VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

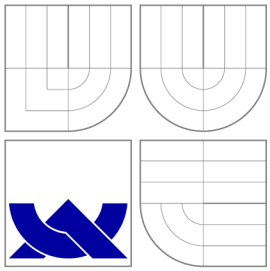
# PŘEHRÁVAČ SKLADEB PRO IPHONE PRO VÝUKOVÉ ÚČELY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

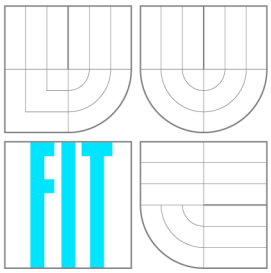
AUTOR PRÁCE
AUTHOR

JIŘÍ KLOBÁSA

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

# PŘEHRÁVAČ SKLADEB PRO IPHONE PRO VÝUKOVÉ ÚČELY

PLAYER FOR IPHONE FOR EDUCATIONAL PURPOSES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JIŘÍ KLOBÁSA
AUTHOR

VEDOUCÍ PRÁCE — Ing. VÍTĚZSLAV BERAN, Ph.D.
SUPERVISOR

BRNO 2013

# Abstrakt

Práce se zabývá návrhem a implementací mobilní aplikace pro zařízení iPhone, která slouží jako přehrávač hudebních skladeb. Aplikace pracuje s metainformacemi ke skladbě a navrhuje jejich využití při procesu memorizace. Práce je rozdělena do kapitol, kde jsou postupně popisovány vlastnosti cílové platformy iOS, návrh uživatelského rozhraní a jeho testování, zásady tvorby počítačové hudby a zvuku. Jádrem práce je návrh uživatelského rozhraní aplikace společně se strukturou a využitím metainformací ke skladbě. Závěrem práce je popsána realizace výsledné aplikace, provedené testy a další možnosti jejího rozšíření.



# Abstract

This thesis deals with the design and implementation of mobile application for iPhone that serves as a music player. The application works with meta-information to a music song and suggests their use in the process of memorization. The thesis is divided into chapters, starting with the characteristics of the target platform iOS, user interface design and its testing, principles of creating computer music and sound. The core of this thesis is to design a user interface along with the structure and use of meta-information related to a song. At the end of the thesis there is described the implementation of the application, the tests performed and possibilities for further improvements of the application.



# Klíčová slova

uživatelské rozhraní, iPhone, iOS, přehrávač, hudba, skladby, mobilní aplikace, memorizace, zvuk

# Keywords

user interface, iPhone, iOS, player, music, songs, mobile application, memorization, sound

# Citace

Jiří Klobása: Přehrávač skladeb pro iPhone pro výukové účely, bakalářská práce, Brno, FIT VUT v Brně, 2013

# Přehrávač skladeb pro iPhone pro výukové účely

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vítězslava Berana, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jiří Klobása
15. května 2013


## Poděkování

Rád bych poděkoval vedoucímu práce panu Ing. Vítězslavu Beranovi, Ph.D. za odborné vedení, ochotu a čas, který mi věnoval při konzultacích. Dále bych také rád poděkoval firmě X Production s.r.o. za pomoc při testování aplikace, slečně Lucii Vomelové a mé rodině za podporu při tvorbě této práce.

# Obsah

# Kapitola 1

# Úvod

Učení je dnes nedílnou součástí života každého člověka. Ať už se jedná o praktické učení, učení nazpaměť nebo celoživotní učení, každý hledá způsob, který mu nejvíce vyhovuje a snaží se usnadnit si tento mnohdy nelehký a zdlouhavý proces. Při učení člověk nejvíce spoléhá na své smysly, díky nimž získává informace z okolí. Největší mírou se na tomto procesu podílí zrakové a sluchové vjemy.

Stále se rozvíjející oblast multimediálních zařízení, jako například chytrých telefonů, tabletů a senzorických zařízení, je čím dál tím více využívána právě pro výukové účely. Je tomu především kvůli možnosti prezentace informací hned několika způsoby najednou. Nejčastější formou bývá kombinace vizuální a zvukové prezentace, stále více je však preferována interakce pomocí gest na dotykových obrazovkách.

Cílem této práce je navrhnout a realizovat uživatelské rozhraní mobilní aplikace, která bude využívat metadata[1] k hudební skladbě a umožní přehrávat skladbu po taktech nebo celých částech. Hlavním úkolem aplikace je usnadnit proces memorizace skladby pro hudebníky a ty, kteří dávají přednost učení zvukovou formou. Pro implementaci bylo zvoleno mobilní zařízení iPhone pracující na platformě iOS[2] a to především proto, že poskytuje celou řadu nástrojů využitelných pro efektivní výuku formou kombinované prezentace.

Následující kapitola nabízí bližší seznámení s teoretickými znalostmi nutnými pro tvorbu návrhu a realizaci aplikace. Čtenář se dozví základní rysy, principy vývoje a charakteristiku uživatelského rozhraní cílové platformy iOS. Dále budou popsány nástroje pro práci s audiem na platformě iOS, návrh a testování uživatelského rozhraní z hlediska interakce s uživatelem. Závěr kapitoly 2 je věnován zásadám tvorby počítačové hudby a zvuku. Kapitola 3 obsahuje popis návrhu aplikace, její stavbu a rozdělení do funkčních bloků. Je zde také diskutována specifikace a využití metadat pro efektivní memorizaci skladby společně s možnostmi testování aplikace. Na realizační část aplikace je zaměřena kapitola 4. V úvodu této kapitoly jsou charakterizovány implementační aspekty aplikace s ukázkou několika hlavních obrazovek. Podstatná část kapitoly je také věnována datovému modelu aplikace. Dále jsou specifikovány provedené testy, zkoumané vlastnosti aplikace a výsledky získané vyhodnocením testů. Poslední kapitola shrnuje postup při řešení této práce, cíl práce a jeho naplnění na základě dosažených výsledků. Závěr kapitoly 5 popisuje další možnosti řešení a současně případná vylepšení řešení stávajícího.

[1] Strukturovaná data o datech

[2] Mobilní operační systém vytvořený společností Apple Inc.

# Kapitola 2

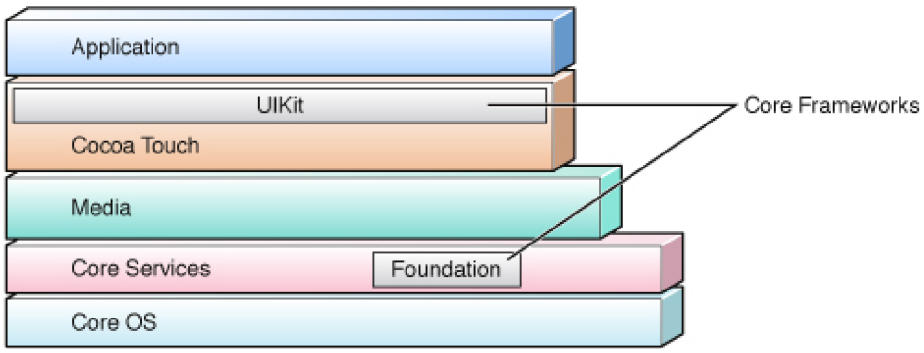
# Teorie

V této kapitole budou vysvětleny základní principy a uvedení do problematiky vývoje aplikací na platformě iOS. Dále budou prezentovány charakteristické rysy uživatelského rozhraní platformy iOS, nástroje využívané pro vývoj a nástroje pro práci s audiem na iOS. Podstatná část kapitoly je také věnována analýze uživatelského rozhraní z hlediska interakce s uživatelem a jeho možná testování. V závěru kapitoly budou popsány zásady tvorby počítačové hudby a zvuku.

## 2.1 Platforma iOS

iOS je operační systém vytvořený společností Apple Inc. pro mobilní zařízení. Jedná se o zjednodušenou verzi operačního systému Mac OS X, která obsahuje podporu dotykového ovládání. Název iOS je využíván až od čtvrté verze tohoto systému, kdy nahradil dříve používaný název iPhone OS. Toto nové pojmenování je především v souladu s politikou pojmenovávání produktů společností Apple Inc.

Tato platforma obsahuje čtyří základní vrstvy [2], které zajišťují funkčnost systému a na něm běžících aplikací. Tyto vrstvy poskytují vývojářům prostředky a rozhraní pro ovládání všech funkcí zařízení. Jsou jimi vrstva Cocoa Touch, Media, Core Services a Core OS. Nejvyšší a zároveň pátou pomyslnou vrstvou je vrstva samotné aplikace, která je již tvořena vývojáři a využívá dříve zmíněné nižší vrstvy. Jejich hierarchické uspořádání lze najít na obrázku 2.1.




Obrázek 2.1: Vrstvy platformy iOS [2]

Vrstva Cocoa Touch je nejvyužívanější vrstvou především kvůli vysoké míře abstrakce a jednoduchému použití, které poskytuje. Tato vrstva obsahuje nástroje pro implementaci grafického rozhraní aplikace, interakce s uživatelem a také celou řadu vysokoúrovňových systémových služeb jako například multitasking, ochranu dat, systém upozornění, rozpoznávání gest a další.

Vrstva Media poskytuje prostředky pro vytváření grafických prvků, přehrávání animací, videí nebo zvuků. Technologie pro práci se zvukem budou podrobněji popsány v kapitole 2.2.

Vrstva Core Services umožňuje přístup k lokalizačním službám, vláknovému programování, nákupům v aplikaci, databázi SQLite pro uložení dat a nástrojům ke zjištění aktuální konfigurace zařízení.

Poslední a také nejnižší vrstvou je Core OS. Tato vrstva především poskytuje nízkoúrovňové funkce ostatním technologiím. I když nebývá v aplikacích využívána přímo, velmi často je využita prostřednictvím některé z vyšších vrstev. Nachází se zde nízkoúrovňové rozhraní pro práci s matematickými funkcemi, výpočty DSP[1], připojení a komunikaci s externími zařízeními, zajištění bezpečnosti citlivých dat.

Z pohledu vývojáře jsou však veškeré nástroje a rozhraní pro přístup k funkcím zařízení na platformě iOS prezentovány ve formě frameworků neboli balíků nástrojů, podpůrných programů a aplikačních rozhraní [5]. Základními frameworky, které jsou téměř vždy využívány, jsou UIKit[2] pro tvorbu uživatelského rozhraní aplikace, zpracování událostí, systému oken a ovládání se zaměřením na dotykové rozhraní a Foundation, který obsahuje základní třídy jazyka Objective-C pro strukturované uchování dat a podporu jednotného vývoje aplikací.

### 2.1.1 Vývoj na platformě iOS

Vývoj na platformě iOS je díky společnosti Apple Inc. téměř uniformní. V roce 2008 byl vydán iOS SDK[3], který poskytuje sadu nástrojů pro vývoj nativních aplikací. Tato sada nástrojů [8] je zdarma k dispozici ke stažení, podmínkou je však nutnost vytvoření tzv. Apple ID, které slouží jako identifikace pro přístup ke službám společnosti Apple Inc. Hlavním vývojovým nástrojem, který je součástí iOS SDK, je Xcode [5, p. 56], integrované vývojové prostředí dostupné pouze pro operační systém Mac OS X. Xcode obsahuje kromě všech balíků potřebných pro vývoj aplikací také precizně zpracovanou dokumentaci.

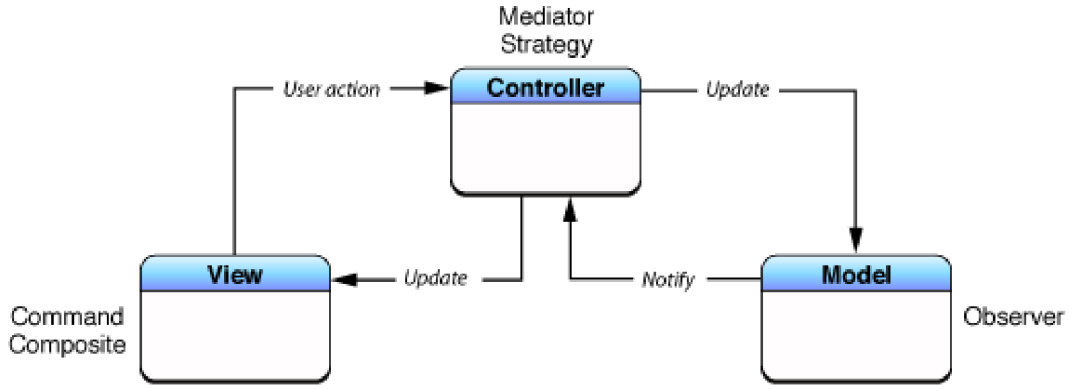
Důležitou součástí Xcode je také simulátor zařízení umožňující simulovat chování aplikace jako na reálném zařízení. Aplikaci lze tedy vyvíjet a testovat bez nutnosti použití reálného zařízení, avšak jsou zde jistá omezení jako například absence kamery nebo knihovny hudebních skladeb. Pro možnost testovat aplikaci na reálném zařízení nebo ji dále distribuovat například do App Store[4] je nutné zakoupit členství ve vývojářském programu pro zařízení se systémem iOS. Poplatek činí aktuálně 99$ ročně. iOS SDK obsahuje také nástroje pro efektivní ladění aplikací. Nejpoužívanější nástroj Instruments [5, p. 58] poskytuje možnost provedení výkonových testů, analýzy zdrojového kódu aplikace nebo míry využitých zdrojů aplikací.

Tvorba aplikace je prováděna především v nativním jazyce Objective-C [7], který je využit ve všech sadách nástrojů a knihovnách poskytnutých společností Apple Inc. Tento

[1] Digitální signálový procesor
[2] User Interface Kit
[3] Software Development Kit
[4] Obchod s aplikacemi pro zařízení se systémem iOS spravovaný společností Apple Inc.

jazyk vznikl jako nadstavba jazyka C spojením syntaxe jazyka Smalltalk a C. Lze jej zařadit mezi vysokoúrovňové objektově orientované jazyky. Mimo Objective-C je také možné využít některého z jazyků rodiny C jako například C, C++ nebo Objective-C++. Vývoj v jiných programovacích jazycích je umožněn pouze externí formou, tedy při sestavení aplikace mohou být připojeny již zkompilované zdrojové kódy.

Ve spojení s objektově orientovaným jazykem Objective-C je také využívána koncepce MVC neboli Model View Controller [4]. Jedná se o softwarovou architekturu, která rozděluje datový model aplikace, uživatelské rozhraní a řídicí logiku do tří nezávislých komponent tak, že modifikace některé z nich má jen minimální vliv na ostatní. MVC se také výraznou mírou podílí na čistotě návrhu aplikace a znovupoužitelnosti zdrojových kódů.




Obrázek 2.2: MVC na platformě iOS [4]

Tento způsob vývoje má své výhody i nevýhody oproti ostatním operačním systémům pro mobilní zařízení. Mezi výhody lze zařadit jednotnost vývoje, která je zajištěna stylem uživatelského rozhraní a hardwarovými prostředky zařízení, protože se jedná o platformu navrženou čistě pro zařízení od společnosti Apple Inc. Mezi další výhody lze zařadit distribuci prostřednictvím aplikace App Store. Nevýhodou vývoje pro platformu iOS je nutnost zřízení placeného vývojářského účtu pro možnost testovat aplikace na reálném zařízení a distribuovat je ke koncovým uživatelům. Jistým omezením také může být striktní politika společnosti Apple Inc. z hlediska vzhledu uživatelského rozhraní. Více o uživatelském rozhraní platformy iOS bude popsáno v následující kapitole.

### 2.1.2 Uživatelské rozhraní platformy iOS

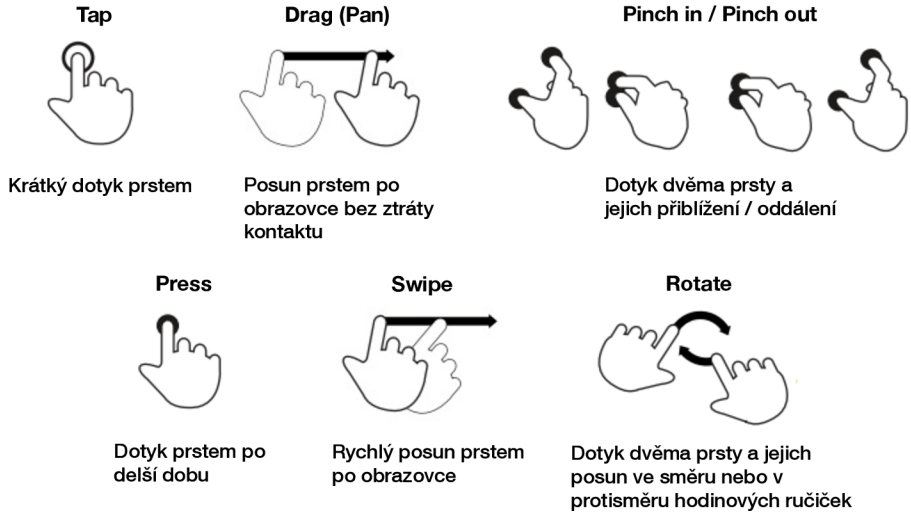
Při vývoji pro mobilní platformy je důležitý správný návrh uživatelského rozhraní. Mobilní zařízení je typicky omezeno velikostí obrazovky a způsobem interakce s obsahem na obrazovce. Současné populární platformy pro mobilní zařízení se poměrně liší důrazem na integritu zobrazení a chování ovládacích prvků. Platforma iOS zde těží z faktu, že byla od počátku navržena pro zobrazení a ovládání obsahu pouze přes obrazovku zařízení, tedy pomocí samotného softwaru. Je zde jediné hardwarové tlačítko pro odchod z jakéhokoliv stavu aplikace na domovskou obrazovku.

Uživatelské rozhraní platformy iOS vychází především z politiky společnosti Apple Inc. Jedním z hlavních důvodů jejího úspěchu je snaha o dodržení jednotného způsobu ovládání a také pravidla, že aplikace fungují tak, jak vypadají[5]. Jinými slovy největší důraz je kladen na jednoduchost a efektivitu ovládání aplikace. Pravidla tvorby aplikací od společnosti

[5]V angličtině byl ustálen výraz *Form follows function*

Apple Inc. jsou popsána v dokumentu HIG[6], který musí každý vývojář respektovat a vyvíjet aplikace v souladu s tímto soupisem pravidel. V opačném případě aplikace ve schvalovacím procesu pro distribuci v App Store neuspěje. Tato pravidla [9] se týkají nejen rozmístění prvků na obrazovce zařízení, ale také jejich funkcionality, sémantiky, hierarchického uspořádání, estetické celistvosti a zpětné vazby jednotlivých prvků.

Dalším specifikem této platformy je také systém gest a dotyků. Interakce s každým prvkem uživatelského rozhraní může být víceznačná, každému prvku lze přiřadit odpovídající akci na jiné provedené gesto. Platforma iOS definuje celkem šest základních typů gest, která lze provádět na vícedotykové obrazovce zařízení. Tato gesta jsou nazývána *tap*, *press*, *pinch*, *pan* nebo *drag*, *swipe* a *rotate*. Jejich názorné provedení a stručný popis zachycuje obrázek 2.3. Další gesta lze programově definovat a přiřadit zvolenému prvku uživatelského rozhraní.

