem aktivit, nebylo potřeba využívat fragmenty (objekty třídy Fragment), které slouží primárně pro rozdílné zobrazení na telefonech a tabletech.

Ve třídě StartScreenActivity v metodě onCreate() provádím kontrolu, zda zařízení podporuje Bluetooth a nastavuji obsluhu tlačítek pro vytvoření hry a pro připojení se ke hře. První tlačítko deleguje žádost o vytvoření hry na třídu BluetoothController a zobrazí uživateli čekací dialogové okno. Druhé tlačítko spustí vyhledávání okolních zařízení a zobrazí dialogové okno se dvěma seznamy. První obsahuje již spárovaná zařízení, ve druhém seznamu se postupně objevují nově nalezená zařízení. To je realizováno pomocí broadcast receiveru, viz kapitola 2.2.5.2. Položka seznamu se skládá ze jména zařízení a jeho MAC adresy. Při kliknutí na zařízení je ukončeno hledání a žádost o připojení se je delegována opět na třídu BluetoothController.

V metodě onStart() třídy StartScreenActivity kontroluji, zda je Bluetooth zapnutý a pokud ne, požádám o to uživatele pomocí dialogového okna, viz kapitola 2.2.5.1. Pokud je Bluetooth zapnutý nastavím třídě BluetoothController handler, aby bylo možné do aktivity posílat zprávy. Objekt třídy Handler vytvořím jako privátní členskou proměnnou a do jeho metody handleMessage() implementuji schopnost reagovat na zprávy zaslané třídou BluetoothControlller.

Poté, co dojde k připojení, zobrazí se zakladateli obrazovka s nastavením kola. Ta je implementovaná třídou SetRoundActivity, která pouze uloží parametry kola nastavené uživatelem do třídy GameState a nastaví obsluhu

tlačítka pro zahájení kola. Ta spočívá v tom, že pomocí třídy BluetoothController přepošle parametry kola protihráči a spustí obrazovku kola – třída RoundActivity. Klient pomocí handleru dostane zprávu, že byly parametry poslány, uloží si je a také spustí obrazovku kola.

Jak bylo zmíněno, obrazovka prototypu se samotnou hrou obsahuje pouze tlačítko pro započtení kliku, časovač a přehled, kolikrát už uživatel kliknul. V metodě onCreate() třídy RoundActivity nastavím opět handler na obsluhu zpráv ze třídy BluetoothController a vytvořím časovač – objekt třídy CountDownTimer. Nastavím mu dobu podle uloženého parametru a interval na jednu sekundu. Dále implementuji do jeho metody onTick() aktualizaci textového pole podle zbývajícího času a do metody onFinish() uložení skóre, zaslaní jeho hodnoty protihráči pomocí třídy BluetoothController a aktualizaci celkového skóre. Pokud již došlo k obdržení skóre od protihráče, spustí se další obrazovka s vyhodnocením kola.

Stejně jako u úvodní obrazovky je handler privátní členská proměnná se schopností reagovat na zprávy od třídy BluetoothController, konkrétně na zaslání zprávy od protihráče. Při obdržení protihráčova skóre dojde k jeho uložení, k vypočtení celkového skóre protihráče a pokud už vypršel časovač, spustí se zde další obrazovka s vyhodnocením kola.

Tu představuje třída EndOfRoundActivity. V metodě onCreate() nastaví všechny potřebné prvky obrazovky, jako je skóre, celkové skóre, skóre protihráče a jeho celkové skóre a textové pole s nápisem VYHRÁL JSI!, nebo PROHRÁL JSI!. To je nastavené na základě jednoduchého porovnání skóre hráčů.

Na obrazovce zakladatele hry jsou ještě zobrazeny tlačítka pro začátek nového kola a pro ukončení hry. Tlačítko pro začátek kola pouze spustí novou obrazovku s nastavením dalšího kola. Tlačítko pro ukončení hry zašle zprávu o konci protihráči. Ten na základě této zprávy uzavře spojení (soket) pomocí třídy BluetoothController a přepne na úvodní obrazovku. Díky uzavření soketu dojde na straně serveru v metodě read() k vyhození výjimky, která je odchycena, a je zavolána metoda connectionLost(). Ta ukončí komunikační vlákno, nastaví spravný vnitřní stav a přes handler pošle aktivitě zprávu o ukončení hry. Na základě této zprávy je stejně jako u klienta zobrazena úvodní obrazovka.

Dále jsem, jak už bylo zmíněno, vytvořil třídu GameState, kde ukládám parametry kola a další potřebné proměnné sdílené mezi aktivitami a interface Constants, kde jsou k dispozici sdílené konstanty.

Poslední velmi důležitou částí aplikace je již několikrát zmiňovaná třída BluetoothController. Je implementovaná formou návrhového vzoru singleton. Obsahuje několik metod a tři privátní třídy reprezentující tři vlákna popsaná v kapitole 2.2.5.3. První vlákno (třída AcceptThread) je vytvořeno jen na straně serveru, vytváří objekt třídy BluetoothServerSocket a vyčkává na připojení klienta. Druhé vlákno (třída ConnectThread) vytváří pouze klient, je v něm vytvořen objekt třídy BluetoothSocket a na něm zavolána metoda con-

nect() pro připojení k serveru. Po připojení obě strany vytváří třetí vlákno (třída ConnectedThread), které slouží ke vzájemné komunikaci.

O vytvoření těchto vláken se starají metody startAsServer(), ta vytváří vlákno AcceptThread, a connectToServer(BluetoothDevice), ta vytváří vlákno ConnectThread. Po úspěšném připojení je v těchto vláknech volána metoda connected(BluetoothSocket, BluetoothDevice), která vytváří vlákno ConnectedThread a oznamuje uživateli jméno zařízení, ke kterému se připojil.

Dále je tu metoda endGame(), která ukončí bežící vlákna a uvede BluetoothController do počátečního nepřipojeného stavu. Další metody connectionFailed() a connectionLost() oznámí situaci uživateli, zavolají metodu endGame() a pošlou zprávu o ukončení hry aktivitě. Poslední metoda stojící za zmínku je stopWaitingForClient(), která je volána při zrušení založení hry na úvodní obrazovce. Metoda nastaví správný stav a pokud už byl vytvořen soket serveru čekající na připojení, uzavře ho.

2.4.3 Wi-Fi prototyp

Wi-Fi prototyp byl vytvořen stejně jako Bluetooth prototyp podle návrhu z kapitoly 2.4.1 a podle postupu popsaném v kapitole 2.3. Převážnou část kódu má stejnou jako Bluetooth prototyp, proto zmíním jen důležité rozdíly.

Wi-Fi prototyp obsahuje navíc třídu WifiBroadcastReceiver, která je popsaná v kapitole 2.3.3.1. Dále tu je pomocná třída PeerList, která se stará o zobrazení nalezených zařízení a zároveň implementuje interface PeerListListener a jeho metodu onPeersAvailable(), která je volána ve chvíli, kdy je k dispozici seznam nalezených zařízení. Rozdíl je i v tom, že připojení přes Wi-Fi peer-to-peer nepracuje s párováním zařízení. Dialog tedy obsahuje pouze jeden seznam, kde se zobrazují objevená zařízení.

Ve třídě StartScreenActivity v metodě onCreate() je provedena inicializace frameworku Wi-Fi P2P – je získána instance třídy WifiP2pManager, na ní je zavolána metoda initialize(), a poté je vytvořen broadcast receiver s příslušnými filtry a v metodě onResume() je zaregistrován.

Dále bych zmínil už jen rozdíl v získávání soketů. Na straně serveru je poslouchající soket vytvořen pomocí volání new ServerSocket(port), kde parametrem je zvolený port pro aplikaci. Na straně klienta je komunikační soket vytvořen voláním new Socket(). Klient se pak k serveru připojuje pomocí metody socket.connect(new InetSocketAddress(host, port), timeout), kde host je IP adresa serveru.

2.5 Porovnání a vyhodnocení

Technologie Bluetooth a Wi-Fi P2P (Wi-Fi Direct) lze porovnat podle mnoha kriterií. Jsou tomu například:

- rychlost přenosu dat,

- dosah (možná vzdálenost mezi zařízeními),
- bezpečnost,
- spotřeba energie,
- uživatelská přívětivost.

Vzhledem k povaze mé aplikace však nemá smysl většinu kritérii srovnávat. Nejdůležitější pro tuto hru je poslední kritérium, uživatelská přívětivost, a hrát roli by mohla i spotřeba energie. Zbylá kriteria srovnám jen letmo, protože nejsou pro aplikaci adekvátní.

2.5.1 Rychlost přenosu dat

Tato aplikace přenáší mezi zařízeními opravdu jen velmi málo dat. Před každým kolem dojde od zakladatele hry k jeho protihráči k přenosu pouze několika proměnných představujících parametry hry. Po skončení jednoho kola si pak navzájem hráči pošlou své výsledky. V prototypu jde pouze o jedno číslo. V samotné hře se pak jedná o seznam slov, kterých bývá kolem deseti až dvaceti (při tří minutovém kole). V závislosti na typu hry může dojít k výměně slov mezi hráči ještě jednou. Je vidět, že jde o opravdu malé množství dat, na rozdíl od například přenášení obrázků.

Proto zmíním jen základní informace o rychlosti těchto technologií. Přes Wi-Fi Direct by mělo jít dosáhnout rychlosti až 250 Mb/s [9], u Bluetooth 4.0 (podobně jako u Bluetooth 3.0) až rychlosti 25 Mb/s. [10]

2.5.2 Dosah

Tato hra je koncipovaná tak, že jsou hráči spolu na jednom místě a v průběhu hry navzájem komunikují. Tudíž zde nejsou žádné požadavky na velkou vzdálenost propojených zařízení. Aby se spolu mohli hráči bavit, nebudou od sebe dál než 5 metrů. A tento dosah obě technologie bez problému splňují.

Pro zajímavost uvedu jen maximální uváděný dosah. Pro Wi-Fi Direct, stejně jako u každého zařízení s Wi-Fi, je to do vzdálenosti 200 metrů [11], u Bluetooth 4.0 do vzdálenosti 60 metrů. [10]

2.5.3 Bezpečnost

Přenášená data mezi zařízeními v této hře rozhodně nejsou potřeba nijak chránit. Proto toto kritérium opět nehraje žádnou roli.

Jen pro informaci zmíním, že technologie Bluetooth 4.0 využívá šifrování AES se 128bitovým klíčem [10] a přenos dat přes Wi-Fi Direct je zabezpečen programem WPA2 využívající šifrování AES s 256bitovým klíčem. [12]

2.5.4 Spotřeba energie

Pro zjištění, která technologie je náročnější na baterii, jsem provedl na vytvořených prototypech testování. Měření jsem prováděl pomocí programu Battery Historian. Tento program analyzuje soubor vytvořený operačním systémem Android zvaný bug report.

Nejprve jsem připojil telefon k počítači a příkazem `adb shell dumpsys batterystats --reset` vymazal předchozí historii. Poté jsem telefon odpojil a provedl jsem sto kol hry nejprve na Bluetooth prototypu. Každému kolu jsem nastavil dobu jednu sekundu, aby více času hry zabralo samotné přeposílání dat. Po sto kolech, která dohromady trvala zhruba 9 minut, jsem hru ukončil, připojil telefon k počítači a vygeneroval soubor s historií příkazem `adb bugreport bugreport.zip`. To samé jsem provedl s Wi-Fi prototypem.

