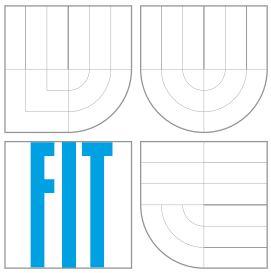
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MOBILNÍ APLIKACE PRO SDÍLENÍ SEZNAMŮ SKLADEB V KAPELE

MOBILE APP FOR SHARING PLAYLISTS IN A BAND

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JIŘÍ SKOTÁK

VEDOUCÍ PRÁCE
SUPERVISOR

prof. Ing. ADAM HEROUT, Ph.D.

BRNO 2016

Zadání bakalářské práce/18276/2015/xskota07

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Skoták Jiří**
Obor: Informační technologie
Téma: **Mobilní aplikace pro sdílení seznamů skladeb v kapele**
Mobile App for Sharing Playlists in a Band
Kategorie: Zpracování obrazu

Pokyny:

1. Specifikujte a analyzujte potřeby indie kapely při sestavování playlistů.
2. Navrhněte dílčí prvky uživatelského rozhraní pro správu playlistů a vyhodnoťte jejich vhodnost / přínos.
3. Navrhněte několik variant uživatelského rozhraní minimalistické verze aplikace pro správu playlistů kapely.
4. Implementujte vitální navržené varianty.
5. Na základě reakcí uživatelů a statistik používání vylepšujte navrženou aplikaci směrem k dokonalosti.
6. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- dle pokynů vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3, značné rozpracování bodů 4 a 5.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Herout Adam, doc. Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Cílem práce je popsat vývoj aplikace pro sdílení seznamů skladeb v kapele. Aplikace je vyvíjena na platformu iOS v jazyce Objective-C. Implementace aplikace je řešena pomocí softwarové architektury MVC. Výsledkem je uživatelsky přívětivá aplikace, kterou lze využít pro správu seznamů skladeb kapel. Hlavním přínosem této práce je shrnutí doporučení pro platformu iOS, shrnutí zásad pro tvorbu uživatelsky přívětivého prostředí a seznámení se způsobem, jakým kapely spravují své seznamy skladeb.

Abstract

This bachelor thesis describes development of an application for sharing playlists in band. The application is developed for iOS in Objective-C language. The application is implemented using MVC software architectural pattern. The result is user-friendly application, suitable for managing band's playlists. The main feature of this thesis is summary of recommendations for iOS platform, summary of principles for creating user-friendly application and introduction of band's playlists management.

Klíčová slova

iPhone, iOS, Sdílení seznamů skladeb, Objective-C, Uživatelské rozhraní, Mobilní aplikace, MVC

Keywords

iPhone, iOS, Playlists sharing, Objective-C, User interface, Mobile application, MVC

Citace

SKOTÁK, Jiří. *Mobilní aplikace pro sdílení seznamů skladeb v kapele*. Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Herout Adam.

Mobilní aplikace pro sdílení seznamů skladeb v kapele

Prohlášení

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

.....
Jiří Skoták
16. května 2016

Poděkování

Děkuji prof. Ing. Adamu Heroutovi, Ph.D. za odborné vedení práce a cenné rady při řešení návrhu aplikace. Dále bych chtěl poděkovat své přítelkyni, sestře a všem ostatním, kteří mě při psaní práce podpořili.

Obsah

Kapitola 1

Úvod

Na světě je dnes tisíce aktivních kapel. Každá z těchto kapel má svůj repertoár písní, ze kterých sestavuje playlisty na koncert nebo zkoušku. Sepisování těchto playlistů na papír nebo případně do textových dokumentů na počítači je často zdlouhavé. Obnáší neustále kopírování předchozích, podobných playlistů, vyhledávání písniček v jiných dokumentech apod. Pro poskytnutí playlistu ostatním členům, je potřeba ho vytisknout nebo odeslat pomocí jiné aplikace. A při každé změně playlistu znovu. Na tento problém mě upozornil můj kamarád, který ve svém volném čase hraje se třemi kapelami a velmi by ocenil jednoduchou mobilní aplikaci, která mu umožní spravovat tyto playlisty přímo v mobilu. Nejlépe bez nutnosti připojení k internetu, na zkoušce nebo ve vlaku. Jakožto studenta informačních technologií mě napadlo řešení, které jsme spolu následně ladili a vedlo ke zhotovení aplikace.

Cílem mé práce je implementovat aplikaci, která uživateli umožní ukládat písně se stěžejními informacemi a následně z těchto písní vytvářet playlisty. Tyto playlisty bude uživatel moci sdílet s ostatními. Aplikace bude často používána během koncertu či uprostřed zkoušky s telefonem na koleni. Proto je důležité jednoduché uživatelské rozhraní aplikace, ve kterém se každý vyzná. Rozhodl jsem se nejprve aplikaci psát pro operační systém iOS, který je dostupný pro telefony iPhone. Značná část muzikantů ke své tvorbě používá zařízení firmy Apple a proto je právě iPhone vhodným primárním cílovým zařízením. Aplikace uživateli umožní vytvářet a spravovat playlisty bez osobního počítače, pouze na mobilním telefonu. Ten má dnes většina lidí neustále u sebe.

Práce je členěna do několika kapitol, které popisují jednotlivé části vývoje. Kapitola 2 je věnována teoretickým znalostem, které jsou nezbytné pro začátek vývoje aplikace. Kapitola 3 se zabývá informacemi získaných od potenciálních uživatelů a analýzou již existujících řešení. V kapitole 4 jsou rozebrány jednotlivé případy užití aplikace a představen návrh uživatelského rozhraní. V kapitole 5 je pak rozebrána implementace některých částí aplikace a na závěr je v kapitole 6 rozebrán způsob a výsledky testování aplikace.

Kapitola 2

Platforma iOS

Tato kapitola se zabývá teoretickými znalostmi nutnými pro implementaci a návrh výsledné aplikace. Je dělena do pěti částí.

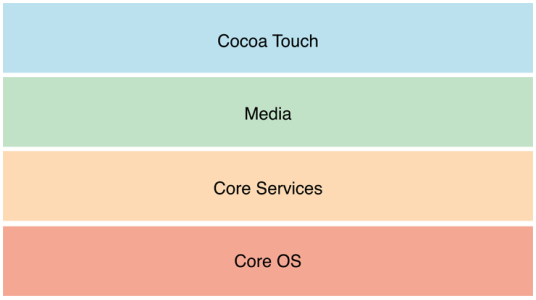
2.1 Úvod do iOS

Jelikož je aplikace implementována pro mobilní platformu iOS, je vhodné ji stručně představit a popsat princip vývoje pro tento mobilní systém.

Mobilní operační systém iOS je od roku 2010 vytvořen a vyvíjen společností Apple Inc. a distribuován výhradně pro Apple hardware. Jedná se o odlehčenou verzi operačního systému Mac OS X, používaného v počítačích Macintosh. Systém iOS představuje pomyslného prostředníka pro komunikaci mezi hardware a aplikacemi. První verze tohoto systému byla společně s prvním revolučním telefonem značky iPhone představena v roce 2007. Tato verze byla postupně rozšiřována pro podporu ostatních zařízení značky Apple jako iPod Touch (září 2007), iPad (leden 2010) a iPad Mini (listopad 2012). Nové verze systému jsou vydávány v pravidelných intervalech. V době psaní této práce je aktuální verzí iOS 9.

2.1.1 Architektura iOS

Operační systém iOS se dělí do čtyř vrstev, které lze vidět na obrázku 2.1. Jednotlivé vrstvy poskytují API a frameworky nutné pro vývoj aplikací. Vyšší vrstvy poskytují sofistikovanější služby než vrstvy nižší. Z hlediska psaní kódu je doporučeno využívat frameworky vyšších vrstev, kdykoliv je to možné.



Obrázek 2.1: Vrstvy iOS (zdroj Apple.com)

Vrstva Cocoa touch

Tato nejvyšší vrstva obsahuje klíčové frameworky pro vývoj iOS aplikací. Tyto frameworky definují vzhled vyvíjené aplikace. Dále poskytují základní aplikační infrastrukturu a podporu pro důležité technologie jako rozpoznávání gest, multitasking, push notifikace a mnoho dalších vysokoúrovňových systémových služeb.

Vrstva Media layer

Tato vrstva umožňuje vytváření graficky a zvukově propracovaných aplikací. Tyto technologie umožňují plynulé přehrávání animací, videí a zvuků.

Vrstva Core Services

Tato vrstva obsahuje základní systémové služby pro aplikace. Klíčovými službami jsou Core Foundation a Foundation frameworky, které definují základní typy, které aplikace využívají. Tato vrstva taktéž zahrnuje technologie, které podporují funkce jako alokace paměti, iCloud a networking.

Vrstva Core OS

Tato vrstva obsahuje nízkoúrovňové funkce, na kterých jsou postaveny ostatní technologie. I přes to, že tyto technologie nejsou v aplikacích používány přímo, jsou s velkou pravděpodobností využívány ostatními vysokoúrovňovými komponenty systému [1].

2.2 Vývoj na platformu iOS

Pro pohodlný vývoj aplikace pro iOS je k dispozici iOS SDK[1]. Jedná se o sadu vývojových nástrojů vydanou v únoru 2008 firmou Apple Inc. Tyto nástroje jsou zdarma, ale spustitelné pouze na operačním systému Mac OS. Základ tvoří vývojové prostředí Xcode, které slouží mimo jiné jako editor zdrojových souborů. Xcode obsahuje uživatelsky přívětivý Interface Builder, díky kterému lze jednotlivé prvky aplikace graficky rozmístit a následně je jednoduše propojit s odpovídajícím kódem pro jejich obsluhu. Součástí Xcode je i iPhone Simulator, který simuluje vzhled a chování zařízení iPhone přímo na vývojářově počítači. Během vývoje tedy není nutno aplikaci neustále nahrávat do zařízení iPhone. Tento simulátor bohužel nedokáže napodobit některou funkčnost reálného zařízení jako například fotoaparát, odesílaní e-mailů nebo některá multidotyková gesta. Z tohoto důvodu je velmi dobré mít při vývoji přítomné cílové zařízení firmy Apple, které je stále podporované nejnovější verzí iOS, a aplikaci průběžně testovat i na něm. Pro možnost testování na reálném zařízení bylo dříve nutno stát se oficiálním vývojářem aplikací pro iOS. K tomu je potřeba certifikát vývojáře, který je možné zakoupit za 99$ na rok. To se ovšem změnilo v září 2015 s vydáním nové verze Xcode 7.0 [7]. Od této verze Xcode je možno testovat své aplikace na fyzických zařízeních iPhone a iPad bez nutnosti vlastnictví certifikátu vývojáře. Pro uvedení aplikace na App Store[2] je vlastnictví takového certifikátu stále nutností.

[1] Software Development Kit

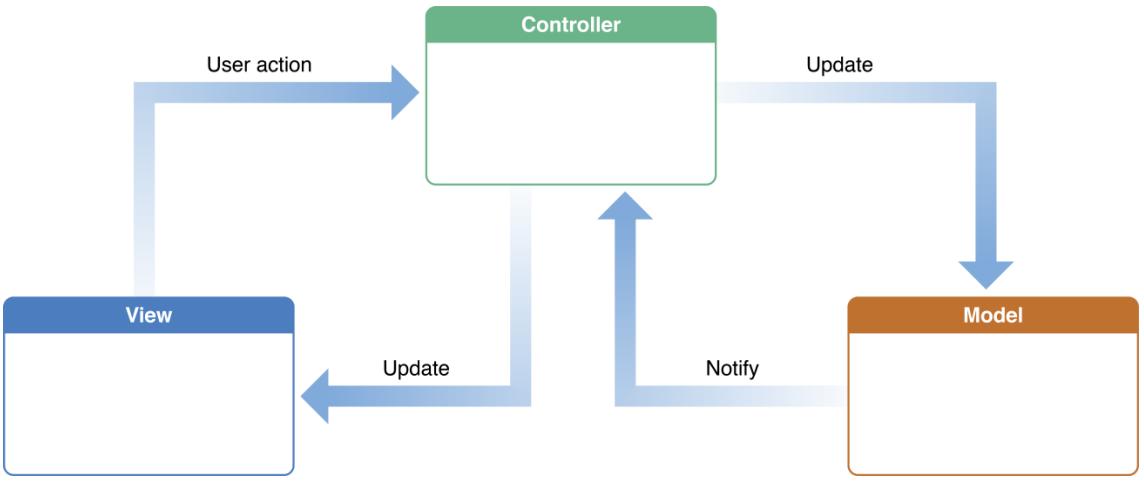
[2] obchod s aplikacemi pro zařízení s operačním systémem iOS a Mac OS X

2.2.1 Jazyky Objective-C a Swift

Pro vývoj aplikací pro iOS lze využít jazyk Objective-C, nebo Swift. Vývojově starší jazyk Objective-C byl představen v roce 1983 jako jazyk pro operační systém NeXTSTEP, z něhož postupně vznikl OS X a iOS. Jedná se o objektově orientovaný jazyk, postavený na jazyku C, do kterého byl přidán systém zasílání zpráv z jazyka Smalltalk. Jazyk Swift je z velké části obdobou jazyka Objective-C. Vývoj na tomto jazyku započal v roce 2010 a představen byl 2. června 2014. Při jeho představení byl zjednodušeně označen jako „Objective-C bez C“ [9]. V jazyce Swift byl nahrazen Smalltalkový způsob volání metod za tečkovou notaci a jmenné prostory, což je běžné pro jazyky jako je Java nebo C#. Jazyk Swift zachovává klíčové vlastnosti jazyka Objective-C za použití jednodušší syntaxe. Měl by tak být flexibilnější a zamezit více chybám programátora, než starší jazyk Objective-C.

2.2.2 Softwarová architektura MVC

Pro správný vývoj aplikace na platformu iOS je důležité využít softwarovou architekturu MVC[3]. Ta 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. Aplikace je z pravidla složena z několika vzájemně komunikujících skupin MVC. Princip komunikace mezi jednotlivými částmi MVC lze vidět na obrázku 2.2.