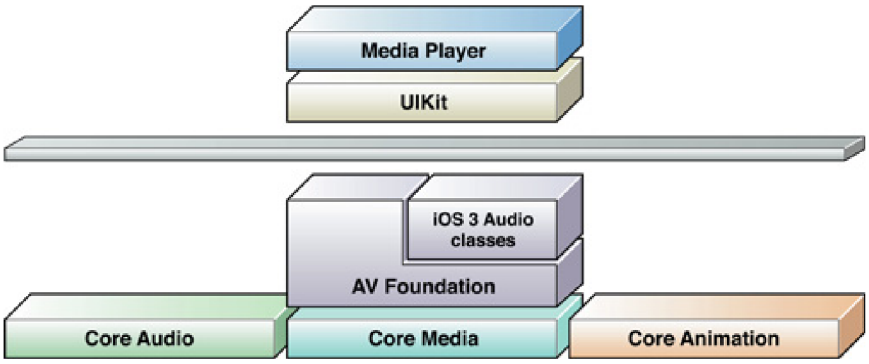



Obrázek 2.3: Gesta platformy iOS[7]

### 2.1.3 Nástroje pro práci se zvukem

Platforma iOS sdružuje nástroje pro práci se zvukem nebo videem, pro tvorbu animací a grafických prvků ve vrstvě Media. Jedinou výjimkou je sada nástrojů Media Player, která je zařazena do vyšší vrstvy Cocoa Touch díky vysoké míře abstrakce, kterou poskytuje. Tyto nástroje jsou optimalizovány pro potřeby zařízení a vyznačují se jednodušším přístupem, protože jsou napsány v nativním jazyce Objective-C. Každý z nástrojů pracuje na jiné technologické úrovni, platí zde tedy stejný princip, který je uplatněn u hierarchie vrstev platformy iOS. Uspořádání nástrojů pro práci s médii je zobrazeno na obrázku 2.4. V následujícím textu je věnována pozornost hlavním technologiím pro práci se zvukem na platformě iOS.

[6] Human Interface Guidelines - soubor doporučení a pravidel tvorby a návrhu uživatelského rozhraní
[7] Převzato z http://adpearance.com/images/uploads/gestures.png




Obrázek 2.4: Nástroje pro práci s médii [3]

### AV Foundation

AV Foundation [3] je jedním z vysokoúrovňových frameworků pro práci s médii. Poskytuje širokou škálu funkcí, z nichž některé jsou zprostředkovány nižší vrstvou Core Media. Svým hierarchickým postavením patří do vrstvy Media. Je zaměřen především na přehrávání audia a práci s audiovizuálními médii. Dále jej lze využít také na zpracování médií, jejich vytváření či úpravu.

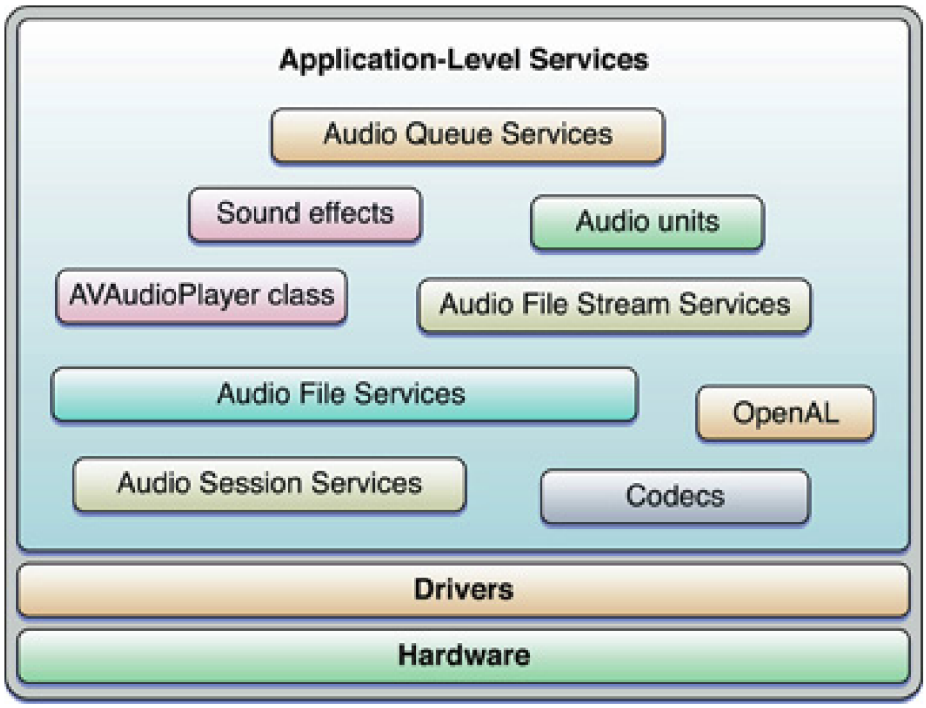
Samotný AV Foundation lze rozdělit na dvě aplikační rozhraní. První z nich je rozhraní využívané pouze pro práci s audiem, které bylo využíváno před verzí systému iOS 4. Toto rozhraní poskytuje především možnost přehrávat a nahrávat audio. Druhé rozhraní bylo uvedeno ve čtvrté verzi systému iOS a přineslo s sebou nástroje pro obecnou práci s médii.

Hlavní stavebním prvkem, který umožňuje reprezentaci multimediálních dat, je třída *AVAsset*. Struktura AV Foundation je právě postavena na této reprezentaci. Tato třída obsahuje jeden nebo více multimediálních zdrojů a jejich metadata jako například název, velikost, délka atd. Výhodou *AVAsset* je uniformní práce se zdroji médií, protože tato třída není fyzicky vázána k určitému formátu dat. Kromě čtení médií prostřednictvím třídy *AVAssetReader* umožňuje také úpravu zdrojů a jejich metadat prostřednictvím třídy *AVAssetWriter*. Funkce přehrávače je zprostředkována třídou *AVPlayer*, která pracuje s třídou *AVPlayerItem* obsahující odkazy právě na médium uložené v uniformním tvaru ve třídě *AVAsset*. Mezi další funkce AV Foundation patří například sekvenční přehrávání nebo vytvoření fronty skladeb.

### Core Audio

Sada nástrojů Core Audio [1] poskytuje aplikační rozhraní pro manipulaci s audiem. Díky své specializaci na audio obsahuje širokou škálu rozhraní, která jsou dle hierarchické úrovně, na které se nacházejí, implementována v nativním jazyce Objective-C nebo v jazyce C. Využití rozhraní nižší úrovně umožňuje větší flexibilitu práce vývojáře, je zde však i faktor nižší abstrakce od vrstvy hardwaru a tedy vyšší složitosti. Aplikační rozhraní sady nástrojů Core Audio na platformě iOS zachycuje obrázek 2.5.

Jedním z frameworků, který je součástí Core Audio je Audio Toolbox [1, p. 74]. Poskytuje nástroje pro úpravy formátu audia, především změny bitové hloubky, vzorkovací frekvence, komprimace a dekomprimace. Audio Toolbox také nabízí možnost čtení a zápisu audia na fyzické médium ve zvoleném formátu, načítání dat prostřednictvím streamů, zjišťování informací o audiu, dynamickou modifikaci audia nebo přehrávání systémových zvuků,




Obrázek 2.5: Aplikační rozhraní Core Audio [1]

mezi které lze zařadit vibrace zařízení nebo jednoduché zvuky maximální délky 30 sekund. Kromě přehrávání umožňuje také nahrávání audia a vytváření seznamů pro přehrávání. Audio Toolbox je však hlavně zaměřen na práci se zvukem na úrovni jeho modifikace a optimalizace.

Další rozšířenou sadou nástrojů, která je zahrnuta v Core Audio, je OpenAL [5, p. 30] neboli Open Audio Library. OpenAL je multiplatformní rozhraní pro práci s audiem, které je prezentováno na platformě iOS především kvůli podpoře 3D audia, které je využíváno při tvorbě herních aplikací. Projekt OpenAL je dnes také udržován za velké podpory společnosti Apple Inc.

### Media Player

Nejvyšší míru abstrakce práce s médii na platformě iOS poskytuje sada nástrojů Media Player [5, p. 29] vrstvy Cocoa Touch. Obsahuje základní funkce pro práci s médii. Z oblasti audia poskytuje přehrávání skladeb, audio podcastu a audio knih. Většina funkcí pro přehrávání audia je spíše jednoduššího typu, avšak jejich použití je snadné. Snadná manipulace s tímto nástrojem je zajištěna díky vysokému postavení v hierarchii vrstev platformy iOS. Složitější funkce jsou zprostředkovány prostřednictvím nižších vrstev. Jedná se o jedinou sadu nástrojů pro práci s médii, která není součástí vrstvy Media.

Jednou z neméně důležitých funkcí je také možnost propojení s knihovnou iTunes, která funguje na platformě iOS jako databáze audia. Přístup je však omezen pouze na čtení, což vyplývá z politiky společnosti Apple Inc. a zajištění bezpečnosti a integrity dat aplikací. Další výhodou tohoto nástroje je také možnost využití již integrovaných prvků pro přehrávání médií. Vývojář tedy nemusí vytvářet uživatelské rozhraní, protože lze využít prvky Media Player frameworku [6].

## 2.2 Uživatelské rozhraní

Uživatelské rozhraní objektu lze charakterizovat jako souhrn postupů, jak s daným objektem pracovat. Uživatelské rozhraní tedy není vázáno výhradně na počítače a jejich software, ale také na ostatní věci, jako například auto, pračka nebo telefon. Mezi zásady tvorby dobré aplikace tedy patří nejen dobré naprogramovaní, ať už z hlediska efektivity fungování nebo vnitřní architektury, ale i použitelnost. Dobré uživatelské rozhraní umožňuje uživateli pracovat s aplikací účelně a pohodlně. Rozhraní, které navíc dodržuje předepsaná pravidla, pomáhá uživateli vytvářet si správné stereotypy práce se systémem.

Téměř ke každé kvalitní vývojařské platformě grafického uživatelského rozhraní jsou k dizpozici oficiální pravidla pro jeho tvorbu, známé pod zkratkou HIG neboli Human Interface Guidelines. HIG specifikuje způsob tvorby uživatelského rozhraní, možnost použití ovládacích prvků a oken i způsob, jak na jejich základě správně vytvářet aplikace. Platí, že kromě specifických pravidel existují pravidla, která platí ve všech systémech univerzálně. Následující kapitola bude věnována aplikaci těchto pravidel při návrhu uživatelského rozhraní.

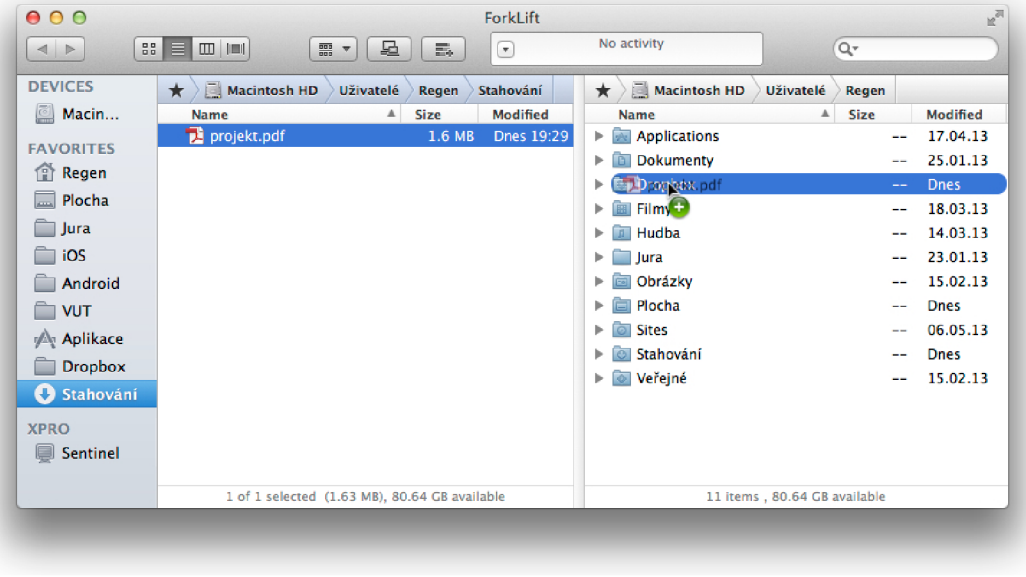
### 2.2.1 Tvorba uživatelského rozhraní

Při tvorbě a návrhu uživatelského rozhraní je vhodné se držet obecných principů, které jsou platné napříč platformami. Důležitost každého pravidla se však může u konkrétní aplikace a platformy lišit. Každé pravidlo také může mít svou výjimku, dobrým postupem je snažit se pravidla dodržet. V následujícím textu budou vysvětleny jednotlivé principy tvorby uživatelského rozhraní [18].

- **Konzistence** - Dodržování pravidel pro tvorbu uživatelského rozhraní na dané platformě a tvorba stereotypů při ovládání aplikace. Důraz na konzistentní terminologii.
- **Zpětná vazba** - Informování uživatele o stavu prováděných akcí s aplikací. Velice důležitá je volba odpovídající zpětné vazby v závislosti na důležitosti informace tak, aby uživatele bezdůvodně neobtěžovala. Zpětnou vazbu lze rozdělit na silnou a slabou. Silná zpětná vazba se vyznačuje nutností reakce uživatele, který musí potvrdit příjem informace. Slabá zpětná vazba nevyžaduje potvrzení, že je uživatel s informací seznámen.
- **Navigace** - Rozdělení složitějších uživatelských úloh na jednodušší, kdy logicky členěné kroky respektující pracovní postup poskytují lepší navigaci.
- **Návrat zpět** - Možnost návratu zpět z provedené akce a tolerance k chybám uživatele. Pokud je to možné, snaha poskytnout uživateli možnost návratu zpět nebo zastavení aktuálně prováděné akce.
- **Předcházení chybám** - Snaha o minimalizaci chyb způsobených uživatelem. V případě vzniku chyby, je důležité uživatele informovat o chybě a jejich možných příčinách.
- **Předvídatelné rozhraní** - Předvídatelné uživatelské rozhraní je ovládáno uživatelem, který je řídicím prvkem a zadává příkazy aplikaci.
- **Skupina uživatelů** - Tvorba uživatelského rozhraní na základě skupin uživatelů, které budou s aplikací pracovat. Preferované ovládání se může u různých skupin uživatelů lišit, doporučuje se více možností ovládání aplikace.

- **Paměť uživatele** - Uživatel by neměl být nucen si rozhraní aplikace pamatovat. Je nutné, aby měl uživatel o aplikaci přehled bez nutnosti si její rozhraní pamatovat.

Kromě výše uvedených principů tvorby uživatelského rozhraní, existují principy vztahující se ke GUI[8] neboli grafickému uživatelskému rozhraní. Grafické uživatelské rozhraní je takové rozhraní, kde jsou jednotlivé objekty reprezentovány graficky nebo i zvukově. Uživatel s těmito objekty provádí akce. Manipulace s těmito objekty je prováděna pomocí myši nebo například dotykové obrazovky. GUI je ovládáno pomocí formulářových prvků, menu a tzv. přímé manipulace.




Obrázek 2.6: Přímá manipulace s GUI

Mezi hlavní principy [18, 17] přímé manipulace s GUI patří trvalá viditelnost objektů a akcí. Další vlastností je také okamžitá odezva systému na požadavky uživatele, uživatel je tak stále informován o probíhajích akcích a stavu aplikace. Podstatným faktem, který napomáhá pochopení uživatelského rozhraní, je zobrazení systému jako rozšíření reálného světa. Uživatel je pak schopen se v takovém systému mnohem rychleji zorientovat. Ovladatelnost GUI také zvyšuje možnost návratu zpět z provedené akce a toleranci k chybám uživatele.

### 2.2.2 Testování uživatelského rozhraní

Testování uživatelského rozhraní aplikace lze definovat jako ověření, že uživatelské rozhraní aplikace splňuje všechny zadáním specifikované požadavky. Požadavky mohou být vzhledové, funkční nebo například systémově orientované. Testování GUI se může zdát jako triviální problém, ale u rozsáhlejších aplikací obsahujících více možných způsobů interakce s uživatelem a s nimi související kombinace událostí může být testování neefektivní. Při vzájemné interakci uživatele s rozhraním aplikace dochází ke změně stavu, jinak řečeno

[8] Graphical User Interface

logiky aplikace. Uživatel vykoná událost, která má za následek vykonání odpovídajícího obslužného kódu, který mění stav aplikace. Testování tedy zahrnuje vykonání sady úloh, porovnání jejich výsledků s očekávanými výsledky a také schopnost vykonání dané sady úloh s odlišnými událostmi, které mohou ovlivnit výsledek testování, jako například události od vstupních periferií.

Testování GUI bývá velice často prováděno pomocí tzv. testovacích případů. Testovací případ[9] popisuje konkrétní akce nebo jednotlivé kroky, které jsou vykonané s určitou softwarovou komponentou, a jejich očekávané výsledky po vykonání jednotlivých kroků. Softwarovou komponentou může být část uživatelského rozhraní nebo také softwarový systém běžící na několika strojích souběžně. Zjednodušeně řečeno, testovací případ je dokument popisující danou činnost, kterou je nutné otestovat. Pro úplné ověření uživatelského rozhraní aplikace je třeba vykonat takovou sadu testovacích úkonů, která pokryje veškerou funkcionalitu aplikace a všech úloh uživatelského rozhraní. U rozsáhlých aplikací může být problematické takovou sadu testovacích úkonů sestavit manuálně. Mnohdy bývá využíváno odvození testovacích případů pomocí heuristik nebo také diagramů, které popisují návrh aplikace, či jiného formalismu [16].

Na testování GUI aplikací existuje celá řada testovacích technik. Jednou z nich, která je prováděna automatizovaně, je *Data-driven* [14] testování. Tato technika využívá testovací skript, který pracuje s daty uloženými v databázi. Využívá se především na oveření správnosti dat, se kterými mohou pracovat objekty GUI. Další technikou, která je využívána pro odhalování nových chyb, které se mohou objevit při vývoji nových verzí aplikace, je regresní testování [15]. Hlavní princip této techniky spočívá v zamezení vykonaným změnám, aby způsobily nové chyby a nezasahovaly do jiné části systému. Při regresním testování je využíváno informací z aktuální verze aplikace, které pomáhájí určit, že aplikace funguje podle očekávání. Další automatizovanou metodou odhalování chyb je *Test Monkeys* [14]. *Test Monkeys* testuje GUI prostřednictvím náhodných akcí, které mohou být vykonávány v různých postupech. Tato metoda je využívána také v různých modifikacích, které se odlišují v mechanismu výběru náhodných akcí pro testování. Výhodou *Test Monkeys* je fakt, že tato metoda neví nic o GUI. Lze ji tedy využít v širokém spektru aplikací a v libovolném stádiu vývoje. Možným úskalím náhodnosti výběru akcí je však možná neočekávaná změna GUI, která provede změnu stavu aplikace, což může vést k nepředvídatelným výsledkům.

