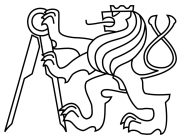
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Mobilní aplikace pro trénink tance (iOS)
Student:	Jan Ševela
Vedoucí:	Ing. Josef Gattermayer
Studijní program:	Informatika
Studijní obor:	Web a multimédia
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce zimního semestru 2017/18

Pokyny pro vypracování

Cílem práce je vytvořit mobilní aplikaci pro iOS, která bude kombinovat poslech playlistů ze zařízení dle nastavení uživatele. Bude sloužit pro trénink společenského či sportovního tance.

Uživatel bude mít skladby roztříděné do playlistů podle tanců. Aplikace bude umět přehrávat skladby o nastavené délce a náhodně vybírat skladby z vybraných playlistů. Mezi skladbami bude možnost nastavení délky pauzy. Přehrávač bude umět přehrát po sobě např. všechny playlisty vždy po jedné skladbě z daného playlistu.

Pokyny:
- Proveďte analýzu jiných hudebních přehrávačů pro společenské tance z App Store.
- Proveďte analýzu požadavků ze strany potenciálních uživatelů.
- Navrhněte funkcionalitu aplikace a jednotlivé obrazovky ve formě wireframů.
- Navrhněte a implementujte aplikaci.
- Aplikaci otestujte.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 4. března 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Mobilní aplikace pro trénink tance (iOS)

Jan Ševela

Vedoucí práce: Ing. Josef Gattermayer

10. ledna 2017

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Josefu Gattermayerovi za důvěru při samostatném výběru tématu práce a za jeho cenné rady a připomínky. Další poděkování patří rodině, všem přátelům a spolužákům, kteří mě podporovali a jakýmkoli způsobem pomáhali během celého studia.

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. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 10. ledna 2017 .....

Odkaz na tuto práci

Ševela, Jan. *Mobilní aplikace pro trénink tance (iOS)*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Tato bakalářská práce se zabývá tvorbou mobilní hudební aplikace, která bude sloužit pro ulehčení tréninku tance. Aplikace je řešena pro iOS zařízení a tvořena pomocí moderního programovacího jazyku Swift. Práce se zabývá řešením pomocí průzkumu jiných aplikací podobného charakteru (hudební aplikace), dále vlastních i cizích zkušeností, důležitých pro nalezení způsobu spojení aplikace s tréninkem tance. Velký důraz se klade na přehlednost a jednoduchost uživatelského rozhraní, na běžné zvyky uživatelů iOS zařízení a především potenciální uživatele aplikace. Práce má ukázat velký přínos mobilních aplikací pro automatizovanější způsob výuky tance s pomocí lehkého přenosného zařízení. Dále je doplněna o doporučení, jakým způsobem aplikaci vyvíjet v budoucnu, aby byla ještě užitečnější.

Klíčová slova mobilní aplikace, hudební aplikace, ulehčení tréninku tance, návrh uživatelského rozhraní, iOS, Swift

Abstract

The bachelor thesis deals with development of a mobile music app, which is going to help with a dance training. This app is designed for iOS devices and was created by a modern programme language called “Swift”. This work is based on a carefull research of apps of a similar character (music apps), own experince and experience of other dancers. These pieces of information were essential for a suitable solution connecting a mobile app with a dance practising. The main importance is dedicated to the simplicity and the clarity of the user interface, which should be easily understandable for the experienced iOS users as well as for the potencial users of the app. The aim of the theses is to show a big benefit of mobil apps for automatization of dance education by means of a lightweight portable device. Furthermore, the thesis is accomplished by recommendation about the next possible steps in the future development of this app to achieve the best use of its functions.

Keywords mobile application, music application, help with a dance training, design user interface, iOS, Swift

Obsah

Seznam obrázků

Úvod

Používání chytrých mobilních zařízení se stalo součástí všedního života. Dnes už se jen málokdo obejde bez chytrého mobilního telefonu nebo tabletu. Téma jsem si zvolil na základě myšlenky, jakým způsobem můžeme tato chytrá mobilní zařízení využít pro podporu tréninku a soutěže v tanci.

Při průzkumu hudebních aplikací jsem nenašel žádnou, která by sama na základě jednoduchého nastavení pouštěla skladby z různých seznamů skladeb (dále jen „playlisty“, za sebou v nastavené délce a se zvolenou pauzou.

V práci se zaměřuji na chytrá mobilní zařízení s operačním systémem iOS, protože mě tato chytrá mobilní zařízení z dlouhodobých osobních zkušeností nejvíce zaujala. Sám jsem jedním z uživatelů takového zařízení, a je pro mě o to jednodušší testování aplikace na iOS.

Nejdříve se zabývám historií a základního přehledu iOS, dále souvisejícími službami a nástroji pro vývoj na této platformě a modernímu programovacímu jazyku Swift.

V další části provádím analýzu potenciálních uživatelů, jejich požadavků a s těmito znalostmi přecházím k průzkumu jiných hudebních přehrávačů.

Následující návrh uživatelského rozhraní a návrh implementace vychází z analýzy a doporučených postupů při vývoji na této platformě.

Při práci s aplikací je nutná základní znalost systémové hudební knihovny na iOS zařízeních, protože pracuji s částí filosofie a daty této hudební knihovny, a to s playlisty, které lze spravovat pouze přes ni.

V předposlední části práce se zabývám samotnou implementací aplikace, po níž následuje již jen testování aplikace.

Práce souvisí s bakalářskou prací „Hudební přehrávač pro OS Android“ studenta Jakuba Petáka z Fakulty informačních technologií Českého vysokého učení technického, který téměř shodnou práci již zpracoval na platformě Android. Já vycházím ze svých praktických zkušeností ze světa tance a vývoje právě pro iOS platformu.

Kapitola 1

Cíl práce

Cílem práce je vytvořit hudební aplikaci pro chytrá mobilní zařízení na platformě iOS, která bude sloužit k ulehčení výuky tance.

Aplikace musí být schopna zobrazovat playlisty ze systémové hudební knihovny iOS zařízení a všechny jejich skladby umět přehrávat. Musí umožnit playlisty přiřadit k takzvaným mixům, pomocí kterých bude schopna poslech skladeb z nich kombinovat. Například první skladba má být z playlistu A, další bude z B a poté má přijít na řadu skladba z playlistu C.

Toto přehrávání je třeba automaticky řídit pomocí přednastavených hodnot – délka skladby, délka pauzy před skladbou a opakování skladby v mixu. Aby mohl být poslech plynulý i při krácení jejich délek jako při poslechu celých skladeb, je potřeba skladbu před umělým ukončením ztlumit.

Nedílnou součástí práce je zajisté tvorba uživatelského rozhraní aplikace, která při naplnění všech funkcionalit také musí být lehce ovladatelná. Je tedy potřeba rozhodnout, kde hledat inspiraci; jestli v jiných hudebních přehrávačích nebo v potřebách potenciálních uživatelů. Je důležité najít ideální kompromis.

Kapitola 2

Základní přehled kolem operačního systému iOS

Tato kapitola se zaměřuje na historii operačního systému iOS a vysvětlení základních pojmů, na které vývojář jistě narazí dříve nebo při samotném vývoji.

2.1 iOS

2.1.1 Historie

Historie operačního systému iOS sahá do nedávné minulosti, a to do roku 2007, kdy jeden ze zakladatelů společnosti Apple, Steve Jobs, představil první iPhone. Jednalo se o první zařízení, na němž bylo koncipováno uživatelské rozhraní pomocí manipulace a ovládání multi-dotykových gest přímo na obrazovce dotykového displeje. Rozhraní se ovládalo pomocí virtuálních tlačítek, přepínačů a posuvníku, žádná přídavná nebo integrovaná klávesnice. Tři pevná tlačítka byly na boční straně telefonu, a to pro ovládání hlasitosti a uzamknutí telefonu. Na přední straně pod displejem bylo pouze jediné tlačítko, které se stalo velikou dominantou všech iPhonu v jeho dalších generacích.

Operační systém iOS se v tomto znění začal používat až od roku 2010. Do té doby vlastně ani oficiální pojmenování neměl. První zmínka o iPhone OS 1 byla uvedena v dokumentaci společnosti Apple roku 2008, po němž přišly na řadu verze iPhone OS 2 (2008) a iPhone OS 3 (2009). Až teprve čtvrtá verze dostala dnes již známé pojmenování iOS s číslem 4.

Od uvedení prvního operačního systému vychází každý rok nová verze; aktuální verze končí prozatím na 10.2.0. Tímto třímístným formátem se označují všechny verze; číslo uvedené na druhém místě značí vetší změny v systému dané verze, třetí pak jen menší opravy.[3]

2.1.2 Soušasnost - iOS 9

Jedná se o nejnovější operační systém pro přenosné mobilní zařízení společnosti Apple, v tuto chvíli podporované na 9 různých iPhonech, 11 iPadech a 2 iPodech.[4]

iPhone 5, 5C, 5S, 6, 6 Plus, 6s, 6s Plus, SE, 7, 7 Plus
iPad mini 2, mini 3, mini 4, 4. generace, Air, Air 2, Pro 9,7placový, Pro 12,9placový
iPod touch 6. generace

Výhody této aktuální verze oproti dřívějším verzím kromě vyššího výkonu, lepšího zabezpečení zejména rozšířená spolupráce mezi nativními a oblíbenými aplikacemi. iOS 10 přináší rozsáhlé možnosti pro zprávy, nové využití Siri s oblíbenými aplikacemi, změny v aplikaci Mapy a z obrázků ve fotogalerii vytváří vzpomínky. Tato verze se tedy kromě znatelných vizuálních změn zaměřuje zejména na širší možnosti interakce pro uživatele. Dále také přinesla hodně výrazné změny v aplikacích Hudba a Domácnost.[5]

2.1.3 Přednosti

Velikou předností je, že společnost Apple vyrábí hardware a zároveň vyvíjí operační systém pro všechna svá mobilní zařízení. Všechno tak může lépe spolupracovat, než tomu bývá u konkurenčních výrobců. Ti staví svá zařízení minimálně na základech operačních systémů Android nebo Windows.

Jedna z největších výhod je bezpečnost. Je čím dál komplikovanější nalézt bezpečnostní díry v tomto operačním systému a vytvořit tzv. Jailbreak, pomocí něhož se stane zařízení nějakým způsobem více zranitelnější. Ale i proti tomu je to při porovnání s konkurenčními operačními systémy v čele s operačním systémem Android nesrovnatelné.

Jailbreak je proces, pomocí kterého je umožněno se dostat do souborového systému iOS a provádět v něm operace pod Root právy. Root práva jsou nejvyšší práva v tomto operačním systému, s nimiž v něm lze provést jakoukoli změnu. Tento proces umožnuje zručným programátorům uvolnit ruce při programování a dostat do tohoto zařízení téměř cokoliv. Je možné například analyzovat jinou aplikaci, nějak ji pozměnit. Tímto způsobem lze udělat veliký zásah do bezpečnosti aplikací a ohrozit tak každého uživatele takové aplikace.

Uživatelské rozhraní je uživatelsky mnohem přívětivější, než je tomu u konkurence. Dokáže uspokojit jak náročnější uživatele, tak uživatele s menší znalostí a potřebou využívat chytré mobilní zařízení na jeho plný výkon. Bezpečnost a uživatelské rozhraní operačního systému iOS jsou těmi největšími výhodami, které dokážou zaujmout širokou veřejnost.

2.1.4 Systémová hudební knihovna

První zařízení tohoto operačního systému bylo uvedeno až poté, co již pár let dominovalo v odvětví přenosných hudebních přehrávačů jiné zařízení společnosti Apple, a to iPod. iPhone tak od základu používal k přehrávání tzv. iPod player. Jednalo se o základní přehrávač vycházející právě z iPodů.

Souběžně s iPhone pravidelně vychází mobilní zařízení určeno především k potřebám poslechu hudby a to iPod Touch, který je jedním z rodiny iPod zařízení, ale zároveň téměř identický právě s aktuální verzí iPhonu. To má nejspíše poukazovat na to, že jsou si iPod a iPhone velice blízké zařízení, protože stejný operační systém široké veřejnosti mnoho neřekne.

Vlastnosti převzaté z rodiny iPod zařízení lze vidět při správě hudební knihovny na všech iOS zařízeních. Nahrávání muziky k poslechu bylo do nedávné doby možné alespoň nějakým způsobem, právě přes desktopovou aplikaci iTunes, která vyšla s prvním iPod zařízením na kterém to byl jediný způsob jak si do něj nahrát hudbu prostřednictvím osobního počítače.

Desktopová aplikace iTunes cílí především na prodej skladeb ze svého iTunes hudebního obchodu, který je jeho součástí, ale také jako samostatná aplikace v iOS zařízeních; a tak je potřeba při nahrávání skladeb z osobního počítače do iOS zařízení trochu zručnosti a zkušeností, anebo dobrého rádce. Po všech překážkách, jakmile je celý proces pochopen, je nutné dodat, že tato forma řízení a nastavování není od věci a dá se s ní pracovat. Alespoň to určuje základní směr a jednotu při vývoji dalších hudebních aplikací, aby byly správně vnímány koncovým uživatelem. Při vývoji této hudební aplikace jsou využity systémové možnosti iPod playeru, jakožto systémové hudební knihovny v iOS zařízeních.

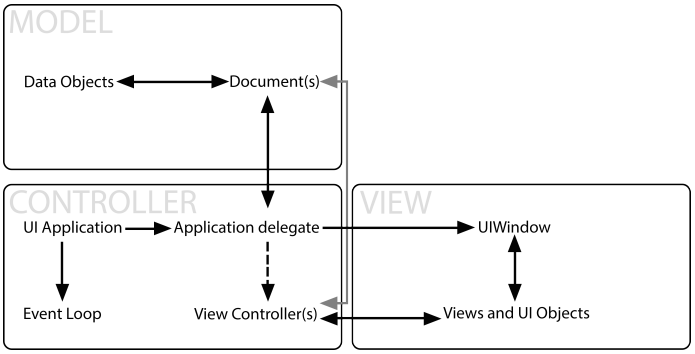
Pomocí desktopové aplikace iTunes lze při vývoji instalovat vytvořenou aplikaci do mobilního zařízení, ale je zde nutností myslet na uvedení ID tohoto zařízení v developer úctu (vizte sekci 2.6)) a následné vygenerování certifikátu s touto informací, pomocí kterého je aplikace z Xcode (vizte sekci 2.4) podepsána, aby mohla být spuštěna na koncovém zařízení.