Obrázek 2.2: Pokud uživatel provede nějakou akci, je o ní z View odeslána informace do Controlleru. Controller pak dle potřeby upraví datový Model, který ho o provedených změnách může zpětně uvědomit. Pokud došlo činností uživatele ke změně stavu aplikace, Controller tento stav uživateli zobrazí pomocí aktualizace View [2].

Model

Model obsahuje data aplikace. Není nijak spojen s částí View, takže v ní můžeme jednoduše provádět změny a stále pracovat se stejnými daty.

[3] Model–View–Controller

View

View definuje způsob zobrazení dat uživateli. Tvoří vzhled a ovládací prvky, se kterými může uživatel komunikovat.

Controller

Controller tvoří mezivrstvu mezi vrstvami předchozími. Je propojen s View a na základě událostí vyvolaných uživatelem provádí změny ve View nebo Modelu [6].

2.2.3 Oficiální schválení aplikace

Při vývoji aplikace je dobré myslet na to, že před uvedením aplikace na App Store musí aplikace získat oficiální potvrzení od společnosti Apple. Ta má velmi specifický pohled na to, které aplikace schválí a které ne. Tento pohled konkretizuje souborem pravidel, jejichž nedodržení vede k zamítnutí aplikace a nemožnosti její oficiální uvedení na trh [4]. Výčet oficiálních pravidel není konečný a s rostoucím množstvím aplikací roste zároveň i množství pravidel. Apple pravidelně každý týden vydává nejčastějších 10 důvodů vedoucích k neschválení aplikace. Při vývoji jsem se tedy snažil těmto obvyklým nedostatkům dopředu předcházet.

2.2.4 Podporovaná zařízení

S vydáváním nových verzí systému iOS jsou kladeny stále vyšší nároky na hardware jednotlivých zařízení. Firma Apple sama rozhoduje, které modely jejich zařízení dané nároky splňují a bude jim umožněno svůj systém aktualizovat. Aplikace jsou pak vyvíjeny s ohledem na zařízení podporující nejnovější systémovou verzi. Vývojář si přesto musí uvědomit, že zařízení podporující nejnovější systém nemusí být schopno splnit všechny jeho nejnovější funkce. Podle toho je pak příhodné implementovat svou aplikaci tak, aby pokud možno podporovala co nejvíc mezi lidmi stále používaných zařízení. Nynější verze systému iOS 9, která byla vydána v září 2015, stále podporuje iPhone 4S, který byl na trh uveden v říjnu 2011, a iPad 2 z března 2011 [4].

2.3 Doporučení pro tvorbu GUI na platformu iOS

Mezi hlavní vlastnosti kvalitní mobilní aplikace patří její vzhled, přehlednost a intuitivnost ovládání. Firma Apple Inc. tyto kvality naplňuje velmi dobře a je v jejím zájmu, aby vývojáři pro jejich platformy nečinili jinak. Aby vývojáře navedla správnou cestou, zveřejnila příručku iOS Human Interface Guidelines [5], která vývojáři představuje základní pravidla a doporučení pro vývoj uživatelského rozhraní pro platformu iOS. Pro vývoj mé aplikace jsem si poznamenal následující doporučení.

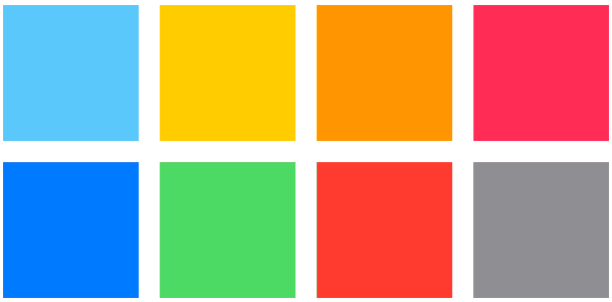
2.3.1 Layout

Je důležité, aby vzhled aplikace odpovídal její funkčnosti. V případě aplikace pro správu playlistů tedy není účelné přidávat mnoho matoucích prvků. Funkcí této aplikace je práci usnadnit a zbytečně věci nekomplikovat. Důležité je také co nejvíce využívat celou plochu obrazovky. Nevyužité prostory působí nekompletním a prázdným dojmem. Prvky je dobré rozmisťovat s vědomím rozdílných velkostí obrazovek jednotlivých modelů iPhone. Taktéž

je vhodné do aplikace zakomponovat prvky typické pro platformu iOS. Ovládací prvky by tedy měly splňovat určité zaběhnuté normy a vzory, na které jsou uživatelé iOS aplikací obecně zvyklí. To uživatelům usnadní orientaci v aplikaci a ta pak působí mnohem lépe a přirozeněji.

2.3.2 Barvy

Mezi důležité prvky designu patří taktéž font a barva textu. V iOS barvy indikují interaktivitu, dodávají vitalitu vzhledu a poskytují vizuální návaznost. Vestavěné aplikace používají kombinace z rodiny čistých barev, které vypadají dobře jak samostatně, tak vzájemné kombinaci na světlém i tmavém pozadí 2.3.



Obrázek 2.3: Typické barvy pro oficiální aplikace platformy iOS. (zdroj Apple.com)

Při vytváření vlastních barev je potřeba dbát na to, aby se k sobě zvolené barvy hodily. Proto je při zvolení určité barvy pro většinu designu aplikace příhodné vytvořit několik dalších barev, které se v rámci aplikace budou v kombinaci používat. Při volení těchto je dobré zohlednit, jak jednotlivé barvy působí. Barvy totiž každý člověk vnímá jinak a různé kultury barvám přiřazují různý význam. Aplikace by taktéž měla myslet na barvoslepost a vyhnout se kombinacím barev, které se barvoslepým lidem špatně rozeznávají. Zvolené barvy je pak žádoucí používat tak, aby nebylo porušeno pravidlo kontrastu.

Pravidlo kontrastu popisuje jak správně kombinovat barvy, aby od sebe byly jednotlivé elementy snadno rozeznatelné. Například při špatně zvoleném kontrastu barev mezi tlačítkem a pozadím, může tlačítko s pozadím splynout a uživatel jej lehce přehlédne. Pro orientační ověření vhodného kontrastu mezi zvolenými barvami je možné si danou aplikaci na zařízení vyzkoušet v různém světelném prostředí, zejména venku za slunečného dne. Pro odbornější ověření dobře zvoleného kontrastu je třeba ověřit poměr mezi svítivosti popředí a pozadí. Tento poměr lze určit pomocí online kalkulátoru kontrastu nebo lze spočítat manuálně dle vzorce, uvedených ve standardu WCAG 2.0[4].

Dále je ze zvoleného souboru barev doporučeno vybrat barvy pro interaktivní a neinteraktivní prvky aplikace. Ostatní barvy by neměly být podobné barvě, která naznačuje možnost interakce. V žádném případě pak barva určená pro interaktivní prvky nesmí být použita u prvků neinteraktivních.

2.3.3 Textový font

Aplikace s uživatelem komunikuje pomocí textu a ikon. Aby komunikace pomocí textu nebyla nepřehledná, je třeba zvolit správnou typografii. Firma Apple u každé verze mobilního systému iOS poskytuje systémový font, který je vhodné používat z několika důvodů.

[4]Web Content Accessibility Guidelines verze 2.0

- Font je používán v rámci vestavěných aplikací a uživateli tak přijde přirozený.
- Při změně velikosti textu v nastavení zařízení lze tento font měnit automaticky i uvnitř aplikace.
- Firma Apple systémový font volí tak, aby korespondoval s celkovým pocitem z designu aktuální verze iOS.

Pro iOS 9 je systémovým fontem San Francisco. Tento font je možno používat v několika velikostech, které automaticky poskytují skvělou čitelnost. U různých velkostí lze taktéž jednoduše specifikovat mezery mezi znaky a výšky řádku. Lze taktéž specifikovat styl textu pro sémanticky odlišné bloky. San Francisco disponuje dvěma optickými velikostmi: Text a Display. Text je používán pro velikosti menší než 20 bodů a Display pro větší než 20 bodů. Při použití tohoto fontu, systém automaticky vhodně zvolí, zda-li použít Text nebo Display. Uvnitř aplikace je taktéž možno vytvořit a používat svůj vlastní font. Ať už je font zvolen jakýkoliv, je dobré uvnitř aplikace využívat pouze jeden. Při použití více fontů může aplikace působit nedbale jako na obrázku 2.4.

(a) Doporučený vzhled obrazovky s využitím jediného fontu.

(b) Nedoporučený vzhled obrazovky s využitím více fontů.

Obrázek 2.4: Obrázek znázorňující porovnání použití jednoho a více fontů. (zdroj Apple.com)

2.4 Prostředky pro sdílení playlistů

Aby uživatel mé aplikace mohl pohodlně konzultovat své vytvořené playlisty s ostatními členy kapely, bylo nutné do aplikace zakomponovat určitou formu jejich sdílení. Jednou z možností bylo implementovat server, který by spravoval data pro jednotlivé skupiny uživatelů. Tato možnost ovšem přináší mnoho komplikací spojených jak s implementací, tak se správou serveru. Další možností bylo využít API nabízené webovým úložištěm Dropbox[5].

[5]Dropbox je webové úložiště, které umožňuje uživateli ukládat a sdílet soubory a složky s ostatními uživateli internetu pomocí synchronizace souborů.

Pak by se data mezi uživateli dala sdílet pomocí Dropboxem poskytovaných sdílených složek. Tento přístup by ovšem od všech uživatelů aplikace požadoval vlastnictví Dropbox účtu. Registrace je sice snadná a zdarma, přesto by s velkou pravděpodobností odradila nejednoho potenciálního uživatele. Pro sdílení playlistů jsem se tedy nakonec, s podporou získaných informací uvedených v kapitole 3.1.2, rozhodl použít klasický e-mail. Uživatel pak jednoduše odešle svůj vytvořený playlist ostatním uživatelům skrz klasický e-mail a ti ho poté na svém zařízení v aplikaci otevřou. V takto odeslaném e-mailu pak bude prostor i pro přílohu v podobě tisknutelné verze odeslaného playlistu, která se bude hodit členům kapely bez iPhonu, nebo iPadu. V následujících podkapitolách popíšu jednotlivé třídy využité pro implementaci tohoto přístupu pro sdílení.

2.4.1 Třída MFMailComposeViewController pro emailové rozhraní

Tato třída poskytuje základní rozhraní pro vytváření, editování a odesílaní emailových zpráv. Toto rozhraní lze použít uvnitř libovolné aplikace. Před zobrazením rozhraní je vhodné naplnit několik hodnot pro předmět, adresáty, text emailu a přílohy. Po zobrazení rozhraní uživatel uvidí takto předpřipravené hodnoty, které před samotným odesláním emailu pak sám ještě může upravit.

Před zobrazením rozhraní je nutno ještě ověřit, zda-li je dané zařízení, na kterém běží aplikace, schopno odesílat emailové zprávy. Pokud není, je dobré o tom informovat uživatele, nebo jednoduše uvnitř aplikace zakázat možnost vytváření a odesílaní emailů. Aplikace by se zajisté neměla pokoušet zobrazit rozhraní, pokud email nedokáže odeslat.

Samotné rozhraní nezaručuje odeslání e-mailové zprávy. Díky němu je možné zprávu pouze předpřipravit. Uživatel pak sám rozhodne, zda-li bude zpráva odeslána. V případě, že se po zobrazení rozhodne uživatel zprávu neodeslat, je její obsah smazán a zahozen. Pokud se uživatel rozhodne e-mail odeslat, zpráva je zařazena do fronty pošty k odeslání v aplikaci pro správu emailů, která přebírá zodpovědnost za doručení této zprávy.

2.4.2 Třída UIDocumentInteractionController pro správu dokumentů

Tato třída pro správu dokumentů poskytuje podporu uvnitř aplikace pro práci uživatele se soubory uvnitř lokálního systému zařízení. Je využívána pro zvolení vhodného rozhraní pro zobrazení, otevíraní, kopírování nebo tisknutí určitého souboru.

Tato třída je využívána například při zobrazování příloh e-mailů a jejich otevírání ve vhodných aplikacích. V případě, že se uživatel rozhodne otevřít přiložený soubor e-mailu v některé z jeho aplikací, třída na základě hodnoty property[6] s názvem `UTI` 2.4.3 rozhodne, které dostupné aplikace jsou pro tento typ souboru vhodné. Pokud je tento soubor zvláštního typu, pro který nelze naleznout vhodnou aplikaci k jeho otevření, property `UTI` nabývá hodnoty *nil*.

2.4.3 Uniform Type Identifier

Uniform type identifier (dále UTI) je řetězec, který unikátně definuje třídu entit, u kterých se předpokládá, že mají nějaký „typ“. Například pro soubor může „typ“ označovat formát dat. Běžně tak UTI poskytuje konzistentní identifikátor pro data, který rozeznají jednotlivé aplikace a služby. Pro příklad soubor `JPEG`[7] se může identifikovat následujícími způsoby:

[6] třídní proměnná uvnitř jazyka Objective-C

[7] standardní metoda ztrátové komprese používané pro ukládání počítačových obrázků ve fotorealistické kvalitě

- Soubor typu `OSType`[8] s pojmenováním `'JPEG'`
- Soubor s příponou `.jpg`
- Soubor s příponou `.jpeg`
- Typ MIME[9] ve formě `image/jpeg`

UTI nahrazuje tyto způsoby identifikace jednotným textovým řetězcem `public.jpeg`. Tento řetězec je pak kompatibilní se všemi předešlými způsoby identifikace. Zároveň je vždy možné z UTI vytvořit ekvivalentní typ ve formě `OSType`, MIME, nebo souboru s příponou[3].

Další výhodou UTI je možnost definovat vlastní UTI speciálně pro jeho využití v aplikaci. Například pokud aplikace využívá vlastní formát dokumentu, lze pro něj definovat UTI. Pokud by další aplikace chtěly používat stejný formát dokumentu, jednoduše pak můžou pomocí tohoto UTI rozpoznat odpovídající soubor a nabídnout se jako vhodné pro jeho zpracování.

Znaková sada UTI

UTI je tvořen z podmnožiny znakové sady ASCII. Pro UTI lze použít velké a malé písmena římské abecedy (A-Z, a-z), číslice (0-9), tečku (".") a pomlčku ("-"). Tato omezení jsou vytvořena na základě omezení záznamů DNS uvedených v RFC 1035.