Testování může být prováděno manuálně, pomocí lidské inteligence uživatelů, nebo automatizovaně, pomocí počítačových nástrojů. Počítačové nástroje dokážou simulovat interakci s aplikací podobnou člověku jako například pohyb myši nebo stisk klávesy. Tento způsob bývá využíván pro testování rozsáhlých systémů. U některých testů, které jsou specializovány na určitou konkrétní vlastnost uživatelského rozhraní, jako například intuitivnost, je nutné provést testování manuální, kde je zapotřebí lidský faktor, který umožní sledovat způsob provedení daného testu. Manuální testování je prováděno uživateli, kteří obvykle provádějí test, který je zaměřen na zkoumání dané vlastnosti. Důležitou vlastností prováděných testů je přesná specifikace jejich účelu, na základě kterého je vytvořeno zadání testovacích úkolů. Dále je třeba specifikovat měřené vlastnosti, které tvoří výstup testu a umožňují vyhodnotit, zda dané uživatelské rozhraní odpovídá dané vlastnosti. Automatizované testování GUI je však v některých případech efektivnější a spolehlivější než testování manuální. Nabízí možnost většího pokrytí testů a menší náklady na lidské zdroje. Mnoho automatizovaných nástrojů ale pracuje s testovacími skripty, které musí být upraveny pro každý test.

[9] Anglicky *test case*

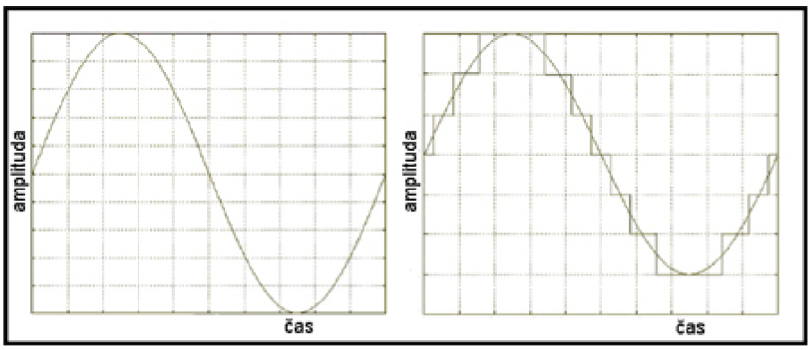
## 2.3 Počítačová hudba a zvuk

Zvuk [10] lze charakterizovat jako podélné mechanické vlnění s odpovídající frekvencí od 16 Hz do asi 20 kHz, které je schopno vyvolat v lidském uchu vjem. Neslyšitelné vlnění s frekvencí nižší než 16 Hz nazýváme infrazvukem, neslyšitelné vlnění s frekvencí vyšší než 20 kHz nazýváme ultrazvukem. Zdroj zvukového vlnění se nazývá zdroj zvuku a hmotné prostředí, ve kterém se toto vlnění šíří, jeho vodič. Zvukovým vodičem bývá nejčastěji vzduch, který zprostředkovává spojení mezi zdrojem zvuku a jeho přijímačem uchem nebo například mikrofonem. Zvuky se však šíří i kapalinami jako například vodou a pevnými látkami, což mohou být kupříkladu stěny domu.

Zdrojem zvuku může být každé chvějící se těleso. Kvalitu vlnění v okolí zdroje zvuku však neovlivňuje jen jeho chvění, ale i okolnost, jestli je tento předmět dobrým nebo špatným zářičem zvuku. Tato vlastnost závisí hlavně na geometrickém tvaru zářiče. Zdrojem zvuku mohou být kromě těles kmitajících vlastními kmity i reproduktory zvuku.

Zvuky lze rozdělit na hudební, neboli tóny, a nehudební, neboli hluky. Tóny vznikají při pravidelném, v čase periodicky probíhajícím pohybu, kmitání. Při jejich poslechu vzniká v uchu obvykle příjemný vjem. Zdrojem hudebních zvuků mohou být například lidské hlasivky či hudební nástroje. Jako hluky označujeme nepravidelné vlnění vznikající jako složité nepravidelné kmitání těles nebo krátké nepravidelné rozruchy, srážku dvou těles nebo výstřel apod.

Zvuk z hlediska zpracování signálů lze charakterizovat jako spojitou analogovou informaci. Pro počítačové zpracování a analýzu zvuku je nutné jej nejprve převést do digitální formy a následně uložit. V následujících kapitolách bude popsán proces digitalizace zvuku, jeho možná komprese a způsoby uložení.




Obrázek 2.7: Digitalizace zvuku[10]

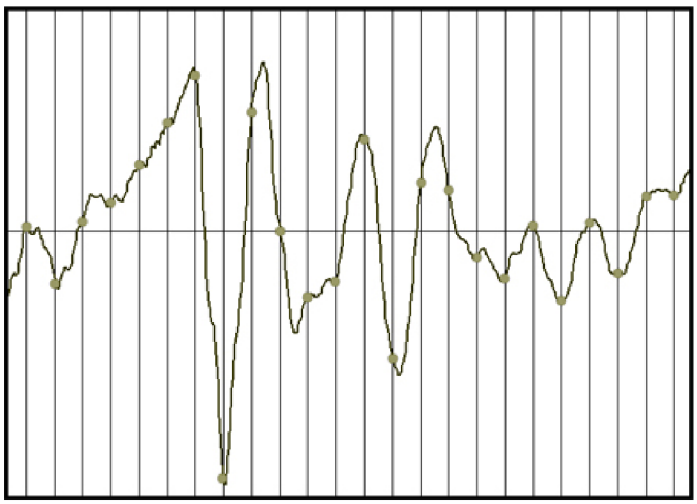
### 2.3.1 Digitalizace zvuku

Při záznamu zvuku je analogový signál teoreticky přesnou kopií původní zvukové vlny. Proces převodu analogového, spojitého signálu do digitální, diskrétní podoby je snahou o nalezení číslicové reprezentace původní zvukové vlny bez ztráty jejích vlastností.

Převod zvuku neboli akustického signálu do digitální podoby [11] je prováděn A/D převodníkem[11], který je dnes součástí každé zvukové karty nebo například mobilního telefonu.

[10] Převzato z http://lide.uhk.cz/fim/student/kubaji1/digitalizace.htm
[11] Analogově-digitální převodník

Analogový signál je nejprve navzorkován a poté kvantován. Při vzorkování je v pravidelných intervalech zaznamenávána aktuální hodnota vstupního analogového signálu. Takto zaznamenané hodnoty představují digitální reprezentaci analogového signálu.

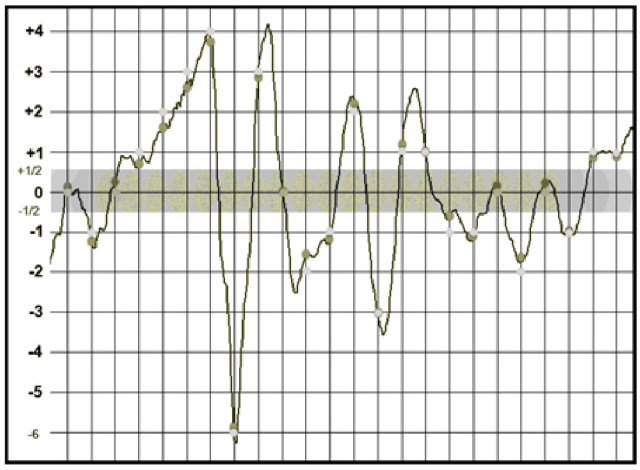


Obrázek 2.8: Vzorkování zvuku[12]

Kvalitu digitální reprezentace lze ovlivnit hodnotou vzorkovací frekvence, která udává frekvenci měření hodnoty analogového signálu. Při vzorkování je také nutné dodržet tzv. Shannonův teorém [13], který říká, že vzorkovací frekvence by měla být minimálně dvakrát větší, než je nejvyšší frekvence vyskytující se ve vstupním signálu. Při nedodržení tohoto teorému dojde k nenávratnému zkreslení signálu. Tento jev se nazývá aliasing. Dopad aliasingu lze omezit tzv. antialiasingovým filtrem, což je vysokofrekvenční filtr zařazený před A/D převodníkem. Nedovolí tak frekvencím vyšším než je polovina vzorkovací frekvence vstoupit do A/D převodníku. Proces vzorkování je zobrazen na obrázku 2.8.

Další proces, který může ovlivnit kvalitu digitální reprezentace je kvantizace. Při kvantizaci je důležitá tzv. bitová hloubka. Bitová hloubka je určena počtem bitů a vyjadřuje rozsah hodnot jednotlivých vzorků. Aby bylo možné určit, které hodnoty má po kvantování nabývat určitý vzorek, je třeba rozdělit prostor kolem jednotlivých naměřených hodnot (vzorků) na toleranční pásy. Všem vzorkům, které se nacházejí v daném tolerančním pásu, je při kvantování přiřazena stejná hodnota. Proces kvantizace je naznačen na obrázku 2.9. Kvantované hodnoty se ve většině případů liší od skutečných navzorkovaných hodnot. Velikost kvantizační chyby je vzdálenost mezi kvantovanými a původními navzorkovanými body. Velikost této chyby se pohybuje v intervalu od $-\frac{1}{2}$ až do $\frac{1}{2}$ kvantizační úrovně. Obecně platí, že čím větší je bitová hloubka, tím lepší je dynamika zvuku, obsahuje méně šumu a je možná jeho kvalitnější reprodukce. Standardní audio CD obsahuje záznam s bitovou hloubkou 16 bitů a vzorkovací frekvencí 44100 Hz.

[12] Převzato z http://lide.uhk.cz/fim/student/kubaji1/digitalizace.htm




Obrázek 2.9: Kvantizace zvuku[13]

### 2.3.2 Komprese a uložení zvuku

Před uložením digitalizovaného zvuku je nutné zvolit zvukový formát a provést kompresi získaných zvukových informací. Komprese je prováděna především pro snížení velikosti digitalizovaných dat. Ke kompresi zvuku je využíván tzv. audio kodek neboli kódovací a dekodovací programy, které umožňují uložení zvukových informací na datový nosič s odpovídající kompresí dat a jejich případné znovunačtení. Kvalitu digitalizovaného zvuku tedy ovlivní i zvolený zvukový formát a audio kodek, což spolu úzce souvisí.

Téměř všechny používané formáty kromě formátu WAV[14] využívájí kompresi [12]. Kompresi lze rozdělit na ztrátovou a bezztrátovou. Ztrátová komprese snižuje velikost uložených dat, avšak na úkor zvukové kvality. U bezztrátové komprese nedochází ke ztrátě dat ani ke snížení kvality zvuku. Ztrátové kompresní algoritmy provádí zmenšení objemu dat za cenu ztráty méně důležitých informací, které jsou při nedokonalosti lidského sluchu zanedbatelné.

Formát WAV nebo také WAVE je neztrátový a nekomprimovaný formát vytvořený společnostmi IBM a Microsoft pro ukládání zvuku na PC. K uchování informací využívá kódování PCM[15]. Zvuk je uchováván přesně tak, jak byl digitalizován. Jeho zpracování je tedy snadné a výpočetně nenáročné, proto je často využíván při úpravách, převodech a archivaci zvuku. Nevýhodou je však vyšší paměťová náročnost daná nekomprimovaným uložením.

Pro uložení zvukových záznamů se dnes mnohem častěji používají ztrátové komprimované formáty především kvůli velikosti uložených dat. Komprese dat je určena tzv. datovým tokem, který je udáván v jednotkách kilobitů za sekundu. Nejvíce používaným formátem je dnes MP3[16] založený na kompresním algoritmu definovaném skupinou MPEG[17]. Umožňuje kompresi zvukových informací až desetkrát, přičemž úroveň komprese závisí na velikosti datového toku. MP3 se snaží odstranit redundanci zvukového signálu na základě psychoakustického modelu. Tedy ze vstupního signálu jsou odebrány informace, které člověk nevnímá.

Mezi další využívané kompresní formáty patří například WMA, AIFF, AAC a OGG Vorbis.

[13] Převzato z http://lide.uhk.cz/fim/student/kubaji1/digitalizace.htm
[14] Waveform audio file format
[15] Pulse Code Modulation - pulzně kódová modulace
[16] MPEG-2 Audio Layer III
[17] Motion Picture Experts Group

# Kapitola 3

# Návrh aplikace

Tato kapitola bude věnována vymezení požadovaných funkcí na aplikaci a formulaci jejich cílů. Dále bude prezentován návrh řešení obsahující strukturu aplikace a možnosti řešení některých požadavků specifikovaných v zadání práce. Závěrem této kapitoly bude diskutována forma metadat pro efektivní memorizaci skladby společně s testováním uživatelského rozhraní a funkcí aplikace.

## 3.1 Zaměření aplikace

Cílem této práce je navrhnout a implementovat uživatelské rozhraní aplikace, která bude využívat metainformace k hudební skladbě a umožní přehrávat skladbu po taktech nebo celých částech. Aplikace bude využívána především hudebníky, je tedy nutné analyzovat jejich potřeby a navrhnout řešení poskytující efektivní uživatelské rozhraní pro výuku a splňující všechny jejich potřeby.

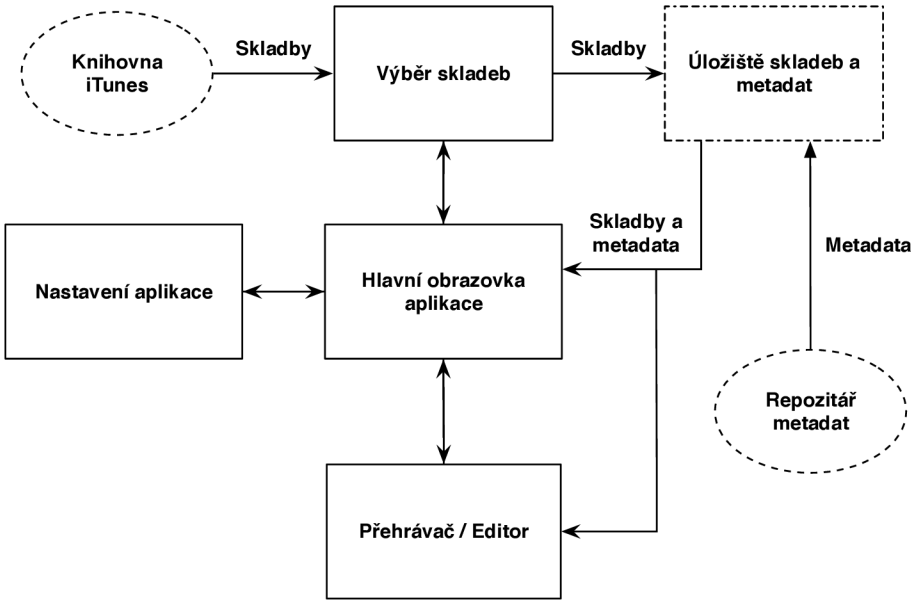
Mezi hlavní požadavky patří přehrávání skladby s možností navigace ve skladbě pomocí rychlého posunu vpřed nebo vzad, přesné nastavení aktuální pozice ve skladbě, změna hlasitosti a rychlosti přehrávání nebo možnost výběru skladeb dle preferencí uživatele. Nejdůležitějším požadavkem je možnost výběru úseku skladby, ať už s přesností danou metadaty nebo ručním nastavením, jeho uložení a případná modifikace počátku a konce výběru. Pro efektivní memorizaci by aplikace měla poskytovat cyklické přehrávání vybraných úseků či celé skladby. Je tedy nutné definovat základní část, po které bude přehrávání aplikace realizováno, a specifikovat strukturu metadat umožňující toto přehrávání. Návrh struktury metadat bude dále popsán v kapitole 3.3.

Pro zachování zvyklostí platformy iOS je vhodné umožnit propojení aplikace s knihovnou iTunes. Tento přístup poskytuje uživateli snadnou manipulaci při nahrávání skladeb do zařízení. Je tedy nutné navrhnout rozhraní, kterým budou načítány skladby z knihovny iTunes, a také, jakým způsobem budou jejich data a metadata v aplikaci uchovávána.

Největší důraz však musí být kladen na uživatelské rozhraní aplikace. Aplikace musí poskytovat jednoduché intuitivní a efektivní rozhraní, které umožní každému uživateli bez nutnosti toho, aby byl hudebník, s aplikací pracovat. Jelikož je aplikace vyvíjena na platformě iOS, uživatelské rozhraní také musí být v souladu s HIG, které svou poměrně striktní strukturou udržují konzistenci aplikací tedy jejich jednotný styl ovládání. V neposlední řadě by aplikace měla poskytovat možnost vizualizace skladby a jejich vybraných úseků pro poskytnutí lepší představy o aktuálně přehrávaném celku, umístění výběru či navigaci ve skladbě.

## 3.2 Návrh řešení

Po analýze všech požadovaných vlastností a cílů jsem navrhl blokovou strukturu aplikace. Tato struktura neodráží přímo jednotlivé obrazovky uživatelského rozhraní, poskytuje však náhled na koncepci aplikace a její způsob činnosti. V následujícím textu budou popsány jednotlivé entity aplikace, návrh jejich funkcionality a uživatelského rozhraní. Struktura aplikace je zobrazena na obrázku 3.1.




Obrázek 3.1: Blokové schéma aplikace

### 3.2.1 Hlavní obrazovka aplikace

Po spuštění aplikace bude uživateli zobrazena hlavní obrazovka aplikace. Tato obrazovka bude tvořit jádro aplikace a umožňovat přístup k dalším funkcím a obrazovkám. Jejím hlavním úkolem bude prezentovat uživateli skladby uložené v aplikaci a informace o nich.

Ke každé skladbě zde budou vizualizovány její vybrané a uložené úseky, titulní obrázek, název, album a interpret skladby. Kromě informací o skladbě by mělo zobrazení skladby obsahovat informaci, zda jsou ke skladbě načtena metadata či ne. Podstatnou vlastností této obrazovky je také možnost přímého přehrávání skladby nebo vybraných úseků, bez nutnosti přechodu na obrazovku přehrávače. Při přehrávání by mělo být patrné, která skladba je aktuálně přehrávána.

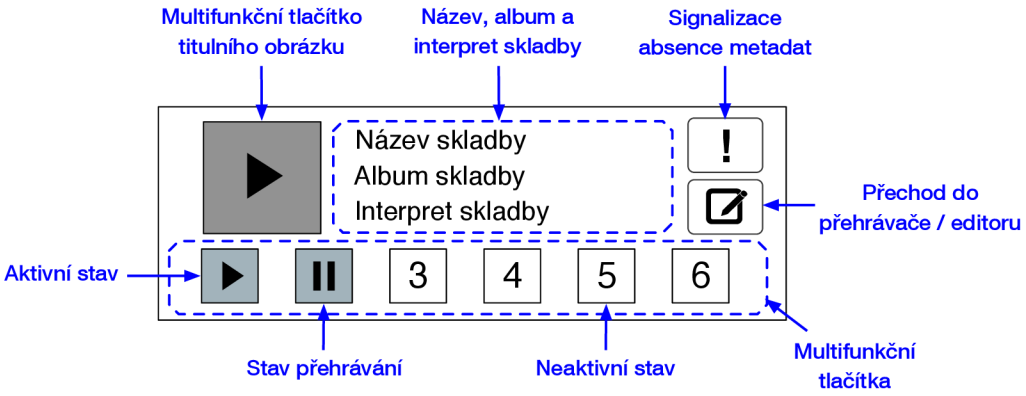
Pro přímý přístup k více skladbám je vhodné využít zobrazení v seznamu, který zároveň tvoří rozcestník pro další možnosti práce se skladbou. Seznam poskytuje hierarchické oddělení jednotlivých skladeb jako jednotlivých řádků a také umožňuje zobrazit neomezený počet skladeb díky možnosti rolování obsahu seznamu. Všechny ovládací prvky vztahující se k dané skladbě budou zobrazeny společně s jejími informacemi v seznamu. Jedná se o přechod do přehrávače skladby, načtení metadat ke skladbě nebo přehrávání dané skladby a jejich úseků.