2.2 Infrastruktura iOS

iOS poskytuje pro vývoj mnoho systémových frameworků, které dokáží vytvářet každé aplikaci její pevný základ. Všechny tyto frameworky pracují na návrhových vzorech jako např. MVC (Model-View-Controller) a delegování jejich implementací. Má-li být aplikace efektivní, měla by tyto frameworky využívat a měl by její vývojář chápat základní infrastrukturu iOS.

2.2.1 Hlavní funkce

Každá aplikace vytvářená programovacím jazykem na bázi jazyku C má vstupní bod funkcí main. Pro iOS se tato funkce nijak nevytváří, Xcode (vizte sekci



Obrázek 2.1: Objekty aplikace[1]

2.4) si ji před kompilací samotné aplikace generuje sám a vloží do ní inicializační kód a storyboard soubory vytvářené vývojářem aplikace. Implementace této funkce by se neměla nijak modifikovat.[1]

2.2.2 MVC struktura aplikace

iOS aplikace využívají **MVC** (Model-View-Controller) architekturu, která odděluje data z aplikací a logiky od vizuální prezentace. Na obrázku 2.1 můžete vidět možnou kooperaci této architektury v iOS.[1]

Pro usnadnění spolupráce mezi systémem a každou aplikací existuje UIApplication objekt, který spravuje smyčku událostí, zaznamenává klíčové změny a komunikuje s delegátem aplikace. Srdcem kódu aplikace je delegát (objekt App delegate), jenž úzce spolupracuje s UIApplication objektem, inicializuje aplikaci při spuštění a řídí její aktuální stav. Oba tyto objekty jsou základními prvky controller části MVC. Kromě těchto objektů má téměř každý pohled aplikace svou **controller** logiku pro řízení a zpracovávání všech požadavků dané pohledové části. UIViewController je základní třídou pro regulaci a funkcionalitu pohledové části.[1]

Modelovou část MVC návrhu tvoří zpravidla datový model a dokumentové objekty podtřídy UIDocument z frameworku UIKit. Každá aplikace je specifická svým datovým modelem, např. bankovní aplikace může ukládat databázi finančních transakcí, naproti tomu v aplikaci s malováním lze ukládat obrázky nebo sekvenci kreslicích příkazů, která vede k výslednému obrázku.[1]

Pohledovým (View) prvkem MVC návrhu je nejvýše v hierarchii objekt UIWindow, pomocí něhož se koordinují různé UI objekty a jeden nebo více pohledů. Kromě začlenění pohledů a ovládacích prvků může aplikace obsahovat vrstvy animačního jádra ve své pohledové a controller hierarchii.[1]

To, co odlišuje jednu iOS aplikaci od jiné, jsou data, která spravuje (spolu s odpovídající byznys logikou) a jak je prezentuje uživateli.[1]

2.2.3 Hlavní smyčka běhu

Jak už z názvu vyplývá, chod hlavní smyčky probíhá na hlavním vlákně dané aplikace. UIApplication objekt nastavuje hlavní smyčku běhu v době spuštění a používá ji ke zpracovávání všech uživatelsky souvisejících událostí a aktualizací zobrazovacího rozhraní. Toto chování je zajištěno zpracováním uživatelsky souvisejících událostí v pořadí, v jakém byly přijaty. Jako první obdrží událost UIApplication objekt, jenž učiní rozhodnutí o tom, co je potřeba provést. Dotyková událost je obvykle zaslána objektu hlavního okna, který jej odešle do pohledu, v němž došlo k dotyku. Další události mohou tvořit poněkud odlišné cesty prostřednictvím různých aplikačních objektů.[1]

2.2.4 Stavy aplikace

Každá aplikace je po instalaci do iOS zařízení v kterémkoli okamžiku v jednom z těchto pěti stavů: neběžící, aktivní, neaktivní, na pozadí a pozastavená.[1]

Systém každé aplikaci mění stav v závislosti na událostech v něm se dějících. V **neběžícím** stavu jsou aplikace, které ještě nebyly spuštěny nebo byly ukončeny systémem a neblokují žádné prostředky v operační paměti. **Neaktivní** stav má aplikace, jež je na popředí a nepřijímá žádné akce a události; v tomto stavu jsou aplikace většinou krátkou dobu. Aplikace v **aktivním** stavu jsou při normálním používání, běží na popředí a přijímají a reagují na akce a události. Aplikace je ve stavu **na pozadí**, pokud vykonává svůj kód, ale neběží na popředí. **Pozastavený** stav aplikace je na pozadí, nevykonává svůj kód, ale data aplikace jsou stále k dispozici v operační paměti.[1]

Jakmile nastane stav nedostatku operační paměti a je nutné ji uvolnit pro jinou aplikaci, systém může uvolnit paměť, již zabírají pozastavené aplikace, a to bez předchozího upozornění. Většina přechodů mezi stavy je doprovázena odpovídajícím voláním metody v delegátu aplikace. Tyto metody dávají možnost reagovat na změny stavu vhodným způsobem a lze je nastavit v hlavním delegátu aplikace appdelegate.swift (vizte sekci 5.1.1).[1]

2.2.5 Ukončení aplikace

Systém je občas nucen nějakou aplikaci ukončit, což dělá z důvodu uvolnění paměťových prostředků pro jinou, která v danou chvíli vyžaduje paměťové prostředky, a proto by měla být aplikace připravena ke svému ukončení kdykoli, tedy nečekat s ukládáním uživatelských dat nebo prováděním jiných důležitých úkolů.[1]

Pokud je aplikace v pozastaveném stavu, systém ji nijak neupozorní na nutnost ukončení, ale běží-li však na pozadí a není pozastavena, systém ji upozorní a umožní jí provést nastavené úkony před nutným ukončením přes delegáta pomocí funkce applicationWillTerminate (vizte sekci 5.1.1).[1]

Kromě systému může aplikaci ukončit uživatel „ručně“ přes rozhraní multitaskingu; ukončení se chová stejně jako u pozastavené aplikace, tedy ni-

jak neupozorňuje aplikaci o jejím ukončení. Při restartu zařízení to funguje stejně.[1]

2.3 App Store

Nezbytnou součástí všech mobilních aplikací pro mobilní zařízení na iOS je služba App Store. Jediný možný online obchod, ze kterého lze stáhnout a následně nainstalovat všechny možné aplikace pro tento operační systém. Jde to samozřejmě obejít, ve starších verzích odemknutých Jailbreakem, ale obecně platí, že do ne-Jailbreaknutého zařízení je jiným způsobem nelze dostat, až na výjimky přes jiné Apple služby jako iTunes (vizte sekci 2.1.4) nebo Xcode (vizte sekci 2.4), což nese určitou bezpečnostní důvěru v tento operační systém, protože každá nahraná aplikace na App Store prošla celou řadou bezpečnostních kontrol. Programátor to sice může brát jako omezení nebo zdržování, ale koncový uživatel tento přístup jistě ocení.

2.4 Xcode

Xcode je jediný software pro plnohodnotné vyvíjení aplikací na zařízení s operačním systémem iOS. Jedná se o freeware dostupný pouze pro operační systém OS X s open source komponenty. Mezi jeho součásti patří například editor zdrojového kódu, kompilátor, debugger, Interface Bulider, iOS Simulátor a Organizér.

2.4.1 Editor zdrojového kódu

Od editoru lze očekávat funkcionality jako od jiných editorů, například zvýraznění syntaxe dle podporovaného programovacího jazyka. Přidanou hodnotou je kontrola a poukazování chyb nebo upozornění v reálném čase, případně nabídka automatické opravy možného problému či jiné chyby, která muže nastat při sestavení aplikace anebo po jejím spuštění. Jeho nedílnou součástí, bez níž si dnes mnoho programátorů neumí představit práci, je našeptávač k doplnění zdrojového kódu. Pokud metoda obsahuje některé parametry, editor zvýrazní místa kam je jaký parametr nutno vložit.

2.4.2 Interface Builder

Tato součást slouží k představě a také nastavení chování uživatelského rozhraní vytvořené aplikace. Jedná se pohodlný nástroj, pomocí kterého lze spoustu vlastností konfigurovat pomocí výběru přednastavených hodnot kurzorem, ale také přivodit mnoho starostí.

2.4.3 Kompilátor a debugger

Kompilátor a debugger pracují tak, jak je jim předurčeno. Slouží k sestavení aplikace a případnému odhalení problému v ní. Při debuggování lze využít dalších funkcí k informačním účelům, jako zkoumání jednoho nebo více procesů, případně práce s pamětí či sítové komunikace. S jeho pomocí lze vypátrat problémy v kódu, provádět různé výkonnostní analýzy, jako hledání memory leaků a jiných problémů. Memory leak je situace, při které si program vezme část paměti a poté, když už ji nepotřebuje, ji neuvolní a stále ji tak blokuje. Po čase může docházet k nekonečnému zvětšování těchto neuvolněných paměťových prostředků a přetížení celého hardware, po němž nastane pád celé aplikace.

2.4.4 iOS Simulátor

iOS Simulátor je součást, která umožní do vysoké míry simulovat vzhled a chování sestavené aplikace ve všech zařízeních s operačním systémem iOS a otestovat si ji tak. Pro testování implementace této práce je simulátor nápomocen jen do jisté míry, protože jeho velikou slabinou je absence systémové hudební knihovny.

2.4.5 Organizér

Organizér slouží ke správě sestavených aplikací a pro jejich následný export. Pomocí desktopové aplikace iTunes (vizte sekci 2.1.4) lze vyexportovanou aplikaci nahrát do koncového iOS zařízení. Anebo je tu možnost exportu do iTunes Connect (vizte sekci 2.8), pomocí kterého pak lze aplikaci vydat do App Store a nebo ji testovat s pomocí aplikace TestFlight (vizte sekci 2.7). Xcode od své šesté verze podporuje vývoj aplikací pomocí programovacího jazyka Swift.

Xcode od své šesté verze podporuje vývoj aplikací pomocí programovacího jazyka Swift.

2.5 Programovací jazyk Swift

Swift je moderní programovací jazyk využívaný především při vývoji aplikací pro operační systémy iOS, desktopové macOS na Mac zařízeních, watchOS, který běží na chytrých hodinkách anebo tvOS, který obsluhuje chytré televize Apple TV. Jazyk je postaven na programovacích jazycích C a Objective-C, a to bez omezení kompatibility k jazyku C. Swift bere bezpečnou část vzoru programování a přidává k ní moderní zvyky tak, aby bylo programování jednodušší, pružnější a tím efektivní. Swift s čistým štítem, opírající se o zralé

a velmi oblíbené Cocoa a Cocoa Touch frameworky, nabízí příležitost ulehčit představy o fungování vývoji softwaru pro výše uvedené operační systémy.[6]

Cocoa a Cocoa Touch jsou aplikační vývojářské frameworky, které samy o sobě tolik frameworky nejsou, ale pokrývají další frameworky. Cocoa je framework, jenž je sestaven ze dvou frameworku a to Foundation a Appkit, a jako celek slouží především jako framework pro vývoj desktopových aplikací na operačních systémech macOS. Cocoa Touch je pak framework, který zahrnuje Foundation a UIKit frameworky.[7]

Foundation je Framework implementující kořenové třídy, které definují ty nejzákladnější objekty. AppleKit a UIKit jsou frameworky, implementující základní prvky pro vývoj uživatelského rozhraní[7]

Swift byl vyvíjen mnoho let společností Apple, která z něj udělala Open source. Společnost Apple pokládala základní kámen při jeho vývoji, a to během zlepšování kompilátoru, debuggování a samotné vývojové infrastruktury programovacího frameworku. Snažila se také zjednodušit správu paměti pomocí automatického počítání referencí (ARC). Zásobu frameworků postavených na pevných základech Cocoa a Cocoa Touch úplně modernizovala a standardizovala.[6]

Swift je velice blízký vývojářům, kteří byli zvyklí na jazyk Objective-C, protože přejímá čitelnost jeho parametrů a sílu v dynamickém Objective-C modelu. Zajišťuje bezproblémový přístup k existujícím Cocoa frameworkům a mixuje a zároveň podporuje vzájemnou spolupráci s Objective-C kódem. Při budování tohoto společného základu přináší Swift mnoho nových funkcí a sjednocuje tak procedurální a objektově orientované části jazyka.[6]

2.6 Apple Developer Program

Je to jediná cesta z kódu ke koncovému uživateli (zákazníkovi) a zároveň jeden z bezpečnostních prvků společnosti Apple, jak udržet všechny programátory aplikací pro svá zařízení na uzdě. Nutí je tím vyvíjet relativně bezpečné aplikace a v případě špatného chování jim přístup omezit či zastavit a mít tak nad nimi jistou kontrolu. Toto celé řízení navíc není zadarmo, ale na oplátku může programátor svou aplikací také něco vydělat, a to díky prodeji své aplikace v App Store, případně dalších rozšíření a bonusů nebo reklamy v ní. K tomu všemu ještě nabídne svůj systém pro reklamy, statistiky pro a také testovací program TestFlight, který ještě před nedávnem společnost Apple koupila od jiného vývojového týmu.

2.7 TestFlight

Jedná se o systém propojený s aplikací na iOS zařízení, díky níž muže programátor aplikaci testovat před samotným uvedením do App Store. V online nástroji iTunes Connect (vizte sekci 2.8) si vytvoří svoji budoucí aplikaci pro

App Store a nastaví ji pro testování. Nabízí dvě možnosti testování, a to interní a externí. Do interního testování lze zapojit všechny uživatele vývojového týmu, do externího pak jakéhokoli uživatele s Apple účtem. Jedná se o základní uživatelský účet, bez kterého se žádný uživatel iOS zařízení neobjede, protože by si bez něj nedokázal stáhnout žádnou aplikaci z App Store.