Syntaxe UTI

UTI používá pro svou identifikaci reverzní záznam DNS. Pro příklad:

```
com.mycompany.myapp.myspecialfiletype
com.apple.pict
com.apple.quicktime-movie
```

Tato syntaxe zaručuje, že budou jednotlivé identifikátory jedinečné i bez nutnosti přítomnosti centrální správy pro jejich registraci a údržbu. Speciální doménu tvoří `public`. Ta je rezervována pro standardní a známé typy, které jsou běžně využívány většinou aplikací. UTI s public doménou momentálně může deklarovat pouze Apple. Pro příklad:

```
public.text
public.plain-text
public.jpeg
```

Doména `dyn` je rezervována pro dynamické UTI. Dynamické UTI jsou využity v případě, že aplikace narazí na neznámý typ dat, pro který není deklarován UTI. Pro takový typ je pak deklarován dynamický UTI. Dynamický identifikátor tvoří pomyslný obal okolo neznámého typu pro neznámou souborovou koncovku, neznámy typ MIME,`OSType` a jiné.

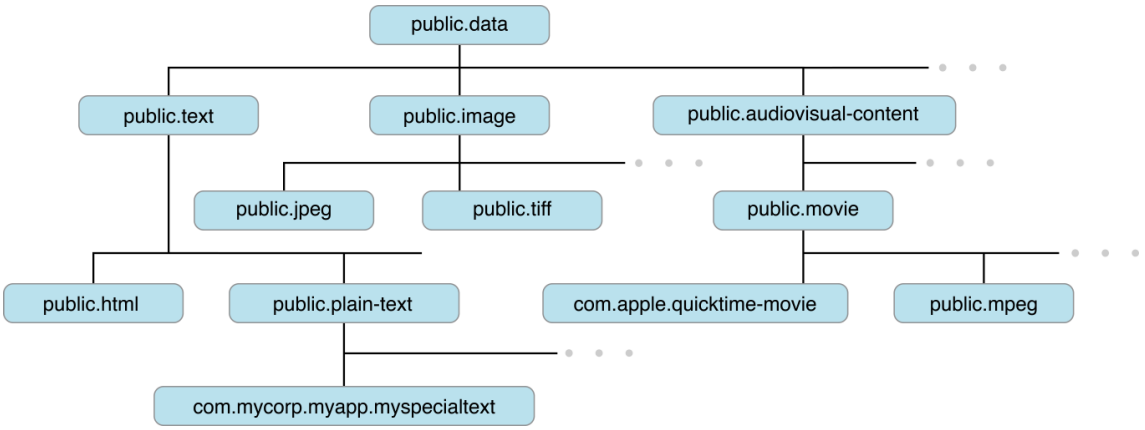
Ostatní domény jsou volně dostupné. Většina firemních UTI pak začíná doménou `com`.

[8] jméno čtyřbajtové sekvence používané jako identifikátor

[9] internetový standard, který umožňuje pomocí elektronické pošty zasílat zprávy obsahující text s diakritikou, přílohami, ...

Hierarchie UTI

Jednotlivé UTI jsou deklarovány uvnitř jisté hierarchie. Tato hierarchie je analogická s třídní hierarchií v objektově orientovaném programování. Na obrázku 2.5 můžeme vidět příklad takové hierarchie.



Obrázek 2.5: UTI `public.html`, jenž definuje text HTML, je podtřídou UTI pro základní text – `public.text`. V tomto případě je tedy možné, aby aplikace, která je schopná otevřít základní textový soubor, otevřela taktéž soubory HTML. Dědičnost může být i vícenásobná. Při deklaraci UTI s dědičností je třeba si uvědomit, že pokud bychom vytvořili UTI, která dědí od UTI `public.tiff`, dědí zároveň i od `public.image` a `public.data`. (zdroj Apple.com)

Díky této dědičnosti aplikace dokáže snadno rozeznat, s kterými typy je kompatibilní. Nemusíme tak psát mnoho podmínek pro ověření správnosti typu a navíc může být naše aplikace kompatibilní i s dosud pro nás neznámými typy.

2.4.4 Vytváření PDF

Bylo by vhodné, aby uživatel, který vytvoří svůj playlist mohl tento playlist sdílet i s členy kapely, kteří nemají iPhone nebo iPad. Do e-mailu, kromě souboru pro přenos dat mezi aplikacemi na zařízení, by bylo dobré přidat taktéž zobrazitelný soubor pro tisk.

Nejlepším formátem pro tento soubor je s velkou pravděpodobností formát PDF[10]. Pro vytvoření PDF na zařízení iPhone poskytuje UIKit framework[11] funkce pro kreslení. Tyto funkce umožňují vytvořit grafický kontext 2.4.4 pro soubor PDF. Díky těmto funkcím lze vytvořit několik stran, do kterých se pak pomocí UIKit a Core Graphics[12] kreslí stejným způsobem jako na obrazovku. Po ukončení kreslení je vytvořen PDF soubor s vykresleným obsahem. Proces kreslení lze shrnout do následovných kroků:

1. Vytvoř PDF kontext.
2. Vytvoř stránku.
3. Použij funkce Core Graphics a UIKit pro kreslení obsahu stránky.

[10] Portable Document Format – Přenosný formát dokumentů
[11] UIKit framework poskytuje základní infrastrukturu pro vytváření a správu aplikací
[12] API na základě jazyka C, které poskytuje základní funkce pro vykreslování

4. Dle potřeby opakuj kroky 2 a 4.

5. Ukonči PDF kontext a dle specifikace při jeho vytváření zapiš data do zadaného souboru PDF.

Grafický kontext

Grafický kontext je objekt obsahující informace, které vyžaduje systém kreslení. Obsahuje například informace o barvách pro kreslení, kreslícím režimu, šířky čáry a informace o fontu. Grafický kontext je vztažen k oknu, bitové mapě, PDF souboru, nebo výstupnímu zařízení a uchovává momentální stav kreslícího prostředí.

Grafický kontext PDF

Grafický kontext PDF lze vytvořit jednou ze dvou funkcí. Na základě zvolené funkce je kontext vytvořen a jeho PDF výstup směřován buď do souboru PDF, nebo do objektu typu *NSMutableData*. Dokumenty PDF organizují svůj obsah po stránkách. Aby mohlo být zahájeno kreslení je nutné, aby byla připravena alespoň jedna stránka s pevně zvolenou velikostí. Po skončení kreslení je PDF kontext uzavřen a je zavolána funkce pro uložení obsahu PDF do zvoleného objektu nebo souboru. Tato funkce zároveň smaže vytvořený grafický PDF kontext.

Kapitola 3

Analýza využití aplikace pro sdílení seznamů skladeb v kapele

Pro správný postup při návrhu a implementaci aplikace pro sdílení seznamů skladeb v kapele bylo nutné zjistit několik informací. Nejprve jsem musel zjistit, jakým způsobem kapely spravují své hrané písně a jak plánují koncerty a zkoušky. Průzkumem získané vědomosti jsem shrnul v první části této kapitoly. V druhé části této kapitoly se zabývám již existujícími aplikacemi, které řeší podobnou problematiku.

3.1 Průzkum cílové skupiny muzikantů

Na světě nyní působí tisíce kapel. Každá kapela má určitý repertoár písní, které umí zahrát. Z tohoto seznamu písní pak sestavuje své koncerty, nebo plány zkoušek. Pro vývoj mé aplikace bylo nezbytné zjistit, jakým způsobem kapely tento seznam skladeb udržují a jakým způsobem pak plánují koncerty a zkoušky. Dále mě zajímalo, jaké jsou pro muzikanty nejdůležitější informace o hraných písních vyjímaje text a noty. Text a noty jsem do aplikace prozatím neplánoval zahrnout, protože by jejich vytváření pouze pomocí iPhonu nebo iPadu nebylo uživatelsky přívětivé. Povedlo se mi spojit s pěti členy různých kapel, od nichž jsem zjistil následující informace.

3.1.1 Seznam hraných písní

Pro údržbu seznamu hraných písní kapely z pravidla využívají Microsoft Word[1] a MS Excel[2]. Správu nad hlavním seznamem písní má zpravidla jedna osoba – kapelník. Ten rozesílá jeho aktuální podobu na vyžádání jednotlivým členům. Ti si pak k písním píší osobní poznámky a dle potřeby seznam tisknou. Na koncertě či zkoušce pak někteří z dotazovaných potřebují aktuální verzi tohoto seznamu pro případ, že by chtěli zahrát původně na koncert nenaplánovanou píseň. Jeden z dotazovaných poznamenal, že tištěný seznam skladeb potřebují pouze pro posluchače. Ti si pak na základě tohoto seznamu můžou vyžádat písně na přání. Svůj osobní seznam i s informacemi ohledně jednotlivých písní totiž mají v notebooku, který si musí brát na koncerty s sebou.

Z těchto informací jsem vyvodil, že by má aplikace mohla předejít několika problémům. Například by nebylo potřeba seznam písní tisknout a riskovat jeho zapomenutí, či ztracení.

[1] textový editor od firmy Microsoft
[2] tabulkový editor od firmy Microsoft

Navíc by aktualizovanou verzi seznamu jednotliví členové mohli od správce jejich seznamu získat i bez potřeby počítače, nebo notebooku.

3.1.2 Příprava kapely na koncert či zkoušku

Kapely se svým přístupem k plánování koncertu liší. V některých kapelách celý koncert plánuje kapelník, v jiných seznam písní na koncert připravují členové společně. Někteří si celý koncert naplánují dopředu, jiní si koncert naplánují jen z části. To znamená, že si předpřipraví několik písní, které by rádi zahráli a následovně je kombinují s písněmi na přání. Jeden z dotazovaných odpověděl, že si na koncert vezme pouze tisknutý aktuální seznam všech písní a následovně o přestávkách podle nálady publika program koncertu průběžně sestavuje. V každém případě s sebou na koncert kapely většinou berou tisknutou přípravu koncertu a tisknutý seznam hraných písní.

S aplikací v mobilním telefonu by kapely nemusely na koncert nosit vytisknutý plán koncertu nebo vytisknutý seznam všech písní. V případě naplánovaného koncertu by v předpřipraveném seznamu mohly dle potřeby jednoduše provést změny a informovat o nich ostatní. Pokud by koncert vymýšlela kapela až na místě, místo papíru s tužkou by jako nástroj pro svižnou přípravu vystoupení pravděpodobně efektivněji posloužila mobilní aplikace.

Vývoj mé aplikace hodně ovlivnila informace o způsobu plánování koncertu, kde kapelník naplánuje v Microsoft Excel koncert a rozešle plán písní jednotlivým členům na e-mail. Ti si pak dle tohoto seznamu připraví noty a text. Tento proces vyžaduje minimálně u jednoho člena počítač nebo notebook, kde složitě připraví seznam písní a v mailu ho postupně rozešle jednotlivým členům. S aplikací v mobilu by stejný úkon mohl provést mnohem snadněji, rychleji a téměř odkudkoliv. Rozhodl jsem se tedy aplikaci modelovat převážně směrem k takovému stylu plánování koncertů.

3.1.3 Stěžejní informace o písni

Jak již jsem zmínil, moje mobilní aplikace neplánuje podporu vložení textů a not. Ptal jsem se tedy na další informace, které by se muzikantům k písni hodily. Kromě názvu písně a autora jsou dalšími podstatnými informacemi o písni její délka, tónina a tempo. Zajímavou informací pro mě byla tónina, která se podle jednoho z dotazovaných může změnit na základě hlasu zpěváka. Stejná píseň se pak na různých koncertech s jinými zpěváky hraje v jiné tónině. Další velmi důležitou informaci pro mě byla potřeba poznámek u jednotlivých písní.

Mobilní aplikace všechny tyto požadované informace může umožnit jednoduše ukládat, zobrazovat a upravovat. V případě změny v písni je pak jednoduší upravit ji uvnitř aplikace, než na tištěném papíře na koncertě nebo doma na notebooku. Poznámky k písni navíc může uživatel na mobil zapisovat téměř kdekoliv a kdykoliv a nemusí si je pamatovat, či složitě zaznamenávat a následně přepisovat.

3.2 Analýza existujících řešení

Existuje několik aplikací, které by měly pomáhat muzikantům s fungováním jejich kapely. Zaměřím se na jejich provedení, výhody i nevýhody.

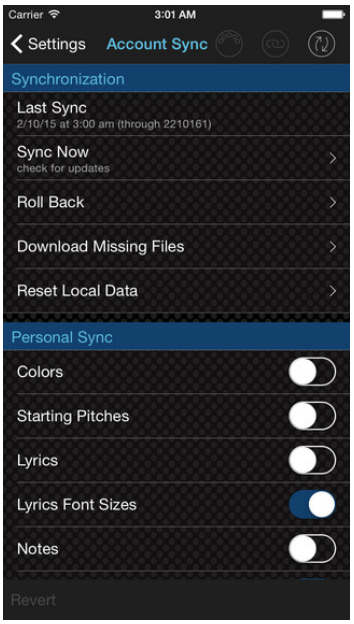
3.2.1 Band Helper

Tato aplikace se kromě vytváření playlistů z vložených písní soustředí na mnoho dalších aspektů spojených s vedením kapely. Pomocí aplikace lze plánovat koncerty do kalendáře, plánovat rozmístění nástrojů na pódiu, sledovat výdaje a příjmy spojené s vedením kapely, ukládat připomínky, videa, nahrávky a k písním lze kromě základních informací vkládat i jejich text a noty. Do aplikace je možno přidávat ostatní uživatele této aplikace a svá data s nimi společně sdílet a upravovat. Ke sdílení je implementován server, ke kterému lze přistupovat i přes webové rozhraní.

Aplikace tedy nabízí mnoho funkcí, to ovšem nemusí být vždy výhodou. Jak již jsem zmínil, mnou navrhovaná aplikace klade důraz na jednoduchost a přehlednost, aby i člověk netechnického zaměření mohl aplikaci označit za užitečnou a praktickou. O složitosti této aplikace vypovídá vzhled jejího grafického rozhraní (viz obrázek 3.1) i vývojáři nabídnuté návody na používaní této aplikace, které v podobě videa trvají dohromady přes hodinu. K používání aplikace je taktéž nutno se registrovat na internetu, což vede k dalším komplikacím, kterými si uživatel před samotným spuštěním aplikace musí projít.