Prvkem, který je charakteristický pro navigaci v aplikacích na platformě iOS, je navigační lišta umístěná v horní části obrazovky. Na navigační liště budou situovány ovládací

prvky, které nejsou vázány ke konkrétním skladbám, jako například přechod na obrazovku výběru skladeb z knihovny iTunes nebo do nastavení aplikace. Navigační lišta slouží také jako informační místo pro vizualizaci průběhu přehrávání některé skladby a s ní souvisejících informací. Po stisknutí navigační lišty při přehrávání skladby dojde k narolování seznamu na aktuálně přehrávanou skladbu, což výrazně usnadní orientaci v případě, že seznam obsahuje velké množství skladeb.

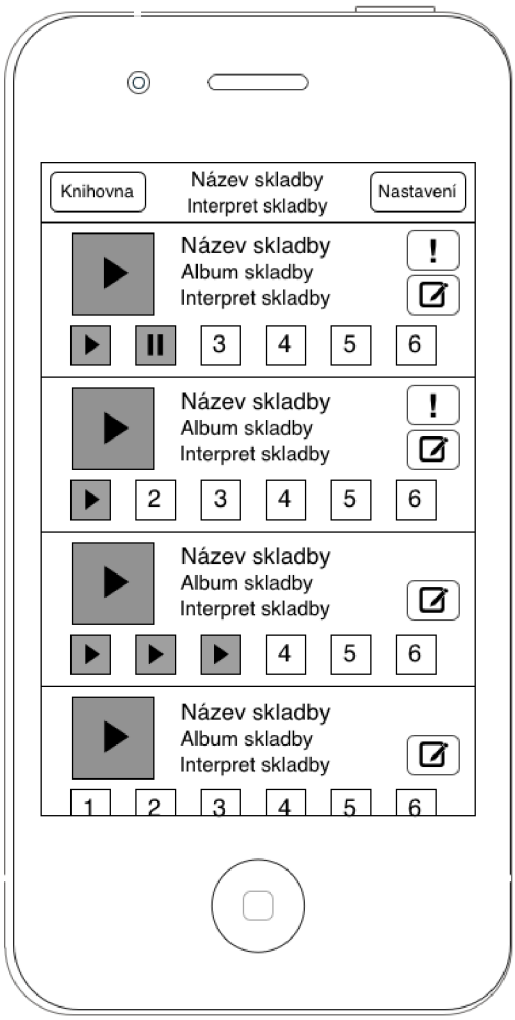

Obrázek 3.2: Popis elementů navigační lišty

Jelikož mobilní zařízení disponuje pouze omezenou velikostí obrazovky, je nutné provést uspořádání ovládacích prvků vztahujících se k určité skladbě. Především z tohoto důvodu je nutné omezit maximální počet možných výběrů ke skladbě na šest, což se jeví pro účely výuky jako dostačující počet. U každé skladby je tedy nutné zobrazit šest možných výběrových pozic, na které je možné uložit určitý úsek skladby. Jako ideální řešení se zde nabízí využití multifunkčních tlačítek, která mění svou funkcionalitu na základě stavu, ve kterém se nacházejí. Mezi stavy lze zařadit neaktivní stav, kdy není na dané pozici výběr uložen, aktivní stav, při kterém je výběr na dané pozici uložen, a stav, kdy probíhá přehrávání daného úseku. Tento způsob řešení je také vhodný využít při zobrazení titulního obrázku skladby, který může zároveň poskytovat možnost přehrání celé skladby. Zbývající ovládací prvky, které jsou úzce spjaty s danou skladbou, jako přechod do přehrávače skladby a signalizace chybějících metadat u skladby, budou umístěny v pravé části řádku seznamu, jelikož z hlediska HIG platformy iOS jsou zde zpravidla umístěny ovládací prvky vztahující se k danému řádku.

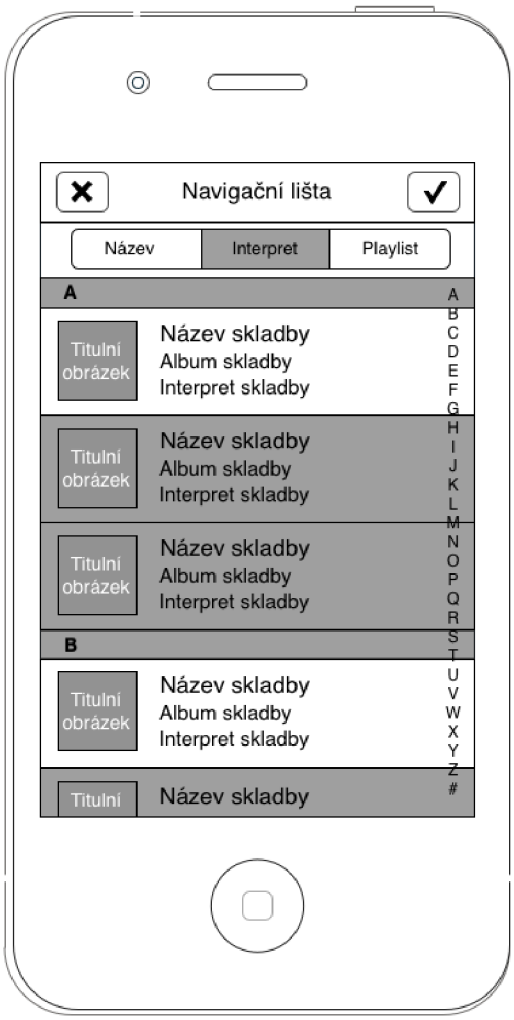



Obrázek 3.3: Popis elementů řádku seznamu skladeb

Tato obrazovka je navržena pouze pro vertikální orientaci zařízení. V horizontální orientaci je zobrazení seznamu nevhodné, jelikož je vidět jen velmi málo řádků seznamu. Návrh rozmístění prvků uživatelské rozhraní hlavní obrazovky zachycuje obrázek 3.4.




Obrázek 3.4: Hlavní obrazovka aplikace



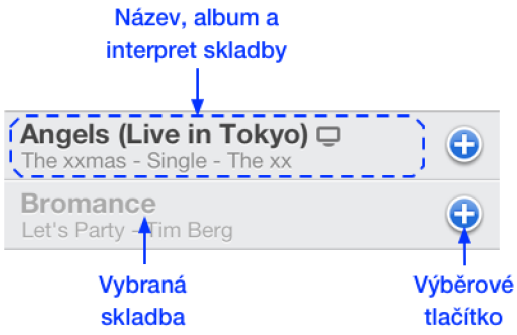

Obrázek 3.5: Obrazovka výběru skladeb

### 3.2.2 Výběr skladeb

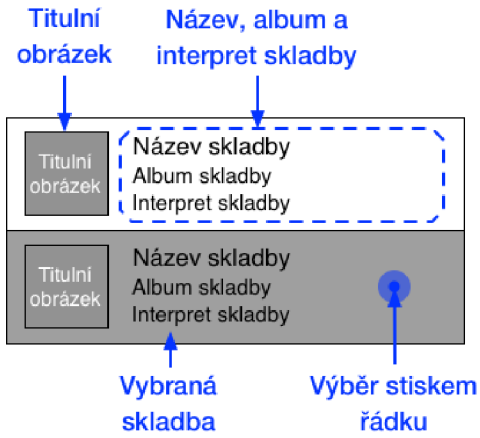
Propojení aplikace s knihovnou iTunes je zprostředkováno obrazovkou výběru skladeb. Jejím úkolem je umožnit uživateli načíst skladby z knihovny iTunes a provést jejich uložení do interní paměti aplikace.

Přístup ke skladbám v knihovně iTunes poskytuje Media Player framework. Kromě samotných dat a metadat skladeb umožňuje využít předdefinované uživatelské rozhraní, které poskytuje výběr a načtení skladeb. Toto rozhraní je však pro účely této aplikace nevhodné, protože neumožňuje zrušit výběr aktuálně vybraných skladeb a lze opakovaně vybrat již zvolenou skladbu. Dalším důvodem pro návrh vlastního uživatelského rozhraní výběru skladeb je také nemožnost řadit skladby dle zvolených kritérií.

Při výběru skladeb by uživatel měl mít k dispozici základní informace o skladbě jako například titulní obrázek, název, album a interpreta skladby. Kromě informací vztahujících




Obrázek 3.6: Ukázka uživatelského rozhraní výběru skladeb, které poskytuje Media Player framework




Obrázek 3.7: Návrh uživatelského rozhraní výběru skladeb

se k dané skladbě by mělo být na první pohled patrné, zda je skladba vybrána nebo ne. Skladby by také mělo být možné řadit dle interpreta, názvu nebo také playlistů, do kterých jsou zařazeny.

Především kvůli proměnlivému počtu skladeb v knihovně iTunes, které bude nutné zobrazit, je vhodné využít zobrazení v seznamu. Seznam poskytuje celistvé zobrazení informací o skladbě, tedy každé skladbě odpovídá jeden řádek seznamu a nabízí zřetelnou signalizaci, zda je skladba vybrána ve formě změny barvy pozadí řádku. V případě velkého počtu zobrazených skladeb urychluje pohyb a orientaci v seznamu rozdělení skladeb do sekcí dle abecedy. K abecedním sekcím je dále možné přímo přistupovat pomocí abecední lišty, která se vyskytuje v pravé části seznamu.

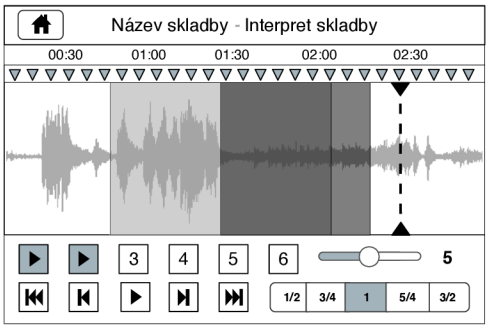
Změnu řazení zobrazených skladeb je vhodné umístit v těsné blízkosti seznamu, aby bylo na první pohled patrné, k čemu se daný ovládací prvek vztahuje. Jelikož byla specifikována tři požadovaná kritéria řazení, jako ovládací prvek jsem zvolil segmentové tlačítko, které je na platformě iOS pro výběr z několika možností charakteristické. Aktuálně vybraný segment tedy určuje kritérium seřazení seznamu skladeb, kdy aktivní je vždy pouze jedna volba. Segmentové tlačítko je umístěno v horní časti seznamu.

Pro zachování konzistence vzhledu jednotlivých obrazovek a usnadnění navigace v aplikaci je v horní části okna umístěna navigační lišta. Tato lišta obsahuje ovládací prvky, které nejsou přímo spjaty s některou skladbou. Obsahuje tedy tlačítko pro možnost přechodu zpět na hlavní obrazovku aplikace a pokud je provedena změna ve výběru skladeb, uživatel vybral skladbu, která dříve nebyla ve výběru, nebo naopak, dojde k zobrazení ovládacích prvků, které buďto potvrzují aktuální stav výběru skladeb, nebo zruší změny ve výběru. Při potvrzení změn ve výběru jsou skladby načteny do interního úložiště aplikace a je proveden přechod zpět na hlavní obrazovku aplikace.

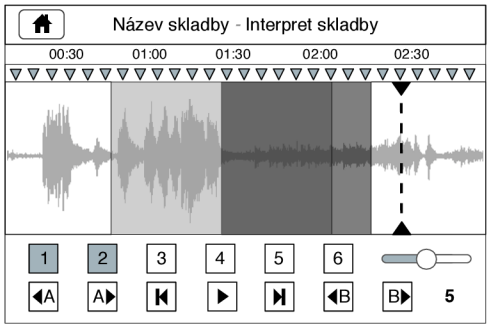
Tato obrazovka je opět navržena pouze pro vertikální orientaci zařízení z důvodu nevhodnosti použití seznamu v horizontální orientaci. Návrh uživatelského rozhraní obrazovky výběru skladeb zobrazuje obrázek 3.5.

### 3.2.3 Přehrávač / Editor

Jádrem funkcionality aplikace je obrazovka pracující jako přehrávač a zároveň editor skladby. V prvním návrhu blokové struktury aplikace byla funkcionalita přehrávače a editoru oddělena. Pozdější analýzou se však ukázalo, že tyto dvě obrazovky poskytují funkce, které spolu úzce souvisí a také, že editor poskytuje velkou většinu funkcí přehrávače. Proto tedy jsem návrh přepracoval do formy, kdy přehrávač a editor jsou prezentovány na jedné obrazovce aplikace. Po sloučení těchto obrazovek byl návrh také optimalizován pro efektivnější využití prostoru obrazovky. V následujícím textu bude prezentován již finální a realizovaný návrh této obrazovky.




Obrázek 3.8: První návrh obrazovky přehrávače

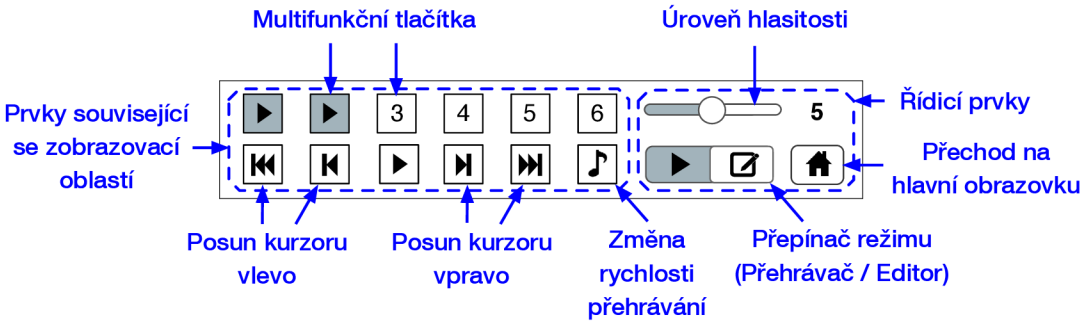



Obrázek 3.9: První návrh obrazovky editoru

Přechod na tuto obrazovku s danou skladbou je zajištěn stisknutím tlačítka v seznamu skladeb na hlavní obrazovce aplikace. Hlavními prvky, které tato obrazovka poskytuje, jsou vizualizace skladby a jejích metadat, vytváření, uložení, úprava a zobrazení vybraných úseků skladby. Skladba je vizualizována především pro lepší orientaci a navigaci ve skladbě, vizualizace vybraných úseků skladby usnadňuje proces zadávání a úpravy formou přímé manipulace. Dále to jsou především funkce přehrávače jako například možnost rychlého nebo skokového posunu ve skladbě, změna hlasitosti a rychlosti přehrávání skladby nebo možnost přesného nastavení aktuální pozice ve skladbě.

Protože vizualizace skladby je využívána nejen staticky pro navigaci ve skladbě, ale také slouží pro přímou manipulaci formou dotykových gest při zadávání úseků či posunu ve skladbě, je nutné co nejvíce uzpůsobit uspořádání prvků uživatelského rozhraní tak, aby vizualizované oblasti bylo věnováno co nejvíce prostoru. Z tohoto důvodu je porušena koncepce navigační lišty, která je využívána na všech obrazovkách aplikace. Navigační lišta zde nemá velký význam, obsahovala by pouze jeden ovládací prvek pro přechod zpět na hlavní obrazovku aplikace. Mnohem efektivnějším řešením je využití prostoru, který by byl vyplněn navigační lištou, zobrazovací oblastí a ovládací prvek pro přechod na hlavní obrazovku aplikace umístit k ostatním ovládacím prvkům této obrazovky.

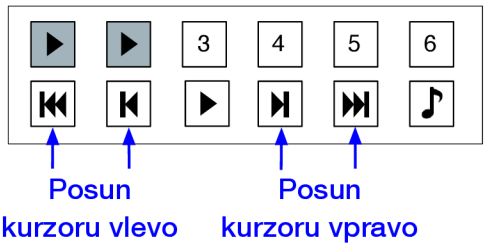
Důležité je také uspořádání ovládacích prvků obrazovky tak, aby spolu významově související byly umístěny pohromadě a nezabíraly příliš mnoho prostoru. Ovládací prvky lze pomyslně rozdělit na prvky řídicí a prvky, které souvisejí se zobrazovací oblastí. Mezi řídicí prvky lze zařadit již zmíněné tlačítko pro přechod na hlavní obrazovku aplikace, změnu hlasitosti, rychlosti přehrávání a také ovládací prvek, který určuje, zda je obrazovka v režimu přehrávače nebo editoru. Zvolený režim určuje především funkcionalitu při manipulaci se zobrazovací oblastí. Prvky vztahující se k zobrazovací oblasti zahrnují šest multifunkčních tlačítek, pod které je možné uložit výběr ve skladbě, a prvky, které umožňují



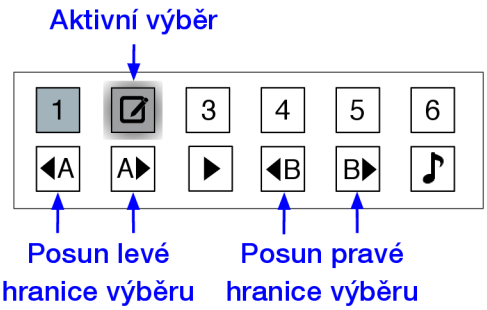

Obrázek 3.10: Ovládací prvky přehrávače / editoru

nastavení přesné pozice ve skladbě, ať už se jedná o pozici aktuálního kurzoru, nebo začátku či konce výběru, a to s přesností danou metadaty skladby. Omezení maximálního počtu možných výběrů ve skladbě bylo diskutováno v kapitole 3.2.1.

Pro ovládací prvek, který provádí přechod zpět na hlavní obrazovku aplikace, jsem zvolil tlačítko obsahující ikonu, která signalizuje přechod zpět a je obvykle umístěna na navigační liště. Alespoň tak je částečně zachována konvence platformy iOS a konzistence ovládání aplikace. Změna úrovně hlasitosti zahrnuje více možných hodnot, respektive deset úrovní a vypnutí zvuku. Je tedy vhodné zvolit posuvník, který zabezpečí potřebnou funkcionalitu tohoto prvku. Změna rychlosti přehrávání je omezena na pět možných výběrů a to od 50% až do 150% skutečné rychlosti skladby. Rychlost lze volit na segmentovém tlačítku s pěti segmenty, z nichž vždy pouze jeden je aktivní. Takový ovládací prvek však zabírá relativně mnoho prostoru vzhledem ke své funkcionalitě. Proto bude prezentován pouze ve formě jednoduchého tlačítka, které dále zpřístupní zmiňované segmentové tlačítko. Posledním řídicím prvkem je volba režimu obrazovky. Režim přehrávače nebo editoru lze ideálně volit jednoduchým přepínačem, který obsahuje dva stavy. Přepínač nezabírá příliš mnoho prostoru a je na první pohled patrné, že se jedná o volbu mezi režimem přehrávače a editoru.