Následně jsem otevřel oba soubory v programu Battery Historian a porovnal naměřené hodnoty. V přehledu je možné vyfiltrovat hodnoty jen pro jednu aplikaci. To jsem udělal a zjistil, že Bluetooth prototyp během tohoto měření spotřeboval odhadem 0,05% baterie. Wi-Fi prototyp na tom byl o trochu lépe, spotřeboval odhadem 0,03% baterie. Nejedná se ale nijak o zásadní rozdíl.

Oba soubory s historií jsou k dispozici na přiloženém CD.

2.5.5 Uživatelská přívětivost

Nejdůležitějším kritériem pro tuto aplikaci je uživatelská přívětivost. Zde jsem objevil dvě nevýhody technologie Wi-Fi Direct oproti Bluetooth.

První zásadní nevýhoda je, že na některých zařízeních nefunguje Wi-Fi Direct zároveň s připojením ke klasické síti Wi-Fi. Wi-Fi Alliance uvádí, že funkce simultánního připojení přes Wi-fi Direct a připojení k síti Wi-Fi je pro zařízení volitelná. [13] Na mých testovacích telefonech Lenovo P70-A a Samsung GT-I8262 tomu tak skutečně bylo a před připojením k druhému zařízení bylo nutné odpojit se od sítě Wi-Fi. Dnes bývají telefony k síti Wi-Fi připojeny velmi často a vyžadovat od uživatelů, aby se před hrou odpojovali, by bylo dosti nešťastné.

Druhá nevýhoda je víceméně zanedbatelná. Tím, že Wi-Fi P2P nevyužívá koncept párování, je při každém připojování k druhému zařízení potřeba provést vyhledávání. Zařízení se tak na seznamu objeví o trochu později, než je tomu u připojení přes Bluetooth, kde je seznam spárovaných zařízení k dispozici hned. Také není potřeba, aby zakladatel hry připojení pokaždé potvrzoval. Na druhou stranu Wi-Fi P2P při prvním připojení nevyžaduje párování, ale to je jen malé zdržení.

2.5.6 Shrnutí

V tabulce 2.1 jsou pro porovnání zobrazeny výše zmíněné parametry a jejich důležitost.

Tabulka 2.1: Shrnutí porovnání technologií Bluetooth a Wi-Fi

Parametr	Důležitost	Bluetooth	Wi-Fi Direct
uživatelská přívětivost	vysoká	žádná nevýhoda	potřeba odpojení od Wi-Fi sítě
spotřeba energie (při testu)	střední	0,05%	0,03%
rychlost přenosu dat	nízká	25 Mb/s	250 Mb/s
dosah	nízká	60 m	200 m
bezpečnost	nízká	AES (128b klíč)	AES (256b klíč)

2.5.7 Vyhodnocení

Na základě tohoto porovnání, zejména kvůli kritériu Uživatelská přívětivost, jehož důležitost je vysoká, jsem se rozhodl pro variantu využívající technologii Bluetooth. Problém s nutností u některých zařízení se před hrou odpojit od Wi-Fi sítě převážil ostatní nedůležité parametry, ve kterých technologie Wi-Fi Direct dosahovala lepších hodnot než Bluetooth.

Dodávám, že nevýhody Wi-Fi Direct nejsou nijak vážné a neznemožňovaly by implementaci hry pomocí této technologie.

Kapitola 3

Návrh a implementace aplikace

Při navrhování aplikace jsem vycházel z funkčních a nefunkčních požadavků. Nejprve bylo potřeba vytvořit wireframes uživatelského rozhraní, abych získal konkrétnější představu o tom, jaké obrazovky bude aplikace obsahovat a jaké prvky jsou na obrazovkách potřeba.

Poté jsem přistoupil k návrhu samotné implementace. Jelikož se nejedná o velký projekt, ani nebyl vytvářen v týmu, nebylo potřeba návrh rozpracovávat do příliš velkých podrobností.

3.1 Wireframes

Jak uvádím ve své bakalářské práci [14], kde jsem se tímto tématem více zabýval, wireframes jsou druh modelu, který slouží především k představě, jakou bude mít artefakt (aplikace) strukturu uživatelského rozhraní. Ukazuje nám, jaké obsahuje stránky a co se na nich bude nacházet. Pomáhá nám uvědomit si, jaké prvky jsou potřeba použít pro naplnění požadavků na artefakt.

Dále slouží jako nástroj komunikace, ať už mezi spolupracovníky návrhu, nebo při jeho prezentaci. Grafické vyjádření je často mnohem srozumitelnější než jen textový či slovní popis.

Wireframes neslouží jako grafický návrh. Často nepoužívají barvy a grafické prvky jsou ve zjednodušené podobě. Při použití grafických prvků a barev může totiž docházet k problémům v komunikaci a to především k odvádění pozornosti od obsahu jednotlivých stránek a soustředění se na grafický vzhled.

Pro svou aplikaci jsem zvolil základní papírovou variantu wireframes, kterou můžete vidět na obrázku 3.1 nebo v příloze B. Jedná se sice jen o jednoduchý náčrtek, ale splnil přesně to, k čemu wireframes slouží. V horní části wireframes jsou uvedeny základní obrazovky hry, níže pak některé další obrazovky. V následujících sekcích je ke každému funkčnímu požadavku uvedena patřičná část návrhu.

Obrázek 3.1: Papírové wireframes

3.1.1 Požadavek F1

Tento požadavek se týká úvodní obrazovky. Vyžaduje pouze možnost založit hru a připojit se k založené hře. Proto jsou na obrazovce umístěna dvě tlačítka, jedno pro založení hry, druhé pro připojení se ke hře.

3.1.2 Požadavek F3

Požadavek F3 specifikuje obrazovku pro samotnou hru. Jedno z možných řešení bylo zobrazit hlavní slovo v horní části obrazovky, vytvořená slova by se psala na běžné klávesnici systému Android, která se zobrazí na displeji po kliknutí na editovatelné textové pole. Toto řešení má však řadu nevýhod. První

nevýhodou je, že by klávesnice zabrala velkou část obrazovky. Další nevýhodou, nebo lépe řečeno zbytečností, je, že by většina písmen na klávesnici byla nevyužita a nebylo by možné nějakým způsobem naznačit, že je nelze použít.

Proto jsem zvolil řešení jiné, a to, které přímo vychází z principu hry. Hráč může vybírat pouze z písmen hlavního zvoleného slova. Využil jsem tedy samotné slovo a navrhl ho tak, aby sloužilo zároveň pro zadávání písmen nových vytvářených slov. Pro lepší ovládání jsem hlavní slovo umístil do dolní části obrazovky, stejně jako se zobrazuje klávesnice. Jednotlivá písmena jsem ohraničil rámečkem, který naznačuje, že je každé písmeno zároveň tlačítkem. Po kliknutí na písmeno se tlačítko změní, aby naznačilo, že na něj již není možné kliknout.

Kromě písmen sloužících k vytváření nových slov bylo potřeba uživateli dovolit také písmena z vytvářeného slova mazat. Pokud hráč například během psaní zjistí, že mu nějaké písmeno do zamýšleného slova chybí, má možnost ho příslušným tlačítkem smazat.

Třetím potřebným ovládacím tlačítkem je tlačítko pro potvrzení vytvořeného slova.

Jak je vidět na obrázku 3.1 s wireframes, v první verzi návrhu byla všechna tři tlačítka umístěna vedle sebe pod hlavní slovo. Toto rozmístění však není příliš vhodné. Tři stejně velká tlačítka vedle sebe naznačují, že jde o tři stejně často používané možnosti nebo o možnosti na stejné úrovni. Zde však je nejčastěji používané tlačítko pro potvrzení slova.

Provedl jsem tedy druhou iteraci návrhu a vytvořil novou verzi obrazovky č. 2 (na obrázku vpravo dole). Tlačítko pro potvrzení slova je umístěno samotné v nejspodnější části obrazovky. Tlačítko pro smazání písmena jsem umístil vedle hlavního slova, kde je uživatel očekává, protože je na to zvyklý z běžných klávesnic. Tlačítko pro smazání slova jsem pak umístil vedle vytvářeného slova stejně, jako tomu bývá například ve vyhledávacích polích.

Do horní části obrazovky je pro přehled hráče umístěn seznam již vytvořených slov a do pravého horního rohu odpočet – zbývající čas do konce kola.

3.1.3 Požadavek F4

Podle specifikace popsané v požadavku F4 má hra umožňovat režim kontroly slov protihráčem. To zobrazuje třetí obrazovka. Obsahuje seznam slov s možností libovolné slovo vyškrtnout. Šipka nad obrazovkou naznačuje, že při jiném režimu kontroly nedochází k jejímu zobrazení.

3.1.4 Požadavek F5

Poslední obrazovka v horní části zobrazuje vyhodnocení kola. Obsahuje velký nápis s informací, zda hráč vyhrál, nebo prohrál. Pod tím se nachází počet bodů hráče i protihráče, dále seznam slov hráče následovaný seznamem slov protihráče. Pro znázornění obou seznamů za sebou je pod obrazovkou nakres-

lena i zbylá část, ke které se musí dorolovat. Vedle každého slova je umístěno jeho bodové skóre. U několika slov je naznačeno znakem 2x, že je slovo za dvojnásobný počet bodů díky tomu, že protihráč toto slovo nevymyslel.

3.1.5 Požadavek F5

Na základě požadavku F5 jsou do dolní části obrazovky s vyhodnocením umístěna dvě tlačítka, jedno pro ukončení hry, druhé pro začátek dalšího kola. Tato tlačítka se zobrazí pouze zakladateli hry.

3.1.6 Požadavek F2 a dialogová okna

Druhý řádek na obrázku 3.1 s wireframes naznačuje obrazovky, které se zobrazí zakladateli hry, na třetím řádku jsou obrazovky jeho protihráče.

Zakladateli hry se nejprve ukáže čekací dialogové okno a po připojení protihráče se objeví obrazovka s nastavením kola. Ta by měla obsahovat možnost nastavit všechny parametry vypsané v požadavku F2. Pro vybrání náhodného slova slouží tlačítko vpravo od editovatelného pole.

3.2 Návrh tříd a popis implementace

Základ návrhu vychází z Bluetooth prototypu. Zachoval jsem všechny jeho třídy, některé jsem upravil a poté několik nových tříd přidal.

3.2.1 Slovník

Začnu s návrhem implementace slovníku. Podle specifikace hry (kapitola 1.1.2) má uživatel na výběr ze třech způsobů kontroly správnosti vytvořených slov:

- kontrola protihráčem až při vyhodnocení,
- podle slovníku,
- bez kontroly.

Kvůli druhému způsobu bylo potřeba implementovat kontrolu slov podle slovníku.

3.2.1.1 Seznam slov

Oficiální elektronický seznam povolených slov pro hru Scrabble není k dispozici. Existuje elektronický slovník Novex, jehož registrace je placená. Zdarma je nabízená verze, která umožňuje kontrolu slov do délky 4 písmen. Program však slova jen kontroluje, nelze z něj získat seznam slov.

Dále existují tištěné slovníky, které jsou ale pro mou práci samozřejmě nevhodné.

Nakonec se mi podařilo získat neoficiální elektronický seznam slov obsahující 707 774 slov, což je velmi blízko slovníku Novex Klasik, který má 776 172 slov.

3.2.1.2 Způsob implementace

Soubor se slovy je dost velký a nebylo by vhodné celý seznam udržovat v paměti a vyhledávat v něm. Proto jsem zvolil uložení slov do databáze. Jedna z nejrozšířenějších relačních databází na systému Android je SQLite, se kterou jsem měl i zkušenosti. Zvolil jsem tedy ji.