Věřím, že i tato aplikace nalezne mezi muzikanty mnoho příznivců. Má aplikace by ovšem měla zapůsobit na uživatele, kteří hledají aplikaci jednoduššího rázu. Aplikaci si lze pořídit za 1$ měsíčně, za tuto cenu ovšem aplikace nenabízí veškerou zmíněnou funkcionalitu, aby mohl uživatel využívat všechny funkce, musí platit dvakrát tolik. Na internetovém obchodě iTunes má aplikace hodnocení čtyř bodů z pěti.

(a) Obrazovka s detailem písně.



(b) Detail obrazovky pro obsluhu synchronizace.

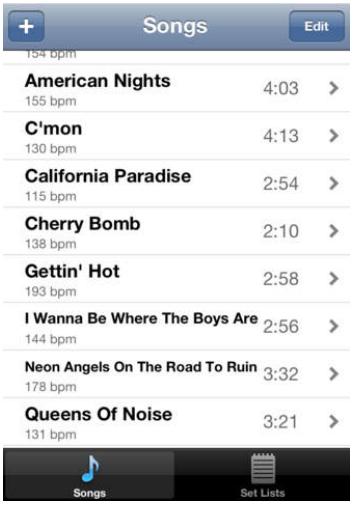
Obrázek 3.1: ScreenShoty z aplikace Band Helper.

3.2.2 Set List Keeper

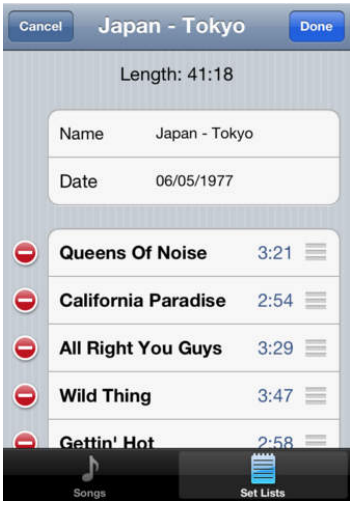
Tato aplikace umožňuje přidávat písně a ukládat jejich název, délku a tempo. Z těchto písní lze vytvořit playlist. Playlist lze pak odeslat na mail ve formě PDF.

Jedná se o velice jednoduchou aplikaci, která působí spíše jako mírně organizovaný poznámkový blok. I přes tento fakt dosahuje v průměru vyššího hodnocení než aplikace Band Helper. Oproti aplikaci Band Helper je pro mě tato aplikace naopak až příliš strohá. Mezi její hlavní nevýhody bych zařadil nemožnost přidání tóniny, jména autora a poznámek k písni a nemožnost editace písně. Navíc i přes to, že aplikace působí jednoduše, práce s ní mi nepřišla nejpohodlnější.

Tato aplikace je zdarma a na App Store sklízí chválu – hodnocení si drží na pěti bodech z pěti možných.



(a) Přehled všech písní.



(b) Obrazovka pro vytváření playlistu.

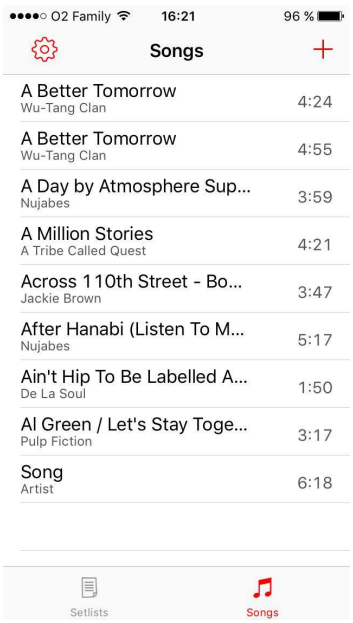
Obrázek 3.2: ScreenShoty z aplikace Set List Keeper.

3.2.3 Setlyst

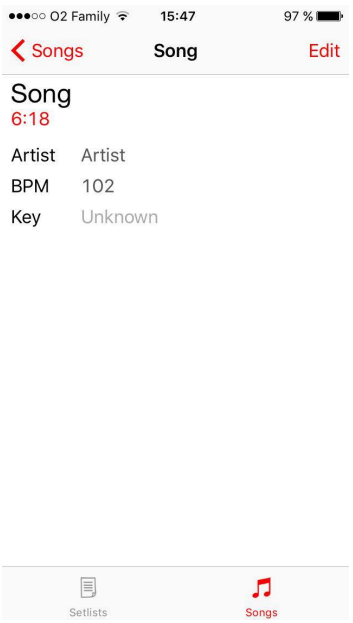
Tato aplikace se svou strukturou blíží více k mé představě než aplikace předchozí. Aplikace umožňuje přidávat písně s názvem, jménem autora, délkou, tempem a tóninou. Z těchto písní lze vytvářet playlisty, které lze pak pomocí e-mailu odeslat ve formě pdf a dalším formátu, který umožní přenos playlistu do této aplikace v jiném zařízení.

I přes to, že se mi aplikace jeví lépe než dvě předchozí, nalezl jsem v jejím provedení opět několik nedostatků. Mezi to patří například nemožnost psaní poznámek k písni, nemožnost zobrazení detailu písně uvnitř playlistu, neexistence správy kontaktů pro rychlejší rozesílaní playlistu a hlavně nemožnost měnění pořadí písni jak v playlistech, tak v seznamu všech písní. Navíc je v rámci aplikace implementováno nastavení barevného zobrazení formou RGB, které mi přijde celkově navíc a zbytečné. Aplikace mi při jejím používaní taktéž několikrát spadla a nepodařilo se mi odeslat e-mail obsahující playlist.

Tato aplikace je zdarma a na internetovém obchodě iTunes se drží na hodnocení třech bodů z pěti.



(a) Přehled všech písní.



(b) Obrazovka s detailem písně.

Obrázek 3.3: ScreenShoty z aplikace Setlyst.

Kapitola 4

Návrh výsledné aplikace

4.1 Případy užití aplikace

Pro správné určení požadovaných funkcí aplikace jsem vytvořil diagram případů užití. Na základě tohoto diagramu jsem se pak mohl rozmýšlet nad rozmístěním ovládacích prvků a layoutu jednotlivých stránek.

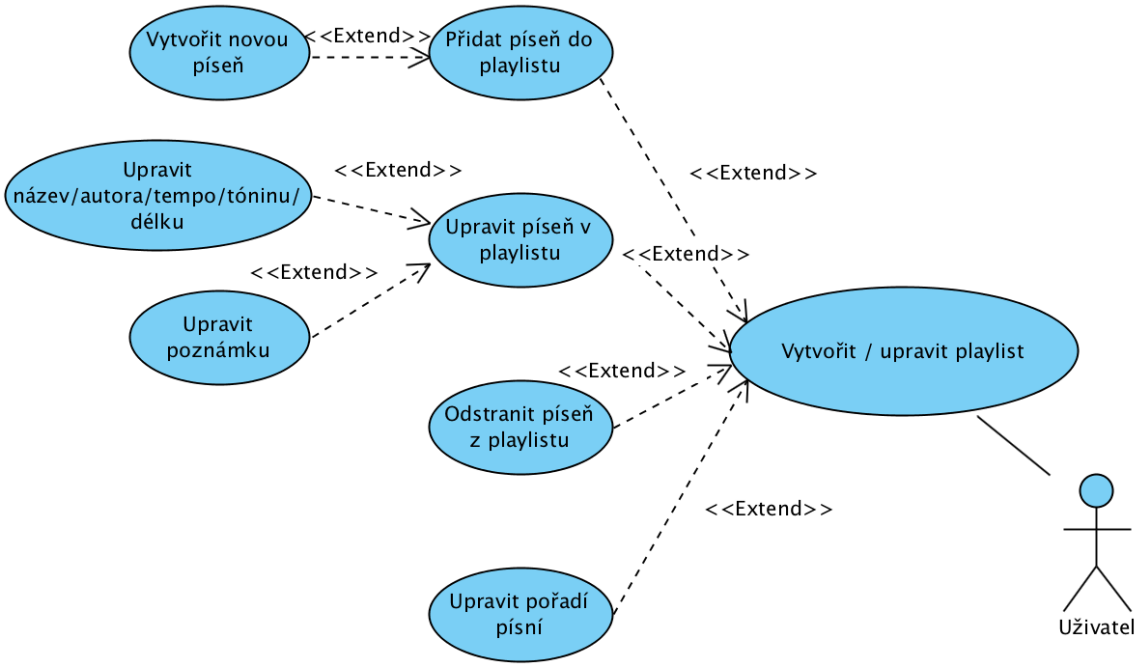
Pro diagram případů užití bylo potřeba zjistit, za jakým účelem by nejčastěji uživatel aplikaci spouštěl. Zjistil jsem, že by aplikace byla používána zejména z těchto důvodů:

- plánování/úprava playlistu na koncert/zkoušky
- sdílení playlistu s ostatními
- sledování písní playlistu v průběhu koncertu/zkoušky
- zapisování poznámek k písni
- správa souhrnu písní kapely

S těmito základními úkony souvisí mnoho dalších úkonů vedlejších. Ty bylo potřeba postupně odhalit a jejich provádění navrhnout tak, aby uživateli nekomplikovaly práci a on se mohl soustředit na svůj původní záměr. Zamýšlel jsem se tedy postupně nad jednotlivými častými užitími aplikace a došel jsem k následujícím výsledkům. Jednotlivá užití aplikace jsou popsána od nejčastějšího.

4.1.1 Plánování/úprava playlistu

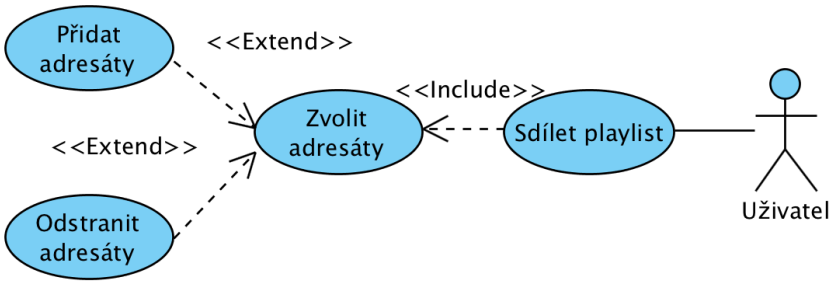
Pro plánování playlistu na koncert, nebo zkoušku se týká část use-case diagramu zobrazená na obrázku 4.1. Jedná se o činnost, kdy uživatel vytváří playlist ze svého seznamu všech písní. Jelikož si zejména ze začátku využívání aplikace uživatel může vzpomenout na píseň, kterou v seznamu všech písní nemá, mělo by mu být umožněno i v procesu vytváření/úpravy playlistu novou píseň vytvořit. V rámci playlistu lze taktéž jednotlivé písně upravovat. Například v případě, že bude na určitém koncertě tónina písně lišit od její běžné podoby, je potřeba ji v rámci playlistu upravit. Samozřejmostí je do playlistu písně přidávat, odstraňovat a měnit jejich pořadí.



Obrázek 4.1: Úkony spojené s vytvářením a úpravou playlistu. S vytvořením playlistu logicky souvisí přidávání písní. Pokud uživatel omylem přidá píseň nevhodnou, bude mu umožněno ji odstranit. Pořadí písní v playlistu by mu taktéž mělo být umožněno měnit co nejsnadněji. Dále může v rámci úpravy playlistu uživatel chtít upravit konkretní píseň nebo k písní napsat určitou poznámku.

4.1.2 Sdílení playlistu

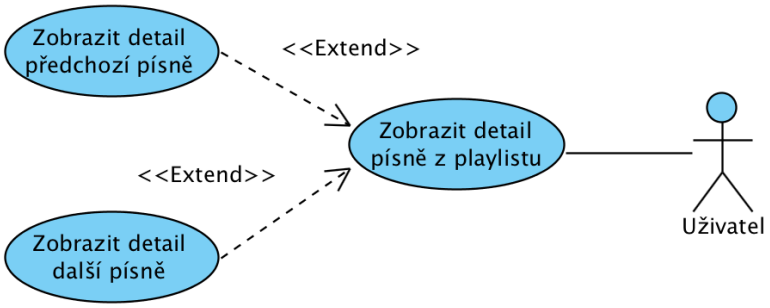
Sdílení playlistu se týká část use-case diagramu zobrazena na obrázku 4.2. Uživatel může playlist sdílet odesláním emailu se souborem pro importování playlistu do aplikace a souborem PDF pro tisk. Tento úkon bude ovšem častý a proto by se měl uživateli co nejvíce ulehčit. Místo toho, aby ze svého telefonního seznamu vyhledával jména uživatelů, kterým chce email odeslat, mělo by mu být umožněno vytvořit si vlastní adresář uvnitř aplikace, ze kterého pak snadno vybere z kontakty, se kterými své playlisty obvykle sdílí.



Obrázek 4.2: Úkony spojené se sdílením playlistu. Pokud bude chtít uživatel sdílet playlist, ve své kontaktní knížce zvolí adresáty a e-mail jim odešle. S kontaktní knížkou samozřejmě souvisí i její správa – adresáty bude uživatel přidávat i odstraňovat.

