Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Sociálně muziko-terapeutické aplikace pro hru na klavír

2020

Daniel Vágner

Vedoucí práce: Mgr. Jiří Zacpal, Ph.D.

Studijní obor: Informatika, prezenční forma

Bibliografické údaje

Autor:	Daniel Vágner
Název práce:	Sociálně muziko-terapeutické aplikace pro hru na klavír
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2020
Studijní obor:	Informatika, prezenční forma
Vedoucí práce:	Mgr. Jiří Zacpal, Ph.D.
Počet stran:	34
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliograhic info

Author:	Daniel Vágner
Title:	Therapeutical piano applications
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2020
Study field:	Computer Science, full-time form
Supervisor:	Mgr. Jiří Zacpal, Ph.D.
Page count:	34
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Bakalářská práce se zabývá tvorbou mobilní aplikace simulující klavír. Aplikace "Klavír" bude sloužit klientům centra sociálních služeb Klíč a pomůže jim ve vnímání hudby. Aplikace je vytvořena pro operační sytém Android. Práce také obsahuje uživatelskou příručku.

Synopsis

The objective of this bachelor's thesis was to create a mobile application behaving as a piano. The "Piano" application will be used in a social services center where it'll help clients to percieve apects of music. The application is created for Android operating system. This thesis also contains user guide.

Klíčová slova: C#; Xamarin.Forms; Android; hudba; klavír

Keywords: C#; Xamarin.Forms; Android; music; piano

Speciální poděkování patří doktoru Zacpalovi a zaměstnancům Klíče za jejich čas při konzultování aplikace v průběhu její tvorby.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

1 Cíle práce **8**
- 1.1 Motivace . . . 8
- 1.2 Požadavky . . . 8
- 1.3 Podobné aplikace . . . 9
  - 1.3.1 Perfect Piano . . . 9
  - 1.3.2 Synthesia . . . 9
- 1.4 Klíčové prvky aplikace . . . 10
- 1.5 Struktura aplikace . . . 11
  - 1.5.1 Volné hraní . . . 11
  - 1.5.2 Hraní dle předlohy . . . 11
  - 1.5.3 Skladatel . . . 11
  - 1.5.4 Správa skladeb . . . 11
  - 1.5.5 Tutoriál . . . 11

2 Použité technologie **11**
- 2.1 .NET . . . 12
- 2.2 C# . . . 12
- 2.3 Xamarin . . . 12
- 2.4 Xamarin.Forms . . . 12
- 2.5 XAML . . . 13
- 2.6 NuGet . . . 13
- 2.7 LINQ . . . 13
- 2.8 Firebase . . . 13
  - 2.8.1 Firebase Realtime Database . . . 13
  - 2.8.2 Firebase Cloud Storage . . . 14
- 2.9 MIDI . . . 14

3 Programátorská dokumentace **14**
- 3.1 Popis tříd . . . 14
  - 3.1.1 Třída AudioRender . . . 14
  - 3.1.2 Třída Score . . . 15
  - 3.1.3 Třída Song . . . 15
  - 3.1.4 Třída MyString . . . 15
  - 3.1.5 Třída MyTuple . . . 15
  - 3.1.6 Třída FirebaseHelper . . . 16
  - 3.1.7 Třída FirebaseStorageHelper . . . 17
- 3.2 Stránky . . . 19
- 3.3 Důležité algoritmy . . . 20
  - 3.3.1 Metoda FlattenList . . . 20
  - 3.3.2 Přehrávání skladby . . . 22
  - 3.3.3 Skladatel . . . 22

4 Uživatelská příručka **23**
4.1 Instalace 23
4.2 Spuštění aplikace 24
4.3 Volné hraní 25
4.4 Hraní dle předlohy 25
4.5 Skladatel 27
4.6 Správa skladeb 29

Závěr **31**

Conclusions **32**

5 Obsah přiloženého CD/DVD **33**

Literatura **34**

Seznam obrázků

1 Perfect Piano . . . 9
2 Synthesia . . . 10
3 Diagram stránek aplikace . . . 20
4 Rozdíl klientské vs. administrátorské menu . . . 24
5 Sekce volné hraní . . . 25
6 Výběr skladby . . . 25
7 Obrazovka přehrávání skladby . . . 26
8 Dokončení přehrávání skladby a uložení skóre . . . 27
9 Výběr tónu ve skladateli . . . 28
10 Úprava zadané sekvence . . . 29
11 Správa skóre skladby . . . 30

Seznam tabulek

Seznam vět

1 Definice (Enharmonická záměna) . . . 22

Seznam zdrojových kódů

1 Třída AudioRender . . . 15
2 Inicializace třídy Score . . . 15
3 Získání seznamu skladeb . . . 18
4 Odeslání souboru na úložiště . . . 19
5 Příkaz zaručující správnost chování kláves . . . 19
6 Navigace mezi stránkami . . . 20
7 Metoda FlattenList . . . 21
8 Algoritmy k hraní dle předlohy . . . 22
9 Enharmonická záměna . . . 23

1 Cíle práce

Cílem práce bylo vytvořit aplikaci na mobilní telefony a tablety simulující hru na klavír. Uživatel může hrát na klavír buďto volně nebo podle předlohy skladeb, které jsou obsaženy v základu. Skladby mohou také uživatelé nebo administrátoři tvořit a poté jimi aplikaci obohatit.
Každý klient vidí v průběhu hraní skladby své průběžné skóre, při dokončení přehrávání je mu umožněno si skóre uložit. Všechny záznamy o výsledcích jsou viditelné u jednotlivých skladeb, administrátor je může spravovat.
Aplikace je rozdělena do dvou sekcí - uživatelské, kde se není potřeba přihlašovat; a administrátorské, ke které je přístup podmíněn heslem.

1.1 Motivace

Jakožto aktivní muzikant se kolem hudby pohybuji takřka denně. Přestože klavír není nástrojem, o kterém bych měl větší než teoretické znalosti, velmi mě zaujala nabídka udělat aplikaci, která by jeho chování simulovala a sloužila zároveň jako výuková. Nejen, že tímto způsobem poznám detailněji další hudební nástroj, ale také při tvorbě aplikace a databáze skladeb využiji své povědomí o hudební teorii. Klavír navíc, ostatně jako všechny hudební nástroje, není levná záležitost, proto se touto formou může svět hudby zpřístupnit pro lidi, kteří by o něm jinak mohli jen snít.

1.2 Požadavky

Na práci jsou kladeny následující požadavky:

1. Aplikace musí obsahovat novou funkcionalitu s podporou přidání vlastní skladby v obvyklém formátu (např. mp3) nebo přímo v midi formátu.

2. Klient bude schopen hrát na virtuálních klávesách jednotlivé své skladby, které si do aplikace dříve nahrál přímo on nebo skrze asistenta.

3. Klient vidí skóre své úspěšnosti, za které získává motivační body, toto se ukládá do účtu klienta.

4. Podpora klientských a asistenčních účtu (viz společné pokyny) s podporou individuálního nastavení komunikace dle potřeb klienta.

5. GUI bude disponovat zřetelnými prvky, intuitivním ovládání s podporou gest. GUI bude také přizpůsobitelné potřebám klienta.

Aplikace musí být nativně spustitelná bez závislosti na internetovém připojení a použitelná alespoň na mobilní platformě OS Android a bude volně dostupná případně i jako open source.