Pro data jsem vytvořil tabulku pouze s jedním atributem, do kterého se budou ukládat slova. Atribut jsem díky jedinečnosti slov nastavil zároveň jako primární klíč. Díky tomu si databáze vytvoří podle tohoto atributu index, a vyhledávání tak bude velmi rychlé.

Databázi jsem si vytvořil předem v programu DB Browser for SQLite ze získaného seznamu slov.

3.2.1.3 Pomocná třída SQLiteAssetHelper

Android neumožňuje pracovat s externí databází přímo. Proto jsem využil pomocnou třídu com.readystatesoftware.sqliteasset.SQLiteAssetHelper [15], která zajistí potřebné překopírování databáze a přístup k datům. Tato třída nabízí jednoduchý přístup k již vytvořené databázi, stačí pouze soubor s databází nahrát do adresáře Assets, a to i v komprimované podobě.

Nejprve jsem si vytvořil svou třídu DatabaseOpenHelper dědící od pomocné třídy SQLiteAssetHelper. Ta obsahuje pouze konstruktor volající konstruktor předka s parametry název databáze a verze.

Poté jsem si udělal (podle návodu na webové stránce [16]) pomocnou třídu DatabaseAccess, která zajišťuje přístup k databázi. Tato třída používá návrhový vzor singleton. Oproti návodu jsem si implementaci trochu upravil. Nahradil jsem lazy inicializaci v metodě getInstance() metodou initialize(Context), protože parametr context je potřeba pouze při vytvoření třídy DatabaseOpenHelper, a tudíž není potřeba ho vyžadovat při každém volání metody getInstance().

Dále už jsem jenom přidal metodu isInDictionary(String) na zjištění, zda se zadané slovo nachází ve slovníku, či nikoliv. Její implementace je vidět zde:

```
public boolean isInDictionary(String word) {
    String[] projection = { BaseColumns._ID };
    String selection = BaseColumns._ID + " = ?";
    String[] selectionArgs = { word.toLowerCase() };
    Cursor cursor = mDatabase.query(TABLE_NAME, projection, selection,
            selectionArgs, null, null, null);
    boolean isWordInDictionary = cursor.moveToNext();
    cursor.close();
    return isWordInDictionary; }
```

BaseColumns._ID značí textový řetězec _id. Takto je potřeba v databázi SQLite na systému Android pojmenovat sloupec s primárním klíčem. Jak bylo zmíněno výše tabulka obsahuje pouze jeden sloupec se slovy, který je zároveň primárním klíčem.

3.2.2 Dialogy

V implementaci prototypů jsem pro zjednodušení využíval na dialogová okna přímo třídu Dialog nebo AlertDialog. To mělo ovšem jednu zásadní nevýhodu. Při otočení displeje (či jiné změně zdrojů, anglicky resources) dochází totiž na systému Android k obnovení celé obrazovky (potomku třídy Activity). Dojde tak ke zmizení dialogového okna.

Abych tomu zabránil, bylo potřeba pro všechna dialogová okna vytvořit vlastní třídy dědící od třídy DialogFragment.

Fragment je část obrazovky, se kterou se pracuje samostatně. Slouží zejména pro navrhování uživatelského rozhraní pro tablety. Příkladem může být fragment obsahující seznam položek a fragment s detailem položky. Na tabletu lze do jedné aktivity vložit oba fragmenty, na telefonu se vloží do aktivity jen fragment se seznamem a do druhé aktivity fragment s detailem. Fragmenty mají stejně jako aktivity svůj životní cyklus a jsou s aktivitou svázané. Podrobněji například zde [17]. Díky tomu se po změně zdrojů fragmenty obnoví.

Třída DialogFragment je speciálním typem fragmentu, který v sobě obsahuje třídu Dialog a nabízí metody pro manipulaci s ním. Takto jsem tedy vytvořil třídy WaitingForClientDialogFragment, DevicesListDialogFragment, WaitingDialogFragment a WordsCheckDialogFragment. Pro komunikaci s aktivitou vytvářím ve fragmentech rozhraní s metodami, které musí aktivita (v roli posluchače, anglicky listener) implementovat.

3.2.3 Úvodní obrazovka

Úvodní obrazovka (obrázek 3.2) je představována třídou StartScreenActivity a využívá také tři dialogová okna reprezentovaná třídami WaitingForClientDialogFragment, DevicesListDialogFragment a WaitingDialogFragment, který je použit i na dalších místech aplikace. Této třídě se pouze v konstruktoru předá text, který se v dialogu zobrazí.

3.2.3.1 Třída StartScreenActivity

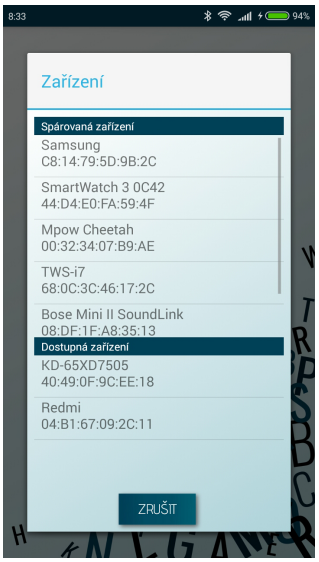
Tato třída se oproti implementaci v Bluetooth prototypu změnila jen minimálně.

V onCreate() se spouští navíc ve vlastním vlákně inicializace databáze se slovníkem.

Implementace dialogů se, jak už jsem zmínil, přesunula do jejich vlastních tříd. Zde pouze zajišťuji jejich korektní vytvoření pomocí třídy FragmentMa-

Obrázek 3.2: Úvodní obrazovka

Obrázek 3.3: WaitingForClientDialogFragment



Obrázek 3.4: DevicesListDialogFragment

nager. Přibyla také implementace metod, které dialogy využívají pro komunikaci s aktivitou.

Poslední novou věcí je reakce na nový typ zprávy MESSAGE_END_GAME v privátní proměnné mHandler, která zajistí opětovné spuštění aktivity, například pokud se nepovede navázat spojení.

3.2.3.2 Třída WaitingForClientDialogFragment

Toto dialogové okno se zobrazuje zakladateli hry. Oproti prototypu jsem zde přidal tlačítko *Zviditelnit se*, jak je vidět na obrázku 3.3. To slouží pro případ, kdy nejsou zařízení spárovaná a telefon zakladatele hry není viditelný. Po zvolení této možnosti a potvrzení uživatelem se zařízení stane viditelným na dobu 120 sekund.

3.2.3.3 Třída DevicesListDialogFragment

Tato třída pouze implementuje původní dialog formou fragmentu. K žádné funkční změně zde nedošlo, jen byl upraven vzhled, jak je vidět na obrázku 3.4.

3.2.3.4 Třída HelpDialogFragment

Do pravého horního rohu jsem přidal tlačítko s ikonou otazníku pro zobrazení pravidel hry. Kliknutím se otevře dialogové okno reprezentované třídou HelpDialogFragment.

Do textu nápovědy bylo potřeba přidat i obrázky některých ikon. To jsem provedl pomocí vlastních značek v textu, které metodou addIconToSpannableString nahradím za příslušné ikony pomocí tříd SpannableString a ImageSpan. Vše je vidět v následujícím kódu:

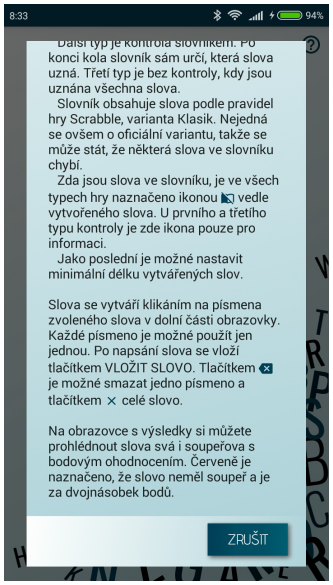
```
public static HelpDialogFragment newInstance() {
    String help = App.getContext().getString(R.string.help_text,
            ICON_MARK_SHUFFLE,
            ...
            ICON_MARK_CLEAR2);
    ...
    args.putString(MESSAGE_ID, help);
    ...
}

@Override
public Dialog onCreateDialog(Bundle savedInstanceState) {
    ...
    SpannableString spannableString = new
        SpannableString(arg.getString(MESSAGE_ID));
    addIconToSpannableString(ICON_MARK_SHUFFLE,
        R.drawable.ic_shuffle_24dp, spannableString);
    ...
    addIconToSpannableString(ICON_MARK_CLEAR2,
        R.drawable.ic_clear_24dp, spannableString);
    builder.setMessage(spannableString);
    ...
}

private void addIconToSpannableString(String iconMark, int
    iconResourceId, SpannableString spannableString) {
    Drawable icon = AppCompatResources.getDrawable(App.getContext(),
        iconResourceId);
    int lineHeight = new TextView(getContext()).getLineHeight();
    icon.setBounds(0, 0, lineHeight, lineHeight);
    ImageSpan span = new ImageSpan(icon, ImageSpan.ALIGN_BOTTOM);
    int index = spannableString.toString().indexOf(iconMark);
    spannableString.setSpan(span, index,index + iconMark.length(),
        Spannable.SPAN_INCLUSIVE_EXCLUSIVE);
}
```

Nejprve do argumentů fragmentu uložím text, který má být v dialogu zobrazen a kam na příslušná místa vložím textové značky ikon. Při vytváření dialogu pak tyto značky nahradím ikonami. Velikost ikony nastavím stejnou jako je velikost řádku.

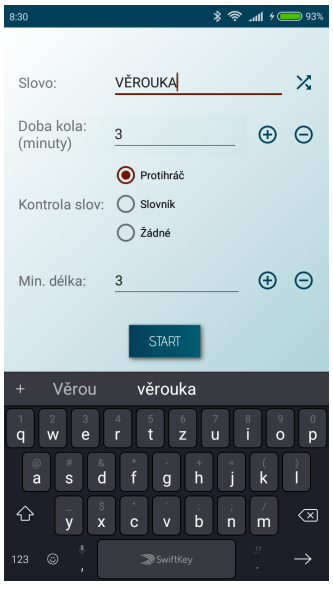
Dialogové okno s textem nápovědy i ikonami v textu je možné vidět na obrázku 3.5.



Obrázek 3.5: HelpDialogFragment

3.2.4 Obrazovka s nastavením kola

Tato obrazovka (obrázek 3.6) je reprezentovaná třídou SetRoundActivity. Oproti prototypu se změnily parametry kola, které zakladatel hry nastavuje.