4.1.3 Sledování písní v průběhu koncertu/zkoušky

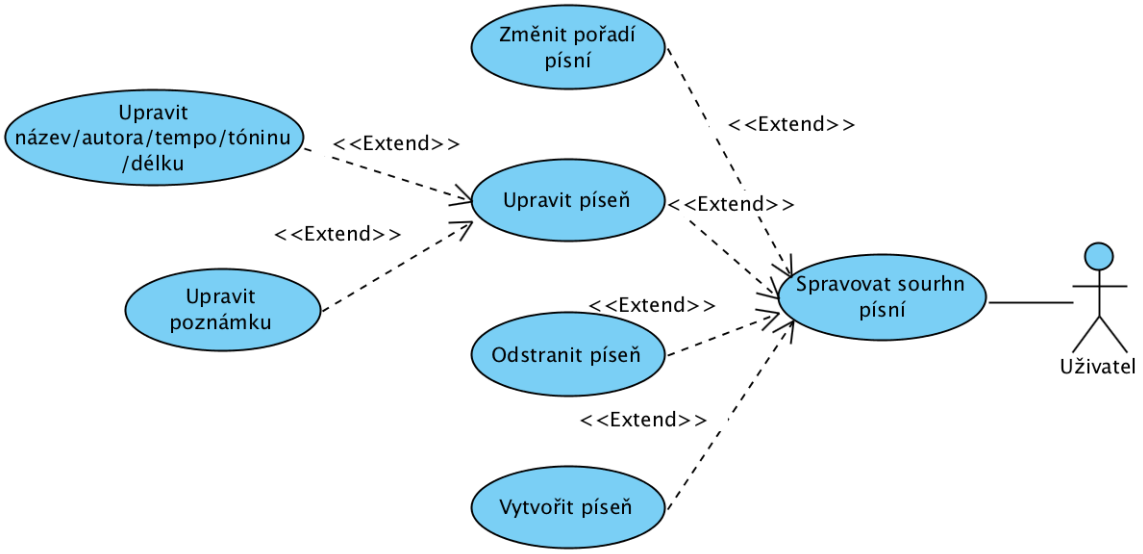
Část use-case diagramu zabývající se sledováním písní během koncertu nebo zkoušky je zobrazena na obrázku 4.3. Uživatel v takové situaci potřebuje zobrazit detail písně, aby viděl její podstatné údaje tzn. název, autora, tóninu, tempo, délku a své poznámky. Po odehrání písně pak uživatel potřebuje přejít na další píseň, což by mu mělo být umožněno co nejjednodušším způsobem. Zároveň by pro pohodlné zacházení měla existovat možnost přejít i na píseň předchozí.



Obrázek 4.3: Úkony spojené se sledováním jednotlivých písní uvnitř playlistu. Uživatel na koncertě bude chtít jednoduše bez námahy procházet jednotlivými písněmi uvnitř vytvořeného playlistu, přičemž bude zjišťovat jednotlivé informace o písních.

4.1.4 Správa souhrnu všech písní

Správou souhrnu všech písní ze zabývá část use-case diagramu vyobrazená na obrázku 4.4. Vedle jednotlivých playlistů existuje i souhrn všech písní. Tyto písně jsou pak dle potřeby vkládány a využity v jednotlivých playlistech. Zjednodušeně by se dalo, že se jedná o nepojmenovaný playlist obsahující všechny písně v aplikaci.



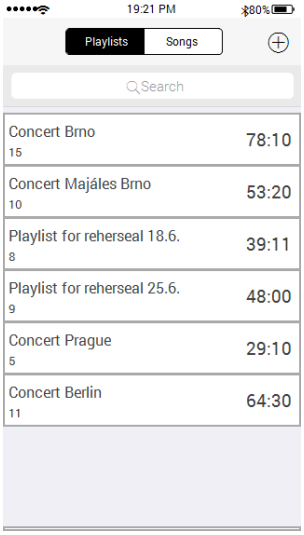
Obrázek 4.4: Úkony spojené se správou globálního seznamu písní. Uživatel zde bude přidávat a odstraňovat všechny jeho písně. Písně pak bude chtít upravovat, mazat a psát k nim poznámky. Písně bude chtít pro přehlednost i řadit.

4.2 Návrh layoutu aplikace

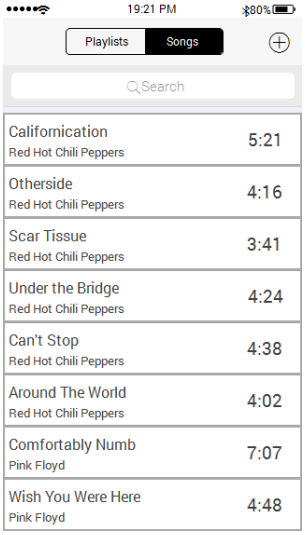
Před samotnou implementací jsem si nakreslil několik možností, jak by moje aplikace mohla vypadat. Nejprve jsem se nedíval na již existující aplikace s podobnou problematikou, abych se podvědomě neupnul ke konkrétnímu řešení. Od začátku jsem k jednotlivým návrhům přistupoval s myšlenkou jednoduchosti. Hlavní výhodou mé aplikace měla být její intuitivnost a praktičnost. Toho jsem se po celou dobu držel. Po zformování základní představy rozvržení layoutu na papír, jsem si na notebooku vytvořil nebarevné návrhy pro jednotlivé obrazovky.

4.2.1 Návrh úvodní obrazovky

Po spuštění aplikace bude uživateli zobrazena úvodní obrazovka. Návrh této úvodní obrazovky lze vidět na obrázku 4.5. Jelikož uživatel bude ve většině případů aplikaci spouštět z důvodu správy playlistů, úvodní obrazovka zobrazuje jejich seznam. Hned vedle playlistů by ovšem uživatel měl mít možnost zobrazit všechny své písně. Přidávání playlistu a přidávání písní by pak mohlo být spravováno stejným tlačítkem, které by se chovalo v závislosti na stavu úvodní obrazovky. Na závěr jsem se rozhodl pro přidání posledního elementu, kterým je vyhledávací okénko. To se nad seznamy v aplikacích na platformě iOS objevuje běžně a určitě u některých uživatelů najde své využití. V případě jeho nevyužití nezabírá velkou plochu obrazovky a uživateli nezpůsobuje zbytečné zmatení.



(a) Obrazovka ve stavu zobrazení seznamu playlistů.



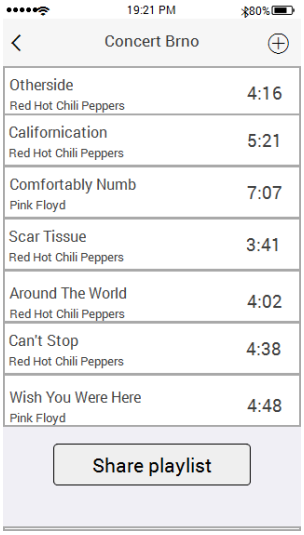
(b) Obrazovka ve stavu zobrazení seznamu písní.

Obrázek 4.5: Na obrázku lze vidět návrh úvodní obrazovky. Vpravo nahoře je tlačítko pro přidání playlistu/písně v závislosti na stavu úvodní obrazovky, který lze měnit ovládacím prvkem nahoře uprostřed. U seznamu playlistů lze vidět počet písní uvnitř playlistu, jeho název a celkovou délku. U seznamu písní název písně, autora a délku.

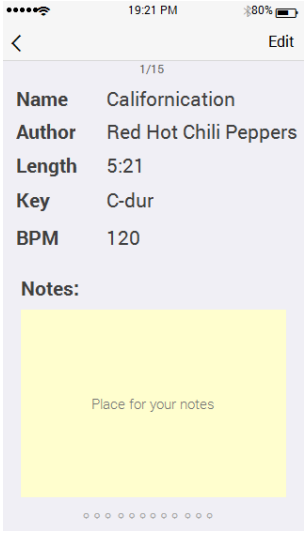
4.2.2 Návrh obrazovky pro detail playlistu a písně

Obrazovka pro detail playlistu se složením o mnoho neliší od úvodní obrazovky v režimu zobrazení seznamu všech písní. Není zde však využito vyhledávací okénko. Playlist ve většině případů nebude obsahovat příliš mnoho písní. Navíc je uvnitř playlistu velmi důležité pořadí jednotlivých písní a vyhledávací okénko by toto pořadí mohlo zdánlivě změnit. Navíc však bude obsahovat tlačítko umožňující sdílení playlistu. Návrh obrazovky detailu playlistu lze vidět na obrázku 4.6a.

Obrazovka pro detail písně bude zobrazovat jednotlivé informace o názvu, autorovi, délce, tempu a tónině. Dále bude obsahovat okno pro poznámky. V horní části obrazovky bude obsahovat pořadí písně v aktuálním playlistu. Jelikož bude možnost procházet playlisty i skrz detaily písní, ve spodní části obrazovky bude grafický element naznačující tuto funkcionalitu. Tento element je využit například i v oficiální aplikaci pro přehled o počasí od firmy Apple. Návrh obrazovky detailu písně lze vidět na obrázku 4.6b.



(a) Návrh obrazovky pro detail playlistu.



(b) Obrazovka ve stavu zobrazení detailu písně.

Obrázek 4.6: Na obrázku (a) lze vidět návrh obrazovky pro detail playlistu. V horní liště je tlačítko pro navrácení se na úvodní obrazovku, vpravo tlačítko pro přidání písně do playlistu. Ve spodní části je pak tlačítko pro sdílení playlistu. Na obrázku (b) lze vidět návrh obrazovky pro detail písně. V horní liště napravo je tlačítko pro editaci. Dále obsahuje podstatné informace o písni a prostor pro poznámky.

4.2.3 Návrh obrazovky pro vytvoření a editace písně

Při vytváření nové písně chce uživatel vyplnit pro něj podstatné informace, ty potvrdit a vytvořit tak píseň. Pro zadávání názvu a autora bude uživatel využívat systémovou klávesnici. Pro zadávání tempa, tóniny a délky pak uživatel využije *UIPickerView*, které umožňuje snadné zadávání přednastavených hodnot. *UIPickerView* je využito například v oficiální aplikaci pro nastavení času pro budík a minutku od firmy Apple. Vytvoření písně uživatel potvrdí tlačítkem.

Pro editaci písně bude využito stejného návrhu obrazovky. Lišit se ovšem bude v tom, že při otevření okna budou jednotlivé kolonky již předvyplněny aktuálními informacemi upravované písně. Obrázek 4.7a ukazuje návrh obrazovky pro vytváření a editaci písně.

4.2.4 Návrh obrazovky pro vytvoření a editaci playlistu

Pro nový playlist uživatel musí zvolit jeho jméno a vybrat písně z globálního seznamu písní, které chce do playlistu zahrnout. Aby uživatel nemusel přidávat písně po jedné, bude mu zobrazen seznam písní, ve kterých požadované písně označí. Označené písně se pak do playlistu přidají, jakmile uživatel použije tlačítko pro jeho vytvoření.

Editace playlistu funguje stejným způsobem. Uživateli se ovšem nebude zobrazovat seznam všech písní, ale jen seznam písní, které doposud nebyly do editovaného playlistu zahrnuty. Navíc bude v kolonce název již vyplněn dosavadní název editovaného playlistu. Editaci pak uživatel potvrdí tlačítkem.

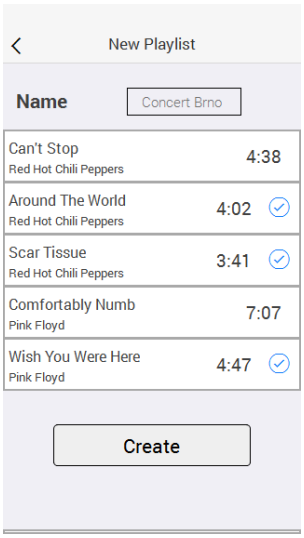
V horní liště obrazovky je tlačítko pro přidání písně do globálního seznamu písní. Rozhodl jsem se ho sem umístit z důvodu, že zejména na začátku používaní aplikace se uživateli může stát, že by do playlistu rád zahrnul i píseň, kterou doposud do aplikace nepřidal. Kdyby tu zde tlačítko pro přidání písně nebylo, musel by se uživatle vrátit na úvodní obrazovku, přidat píseň a následně playlist editovat nebo znova vytvářet. Návrh obrazovky je uveden na obrázku 4.7b.

4.2.5 Návrh obrazovky pro správu kontaktů

Jak jsem již zmínil, aby uživatel mohl při sdílení playlistu pohodlně zvolit obvyklé adresáty, bude uživateli při pokusu o sdílení playlistu zobrazena obrazovka s jeho kontakty.

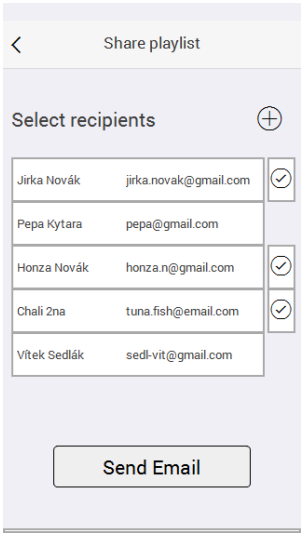
Na této obrazovce bude seznam jeho kontaktů a tlačítko pro přidání kontaktu nového. Při stisknutí tlačítka pro přidání kontaktu se uživateli zobrazí okno s dvěma okny, kam zadá jméno a email kontaktu. Pro odeslání emailu pak uživatel zvolí požadované adresáty a stiskne tlačítko pro odeslání. Tím se přesměruje do rozhraní *MFMailComposer*, kde se mu zobrazí předpřipravený email, který stačí potvrdit a email bude zpracován. Návrh této obrazovky lze vidět na obrázku 4.8a, návrh obrazovky při přidávání nového kontaktu lze vidět na obrázku 4.8b.

(a) Návrh obrazovky pro vytváření písně.

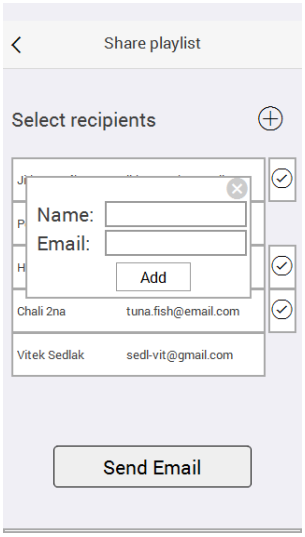


(b) Návrh obrazovky pro vytváření playlistu.

Obrázek 4.7: Na obrázku (a) lze vidět návrh obrazovky pro sdílení playlistu. V horní liště je tlačítko pro navrácení se na předchozí obrazovku. Hlavní část obrazovky tvoří seznam kontaktů, ze kterých může uživatel označením vybrat vhodné příjemce. Vpravo nad tímto seznamem je tlačítko pro přidáni nového kontaktu. Na obrázku (b) lze vidět stav obrazovky pro sdílení playlistu ve chvíli přidávání nového kontaktu. Po vyplnění jména a e-mailu kontaktu se tlačítkem pro přidání vytvoří nový kontakt.



(a) Návrh obrazovky pro sdílení playlistu.