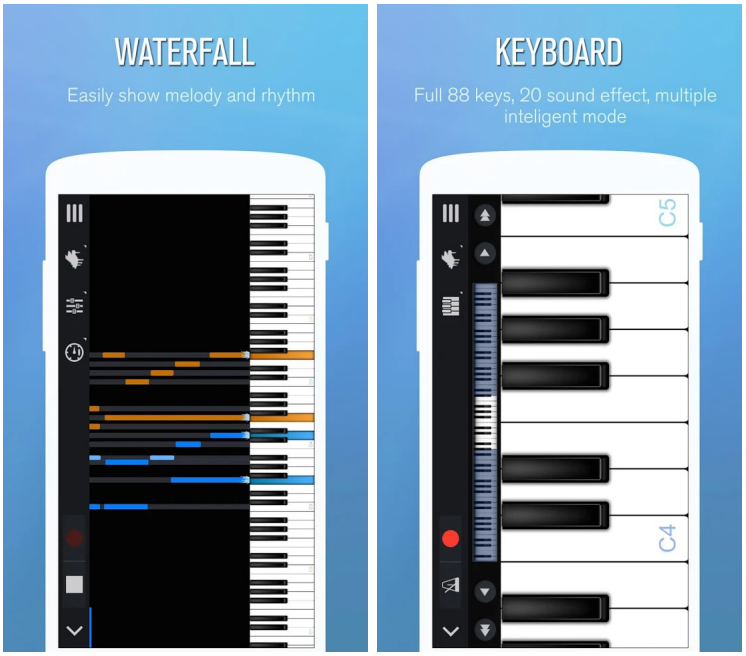
1.3 Podobné aplikace

Pro operační systém Android se v Google Play Store vyskytuje spousta podobných aplikací, přičemž jako ukázky jsem vybral tyto:

1.3.1 Perfect Piano

https://play.google.com/store/apps/details?id=com.gamestar.perfectpiano

Tato aplikace simuluje skutečné piano, avšak, jak jde vidět už ze screenshotů v Play Store, klávesy piana se na obrazovku telefonu nevejdou všechny. Bylo by tedy nutné je při hraní posouvat nebo disponovat více obrazovkami. Také zde není výuková hra nebo možnost naučit se skladbu z databáze ať už vlastní, či online. Plusem aplikace je notový popis jednotlivých kláves.

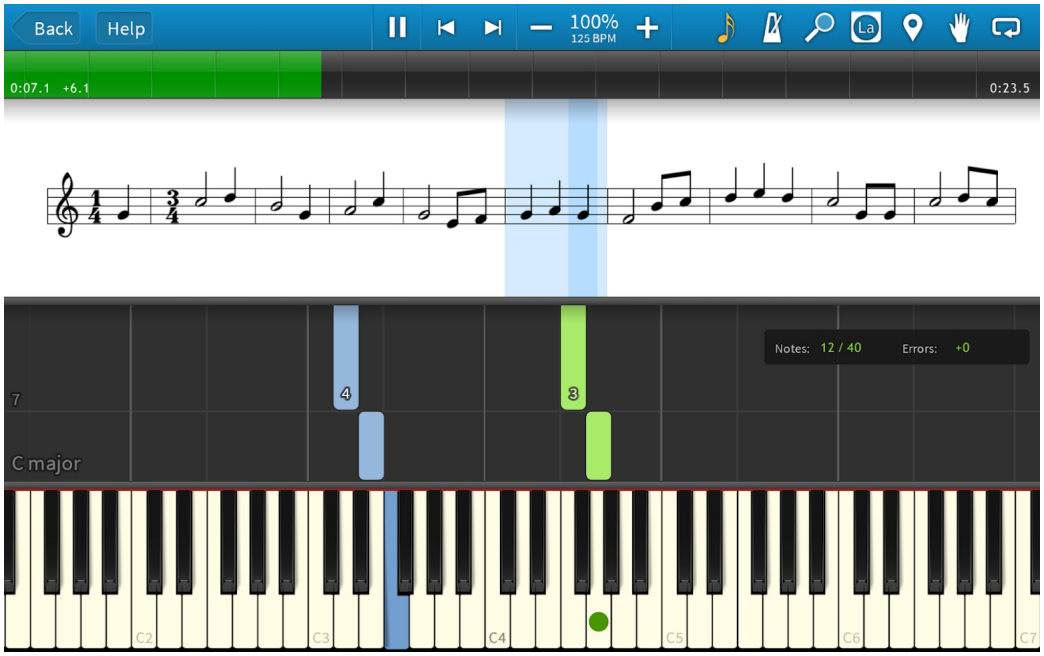


Obrázek 1: Perfect Piano

1.3.2 Synthesia

https://play.google.com/store/apps/details?id=com.synthesia.synthesia

Za přednost aplikace považuji její velmi profesionálně vypadající zpracování. Uživatel si může nastavit rychlost skladby buď v procentech nebo v BPM. Aplikace ukazuje, jak rozložit ruce na klávesách, umí navíc zobrazovat jak noty skladby, tak barevně označovat klávesy. K aplikaci také lze připojit digitální klávesy. Mezi nedostatky řadím absenci historii skóre skladeb.



Obrázek 2: Synthesia

1.4 Klíčové prvky aplikace

Aplikací simulujících klavír či piano lze najít desítky, avšak málo z nich je vhodných k nasazení do center, která se starají o mentálně postižené osoby. Některé aplikace postrádají různé úrovně nastavitelnosti, jiné naopak nemají motivaci v podobě skóre a možnosti se zlepšovat. Většina z nich navíc obsahuje reklamy, což není pro kýžené použití aplikace žádoucí. Pro přiblížení se aplikaci použitelné v centru sociálních služeb by tedy aplikace měla splňovat několik bodů.

- **Přehlednost** Aplikace by měla být co nejjednodušší k použití, vzhledem k tomu, že ji budou používat lidé s různými stupni mentálních retardací.
- **Stupně obtížnosti** Pokud má aplikace splňovat výukovou roli, nelze jedince, kteří nemají žádnou zkušenost s hudebními nástroji, hodit takzvaně "do vody". Pokud by hned klient měl pouze jednu úroveň obtížnosti, ztrácel by motivaci se postupně posouvat. Navíc, neměl by-li dostatečnou zručnost hned na začátku, mohlo by jej to od používání aplikace odradit.
- **Lidové skladby** Jakožto aktivní hudebník vím, že při začátcích s hudebním nástrojem se člověk snaží naučit se skladby, které zná. Z tohoto důvodu jsem zvolil přiložit do aplikace lidové skladby, které v Česku jsou známé a klienti tak budou moci si při hraní také zpívat.
- **Vlastní tvorba** Žádná databáze skladeb není dostatečně velká, aby nemohla být ještě větší. Klienti mohou tvořit vlastní skladby a ty pak jsou k dispozici všem dalším. Stejně tak se dá databáze doplnit o písně například ze zpěvníku, stačí jen přepsat jejich notový zápis.

1.5 Struktura aplikace

Aplikace je rozdělena do pěti částí, každá část se liší účelem i způsobem použití. Čtyři z těchto pěti částí jsou volně přístupné, pouze správa skladeb je dostupná pouze administrátorům. Na následujících stranách následuje stručný popis jednotlivých částí aplikace.

1.5.1 Volné hraní

V této části se uživatelé mohou seznámit se samotným fungováním virtuálního klavíru. K dispozici mají jednu oktávu kláves rozložených stejně jako na klavíru, přičemž je zde také přítomen přepínač oktáv, díky kterému uživatelé nejsou omezeni pouze na například první oktávu.

1.5.2 Hraní dle předlohy

Tato část má výukovou funkci. Uživatel si vybere skladbu, kterou se chce naučit a může hrát. Uživatel si může zvolit mezi 3 stupni obtížnosti a začít hrát. V průběhu hraní se mu zobrazuje průběžné skóre, se kterým, je-li spokojen, se může pochlubit a uložit své jméno do záznamu odehraných skóre.

1.5.3 Skladatel

Jsou-li uživatelé hudebně zdatnější, mohou si zkomponovat vlastní skladby nebo alespoň znělé sekvence tónů, které pak zpřístupní pro hraní všem dalším, kteří mají k dispozici tuto aplikaci. Skladatel mohou využívat také pracovníci centra a s jeho pomocí přepsat další skladby například ze zpěvníku.

1.5.4 Správa skladeb

Tato část je přístupná pouze administrátorům. Slouží především k úpravě záznamů odehraných skladeb, mimo jiné ale mohou být smazány skladby vytvořené uživateli, které se například příliš nepovedly. Administrátoři tak mohou řešit například překlepy ve jménech u skóre, nebo přidávat záznamy, které byly například odehrány na zařízení bez přístupu na internet.