Obrázek 3.6: Obrazovka s nastavením kola

První pole je editovatelný text sloužící pro vložení vybraného hlavního slova. Kvůli grafické jednotě je na toto pole aplikován filtr, který mění všechna

písmena na velká. Pro univerzální použití, která zachová případné jiné filtry, je filtr nastaven takto:

```
InputFilter[] editFilters = mWordEditText.getFilters();
InputFilter[] newFilters = new InputFilter[editFilters.length + 1];
System.arraycopy(editFilters, 0, newFilters, 0, editFilters.length);
newFilters[editFilters.length] = new InputFilter.AllCaps();
mWordEditText.setFilters(newFilters);
```

Vedle tohoto editovatelného pole jsem vložil tlačítko pro vygenerování náhodného přednastaveného slova do hry. Oproti wireframes, kde byl naznačen obrázek hracích kostek, jsem nakonec použil ikonu překřížených šipek, která se používá například pro náhodné přehrávání skladeb v hudebním přehrávači. Tlačítko spouští metodu randomWord(), které vloží do textového pole náhodné slovo z předdefinovaného seznamu a zároveň zajistí, aby slovo nebylo vygenerováno znovu. V příštím kole je ale opět možné vygenerovat slovo z celého předpřipraveného seznamu kromě slov, se kterými se hrálo. To provedu pomocí dvou polí třídy ArrayList<String>. První pole obsahující doposud nepoužitá slova si udržuji celou hru. Druhé pole vytvářím každé kolo jako kopii prvního pole. Slova pak generuji z této kopie. Každé vygenerované slovo následně z kopie smažu a tím zajistím, že nedojde v jednom kole k vygenerování stejného slova. Pokud už nejsou k dispozici žádná předdefinovaná slova, vložím do textového pole prázdný řetězec. Nakonec při odstartování kola smažu vybrané slovo i z původního pole, takže v dalších kolech nedojde k jeho vygenerování.

```
protected void onCreate(Bundle savedInstanceState) {
   ...
   mRandomWords = new ArrayList<>(mGameState.randomWords);
}

public void randomWord(View v){
    if (!mRandomWords.isEmpty()) {
        int randomIndex = new Random().nextInt(mRandomWords.size());
        mWordEditText.setText(mRandomWords.get(randomIndex));
        mRandomWords.remove(randomIndex);
    } else {
        mWordEditText.setText("");
    }
}

public void startRound(View view){
   ...
   mGameState.randomWords.remove(mGameState.word);
}
```

Dále je zde pole pro nastavení doby kola. Pole má nastavený parametr inputType na number, aby do něj šla zadávat jenom čísla. To zároveň způsobí, že se uživateli při kliku do pole zobrazí rovnou klávesnice s čísly. V původní implementaci se zadávala doba v sekundách. Na základě uživatelského testování však byly jednotky změněny na minuty a byla přidána tlačítka pro prodlužování a zkracování doby. Podrobněji to rozvádím v kapitole 4.4.5.

Dalším polem je skupina takzvaných radio tlačítek (anglicky radio buttons), které zaručují, že uživatel vybere jen jednu možnost. Slouží pro výběr typu kontroly slov.

Posledním polem je minimální délka vytvářených slov. Parametr inputType byl opět nastaven na number.

V dolní části obrazovky se nachází tlačítko na spuštění kola. Po kliknutí je zavolána metoda StartRound(), která vezme vyplněné údaje, uloží je, pošle protihráči a spustí obrazovku se samotnou hrou. U zvoleného slova se provede ještě případné odstranění bílých znaků (nejčastěji se může jednat o mezery).

Aby uživatel viděl tlačítko pro zahájení hry, i když se mu zobrazí na obrazovce klávesnice, bylo potřeba u této aktivity v manifestu nastavit parametr android:windowSoftInputMode na stateVisible|adjustResize a všechny pole obalit do třídy ScrollView. Tlačítko se tak při zobrazení klávesnice posune nahoru a je možné na něj kliknout, aniž by bylo potřeba klávesnici skrýt.

3.2.5 Obrazovka kola hry

Tato obrazovka je reprezentovaná třídou RoundActivity a využívá dvě dialogová okna třídy WordsCheckDialogFragment a WaitingDialogFragment. Oproti prototypu se obrazovka dosti liší, zůstává jen základní přeposlání dat protihráči, a to pouze v režimu bez kontroly slov. Ostatní části kódu jsou nové.

Obrazovka využívá rozvržení prvků podle návrhu ve wireframes popsaném v kapitole 3.1 a to druhou upravenou variantu na obrázku 3.1 vpravo dole. Náhled obrazovky je možné vidět na obrázku 3.7.

3.2.5.1 Třída RoundActivity

V této aktivitě provedu nejprve v metodě onCreate() inicializaci všech prvků na obrazovce. Zejména se jedná o tlačítka s písmeny zvoleného slova. Slovo je známé až za běhu aplikace, tudíž není možné nastavit tlačítka v XML šabloně a je potřeba je vytvořit programově. V cyklu projdu všechna písmena slova a pro každé vytvořím příslušné tlačítko, nastavím jeho vzhled, text a akci po kliknutí, jak je vidět v následujícím kódu:

```
for (int i = 0; i < mGameState.word.length(); i++) {
    Button button = new Button(this);
    LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.MATCH_PARENT,
            LinearLayout.LayoutParams.MATCH_PARENT,
```

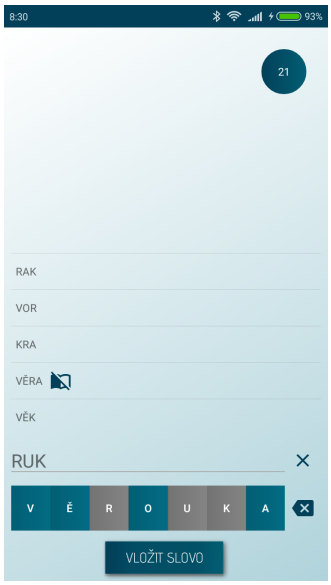
```
            1.0f);
    button.setLayoutParams(params);
    button.setPadding(0, 0, 0, 0);
    button.setBackgroundResource(R.drawable.letter_button);
    button.setTextColor(getResources().getColor(R.color.color7));

    button.setText(String.valueOf(mGameState.word.charAt(i)));
    button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            appendLetter((Button) v);
        }
    });
    mLettersLinearLayout.addView(button);
}
```

Dále nastavím třídě BluetoothController handler pro komunikaci s aktivitou.

Poté vytvořím adaptér pro třídu ListView, která bude zobrazovat seznam hráčových vytvořených slov na obrazovce. Abych mohl do ListView vkládat vlastní položky, vytvořil jsem si třídu MyArrayAdapter. Adaptér jsem udělal univerzální, abych ho mohl využít i na dalších místech aplikace. Podrobněji ho popisuji dále v kapitole 3.2.6.

Do adaptéru vkládám položky třídy ItemWordWithDictionaryMark reprezentující vytvořená slova. Nehledě na typ hry jsem se rozhodl uživateli pro informaci zobrazovat u každého slova, zda nepatří do slovníku. To provádím



Obrázek 3.7: Obrazovka kola hry

pomocí vlastní vytvořené ikony znázorňující přeškrtlou knihu. Položka sama se při svém vytvoření v konstruktoru podívá do databáze a zjistí si, zda nepatří do slovníku. Na základě toho se ikona buď zobrazí, nebo ne.

Dále v metodě onCreate() odstartuji odpočet. Ten je implementován pomocí samostatné vlastní třídy Timer, která je potomkem třídy CountDownTimer. Tato třída mi umožňuje s časovačem dále pracovat. Obsahuje rozhraní (interface) TimerListener, s metodami onTick() a onFinish(), které definuje komunikaci s aktivitou (či jiným listenerem). Instance listeneru je ve třídě uložena v privátní proměnné a je nastavována metodami registerListener() a unregisterListener(). Své vlastní metody onTick() a onFinish(), které musí implementovat potomek třídy CountDownTimer, deleguje třída přímo na listener. V konstruktoru třídy je volán konstruktor předka, kterému se předají dva parametry – doba odpočtu a interval určující, jak často se bude spouštět metoda onTick(). V mé aplikaci dobu odpočtu nastavuje zakladatel hry a interval je vždy jedna sekunda.

Ve třídě RoundActivity, která rozhraní TimerListener implementuje, najdeme tedy tyto metody implementované. V metodě onTick(), která je volána každou sekundu, dochází pouze k aktualizaci textového pole reprezentujícího počet sekund do konce kola.

V metodě onFinish() dochází k ukončení kola. Nejprve jsou kvůli algoritmu porovnání slova seřazena abecedně. Pokud uživatel žádné slovo nevyplnil, je do seznamu slov vložen speciální znak TAG_NO_WORDS. Poté jsou slova v závislosti na roli hráče a typu hry poslána soupeři. Pokud je hráč v roli klienta, posílá slova vždy. Zakladatel hry slova posílá pouze v režimu kontroly slov protihráčem.

Na obrazovce se nachází několik tlačítek. Ke každému patří příslušná metoda, která je volána po kliknutí na tlačítko. Dále je tu metoda appendLetter() pro připsání písmena k vytvářenému slovu, která se spustí, když uživatel klikne na písmeno. Tlačítko s písmenem se zároveň stane neaktivní. Kvůli možnosti smazat poslední písmeno vkládám tlačítko do zásobníku. Díky tomu jsem schopen při smazaní písmena tlačítko udělat opět aktivní.

```
public void appendLetter(Button letterButton){
    String letter = letterButton.getText().toString();
    mNewWord.setText(mNewWord.getText() + letter);
    letterButton.setEnabled(false);
    mLetterStack.add(letterButton);
    mInsertWordButton.setEnabled(true);
}
```

Metoda backspace() je spouštěna příslušným tlačítkem a odmazává poslední písmeno vytvářeného slova. Jak už bylo zmíněno také znovu aktivuje tlačítko s písmenem.

```
public void backspace(View view) {
    String text = (String) mNewWord.getText();
```

```
    if (!text.equals("")) {
        mNewWord.setText(text.substring(0, text.length() - 1));
        mLetterStack.pop().setEnabled(true);
    }
}
```

Metoda clear() je volána po kliku na tlačítko s ikonou křížku a vymazává celé slovo. Pomocí zásobníku opět aktivuje všechna neaktivní tlačítka s písmeny.

```
public void clear(View view) {
    mNewWord.setText("");
    while (!mLetterStack.empty()) {
        mLetterStack.pop().setEnabled(true);
    }
    mInsertWordButton.setEnabled(false);
}
```

Poslední metodou je insertWord(). Před vložením vytvářeného slova do seznamu provádí několik kontrol. Nejprve kontroluje délku slova oproti nastavené minimální délce. V případě, že je slovo kratší, oznámí to aplikace uživateli.

Dále kontroluje, zda se slovo neshoduje s hlavním zvoleným slovem. Poslední kontrola projde všechna doposud vytvořená slova a zjistí, zda už stejné slovo nebylo vloženo. Pokud ano, je i v těchto případech uživateli zobrazeno upozornění a vytvářené slovo je vymazáno.

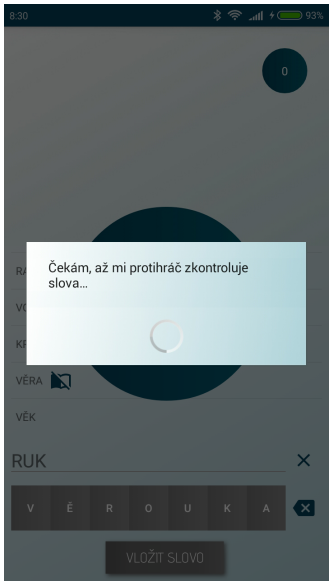
Nakonec je slovo přidáno do ListView a také uloženo. V případě, že je nastavena kontrola slovníkem a slovo se ve slovníku nenachází, uloženo není.

K implementaci reakce na příchozí zprávu od protihráče v Bluetooth prototypu, bylo potřeba doplnit několik možností, které mohou nastat v závislosti na typu kontroly slov. Také byl změněn způsob vyhodnocení, který se provádí jen na straně zakladatele hry, aby se zamezilo zbytečnému posílání dat.

V případech bez kontroly nebo kontroly slov slovníkem je scénář následující. Zakladatel hry je v roli serveru, jeho protihráč v roli klienta. Po skončení kola posílá klient svá slova serveru. Ten provede vyhodnocení (metoda evaluateRound()) a výsledky spolu se svým seznamem slov zasílá zpět klientovi. Výsledky obsahují skóre obou hráčů a příznaky, jaká slova byla ohodnocena dvojnásobným počtem bodů (za to, že je neměl soupeř).