(b) Návrh obrazovky pro přidání adresáta.

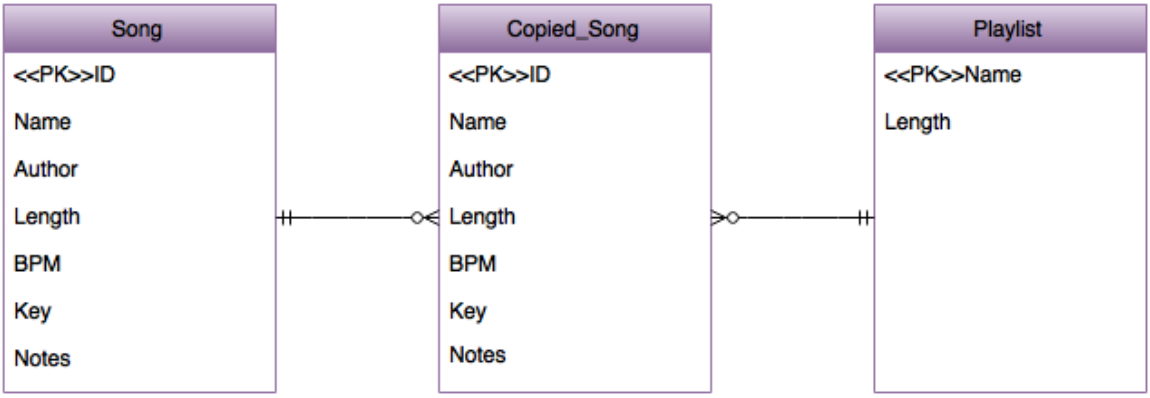
Obrázek 4.8: Na obrázku (a) lze vidět návrh obrazovky pro sdílení playlistu. V horní liště je tlačítko pro navrácení se na předchozí obrazovku, dole uprostřed tlačítko pro provedení vytvoření písně. V dolní části je vidět vysunutý UIPickerView, pomocí kterého se nastavuje délka písně. Na obrázku (b) lze vidět návrh obrazovky pro vytvoření playlistu. Je zde zobrazeno pole pro vyplnění názvu playlistu, seznam písní, které lze importovat, a tlačítko pro vytvoření playlistu.

Návrh datového modelu

Datový model se v průběhu testování od jeho původního návrhu několikrát změnil. Jelikož se jedná o jednu z nejdůležitějších částí návrhu, odrazila se každá změna do celkové podoby aplikace. Zejména se jednalo o problémy spojené se vztahem mezi písněmi a playlisty. Konečnou podobu uložení dat znázorňuje E-R diagram 4.9.

Nejprve bylo zamýšleno, že playlist bude obsahovat pouze odkaz na píseň, která bude uložena v globálním seznamu. To se však po prvním testování jevilo jako nevhodný přístup. Uživatelé totiž potřebovali, aby se jedna píseň mohla vyskytovat v různých playlistech s jinými informacemi o tónině a tempu. Dalším problémem bylo použití názvu písně jako jejího jednoznačného identifikátoru. V playlistech kapel se totiž může vyskytovat píseň stejného názvu, ovšem od jiného autora. Tyto problémy byly vyřešeny následovně.

Globální seznam písní obsahuje písně v originální podobě. Tyto písně tvoří základní šablonu pro kopie písní, které jsou využity uvnitř playlistu. Playlist tedy obsahuje pouze kopie originálních písní a může je měnit bez zásahu do globálního seznamu písní. Pro uživatelskou přívětivost by ovšem při změně písně uvnitř playlistu měla existovat možnost provést tyto změny i v globálním seznamu písní. V případě, že uživatel bude provádět změny písně uvnitř globálního seznamu písní, bude mu umožněno provést tyto změny i ve všech kopiích této písně.



Obrázek 4.9: Entity-relationship diagramu zobrazuje návrh uložení dat. Jak lze vidět, playlist obsahuje vždy pouze kopie originálních písní a každá kopie má jeden originál.

Kapitola 5

Realizace mobilní aplikace pro sdílení seznamů skladeb v kapele

V této kapitole nejdříve uvedu využité implementační prostředky a následovně popíši zajímavé části implementace uživatelského rozhraní. Dále rozvedu způsob, jakým jsou v aplikaci uchovávána data a na závěr přiblížím způsob implementace funkcionality aplikace, která umožňuje sdílení playlistů.

5.1 Implementační prostředky

I přes to, že je nyní většina nových aplikací pro platformu iOS vyvíjena v jazyce Swift 2.0, rozhodl jsem se svou práci psát ve starším jazyce Objective-C. Rozhodl jsem se tak hlavně z důvodu, že většina aplikací je i přes rozmach jazyka Swift stále napsána v Objective-C a přechod z programování v jazyku Objective-C na Swift je jednodušší, než zpětná snaha pochopit kód v jazyce Objective-C [8]. Pro implementaci jsem využil základní frameworky UIKit, Foundation, CoreGraphics a Message UI, které mi poskytly dostatečné prostředky pro práci s daty a ovládacími prvky aplikace. Celý vývoj probíhal ve vývojovém prostředí Xcode.

5.2 Implementace uživatelského rozhraní

Uživatelské rozhraní jsem vytvořil pomocí nástroje Storyboard. Jedná se o grafický editor, který umožňuje poskládat jednotlivé obrazovky aplikace (*View*) a přechody mezi nimi (*Segue*). Jednotlivé grafické prvky lze propojit s kódem a upravovat je v závislosti na stavu aplikace.

Navigaci mezi jednotlivými okny zajišťuje *Navigation Controller*. Ten z úvodní obrazovky jednotlivými přechody dává do popředí nové obrazovky. Ty se skládají na zásobník tak, jak jsou postupně zobrazovány. Pro návrat na předchozí obrazovku stačí, aby *Navigation Controller* odstranil z vrcholu zásobníku aktuální obrazovku. To udělá tehdy, dostane-li k tomu pokyn od uživatele. Pro řízení Navigation Controlleru slouží uživateli horní lišta. Mimo ovládací prvek umožňující návrat na předchozí obrazovku, může tato lišta obsahovat např. titulek aktuální obrazovky a další elementy. Tento prostor jsem využil například pro umístění tlačítka pro přidání nebo editaci.

Jednotlivé grafické prvky lze v nástroji Storyboard dle potřeby rozmístit v prostoru obrazovky. Aby byly prvky rozmístěné dle představy vývojáře na displejích o různých ve-

likostech, zavedl Apple tzv. *Auto layout*. *Auto layout* pravidelného rozmístění prvků docílí pomocí tzv. *constraints*. *Contraints* jsou limity a předpisy pro jednotlivé grafické prvky, které musí jednoznačně definovat pozici prvku v rámci dané obrazovky. Pro příklad použití *Constraints* uvedu umístění tlačítka s textovým polem. Tlačítku definujeme pevnou vzdálenost od levého a horního okraje a pevnou šířku. Tím se pozice tlačítka stává jednoznačná. Vedle tlačítka pak lze umístit textové pole, kterému zadáme vzdálenost od horního a pravého okraje. Místo šířky mu zadáme odsazení od již pevně umístěného tlačítka. Velikost tohoto textového pole se tedy bude dopočítávat v závislosti na velikosti aktuální obrazovky. Tímto způsobem jsou umístěny všechny grafické prvky uvnitř aplikace. Aby nedošlo k nesmyslně nebo nejednoznačně definovaným *constraints*, kontroluje prostředí Xcode jejich postupnou definici a v případě nejasnosti na možný konflikt vývojáře upozorní a pokud to dovede, nabídne i řešení.

Chování prvků na obrazovce definuje controller. Každá obrazovka má svůj controller, který poskytuje data pro ním řízenou obrazovku a reaguje na činnost uživatele. V Objective-C je controller třída nejčastěji odvozená od *UIViewController*, nebo *UITableViewController*.

5.2.1 Zobrazení písní a playlistů

Jelikož moje aplikace pracuje z velké části se seznamy písní a playlistů, tvoří jednu z jejích stěžejních součástí tabulky. Obrazovku s tabulkou zobrazující seznam informací lze vytvořit pomocí instance třídy *UITableView*. Chování, vzhled a zobrazované informace takové tabulky řídí *UITableViewController*.

Třída controlleru obsahuje metodu `ViewDidLoad`. Tato metoda, jak již její název naznačuje, je volána při prvotním načtení obrazovky. Je zde tedy možné předpřipravit data, které budou následně v tabulce zobrazena. Jelikož se ale v mé aplikaci zobrazovaná data neustále mění, jejich přípravu obsluhuji v metodě jiné a to metodě `ViewWillAppear`. Tato metoda je volána vždy, když se má obrazovka zobrazit. Tak je zajištěno, že uživateli bude vždy zobrazen aktuální stav jeho dat. Načtená data je pak potřeba v tabulce zobrazit. K tomu uvnitř třídy *UITableViewController* slouží následující metody.

První z nich je povinně implementovaná metoda `numberOfRowsInSectionTable`. Tato metoda je volána pro zjištění počtu řádků v zadané sekci tabulky. Počet řádků odpovídá počtu načtených prvků. Tabulky v mé aplikaci nejsou děleny do sekcí. Metoda pro správu sekcí tedy nebylo nutno využít a implicitně vrací hodnotu 1.

Další povinnou metodou je `CellForRowAtIndexPath`. Property *IndexPath* značí index řádku v tabulce. Tato metoda pro tento index vrací buňku, která je vložena na odpovídající pozici v tabulce. Jelikož se jednotlivé buňky liší zobrazovanými daty, nikoliv strukturou, je uvnitř této metody využita metoda `dequeueReusableCellWithIdentifier`. Tato metoda vrátí buňku, která odpovídá názvu v parametru předaného identifikátoru, jejíž vzhled je možno definovat ve Storyboardu. Získaná buňka je naplněna odpovídajícími daty a umístěna do tabulky.

Již nepovinná metoda `comitEditingStyle:forRowAtIndexPath` umožňuje provést editaci pro zvolený řádek v tabulce. Editace je v mé aplikaci využita pro odstranění záznamu z tabulky. Stiskne-li tedy uživatel na buňce tlačítko *delete*, je zavolána tato metoda, která zajistí její odstranění a aktualizaci dat. Tlačítko *delete* je zobrazeno táhlým pohybem zprava doleva po těle buňky. S tímto způsobem odstraňování záznamu z tabulky by každý uživatel platformy iOS měl být seznámen.

5.2.2 Změna pořadí písní a playlistů

Zajímavou implementační částí aplikace je několik metod, obstarávajících pohodlné měnění pořadí písní a playlistů. Tento způsob změny pořadí je využit v oficiální aplikaci pro počasí. Uživatel zde podrží prst na buňce, která následně jakoby vystoupí z řady a uživatel ji táhlým pohybem může přemístit na požadované místo. Tohoto chování jsem dosáhl následující implementací.

Nejprve bylo potřeba přidat detektor dlouhého stisknutí, nebo-li *UILongPressGestureRecognizer*. Jedná se o podtřídu třídy *UIGestureRecognizer*, která detekuje delší stisknutí jedním či více prsty. Aby takové stisknutí bylo detekováno, musí uživatel toto stisknutí držet po určitou dobu. Během stisknutí uživatel nesmí prsty posunout o více než předem specifikovanou vzdálenost. V případě, že je takové stisknutí detekováno, je zavolána metoda `longPressGestureRecognized`.

Metoda `longPressGestureRecognized` nejdříve zjistí, kde a v jakém stavu se stisknutí nachází. Pro mě byly podstatné zejména dva stavy. Prvním z nich je *UIGestureRecognizerStateBegan*. Tento stav značí, že bylo gesto započato. Je tedy potřeba provést následující úkony:

1. Najít buňku, která odpovídá místu dotyku.
2. Vytvořit obrázek této buňky.
3. Z obrázku vytvořit *View*(obrazovka) a umístit jeho centrum do centra původní buňky.
4. V animaci trvající 0.25 sekundy pak *View* postupně změnit tak, že se jeho centrum posune v rámci osy y do pozice dotyku, postupně se nastaví jako neprůhledné, jeho rozměry se zvětší na 1.05-ti násobek původní velkosti buňky, což zajistí efekt pomyslného vystoupení obrázku buňky z řady ostatních buněk. Původní buňka je postupně nastavena jako průhledná.
5. Po skončení animace je celá původní buňka v rámci tabulky schována.

Druhý stavem je *UIGestureRecognizerStateChanged*. Tento stav znamená, že původní dotyk přetrvává, mění se ovšem jeho pozice. To pro aplikaci znamená, že uživatel buňku přesouvá na jinou pozici. V tomto stavu je třeba přesunout centrum *View* vytvořeného z buňky do centra dotyku na ose y a vhodně změnit indexy buněk v rámci tabulky. Jelikož se může stát, že uživatel bude chtít buňku přesunout na pozici v tabulce, která není momentálně na displeji zobrazena, je potřeba posunout obrazovku ve správném směru po ose y. To se děje v následujcících krocích:

1. Nejprve je potřeba zjistit vzdálenost dotyku od horního a spodního kraje.
2. Je-li jedna z těchto vzdáleností menší než výška buňky, je potřeba posunout offset obrazovky ve správném směru. Offset obrazovky se bude posouvat v závislosti na vzdálenosti dotyku od horní, či spodního okraje. Čím blíže okraji se dotyk nachází, tím větší číslo se k offsetu obrazovky přičítat.
3. Pokud se tak ještě nestalo, je spuštěn časovač, který v pravidelných intervalech přičítá vzdálenost zjištěnou v předchozím kroku. Pokud ovšem pozice dotyku přesahuje velikost celé tabulky a uživatel by se snažil dostat buňku mezi rozsah, je v časovači vzdálenost posuvu nastavena na 0 a offset se přestane přičítat.

Další gesta není třeba rozlišovat, v rámci aplikace je jakýkoliv jiný stav indikátorem konce přesunu buňky. S koncem přesunu se provedou následující úkony:

1. Pokud byl během přesunu spuštěn časovač pro přičítaní offsetu, je potřeba zajistit, že bude vypnut a odstraněn.

2. Původní buňka, která byla na počátku schována, se opět objeví, ale prozatím je nastavena jako celkově průhledná.

3. Následuje animace trvající 0,25 sekund, ve které se postupně centrum *View* vytvořeného z buňky nastaví do centra pozice odpovídající buňky v tabulce, získá původní velikost buňky a celý obrázek se postupně nastaví jako průhledný. Původní originální buňka je postupně nastavena jako neprůhledná.