1.5.5 Tutoriál

Zkrácená verze uživatelské příručky přímo v aplikaci by měla objasnit nejasnosti používání aplikace.

2 Použité technologie

Mobilní aplikace se dají vyvíjet v celé škále programovacích jazyků. Pro Android se může jednat o nativní jazyky jako Java nebo v současné době stále populárnější Kotlin, pro iOS pak jazyk Swift, který nahradil nedostačující Objective-C. Volba

operačního systému pro mě byla jednoduchá ze dvou důvodů. Za prvé jsem sám uživatelem Androidu a v této platformě nacházím zalíbení, zadruhé samotné centrum Klíč disponuje tablety právě s operačním systémem Android. Co se volby jazyku týče, nejblíže mám pak k jazyku C# od Microsoftu, proto jsem si vybral jazyk Xamarin. Konkrétněji Xamarin.Forms, který umožňuje snadnou implementaci multiplatformních aplikací.

2.1 .NET

.NET je open-source vývojářská platforma vytvořená společností Microsoft. .NET poskytuje jazyky a knihovny pro vývoj webových, mobilních nebo desktopových aplikací. K vývoji desktopových aplikací slouží jazyky C#, F# nebo Visual Basic, mobilní aplikace pak lze vyvíjet v jazyku Xamarin, projektu společnosti Mono. [1]

2.2 C#

C# je objektově-orientovaný, typově bezpečný jazyk vyvinutý společností Microsoft. Jazyk lze využít na tvorbu robustních aplikací pomocí rozhraní .NET Framework. Dá se využít k tvorbě klientských aplikací systému Windows, webových služeb nebo mobilního softwaru. [2]

2.3 Xamarin

Xamarin je open-source platforma od dceřinné společnosti firmy Microsoft rozšiřující platformu .NET. Slouží k vývoji mobilních aplikací pro Android a iOS pomocí jazyku C# a XAML. Lze využít všechny dostupné knihovny .NET. Kód Xamarinu se kompiluje do nativních kódů jednotlivých systémů a je tak možné tvořit aplikace, které vypadají a chovají se jako nativní. [3]

2.4 Xamarin.Forms

Xamarin.Forms je odvětví jazyka Xamarin pro vývoj mobilních aplikací. Oproti standardnímu Xamarinu se liší v tom, že pro vývoj na různé operační systémy je možno využít jednoho kódu, který se následně přeloží na kýžené platformy. Přitom zachovává prvky specifické pro jednotlivé systémy jako jsou animace, navigace, poskytované služby apod. Vzhled stránek se dá psát nejen v XAML, ale také přímo v kódu C#. [4]

Xamarin.Forms jsem zvolil, jelikož mi tento způsob tvorby aplikací přijde progresivnější v ohledu, že stačí napsat jeden kód a poté jej jen přeložit na kýžený operační systém. Odpadá tak nutnost psát kód pro každou platformu zvlášť. Xamarin.Forms navíc nabízí spoustu pluginů, tzv. NuGetů, kterými lze projekt obohatit.

2.5 XAML

XAML je značkovací jazyk, podobně jako HTML. Zjednodušuje vytváření uživatelských rozhraní aplikací oddělením definicí UI od kódu programu. Využití nachází při tvorbě formulářových aplikací pro Windows nebo mobilních aplikací. [5]

Možnost rozdělit si návrh vzhledu stránky od samotného kódu pro mě byl hlavním důvodem, proč jsem volil tuto metodu práce na grafické stránce. Kód pak zůstal lépe čitelný a nenabobtnal do několikasetřádkových souborů, ve kterých by se podstatně hůře orientovalo.

2.6 NuGet

Balíček NuGet je soubor, který obsahuje zkompilovaný kód, jako knihovnu DLL, a další soubory související s kódem, které umožňují využívat tento kód. [6]

Ve své práci používám několik NuGetů, například prohlížeč souborů CrossFilePicker nebo NuGet zpřístupňující API cloudového úložiště a databáze Firebase.

2.7 LINQ

Language Integrated Query, zkráceně LINQ, je dotazovacích funkcí jazyka C#. Pomocí LINQ se tak může přistupovat k objektům nebo kolekcím jako k databázi a vyvolávat dotazy identické těm z SQL. Lze tak zjednodušit některé příkazy, například foreach nebo filtrování. [7]

2.8 Firebase

Firebase je platforma společnosti Google, která poskytuje celou škálu produktů, které lze využít k synchronizaci dat mezi webovými službami a mobilními operačními systémy iOS nebo Android. Ve své bakalářské práce využívám 2 produkty Firebase a to konkrétně Firebase Realtime Database a Firebase Cloud Storage.

2.8.1 Firebase Realtime Database

Tato databáze umožňuje uložení a synchronizaci dat pomocí cloudového úložiště řešené přes NoSQL. Data jsou uložena ve formátu JSON a data jsou synchronizována v reálném čase mezi všechny připojené klienty. Data synchronizovaná do zařízení zůstávají k dispozici i po ztrátě internetového připojení.[8] Každý záznam je odesílán jako objekt, jehož elementy jsou uloženy ve formátu key-value, například chceme-li odeslat záznam se jménem Jakub, je potřeba vytvořit objekt, který v sobě bude mít obsažen element jméno, jehož hodnota bude "Jakub".

Databázi využívám k uchovávání záznamů o skladbách, jako například jméno nebo maximální skóre, dále k záznamům odehraných skóre skladeb, které mají vazbu na skladbu, u každé skladby se tak dá dohledat, kteří hráči dosáhli jakých skóre při jaké obtížnosti.

2.8.2 Firebase Cloud Storage

Firebase Cloud Storage je jednoduše přístupné a bezpečné cloudové úložiště. Ve své aplikaci jej využívám k uložení skladeb, které si napíší samotní uživatelé. Tyto skladby jsou pak v nabídce hraní dle předlohy. V NuGetu pro Xamarin.Forms je potřeba při odeslání nejdříve převést sekvenci/soubor do Streamu a pak jej odeslat, při stahování se jedná o opačný postup.

2.9 MIDI

MIDI je zkratka pro Musical Instrument Digital Interface, což je technický standard, který popisuje komunikační protokol mezi hudebními nástroji a počítačem. Využití nachází mezi muzikanty, producenty, DJ po celém světě. [9]

Tento formát jsem zvolil z několika důvodů. Hudební soubory například ve formátu MP3 jsou mnohem větší, zvuk je často zkreslen a člověk musí najít opravdu kvalitní databázi, kde není před samotným zvukem noty kupříkladu půlvteřinová prodleva. MIDI naopak lze jednoduše vygenerovat spoustou hudebních programů, ať už se jedná o Guitar Pro, Reaper nebo Cubase, s jistotou, že zaznamenán bude pouze konkrétní tón přehratelný nezávisle na operačním systému.

3 Programátorská dokumentace

V následující kapitole popíši nejdůležitější algoritmy, které využívám ve své práci, tak, aby se v případě potřeby dala nadále rozvíjet. Dané problémy budou popsány z programátorského pohledu.

3.1 Popis tříd

Většina tříd jsou pouze pomocné třídy k odesílání dat na Firebase, jejich definice jsou tedy často triviální.

3.1.1 Třída AudioRender

Třída slouží k přehrávání tónů klavíru. K přehrání není potřeba žádný externí přehrávač, použije se ten systémový díky využití Dependendy Service. Tato služba umožňuje využívat nativní funkce platformy, jako je například zde zmíněný mediální přehrávač. Implementace vychází z návodu YouTube kanálu Soft.Web, jehož kód lze nalézt také na přiloženém úložišti GitHub. [1]

[1] https://github.com/zekiriabd/Xamarin/tree/master/Mp3Demo/Mp3Demo/Mp3Demo.Android

```
1  public class AudioRender : IAudioService
2     {
3        public void PlayAudioFile(string fileName)
4        {
5           var player = new MediaPlayer();
6           var file = global::Android.App.Application.Context.Assets.
                OpenFd(fileName);
7           player.SetDataSource(file.FileDescriptor, file.StartOffset,
                 file.Length);
8           player.Prepared += (s, e) => { player.Start(); };
9           player.Prepare();
10          player.Completion += (s, e) => { player.Release(); };
11       }
12    }
```

Zdrojový kód 1: Třída AudioRender

3.1.2 Třída Score

Instance této třídy nesou informace o skóre, konkrétněji jméno hráče, obtížnost, výsledek a databázové ID skladby. Objekt je možno vytvořit jak prázdný, tak zadáním všech potřebných údajů.

```
1  public Score(string userName, string songID, int result, string
       difficulty)
```

Zdrojový kód 2: Inicializace třídy Score

3.1.3 Třída Song

Objekty této třídy nesou informace o názvu skladby a nejvyšším dosažitelném skóre. Využívám ji k práci s informacemi o skladbě a k synchronizaci do databáze Firebase.