V případě kontroly slov protihráčem probíhá komunikace takto. Nejprve si obě strany předají seznam svých slov. Uživateli se zobrazí soupeřova slova a má možnost libovolná slova odstranit. Po potvrzení uživatelem už přenos probíhá jako v předchozím případě. Klient zasílá slova serveru, ten kolo vyhodnotí a pošle zpátky výsledky a svá slova.

Pokud uživatel zkontroluje slova dříve než jeho protihráč, ukáže se mu jednoduché čekací dialogové okno reprezentované třídou WaitingDialogFragment (obrázek 3.8).



Obrázek 3.8: WaitingDialogFragment

Tyto scénáře jsou implementovány v privátní proměnné mHandler, která stejně jako v ostatních aktivitách zajišťuje komunikaci s třídou BluetoothController, konkrétně v reakci na zprávu typu MESSAGE_READ. Tato zpráva je do aktivity zaslána, pokud BluetoothController obdrží data od protihráče.

Vyhodnocení kola, jak už bylo zmíněno, provádí strana serveru v metodě evaluateRound(). Pro vyhodnocení je potřeba, aby byly seznamy slov abecedně seřazené. Díky tomu je možné efektivně slova porovnat a zjistit, která se nacházejí v seznamech obou hráčů.

Algoritmus porovnání je podobný části řadícího algoritmu Mergesort. Pracuje s dvěma indexy, každý pro jeden seznam. Na začátku ukazují na první slova seznamů. Algoritmus porovná slova na daných indexech a provede jednu ze tří možností. Pokud jsou slova stejná, posune oba indexy na další pozici. Pokud je první slovo v abecedě dříve než druhé, posune se první index a slovu přidá příznak, že ho soupeř nemá. V opačném případě se posune druhý index a nastaví se příznak u druhého slova.

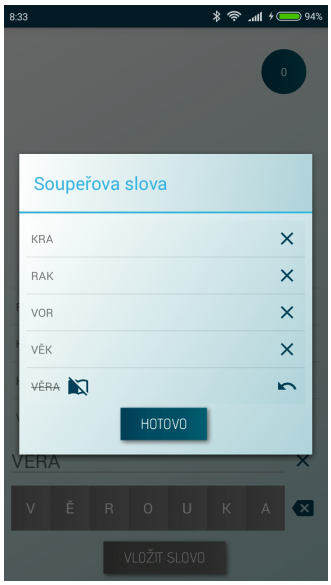
Ve všech případech dochází rovnou k aktualizaci skóre. Příznaky slouží pouze pro obrazovku s vyhodnocením, kde se v seznamu slov u každého slova zobrazí jeho bodová hodnota a zvýrazní se, pokud protihráč slovo neměl.

Toto se provádí, dokud jeden ze seznamů slov nedojde. Poté se se projdou zbylá slova delšího seznamu a všem se nastaví příznak, že je protihráč neměl.

Ještě je potřeba ošetřit případy, kdy první hráč nevyplní žádné slovo, druhý hráč nevyplní žádné slovo, nebo oba hráči nic nevyplní.

3.2.5.2 Třída WordsCheckDialogFragment

Tato třída reprezentuje dialogové okno se slovy protihráče, které se zobrazí po konci kola v režimu kontroly slov protihráčem (obrázek 3.9).



Obrázek 3.9: WordsCheckDialogFragment

Okno obsahuje třídu ListView opět s adaptérem MyArrayAdapter. Do adaptéru jsou vložena soupeřova slova formou třídy ItemWordWithCheck. Třída představuje jednu položku, která obsahuje samotné slovo, tlačítko s ikonou křížku pro smazání slova a může obsahovat ikonu značící, že se slovo nenachází ve slovníku. Ta se zobrazuje na stejném principu jako tomu bylo u třídy ItemWordWithDictionaryMark.

Třída implementuje funkcionalitu smazaní slova. K tomu využívá přístup k celému seznamu protihráčových slov, který je do třídy předán v konstruktoru. Při kliku na tlačítko pro smazání je slovo v seznamu přeškrtnuto a je změněna ikona tlačítka na šipku zpět. Při opětovném kliku dojde k navrácení slova tak, že je zrušeno přeškrtnutí a tlačítku vrácena ikona křížku. Implementačně je volána následující metoda třídy View.OnClickListener:

```
@Override
public void onClick(View v) {
    if (mIsRemoved) {
        mIsRemoved = false;
        setViewsAddedWord(holder);
        mOpponentWords.add(mWord);
    }
    else {
        mIsRemoved = true;
```

```
            setViewsRemovedWord(holder);
            mOpponentWords.remove(mWord);
        }
    }
```

Položka si uchovává informaci o tom, zda je slovo odstraněné, a na základě toho se chová. Při odstranění slova jej rovnou odstraní z uložených soupeřových slov, při vrácení slova slovo opět vloží.

Metoda setViewsRemovedWord() nastaví zmíněné přeškrtnutí a změnu ikony, metoda setViewsAddedWord() udělá opak. Přeškrtnutí a odškrtnutí se provede následovně pomocí metody setPaintFlags() a bitových operací:

```
private void setViewsRemovedWord(ViewHolder holder){
    holder.wordTextView.setPaintFlags(holder.wordTextView
      .getPaintFlags() | Paint.STRIKE_THRU_TEXT_FLAG);
    holder.checkWordButton.setImageResource(R.drawable.ic_undo_24dp);
}

private void setViewsAddedWord(ViewHolder holder) {
    holder.wordTextView.setPaintFlags(holder.wordTextView
      .getPaintFlags() & ~Paint.STRIKE_THRU_TEXT_FLAG);
    holder.checkWordButton.setImageResource(R.drawable.ic_clear_24dp);
}
```

3.2.6 Třída MyArrayAdapter

Jak už jsem zmínil, vytvořil jsem univerzální adaptér umožňující vkládat do ListView vlastní položky. Třída je potomkem třídy ArrayAdapter<Item>. Item je rozhraní se dvěma metodami getViewType() a getView().

Hlavní funkcí tohoto adaptéru je možnost vkládat položky dvojího typu. Prvním typem je LIST_ITEM reprezentující běžnou položku seznamu, druhým typem je HEADER_ITEM, který představuje záhlaví. Seznam je tak možné dělit na sekce vložením této položky před každou část.

Do adaptéru je tedy možné vkládat libovolné položky implementující rozhraní Item. V této aplikaci používám tři třídy typu LIST_ITEM (ItemWordWithDictionaryMark, ItemWordWithCheck a ItemWordWithScore) a jednu třídu typu HEADER_ITEM (ItemHeader). Ve všech položkách implementuji návrhový vzor holder [18] – v privátní třídě ViewHolder si udržuji reference na všechny prvky položky, aby je nebylo potřeba vyhledávat náročnou metodou findViewById() při rolování seznamu. ViewHolder ukládám do View pomocí metody setTag() a získávám metodou getTag().

V předchozích aktivitách jsem adaptér využil pouze pro jeden typ položky, konkrétně ItemWordWithDictionaryMark a ItemWordWithCheck. V následující aktivitě využiji již jeho hlavní funkci a budu vkládat položky dvojího typu – ItemWordWithScore a ItemHeader. Vytvořím tak v seznamu slov dvě sekce, jednu pro hráčova slova, druhou pro slova protihráče.

3.2.7 Obrazovka vyhodnocení kola

Tuto poslední obrazovku představuje třída EndOfRoundActivity. Na obrazovce (obrázek 3.10) se nachází velký nápis s jedním z textů VYHRÁL JSI, PROHRÁL JSI, nebo REMÍZA. Pod ním se zobrazuje skóre hráče v tomto kole a v závorce celkové skóre, pod tím skóre a celkové skóre protihráče. Následuje seznam slov s bodovým ohodnocením každého slova a zvýrazněním, které slovo mělo dvojnásobnou hodnotu. Seznam má dvě sekce, nejprve se zobrazují slova hráče, poté slova protihráče. Zakladateli hry se ve spodní části obrazovky zobrazí tlačítka na spuštění dalšího kola a ukončení hry.

Obrázek 3.10: Obrazovka vyhodnocení kola

Nejprve se v metodě onCreate() aktualizuje celkové skóre hráče i protihráče. Poté se nastaví všechna textová pole. Porovnáním skóre se vybere jeden ze tří nápisů a na příslušná místa se doplní čísla bodů.

Následně se vytvoří adaptér (třídy MyArrayAdapter) pro třídu ListView zobrazující seznamy slov. Nejprve se do něj vloží hlavička první sekce s nápisem *Má slova:*. Tu představuje třída již zmiňovaná ItemHeader. Poté se přidají všechna hráčova slova. Položku se slovem a s bodovým ohodnocením představuje třída ItemWordWithScore. Při vytváření objektu této třídy je do konstruktoru předán nejen text samotného slova, ale také příznak, zda je slovo za dvojnásobný počet bodů. Položka si pak sama spočítá své skóre podle počtu písmen slova a podle příznaku. V případě dvojnásobného počtu bodů je skóre ztučněno a zvýrazněno červenou barvou.

Po přidání všech hráčových slov je vytvořena nová sekce vložením další položky třídy ItemHeader s nápisem *Protihráčova slova:* a následně jsou vložena všechna slova protihráče.

Při kliknutí na tlačítko pro novou hru se spustí metoda startNewRound(), která pouze zobrazí obrazovku s nastavením dalšího kola. Protihráč čeká, dokud mu zakladatel hry nepošle parametry nového kola, a poté je na obou zařízeních spuštěno další kolo.

Kliknutím na tlačítko pro konec hry se zavolá metoda endGameAsServer(). Ta pouze pošle protihráči zprávu, že je hra ukončena. Po přijmutí této zprávy uzavře protihráč přes třídu BluetoothController metodou endGame() komunikační soket, ukončí komunikační vlákno a spustí úvodní obrazovku hry.

Jakmile dojde k uzavření soketu, je vyhozena v metodě read() výjimka. Ta je odchycena a voláním metody ConnectionLost je stejně jako u protihráče hra ukončena a je zobrazena úvodní obrazovka.

3.2.8 Textové řetězce

U prototypu jsem pro zjednodušení vkládal texty rovnou do kódu. Tento způsob má ale několik nevýhod.

Systém Android nabízí vkládání textových řetězců pomocí takzvaných String resources [19]. Zároveň se jedná o doporučené řešení. Využil jsem ho tedy a všechny texty v aplikaci převedl.

První výhoda String resources je, že se textové řetězce nachází ve vlastním souboru strings.xml. Je tak možné všechny textové úpravy provádět na jednom místě. Důležitější výhoda je však v tom, že nahrazením tohoto souboru jiným lze velmi jednoduše aplikaci přeložit. Není dokonce potřeba nic programovat. Soubor strings.xml je umístěn v adresáři values. Pro překlad například do angličtiny stačí vytvořit nový adresář s názvem values-en, překopírovat do něj soubor strings.xml a všechny texty v něm přeložit.

Další výhodou je, že je možné textům několika způsoby nastavovat vzhled a parametrizovat je, viz dále.

Existují tři typy String resources:

- String – XML zdroj poskytující prostý textový řetezec,
- String Array – XML zdroj poskytující pole textových řetězců,
- Quantity Strings (Plurals) – XML zdroj poskytující různé textové řetězce v závislosti na zadaném počtu

Typ String Array využívám pro získání seznamu slov, ze kterých se na vyžádání uživatele vygeneruje náhodné slovo pro hru.