4. Po skončení animace je *View* vytvořené z originální buňky smazáno a odstraněno z obrazovky.

5. Nové pořadí buněk je zaznamenáno a uloženo.

5.2.3 Přechod na detail písně/playlistu

Pro přechod na detail buňky je v tabulce využita metoda `didSelectRowAtIndexPath`. Tato metoda je volána v případě, že uživatel stiskl některou z buněk tabulky. V mé aplikaci tento úkon znamená, že se uživatel snaží dostat na její detail. V této metodě je tedy proveden vhodný přechod.

Před samotným provedením přechodu (*segue*) je volána metoda `prepareForSegue:`. V této metodě se na základě předaného parametru nastaví data pro kontrolér cílového *View*. Do kontroléru cílového *View* jsou předány informace o písních, playlistech a indexu zvolené buňky. Jelikož je v rámci úvodní obrazovky implementováno vyhledávací okénko, je potřeba zvolený index při použití vyhledávání v rámci filtrovaných záznamů vhodně upravit. V případě přechodu na detail playlistu/písně za použití vyhledávače je třeba index přepočítat, aby byl předán index zvoleného playlistu/písně v rámci celého seznamu.

Práce s daty

5.2.4 Uložení a načítání dat v aplikaci

Při vývoji mé aplikace jsem se rozhodl data zapisovat do několika souborů. Tyto soubory následně ukládám do složky Documents, do které lze v rámci aplikace ukládat uživatelova data. Datovým formátem souborů je JSON[1]. Data ukládám do několika souborů z důvodu jejich častého načítání a zapisování.

První soubor obsahuje globální seznam písní. Jedná se o pole slovníků, kde pole zajišťuje udržení pořadí písní při práci se souborem a jednotlivé slovníky pak obsahují detaily písní. Ukázku obsahu takového souboru lze vidět v kódu 5.1.

[1] JavaScript Object Notation je způsob zápisu dat (datový formát) nezávislý na počítačové platformě, určený pro přenos dat.

```
[
  {
    "id" : 1,
    "author" : "Author Name",
    "name" : "Song Name",
    "key" : "A major",
  "pace" : 5,
    "length": 125,
    "details": "Details"
  },
  {
  "id" : 2,
  "author" : "Author Name",
    "name" : "Song Name",
    "key" : "B major",
    "pace" : 6,
  "length": 200,
  "details": "Details"
  }
]
```

Listing 5.1: Ukázka obsahu souboru pro uložení všech písní. Jedná se o pole slovníků. Číselná hodnota u klíče *id* je v rámci všech originálních písní unikátní. Klíč *author* představuje jméno autora, klíč *name* jméno písně, klíč *key* tóninu, klíč *pace* tempo a klíč *length* délku písně v sekundách. Pod klíčem *details* jsou uloženy poznámky k písni.

Druhý typ souboru obsahuje playlist. Jedná se o slovník, který uchovává informace ohledně názvu a délky playlistu, navíc k tomu obsahuje pole kopií písní z globálního seznamu písní. Kopie obsahuje z důvodu, aby v jednotlivých písních mohly vznikat změny i lokálně uvnitř playlistu. ID písně zůstává v kopii neměnné, díky němu je možno provézt lokální změny i do globálního seznamu a naopak. Souborů tohoto typu může být libovolné množství. Ukázku obsahu takového souboru lze vidět v kódu 5.2.

```
{
  "name" : "Playlist",
  "length" : 125,
  "songs" : [ ]
}
```

Listing 5.2: Ukázka obsahu souboru pro uložení playlistu. Jedná se o slovník. Pod klíčem *name* je uloženo unikátní jméno playlistu, pod klíčem *length* pak délka v sekundách. Pod klíčem *songs* je uloženo pole slovníků obsahující kopie písní.

Třetí typ souboru uchovává seznam všech playlistů uvnitř aplikace. Jedná se tedy o jednoduché pole řetězců s názvy playlistů. Tento soubor existuje ze dvou důvodů. Prvním z nich je nutnost udržení pořadí mezi playlisty. Playlisty jsou na obrazovce řazeny právě podle tohoto souboru. Provedené změny v pořadí uživatelem pak stačí zaznamenat do tohoto souboru. Zároveň soubor nese seznam všech playlistů a umožňuje jejich pohodlné načítaní a kontrolu unikátnosti jejich názvu. Ukázku obsahu takového souboru lze vidět v kódu 5.3.

```
[
  "First playlist name",
  "Second playlist name"
]
```

Listing 5.3: Ukázkou obsahu souboru pro uložení seznamu playlistů. Jedná se o pole názvu jednotlivých playlistů.

Posledním typem souboru je soubor pro uložení kontaktů. Jedná se pole slovníku, kde jednotlivé slovníky nesou údaje o kontaktech uživatele. Ukázku obsahu takového souboru lze vidět v kódu 5.4.

```
[
  {
    "email" : "first.mail@gmail.com",
    "name" : "First Mail"
  },
  {
    "email" : "second.mail@seznam.cz",
    "name" : "Second Mail"
  }
]
```

Listing 5.4: Ukázka obsahu souboru pro uložení kontaktních údajů adresátů. Jedná se o pole slovníků, ve kterých je pod klíčem *email* uložen e-mail adresáta a pod klíčem *name* jeho jméno.

Načítaní takto uložených dat pak probíhá následovně. Nejprve je nutno nastavit cestu do složky Documents. Cestu lze získat pomocí metody `NSHomeDirectory`, která vrací cestu do hlavní složky aplikace ve formě řetězce. V této složce se nachází složka Documents. Ze složky Documents jsou pak načítány konkrétní soubory. Syntaktickou analýzu souborů v datovém formátu JSON pak provede metoda `JSONObjectWithData:options:error:`. Načtená data jsou metodou převedena do editovatelného slovníku (*NSMutableDicitonary*) nebo pole (*NSMutableArray*) a je s nimi možno v rámci aplikace pracovat. Zapsání nových dat do souboru pak probíhá obdobně s rozdílem volané metody pro vytvoření souboru ve formátu JSON. Touto metodou je `dataWithJSONObject:options:error:`, které se předá slovník a metoda vytvoří soubor ve formátu JSON. Tento soubor je pak zapsán do složky Documents.

5.2.5 Načtení a uložení dat ze souboru v emailu

Pokud bude uživatel chtít importovat playlist přijatý v e-mailu, stačí mu v mé aplikaci otevřít soubor s koncovkou *.plsts*. Aplikace se pak již o vše postará a uživateli se nový playlist v aplikaci uloží. Pro tuto funkcionalitu bylo potřeba vytvořit novou UTI a zaregistrovat aplikaci jako vhodnou pro její zpracování. Toho jsem docílil následujícím způsobem.

Vytvoření UTI

Pro vytvoření nové UTI se využívá soubor uvnitř aplikace s názvem (*info.plist*). Jedná se o strukturovaný soubor, které nese nezbytné informace o konfiguraci aplikace. Obsah tohoto

souboru je strukturován pomocí XML[2]. Jelikož jsem vytvořil UTI novou, pro její deklaraci jsem do souboru *info.plist* přidal slovník s klíčem *UTExportedTypeDeclarations*. Jedná se o pole slovníku, kde každý slovník deklaruje nový typ, který vlastní tvůrce aplikace. Jeho celkovou podobu lze vidět v kódu 5.5.

```
<key>UTExportedTypeDeclarations</key>
<array>
  <dict>
      <key>UTTypeDescription</key>
      <string>New Playlist data</string>
    <key>UTTypeIdentifier</key>
    <string>asty.playlistsolution.plsts</string>
    <key>UTTypeConformsTo</key>
    <array>
      <string>public.data</string>
    </array>
    <key>UTTypeTagSpecification</key>
    <dict>
      <key>public.filename-extension</key>
      <string>plsts</string>
    </dict>
  </dict>
</array>
```

Listing 5.5: Pole *UTExportedTypeDeclarations* obsahuje jednotlivé slovníky deklarující mnou vlastněné, nově vytvořené UTI. V rámci mé aplikace mi stačil slovník jeden, u něhož nyní popíšu jednotlivé položky. Klíč *UTTypeDescription* obsahuje uživateli viditelný popis tohoto typu. V mém případě tento typ nese data obsahující v aplikaci vytvořený Playlist. *UTTypeIdentifier* je jednoznačný identifikátor pro novou UTI. *UTTypeConformsTo* určuje, od které UTI nově definovaná UTI dědí. *UITypeTagSpecification* je slovník, který definuje jeden či více koncovek, které odpovídají tomuto typu. V mém případě se jedná o jednu koncovku *.plsts*.

Po vytvoření nové UTI se aplikace musí přihlásit k podpoře pro jeho zpracování. To se opět provede uvnitř souboru *info.plist*. Tentokrát je sem třeba pole slovníků *CFBundleDocumentTypes*. Každý slovník nese informace o typu, se kterým bude aplikace schopna pracovat. Moje aplikace se přihlásila k podpoře jednoto typu deklarací slovníku, jehož celkovou podobu lze vidět v kódu 5.6.

Uvnitř slovníku je zajímavou položkou hodnota u klíče *LSHandlerRank*. Tato hodnota určuje, jakým způsobem je určováno pořadí nabízených aplikací pro zpracování souboru tohoto typu. Možné hodnoty jsou *OWNER* (tato aplikace stojí za vytvořením tohoto typu), *ALTERNATE* (tato aplikace je vhodná pro zpracování tohoto typu), *NONE* (tato aplikace nemá být použita k otevíraní tohoto typu souboru). Nabízené aplikace pro otevření souboru tohoto typu jsou pak řazeny podle této hodnoty v pořadí: *OWNER*, *ALTERNATE* a *NONE*.

[2]Extensible Markup Language – obecný značkovací jazyk

```
<key>CFBundleDocumentTypes</key>
<array>
  <dict>
    <key>CFBundleTypeExtensions</key>
    <array>
      <string>plsts</string>
    </array>
  <key>CFBundleTypeName</key>
  <string>New Playlist data</string>
     <key>CFBundleTypeRole</key>
     <string>Editor</string>
     <key>LSHandlerRank</key>
     <string>Owner</string>
     <key>LSItemContentTypes</key>
     <array>
       <string>asty.playlistsolution.plsts</string>
     </array>
  </dict>
</array>
```

Listing 5.6: Pole *CFBundleDocumentTypes* obsahuje jednotlivé slovníky nesoucí informace o podporovaných typech. V rámci mé aplikace mi stačil slovník jeden, u něhož nyní popíšu jednotlivé položky. Klíč *CFBundleTypeExtensions* obsahuje pole řetězců. Každý z těchto řetězců obsahuje souborovou koncovku, kterou aplikace rozezná a dokáže zpracovat. V mé aplikaci se jedná o souborovou koncovku *.plsts*. *CFBundleTypeName* obsahuje abstraktní jméno, který označuje typ dokumentu. Položka s klíčem *CFBundleTypeRole* popisuje vztah aplikace k danému typu. Hodnota klíče *LSHandlerRank* je *OWNER*, jelikož je aplikace zároveň vlastníkem tohoto typu. Pod klíčem *LSItemContentTypes* je uloženo pole řetězců. Každý tento řetězec obsahuje UTI definující podporovaný typ souborů. Je zde potřeba zadat její jednoznačný identifikátor, v mém případě tedy *asty.playlistsolution.plsts.*

5.3 Implementace vytváření PDF

Při vytváření PDF jsem se držel pravidel uvedených v kapitole 2.4.4. Samotná implementace pak probíhala následovně.

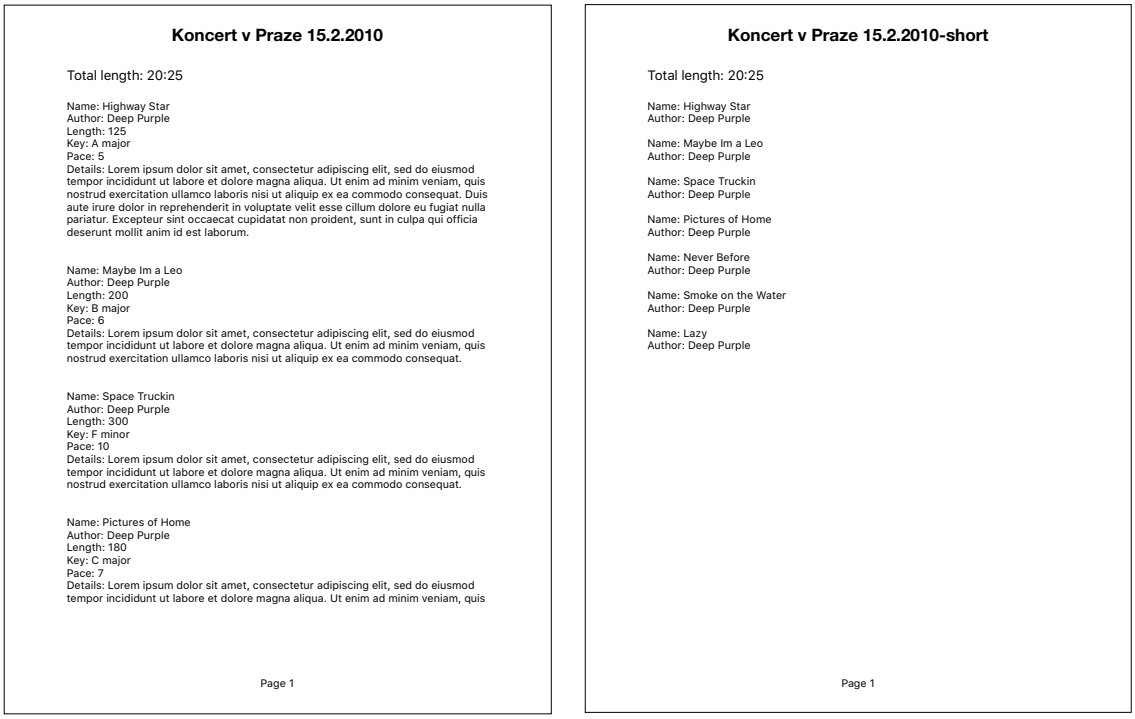
V prvním kroku se z jednotlivých položek uvnitř Playlistu vytvoří *NSAttributedString*. *NSAttributedString* obsahuje textový řetězec. Jednotlivým částem tohoto řetězce je možno nastavit různé atributy, jako je barva, velikost, font a jiné. Z vytvořeného řetězce *NSAttributedString* je pak vytvořeno samotné PDF.