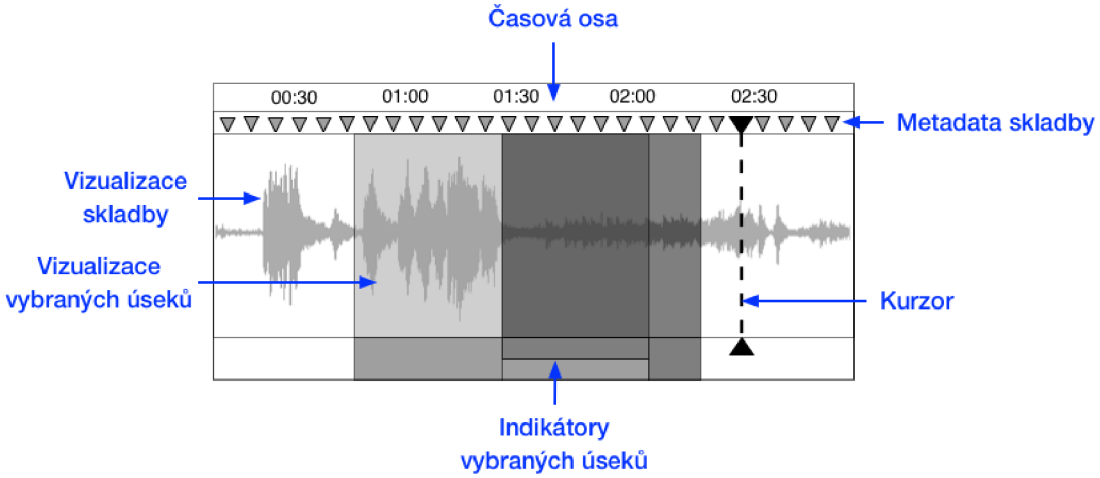

Obrázek 3.11: Ovládací prvky v režimu přehrávače




Obrázek 3.12: Ovládací prvky v režimu editoru

Funkcionalita ovládacích prvků, které souvisejí se zobrazovací oblastí, závisí na aktuálně zvoleném režimu obrazovky. V režimu přehrávače prvky slouží pro nastavení aktuální pozice kurzoru v zobrazovací oblasti s přesností danou metadaty. Je tedy umožněn posun o jeden nebo čtyři úseky dané metadaty vpřed a vzad. V režimu editoru tyto prvky poskytují přesné nastavení hranic aktivního výběru ve skladbě o jeden úsek vpřed nebo vzad. Multifunkční tlačítka pro uložení úseku skladby, která již byla použita na hlavní obrazovce aplikace,

zde mají také dvojí úlohu, která je závislá na zvoleném režimu. V režimu přehrávače poskytují stejné funkce jako na hlavní obrazovce aplikace, tedy nacházejí se v neaktivním stavu, kdy není na dané pozici výběr uložen, aktivním stavu, při kterém je výběr na dané pozici uložen, a stavu, kdy probíhá přehrávání daného úseku. Pokud je nastaven režim editoru, mění se funkcionalita tlačítek, která se nachází buďto v neaktivním stavu, kdy není na dané pozici výběr uložen, aktivním stavu, při kterém je výběr na dané pozici uložen, a stavu úprav, kdy probíhá úprava zvoleného úseku skladby. Přechod do stavu úprav je signalizován zvýrazněním daného tlačítka a změnou jeho ikony. Ve stavu úprav je umožněno měnit počáteční a koncovou hranici úseku pomocí ovládacích prvků nebo přímou manipulací v zobrazovací oblasti. Ostatní úseky, které nejsou ve stavu úprav, protože vždy může být pouze jeden úsek ve stavu úprav, nelze modifikovat. Pokud nejsou ke skladbě načtena metadata, ovládací prvky poskytující posun s přesností danou metadaty jsou neaktivní.




Obrázek 3.13: Zobrazovací oblast přehrávače / editoru

Zobrazovací oblast lze rozdělit do tří částí. Horní část poskytuje přehled o časovém měřítku, je zde tedy zobrazena časová osa, která se přizpůsobuje na základě aktuální hodnoty přiblížení zobrazení. V případě, že ke skladbě byla načtena metadata, jsou zde rovněž zobrazeny jejich ukazatele. Prostřední část poskytuje přehled o audio signálu skladby a vybraných úsecích skladby, které jsou zde vizualizovány. V dolní části se nachází indikátory vybraných úseků, které zlepšují orientaci ve vizualizovaných úsecích v případě jejich překryvu. Zobrazovací oblast také obsahuje kurzor, který určuje aktuální pozici ve skladbě.

Prvkem, který výrazně ovlivňuje uživatelské rozhraní této obrazovky, je systém přímé manipulace se zobrazovací oblastí. Platforma iOS poskytuje důmyslně propracovaný systém gest, které je zde vhodné využít. Je však nutné jednotlivá gesta zvolit tak, aby nedocházelo k rušení jejich účinku nebo jiným kolizím. Dobrým zvykem je také snaha udržet konzistenci používání daných gest na platformě.

Charakteristickým gestem, které je na platformě iOS používáno pro změnu přiblížení obsahu je tzv. *pinch* neboli štípnutí. V zobrazovací oblasti tedy bude sloužit pro změnu hodnoty přiblížení jejího obsahu. Dalším gestem je tzv. *swipe* neboli rychlý posun prstem. *Swipe* je zažitým pohybem pro posun nebo rolování obsahu oblasti. V případě, že bude zobrazovací oblast přiblížena tak, že nebude viditelná celá délka skladby, bude toto gesto sloužit pro rolování obsahu v horizontálním směru. Skoková změna pozice kurzoru bude realizována gestem zvaným *tap* neboli prostým dotykem. Pozice kurzoru bude poté nastavena

na polohu provedeného dotyku. Zbývající akcí, která nesmí v zobrazovací oblasti chybět, je tzv. *drag&hold* neboli podržet a přetáhnout. Nejedná se přímo o určité gesto, spíše o kombinaci více gest. Tato akce bude využita pro posun kurzoru a v režimu editoru na úpravu hranic úseku, který se nachází ve stavu úprav.

Zbývajícím mechanismem, který je nutné pro plnou funkci obrazovky specifikovat, je přidávání nových úseků, jejich odstranění a řešení překryvu jednotlivých výběrů.

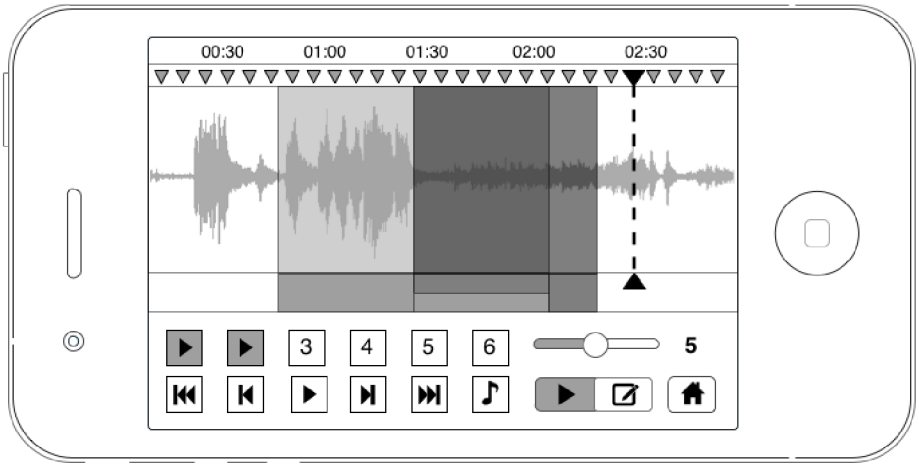
Po přechodu do režimu editoru se nacházejí všechna tlačítka, na která je možné uložit vybraný úsek skladby, v neaktivním stavu, kdy není na dané pozici výběr uložen, nebo v aktivním stavu, při kterém je výběr na dané pozici uložen. Po stisku tlačítka v aktivním stavu přechází do stavu úprav, kdy je možné měnit hranice vybraného úseku. V případě, kdy není na dané pozici výběr uložen, nachází se v neaktivním stavu, je po stisku tlačítka automaticky přidán nový úsek, který začíná na aktuální pozici kurzoru v zobrazovací oblasti.


Obrázek 3.14: Ovládací prvky při možnosti odstranění uložených výběrů

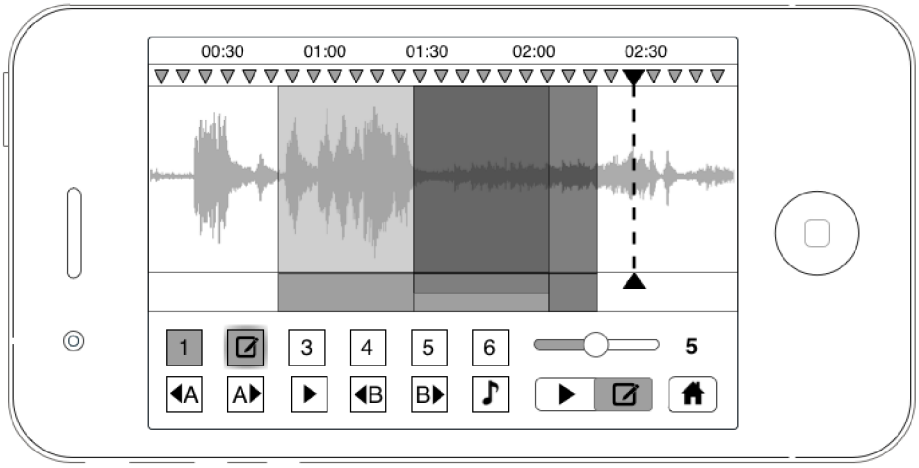
Pro odstranění úseku by však bylo nutné využít speciálního ovládacího prvku. Protože však na platformě iOS je využíván jiný styl odstranění objektů, jako například na domovské obrazovce zařízení při odstranění aplikací, bude tento způsob využit i v aplikaci. Při přidržení prstu nad některým z multifunkčních tlačítek, pod kterými jsou uloženy jednotlivé výběry úseků skladby, dojde k zobrazení malých křížků nad tlačítky, po jejichž stisknutí bude uložený úsek na dané pozici odstraněn. Tímto mechanismem budou zachovány stereotypy ovládání charakteristické pro cílovou platformu a zároveň není nutné rozšířit uživatelské rozhraní o další ovládací prvek.

Vzhledem k omezenému počtu možných výběrů se nabízí řešení jejich překryvu pomocí průhlednosti. Označení každého vybraného úseku bude tedy vizualizováno danou barvou s odpovídající hodnotou průhlednosti, kdy při překryvu více úseků bude výsledná vizualizace tvořena kombinací barev překrývajících se úseků. V krajních případech, kdy dojde k překryvu čtyř a více úseků, může zobrazení působit nezřetelně. Pro lepší orientaci jsem tedy navrhl doplnit vizualizaci úseků o signalizační pásmo, které je umístěno ve spodní části zobrazovací oblasti. V signalizačním pásmu je v každé části na první pohled zřetelné, kolik výběrů je v daném rozmezí, a které to jsou. Vizualizace úseků je naznačena na obrázcích 3.15 a 3.16.

Tato obrazovka je navržena pouze pro horizontální orientaci zařízení. Tuto orientaci jsem zvolil pro maximální využití rozměrů obrazovky zařízení a zároveň tedy maximální velikost zobrazovací oblasti. Návrh uživatelského rozhraní v režimu přehrávače je prezentován na obrázku 3.15. Režim editoru je zachycen na obrázku 3.16.




Obrázek 3.15: Přehrávač / Editor - režim přehrávače




Obrázek 3.16: Přehrávač / Editor - režim editoru

### 3.2.4 Nastavení aplikace

Obrazovka nastavení aplikace je určena k prezentování méně důležitých a doplňujících funkcí aplikace. Přístup na tuto obrazovku je umožněn z hlavní obrazovky aplikace.

Mezi tyto funkce lze zařadit nápovědu k jednotlivým funkcím aplikace, návod ke způsobu ovládání, možnost kontaktovat vývojáře aplikace a nastavení URL[1] adresy repozitáře pro načítání metadat. Účel nastavení URL adresy repozitáře metadat bude podrobněji popsán v kapitole 3.3.

Nápovědu lze zpracovat více způsoby. Pro účely této aplikace, která obsahuje na některých obrazovkách velké množství ovládacích prvků, je efektivní prezentovat funkcionalitu jednotlivých ovládacích prvků přímo na jednotlivých snímcích obrazovky. Nápovědu tedy tvoří snímky obrazovek aplikace, které jsou doplněny vysvětlujícími komentáři. Problematické však může být pochopení využití gest při ovládání některých prvků. Z tohoto důvodu

[1] Uniform resource locator

bude aplikace doplněna ukázkovým videem, ve kterém bude velká většina ovládacích prvků prezentována včetně způsobu použití jednotlivých gest. Návrh aplikace je však koncipován tak, že by nemělo k nepochopení uživatelského rozhraní dojít. Nápověda je zde uvedena spíše jako doplňující prvek.

Možnost kontaktovat vývojáře aplikace je nejčastěji využívána ve formě e-mailu. Ideálním řešením je tedy při zvolení této volby otevřít e-mailového klienta s předvyplněnými informacemi. Ať uživatel využije tohoto způsobu pro nahlášení chyby v aplikaci nebo z jiného důvodu, je také vhodné zprávu automaticky doplnit o systémové informace vztahující se k danému zařízení, ze kterého byla odeslána.

Vzhledem k tomu, že tato obrazovka obsahuje pouze doplňující nebo méně často používané funkce, bylo by možné využít systémového nastavení na platformě iOS. Pro dosažení těchto funkcí by však poté bylo nutné odejít z aplikace do nastavení a následně se vrátit zpět. Efektivnější je tedy zahrnout tyto funkce do aplikace, přestože jejich důležitosti není přikládana taková váha.

## 3.3 Metadata skladby

Úkolem této aplikace je pracovat s explicitně specifikovanými metadaty ke každé skladbě. Pro umožnění efektivní výuky a memorizace skladby je nutné nejprve specifikovat strukturu těchto metadat, jejich způsob využití a mechanismus jejich získání.

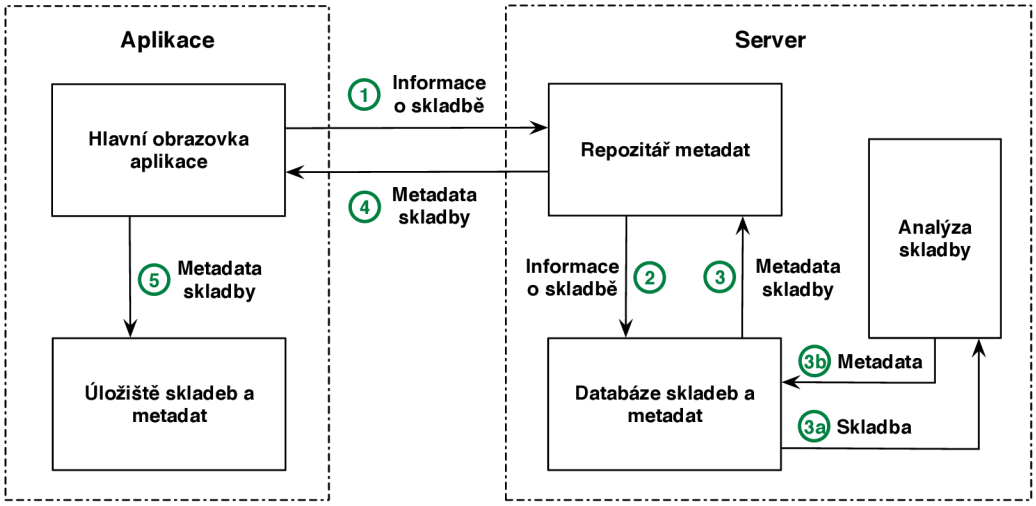
Struktura metadat by se měla vztahovat vždy ke konkrétní skladbě. Jejich využití se předpokládá především při výběru úseku dané skladby, který je následně uložen a případně cyklicky přehráván. Teoreticky by tedy metadata mohla obsahovat libovolné momenty ve skladbě, kterých následně uživatel využije při práci s přehrávačem nebo editorem. Tento fakt však vyžaduje příliš mnoho účasti uživatele při tvorbě metadat, což je při interakci s aplikací vzhledem k jejímu návrhu uživatelského rozhraní nevhodné.

Pro automatizaci procesu získání metadat k dané skladbě se nabízí možnost algoritmicky detekovat ve skladbě určité momenty. Z hlediska hudby by se mohlo jednat o takty, detekce taktů ve skladbě pouze na základě digitalizovaného signálu je však velmi obtížná. Další, a z hlediska zpracování audia jednodušší, variantou je ve skladbě detekovat tzv. *beaty*, jinak řečeno doby, které tvoří jednotlivé takty. Po výběru skladby z knihovny iTunes by tedy mohla být provedena analýza skladby, jejímž výsledkem by byly pozice jednotlivých dob ve skladbě. V mnoha případech by však tento proces vyžadoval manuální korekci výsledků získaných prostřednictvím algoritmického výpočtu. Na přesnosti výsledků se také do značné míry může podílet kvalita audio dat skladby, jako například vzorkovací frekvence a bitová hloubka, a to především z toho důvodu, že knihovna iTunes nezajišťuje kvalitu poskytnutých skladeb. Proces získání metadat ke skladbě však není součástí zadání této práce a tudíž zde není dále diskutován.

Pro účely této aplikace je využito serverové řešení ve formě tzv. *Cloud computing*[2] modelu. Aplikace je schopna komunikovat se serverem, na kterém budou buďto umístěna daná metadata, nebo bude provedena analýza skladby, a její výsledky budou poskytnuty aplikaci. Tento způsob umožní provést analýzu skladby odpovídající digitalizované kvality a případnou manuální korekci jejich výsledků.

Metadata budou na serveru umístěna v repozitáři, který dále zpřístupní dané soubory s metadaty. Především z tohoto důvodu je v nastavení aplikace zahrnuta volba specifikovat URL adresu repozitáře metadat, aby bylo možné využívat více možných umístění serverů

[2]Poskytování služeb či programů uložených na serverech na Internetu




Obrázek 3.17: Návrh komunikace aplikace se serverem

s metadaty. Uživateli tedy bude umožněno například využít vlastní soubory s metadaty, umístí-li je na některý server a vytvoří repozitář odpovídající struktury. Struktura repozitáře metadat je naznačena na obrázku 3.18.

```
<?xml version="1.0" encoding="UTF-8"?>
<metadata>
    <item>
        <name>Název skladby 1</name>
        <url>http://www.metadata.com/Identifier1</url>
        <id>Identifikátor 1</id>
    </item>
    <item>
        <name>Název skladby 2</name>
        <url>http://www.metadata.com/Identifier2</url>
        <id>Identifikátor 2</id>
    </item>
    <item>
        <name>Název skladby 3</name>
        <url>http://www.metadata.com/Identifier3</url>
        <id>Identifikátor 3</id>
    </item>
    <item>
        <name>Název skladby 4</name>
        <url>http://www.metadata.com/Identifier4</url>
        <id>Identifikátor 4</id>
    </item>
</metadata>
```

Obrázek 3.18: Repozitář metadat

```
<?xml version="1.0" encoding="UTF-8"?>
<metadata>
    <beats>
        <beat>1.40</beat>
        <beat>2.05</beat>
        <beat>2.56</beat>
        <beat>3.20</beat>
        <beat>3.55</beat>
        <beat>3.97</beat>
        <beat>4.30</beat>
        <beat>4.85</beat>
        <beat>5.30</beat>
        <beat>5.82</beat>
        <beat>6.20</beat>
        ...
        <beat>187.20</beat>
        <beat>187.86</beat>
        <beat>188.46</beat>
        <beat>188.92</beat>
    </beats>
</metadata>
```

Obrázek 3.19: Soubor s metadaty

Pro efektivní komunikaci aplikace se serverem je nutné také zvolit vhodný způsob uložení metadat. Pro přenos dat síťovou komunikací lze využít libovolný ze serializačních[3] jazyků. V rámci vývoje aplikace v nativním jazyce Objective-C je vhodné využít jazyka XML[4], který poskytuje vhodné prostředky pro strukturovaný zápis metadat, a také proto, že jejich

[3]Serializace je obecně proces, který převádí nějaký objekt do jeho sériové (sekvenční) podoby.
[4]Extensible Markup Language

načtení a deserializace je podporována prostředky jazyka Objective-C. Struktura souboru s metadaty je znázorněna na obrázku 3.19.