Do souboru strings.xml se vloží String Array takto:

```
<string-array name="random_words">
    <item>LOKOMOTIVA</item>,
    <item>PARKOVIŠTĚ</item>,
    <item>NÁDRAŽÍ</item>,
    <item>INFORMACE</item>,
```

```
    <item>KAPRADÍ</item>,
    <item>KLOBÁSA</item>,
   ...
</string-array>
```

V kódu pak pole řetězců získám následovně:

```
String words[] = getResources().getStringArray(R.array.random_words);
```

Poslední zmíněný typ Quantity Strings je velmi užitečný, zvláště pro češtinu. Využívám ho na několika místech aplikace. Díky tomu jsem nemusel složitě vytvářet různé větve programu pro různé případy textového řetězce. Pro lepší představu přikládám ukázku.

V souboru strings.xml vytvořím tento String resource:

```
<plurals name="difference">
    <item quantity="one">o %d bod.</item>
    <item quantity="few">o %d body.</item>
    <item quantity="other">o %d bodů.</item>
</plurals>
```

Řetězec pak získám a použiji tímto způsobem.

```
differenceTextView.setText(getResources().getQuantityString(
     R.plurals.difference, difference, difference));
```

Tento textový řetězec se zobrazuje na obrazovce vyhodnocení kola. Říká uživateli o kolik bodů vyhrál, nebo prohrál. V češtině ale mohou nastat tři případy. Stejně jako v angličtině je potřeba rozlišit jednotné a množné číslo. V češtině se navíc liší skloňování počtu 2, 3, 4 od počtu 5 a více.

Jak je vidět v kódu výše, jsou tyto tři varianty reprezentovány XML elementem *item* s parametrem *quantity*. Pro případ počtu 2, 3 a 4 je potřeba jeho hodnotu nastavit na *few*. Pro jiné jazyky ještě existují možnosti *zero*, *two* a *many*.

Vybrání správné varianty zajistí metoda getQuantityString(), která jako druhý parametr očekává číslo značící počet, podle kterého se rozhodne, jakou variantu vybrat.

V ukázce se do metody předává i třetí parametr. Jedná se o již zmíněné parametrizování String resources. Pomocí značek, jako zde například *%d*, lze určit, kam do řetězce se parametr vloží a jakého bude typu. Zde písmeno *d* značí číslo. Do značky lze vložit i formátování, například kolik má mít číslo desetinných míst.

3.3 Grafický design aplikace

Při návrhu grafického zpracování aplikace bylo potřeba vycházet z toho, pro koho je hra určena. Jedná se o hru se slovy a písmeny, takže je možné vyloučit

předškolní děti. Jinak je hra určena pro kohokoliv, kdo má tento typ hry hrát a rád si ji zahraje ještě s někým dalším.

Soustředil jsem se tedy na to, aby hra nepůsobila nijak dětsky a neobsahovala příliš barevných prvků, animací a zvuků. Zvolil jsem pro to minimalistický design laděný v jedné barvě, který by byl vhodný pro širokou věkovou kategorii.

3.3.1 Barvy

Barevnou paletu jsem si našel na webových stránkách `color.adobe.com`. Zvolil jsem barvy v jednom odstínu modrozelené působící klidným dojmem. Jedná se o paletu s názvem Color Theme 8 od autora *carmit cohen*. Barvy s hexadecimálními kódy je možné vidět na obrázku 3.11.

Obrázek 3.11: Barevná paleta

Pro aplikace na systému Android se využívají v různých částech uživatelského rozhraní tři definované barvy [20]

- *colorPrimary*,
- *colorPrimaryDark*,
- *colorAccent*.

Barva *colorPrimary* se používá na záhlaví aplikace (anglicky app bar) a další základní prvky uživatelského rozhraní. V mé aplikaci jsem tuto barvu zvolil jako čtvrtou barvu z palety.

Barva *colorPrimaryDark* by měla být tmavší varianta primární barvy, která se od verze Androidu 5.0 používá pro notifikační lištu (anlicky status bar) a pro kontextuální záhlaví aplikace (anglicky Contextual action bar), které se zobrazí například při výběru položek. V mé aplikaci jsem tuto barvu zvolil jako druhou barvu z palety.

Třetí barva *colorAccent* se využívá pro prvky, jako jsou textová pole formulářů nebo zaškrtávací políčka, či radio tlačítka. Tato barva by měla být kontrastnější, protože zvýrazňuje aktivní prvky. Rozhodl jsem se vytvořit ještě šestou barvu, která bude doplňková k prostřední barvě palety. Náhled této barvy je zobrazen na obrázku 3.12.

Obrázek 3.12: Doplňková barva

3.3.2 Prvky na obrazovkách

Úvodní obrazovka je to první, co uživatel při spuštění aplikace uvidí. Měla by ho na první pohled zaujmout a představit mu samotnou hru. Do horní části obrazovky jsem tedy vložil velký nápis se jménem hry a jako pozadí jsem zvolil jemný přechod s motivem rozházených písmen v pravém dolním rohu, který hru charakterizuje. Vše je provedeno v barvách ze zvolené palety.

Rozházená písmena jsou vložena pomocí vektorového obrázku, který je možné bez ztráty kvality použít na jakkoliv velkém displeji.

Pro velké nadpisy jako je název hry nebo na obrazovce s výsledky text VYHRÁL JSI, PROHRÁL JSI, nebo REMÍZA jsem vybral jemný bezpatkový font se zaoblenými hranami jménem Bahamas (dostupný na `ceskefonty.cz/ceske-fonty/bahamas`).

Pro tlačítka jsem zvolil tenčí, jednodušší, ale přesto zajímavý font Advent Pro z databáze Google Fonts (dostupný na `fonts.google.com/specimen/Advent+Pro`).

Jako pozadí tlačítek jsem si vytvořil vlastní 9-Patch soubor. Jedná se o rastrový obrázek, který je možné roztáhnout do výšky i do šířky bez ztráty kvality. Toho je docíleno díky vyznačeným sekcím, které se mohou roztahovat. Obrázek může také obsahovat vyznačení určující umístění obsahu (anglicky padding box) [21]. Toto řešení jsem zvolil proto, abych mohl pod tlačítkem vytvořit stín.

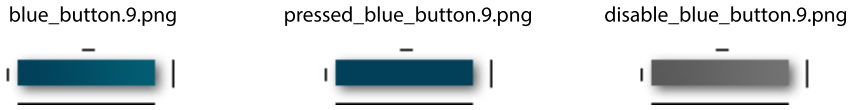
Pozadí tlačítek bylo potřeba vytvořit ve třech variantách. Tlačítko se může nacházet v různých stavech a vzhledem by na tyto stavy mělo reagovat a poskytnout tak uživateli zpětnou vazbu. V mém případě se tlačítka mohou nacházet ve stavu neaktivním (disable), stisknutém (pressed) a běžném (other). Systém Android na toto myslí a nabízí při vytváření šablony prvek selector, který změnu vzhledu tlačítka zařídí bez nutnosti cokoliv dělat programově. Takto vypadá XML šablona, která se použila jako pozadí tlačítek:

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_enabled="false"
    android:drawable="@drawable/disable_blue_button" />
    <item android:state_pressed="true"
    android:drawable="@drawable/pressed_blue_button" />
    <item android:drawable="@drawable/blue_button" />
</selector>
```

`disable_blue_button` představuje 9-Patch soubor jménem `disable_blue_button.9.png`. Pro neaktivní stav tlačítka jsem vzal původní modré přechodové pozadí a snížil sytost na 0, čím vzniklo šedé přechodové pozadí se stejnou světlostí.

Pro stisknuté tlačítko jsem pozadí upravil tak, že je vyplněné pouze nejtmavší barvou z původního barevného přechodu.

Všechny tři 9-Patch soubory i s vyznačenými oblastmi je možné vidět na obrázku 3.13.



Obrázek 3.13: Pozadí tlačítka v různých stavech

Co se týče pozadí dalších obrazovek, na herní obrazovce a obrazovce s nastavením jsem nechal pouze jemný modrý přechod a to hlavně kvůli čitelnosti. Stejně tak je to i u dialogových oken. Grafický prvek s rozházenými písmeny jsem vložil už jen na obrazovku s výsledky, kde je čitelnost zajištěna poloprůhledným bílým pozadím pod seznamem slov. Obrazovka je vidět na obrázku 3.10 v kapitole 3.2. Stejně tak i další obrazovky.

3.3.3 Ikona aplikace

Ikona aplikace vychází z motivu rozházených písmen z úvodní obrazovky, které charakterizují hru. Je tvořena stejným přechodovým pozadím a písmeny různých velikostí, barev z barevné palety a s různým natočením. Vybraná písmena v ikoně zároveň tvoří název hry – Slova ze slova.

Vývojové prostředí Android Studio nabízí vytvoření všech typů ikon. Pomocí nahrání obrázku hlavního motivu a obrázku pozadí vygeneruje všechny typy ve všech potřebných velikostech. Náhled všech typů z Android studia je zobrazen na obrázku 3.14.

Obrázek 3.14: Náhled ikony aplikace v Android Studiu

Kapitola 4

Testování a úpravy

4.1 Unit testy

Unit testy, někdy také jednotkové testy mají za cíl ověřit správnou funkčnost konkrétní programové jednotky – metody, třídy, komponenty apod. Programátoři vytváří jednotkové testy podle přístupu white-box s cílem ověřit (resp. potvrdit) požadované chování testované entity (jednotky) ve všech situacích. Cílem není nalezení chyb v testované entitě. Účelem jednotkových testů je pouze ověření funkčnosti konkrétní entity nezávisle na ostatních entitách či částech systému. [22]

Na operačním systému Android existují dva základní typy unit testů: [23]

- lokalní testy (local tests),
- testy na zařízení (instrumented tests).

Lokální testy jsou kompilovány pro lokální spuštění na virtuálním stroji Java (JVM), aby se minimalizovala doba provádění.

Druhý typ testů běží na zařízeních Android nebo emulátoru. Tyto testy mají přístup k informacím o prostředí, na kterém běží, jako je například třída Context.

Ve své práci implementuji oba typy unit testů a pokrývám jimi základní logiku aplikace. Příkladem testu na zařízení (instrumented test) může být otestování metody appendLetter ve třídě RoundActivity, kdy je potřeba tuto aktivitu na zařízení spustit:

```
public class RoundActivityTest {

    @Rule
    public ActivityTestRule<RoundActivity> rule = new
        ActivityTestRule<>(RoundActivity.class);
```

```
    @Test
    @UiThreadTest
    public void appendLetter() {
        RoundActivity activity = rule.getActivity();

        Button button = new Button(activity);
        button.setText("S");
        activity.appendLetter(button);
        TextView textView = activity.findViewById(R.id.newWord);

        Assert.assertEquals(textView.getText(), "S");

        button.setText("E");
        activity.appendLetter(button);