3.1.4 Třída MyString

Pomocná třída k odesílání do Firebase Realtime Database. Třída je nutná z toho důvodu, že databáze Firebase přijímá objekty, nikoliv jen samotné sekvence datového typu string.

3.1.5 Třída MyTuple

Analogie typu Tuple, který však neumožňuje modifikovat data v něm uchovaná. Tuto třídu používám ke komunikaci mezi stránkami, přičemž data se nejen přenáší, ale také dle potřeby modifikují.

3.1.6 Třída FirebaseHelper

Účelem této třídy je odesílat data do databáze Firebase. K propojení s databází slouží instance třídy Firebase Client, kterou poskytuje NuGet FirebaseStorage.net. V databázi používám 3 tabulky: Songs, kde jsou uloženy informace o skladbách; Scores, uchovávající záznamy skóre odehraných skladeb; a UserSongList, kde se nachází skladby vytvořené uživateli a které jsou dostupné online. S využitím návodu od C# Corner[2] jsem implementoval následující metody pro práci s databází.

- **static async Task<List<Score>> GetScoresForSong(string songName)**
  Metoda načte všechny uložené skóre dané skladby. Využití má při zobrazování výsledků jak u jednotlivých skladeb, tak v sekci pro správu skladeb. Vstupním parametrem je název skladby, návratová hodnota je List s prvky třídy Score.

- **public static async Task<List<Song>> GetAllSongs()**
  Tato metoda slouží k získání seznamu všech skladeb, které jsou přístupné v databázi. Záznamy o skladbách jsou synchronizovány jako objekty třídy Song, metoda vrací jejich List.

- **public static async Task<bool> AddNewScore(string userName, string songName, int score, string difficulty)**
  Přidání skladby do databáze, je k tomu potřeba uživatelské jméno, název skladby, skóre a obtížnost. Návratová hodnota je typu bool podle toho, zda byla operace úspěšná.

- **public static async Task<bool> UpdateNameForScore(string oldName, string newName, string songName, string result, string difficulty)**
  Tato metoda slouží k přejmenování hráče u dané skladby.

- **public static async Task<bool> UpdateResultForScore(string userName,string songName, string oldResult, string newResult, string difficulty)**
  Úprava záznamu skóre.

- **public static async Task<bool> HasUserPlayedSongBefore(string userName, string songName, string difficulty)**
  Metoda zjišťuje, jestli už byla daná skladba někdy přehraná, tedy jestli existuje její záznam v databázi. Využívám ji v sekci skladby do hraní dle předlohy, pokud již byla hrána dříve, je uživatel dotázán, zda chce svým nově přehraným skóre přepsat předchozí výsledek.

[2] https://www.c-sharpcorner.com/article/xamarin-forms-login-forms-with-firebase-real-time-database-mvvm/

- **public static async Task<bool> DeleteScore(string userName, string songName, int score, string difficulty)**
  Odstraní záznam skóre pro skladbu daného uživatele.
- **public async static Task<bool> DeleteScoresForSong(string songName)**
  Odstraní všechny záznamy všech hráčů dané skladby.
- **public static async Task<bool> AddNewSong(string name, int maxScore)**
  Přidá do databáze novou skladbu a nejvýše dosažitelné skóre.
- **public static async Task<bool> DeleteSong(string name)**
  Odstraní skladbu z databáze.
- **public static async Task<bool> IsSongInDB(string songName)**
  Zjistí, zda je daná skladba už obsažena v databázi. Tato metoda je volána při inicializaci skladby pro hraní dle předlohy, pokud je výsledek false, pak se skladba přidá do databáze i s maximálním skóre.
- **public static async Task<bool> IsUserMadeSongInDB(string songName)**
  Zkontroluje, zda je skladba vytvořená v části Skladatel v databázi.
- **public static async Task<List<string>> GetAllUserMadeSongs()**
  Vrátí seznam všech skladeb vytvořených v části Skladatel, které se nachází také ve Firebase Cloud Storage.
- **public static async Task<bool> AddUserMadeSong(string name)**
  Přidá skladbu ze Skladatele do databáze.
- **public static async Task<bool> RemoveUserMadeSong(string name)**
  Odebere uživatelsky vytvořenou skladbu z databáze.

3.1.7 Třída FirebaseStorageHelper

Podobně jako u databáze, i k online úložišti poskytovaném společností Firebase je zapotřebí příslušný NuGet. Pro Firebase Cloud Storage se jedná o NuGet FirebaseStorage.net, který zpřístupňuje třídu FirebaseStorage, jejíž instance slouží k odesílání souborů na úložiště. Opět s pomocí návodu[3] využívám 3 metody.

- **public static async Task<bool> UploadFile(Stream stream, string filename)**
  Nahraje soubor převedený na Stream do úložiště.

[3] https://www.c-sharpcorner.com/article/xamarin-forms-working-with-firebase-storage-crud-operations2/

```
1  public static async Task<List<Song>> GetAllSongs()
2        {
3           try
4           {
5              var allSongs = (await firebase
6                 .Child("Songs")
7                 .OnceAsync<Song>()).Select(item =>
8                  new Song (item.Object.Name, item.Object.MaxScore)).
                      ToList();
9              return allSongs;
10          }
11          catch
12          {
13             return null;
14          }
15       }
```

Zdrojový kód 3: Získání seznamu skladeb

- **public static async Task<string> GetFile(string filename)**
  Metoda poskytující url adresu souboru, ze které je možné jej následně stáhnout do zařízení.
- **public static async Task<bool> DeleteFile(string filename)**
  Odstraní soubor z úložiště.

```
1  public static async Task<bool> UploadFile(Stream stream, string
       filename)
2         {
3            try
4            {
5               var url = await firebaseStorage
6                  .Child("songs")
7                  .Child(filename+".txt")
8                  .PutAsync(stream);
9               if (url != null)
10                 return true;
11              return false;
12           }
13           catch
14           {
15              return false;
16           }
17        }
```

Zdrojový kód 4: Odeslání souboru na úložiště

3.2 Stránky

U všech stránek je rozložení jejích prvků definováno v příslušném XAML souboru stránky. Pro docílení požadovaného vzhledu a chování na stránkách PlaySongPage a PianoFreePlay (tedy rozložení kopírující vzhled klavíru) bylo zapotřebí přidat řádek kódu do souboru MainActivity.cs, aby se klávesy mohly překrývat a nedocházelo k nežádoucím efektům jako upozadění půltónů (černé klávesy) při stisku bílých kláves okolo nich.

```
1  Forms.SetFlags("UseLegacyRenderers");
```