Na hlavní obrazovce aplikace jsem navrhl v rámci zobrazení skladby signalizaci, která upozorňuje, že k dané skladbě nejsou načtena metadata. Kromě signalizace po stisku na varovný znak dojde k odeslání požadavku na server, načtení metadat v případě jejich dostupnosti a uložení do interní paměti aplikace.

## 3.4 Možnosti testování

Vzhledem k zaměření této práce je možné aplikaci otestovat více způsoby. V rámci návrhu testování lze rozdělit tento proces do dvou kategorií. Do první kategorie lze zařadit testování zkoumající určitou vlastnost uživatelského rozhraní jako například, zda je rozhraní intuitivní, efektivní, použitelné apod. Jelikož v zadání práce nebyly specifikovány přesné požadavky na vzhled a funkcionalitu jednotlivých prvků uživatelského rozhraní, nebude zde diskutována možnost testování z hlediska ověření funkčnosti daných ovládacích prvků. Druhá kategorie obsahuje testy zabývající se ověřením vlastností aplikace z hlediska výuky za pomocí využití metadat ke skladbě.

Návrh uživatelského rozhraní je především koncipován pro snadnou, intuitivní a rychlou práci s aplikací. Pro ověření těchto vlastností a především toho, zda je rozhraní intuitivní, lze provést kombinaci dvou testů na dané skupině uživatelů. Úkolem těchto testů bude zjistit, zda je uživatel schopný s aplikací pracovat bez nutnosti jeho zaškolení a zda jej aplikace sama navede ke splnění daných úkolů.

Pro získání relevantních výsledků je nutné pro testování zvolit odpovídající skupinu uživatelů. Uživatelé by měli mít alespoň základní zkušenosti s ovládáním aplikací na cílové platformě. Pokud bude testování prováděno bez přítomnosti pozorovatele, je nutné vypracovat testovací protokol, který zajistí pro všechny uživatele jednotné podmínky při testování.

Pro účely provedení těchto testů bude vytvořena testovací sada úkolů, které se vztahují k daným funkcím a ovládacím prvkům uživatelského rozhraní aplikace. Oba testy budou obsahovat stejnou sadu testovacích úkolů. Důležitým prvkem, který tyto testy odlišuje, je formulace zadání daných úkolů. Jeden z testů bude obsahovat po formální stránce co nejvíce abstraktní popis zadání úkolů, kde daný mechanismus provedení úkolu není specifikován. Druhý bude naopak uživateli poskytovat stručný návod na provedení jednotlivých úkolů. V rámci tvorby testovacích úkolů je nutné počítat s možností toho, že uživatel daný úkol nesplní. V případě, že tato situace nastane a další úkoly navazují na nesplněný úkol, musí být definován další, alternativní postup v testu.

Podstatným prvkem testů je také specifikace měřených vlastností. Dobře zvolené vlastnosti mohou výrazně usnadnit vyhodnocení testů a tedy zkoumaných vlastností. V rámci výše zmíněných testů budou měřena dvě kritéria. Prvním z nich bude časový údaj, po který uživatel prováděl daný úkol, druhým bude počet kliknutí, neboli v terminologii dotykových zařízení dotyků, na obrazovce zařízení.

Testy mohou být provedeny v libovolném pořadí, je zde však vliv učení, který může ovlivnit výsledek testu, který je proveden jako druhý v pořadí. Pro získání relevantních výsledků je možné provést s danou skupinou uživatelů vždy pouze jeden ze dvou testů. Pro zjištění a vyhodnocení by byly následně porovnány výsledky provedení jednotlivých testů. Další možností je provést oba testy s danou skupinou uživatelů a následně nechat jinou skupinu uživatelů vypracovat testy v opačném pořadí. Zkoumané vlastnosti by poté byly vyhodnoceny porovnáním výsledků mezi oběma skupinami uživatelů.

Druhou kategorii testů zaměřených na ověření vlastností aplikace z hlediska výuky a me-

morizace lze nejlépe otestovat v praxi. Testovací skupinou uživatelů by tedy mohli být hudebníci, zpěváci nebo například kdokoliv, kdo preferuje učení formou poslechu. V závislosti na uživateli by bylo nutné zvolit odpovídající testovací mechanismus. Pro hudebníky by tomu mohla být například sloka nebo refrén skladby, pro běžného uživatele několik veršů básně.

Uživatel by tedy provedl test ve formě naučení se daného celku s běžným hudebním přehrávačem, který je například k dispozici v zařízeních se systémem iOS, a následně by zopakoval stejný proces s celkem odpovídající náročnosti s touto aplikací. Měřeným kritériem by zde mohl být čas, za jaký se uživatel daný celek naučí. Před zahájením tohoto testování je důležité, aby se uživatel důkladně seznámil se všemi funkcemi této aplikace a mohl tak využít měřený čas výhradně pro proces memorizace.

Vyhodnocením tohoto testu by byly porovnány časy, kterých uživatel dosáhl při memorizaci celků odpovídající náročnosti. Při provádění testu by uživatel měl být pozorován, aby bylo možné analyzovat případná zlepšení nebo úskalí práce s aplikací.

# Kapitola 4

# Realizace aplikace

Tato kapitola se věnuje realizační části aplikace. V úvodu jsou charakterizovány implementační aspekty aplikace s ukázkou uživatelského rozhraní několika obrazovek aplikace. Dále je popsán datový model aplikace, datové struktury využité pro uložení skladeb a jejich metadat. Závěr kapitoly se zabývá provedenými testy, jejich zkoumanými vlastnostmi aplikace a výsledky získanými jejich vyhodnocením.

## 4.1 Popis implementace

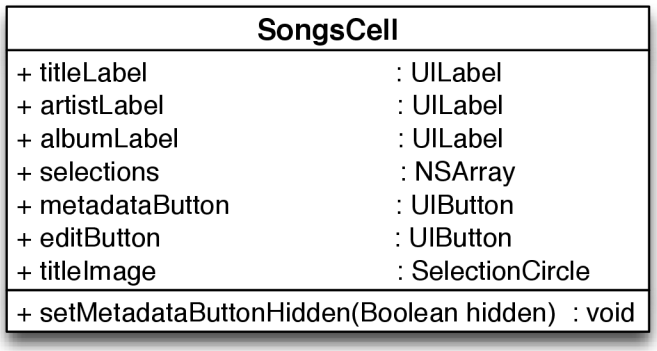
Pro implementaci aplikace byl zvolen nativní jazyk Objective-C. Hlavním důvodem pro použití Objective-C byla snažší práce s nástroji pro zpracování hudby a možnost využití podrobné programové dokumentace ke všem prvkům iOS SDK. Vzhledem k tomu, že aplikace pracuje s knihovnou iTunes, probíhal její vývoj převážně na reálném zařízení. Simulátor, který je součástí iOS SDK, totiž neposkytuje možnost pracovat s knihovnou iTunes. V rámci vývoje, který byl prováděn v prostředí Xcode, byl také využit nástroj Instruments, který je rovněž součástí iOS SDK, pro formální verifikaci správné práce se zdroji platformy iOS.

Kromě standardních prvků uživatelského rozhraní, které poskytuje framework UIKit, je implementováno několik nových prvků, které umožňují zlepšit intuitivitu a efektivitu uživatelského rozhraní aplikace. Více o podrobnostech implementace jednotlivých obrazovek aplikace bude popsáno v následujícím textu.

### 4.1.1 Hlavní obrazovka aplikace

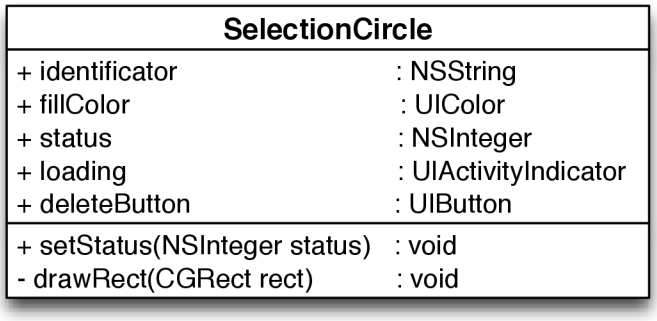
Hlavní obrazovka aplikace byla implementována jako podtřída třídy *UIViewController*, což je základní třída pro reprezentaci obsahu obrazovky frameworku UIKit. V rámci dědičnosti, kterou jazyk Objective-C poskytuje, je využito základních atributů třídy *UIViewController* s možností definovat vlastní atributy pro přizpůsobení vlastností obrazovky.

Implementace je tvořena především seznamem načtených skladeb, který prezentuje třída *UITableView*. Každému řádku seznamu odpovídá právě jedna skladba. Třída *UITableView* pracuje s třídou *UITableViewCell*, která reprezentuje obsah každého řádku. Standardně však neumožňuje zobrazit všechny komponenty a ovládací prvky, které byly v návrhu specifikovány. Proto byla vytvořena třída *SongsCell* jako podtřída třídy *UITableViewCell*, do které byly implementovány všechny potřebné atributy skladby jako například její vybrané a uložené úseky, titulní obrázek, název, album a interpret skladby nebo ovládací prvky vztahující se k dané skladbě.




Obrázek 4.1: Třída SongsCell

Textové popisky jako název, album a interpret skladby byly implementovány třídou *UILabel*. Tato třída umožňuje zobrazit daný text společně s jeho možným formátováním. Pro zobrazení vybraných úseků ke skladbě byla implementována multifunkční tlačítka. Vzhledem k požadavkům na jejich vzhled a funkcionalitu bylo vhodné využít třídy *UIControl*. Třída *UIControl* je podtřídou třídy *UIView*, která je základní zobrazovací komponentou frameworku UIKit. Kromě možnosti vykreslit požadovaný vzhled multifunkčních tlačítek v závislosti na jejich stavu poskytuje také třída *UIControl* možnost definovat obsluhu gest a dotyků. Implementace multifunkčních tlačítek třídou *třídy SelectionCircle* je tedy tvořena jako podtřída třídy *UIControl*, která je doplněna o definici jednotlivých stavů těchto tlačítek a o jejich identifikátory. Vzhledem k omezenému počtu maximálně vybraných úseků bylo zvoleno barevné odlišení, které bude patrné především při vizualizaci daných úseků na obrazovce přehrávače. Titulní obrázek skladby, který rovněž obsahuje po stisku možnost přehrát skladbu, je tvořen stejným mechanismem jako multifunkční tlačítka reprezentující výběry úseků. Zbývající ovládací prvky, které se vztahují k dané skladbě, jako přechod do přehrávače s danou skladbou nebo varovný znak absence metadat u skladby, jsou realizovány třídou *UIButton*, tedy klasickým tlačítkem. Po stisku daného tlačítka je provedena odpovídající akce, jejich funkcionalita je prezentována ikonou tlačítka.




Obrázek 4.2: Třída SelectionCircle

Navigační lištu ve frameworku UIKit standardně reprezentuje třída *UINavigationBar*. Pro účely této aplikace byla rovněž využita. Na navigační liště jsou implementovány ovládací prvky, které přímo nesouvisí s určitou skladbou. Jedná se o přechod na obrazovku výběru skladeb z knihovny iTunes a přechod do nastavení aplikace. Tuto funkcionalitu

plně poskytují tlačítka třídy *UIBarButtonItem*. Třída *UIBarButtonItem* je podtřídou třídy *UIButton*, jedná se tedy o klasické tlačítko s tím rozdílem, že je uzpůsobeno pro prezentaci na navigační liště. Funkcionalita těchto tlačítek je opět naznačena jejich ikonami. Kromě ovládacích prvků obsahuje navigační lišta také vizualizaci aktuálně přehrávané skladby, pokud je přehrávána. Třída *UINavigationBar* obsahuje atribut, který zobrazuje její titulek. Jako titulek lze zvolit libovolný objekt, který je podtřídou třídy *UIView*, nebo přímo třídu *UIView*. Mimo zobrazení informací o aktuálně přehrávané skladbě je zde také implementována reakce na stisknutí titulku navigační lišty. Pro tyto účely je opět využita podtřída třídy *UIControl*, která je doplněna o vizualizované informace a zajišťuje odpovídající obsluhu po jejím stisku.

### 4.1.2 Výběr skladeb

Pro implementaci této obrazovky byla opět vytvořena podtřída třídy *UIViewController*. Zobrazení načtených skladeb z knihovny iTunes je realizováno ve formě seznamu tedy třídy *UITableView*. Jednotlivé skladby jsou reprezentovány jako řádky seznamu. Obsah jednotlivých řádků je spravován podtřídou třídy *UITableViewCell*, ve které jsou doplněny zobrazené informace skladby. Vhodným nastavením třídy *UITableView* je provedeno zobrazení seznamu ve formě abecedních sekcí. Na základě názvu těchto sekcí je také vytvořena abecední navigace na pravé straně seznamu, která poskytuje rychlý přechod na danou sekci. Tato navigace je součástí třídy *UITableView*.

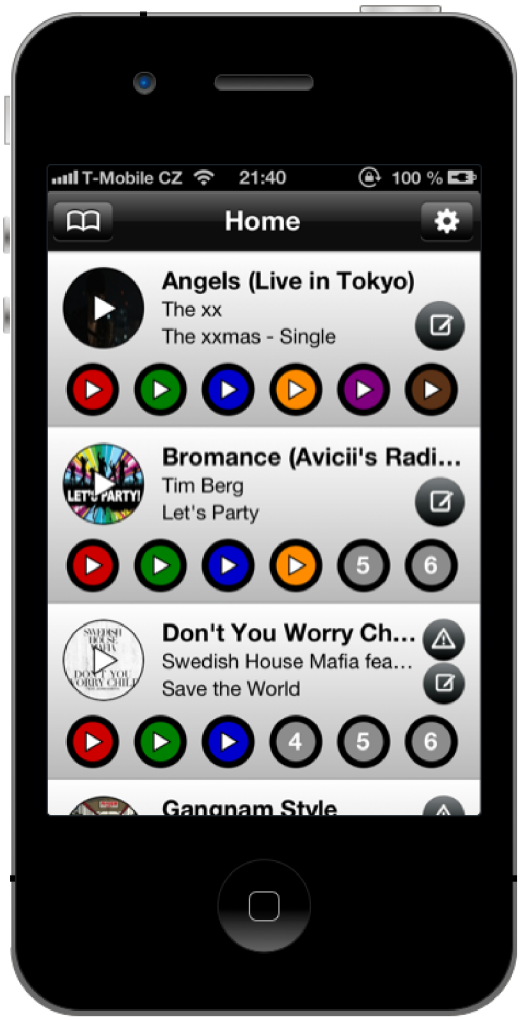
Mechanismus výběru skladeb je implementován ve formě stisku daného řádku seznamu, přičemž dojde k jeho zřetelnému podsvícení. V horní části seznamu je navržené segmentové tlačítko pro možnost změny řazení skladeb v seznamu. Framework UIKit přímo poskytuje třídu *UISegmentedControl*, která implementuje dané segmentové tlačítko s možností nastavení pouze jedné aktivní volby.

Navigační lišta, tvořena třídou *UINavigationBar*, zobrazuje ovládací prvky pro přechod zpět na hlavní obrazovku aplikace a potvrzení či zrušení aktuálního výběru skladeb. Tyto prvky byly implementovány pomocí třídy *UIBarButtonItem*, tedy tlačítka, které je přizpůsobeno pro zobrazení na navigační liště. Pokud nedošlo ke změně aktuálního výběru a aktuální výběr je shodný s předchozím, pak je na navigační liště zobrazeno pouze tlačítko pro přechod zpět na hlavní obrazovku aplikace. V případě, kdy dojde ke změně výběru, je na navigační liště zobrazena dvojice tlačítek, z nichž levé je implementováno pomocí ikony křížku a značí zrušení aktuálního výběru, a pravé je označeno ikonou zatržítka, které symbolizuje potvrzení aktuálního výběru.

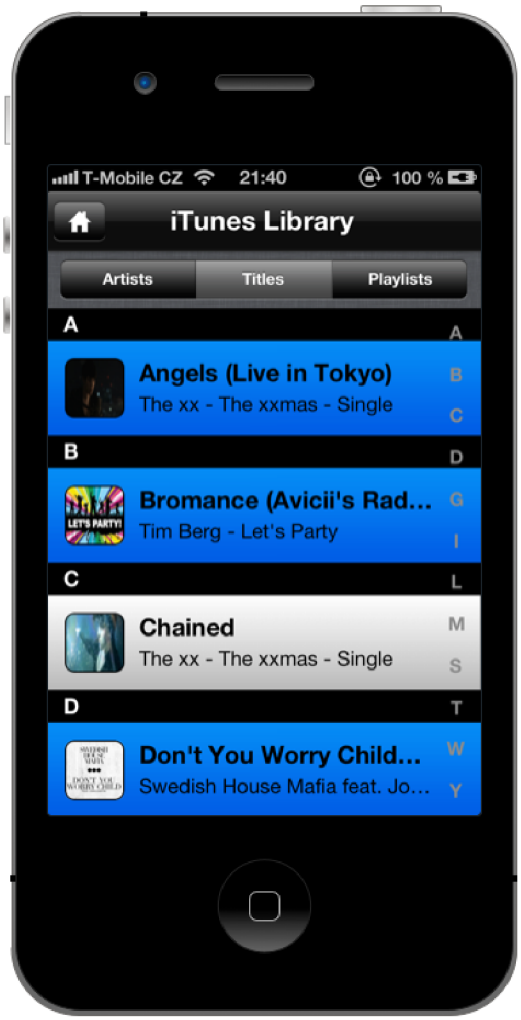
Jediným nástrojem, který na platformě iOS umožňuje přístup ke knihovně iTunes, je Media Player framework. Pro načtení skladeb z knihovny iTunes je tedy využit tento nástroj. Načtení skladeb pomocí tohoto nástroje je provedeno pomocí filtrů, které je třeba definovat. Vzhledem k zobrazení skladeb a jejich možnosti řazení, bylo načtení realizováno ve dvou krocích. V prvním kroku byly načteny všechny audio skladby a ve druhém názvy playlistů společně s identifikací jejich skladeb. Vzhledem k tomu, že je přístup do knihovny iTunes omezen pouze na čtení, je nutné vybrané skladby a jejich informace dále uložit do interní paměti aplikace. Způsob uložení těchto skladeb a jejich metadat bude dále popsán v kapitole 4.2.