        Assert.assertEquals(textView.getText(), "SE");
    }
    ...
}
```

4.2 Uživatelské testování

Uživatelské testování jsem prováděl na svých kolezích a přátelích. Kvůli jednoduchosti hry nebylo potřeba vytvářet žádné testovací scénáře ani vstupní a výstupní dotazníky pro testery, jako se to dělá u velkých projektů a rozsáhlejších aplikací.

Má hra je také určená široké veřejnosti, každému, kdo má rád hry s písmeny. Tomuto popisu mí kolegové a přátelé odpovídali, takže nebyl problém testování provést na nich.

Testování probíhalo následovně. Připravil jsem pro dva testery telefony s nainstalovanou aplikací a nechal je si spolu zahrát několik kol hry. Přitom jsem sledoval jejich postup a dělal si případné poznámky. Poté jsem se testerů zeptal na jejich dojmy a postřehy a podstatné věci si poznamenal.

4.3 Výsledky uživatelského testování

Celkově uživatelé hru hodnotili jako hezky udělanou a zábavnou. Líbilo se jim grafické zpracování a jednoduchost uživatelského rozhraní.

Přesto testování ukázalo několik věcí, které bylo možné vylepšit. Nyní uvedu seznam nalezených problémů a v následující sekci 4.4 seznam úprav, jimiž jsem dané problémy vyřešil.

4.3.1 Problém 1

Při spuštění kola zakladatelem hry se stávalo, že druhý hráč nevěděl, kdy hra přesně začne, a díky nepozornosti se mohlo stát, že si spuštěné hry všiml o chvilku později. Ztratil tak několik vteřin a dostal se do nevýhody.

4.3.2 Problém 2

Problém se objevil ve chvíli, kdy vypršel čas kola. Tester právě dopisoval poslední slovo a ve chvíli, kdy chtěl kliknout na další písmeno, objevilo se mu dialogové okno se soupeřovými slovy. On pak místo na písmeno klikl na tlačítko Hotovo, které potvrzovalo soupeřova slova.

4.3.3 Problém 3

Původní návrh počítal s tím, že stačí, aby se tlačítka pro ukončení hry a pro zahájení nového kola zobrazovala jenom zakladateli hry. Během testovaní se ale ukázalo, že by bylo pro hráče pohodlnější, aby měli oba možnost hru ukončit nebo zahájit nové kolo.

Návrh vycházel z původní hry na papíře, kde se hráči domluví na slově, napíší si ho a začnou hrát. Stačilo tedy, aby tlačítko pro zahájení kola měl jenom jeden hráč, což bylo implementačně snadnější. Testování ale ukázalo, že někteří zakladatelé hry rádi zvolí slovo sami, aniž by ho řekli druhému hráči, což vnášelo do hry malý prvek překvapení.

Pokud chtěl slovo vymýšlet druhý hráč, museli si uživatelé vyměnit mobilní zařízení nebo hru ukončit a znovu ji spustit.

4.3.4 Problém 4

Nejedná se o nijak závažný problém. Občas se testerům stalo, že se při mazání slov v dialogovém okně s kontrolou protihráčových slov netrefili přesně na tlačítko s křížkem v pravé části, ale klikli více ke středu řádku.

4.3.5 Problém 5

Testováním se ukázalo, že jsou kroky pro změnu času kola nebo minimální délky slov na obrazovce s nastavením kola zbytečně složité. Pokud chtěl hráč tyto parametry upravit, musel kliknout do textového pole s číslem, smazat původní číslo a napsat nové.

4.4 Úpravy

4.4.1 Řešení problému 1: Odpočítávací obrazovka

Tento problém jsem vyřešil pomocí odpočítávací obrazovky. Po startu kola zakladatelem hry se oběma hráčům zobrazí obrazovka s třísekundovým odpočtem (obrázek 4.1).

Obrázek 4.1: Odpočítávací obrazovka

Díky odpočítavací obrazovce má druhý hráč dost času si všimnout, že už bylo kolo odstartováno, a připravit se na hru.

4.4.2 Řešení problému 2: Upozornění, že vypršel čas

Problému 2 jsem zamezil pomocí oznámení o vypršení času. Na obrazovce kola se objeví kruh s nápisem Čas vypršel, který zmizí po třech sekundách. Až poté se, v případě kontroly slov protihráčem, objeví dialogové okno se soupeřovými slovy. Oznámení je vidět na obrázku 4.2.

4.4.3 Řešení problému 3: Možnost ukončit hru a začít nové kolo pro druhého hráče

Na základě popisu problému 3 jsem upravil implementaci tak, aby i druhý hráč mohl zahájit nové kolo. Kdo tedy první klikne na tlačítko Začít nové kolo, získá roli serveru a objeví se mu obrazovka s nastavením nového kola. Druhý hráč se stává klientem a na obrazovce se mu objeví čekací dialogové okno.

Obrázek 4.2: Upozornění, že vypršel čas

4.4.4 Řešení problému 4: Neuznání slova kliknutím na celý řádek

Tento drobný problém jsem vyřešil tak, že jsem rozšířil klikatelnou oblast z tlačítka na celý řádek.

4.4.5 Řešení problému 5: Tlačítka + a -

Komplikované kroky popsané v problému 5 jsem se rozhodl zjednodušit pomocí přidání tlačítek pro zkrácení a prodloužení doby kola. Tlačítky + a - je tak možné měnit dobu kola o půl minuty.

Při změně doby kola se však číslo zobrazovalo v desetinném tvaru i pro celá čísla, například 4,0 minut. Tomu jsem zabránil pomocí třídy DecimalFormat, které jsem do konstruktoru vložil parametr `"#.#"`, jenž zajistí zobrazení celých čísel bez desetinné části. Tato třída zároveň zajistí použití správného oddělovače desetinných míst, pro češtinu je to čárka, pro angličtinu je to tečka.

Objevil se ale problém s typem textového pole. V XML šabloně jsem měl typ nastavený na *numberDecimal*. Ten však nepodporuje českou desetinnou čárku. Bylo proto nutné přidat čárku mezi povolené znaky pomocí `android:digits="0123456789.,"`.

Další problém nastal u starších klávesnic, které v režimu čísel nenabízejí desetinnou čárku, ale jen tečku. Proto jsem doimplemetoval i možnost vložit číslo s desetinnou tečkou.

Tlačítka + a - jsem doplnil i k poli s minimální délkou vytvořených slov. Zde se však jedná o celé číslo, takže se tu výše zmíněné problémy neobjevily. Tlačítka sníží nebo zvýší minimální délku o jedna.

Závěr

Cílem mé práce bylo vytvořit aplikaci na mobilní zařízení s operačním systémem Android pro dva hráče, která by implementovala hru na papíře.

Specifikace pravidel

Kompletně jsem specifikoval pravidla hry, včetně systému bodování pro více kol hry. Definoval jsem chování aplikace v různých případech a specifikoval funkční a nefunkční požadavky na aplikaci.

Analýza způsobů přímého propojení

Dále jsem provedl analýzu možných způsobů přímého propojení dvou mobilních zařízení. Analyzoval jsem technologie Bluetooth a Wi-Fi. Vytvořil jsem prototypy těchto způsobů propojení a provedl jejich porovnání. Na základě tohoto porovnání jsem vybral vhodnější způsob propojení zařízení pro hru, jímž byla technologie Bluetooh.

Návrh a implementace

Následně jsem navrhl a implementoval aplikaci pro operační systém Android. Vytvořil jsem wireframes, ve kterých jsem rozvrhl jednotlivé obrazovky a prvky na nich. Provedl jsem návrh tříd a poté aplikaci naprogramoval. Také jsem vytvořil grafický design aplikace, včetně grafických prvků jako jsou pozadí tlačítek, pozadí obrazovek nebo ikona aplikace.