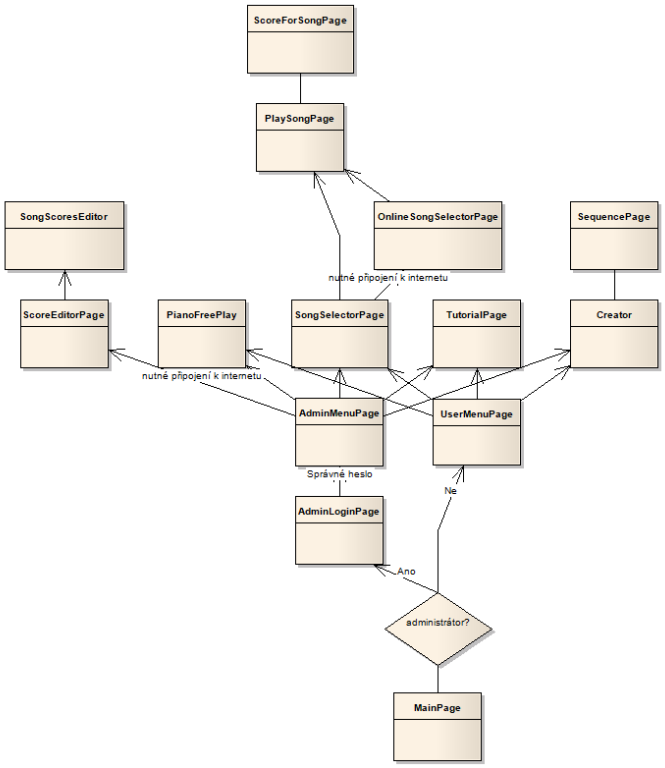
Zdrojový kód 5: Příkaz zaručující správnost chování kláves

Navigaci mezi stránkami zpřístupňuje třída NavigationPage fungující na principu zásobníku typu LIFO. Pro přechod na jinou stránku se stránka vloží do navigačního zásobníku, pro návrat zpět se naopak ze zásobníku odebere. [10] Pro odebrání všech stránek ze zásobníku kromě první strany (kořenové) slouží příkaz PopToRoot.

Přikládám diagram zobrazující dostupnost jednotlivých stránek.

```
1 Navigation.PushAsync(new PianoFreePlay()); //přechod na stránku
2 await Navigation.PopAsync(false); //návrat na předchozí stránku,
     volitelný parametr false předává informaci, že se nemá
     zobrazovat animace
3 await Navigation.PopToRootAsync(); //návrat na kořenovou stránku
```

Zdrojový kód 6: Navigace mezi stránkami



Obrázek 3: Diagram stránek aplikace

3.3 Důležité algoritmy

V následující části popíši důležité algoritmy své práce i s jejich ukázkami tak, aby případní zájemci o pokračování v práci na aplikaci mohli snadno pochopit, k čemu daný algoritmus slouží.

3.3.1 Metoda FlattenList

Tato metoda má klíčovou funkci krok před přehráváním skladby v sekci hraní dle předlohy. V seznamu tónů se mohou vyskytovat repetice, jejichž začátek je označen "začátek repetice", ukončen pak slovy "opakovat"spolu s počtem opakování, například "opakovat 2x". Před přehráváním skladby tento algoritmus zkopíruje požadované repetice za sebe a odebere všechny položky seznamu, které nejsou tóny.

```
1  public static List<string> FlattenList(List<string> content)
2        {
3           List<string> newList = new List<string>();
4           bool needsToBeFlattened = false;
5           for (int i = 0; i < content.Count; i++)
6           {
7              List<string> tempList = new List<string>();
8              if (content[i].ToLower().Equals("začátek repetice"))
9              {
10                int j = i + 1;
11                int cycleCounter = 0; //počet vnitřních repeticí
12                bool innerCycle = false;
13                while (j < content.Count && (cycleCounter != 0 || (!
                      content[j].ToLower().StartsWith("opakovat"))))
14                {
15                   tempList.Add(content[j]);
16                   if (content[j].ToLower().Equals("začátek repetice"
                         ))
17                   {
18                   //vnitřní repetice, je potřeba alespoň ještě
                         jednou spustit metodu FlattenList
19                      needsToBeFlattened = true;
20                      cycleCounter++;
21                      innerCycle = true;
22                   }
23                   if (content[j].ToLower().StartsWith("opakovat"))
24                      cycleCounter--;
25                   j++;
26                }
27                if (innerCycle)
28                   tempList = FlattenList(tempList);
29                string repeatHelper = content[j];
30                repeatHelper = repeatHelper.Remove(repeatHelper.
                      Length - 1);
31                int.TryParse(repeatHelper.Split(' ')[1], out int
                      numOfRepeats);
32                while (numOfRepeats > 1) //>1 protože jednou to tam u
                      ž je
33                {
34                   foreach (string str in tempList)
35                      newList.Add(str);
36                   numOfRepeats--;
37                }
38             }
39             else if(!content[i].ToLower().StartsWith("opakovat"))
40                newList.Add(content[i]);
41          }
42          if (needsToBeFlattened)
43             newList = FlattenList(newList);
44          if (newList.Count == 0)
45             return content;
46          return newList;
47       }
```

Zdrojový kód 7: Metoda FlattenList

3.3.2 Přehrávání skladby

Je-li po spuštění přehrávání zvolena lehká obtížnost, každá nota, která se zobrazuje v labelu je také označena na příslušné klávese, kterou má hráč stisknout. Pokud se trefí, je mu přičteno 10 bodů. Ve střední a těžké obtížnosti se klávesy neoznačují, zobrazují se noty pouze v labelech, přičemž při těžké obtížnosti má hráč 3 vteřiny na stisknutí klávesy, pokud to nestihne, nedostane za tón body a pokračuje se další notou ze seznamu. Odpočet je realizován pomocí objektu třídy BackgroundWorker.

```
1  //na začátku přehrávání skladby je vždy inicializován
       BackgroundWorker
2    background.DoWork += new DoWorkEventHandler(Countdown); //odpočet
       3 vteřin
3      background.RunWorkerCompleted += new
         RunWorkerCompletedEventHandler(CountDownEnded); //skončí-li
         odpočet a uživatel nestiskne klávesu, tak se změní noty v
         labelech
4
5  //uvnitř metody registrující stisknutí klávesy se pak odpočet zastav
       í
6  if (background.IsBusy)
7  {
8     background.RunWorkerCompleted -= CountDownEnded; //zajistí, že tó
         n nebude přeskočen
9    background.CancelAsync();
10 }
11 private void Countdown(object sender, DoWorkEventArgs e)
12       {
13          int i = 0;
14          while (i < 100)
15          {
16             if (this.background.CancellationPending)
17                return;
18             Thread.Sleep(30);
19             if (this.background.CancellationPending)
20                return;
21             i++;
22          }
23       }
```

Zdrojový kód 8: Algoritmy k hraní dle předlohy

3.3.3 Skladatel

Důležitým prvkem ve skladateli pro správné zapsání tónu je tzv. enharmonická záměna.

Definice 1 (Enharmonická záměna)
2 tóny, které zní stejně, v případě klavíru jsou tedy na stejné černé klávese, ale mají jiné názvy jsou takzvaně enharmonické. V tomto případě lze provést enharmonickou záměnu, během které použijeme název tónu, který se nám hodí. Jelikož půltóny se nepoužívají mezi tóny E a F a mezi H a C, provádí se záměna na celé tóny.

Příklad: Tón des1 je na první černé klapce zleva, stejně jako tón cis1, lze tedy provést enharmonickou záměnu.

V práci přesto ponechávám možnost zvolit například tón eis, který se převede na tón f, činím tak z toho důvodu, že v některých zpěvnících může být rozdílné značení a po uživateli by byla tím pádem vyžadována znalost hudební teorie. Pro tento přepis používám jednoduchý switch.

```
1  private string FixEnharmony(string tone)
2        {
3           switch (tone)
4           {
5              case "eis": tone = "f"; break;
6              case "fes": tone = "e"; break;
7              case "aes": tone = "as"; break;
8              case "ees": tone = "es"; break;
9              default: break;
10          }
11          return tone;
12       }
```

Zdrojový kód 9: Enharmonická záměna

4 Uživatelská příručka

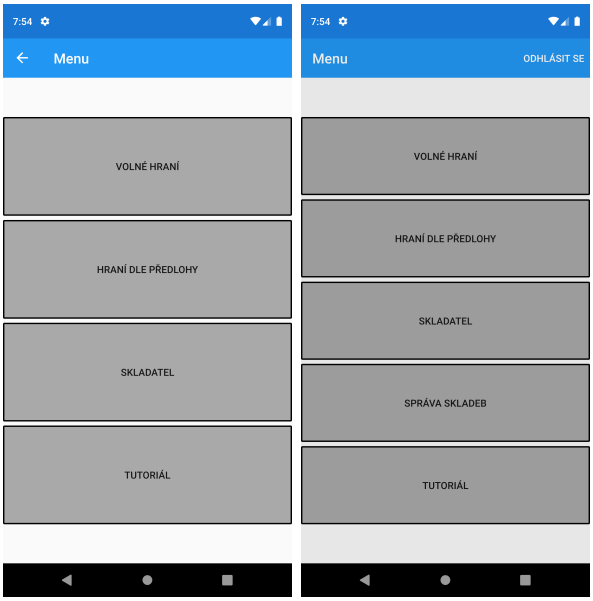
Následující kapitola slouží jako návod pro uživatele, jak aplikaci nainstalovat, používat ať už v klientském nebo administrátorském módu a přidávat a spravovat skladby pro klienty. Manuál je doplněn o screenshoty zmiňovaných částí aplikace a demonstujících její použití.