Po nahrání aplikace do iTunes Connect přes Xcode a jeho komponentu Organizér (vizte sekci 2.4.5) lze nahranou verzi aplikovat pro testování. Každá nová verze se uživateli interního testování objeví v aplikaci TestFlight s možností nainstalování nebo aktualizace testované aplikace. Externímu uživateli testování je třeba každou novou verzi popsat, co v ní bylo upraveno za chyby, případně co je v této verzi nového. Aplikace se poté nainstaluje do přístroje testujícího uživatele, ale má to své omezení, a to 60 dní, po jejichž uplynutí se možnost použití aplikace k testování ukončí a aplikaci uživatel již nespustí. Celý tento proces je ale nástrojem pro testování v delším časovém úseku, protože má své časové prodlevy. Nejrychlejší nástroje pro testování jsou iOS Simulátor (vizte sekci 2.4.4), napojené zařízení k aplikaci Xcode, anebo exportování aplikace pomocí Organizéru do souboru, který následně lze nahrát do zařízení přes aplikaci iTunes na desktopovém zařízení (vizte sekci 2.1.4).

Jedná se o systém propojený s aplikací na iOS zařízení, díky níž může programátor aplikaci testovat před samotným uvedením do App Store. V online nástroji iTunes Connect (vizte sekci 2.8) si vytvoří svoji budoucí aplikaci pro App Store a nastaví ji pro testování. Nabízí dvě možnosti testování, a to interní a externí. Do interního testování lze zapojit všechny uživatele vývojového týmu, do externího pak jakéhokoli uživatele s Apple účtem. Jedná se o základní uživatelský účet, bez kterého se žádný uživatel iOS zařízení neobjede, protože by si bez něj nedokázal stáhnout žádnou aplikaci z App Store.

Po nahrání aplikace do iTunes Connect přes Xcode a jeho komponentu Organizér (vizte sekci 2.4.5) lze nahranou verzi aplikovat pro testování. Každá nová verze se uživateli interního testování objeví v aplikaci TestFlight s možností nainstalování nebo aktualizace testované aplikace. Externímu uživateli testování je třeba každou novou verzi popsat, co v ní bylo upraveno za chyby, případně co je v této verzi nového. Aplikace se poté nainstaluje do přístroje testujícího uživatele, ale má to své omezení, a to 60 dní, po jejichž uplynutí se možnost použití aplikace k testování ukončí a aplikaci uživatel již nespustí. Celý tento proces je ale nástrojem pro testování v delším časovém úseku, protože má své časové prodlevy. Nejrychlejší nástroje pro testování jsou iOS Simulátor (vizte sekci 2.4.4), napojené zařízení k aplikaci Xcode, anebo exportování aplikace pomocí Organizéru do souboru, který následně lze nahrát do zařízení přes aplikaci iTunes na desktopovém zařízení (vizte sekci 2.1.4).

2.8 iTunes Connect

Jak bylo již uvedeno v předchozí podkapitole, jedná se o online nástroj, pomocí kterého lze nastavit aplikaci pro testování. Nabízí ale další nástroje, a to správu všech aplikací vytvořených pro testování anebo již vydaných aplikací v App Store, dále statistiky aplikací, informace o prodejích a trendech, platební a finanční reporty, nabídku prodeje reklam v aplikacích a správu uživatelů a rolí.

Kapitola 3

Analýza

3.1 Analýza potenciálních uživatelů

Hudební aplikace jako výsledek práce bude sloužit především pro ulehčení výuky nebo tréninku tance. Mimo to nabídne i podobné funkce, které uspokojí běžného uživatele hudebních aplikací. Z těchto důvodu je v první řade nutné myslet na trenéry či tanečníky a až poté na uspokojení širší veřejnosti. Uživatelské role můžeme tudíž rozdělit na trenéra, tanečníka a všeobecného posluchače.

Trenérovy potřeby spočívají v nastavení fronty přehrávání skladeb některého tance po určitou dobu, po níž se má plynule přejít na jiný tanec nebo stejný v rychlejším tempu. Další potřeba je nastavení při tréninku, tzv. practise, při kterém si tanečníci zkoušejí své sestavy v soutěžním režimu, kdy trénují tance střídavě za sebou a dokola jen s nutnými přestávkami pro odpočinek. Při tomto tréninku trenér dohlíží na tanečníky a upozorňuje je na časté chyby, případně je motivuje k lepším výsledkům.

Tanečníkovy potřeby spočívají v nastavení poslechu určitého tance a jeho zkoušení, případné zvyšování tempa. Dále, jak už bylo popsáno v potřebách trenéra, jsou nedílnou součástí potřeb tanečníka tréninky v režimu practise.

Požadavky všeobecného posluchače jsou především poslech nastavených playlistů, ale muže být náročnější a chtít je po určité době mezi sebou mixovat po určité době poslechu, např. po 20 minutách střídat playlisty s rozdílnými žánry. Nabízí se zde i možnost automatického mixování radiových streamů.

3.2 Analýza požadavků a případů užití

Tato kapitola se zabývá sestavením analýzy funkčních či nefunkčních požadavků a také určení případů užití, které by aplikace měla uspokojit.

3.2.1 Funkční požadavky

3.2.1.1 Seznam playlistů

Uživatel si musí být schopen jednoduchým způsobem vypsat všechny svoje playlisty, které si vytvořil a synchronizoval za pomocí desktopové knihovny iTunes.

3.2.1.2 Výpis skladeb playlistů

Skladby z playlistu si bude moci uživatel vypsat a vybranou poslechnout.

3.2.1.3 Vytvoření mixu playlistů

Uživatel musí mít možnost vytvoření tzv. mixu, pomocí nichž bude moci kombinovat poslech playlistu.

3.2.1.4 Nastavení mixu

U každého vytvořeného mixu bude uživatel moci nastavit jeho parametry a případně u každé položky zvlášť.

3.2.1.5 Přehrání mixu

Každý vytvořený a funkčně nastavený mix bude moci uživatel ihned přehrát.

3.2.2 Jednoduchá správa přehrávaného mixu

Přehrávaný mix bude moci uživatel pauzovat, spouštět si jej na pozadí a ovládat tak přehrávání na uzamčené obrazovce displeje.

3.2.3 Nefunkční požadavky

3.2.3.1 Aktuálnost

Z důvodu rychlého vývoje každoroční nové verze iOS je důležité myslet na aktuálnost samotné aplikace. Z bezpečnostních důvodů, na kterých si zakládá i samotný vývoj iOS, je doporučené aplikaci vyvíjet podle tohoto trendu a snažit se společně se jeho vývojem dávat důraz na její aktuálnost. Lze tak docílit i vetší dostupnosti pro všechny iOS zařízení s velikým důrazem na bezpečnost. Předpokládá se tedy, že bude aplikace vyvinuta pro aktuální verzi

iOS 10, která poskytuje kompatibilitu s až 4 roky starým zařízením a zároveň garantuje nejvyšší možnou bezpečnost na této platformě.

3.2.3.2 Použitelnost

Aplikace by mela být použitelná na co nejširším portfoliu iOS zařízení. Proto je nutné myslet na různá rozlišení těchto zařízení.

3.2.3.3 Přehlednost

Aplikace by mela působit čistým a přehledným dojmem. Uživatel by mel být schopen na všechny její možnosti přijít intuitivně.

3.2.3.4 Spolehlivost

Aplikace by měla být spolehlivá a reagovat tak na všechny možné vstupy uživatele s eliminací jakýchkoli možných pádu.

3.2.3.5 Zvyklost

Aplikace by měla myslet na všeobecné zvyky uživatelů iOS, zejména v pohledu na uživatelské rozhraní. Určitým zvykem je i způsob nahrávání skladeb a správa hudební knihovny přes iTunes.

3.2.3.6 Zpětná vazba od uživatele

Uživatel by mel být schopen dát zpětnou vazbu vývojáři ohledně funkčností a chyb aplikace.

3.2.4 Případy užití

3.2.4.1 Spuštění vybrané skladby

Uživatel si chce poslechnout vybranou skladbu z určitého playlistu ze systémové hudební knihovny.

3.2.4.2 Přehrávání playlistu

Uživatel si chce přehrát vybraný playlist ze systémové hudební knihovny.

3.2.4.3 Vytváření a editování mix

Uživatel si bude chtít vytvořit mix z playlistu, který si bude moci pojmenovat a nastavit dle svých představ.

3.2.4.4 Nastavení mixu

Nastavení mixu potřebuje uživatel provést globálně, aby aplikace jednoduchý požadavek obsloužila rychle. Na druhou stranu chce také občas nastavit různé parametry každé položce mixu zvlášť.

3.2.4.5 Přehrávání mixu

Přednastavené mixy bude moci uživatel přehrávat přímo z jejich seznamu a bude moci jednoduchým způsobem přepnout na jiný.

3.2.4.6 Správa přehrávání

Uživatel potřebuje mít alespoň základní kontrolu nad ovládáním přehrávače (hlasitost, pauza a opětovné přehrávání).

Aplikace by tedy obecně měla být schopna načítat interní hudební knihovnu, hlavně její playlisty, dále vytvářet mixy těchto playlistů a dovolit jejich přehrávání dle individuálně nastavených parametrů. Jako bonus by mohla přehrávat internetové streamy rádií a dovolit mixovaný poslech těchto streamů tak jako by to byl mix playlistů.

3.3 Analýza jiných hudebních přehrávačů

Předchozí kapitola analyzuje komplexní funkčnost a požadavky na vytvářenou aplikaci, proto nyní přichází na řadu průzkum jiných aplikací; jestli již podobná aplikace neexistuje, případně jestli neexistuje aplikace, která tento problém řeší například jiným způsobem. Je vhodné dát také prostor zajímavým hudebním aplikacím z hlediska uživatelského rozhraní zejména při ovládání poslechu.

Vyhledávání aplikací pro analýzu probíhalo dle těchto kritérií:

1. vyhledávání dle nejpřesnějších klíčových slov (dance music, dance music player, dance music manager, music manager, plan music, hudební přehrávač, hudební manažer, taneční hudba, plánovač hudby, apod.)
2. obecný předpoklad zaměření aplikace - hudební aplikace pro podporu výuky tance
3. hudební aplikace pro mixování skladeb, nastavování jejich délky
4. jakákoli zajímavá hudební aplikace s přehledných a jednoduchým UI

3.3.1 Audio přehrávače

Během průzkumu iOS aplikací bylo nalezeno několik zajímavých aplikací, ale žádná neuměla to, co se očekává jako výsledek této práce. Následuje rozbor vybraných aplikací.

3.3.1.1 Standardní přehrávač každého iOS zařízení "Hudba"

Standardní hudební aplikace každého iOS zařízení, která pracuje se systémovou hudební knihovnou a především ve spojení s placenou službou Apple Music. Je to služba, která nabízí neomezený poslech skladeb z nabídky iTunes obchodu za paušální cenu. Aplikace dále nabízí filtrování skladeb podle umělců, alb, skladeb, hudebních videí, žánrů, skladatelů a kompilací. Zvláště je zde také nabídka pro zobrazení playlistů. Součástí je i poslech vybraných internetových rádio streamů, k němuž je ale nutností mít aktivní účet Apple Music. Jako bonus aplikace nabízí hudební sociální síť Connect, do které dávají své příspěvky, povětšinou s obrázky nebo i videi, pouze známé osobnosti z hudebního průmyslu. Aplikací se lze inspirovat filozofií knihovny, prostým a přehledným designem anebo přehrávačem skladeb v této knihovně. Jediné v čem může být aplikace otravná je nabízení obsahu, který lze spustit pouze s členstvím Apple Music.

3.3.1.2 Listen

Hudební aplikace, která je kromě práce se systémovou hudební knihovnou zaměřena především na práci s gesty na displeji. Gesta jsou v přehrávači na-

stavena tak, že není nutností úplná přesnost, stačí jen pomyslně od středu táhnout prstem do osmi různých směrů, čímž se provede určitá akce. Jedna z akcí je například zajímavý způsob sdílení aktuálně přehrávané skladby, a to vyfocením obrazovky a možností sdílení tohoto snímku všemi dostupnými komunikačními aplikacemi.

3.3.1.3 MusixMatch

Hudební aplikace zaměřena na zobrazení textu písniček; automaticky je stahuje z vlastní online databáze, pokud skladbu pozná. Přes ni je dále možné poslouchat placenou hudební online knihovnu Spotify; také používá gesta pro přepínání hudby. Samozřejmostí je práce se systémovou hudební knihovnou.

3.3.1.4 Groove

Hudební aplikace s funkčnostmi podobnými standardní hudební aplikace, která nabízí výpis interpretu, alb, skladeb a playlistu ze standardní knihovny. K tomu ale nabízí zajímavý přehrávač, který lze také ovládat gesty na displeji. Možnost sdílení vyfocené obrazovky zde také nechybí. Pro registrované uživatele nabízí ještě například statistiku jejich poslouchání, na základe níž, nejspíše na tomto základ, začne nabízet poslech sama.

3.3.1.5 Music D/L

Hudební přehrávač, který zobrazuje hudební žebříček z iTunes obchodu podle nastavené lokalizace. Téměř každou skladbu zvládne vyhledat a nabídnout k poslechu nebo stažení. Vzhledem k době trvání načítání nebo stahování lze těžce hledat odpověď, odkud skladby získává. Je také možné, že je překódovává z videí, a proto je tento proces tak zdlouhavý. Ze stažených souborů lze skládat playlisty a vytvořit si tak slušnou hudební knihovnu. O legálnosti stahování zde ale nelze polemizovat. Jedná se o aplikaci s jednoduchým uživatelským rozhraním vytvořeným pomocí horizontálního spodního menu. Jako bonus například nabízí časovač pro přehrávání, po kterém poslech ukončí.

3.3.1.6 Player – Jamn

Hudební aplikace, která kromě práce se systémovou knihovnou nabízí analýzu vybrané skladby, již poté rozebere na akordy, které lze zahrát např. na kytaře.