Testování

Pro aplikaci jsem vytvořil unit testy a provedl uživatelské testování. Pro všechny zjištěné problémy z testování jsem navrhl úpravy, které příslušný problém eliminují. Tyto úpravy jsem následně zanesl do aplikace.

Literatura

[1] MLEJNEK, Jiří. *Softwarové inženýrství I – 3. Analýza a sběr požadavků*. [přednáška]. Praha: ČVUT, 2018, Podklady dostupné z: `https://edux.fit.cvut.cz/courses/BI-SI1/_media/lectures/03/03.prednaska.pdf`

[2] Pravidla přípustnosti slov ČAS. *Česká asociace Scrabble* [online]. Dostupné z: `http://scrabble.hrejsi.cz/pravidla/pripustnost-slov`

[3] Near field communication overview. *Android Developers*. [online]. [cit. 2018-11-29]. Dostupné z: `https://developer.android.com/guide/topics/connectivity/nfc/`

[4] BRAY, Jennifer a Charles F STURMAN. *Bluetooth: connect without cables*. 2nd ed. Upper Saddle River, N.J.: Prentice Hall, c2002. ISBN 01-306-6106-6.

[5] Member Directory. *Bluetooth Technology Website*. [online]. [cit. 2018-12-04]. Dostupné z: `https://www.bluetooth.com/develop-with-bluetooth/join/member-directory`

[6] Bluetooth overview. *Android Developers*. [online]. [cit. 2018-12-04]. Dostupné z: `https://developer.android.com/guide/topics/connectivity/bluetooth`

[7] HORSKÝ, Radek. *Bezdrátové sítě Wi-Fi v rekordním čase*. Praha: Grada, 2006. V rekordním čase. ISBN 80-247-1790-5.

[8] Wi-Fi peer-to-peer overview. *Android Developers*. [online]. [cit. 2018-12-10]. Dostupné z: `https://developer.android.com/guide/topics/connectivity/wifip2p`

[9] How fast is Wi-Fi Direct?. *Wi-Fi Alliance* [online]. Copyright © 2018 Wi [cit. 17.12.2018]. Dostupné z: `https://www.wi-fi.org/knowledge-center/faq/how-fast-is-wi-fi-direct`

[10] Wi-Fi Direct vs. Bluetooth 4.0: A Battle for Supremacy. *PCWorld* [online]. Copyright © 2018 IDG Communications, Inc. [cit. 17.12.2018]. Dostupné z: https://www.pcworld.com/article/208778/Wi_Fi_Direct_vs_Bluetooth_4_0_A_Battle_for_Supremacy.html

[11] How far does a Wi-Fi Direct connection travel?. *Wi-Fi Alliance* [online]. Copyright © 2018 Wi [cit. 17.12.2018]. Dostupné z: https://www.wi-fi.org/knowledge-center/faq/how-far-does-a-wi-fi-direct-connection-travel

[12] Wi-Fi Alliance Introduces Next Generation of Wi-Fi Security. *Wi-Fi Alliance* [online]. Copyright © 2018 Wi [cit. 17.12.2018]. Dostupné z: https://www.wi-fi.org/news-events/newsroom/wi-fi-alliance-introduces-next-generation-of-wi-fi-security

[13] Can a device simultaneously connect to a regular Wi-Fi network and a group of Wi-Fi Direct-certified devices at the same time?. *Wi-Fi Alliance* [online]. Copyright © 2018 Wi [cit. 18.12.2018]. Dostupné z: https://www.wi-fi.org/knowledge-center/faq/can-a-device-simultaneously-connect-to-a-regular-wi-fi-network-and-a-group-of

[14] HOMOLKA, Jakub. *ČVUT navigátor - návrh multiplatformního mobilního kli- enta*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

[15] Gilfelt, Jeff. *Android SQLiteAssetHelper* [online]. Copyright © 2011 readyState Software Ltd [cit. 18.12.2018]. Dostupné z: https://github.com/jgilfelt/android-sqlite-asset-helper

[16] Import and Use External Database in Android. *Java Helps* [online]. [cit. 18.12.2018]. Dostupné z: https://www.javahelps.com/2015/04/import-and-use-external-database-in.html

[17] Fragments *Android Developers* [online]. [cit. 18.12.2018]. Dostupné z: https://developer.android.com/guide/components/fragments#Lifecycle

[18] HAVRYLUK, Michal. *Programování pro operační systém Android – 7. Loader, Adapter a ListView*. [přednáška]. Praha: ČVUT, 3. dubna 2018. Podklady dostupné z: https://courses.fit.cvut.cz/BI-AND/media/lectures/07.pdf

[19] String resources. *Android Developers*. [online]. [cit. 2018-12-20]. Dostupné z: https://developer.android.com/guide/topics/resources/string-resource

[20] Styles and Themes. *Android Developers*. [online]. [cit. 2018-12-22]. Dostupné z: `https://developer.android.com/guide/topics/ui/look-and-feel/themes`

[21] Drawables overview. *Android Developers*. [online]. [cit. 2018-12-22]. Dostupné z: `https://developer.android.com/guide/topics/graphics/drawables#nine-patch`

[22] HAVELKA, Arnošt a Rudolf PECINOVSKÝ. *JUnit 5: jednotkové testování na platformě Java*. Praha: Grada Publishing, 2018. Knihovna programátora (Grada). ISBN 978-802-7107-339.

[23] Build effective unit tests. *Android Developers*. [online]. [cit. 2019-01-08]. Dostupné z: `https://developer.android.com/training/testing/unit-testing/`

Příloha A

Seznam použitých zkratek

AES Advanced Encryption Standard

API Application Programming Interface

HTML Hypertext Markup Language

IP Internet Protocol

JVM Java Virtual Machine

MAC Media Access Control

OS Operační systém

P2P Peer to peer

RFCOMM Radio Frequency Communication

SDK Software development kit

SDP Service Discovery Protocol

UI User interface

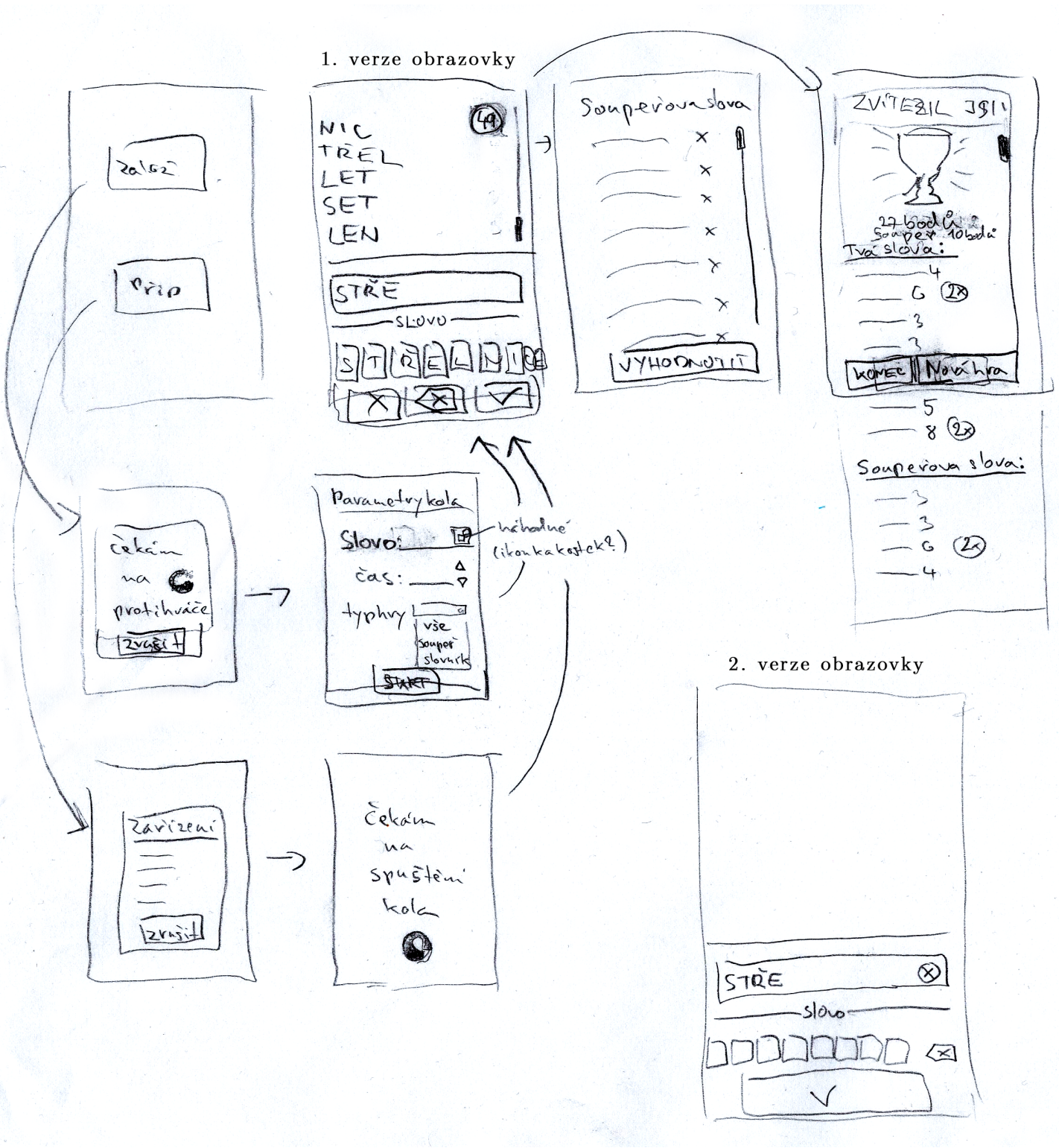
UUID Universally Unique Identifier

WPA Wi-Fi Protected Access

XML Extensible Markup Language

Příloha B

Papírové wireframes



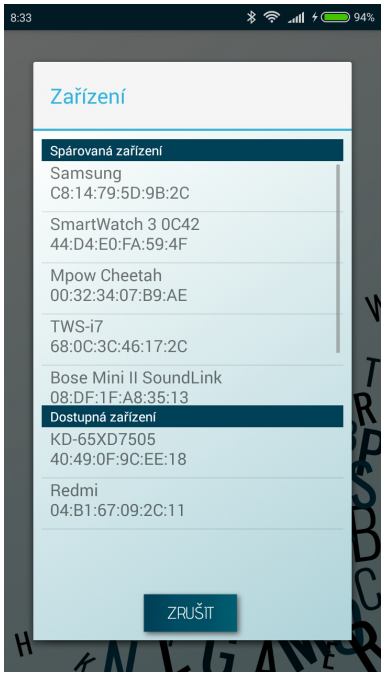
Obrázek B.1: Papírové wireframes

Příloha C

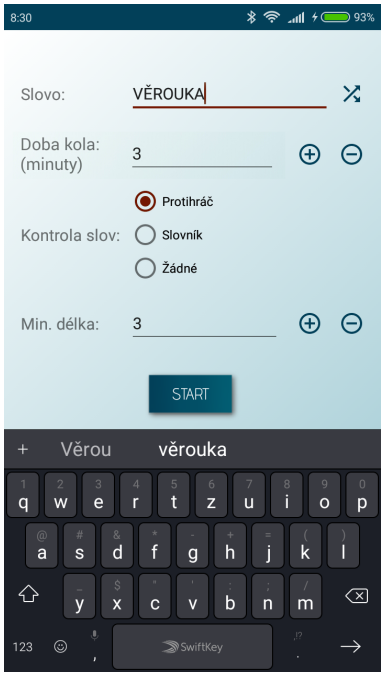
Obrazovky aplikace

Obrázek C.1: Úvodní obrazovka hry

Obrázek C.2: Čekání zakladatele hry na protihráče

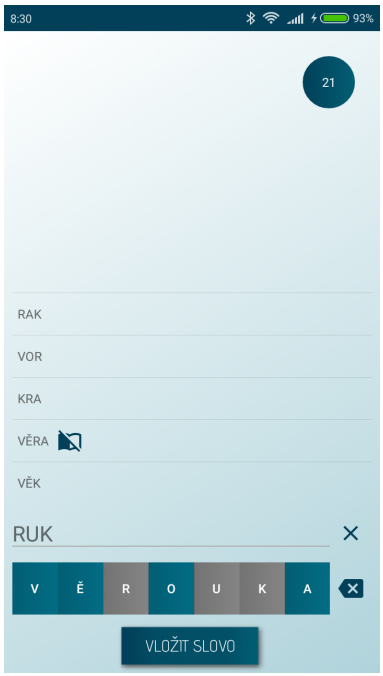


Obrázek C.3: Připojení se k protihráči

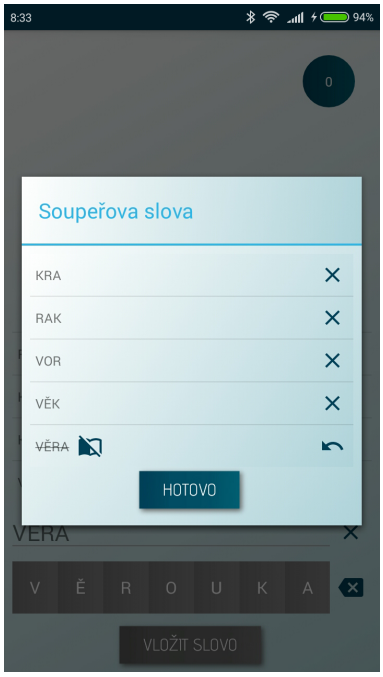


Obrázek C.4: Obrazovka nastavení kola

Obrázek C.5: Odpočet kola před zahájením hry



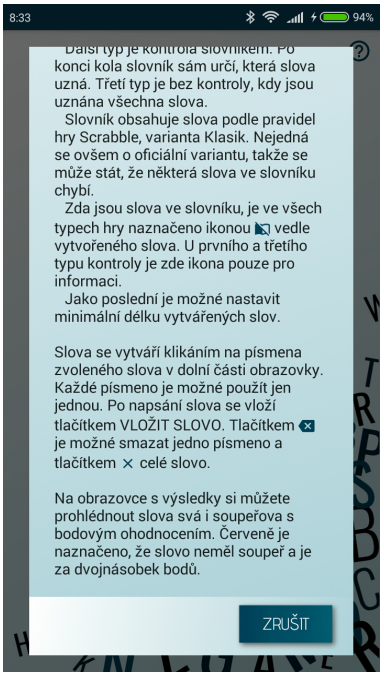
Obrázek C.6: Obrazovka, kde probíhá hra



Obrázek C.7: Kontrola slov protihráčem

Obrázek C.8: Upozornění na konec kola

Obrázek C.9: Obrazovka s výsledky



Obrázek C.10: Obrazovka s nápovědou

Příloha D

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `slova-ze-slova.apk` ..... instalační soubor aplikace
- `bluetooth-prototyp.apk` ..... instalační soubor Bluetooth prototypu
- `wifi-prototyp.apk` ..... instalační soubor Wi-Fi prototypu
- `src` ..... adresář se zdrojovými kódy – projekty do Android Studia
  - `SlovaZeSlova` ..... hlavní aplikace
  - `SlovoBluetoothPrototyp` ..... Bluetooth prototyp
  - `SlovoWiFiPrototyp` ..... Wi-Fi prototyp
- `text` ..... text práce
  - `DP_Homolka_Jakub_2018.pdf` ..... text práce ve formátu PDF
  - `DP_Homolka_Jakub_2018.tex` zdrojová forma práce ve formátu LaTeX
  - `FITthesis.cls` ..... LaTeX třída definující šablonu závěrečných prací
  - `cvut-logo-bw.pdf` ..... logo ČVUT použité v šabloně
- `tesovani`
  - `bugreport-bt.zip` ..... výstup z měření Bluetooth prototypu
  - `bugreport-wifi.zip` ..... výstup z měření Wi-Fi prototypu