4.1 Instalace

K instalaci je nutné povolit instalaci z neznámých zdrojů, jelikož se nejedná o aplikaci uloženou na Google Play Store. Tato možnost se povoluje v nastavení, typicky v sekci zabezpečení, případně v sekci aplikace, může se to lišit dle nadstavby a verzi operačního systému Android. Po povolení instalace z neznámých zdrojů stačí přímo v telefonu otevřít balík klavir.apk, načež proběhne instalace. Aplikace je cílena na Android 9.0, avšak má podporu i na předchozí verze.

4.2 Spuštění aplikace

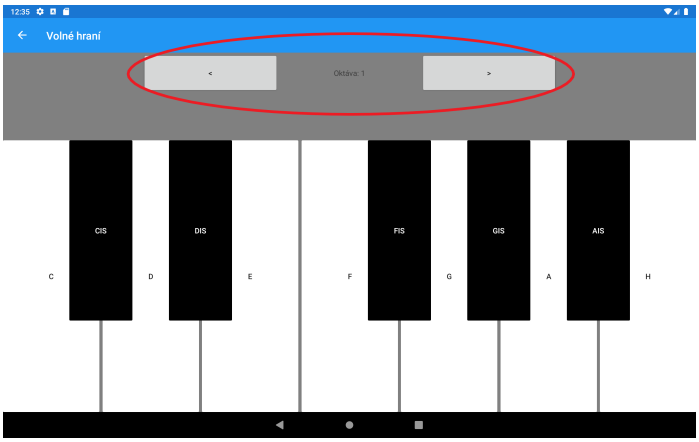
Po spuštění aplikace se na úvodní obrazovce zobrazí 2 tlačítka, kde si uživatel zvolí, zda chce pokračovat jako administrátor nebo jako klient. Při zvolení administrátora bude vyžadováno přístupové heslo, které pracovníci centra Klíč dostali spolu s aplikací. Po tomto kroku se zobrazí hlavní menu, které se v administrátorské verzi od klientské liší navíc položkou správy skladeb, která bude popsána níže.



Obrázek 4: Rozdíl klientské vs. administrátorské menu

4.3 Volné hraní

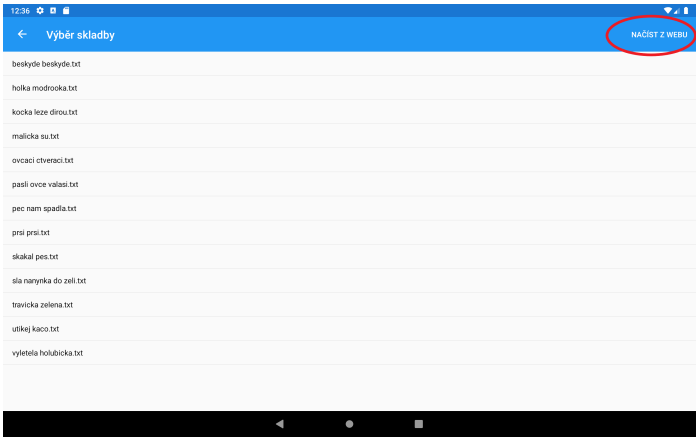
V této části aplikace se uživatelé mohou seznámit s klavírem jako takovým. V horní části obrazovky je přepínač oktáv (označeno červeným kolečkem), který lze posouvat mezi první až čtvrtou oktávou. ve spodní části obrazovky jsou pak klávesy klavíru představující jednu oktávu. Bílé klávesy jsou celé tóny, zatímco černé klávesy jsou půltóny, stejně jako na standardním klavíru.



Obrázek 5: Sekce volné hraní

4.4 Hraní dle předlohy

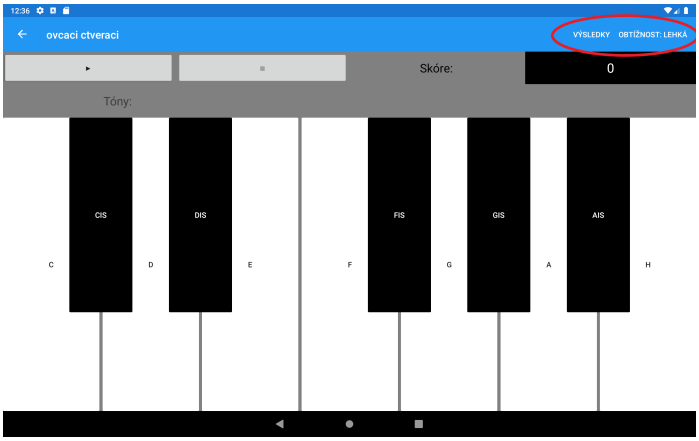
Tato část má výchovný účel, protože klienti se v nich mohou naučit přehrávat skladby. Uživatel si nejprve zvolí skladbu ze seznamu. Mají-li zájem o skladbu, která není v základní nabídce, je k dispozici tlačítko „načíst z webu“ (označeno červeným kolečkem), které zobrazí seznam skladeb, které jsou na online úložišti.



Obrázek 6: Výběr skladby

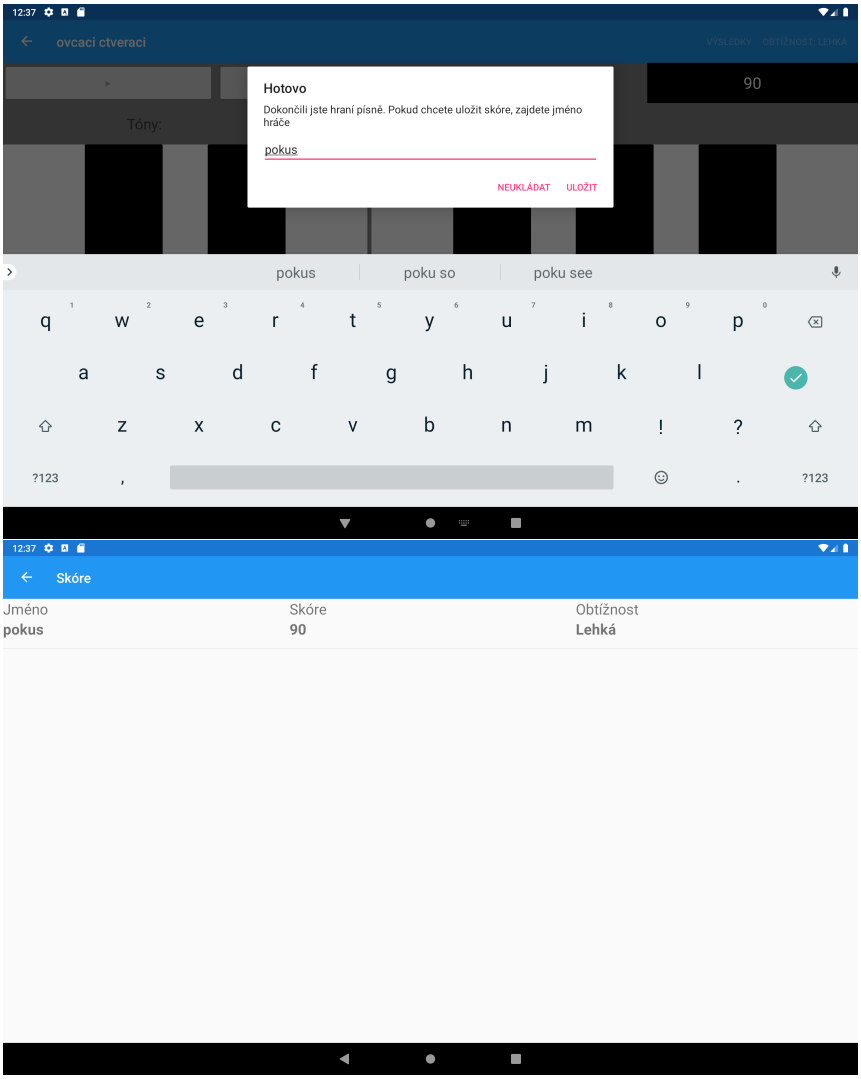
Po výběru skladby je načtena stránka, na které mohou klienti vybranou skladbu přehrát. V horní části obrazovky je možno zobrazit výsledky ostatních hráčů, kteří již skladbu odehráli, a přepínač obtížnosti (v červeném kolečku). Stupně obtížnosti jsou 3, rozděleny následovně:

- **Lehká** - Při hraní se zobrazují nejen tóny, které se mají přehrát, ale navíc se zabarví klávesa, která je není na řadě
- **Střední** - Při hraní se zobrazují tóny, ale klávesy nejsou zvýrazněny
- **Těžká** - Při hraní se zobrazují tóny, klávesy nejsou zvýrazněny a hráč má 3 vteřiny na stisknutí požadované klávesy, aby za tón dostal body.



Obrázek 7: Obrazovka přehrávání skladby

Máte-li kýženou obtížnost zvolenu, skladba se zahajuje stiskem šipky. Během přehrávání můžete sledovat své průběžné skóre. Za každý správně odehraný tón se přičítá 10 bodů, při stisku nesprávné klávesy se nepřičítá nic. Skladba lze kdykoliv během přehrávání zastavit, v takovém případě se skóre neukládá. Není-li přehrávání skladby zastaveno a uživatel dokončí přehrávání, vyskočí vyskakovací okno, do kterého může hráč zadat své jméno, přeje-li si uložit své skóre dané skladby. Toto skóre lze ihned po odeslání vidět ve výsledkové tabuli skladby, kterou lze zobrazit stiskem tlačítka „výsledky“.

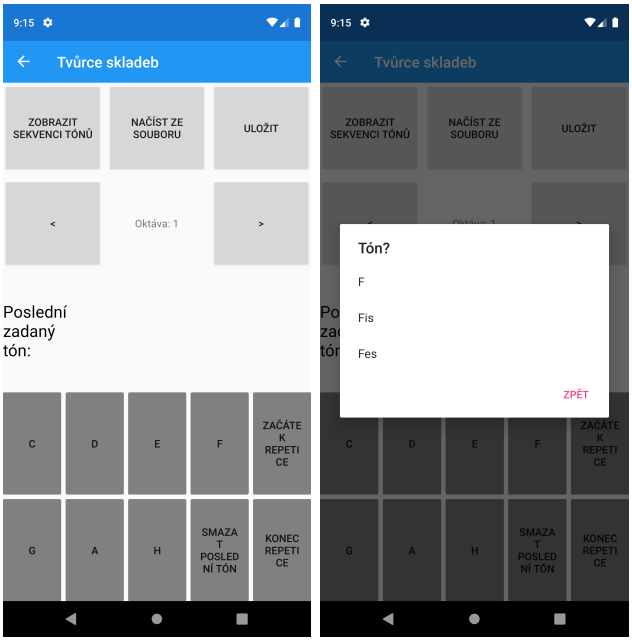


Obrázek 8: Dokončení přehrávání skladby a uložení skóre

4.5 Skladatel

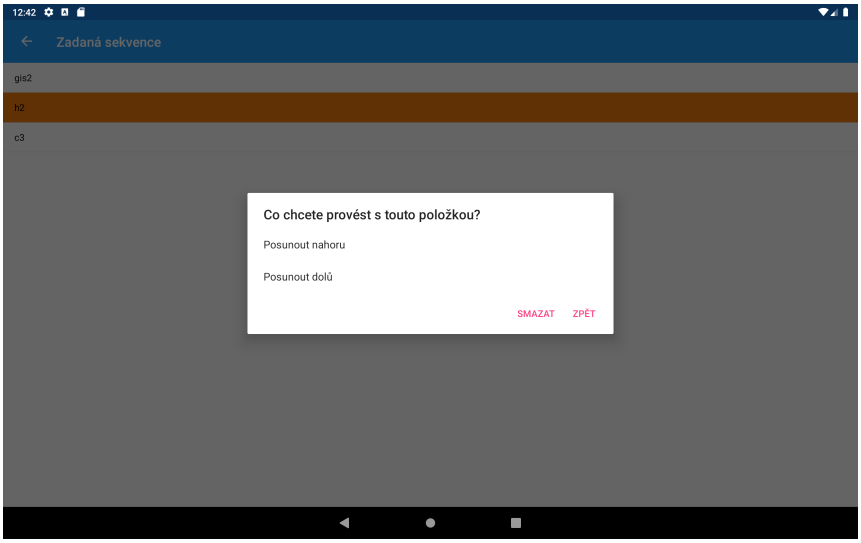
V části skladatel mohou hudebněji nadanější klienti zužitkovat své znalosti a tvořit si vlastní skladby, které po uložení budou k dispozici na online úložišti aplikace. Opět je zde přítomen přepínač oktáv, klávesy však nejsou v tradičním provedení, ale jsou reprezentovány pojmenovanými tlačítky. Při stisku tlačítka vyskočí dialogové okno, ve kterém může uživatel zvolit, zda chce zapsat notu celou nebo nějaký půltón, viz níže na obrázku, kde po stisku klávesy F je nabídka také not Fis a Fes. Z důvodu korektního zápisu se provede tzv. enharmonická záměna, místo tónu hes se tak zapíše do seznamu tón gis. Toto není chyba, ale pouze část hudební teorie, od jejíž znalosti jsou uživatelé co nejvíce oproštěni.

Po zvolení patřičného tónu se tón přidá do sekvence, kterou lze zobrazit pomocí tlačítka „zobrazit sekvenci tónů“. Mimo tónů se zde nachází taktéž repetice, dbejte prosím na to, že v sekvenci musí být stejný počet začátků repetic a jejich



Obrázek 9: Výběr tónu ve skladateli

konců, které jsou značeny slovy opakovat (například opakovat 2x). V opačném případě se nepodaří sekvenci uložit. V zobrazení sekvence tónů lze s položkami manipulovat, ať už se jedná o posouvání nahoru a dolů nebo smazání tónu, který jste například umístili omylem.