3.3.1.7 iMusic

Funkcemi klasický, po stránce uživatelského rozhraní velice přehledný hudební přehrávač, nejspíše díky posouvání se na jinou sekci indikovaného spodním ukazatelem v jaké sekci se uživatel nachází. Něco navíc nabízí snad jen přepínání písničky pomocí třepání se zařízením a vcelku nezvyklá možnost pauzy, a to položením zařízení displejem směrem dolů.

3.3.1.8 Beat

Přehrávač, který kromě procházení systémové hudební knihovny pomocí posouvání na jinou sekci se spodním indikátorem aktuální sekce nabízí širší možnosti v nastavení vzhledu anebo právě možnosti zobrazení a skrytí zobrazovaných sekcí. Dále pak využívá snad všechny druhy ovládání gest na displeji, mezi kterými nechybí ani gesto pro akci sdílení informace o aktuální přehrávané skladbě.

3.3.1.9 Plan Player L

Tato aplikace se asi nejvíce podobá snaze přehrávání skladeb podle přednastavených hodnot a muže se nejblíže podobat aplikaci, která má být výsledkem této práce. Nicméně pokud se uživatel po delší době zorientuje, jaké jsou možnosti této hudební aplikace, zjistí, že jde v podstatě o nastavení přehrávání vybraných skladeb, různé upravování této fronty, a pak ještě jeden časovač, po němž se každá skladba přepne a spustí se stejný časovač. Asi jediným bonusem je možnost připojení k FTP serveru, cloud úložišti Dropbox a nebo ještě historickému SkyDrive, který nemůže fungovat. Zkoušet to, z bezpečnostního hlediska a vzhledem k celému dojmu z aplikace nedoporučuji.

3.3.2 Video přehrávače

Mezi hudebními přehrávači se často objevují aplikace pracující s videi z YouTube. Z tohoto důvodu nejde ani tak o hudební audio přehrávače, ale video přehrávače. Následuje rozbor vybraných video přehrávačů.

3.3.2.1 Music Player

Jeden z video přehrávačů, který má sice obvyklé funkce vyhledávání, přehrávání a přidávání videí z Youtube do playlistu, co se ale týče uživatelského rozhraní, působí hodně čistě a jasně, a tedy velice přehledně.

3.3.2.2 VLC

Po přehrávači YouTube videí je třeba zmínit i tuto aplikaci. Nespolupracuje sice s YouTube či jiným video portálem jako mnoho jiných videopřehrávačů, ale je to jeden z nejoblíbenějších video přehrávačů. Nabízí prohlížení a přehrávání videí z cloudových úložišť, stahování videí z vloženého odkazu, přehrávání online audio i video streamu, zobrazení dat z jejich metainformací o názvu skladby a jejího interpreta. Přehrávač podporuje volby užitečné hlavně pro přehrávání videa a práci s několika gesty na displeji.

3.3.3 Ostatní přehrávače

Mezi hudebními aplikacemi je mnoho podobných, které ve většině případů nabízejí žebříčky iTunes obchodu, možnost vyhledání skladby na YouTube nebo podobných zdrojů, její následné přehrávání, v některých případech stažení, a přidávání do vlastního playlistu. Ale ve většině případů jde o aplikace, které obsahují spoustu reklamních bannerů, anebo každá druhá funkce vyžaduje koupi placené verze. Lze u nich polemizovat, do jaké míry rovnou nenutí uživatele k jejímu odstranění, než aby dokázaly přilákat svými funkcemi uživatele k placené verzi nebo placenému doplňku. Většina z nich často během několika měsíců z App Store úplně mizí. V některých případech se aplikace zdají být z bezpečnostního hlediska nevhodné, z důvodu zajišťování uživatelských údajů, protože nabízejí připojení s nutnou autentizací do různých služeb nebo e-mailu a jsou si v mnoha pohledech velice podobné.

Kapitola 4

Návrh

Z uživatelských požadavku vyplývá použití následujících funkcionalit. V první řade je nutné vypsat systémovou knihovnu s playlisty a jejich skladbami. Poté je nutno tyto skladby odzkoušet a přehrát. Nyní přichází na řadu vytváření uživatelských mixů pro poslech kombinací playlistů a nastavení časových požadavků k tomuto poslechu.

4.1 Návrh uživatelského rozhraní

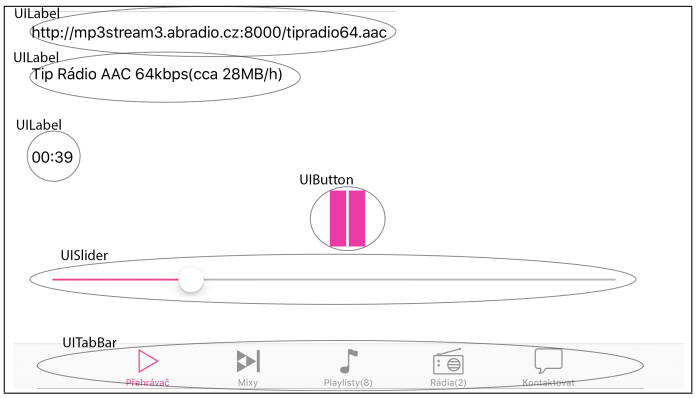
Návrh uživatelského rozhraní vychází z analýzy požadavků uživatele na aplikaci, jiných hudebních přehrávačů, ale také velkou měrou z doporučených postupů „iOS Human Interface Guidelines“, jimiž je inspirována celá tato podkapitola.

4.1.1 Pilíře iOS stylu

Styl uživatelského rozhraní iOS dává důraz na tři pilíře, konkrétně jasnost, úctu a hloubku. **Jasnost** by měla být komplexně v celém systému tak, aby byl pro všechny stejně dobře čitelný a zdůrazňoval ty správné prvky s cílem zprostředkovat interakci od jeho uživatelů, i když se změní např. velikost textu. **Úcta** by měla být k uživateli, aby mu všechny aplikace ve všech stránkách napomáhaly k pochopení obsahu a zdůrazňovaly jej, neodváděly pozornost a nesoutěžily s ním. **Hloubka** se má ukazovat v porozumění a zprostředkování správné hierarchie obsahu, dále ve zvýšení potěšení a umožnění přístupu uživatele k funkcím.[8]

4.1.2 Zásady iOS stylu

iOS má také své zásady, které je důležité mít na paměti pro udržení identity aplikace, aby byly její dosah a dopad maximální; jsou to estetická integrita, konzistence, přímá manipulace, zpětná vazba, metafory a řízení uživatele.[8]



Obrázek 4.1: Přehrávač vyvíjené aplikace PlayMix s návrhem použití standardních UI prvků

Estetickou integritou je myšlen takový vývoj aplikace, kdy jsou její vzhled a chování uzpůsobeny jejími funkcemi. Aplikace zaměřená na vážné činnosti by tedy měla mít jemnou nenápadnou grafiku orientovanou na standardní ovládací prvky s předvídatelným chováním, kdežto třeba hra by měla přinášet originální vzhled i prostředí, slibovat zábavu a vzrušení a vtahovat uživatele do děje s cílem více a více ji objevovat.[8]

Konzistence souvisí s implementací známých standardů za pomoci běžných systémových prvků, rozhraní, ikon, textových stylů, jednotné terminologie a chování samotné aplikace tak, jak jsou uživatelé iOS zvyklí.[8]

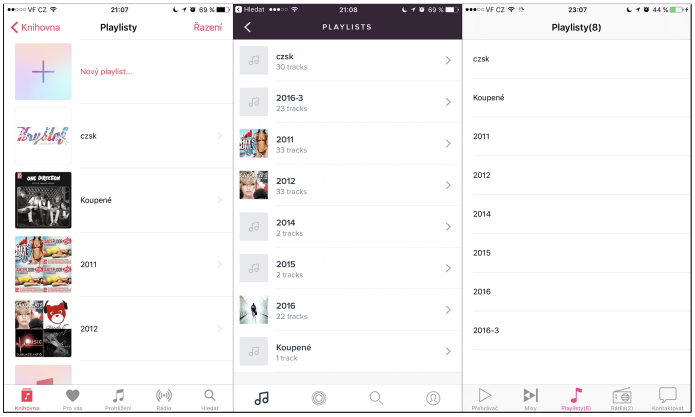
Použití konzistence v návrhu aplikace na obrázku 4.1.

Přímou manipulací by měla každá aplikace být schopna reagovat, ovlivňovat a zapojovat uživatele do výsledného obsahu na obrazovce, např. využitím rotace a gest uživatele.[8]

Zpětná vazba je důležitá z důvodu informace o reakci aplikace na pokyny uživatele a následném zpracování požadovaných pokynů; v první řadě je kladen důraz na interaktivní prvky na obrazovce jejich zvýrazněním. Dále může být zpětná vazba doprovázena zvukem, animací nebo vibrací, aby uživatele zabavila při čekání na vyřízení pokynu, případně naznačila vážnost jeho výsledku.[8]

Metafora je jednou z důležitých vlastností chování aplikace vedoucích k jejímu rychlému pochopení. Je dobré, když se chová podobně jako již známé ostatní aplikace, nejlépe se stejným zaměřením; toho je nejčastěji dosaženo využitím systémových prvků, jako jsou tabulkový výpis, tab bar menu, přepínače, posuvníky apod.[8] Použití metafory v návrhu aplikace na obrázku 4.2.

Aplikace by měla navrhovat postup nebo varovat před nebezpečnými úkony, ale určitě **nerozhodovat za uživatele**. Ty, které najdou správnou rovnováhu mezi možnostmi uživatele a snahou se vyhnout nežádoucím výsledkům, mají



Obrázek 4.2: Metafora v porovnání standardního hudebního přehrávače Hudba, přehrávače MusixMatch a vyvíjeného Playmix

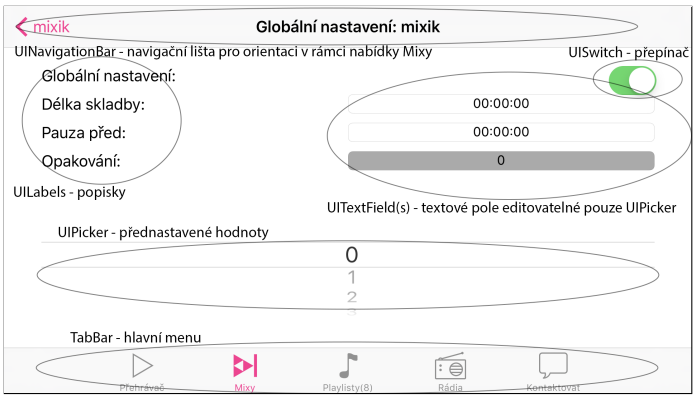
vyhráno. Z aplikace má mít každý uživatel pocit, že ji má pod kontrolou za pomoci známých interaktivních prvků, aby dokázal předvídat její chování. Měl by mít možnost potvrzovat destruktivní akce a každou operaci zrušit, a to i během jejího průběhu.[8]

4.1.3 Rozhraní iOS

Většina iOS aplikací je vyvíjena s pomocí komponentů z UIKit a programového frameworku, který definuje společné prvky uživatelského rozhraní. Tyto komponenty umožňují aplikacím dosahovat konzistentního vzhledu, i když nabízejí vysokou úroveň přizpůsobení. Velikou výhodou jsou jejich použitelnost na téměř každém iOS zařízení a automatická aktualizace, i když se samotný systém sám mění. Takové prvky se řadí do 3 základních skupin na bary, pohledy a řídící prvky.[8]

Pomocí barů (bars), které obvykle obsahují tlačítka či jiné prvky pro možnou činnost a předávání informací, je aplikace schopná uživateli vyjasnit jeho aktuální polohu v ní. Pohledy (views), kde se nacházejí hlavně texty, animace a jiné grafické a interaktivní prvky, slouží primárně k zobrazení hlavního obsahu aplikace; většinou také umožňují chování jako např. rolování, vložení, odstranění či uspořádání obsahu. Řídicí prvky (controls) slouží primárně k řízení a předávání informací; jsou to např. tlačítka, přepínače, textová pole a ukazatele pokroku.[8]

Rozhraní aplikace se bude ve většině zobrazeních skládat ze tří základních částí; jedná se o UITabBar, který bude sloužit k hlavní navigaci vyvíjené aplikace, dále UIView, ve kterém bude probíhat zobrazování hlavního obsahu aplikace a poslední částí je UINavigationBar, který bude sloužit k orientaci v rámci hlubšího poznávání obsahu jedné z hlavních kategorií. Na obrázku 4.3 lze vyčíst toto rozdělení, jasněji jdou rozpoznat části UITabBar a UINavi-



Obrázek 4.3: Rozhraní aplikace Playmix - nastavitelné parametry mixu

gation Bar, UIView je celý prostor uprostřed, ve kterém jsou další obsahově orientované prvky.

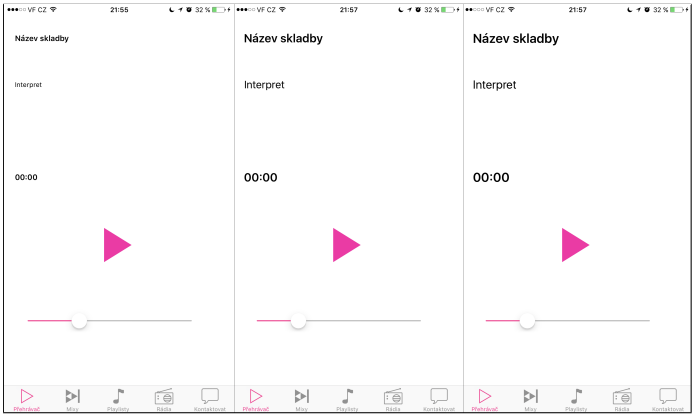
4.1.4 interakce

4.1.4.1 Přístupnost

iOS obsahuje mnoho bezbariérových nástrojů pro uživatele se ztrátou zraku či sluchu, případně jinými postiženími. Většina aplikací postavených na knihovně UIKit může být s menší snahou zpřístupněna postiženým uživatelům a poskytovat tak stejný zážitek z aplikace úplně všem, čehož může docílit například pomocí textových popisků u obrázků, ikon či jiných prvků rozhraní. Tyto popisky nejsou na rozhraní viditelné, ale mohou být použity právě s funkcí VoiceOver, která tyto prvky nahlas popisuje.[8]

Dále by měla aplikace reagovat na preference nastavené uživatelem, jako jsou například velikost a tučnost textu či omezení pohybu. Tyto změny je doporučeno si otestovat přenastavením systému dle různých hodnot. Další změny lze vypozorovat také změnou kontrastu, inverzí barev, snížením průhlednosti apod.; systém také podporuje implementaci skrytých titulků a zvukových popisů. Skryté titulky umožní neslyšícím vnímat mluvený dialog a další zvukové obsahy videa, zatímco zvukové popisky pro změnu popisují důležitý obsah videa pro zrakově postižené.[8]

Aplikaci bude využívat textové popisky a dynamickou změnu velikosti písma, tak aby byla uzpůsobitelná pro uživatele z různými nároky na vzhled uživatelského rozhraní. Na obrázku 4.4 je zobrazen návrh přehrávače, který je zobrazen ve třech různých zobrazeních se změnou písma, která je vyvolána systémovým nastavením přístupnosti pro uživatele.