### 4.1.3 Přehrávač / Editor

Obrazovka přehrávače nebo také editoru je implementována v závislosti na třídě *UIViewController*. Realizaci této obrazovky lze rozdělit do dvou částí, které jsou patrné již z fáze




Obrázek 4.3: Hlavní obrazovka aplikace

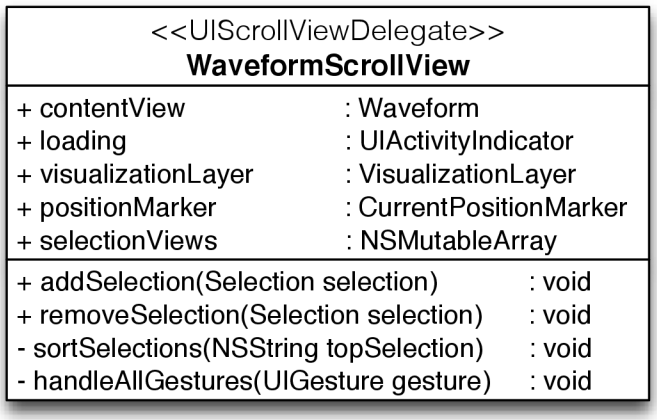



Obrázek 4.4: Obrazovka výběru skladeb

návrhu. První z nich je implementace zobrazovací oblasti, kde je vizualizována skladba společně s jejími možnými výběry, druhou částí je realizace ovládacích prvků a jejich funkcionality.

Sada nástrojů UIKit neposkytuje žádnou třídu pro efektivní implementaci zobrazovací oblasti. Je tedy nutné zvolit hierarchii tříd, které zajistí požadované vlastnosti. Jako kořenová třída byla implementována třída *WaveformScrollView*, která je podtřídou *UIScrollView*. *UIScrollView* poskytuje nejen možnost vlastního vykreslení svého obsahu, ale také zobrazit obsah větší než je velikost samotného okna a pomocí rolování jej prohlížet. Tento fakt byl využit především při implementaci možnosti přiblížit obsah zobrazovací oblasti. Zbývající prvky zobrazovací oblasti, jako například kurzor určující aktuální pozici ve skladbě nebo vizualizované výběry úseků, jsou realizovány jako podtřídy třídy *UIView*. Tyto prvky nemusí obsahovat implementaci reakcí na dotyky a gesta, protože byl zvolen způsob centrální obsluhy ve třídě *WaveformScrollView*. Centrální implementace obsluhy doteků a gest zpracovává všechny dotyky a gesta v zobrazovací oblasti a následně provede jejich odpovídající obsluhu. Výhodou tohoto způsobu je možnost delegovat doteky a gesta k odpovídajícím objektům nebo je v určitých případech ignorovat. Všechny prvky však musí být součástí

zobrazovací oblasti tedy *WaveformscrollView*.




Obrázek 4.5: Třída WaveformScrollView

Třída *UIScrollView* implicitně podporuje gesto *pinch*, neboli štípnutí, a *swipe*, neboli rychlý posun prstu. Gesto *pinch* je v implementaci *WaveformScrollView* využito pro přiblížení obsahu zobrazení, což také odpovídá konzistenci ovládání aplikací na platformě iOS. Jistým problémem je však realizace tohoto přiblížení. Implicitně je v rámci *UIScrollView* provedeno přiblížení obsahu a následné zvětšení jeho rozměrů. V návrhu aplikace je však žádoucí provést přiblížení vizualizované skladby a změnit rozměr obsahu zobrazovací oblasti pouze v horizontálním směru. V implementaci *WaveformScrollview* je tedy definována metoda, která eviduje aktuální hodnotu přiblížení a následně provede překreslení obsahu a nastavení nových odpovídajících rozměrů. Jelikož při větších úrovních přiblížení dochází k překreslování velkého množství vizualizovaných dat, při každém přiblížení vždy dojde pouze k překreslení viditelné části zobrazovací oblasti. Zbývající neviditelné části jsou překresleny po ukončení změny přiblížení. *Swipe* je ve výchozí implementaci využíván k posunu obsahu, což plně vyhovuje potřebám zobrazovací oblasti. Implementace ostatních gest a dotyků je provedena manuálně, tedy doplněním metod pro jejich obsluhu do třídy *WaveformScrollView*. Jedná se o *tap*, neboli dotyk, pro nastavení aktuální pozice kurzoru a *drag&hold*, neboli podržet a přetáhnout, pro přesun pozice kurzoru nebo změnu pozice hranic upravovaného výběru.

Pro realizaci ovládacích prvků vztahujících se k vybraným úsekům skladby byla využita multifunkční tlačítka, která byla implementována jako podtřída třídy *UIControl*, stejně jako na hlavní obrazovce aplikace. Jejich jediným rozdílem vzhledem k implementaci na hlavní obrazovce je rozšíření jejich počtu možných stavů, které jsou využity v režimu editoru. Po výběru daného tlačítka v režimu editoru byla implementována změna ikony tlačítka a jeho podsvícení pomocí vykresleného gradientu, čímž je signalizován aktivní výběr umožňující úpravu úseku uloženého na dané pozici. Ovládací prvky, které umožňují nastavení pozice kurzoru s přesností danou metadaty v režimu přehrávače a nastavení hranic vybraného úseku skladby v režimu editoru, byly implementovány ve formě jednoduchých tlačítek tedy třídy *UIButton*. Jejich funkcionalita je prezentována opět ve formě ikony.

Změna režimu obrazovky je realizována pomocí třídy *UISwitch*. Tento ovládací prvek funguje jako přepínač, umožňuje prezentovat dvě volby, z nichž právě jedna je aktivní. Ovladač hlasitosti implementuje třída *UISlider*, která umožnuje prezentovat hodnoty v daném

rozmezí. Rozmezí této komponenty je nastaveno na 11 hodnot, tedy 10 úrovní hlasitosti a možnost vypnutí zvuku. Nastavení rychlosti přehrávání je prezentováno tlačítkem, třídou *UIButton*, po jehož stisku je zobrazena podtřída třídy *UIView*, která obsahuje segmentové tlačítko třídy *UISegmentedControl*. Dané segmentové tlačítko obsahuje pět možných voleb, z nichž právě jedna je vždy aktivní. Tento návrh implementace byl diskutován podrobněji v kapitole 3.2.3.

Pro samotné přehrávání skladby byl zvolen AV Foundation framework. Tato sada nástrojů umožňuje specifikovat hlasitost a rychlost přehrávání, nastavení opakování daného úseku. AV Foundation je rovněž využit pro přehrávání skladby na hlavní obrazovce aplikace. Jeho hlavní výhodou je efektivní a snadné načtení audio dat skladby z interní paměti aplikace.

Implementace této obrazovky v režimu přehrávače je zobrazena na obrázku 4.6 a v režimu editoru na obrázku 4.7.


Obrázek 4.6: Přehrávač / Editor - režim přehrávače


Obrázek 4.7: Přehrávač / Editor - režim editoru

### 4.1.4 Nastavení aplikace

Nastavení aplikace rovněž implementuje třídu *UIViewController*. Implementace jednotlivých funkcí této obrazovky je provedena formou seznamu, tedy třídy *UITableView*. Seznam je rozdělen do dvou sekcí, z nichž první obsahuje funkce související s metadaty skladby a druhá prezentuje doplňující funkce aplikace.

Obsah řádku seznamu je standardně definován třídou *UITableViewCell*. V případě zobrazení volby pro nastavení URL adresy repozitáře je však nutné vytvořit podtřídu třídy *UITableViewCell*. Tato podtřída obsahuje kromě zděděných atributů také třídu *UITextField*, která poskytuje upravovatelné textové pole. Zbývající řádky seznamu obsahují pouze textový popisek, pro jehož zobrazení dostačuje třída *UITableViewCell*.

Po stisknutí řádku, který obsahuje textové pole pro zadání URL adresy repozitáře metadat, je zobrazena klávesnice, která umožní změnit hodnotu tohoto pole. Klávesnici je možné skrýt stiskem tlačítka s ikonou křížku třídy *UIBarButtonItem*, které bylo přidáno na navigační lištu, nebo libovolným dotykem mimo daný řádek seznamu. Zpřístupnění ostatních funkcí, jako například nápovědy aplikace či kontaktu na vývojáře aplikace, je ve druhé sekci rovněž prezentováno stiskem daného řádku.

V horní části obrazovky se nachází navigační lišta, kterou poskytuje třída *UINavigationBar*. Na navigační liště je umístěno tlačítko třídy *UIBarButtonItem* pro přechod zpět na hlavní obrazovku aplikace a v případě viditelnosti klávesnice tlačítko pro její skrytí.

Implementaci nastavení aplikace zachycuje obrázek 4.8.


Obrázek 4.8: Nastavení aplikace

## 4.2 Datový model

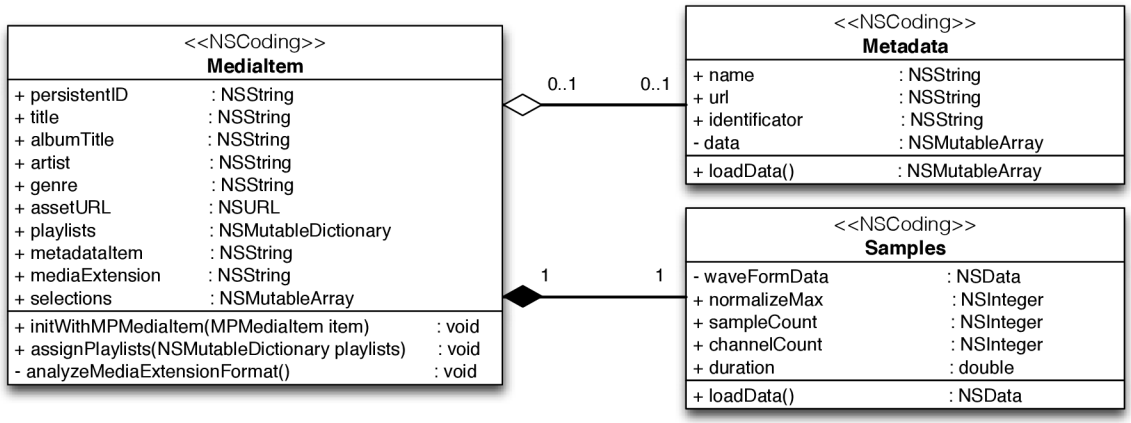
Pro efektivní manipulaci s informacemi o skladbách a jejich metadaty byl implementován datový model, jehož schéma je naznačeno na obrázku 4.9.

Základem tohoto datového modelu je třída *MediaItem*. Tato třída je rozšířením třídy *NSObject*, tedy základní třídy jazyka Objective-C. Po načtení skladby z knihovny iTunes pomocí Media Player frameworku jsou informace o skladbě poskytnuty jako třída *MPMediaItem*. Pomocí této třídy je následně inicializována vytvořená třída *MediaItem* a jsou zkopírovány všechny informace o skladbě jako například identifikátory skladby v rámci knihovny, alba nebo autora, název, album a interpret skladby, žánr skladby a další. Kromě informací o skladbě je zde také uložen odkaz k fyzickému umístění audio dat skladby, který je následně využit pro jejich načtení.

Načtení audio dat skladby je realizováno za pomocí AV Foundation frameworku. Nejdříve je provedeno vytvoření třídy *AVAsset* na základě odkazu k fyzickému umístění audio dat skladby a poté je proveden export z knihovny iTunes a uložení audio dat skladby do interní paměti aplikace. V rámci exportu jsou nejdříve zjištěny parametry skladby jako například počet kanálů audia, celkový počet vzorků, délka skladby, hodnota největšího vzorku a formát audia. Následně je na základě těchto parametrů zvolen odpovídající formát exportu, který zachová všechny parametry skladby.

Pro reprezentaci a efektivní manipulaci s audio daty skladby byla implementována třída *Samples*. Tato třída je opět vytvořena na základě třídy *NSObject*. V této třídě jsou uloženy všechny parametry skladby, na základě kterých byl proveden její export. Mimo tyto parametry třída *Samples* také obsahuje normalizované hodnoty vzorků audia, které jsou především využívány pro vizualizaci skladby. V rámci normalizace bylo u skladeb s více audio kanály provedeno vytvoření průměru odpovídajících hodnot a jejich transformace do intervalu, ve kterém budou vykresleny.

Posledním blokem datového modelu aplikace je třída *Metadata*. Jak již je patrné z jejího názvu, vztahuje se k metadatům skladby. Reprezentuje tedy načtená metadata, jejich název, URL adresu, ze které byla načtena, a identifikátor. Načtená metadata jsou zde uložena již ve formě, ve které budou vizualizována, což také poskytuje snažší manipulaci. Tato třída je opět vytvořena pouze na základě třídy *NSObject*, protože nejsou třeba žádné další vlastnosti kromě reprezentace uložených dat.




Obrázek 4.9: Třídy datového modelu aplikace

Pro uložení a persistenci těchto datových tříd byl zvolen mechanismus archivace třídou *NSKeyedArchiver*. Proces archivace provede uložení daných tříd do interní složky aplikace jako datových souborů. Jejich zpětné načtení je realizováno třídou *NSKeyedUnarchiver*, která umožňuje zpětné vytvoření uložených objektů. V rámci archivace všechny výše zmíněné datové třídy implementují protokol *NSCoding*, jehož metody definují způsob uložení a načtení dat.

Tyto datové třídy jsou využívány napříč celou aplikací. Pro jednoznačnou identifikaci informací o skladbě byl zvolen číselný identifikátor, který poskytuje knihovna iTunes a je persistentní pro danou skladbu. Pomocí tohoto identifikátoru jsou také uložena audio data skladby společně s třídou *Samples*. Ke třídě *Metadata* lze přistupovat pomocí jejího unikátního identifikátoru, který je uložen rovněž ve třídě *MediaItem*. V rámci blokového schématu aplikace na obrázku 3.1 reprezentuje výše zmíněný datový model aplikace entitu úložiště skladeb a metadat.

## 4.3 Testování

Za účelem komplexního otestování aplikace byly provedeny dva způsoby testování. První z nich byl zaměřen především na uživatelské rozhraní aplikace a druhý na ověření vlastností aplikace z hlediska výuky a memorizace. Popis obou způsobů testování byl již navržen v kapitole 3.4, v následujícím textu bude popsáno jejich provedení, zpracování a analýza naměřených hodnot.

### 4.3.1 Test uživatelského rozhraní

V rámci otestování uživatelského rozhraní byla vytvořena sada 10 úkolů specifických pro práci s aplikací. Na základě těchto úkolů byly provedeny dva testy, které se odlišovaly specifikací zadání úkolů. Účelem těchto testů bylo především zjistit, zda uživatel bude schopen s aplikací pracovat a bude jí naveden ke splnění úkolů, jinak řečeno, zda je uživatelské rozhraní aplikace intuitivní. Pro další popis nazývejme test obsahující abstraktní zadání úkolů bez specifikace postupu k jejich splnění testem A a test obsahující stručný návod k provedení úkolů testem B.

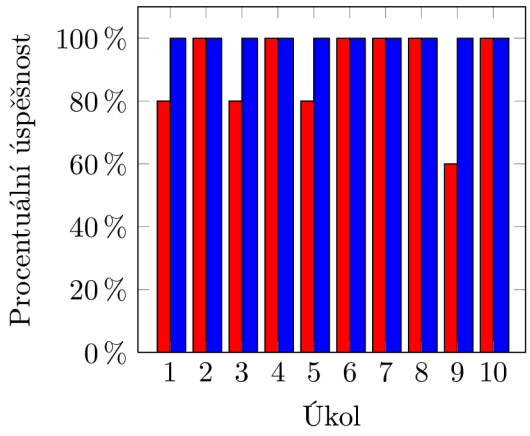
Testování bylo rozděleno do tří podskupin, které měly za úkol ověřit výsledky při provedení testů v různém pořadí. Celkově se tohoto testování zúčastnilo 14 uživatelů, z nichž 5 provedlo oba testy v pořadí A a následně B, 5 uživatelů v pořadí B a následně A a zbývající 4 provedli pouze jeden z testů, tedy 2 uživatelé pouze test A a 2 uživatelé pouze test B. Měřenými veličinami zde byl čas, za který uživatelé provedli daný úkol, a počet dotyků na obrazovce zařízení. Testování bylo provedeno za přítomnosti pozorovatele, který zaznamenával naměřené hodnoty a zajistil před začátkem testování odpovídající a pro všechny uživatele shodný stav zařízení, který umožnil provést všechny úkoly bez nutnosti jejich návaznosti.

V grafech na obrázcích 4.10 a 4.11 jsou zobrazeny naměřené hodnoty při plnění úkolů testu A a následně testu B a naopak. V obou pořadích testů je patrné znatelné zlepšení v úspěšnosti plnění úkolů při vypracování druhého testu, který byl do značné míry ovlivněn faktorem učení.

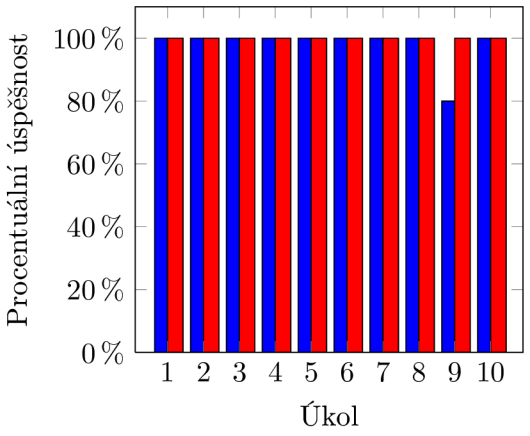
Pokud byl nejdříve proveden test A, který uživatele méně naváděl ke splnění daných úkolů, byly v prvním testu splněny všechny úkoly u více než 80% uživatelů. Výjimku zde tvořil úkol č. 8, který požadoval po uživateli odstranění uloženého výběru skladby. Tento

úkol splnilo pouze 60% uživatelů v prvním testu, ve druhém však již byla úspěšnost 100%, a to zejména díky zmínce v zadání úkolu, že mechanismus odstranění vybraného úseku je shodný s mechanismem odstranění aplikací na platformě iOS.

Při opačném provedení pořadí testů, tedy testu B a následně A, byla naprostá většina úkolů splněna všemi uživateli napoprvé. Následné provedení druhého testu již jen potvrdilo 100% úspěšnost při vypracování úkolů. Na základě procentuální úspěšnosti naměřené při provedených testech lze usoudit, že uživatelské rozhraní aplikace bylo uživateli pochopeno velmi rychle. Počáteční nižší procento úspěšnosti úkolů při provedení testů A a následně B odpovídá zkušenostem uživatelů s cílovou platformou.




Obrázek 4.10: Úspěšnost splnění úkolů při provedení testu A ( ) a následně B ( )



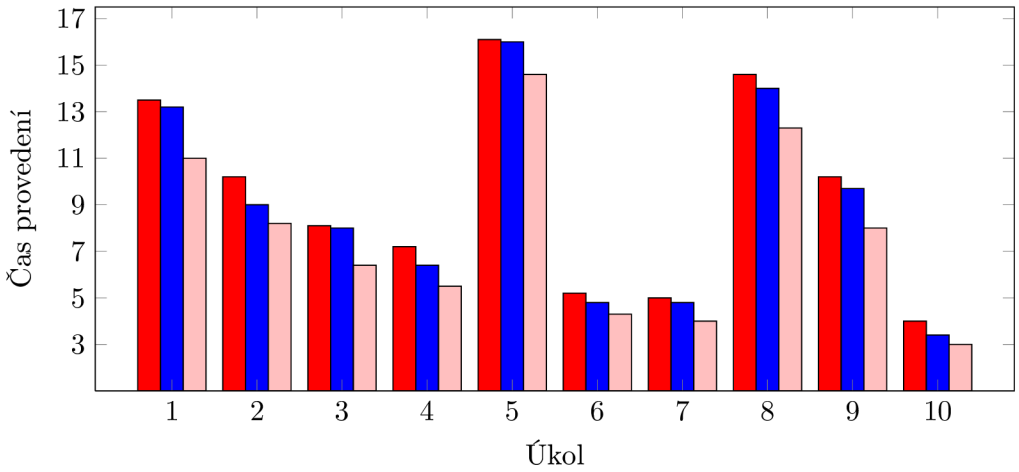

Obrázek 4.11: Úspěšnost splnění úkolů při provedení testu B ( ) a následně A ( )

Naměřené časy při provedení úkolů testu A a následně B byly při testu B u většiny úkolů nižší. U úkolů, kde byla v druhém testu naměřena vyšší úspěšnost provedení, byly naměřené časy téměř identické s výsledky v prvním testu. Tento fakt je dán pomalejším avšak úspěšným provedením daných úkolů pomocí více specifického zadání. Naměřené hodnoty odpovídají náročnosti jednotlivých úkolů. U úkolů č. 2, 3, 4, 6 a 7 je patrné zlepšení časů i přes velmi podobnou náročnost těchto úkolů. Zlepšení reflektuje seznámení uživatele s aplikací a jeho rychlé zorientování. Úkoly vyžadující řádově vyšší počet dotyků k jejich splnění jako například úkoly č. 1, 5 a 8, tedy výběr skladeb z knihovny iTunes, přidání nového výběru skladby a nastavení přesné pozice kurzoru, ukazují odpovídající časové navýšení.