Obrázek 10: Úprava zadané sekvence

Skladby se nemusí zapisovat pouze ve skladateli, k jejich zápisu je možné použít jakékoliv zařízení, pokud jsou dodrženy podmínky zápisu a zvolen správný formát. Soubor musí být ve formátu txt a podmínkou je, že na každém řádku je 1 informace, tedy buďto tón nebo označení repetice.
Příklad korektní sekvence:

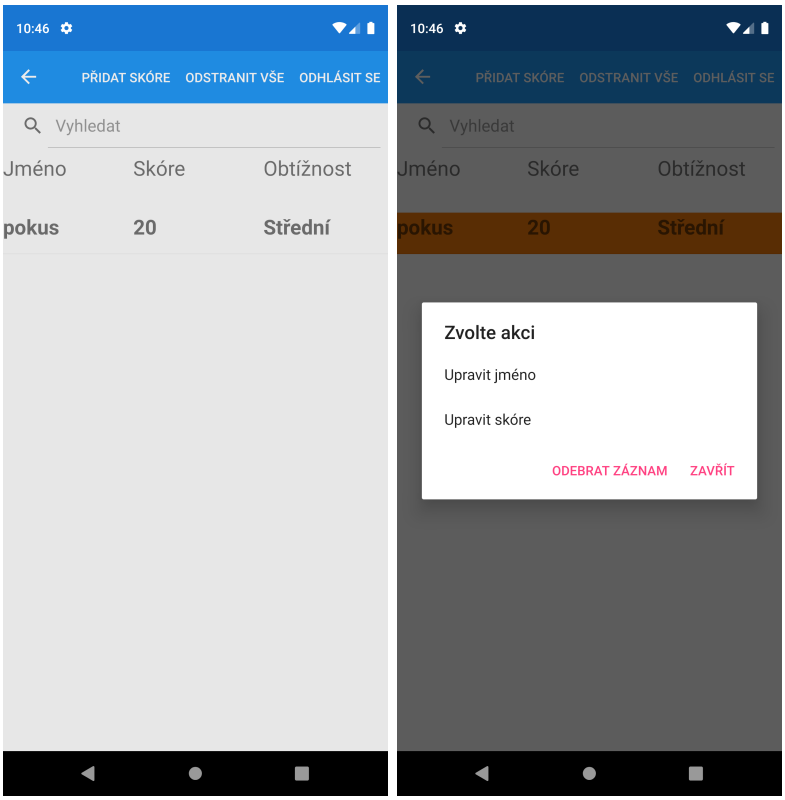
začátek repetice
c2
dis2
f2
opakovat 2x

Po zápisu stačí soubor txt přetáhnout do telefonu a ve skladateli zvolit tlačítko načtení ze souboru. Aplikace se zeptá, jestli je soubor uložen na místním nebo online úložišti, pokud jej máte v telefonu, zvolte místní úložiště, pokud chcete upravit už dříve napsaný a uložený soubor, zvolte online úložiště.
Před uložením souboru aplikace zkontroluje, zda je sekvence ve správném tvaru, tedy zda neobsahuje neplatné znaky a je stejný počet začátků a konců repetic. Je-li vše v pořádku, pak se sekvence uloží na online úložiště se zadaným názvem a skladba bude přístupná v hraní dle předlohy.

4.6 Správa skladeb

Tato část je přístupná pouze pro administrátory. Ti na obrazovce uvidí seznam skladeb, které je možno spravovat, tedy upravovat skóre uživatelů pro skladby nebo případně mazat skladby, které jsou vytvořeny uživateli. Jelikož skladatel je volně přístupný i klientům, dává tato možnost administrátorovi kontrolu nad

těmito skladbami a nevhodné výtvory může smazat, pokud je nechce upravit ve skladateli. U skladeb, které jsou v základu aplikace, se aplikace rovnou přesune na úpravu skóre, u skladeb vytvořených uživateli je administrátor nejprve dotázán, zda chce u skladby upravit skóre nebo ji smazat.
Na obrazovce samotné správy skóre má administrátor možnost upravovat jednotlivé záznamy prostým kliknutím, přidat skóre, které se například neodeslalo do databáze kvůli absenci internetového připojení, nebo odebrat všechna skóre skladby.



Obrázek 11: Správa skóre skladby

Závěr

Cílem práce bylo vytvořit aplikací simulující klavír pro klienty centra sociálních služeb Klíč, kteří mohou mít různé stupně mentální retardace nebo poruchy autistického spektra. Je tedy určena primárně pro toto centrum. Aplikace je vytvořena pro operační systém Android.
Prostřednictvím této aplikace by měli klienti lépe vnímat hudbu, což na řadu z nich může mít terapeutický efekt. Klienti se mohou seznámit s fungováním klavíru, ti zdatnější se na něj naučit také přiložené lidové skladby. Ti nejnadanější si také mohou složit skladby vlastní, případně si ze zpěvníku přepsat jejich oblíbené písně, což mohou dělat také administrátoři, a tímto způsobem rozšiřovat hudební databázi aplikace. Aplikace je tedy rozdělena na klientskou a administrátorskou část, aby měli zaměstnanci klíče větší kontrolu nad prací klientů v aplikaci.
Aplikace by se dala vylepšit tím, že by se nezobrazovaly klávesy klavíru jen na jednu oktávu, ale celé spektrum klavíru, což by přineslo mnohem reálnější zkušenost. Já jsem tuto možnost nezvolil z důvodu, že by pak na obrazovce bylo buď příliš mnoho kláves a tím se snížila přehlednost, nebo by se muselo po obrazovce posouvat do kýžených částí klavíru, což by pro mohlo být příliš složité pro přehrávání skladeb, zvláště pak pro méně zkušené klienty.

Conclusions

The main goal of this thesis was to create piano simulating application for Klíč social services center, whose clients may have different stages of mental retardation or autistic disorders. It's mainly intended for the center's purposes. Application was created for Android operating system.
In this app clients should be able to learn to perceive music which can have therapeutical effect on many of them. Clients can learn how piano works or even learn some czech traditional songs. The more musically experiences ones can also compose their own songs or rewrite their favourite songs using sheets of music notes. This is also allowed for administrators and it's the way of expanding the application's music database. The application has two parts - the client part and the administrator part. With this division, administrators can supervise the clients' behaviour in app.
The application can be improved by showing all of the piano keys, not just one octave, which would bring much more realistic experience. I've chosen not to go this way because there would be either way too many keys on one screen or the clients would have to slide to wanted piano parts, which would've been too complicated for simple songs playing, especially for less experienced clients.

5 Obsah přiloženého CD/DVD

Přikládám popis obsahu CD/DVD, které je přiloženo k mé práci.

`bin/`

- **klavir.apk** - instalační balíček aplikace

`doc/`

- **vagner-bakalarska-prace.pdf** - text práce ve formátu pdf
- **src/** - složka se zdrojovým kódem a obrázky textu práce

`src/`

- zdrojové kódy aplikace

`readme.txt`

- instrukce pro instalaci aplikace a požadavky na její provoz

Literatura

[1] *What is .NET? An open-source developer platform .NET | Free. Cross-platform. Open Source. [online].* Dostupný z: ⟨https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet⟩.

[2] *Úvod do jazyka C# a rozhraní .NET Framework | Microsoft Docs. [online]. Copyright © Microsoft 2020.* Dostupný z: ⟨https://docs.microsoft.com/cs-cz/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework⟩.

[3] *What is Xamarin? | .NET. .NET | Free. Cross-platform. Open Source. [online].* Dostupný z: ⟨https://dotnet.microsoft.com/learn/xamarin/what-is-xamarin⟩.

[4] *Xamarin.Forms | .NET. .NET | Free. Cross-platform. Open Source. [online].* Dostupný z: ⟨https://dotnet.microsoft.com/apps/xamarin/xamarin-forms⟩.

[5] *Přehled XAML - WPF | Microsoft Docs. [online]. Copyright © Microsoft 2020.* Dostupný z: ⟨https://docs.microsoft.com/cs-cz/dotnet/desktop-wpf/fundamentals/xaml⟩.

[6] *Co je NuGet a co dělá? | Microsoft Docs. [online]. Copyright © Microsoft 2020.* Dostupný z: ⟨https://docs.microsoft.com/cs-cz/nuget/what-is-nuget⟩.

[7] *Dotaz integrovaný jazykem (LINQ) (C#) | Microsoft Docs. [online]. Copyright © Microsoft 2020.* Dostupný z: ⟨https://docs.microsoft.com/cs-cz/dotnet/csharp/programming-guide/concepts/linq/⟩.

[8] *Firebase Realtime Database. Firebase [online].* Dostupný z: ⟨https://firebase.google.com/docs/d

[9] *MIDI.Org Home. MIDI.Org Home [online].* Dostupný z: ⟨https://www.midi.org⟩.

[10] *Hierarchická navigace - Xamarin | Microsoft Docs. [online]. Copyright © Microsoft 2020.* Dostupný z: ⟨https://docs.microsoft.com/cs-cz/xamarin/xamarin-forms/app-fundamentals/navigation/hierarchical⟩.