Obrázek 4.4: Přehrávač aplikace Playmix v různých velikostech písma nastaveného v systému

4.1.4.2 Audio

Se zvukem může uživatel manipulovat prostřednictvím tlačítek hlasitosti, přepínače tichého režimu, ovládacích tlačítek na sluchátkách anebo posuvníku hlasitosti na obrazovce. Mnoho doplňků třetích stran má své vlastní ovládací prvky. Zvukový výstup může být vydáván přes interní nebo externí reproduktory či sluchátka, a to i bezdrátově pomocí Bluetooth nebo AirPlay.[8]

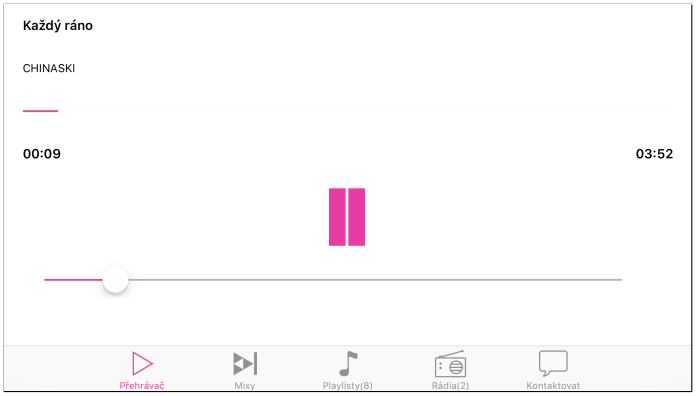
Při ovládání tlačítky hlasitosti uživatel očekává, že změny zvuku ovlivní chování hlasitosti celého přístroje, včetně hudby a zvukových efektů aplikací. Jedinou výjimkou je zde hlasitost vyzvánění, kterou lze měnit pouze přímo v nastavení operačního systému.[8]

Pomocí systémového pohledu hlasitosti nalezne aplikace nejlepší způsob k jejímu ovládání; je zde k dispozici i ovládací prvek pro přesměrování zvukového výstupu. Implementace je pomocí MPVolumeView; s tímto prvkem se při vývoji aplikace počítá (vizte obrázek 4.5), a to z důvodu nedvojení ovládání hlasitosti, ale používání jen systémové úrovně hlasitosti.

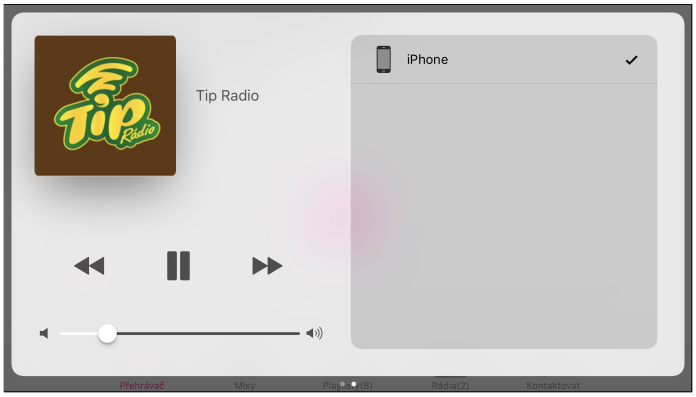
Uživatel často ovládají zvukový výstup z vnějšího rozhraní aplikace, jako je Control Centrum (vizte obrázek 4.6), nebo ovládacími tlačítky na sluchátkách bez ohledu na to, je-li aplikace na popředí nebo na pozadí. Reakce na ovládání zvuku je zcela v pořádku v případě, že aplikace aktivně přehrává v čistě audiosouvisejícím kontextu nebo pokud je připojena k AirPlay zařízení.[8]

4.1.4.3 Datový vstup

Poklepávání na položky uživatelského rozhraní nebo vkládání dat klávesnicí může být zdlouhavý proces. Pokud aplikace celý proces pro uživatele zpomaluje tím, že vyžaduje mnoho vstupů před začátkem využívání, může jej odradit a jednoduše jej přinutit k opuštění. [8]



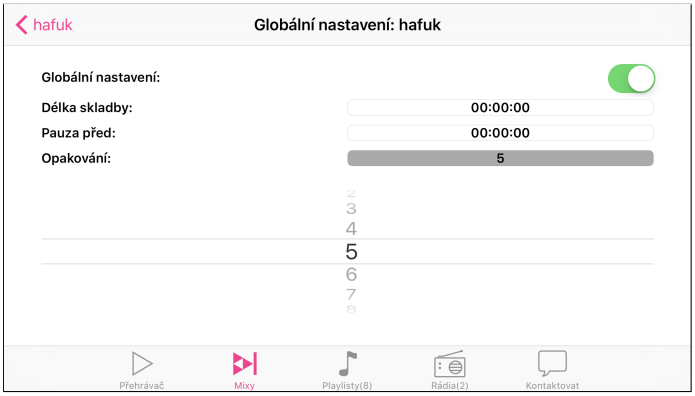
Obrázek 4.5: Přehrávač aplikace Playmix se systémovým MPVolumeView



Obrázek 4.6: Control Centrum iOS zařízení

Vkládání datového vstupu by mělo být co nejefektivnější; např. místo textového pole používat nástroj pro výběr nebo tabulku, protože není nic jednoduššího než si vybrat z předdefinovaných hodnot. Dále jsou to třeba hodnoty, které lze generovat automaticky nebo se souhlasem uživatele, například vkládání aktuálního času a data, nebo informací jako jsou kontakty nebo data z kalendáře uživatele. Aby to celý proces opravdu urychlovalo, je nutné mít relevantní předdefinované hodnoty.[8]

Aplikaci se navrhuje využívání přednastavených hodnot pro nastavení parametrů mixů, jedná se o parametry pro časovou délku skladby, délku pauzy před další skladbou, počtu opakování tohoto nastavení a také pro možnost globálního nebo lokálního nastavení parametrů vybraného mixu4.7. Globální bude sloužit pro urychlení nastavení, tak aby se nastavily hodnoty globálně pro každý playlist v mixu. Lokální nastavení bude sloužit k tomu, že vyžaduje každý playlist z mixu jinou časovou délku skladby, délku pauzy mezi skladbami a počet opakování tohoto playlistu.



Obrázek 4.7: Nastavení parametrů pro mix u vyvíjené aplikace Playmix

4.1.4.4 Uvítací obrazovka

Úvodní obrazovka při spouštění je první příležitostí aplikace, jak rychle a zábavně seznámit nové uživatele s jejím obsahem a naopak také způsob, jak znovu navázat spojení s uživateli, kteří ji již v minulosti použili. Je důležité, aby aplikace spouštěcí obrazovku obsahovala; ta kromě vytváření prvního dojmu uživatele může zakrývat počáteční přípravu k zahájení svého plného provozu. Zároveň je ale nutné plný provoz nijak nezdržovat, tedy nedávat před něj zbytečné nabídky a instrukce.[8]

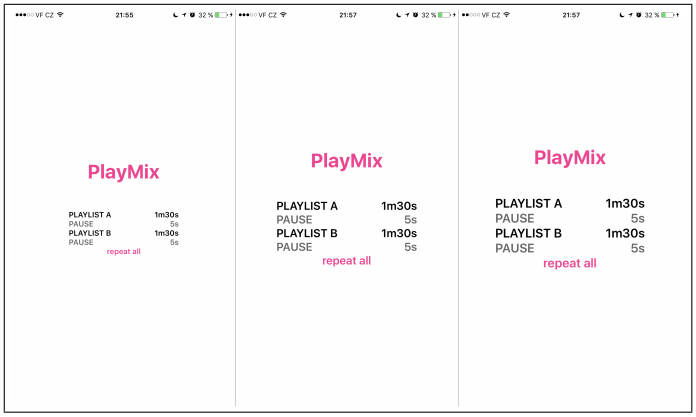
Instrukce, jako rychlé seznámení s celou aplikací, jsou jistě dobrou cestou pro nové uživatele, nicméně by měly být stručné, případně s možností svého přeskočení a neměly být nabízeny již vracejícímu se uživateli. Zároveň by se měla aplikace vyhnout dotazu na souhlas s podmínkami ke svému využívání; tyto podmínky by měly být k potvrzení již před stažením aplikace v App Store.[8]

Dobrým ukazatelem úspěšnosti je ohodnocení aplikace, které slouží jako užitečná zpětná vazba. Aplikace by o toto hodnocení neměla žádat často a hlavně ne brzy, v žádném případě ne při prvním spuštění. Je dobré dát uživateli čas, aby si o ní utvořil relevantní úsudek.[8]

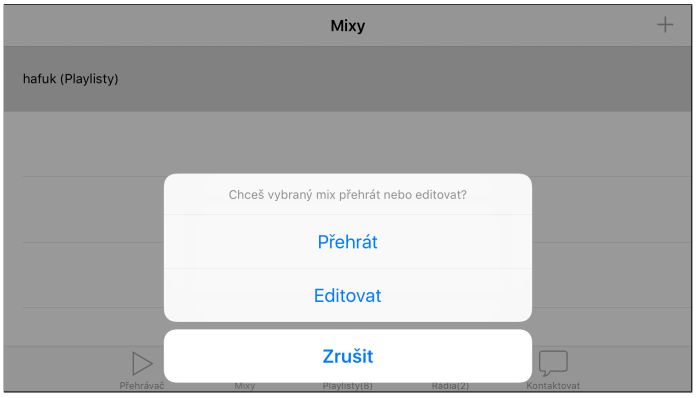
Úvodní obrazovka aplikace bude naznačovat její hlavní funkci a to mixování playlistů s možností nastavování parametrů tohoto mixovacího procesu (vizte obrázek 4.8). Neměla by tak ani snižovat zájem opakovaně se vracejícímu uživateli, protože je to obrazovka, která se zobrazí na opravdu krátkou dobu, jež je nezbytně nutná k načtení aplikace.

4.1.4.5 Modality

Modality systému vytvářejí v uživateli větší pozornost pro důležité zprávy, volby a akce tak, aby mu zabránily pokračovat, dokud něco nepotvrdí, nezvolí správnou volbu akce pro další činnost, případně aby dokázal včas zrušit



Obrázek 4.8: Uvítací obrazovka vyvíjené aplikace Playmix

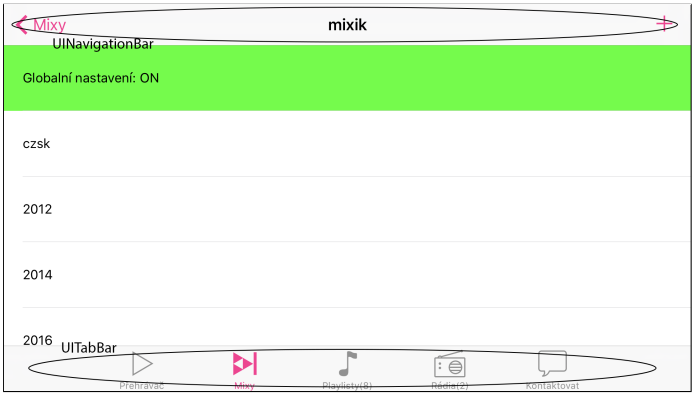


Obrázek 4.9: Jedna z modalit vyvíjené aplikace Playmix

činnosti nežádoucí. iOS poskytuje tyto typy modálních zážitků: action sheets, alerts a activity views. Pokud se některý z těchto prvků objeví na obrazovce, uživatel musí učinit volbu klepnutím na tlačítko, jestli akci chce provést, případně vybrat z několika možností či celou akci zrušit.[8]

Modalita může zabírat celou obrazovku nebo její část a chová se jako nadřazený pohled tomu současnému. Typicky obsahuje volbu dokončení, možnost svého zrušení pomocí tlačítka a také, pokud to je vhodné, popis pro rychlou orientaci uživatele. Používání modalit by mělo být minimální, tedy použito pouze tehdy, pokud má aplikace uživatele na něco upozornit, a to v případě splnění nějakého úkolu, rozšířených možností v používání aplikace nebo při ukládání důležitých dat.[8]

Modality budou v aplikaci sloužit při žádosti přístupu do systémové hudební knihovny, dále u mixů playlistů při jejich správě (vizte obrázek 4.9). Jedná se o místa, kde bude docházet k editaci vybraného mixu, proto je důležité upozornit uživatele na tyto změny.