Při opačném pořadí provedení testů byla většina úkolů při prvním testu provedena pomaleji, ve druhém testu bylo dosaženo výraznějšího zlepšení než tomu bylo u opačného pořadí. Toto zlepšení o průměrných 12% pramení z již téměř neměnící se procentuální hodnoty úspěšnosti provedení daných úkolů a také vyšší úspěšnosti naměřené při prvním z testů. Rovněž zde lze pozorovat zlepšení mezi prvním a druhým testem, které bylo zjištěno u opačného provedení testů.

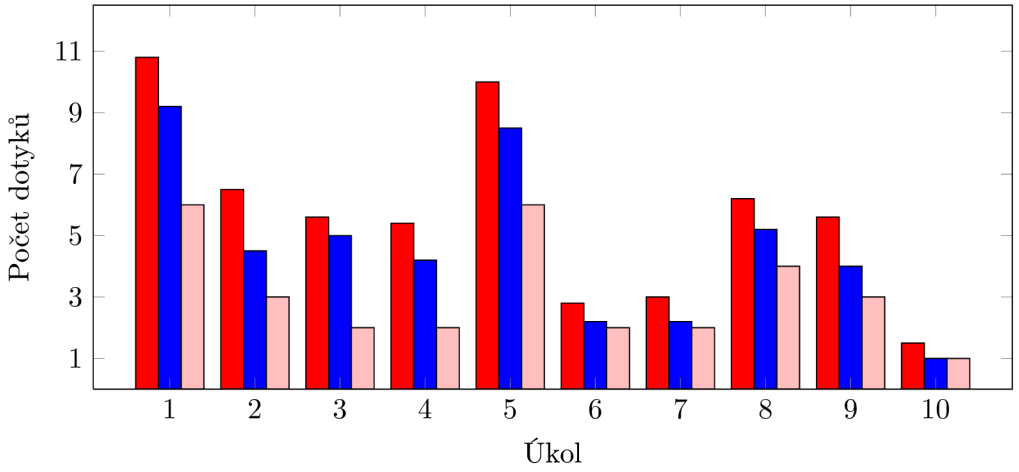
Naměřené hodnoty časů při provedení testu A a následně B jsou znázorněny v grafu na obrázku 4.12. Získané výsledky byly také porovnány s experimentálně zjištěnými časy, které lze považovat za ideální, protože jsem je stanovil při zkoušce úkolů testu. Zjištěné výsledky testů korelují s ideálními hodnotami, které vykazují hodnoty o průměrných 20% nižší než u testu A při provedení testů v pořadí B a následně A. Tato odchylka od ideálních

hodnot ukazuje, že uživatelé byli schopni s aplikací pracovat téměř jako vývojář aplikace i přes jejich výrazně nižší zkušenosti s ovládáním aplikací na platformě iOS.




Obrázek 4.12: Čas provedení úkolů při testu A ( ) a následně B ( ), ideální čas provedení úkolů ( )

Počet dotyků na obrazovce při plnění úkolů testu A a následně B společně s ideálním počtem dotyků je zobrazen v grafu na obrázku 4.13. V případě, kdy byl test A proveden jako první, ukazují naměřené hodnoty markantnější snížení počtu dotyků při druhém testu než v opačném případě. Test B poskytuje podrobnější návod ke splnění úkolu, je tedy zřejmé, že počet dotyků na obrazovce při provedení tohoto testu jako prvního bude nižší, což naměřené hodnoty potvrdily. Při provedení testu B jako prvního a následně testu A je pak následné snížení počtu dotyků téměř poloviční než při provedení testů v opačném pořadí. Výraznější snížení počtu dotyků potvrzuje návrh a implementaci konzistentního uživatelského rozhraní aplikace.

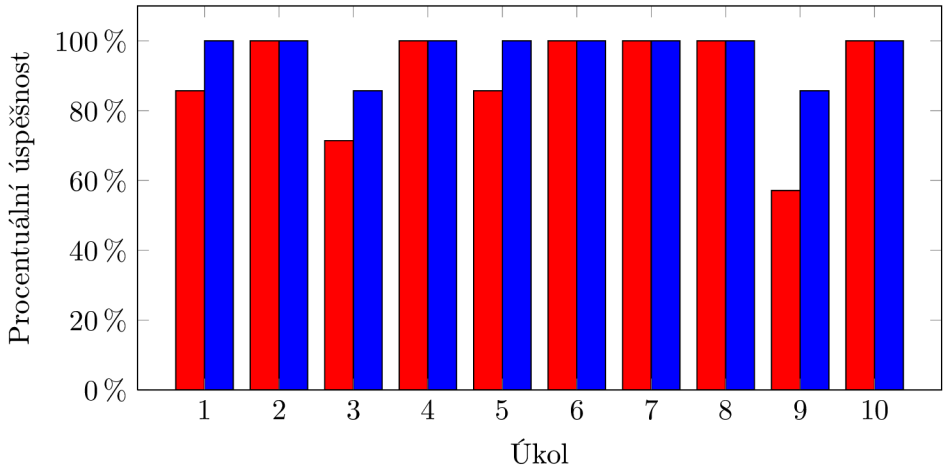



Obrázek 4.13: Počet dotyků na obrazovce při provedení úkolů testu A ( ) a následně B ( ), ideální počet dotyků ( )

Stejně jako tomu bylo u naměřených hodnot časů i zde je patrné zlepšení při provedení úkolů odpovídající náročnosti. Počty dotyků rovněž korelují s ideálními hodnotami. Zjištěná odchylka je ale téměř dvojnásobná oproti odchylce u naměřených hodnot časů. Velikost této odchylky je do značné míry ovlivněna možností provést některý úkol více způsoby, tedy s různým počtem dotyků. Důležitým prvkem je však úspěšnost při splnění úkolů, která ukazuje, že aplikace uživatele navedla ke splnění úkolů jinou než ideální cestou.

Kromě srovnání provedení obou testů v různých pořadích lze také srovnat provedení samostatných testů. Relevantní srovnání lze získat od skupiny uživatelů, kteří provedli pouze jeden z testů, nebo v případě, kdy byl daný test proveden jako první, tedy jeho výsledek není ovlivněn některým z předcházejících testů.

Z hlediska úspěšnosti provedení úkolů testu A a testu B bylo při testu B dosaženo vyšší úspěšnosti u úkolů zaměřených na výběr skladeb z knihovny iTunes, načtení metadat ke skladbě, přidání nového výběru a odstranění uloženého výběru. I přes nižší úspěšnost splnění úkolů testu B, byly všechny úkoly splněny více než 70% uživateli. Nejnižší procento úspěšnosti bylo zaznamenáno u úkolu odstranit uložený úsek a to necelých 64%. Z rozhovoru s uživateli bylo následně zjištěno, že mechanismus odstranění, který byl v aplikaci implementován, není v běžných aplikacích příliš častý, přestože je využíván na platformě iOS pro odstranění aplikací. Úspěšnost provedení úkolů testu A a testu B zachycuje graf na obrázku 4.14.




Obrázek 4.14: Úspěšnost splnění úkolů při provedení testu A ( ) a testu B( )

Intuitivní splnění časově náročnějších úkolů u testu A vede k dosažení lepších časů než u testu B za použití přesnější specifikace zadání úkolů. Je tomu tak především u úkolu vybrat skladby z knihovny iTunes, přidat nový úsek skladby nebo odstranit uložený úsek skladby. U ostatních úkolů v obou testech byly naměřeny téměř shodné časy.

Naměřené hodnoty počtu dotyků vykazují opačný charakter, než tomu bylo u naměřených časů. Časově náročnější úkoly byly provedeny při testu A s větším počtem dotyků než u testu B. Ostatní úkoly vykazují téměř shodné hodnoty.

Provedené testy ukazují, že navržené uživatelské rozhraní aplikace umožňuje uživatelům s aplikací pracovat i přes různé úrovně zkušeností s ovládáním aplikací na cílové platformě. Kromě podobnosti výsledků testů s odlišnou úrovní specifikace zadání úkolů byly také naměřené hodnoty u většiny úkolů blízké ideálním hodnotám. Vícenásobné provedení testovacích úkolů ukázalo znatelné zlepšení při práci s aplikací, které lze přisuzovat konzistentnímu

a intuitivnímu uživatelskému rozhraní aplikace.

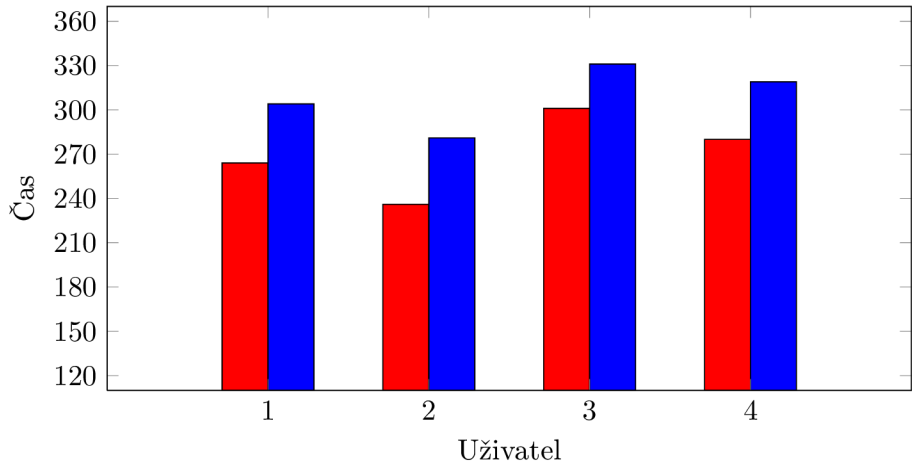
### 4.3.2 Test memorizace

Tento test byl proveden především pro ověření vlastností aplikace z hlediska výuky a memorizace. Aplikace byla testována na 4 uživatelích, z nichž 3 aktivně působí jako hudebníci. Pro účely tohoto testu byla zvolena skladba skupiny *Turbo - To bude pánové jízda* a to především proto, že byla k této skladbě zjištěna metadata a žádný z uživatelů neznal úplný text této skladby.

Před zahájením testu byli uživatelé seznámeni s ovládáním aplikace. Měřeným kritériem zde byl čas, za jaký se uživatel danou část skladby naučil. Poté byla zvolena první část skladby pro memorizaci s aplikací a druhá část skladby odpovídající délky a náročnosti pro memorizaci pomocí běžného přehrávače dostupného v zařízení iPhone. Testování bylo provedeno za přítomnosti pozorovatele. Naměřené hodnoty časů zachycuje graf na obrázku 4.15.

Z naměřených hodnot je patrné, že za pomocí aplikace byl proces memorizace rychlejší. Proces byl urychlen především díky usnadnění navigace ve skladbě a možnosti uložení vybraného úseku skladby. Uživatelé byli schopni daný text skladby reprodukovat v průměru za 309 vteřin bez použití aplikace a s využitím této aplikace za 270 vteřin, což reprezentuje zvýšení efektivity procesu memorizace o více než 14%. Nutným faktem, který je třeba vzít v úvahu, však zde jsou preference uživatelů z hlediska skladeb a jejich žánru a také optimálního nastavení aplikace. Nejlepšího zlepšení času dosáhl uživatel, který aktivně působí především v žánru memorizované skladby a při začátku skladby provedl nastavení aplikace tak, že její využití bylo maximální.

S velkou pravděpodobností by testovaní uživatelé dosáhli podobných výsledků i při memorizaci hudby skladby. Bylo by však nutné, aby všichni byli hudebníky a měli potřebnou úroveň zručnosti s daným nástrojem. Především proto byla aplikace otestována na memorizaci textu skladby.




Obrázek 4.15: Časy naměřené při memorizaci skladby pomocí aplikace (▭) a pomocí běžného přehrávače (▭)

# Kapitola 5

# Závěr

Úkolem této práce bylo navrhnout a realizovat uživatelské rozhraní mobilní aplikace pro zařízení iPhone, která pracuje s metadaty k hudební skladbě a umožňuje přehrávat skladbu po taktech nebo celých částech. Cílem práce bylo za pomocí aplikace usnadnit proces memorizace pro hudebníky a ty, kteří dávají přednost učení formou poslechu.

V rámci vývoje bylo nutné nastudovat specifika platformy iOS a to především způsob návrhu, vývoje a také dostupných nástrojů pro práci se zvukem. Velká část práce byla věnována studiu uživatelského rozhraní a také HIG platných na cílové platformě. Pro implementaci vizualizace skladby bylo také nutné nastudovat základní principy zpracování zvuku a jeho uložení v digitalizované formě.

Při návrhu uživatelského rozhraní aplikace byl kladen důraz na jednoduchost ovládání a konzistenci ovládání s platformou iOS. Návrh byl několikrát iterativně vylepšován a následně implementován v jazyce Objective-C. Mimo návrhu uživatelského rozhraní aplikace byla také navržena struktura metadat pro efektivní výuku a možné způsoby jejich získání. Výsledná implementace byla otestována na reálném zařízení pomocí skupiny 18 uživatelů. Testy byly zaměřeny na uživatelské rozhraní aplikace a proces memorizace skladby. Naměřené výsledky ukazují, že uživatelské rozhraní aplikace splňuje požadavky stanovené v návrhu a že proces memorizace s danou skupinou uživatelů byl usnadněn.

V rámci návrhu aplikace byla snaha o automatizaci procesu získání metadat ke skladbám. Mezi možná vylepšení aplikace do budoucna tedy lze zařadit automatickou detekci metadat přímo v aplikaci bez nutnosti využití repozitáře metadat na vzdáleném serveru. Lepším řešením by zde možná bylo implementovat uživatelské rozhraní systému pro správu a tvorbu metadat na serveru, především kvůli možnosti manuální korekce získaných výsledků pomocí automatické detekce. Serverové řešení správy metadat by také do budoucna mohlo usnadnit případnou optimalizaci metadat a jejich rozšíření o uživatelem definovaná metadata pro efektivnější výuku.

# Literatura


[1] Apple Inc.: Core Audio Overview [online]. Dostupné z: https://developer.apple.com/library/mac/documentation/MusicAudio/Conceptual/CoreAudioOverview/CoreAudioOverview.pdf, 2008 [cit. 2013-05-01].

[2] Apple Inc.: Cocoa Fundamentals Guide [online]. Dostupné z: http://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/CocoaFundamentals/CocoaFundamentals.pdf, 2010 [cit. 2013-05-01].

[3] Apple Inc.: AV Foundation Programming Guide [online]. Dostupné z: http://developer.apple.com/library/ios/DOCUMENTATION/AudioVideo/Conceptual/AVFoundationPG/AVFoundationPG.pdf, 2011 [cit. 2013-05-01].

[4] Apple Inc.: Concepts in Objective-C Programming [online]. Dostupné z: http://developer.apple.com/library/ios/documentation/general/conceptual/CocoaEncyclopedia/CocoaEncyclopedia.pdf, 2012 [cit. 2013-05-01].

[5] Apple Inc.: iOS Technology Overview [online]. Dostupné z: http://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechOverview.pdf, 2012 [cit. 2013-05-01].

[6] Apple Inc.: Media Player Framework Reference [online]. Dostupné z: http://developer.apple.com/library/ios/documentation/MediaPlayer/Reference/MediaPlayer_Framework/MediaPlayer_Framework.pdf, 2012 [cit. 2013-05-01].

[7] Apple Inc.: Programming with Objective-C [online]. Dostupné z: https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/ProgrammingWithObjectiveC.pdf, 2012 [cit. 2013-05-01].

[8] Apple Inc.: Developer Tools Features [online]. Dostupné z: https://developer.apple.com/technologies/tools/features.html, 2013 [cit. 2013-05-01].

[9] Apple Inc.: iOS Human Interface Guidelines [online]. Dostupné z: http://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/MobileHIG.pdf, 2013 [cit. 2013-05-01].

[10] Ballou, G.: *Handbook for sound engineers*. Focal Press, 2008.

[11] Blesser, B. A.: Digitization of audio: a comprehensive examination of theory, implementation, and current practice. *Journal of the Audio Engineering Society*, 1978, 26.10: 739–771.

[12] Jagadale, B.: Audio Signal Processing Using Wavelet Transform. *Journal of Computer and Mathematical Sciences Vol*, 2012, 3.6: 557–663.

[13] Lavry, D.: Sampling Theory For Digital Audio. *Lavry Engineering, Inc.*, 2004, Dostupné z: http://www.lavryengineering.com/pdfs/lavry-sampling-theory.pdf.

[14] Li, K.; Wu, M.: *Effective GUI testing automation: Developing an automated GUI testing tool*. Sybex, 2006, ISBN 978-0782150674.

[15] Memon, A. M.: GUI testing: Pitfalls and process. *IEEE Computer*, 2002, 35.8: 87–88.

[16] Memon, A. M.; Pollack, M. E.; Soffa, M. L.: Hierarchical GUI test case generation using automated planning. *Software Engineering, IEEE Transactions on*, 2001, 27.2: 144–155.

[17] Raskin, J.: *The Humane Interface: New Directions for Designing Interactive Systems*. Addison-Wesley, 2000, ISBN 978-0201379372.

[18] Shneiderman, B.; Plaisant, C.: *Designing the User Interface: Strategies for Effective Human-Computer Interaction (5th Edition)*. Prentice Hall, 2009, ISBN 978-0321537355.

# Příloha A

# Obsah CD

- Plakát prezentující implementovanou aplikaci, kterou popisuje tato práce.
- Video prezentující implementovanou aplikaci, její funkce a způsob ovládání.
- Zdrojový kód implementované aplikace ve formě zdrojových souborů a projektu v prostředí Xcode. Kromě zdrojových dat je také přiložen balík aplikace ve formátu ipa, který lze prostřednictvím aplikace iTunes nahrát do zařízení iPhone. Způsob instalace aplikace je popsán v přiloženém manuálu (Manual.pdf).
- Zdrojové soubory této technické zprávy pro LaTeXvčetně souboru makefile pro jejich přeložení. Součástí je také elektronická verze této práce ve formátu pdf.
- Zadání testů, které byly provedeny v rámci otestování uživatelského rozhraní aplikace.
- Zadání testu, který byl proveden pro otestování aplikace z hlediska memorizace.