Obrázek 4.10: Navigační prvky vyvíjené aplikace Playmix

4.1.4.6 Navigace

Obecně mají uživatelé tendenci mít povědomí o navigaci v aplikaci. Úkolem aplikace je provést takovou navigaci, která podporuje strukturu a účel, ale zároveň se chová přirozeně, povědomě a rozhraní by nemělo nijak dominovat a odvádět pozornost od obsahu. U iOS existují tři známé navigační styly; jedná se o **hierarchickou**, **plochou** a **obsahově nebo zážitkově řízenou** navigaci.[8]

První z nich je zaměřen na výběr jedné z možností na obrazovce, čímž se postupně dosáhne cíle; v případě přechodu na jiné místo je nutné se po krocích vracet nebo začít od začátku a vybírat jiné možnosti dle očekávaného cíle. Plochá navigace cílí na přepínání mezi kategoriemi. Obsahově nebo zážitkově orientované navigace jsou pro volný pohyb v obsahu, kde k jednomu cíli může existovat několik různých možností a samotný obsah v podstatě definuje navigaci. Aplikace velice často všechny tyto styly kombinují, a tak kromě ploché navigace lze třeba k cíli pokračovat hierarchickou. Nicméně stále platí, že by měla navigace poskytovat jasnou cestu bez ohledu na svůj styl, a musí proto být její cesta logická, předvídatelná a snadno sledovatelná.[8]

Jak už byl uvedeno v návrhu základního rozhraní (vizte sekci 4.1.3), aplikaci se navrhují dva základní navigační prvky, a to hlavní UITabBar, který bude sloužit k základní orientaci v rámci aplikace a dále UINavigationBar, který bude sloužit k orientaci v hlubším kontextu jejího obsahu. Orientace v aplikaci je tedy navržena kombinací ploché a hierarchické navigace (vizte obrázek 4.10).

4.1.4.7 Vyžadování povolení

I když uživatelé ocení pohodlí při používání svých osobních údajů v aplikacích, rádi nad nimi mají kontrolu; je tak nutné mít povolení od uživatele při jejich používání. K informacím, ke kterým musí uživatel udělit povolení, patří

aktuální poloha, kontaktní údaje, informace z kalendáře, z hudební knihovny a fotografie; mimo osobní údaje vyžaduje povolení také například mobilní internet. Využívání těchto zdrojů musí mít reálný důvod a jeho pochopení u ze strany uživatele. Povolení přístupu k těmto zdrojům by měly být žádány až před jejich faktickým využitím; pokud je využití oprávněné a provoz aplikace na nich závisí, uživatele taková povolení nijak neobtěžují. Ke každé žádosti může mít aplikace vlastní text, sloužící k přesnější specifikaci využití; tento text by neměl na uživatele působit s nátlakem.[8]

4.1.4.8 Terminologie

Uživatel se má při využívání aplikace cítit pohodlně; toho se jistě docílí, pokud je každé slovo z jejího rozhraní součástí pomyslného rozhovoru mezi ní a uživatelem. Takový rozhovor bývá plný známých a srozumitelných slov a frází. Texty a jiné prvky v rozhraní by měly být jasné a stručné. Jazyk aplikace by se měl vyhnout povýšenému znění; popisky v rozhraní jako „my“, „náš“, „já“ či „můj“ mohou v některých případech znět urážlivě. V nejlepším případě je vhodné dobré usilovat o neformální a přátelský tón.[8]

Popis prvků vyvíjené aplikace vychází z terminologie obecně známých výrazů z hudebních aplikací pro uživatele iOS zařízení, nicméně je zde nutnost vytvoření několika méně známých výrazů pro popis prvků sloužících hlavnímu zaměření vyvíjené aplikace; jde především o spravování a nastavování mixů playlistů. Navrhuje se zde použití popisků, které budou především ctít výše uvedené terminologické zásady.[8]

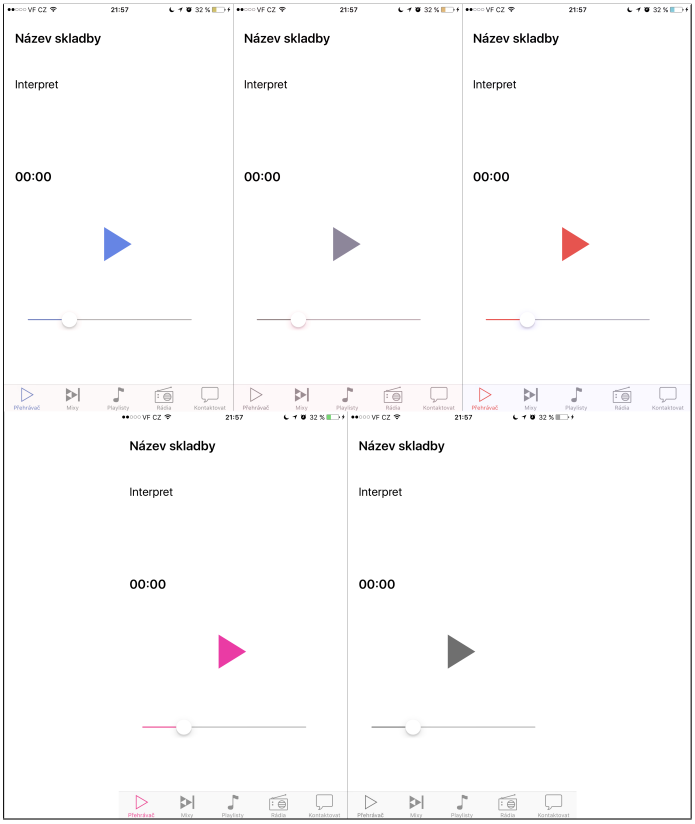
4.1.5 Vizuální dojem

4.1.5.1 Vytváření značky (branding)

Není to jen o know-how a obsahu aplikace, ale i o vyjádření identity aplikace pomocí barev, písma a vhodných obrázků. Každá aplikace slouží především k získávání informací, zábavě nebo podobným zážitkům, ne k inzerci vlastní značky. Branding je tedy nejlepší využít nenápadně v rozhraní aplikace; jednou z dobrých možností je použití barev její ikony. Aplikace musí odolat pokušení zobrazování loga nebo jiných neaktivních prvků v ní tak, aby to neomezovalo obsah, např. titulek v navigační liště je užitečnější než logo.[8]

4.1.5.2 Barvy

Barevnost prostředí v aplikaci by měla korespondovat s barevností loga. Barva indikující interaktivitu má být jasně dána a nemá být použita u neinteraktivních prvků. Všechny použité obrázky je doporučeno převést do režimu RGB, aby se správně zobrazovaly ve všech zařízeních iOS. Důležité je také testování barevného schématu aplikace v různých světelných podmínkách; uvnitř, venku, na slunci a ve tmě anebo v módu nočního režimu (tlumení modrého



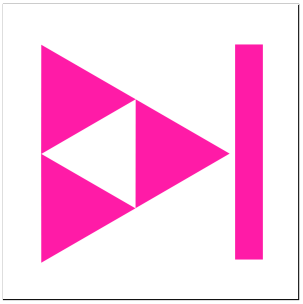
Obrázek 4.11: Navrhovaný přehrávač aplikace Playmix z pohledů různých barvoslepostí[2]

světla, aby zařízení před spánkem rušilo co nejméně) či bez něj. Je nutné také dávat pozor na barvoslepost; někdy je lepší měnit tvary pro rozlišení rozdílných funkcí než barvy, které barvoslepému mohou splynout.[8]

Lze vyzkoušet přes generátor obrázků pro barvoslepé, např. Coblis. Na obrázku 4.11 lze vidět barevnost jakou v navrhované aplikaci barvoslepí mohou vidět.

4.1.5.3 Univerzální rozvržení

Každý uživatel očekává využití aplikace v různých iOS zařízeních, především ve všech svých orientacích displeje. Při různých kontextových změnách je důležité udržovat stejné zaměření na obsah. Celkově je nutné udržovat konzistentní vzhled, tedy prvky s obdobnými funkcemi by měly mít podobný tvar. Větší prvky lépe upoutají pozornost pro interaktivitu – snadněji se na ně poklepe, což je zvláště důležité v rozptylujícím prostředí jako třeba v kuchyni nebo tělocvičně; při jejich tvorbě se musí dbát na minimální dotykovou plochu 44pt x 44 pt. Zarovnání prvků je další z důležitých vlastností, která působí čistě



Obrázek 4.12: Navržená ikona vyvíjené aplikace Playmix

a organizovaně a usnadňuje orientaci v aplikaci. Pokud aplikace podporuje jednu orientaci displeje, je dobré podpořit obě varianty.[8]

4.1.6 Grafické prvky

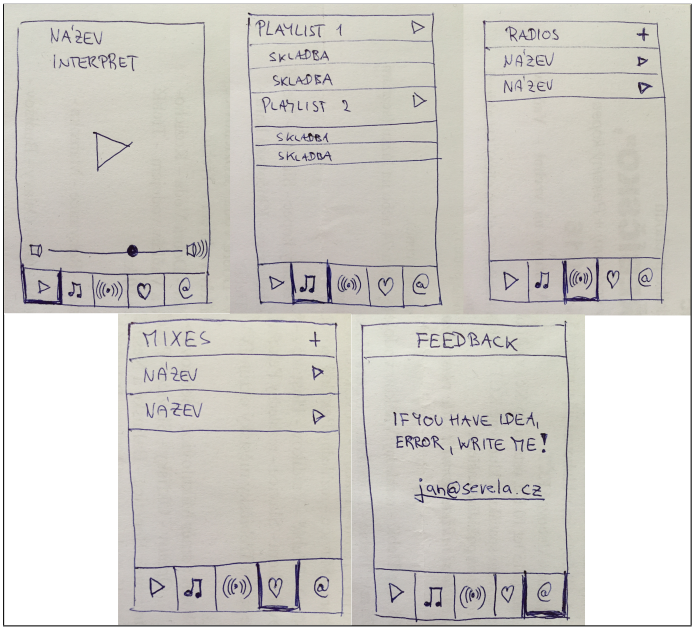
4.1.6.1 Ikona aplikace

Každá aplikace by měla mít krásnou a nezapomenutelnou ikonu tak, aby přitahovala pozornost v App Store a na domovské obrazovce a na první pohled výtvarně vysvětlovala svůj účel, čímž je dána první příležitost ke komunikaci s uživatelem. Svůj účel má představit prostým vzhledem, např. nějakým prvkem či prvky s jednoduchým a jedinečným tvarem. Ikona se objevuje na různých místech v sytému a v různých velikostech; z tohoto důvodu by v ní neměly být příliš složité tvary, protože by mohla být těžce rozpoznatelná. Její pozadí by mělo být jednoduché a bez transparentnosti; určitě není dobré zahrnovat snímky nebo prvky z uživatelského rozhraní. Ikona by se také měla vyhnout textu; na mnoha místech, kde se zobrazuje, má totiž pod sebou název aplikace, který danou funkci dostatečně zastává. Ikona se k aplikaci přikládá ve tvaru čtverce a v různých velikostech pro použití na různých zařízeních a pro App Store – systém sám aplikuje masku pro zaoblení rohů. Důležitou součástí je i testování ikony na různých tapetách v zařízeních.[8]

Dle výše uvedených doporučení byla navržena následující ikona aplikace:4.12

4.1.6.2 Systémové ikony v aplikaci

Většina aplikací se ve svém uživatelském rozhraní, pro jednoduché zobrazování informací nebo navigačních prvků, neobejde bez ikon. V první řadě se doporučují ty nabízené systémem, pokud ale nestačí, je tady možnost vložení vlastních. Systém doporučuje ikony ve třech skupinách, tedy pro Navigation Bar a Toolbar, Tap Bar ikony a ikony pro rychlé akce. V UIKit podpoře se o to starají třídy UIBarButtonItem, UITabBarItem a UIApplicationShortcutIcon.[8]



Obrázek 4.13: Wireframe z papíru při navrhování vyvíjené aplikace Playmix

4.1.6.3 Vlastní ikony v aplikaci

Všechny ikony by měly být v rámci aplikace, v podstatě i celého systému, od sebe rozeznatelné, aby bylo uživateli na první pohled zřejmé, k čemu slouží. Téměř vždy ale platí pravidlo, že v jednoduchosti je síla. Je kladen důraz na konzistenci v rámci aplikace, tedy na velikost, úroveň detailů, perspektivu a tloušťku tahu. Všechny ikony jsou definovány jednobarevně, s průhledností, antialiasingem, beze stínu. iOS ignoruje jakékoli barevné informace – z ikony udělá pouze masku, které je možno nastavit jednu barvu. V neposlední řadě je třeba myslet na alternativní popisky ikon, které při Voice Over pomáhají přístupnosti k navigaci nevidomého uživatele.[8]

4.1.7 Wireframy

Během vytváření návrhu uživatelského rozhraní aplikace vznikaly první náčrty na papír. Na obrázku 4.13 je možné vidět wireframy těchto posledních verzí. Wireframe je obrázek zobrazující vzor uživatelského rozhraní. Další změny byly již zpracovávány přímo v Interface Builderu vývojového prostředí Xcode (vize sekci 2.4).

4.2 Návrh implementace

4.2.1 Nativní vývoj

Implementace je vytvářena nativním vývojem iOS aplikací z důvodu jeho výuky na naší fakultě, dále z důvodu široké dostupnosti zdrojů k tomuto vývoji a nelze opomenout "C friendly" jazyk Swift; jelikož jazyk C a C++ jsou základními kameny programovacích schopností snad každého studenta naší fakulty.

Návrh implementace vychází z návrhu uživatelského rozhraní a dále je inspirován několika různými návody k tvorbě hudebních přehrávačů. Bylo využito poznatků Jonathana Sagorina, Jareda Davidsona, online serveru Ray Wenderlich. Implementace prvků uživatelského rozhraní aplikace jsou inspirovány především dle znalostí získaných z předmětů této fakulty BI-IOS a MI-IOS.

K implementaci se navrhuje použití tří frameworků pro vývoj iOS aplikací a to Cocoa Touch, AVFoundation a MediaPlayer.

První uvedený je určený pro poskytnutí základních a vizuálních objektů k potřebám při tvorbě a správě aplikace na iOS (vizte sekci 2.5) Poskytuje zobrazovací a pohledové prvky architektury, potřebné ke správě uživatelského rozhraní aplikace, manipulaci, tvoření akcí a jiné infrastruktury nutné pro reagování na vstupy uživatele. Dále poskytuje model aplikace nezbytný k běhu hlavní smyčky a interakce se systémem. Je to základní kámen uživatelského rozhraní všech aplikací a operačního systému iOS.[9]

Framework AVFoundation nabízí širokou podporu práce s audiovizuálními médii. Slouží k jejich přehrávání, nahrávání, editaci nebo kódování, například při práci s fotoaparátem, práci s videosnímky a také k rozšířené práci s audio soubory. V této práci lze využít služeb její třídy AVQueuePlayer, která slouží k přehrávání mnoha formátů hudebních médií, a to s plynulou správou této sekvenční fronty.[10]

Framework Media Player slouží k přehrávání videí, hudby, podcastů a audio knih. Jeho pomocí je možno získat přístup k informacím ze systémové hudební knihovny. Je tedy možné vyhledat a přehrávat mediální soubory synchronizované do systémové hudební knihovny z hudební knihovny iTunes desktopového zařízení. Se systémovou hudební knihovnou pracuje v režimu pouze ke čtení.[11]

Framework Media Player je dostačující k tvorbě základních funkcí této práce a zastoupí i práci AVQueuePlayer. Nicméně je zde zvolena možnost využití AVQueuePlayer a to z důvodu testování funkčnosti aplikace v simulátoru testovacího prostředí XCode (vizte 2.4). Na simulátoru jsou možnosti testování aplikace při práci se systémovou hudební knihovnou omezeny, simulátor ji nemá. AVQueuePlayer má širší možnosti využití fronty přehrávání oproti Media Player. Hlavní důvod použití je ale pravidelné testování funkcí aplikace v simulátoru pomocí přehrávání rádio streamů. Testování aplikace na zařízeních s iOS může mít větší časovou prodlevu a tak omezenou možnost odhalení

základních chyb.

Pomocí výše uvedených frameworků lze v podstatě přímo implementovat základní funkce každého hudebního přehrávače, protože patří k základním kamenům a nejlépe dostupným zdrojům k vývoji hudebních přehrávačů.

Problematikou této práce je však mixování playlistů hudební knihovny, což žádný z výše uvedených frameworků svými funkcemi a prvky napřímo nezajistí. Je tedy nutné doimplementovat struktury pro tvorbu, ukládání mixů a jejich parametrů, dále logiku, která bude automaticky dle nastavených parametrů v každém mixu toto přepínání playlistů obstarávat a řídit přehrávací frontu AVQueuePlayer.

Struktury budou vycházet za základních objektů frameworku Cocoa Touch a logika řízení bude postavena na posloupnosti příkazů pro ovládání přehrávací fronty za pomocí časovače dle nastavených hodnot přehrávaného mixu.

4.2.2 Multiplatformní vývoj

Dalším způsobem vývoje mobilních aplikací jsou vývojové platformy a metody pro více platforem zároveň. Mezi nejznámější patří PhoneGap, Appcelerator Titanium a Xamarin.

Prvně uvedený pracuje na webovém jádru každé platformy, vytváří tedy aplikace za pomocí znalostí vývoje webových aplikací, které jsou postaveny především na HTML5/CSS3 a Javascriptu. V případě aplikace na iOS, jsou to v podstatě aplikace postavené na jednom pohledu třídy UIWebView z frameworku UIKit, pomocí kterého se renderuje obsah postavený jako mobilní webová aplikace. Nicméně i PhoneGap přináší širokou nabídku přístupů k různým schopnostem mnoha zařízení, jako je akcelerometr, fotoaparát, přístup ke kontaktům apod.[12]

Appcelator Titanium je vývojové prostředí, které pracuje a renderuje obsah každé aplikace za pomocí nativních knihoven. Je sice postaveno na jazyku Javascript, ten ale komunikuje přímo s Appcelerator Titanium API, které všechny objekty převádí do objektů nativního vývoje dané platformy. Umožňuje tak vytvářet aplikace pro platformy Android, iOS i Windows za pomocí multiplatformního jazyku Javascript. Jedinou překážkou multiplatformního vývoje je prostředí na kterém je aplikace vyvíjena, nejde totiž vyvinou aplikaci pro Windows na systému macOS, pro iOS a Windows na platformě Linux a pro iOS na platformě Windows.[13]

Xamarin pracuje na podobném principu jako Appcelerator s tím rozdílem, že je zde komunikace s API zajištěna za pomocí programovacího jazyku C#.

Každé z výše uvedených multiplatformních vývojových prostředí nabízí kromě svých funkcionalit anebo API mnoho dalších užitečných nástrojů pro vývoj mobilních aplikací, dále jistě přitahuje velikou pozornost svým širokým spektrem mobilních zařízení, čímž se snaží dohánět aktuálnost, silnou vývojovou základnu a přístup ke zdrojům jako má nativní vývoj.

Kapitola 5

Implementace

Souhrn všech částí aplikace:

- Předpřipravené třídy a struktury
  - AppDelegate.swift
  - Assets.xcassets
  - Main.storyboard
  - LunchScreen.storyboard
  - Info.plist
- Player
  - Items.swift
  - Musicquery.swift
  - PlayerViewController.swift
- Mixes
  - MixesViewController.swift
  - MixItemsViewController.swift
  - MixAddItemsViewController.swift
  - DetailMixSettingsViewController.swift
- Playlists
  - PlaylistsViewController.swift
  - PlaylistViewController.swift
- Radios
  - RadiosViewController.swift
- FeedbackViewController.swift
- Images

Zdrojové kódy souborů výše uvedených skupin jsou součástí přílohy práce. Nyní budou popsány všechny jednotlivé skupiny, jejich třídy a jiné objekty.

5.1 Předpřipravené třídy a struktury aplikace

5.1.1 AppDelegate

Soubor AppDelegate.swift obsahuje předpřipravený zdrojový kód třídy delegáta aplikace s možností vytváření změn, který začíná pracovat po spuštění aplikace. Jeho metody mají na starost globální řízení chování aplikace. Jedná se o metody:

```
func application(application: UIApplication,
      didFinishLaunchingWithOptions launchOptions: [NSObject:
      AnyObject]?) -> Bool
```

Metoda, která dává možnost inicializovat objekty před tím, než se zobrazí uživatelské prostředí aplikace.[1]

```
func applicationWillResignActive(application: UIApplication)
```

Metoda, která se volá při přerušení aplikace, při změně stavu aplikace z aktivního do neaktivního režimu. Toto nastává například u situací jako je telefonní hovor nebo obdržení SMS zprávy, případně když uživatel z aplikace odejde do jiné nebo obecně přejde na výchozí systémovou obrazovku. V této metodě lze případně uložit data, která byla vytvořena před přerušením, aby o ně uživatel nepřišel, případně aplikaci připravit do klidového režimu.

```
func applicationDidEnterBackground(application: UIApplication)
```

Tuto předpřipravenou metodu lze použít k uvolnění sdíleného zdroje, při přechodu do stavu na pozadí, uložení uživatelských dat, zneplatnění časovače a uložení dostatku informací o stavu aplikace k jejímu případnému obnovení do současného stavu v případě, že je později převedena do neaktivního režimu nebo ukončena systémem.

```
func applicationWillEnterForeground(application: UIApplication)
```

Metoda, která je volána jako část přechodu stavu na pozadí do neaktivního režimu. Zde se doporučuje vracet nebo ukládat změny vytvořené při změně aplikace do stavu na pozadí.

```
func applicationDidBecomeActive(application: UIApplication)
```

Metoda, která může restartovat úkoly, jež byly pozastaveny nebo ještě nezačaly, zatímco byla aplikace v neaktivním režimu. V případě, že byla aplikace v režimu na pozadí, může obnovit uživatelské prostředí.

```
func applicationWillTerminate(application: UIApplication)
```

Delegát dále obsahuje také globální proměnou pro nejvyšší zobrazovací objekt, a to okno aplikace, pomocí něhož se celé uživatelské prostředí zobrazuje:

```
var window: UIWindow?
```

Aplikace používá většinou jen jedno okno, pomocí kterého svůj obsah vykresluje[1].

Zdrojovému kódu delegáta lze dodefinovat další podporované metody, které delegát při různých stavech nebo akcích uživatele delegovaných systémem volá. V práci byla dodefinována následující funkce:

```
override func remoteControlReceivedWithEvent(event: UIEvent?)
```

Jedná se o metodu, jíž může aplikace reagovat na akce vyvolané uživatelem mimo ni, určené pro ni. V aplikaci je využívána pro ovládání play/pause z Control Centra nebo uzamčeného telefonu. Control Centrum je vysouvací nabídka operačního systému, která obsahuje rychlé ovládací prvky, například pro ovládání hudebního programu, spouštění led diody jako světla, odkaz na nastavení budíku nebo časovače, spuštění fotoaparátu, vypnutí/zapnutí wi-fi a další podobné rychlé volby.[14]

5.1.2 Assets.xcassets

Konfigurační soubor pro vytvoření sestav obrázků využitých v aplikaci a použitých dle typu zařízení, na kterém jsou zobrazovány. Jsou zde vytvořeny sestavy pro hlavní ikonu aplikace a sestavy pro všechny ikony hlavního menu a tlačítek play/pause uvnitř aplikace.

5.1.3 Main.storyboard

Soubor, jehož konfigurace se ve větší míře provádí výběrem přednastavených hodnot pomocí kurzoru. Slouží také pro lepší vizuální představu všech možných pohledů aplikace a jejich ovládacích prvků. S jeho pomocí je vytvořeno hlavní menu aplikace, všechny pohledy a jejich základní chování.

5.1.4 LunchScreen.storyboard

Konfigurační soubor aplikace, pomocí něhož se může nastavit vzhled a jeho chování před samotným zobrazením uživatelského rozhraní aplikace. Je využit pro zobrazení názvu programu při spuštění aplikace.

5.1.5 Info.plist

Soubor informací o vlastnostech aplikace. Jedná se o informace konfigurace pro spustitelný soubor aplikace. Jsou v něm popsány ty nejzákladnější vlastnosti, které určují chování a možnosti aplikace.

5.2 Player

Jedná se o skupinu souborů, v nichž jsou implementovány globální objekty, globální metody používané v aplikaci a třída, která implementuje přehrávač aplikace.

5.2.1 items.swift

Zde jsou implementovány globální objekty použité ve více třídách aplikace. Jedná se o frontu přehrávače AVQueuePlayer, časovač, fronty mixů typu playlist a mixů typu rádio stream, pomocné struktury pro uložení mixu typu playlist a jeho jedné položky, mixu typu rádio stream a jeho jedné položky. Dále jsou to pomocná proměnná pro uložení aktuálního stavu hlasitosti před automatickým ztlumením, jež slouží pro plynulejší přechod poslechu zkrácených položek mixu a pomocná proměnná pro určení módu, v jakém aplikace přehrává. Je zde také definována univerzální šablona fronty, pro použití ve více typech front a předpřipravená data rádio streamů.

5.2.2 Musicquery.swift

Tento soubor obsahuje implementaci třídy, která realizuje dotazy pro systémovou hudební knihovnu. Jsou zde dvě metody, a to metoda pro dotaz na informace o playlistech a metoda pro dotaz na skladbu podle jejího ID. Tyto metody jsou inspirovány z aplikace Jonathana Sagorina; metoda pro dotaz na informace o playlistech je vytvořena dle podobném principu jeho dotazování na alba. Metoda na dotaz ohledně skladby podle ID je od autora kompletně převzata[15].

5.2.3 PlayerViewController.swift

Zde se nachází implementace třídy samotného přehrávače. Jsou zde metody pro spuštění přehrávání, jeho přerušení, metoda pro kontrolu přehrávané položky, spouštěna globálním časovačem; v případě přehrávání mixu hlídá jeho nastavení a dle něj aplikuje jiné metody třídy nebo položky přepíná. Dále metoda pro aktualizaci informací o přehrávané položce, metoda pro její plynulé ztlumení, metoda pro přehrávání rádio streamu podle jejího URL, metoda pro přehrávání skladby ze systémové hudební knihovny podle jejího id. A také metoda pro inicializaci možnosti přehrávání na pozadí, když je aplikace v režimu na pozadí[15].

5.3 Mixes

Tato skupina souborů implementuje hlavní aplikační problematiku této práce. Níže uvedené controllery řídí vytváření, zobrazení, správu a nastavení každého mixu za cílem mixování playlistů hudební knihovny.

5.3.1 MixesViewController.swift

Implementace třídy pro řízení zobrazení seznamu všech vytvořených mixů. Třída obsahuje metody pro zobrazení těchto informací s pomocí UITableView[16] z frameworku UIKit, dále metodu pro přidání nového mixu, s pomocí vyskakovacího okna UIAlertController[17] také z frameworku UIKit, metodu pro zobrazení nabídky výběru přehrávání nebo editování mixu, také vyskakovacím oknem s pomocí UIAllerControlleru. A také metodu pro přípravu přechodu převzatou z návodu Jareda Davidsona[18] k editaci mixu a metodu pro spuštění přehrávání vybraného mixu.

5.3.2 MixItemsViewController.swift

Třída, jež vytváří rozhraní aplikace, zobrazující všechny položky mixu a odkaz na ně nebo globální nastavení tohoto mixu. Jsou zde metody, které se starají o toto zobrazení s pomocí tabulky UITableView, přidání položek mixu s pomocí UIAllertControlleru a metoda k přípravě přechodu inspirovaná opět dle návodu Jareda Davidsona[18] na zobrazení pohledu k možnosti přidávání dalších položek.

5.3.3 MixAddItemsViewController.swift

Zde se nachází implementace třídy, která zajišťuje zobrazení seznamu položek, které je možné přidat do mixu, podle typu mixu; to jsou buď playlisty ze systémové hudební knihovny nebo seznam všech rádio streamů. O zobrazení tohoto seznamu se stará již známý pohledový prvek UITableView. Dále se tu nacházejí metody pro přidání vybraných položek do playlistu, vyskakovací okno spomocí UIAlertControlleru pro potvrzení a metoda pro zavolání dotazu na playlisty, která k tomu využívá třídu Musicquery (vizte sekce 5.2.2).

5.3.4 DetailMixSettingsViewController.swift

Poslední implementovaná třída této skupiny řídí zobrazení šablony pro globální nastavení mixu nebo lokální nastavení položky mixu. Nacházejí se zde především zobrazovací prvky UIKit frameworku, a to UILabel, UITextField, UISwitch a také UIPickeView, které slouží pro zobrazení a změnu nastavení mixu.

Třída obsahuje metody a pomocné proměnné nebo konstanty, pro něž se data před zobrazením nahrají, během práce s nimi se pouze zobrazují uživateli jeho nastavené změny a až těsně před ukončením práce s touto třídou se data z ní uloží do globální objektů aplikace příslušného mixu. Jsou zde metody pro práci s UIPickerView, jenž dovolí uživateli nastavovat hodnoty podle předem daného rozsahu hodnot, metody akce při kliknutí do textového pole, zabarvení textového pole upravované hodnoty, načtení dat z mixu, uložení dat do lokálních proměnných, přepsání dat mixu, obnovení vzhledu zobrazovacích prvků,

které simulují uzamčení položek pro editaci dle typu nastavení mixu nebo zobrazení hodnoty v textových polích dle aktuálních změn zadaných uživatelem.

5.4 Playlists

Soubory, kterými se řídí zobrazení playlistů systémové hudební knihovny a všech skladeb vybraného playlistu. Dále nabízí možnost spuštění vybrané skladby.

5.4.1 PlaylistsViewController.swift

Soubor, který implementuje třídu, jež zajišťuje zobrazení všech playlistů systémové hudební knihovny. Jsou zde definovány metody pro správné zobrazení výpisu playlistů pomocí tabulky UITableView. Dále metoda pro přípravu k přechodu na zobrazení výpisu skladeb vybraného playlistu inspirována návodem Jareda Davidsona a metoda, která provádí dotaz na informace o všech playlistech pomocí třídy Musicquery (vizte sekce 5.2.2).

5.4.2 PlaylistViewController.swift

Tato třída implementuje řízení zobrazení skladeb vybraného playlistu s možností vybranou skladbu přehrát. Zobrazení výpisu skladeb je realizováno pomocí tabulky UITableView. Dále implementuje metodu pro přípravu a spuštění přehrávání vybrané skladby.

5.5 Radios

Tato skupina obsahuje jediný soubor a to implementaci třídy RadiosViewController, pomocí níž aplikace zobrazuje všechny uložené rádio streamy a umožňuje přidávat další. Třída implementuje způsob zobrazení všech uložených rádio streamů pomocí tabulky UITableView, dále možností přidání dalšího rádio streamu pomocí vyskakovacího okna UIAlertControlleru, metody pro smazání nebo editaci rádio streamu pomocí akce nad řádkem tabulky a metody pro přípravu a spuštění přehrávání vybraného rádio streamu.

5.6 FeedbackViewController.swift

Soubor implementující třídu, řídící zobrazení pohledu s informací o možnosti zpětné vazby. Možnost zpětné vazby vytvořením e-mailu bylo provedeno shodným způsobem jako návod od Andrew Bancrofta[19].

5.7 Images

Skupina složek a souborů, které obsahují zdrojové obrázky použité v uživatelském rozhraní aplikace, jsou to ikony aplikace a ikony všech možných grafických prvků uživatelského rozhraní aplikace.

Kapitola 6

Testování

Uživatelské rozhraní a funkce aplikace byly testovány od počátku vývoje aplikace, a to jak na simulátoru, tak na mobilních zařízeních s různým rozlišením a v různých orientacích displeje přístroje. Testování u třetích stran lze provádět zkompilovaným souborem podepsaným certifikátem od společnosti Apple, ve kterém je uvedeno číslo přístroje. Nelze vytvořit zkompilovaný spustitelný soubor, který by šel kdekoli odzkoušet. K závěrečnému testování byla vybrána Heuristická analýza Jakoba Nielsena.[20]

6.1 Test použitelnosti - heuristická analýza Jakoba Nielsena

6.1.1 Viditelnost stavu systému

Aplikace dává najevo uživateli v jakém stavu je aplikace dle spodního Tab Bar menu a pokud je někde hlouběji v rámci jedné z hlavních položek Tab Bar menu, tak jej o stavu informuje pomocí Navigation Baru. Při přehrávání informuje o názvu skladby, názvu interpreta, uplynutého a zbývajícího času a také pomocí ikon play/pause o stavu přehrávání. V případě přehrávání rádio streamu uživatel pozná načítání streamu pomocí horní systémové stavové lišty, kde je ukazatel, že přenos dat probíhá.

6.1.2 Propojení systému a reálného světa

Všechny prvky rozhraní jsou popsány všeobecně známými pojmy. Reakce na interakci uživatele přichází v logické posloupnosti. Playlisty mohou působit neohebně, když jsou využívány ze systémové hudební knihovny, která je dává k dispozici v režimu jen pro čtení, ale vychází z obecných zvyků uživatele iOS, který si je obyvkle spravuje na desktopovém zařízení.

6.1.3 Uživatelská kontrola a svoboda

Uživatel má vždy kontrolu nad tím v jakém stavu je a jaké změny provádí. Ze všech různých stavů může odejít a svoji akci odvolat. Jediná "nesvoboda" může nastat při požadování určité skladby z playlistu při mixování playlistů, protože poslech mixů probíhá s náhodným vygenerováním skladby z každého playlistu v každém mixu.

6.1.4 Standardizace a konzistence

Aplikace využívá standardní a známe řídící a zobrazovací prvky z prostředí iOS. Uživatel tak dokáže předvídat jaká reakce na jeho akci nastane.

6.1.5 Prevence chyb

Aplikace má ošetřeny všechny možné vstupy uživatele a dává uživateli přístup jen k tomu k čemu je předurčena. Jedinou slabinou se jeví býti správa radiových streamů a jejich pouštění, nicméně se jedná o bonusovou funkci a může být řešena nějakou aktualizací.

6.1.6 Rozpoznání namísto vzpomínání

V aplikaci se dá snadno orientovat a svými vizuálními prvky dává jasně najevo jejich akce, uživatel si tak nemusí nic nutně pamatovat a učit. Nejsložitější operací se jeví vytváření a správa mixů, nicméně pokud ví co chce a aplikace tuto volbu podporuje, v relativně krátká době to dokáže vyřešit.

6.1.7 Flexibilní a efektivní použití

O přepínání mezi základními funkcemi se stará spodní menu, uživatel tak může relativně rychle měnit cíle svých potřeb.

6.1.8 Estetický a minimalistický

Aplikace využívá jen nutné ovládací prvky, které doplňují jednoduché popisky, neobsahuje žádné prvky, které by rušily hlavní obsah na obrazovce. Všechno doplňuje jedna barva, která zvýrazňuje jen aktivní prvky, jako celek to vypadá velice esteticky a minimalisticky.

6.1.9 Pomoc uživatelů pochopit, poznat a vzpamatovat se z chyb

Každé akci, která provede nějaké změny předchází potvrzení, dává tak na vědomí uživateli jestli to chce opravdu provést a nebo zrušit.

6.1.10 Nápověda a návody

Aplikace neobsahuje funkce, které by bylo nutné dokumentovat, v rámci krátké doby je lze všechny otestovat a to zkoušením vkládání různých vstupů nad uživatelským rozhraním.

Závěr

V práci jsem se zabýval vytvořením hudební aplikace pro podporu výuky tance na zařízení pod operačním systémem iOS. Základní požadavky zahrnovaly práci s playlisty, ukládání vybraných playlistů do mixů a jejich následný poslech, čímž bylo docíleno střídavého poslechu skladeb různých playlistů s pomocí přednastavených parametrů.

K návrhu řešení uživatelského rozhraní jsem dospěl pomocí analýzy potenciálních uživatelů, jejich požadavků a průzkumu jiných hudebních přehrávačů. Při nastavování chování stavebních prvků uživatelského rozhraní jsem z velké míry využil znalostí z předmětů Fakulty informačních technologií Českého vysokého učení technického v Praze BI-IOS a MI-IOS.

Při návrhu implementace jsem vycházel z rad a poznatků vývojářů, jimiž byli Jonathan Sagorin, Jared Davidson a redaktoři online serveru Ray Wenderlich.

Řešení bylo implementováno s pomocí základních stavebních prvků pro tvorbu uživatelského rozhraní a práci s hudební knihovnou. Jedná se o frameworky Cocoa Touch, AVFoundation a Media Player, které jsou součástí dokumentace pro vývojáře aplikací s operačním systémem iOS.

Aplikaci plánuji dále rozvíjet a rozšiřovat její funkcionality pro širší pole působnosti, např. pro použití při tréninku jiného sportu, než jen tanec.

Literatura

[1] Apple Inc.: The App Life Cycle [online]. 2016, [cit. 2016-05-05]. Dostupné z: https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/TheAppLifeCycle/TheAppLifeCycle.html

[2] Colblindor : Coblis — Color Blindness Simulator [online]. 2017, [cit. 2017-01-05]. Dostupné z: https://developer.apple.com/ios/human-interface-guidelines/overview/design-principles/

[3] Ball, M.: History for iOS (version 4.1) [mobilní aplikace]. 2016, [cit. 2016-05-05]. Dostupné z: https://itunes.apple.com/cz/app/history-for-ios/id879016024

[4] Apple Inc.: iOS 10 [online]. 2016, [cit. 2016-12-03]. Dostupné z: http://www.apple.com/cz/ios/ios-10/

[5] Apple Inc.: Novinky [online]. 2016, [cit. 2016-12-10]. Dostupné z: http://www.apple.com/newsroom/2016/09/whats-new-in-ios-10.html

[6] Apple Inc.: The Swift Programming Language (Swift 2.2) [online]. 2016, [cit. 2016-05-08]. Dostupné z: https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/

[7] Apple Inc.: Cocoa (Touch) [online]. 2016, [cit. 2016-05-08]. Dostupné z: https://developer.apple.com/library/ios/documentation/General/Conceptual/DevPedia-CocoaCore/Cocoa.html

[8] Apple Inc.: iOS Human Interface Guidelines — Apple Developer [online]. 2016, [cit. 2016-11-12]. Dostupné z: https://developer.apple.com/ios/human-interface-guidelines/overview/design-principles/

[9] Apple Inc.: UIKit Framework Reference [online]. 2016, [cit. 2016-05-01]. Dostupné z: https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIKit_Framework/

[10] Apple Inc.: AV Foundation [online]. 2016, [cit. 2016-05-01]. Dostupné z: https://developer.apple.com/av-foundation/

[11] Apple Inc.: Media Player Framework Reference [online]. 2016, [cit. 2016-05-01]. Dostupné z: https://developer.apple.com/library/ios/documentation/MediaPlayer/Reference/MediaPlayer_Framework/

[12] Traeg, P.: Four Ways To Build A Mobile Application, Part 3: PhoneGap – Smashing Magazine [online]. 2014, [cit. 2016-27-12]. Dostupné z: https://www.smashingmagazine.com/2014/02/four-ways-to-build-a-mobile-app-part3-phonegap/

[13] Traeg, P.: Four Ways To Build A Mobile Application, Part 4: Appcelerator Titanium – Smashing Magazine [online]. 2014, [cit. 2016-27-12]. Dostupné z: https://www.smashingmagazine.com/2014/03/4-ways-build-mobile-application-part4-appcelerator-titanium/

[14] Apple Inc.: Use Control Center on your iPhone, iPad, and iPod touch [online]. 2016, [cit. 2016-05-01]. Dostupné z: https://support.apple.com/en-us/HT202769

[15] Sagorin, J.: iOSBackgroundAudioSwift [online]. 2016, [cit. 2016-05-01]. Dostupné z: https://github.com/jsagorin/iOSBackgroundAudioSwift

[16] Apple Inc.: UITableView Class Reference [online]. 2016, [cit. 2016-05-07]. Dostupné z: https://developer.apple.com/library/ios/documentation/UIKit/Reference/UITableView_Class/

[17] Apple Inc.: UIAlertController Class Reference [online]. 2016, [cit. 2016-05-07]. Dostupné z: https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIAlertController_class/

[18] Davidson, J.: Passing Data Between Table Views using Structs and Arrays (Swift : Xcode) [online]. 2015, [cit. 2016-05-07]. Dostupné z: https://www.youtube.com/watch?v=pR6dR-vVZeY

[19] Bancroft, A.: Send Email In-App — Using MFMailComposeViewController with Swift [online]. 2014, [cit. 2016-05-16]. Dostupné z: https://www.andrewcbancroft.com/2014/08/25/send-email-in-app-using-mfmailcomposeviewcontroller-with-swift/

[20] Nielsen, J.: 10 Usability Heuristics for User Interface Design [online]. 1995, [cit. 2017-01-03]. Dostupné z: https://www.nngroup.com/articles/ten-usability-heuristics/

Příloha A

Seznam použitých zkratek

API Application Programming Interface (rozhraní pro programování aplikací)

ARC Automatic Reference Counting (automatické počítání referencí)

BI-IOS Základy vývoje iOS aplikací pro iPhone a iPad

CSS3 Cascading Style Sheets 3 (kaskádové styly verze 3)

FTP File transfer protocol (protokol pro přenos souborů mezi počítači)

UI User interface (Uživatelské rozhraní)

HTML5 HyperText Markup language 5 (značkovací jazyk verze 5)

MI-IOS Pokročilé techniky v iOS aplikacích

MVC Model View Controller

sRGB standardní RGB barevný prostor

URL Uniform Resource Locator (jednotná adresa zdroje)

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `PlayMix` ..... adresář s celým projektem aplikace pro Xcode
  - `images` ..... obrázkové soubory k projektu
  - `PlayMix` ..... zdrojové kódy implementace
  - `PlayMix.xcodeproj` .. zdrojové soubory s metainformacemi k projektu aplikace Xcode
- `TexSrc` ..... zdrojové soubory textu práce
  - `BP-Sevela-Jan-201617ZS.pdf` ..... text práce ve formátu PDF
  - `BP-Sevela-Jan-201617ZS.tex` zdrojová forma práce ve formátu LaTeX
- `zadani-BP-Sevela-Jan-201617ZS.pdf` ..... zadání bakalářské práce