Následně je pomocí funkce `UIGraphicsBeginPDFContextToFile` vytvořen kontext, do kterého bude výsledné PDF zapsáno a následně uloženo do souboru. Před začátkem kreslení je stanovena velikost stránky. Pokud se jedná o první stránku, je na ni nakreslen název celého playlistu. Následně je zavolána funkce pro vykreslení jednotlivých položek uvnitř playlistu.

V této funkci je vybrán prostor na stránce pro kreslení obsahu playlistu. Jelikož je normálně uvnitř kontextu kresleno zleva nahoru, je pro psaní textů zleva dolů potřeba nad kontextem zavolat následující funkce. První funkce `CGContextTranslateCTM` je funkce, pomocí které je souřadný systém pro kreslení uvnitř kontextu posunut v ose y k hornímu

okraji stránky. Druhou funkcí je `CGContextScaleCTM`, kterou lze upravit posouvání souřadného systému pro kreslení v ose x a y. V ose y je pro mou aplikaci nastavena hodnota -1, takže se souřadný systém v průběhu kreslení textu posouvá směrem dolů. Po dokončení kreslení je zjištěno, kolik textu se vešlo na stránku. Pokud nějaký text přesahoval, je uložena pozice posledního zakresleného znaku. Zbylý text je dokreslen na další stránky obdobným způsobem.

Ve chvíli, kdy je pozice posledního zakresleného znaku rovna délce celého textu, je proces kreslení ukončen. Následuje zavoláni metody `UIGraphicsEndPDFContext`, která uzavře kontext a jeho obsah uloží do na začátku specifikovaného souboru. Výsledek takto provedeného vykresleného PDF je vidět na obrázku 5.1.



(a) Dlouhá verze.

(b) Zkrácená verze.

Obrázek 5.1: Na obrázcích (a) a (b) lze vidět ukázku playlistu ve formě dvou PDF, vytvořených uvnitř aplikace. Na obrázku (a) je v PDF obsáhnutý celý playlist, včetně všech údajů o písních. Na obrázku (b) je zkrácená verze playlistu, která u písní uvádí pouze název a autora.

Kapitola 6

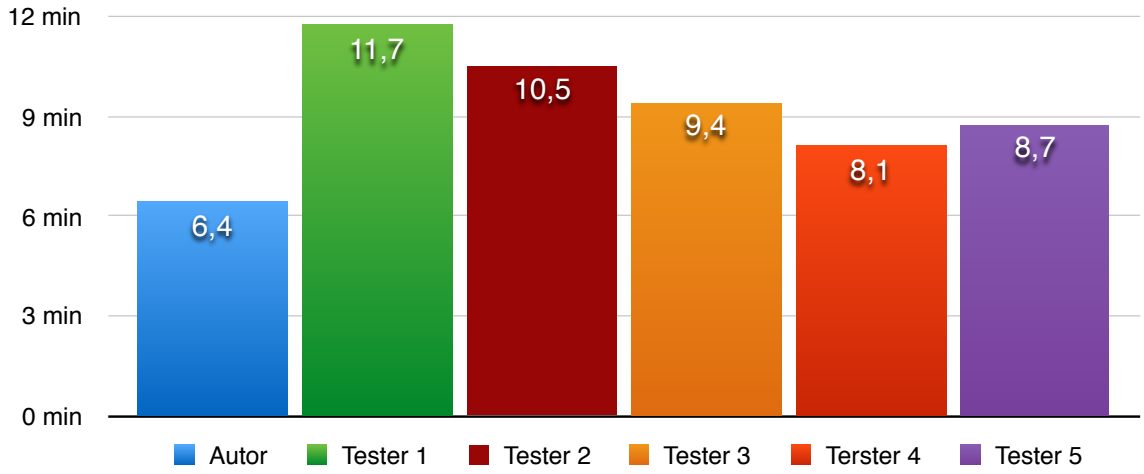
Testování

Díku tomu, že mám kamaráda, který hraje v kapele a vlastní zařízení iPhone jsem mohl aplikaci s jeho pomocí testovat během celého jejího vývoje. Díky této možnosti jsem odhalil několik chyb v návrhu datového modelu, tak i návrhu uživatelského rozhraní. Hlavní chybou datového modelu byla původní existence písně pouze v jedné podobě. Jak jsem již zmiňoval, stejná píseň se může hrát jiné tónině a tempu a tak je vytvářet její kopie. Další odhalenou chybou bylo používaní názvu písně jako jednoznačného identifikátoru. Píseň stejného názvu může existovat v jiné podobě od jiných autorů. Postupně jsem tedy aplikaci uvedl do stavu, kdy jsem se rozhodl ji otestovat na více lidech. Jelikož uvnitř aplikace používám některé ovládací prvky, které jsou specifické právě pro platformu iOS, rozhodl jsem se, že podmínkou pro vhodného testera bude předchozí zkušenost se zařízením iPhone, nebo iPad. Mojí podmínkou však nebylo, aby byl testující členem kapely. Cílem tohoto testování totiž bylo zjistit, jestli je aplikace intuitivní a zda jsou jednotlivé ovládací prvky snadno pochopitelné a zapamatovatelné. Testování jsem si připravil podle následujícího scénáře.

Testerovi jsem aplikaci představil jako produkt, který mám otestovat sám. To jsem udělal z důvodu, aby tester neměl nutkání ze slušnosti aplikaci chválit. Vysvětlil jsem mu, o co v aplikaci jde a způsob jakým jsem ji testoval. Následně jsem mu řekl, že jsem aplikaci testoval průběžně a teď si nejsem jist, jestli ji dokážu otestovat stejným stylem, jako bych ji viděl poprvé. Poprosil jsem ho tedy, jestli by nesplnil mnou připravené úkoly týkající se ovládání aplikace a pak mi neřekl, jestli nenarazil na nějaký nedostatek. Vysvětlil jsem mu, že kdyby náhodou nebyl schopný některý úkol splnit, ať si takový úkol poznačí a jde na další. Ujistil jsem ho, že pokud se do takové situace dostane, není to chyba jeho, ale aplikace. Pak jsem se od testera vzdálil a nechal ho plnit jednotlivé úkoly. Kdybych stál při testování vedle něj, mohl bych způsobit, že by se tester cítil hodnocen a nechoval se přirozeně. Mimo vědomí testera jsem stopoval, za jak dlouho zhotoví veškeré úkoly. Po dokončení testu jsem se testera zeptal, jakou část testu by označil za nejhorší a u kterého úkolu strávil nejvíc času. Samotné znění testu je uloženo v příloze. Celkově jsem uživatelské rozhraní testoval na pěti lidech. Výsledky testování jsou ukázány v grafu 6.1.

Po dokončení testování jsem od testerů zjistil nejslabší místa uživatelského rozhraní aplikace. Celkově jsem odhalil dva problémy, které jsem následně opravil.

První problém jsem odhalil hned díky prvním dvou testerům. Ti mě upozornili, že měli potíže s pochopením rozdílu mezi originální písní a její kopií uvnitř playlistu. Došlo mi, že jsem při přidávání písní do playlistu uživatele špatně naváděl slovy: „Add new songs to playlist“. Tester tak měl pocit, že nevytváří kopii, ale přidává originální píseň. Pro další testování jsem tedy tento nápis upravil na: „Copy new songs to playlist“. Další testeři už rozdíl mezi kopií a originálem písně chápali lépe.



Obrázek 6.1: Graf zobrazuje čas, který jednotliví testeři potřebovali pro splnění připraveného testu. *Autor* jsem já, svůj čas jsem uvedl pro srovnání. Jelikož jsem osobně nad jednotlivými úkoly nepřemýšlel a rovnou je plnil, lze z grafu vidět, kolik času jednotliví testeři strávili přemýšlením nad provedením požadovaných úkonů.

Druhým problémem byla editace playlistu. Testeři mě upozornili, že po stisknutí tlačítka *Edit* uvnitř detailu playlistu se uživateli rovnou zobrazí nové písně pro přidání. Testeři očekávali, že v editaci playlistu stále uvidí seznam obsažených písní, pouze se jim nabídnou nové možnosti pro práci s nimi. Tento problém jsem vyřešil následovně. V editaci playlistu se uživateli zobrazí obsažené písně, u kterých je přítomné tlačítko pro jejich rychlé smazání. Nad seznamem obsažených písní je navíc tlačítko, které indikuje možnost kopírování nových písní do playlistu. Seznam doposud v playlistu nezahrnutých písní se tak uživateli zobrazí až po stisknutí tohoto tlačítka. Jiné problémy testeři s uživatelským rozhraním nenalezli. Zdálo se jim intuitivní a přirozené.

Aplikace se díky průběžnému i závěrečnému testování od prvotního návrhu znatelně změnila. Její celkovou podobu lze vidět v příloze této práce.

Kapitola 7

Závěr

Cílem práce bylo vytvořit aplikaci na platformu iOS pro sdílení seznamů skladeb v kapele. Před samotnou implementací jsem musel nastudovat způsob správy skladeb ve fungujících kapelách. Ze získaných informací jsem vytvořil několik návrhů uživatelských rozhraní, na jejichž základě jsem implementoval první prototyp. Testováním prvotního prototypu jsem odhalil chyby, které jsem při vývoji nové verze aplikace odstranil.

Výsledná aplikace se jmenuje Playlists a je v angličtině. Umí ukládat skladby a vytvářet z nich playlisty. Tyto playlisty je pak možno snadno sdílet pomocí e-mailu. V aplikaci je oproti konkurenci možné mít adresář kontaktů, což uživatelům usnadní rozesílání playlistů obvyklým adresátům a u písní poskytuje prostor pro poznámky. Navíc umožňuje výskyt stejné písně v několika verzích. Výsledná aplikace byla průběžně testována a její uživatelské rozhraní bylo postupně vylepšováno směrem k dokonalosti.

Pro aplikaci mám vymyšleno mnoho rozšíření, které bych do ní v budoucnu rád zakomponoval. Mezi některé z nich patří možnost vytvoření přesné kopie již hotového playlistu, umožnění vkládání textů skladeb z notebooku, nebo počítače a přidání nové formy sdílení. Dále bych rád aplikaci lépe přizpůsobil pro zařízení iPad a využil tak potenciál větší obrazovky.

Aplikaci v blízké době plánuji umístit na App Store. Před tím si chci být ale jistý, že je aplikace kompletní a nebude zbytečně sbírat negativní hodnocení. Z toho důvodu je aplikace dále testována a vylepšována. Po jejím umístění na App Store plánuji reagovat na připomínky a návrhy širší škály uživatelů. Aplikaci plánuji dále rozšiřovat.

Projekt byl pro mě velice přínosný. Naučil jsem se programovat na mobilní platformu iOS, zdokonalil jsem se v objektovém programování a prací se systémem pro správu verzí. Také jsem si vyzkoušel celý proces vývoje mobilní aplikace od prvotního návrhu prototypu, přes konzultace s potenciálními zákazníky, až po finální testování a přípravy verze pro vydání.

Literatura

[1] Apple: *About the iOS Technologies*. 2014-09-17, [online]. [cit. 2016-05-8].
URL
https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html

[2] Apple: *Model–View–Controller*. 2015-10-21, [online]. [cit. 2016-04-10].
URL https://developer.apple.com/library/ios/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html

[3] Apple: *Uniform Type Identifiers Overview*. 2015-10-21, [online]. [cit. 2016-05-7].
URL https://developer.apple.com/library/ios/documentation/FileManagement/Conceptual/understanding_utis/understand_utis_intro/understand_utis_intro.html

[4] Apple: *App Store Review Guidelines*. 2016, [online]. [cit. 2016-05-7].
URL https://developer.apple.com/app-store/review/guidelines/

[5] Apple: *iOS Human Interface Guidelines*. 2016-03-21, [online]. [cit. 2016-05-5].
URL https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/

[6] Hegarty, P.: *Stanford University - Developing iOS 7 Apps for iPhone and iPad*. YouTube, 2014, [online]. [cit. 2016-05-7].
URL https://www.youtube.com/watch?v=ZqKbN_C4Yvgl

[7] Mayo, B.: *Xcode 7 allows anyone to download, build and 'sideload' iOS apps for free*. 9to5mac.com, 2016-05-10, [online]. [cit. 2016-05-6].
URL http://9to5mac.com/2015/06/10/xcode-7-allows-anyone-to-download-build-and-sideload-ios-apps-for-free/ [online]

[8] Miller, E.: *What should I learn first, Objective C or Swift?* YouTube, 24.3.2015, [online]. [cit. 2016-05-4].
URL https://www.youtube.com/watch?v=uC0PC1M0Mas

[9] Weber, H.: *Apple Seeks a Swift Way to Lure More Developers*. 2015-02-06, [online]. [cit. 2016-05-10].
URL http://venturebeat.com/2014/06/02/apple-introduces-a-new-programming-language-swift-objective-c-without-the-c/

Přílohy

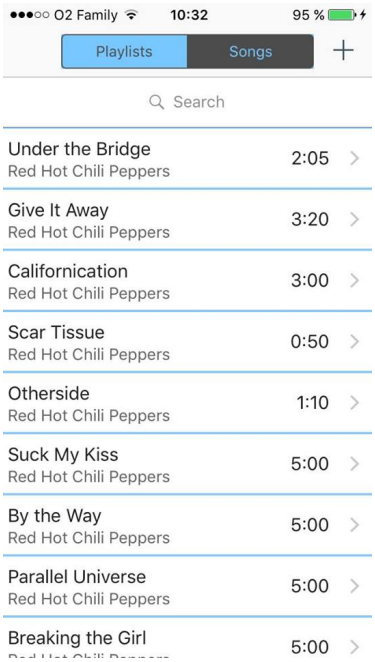
Příloha A

Protokol testování

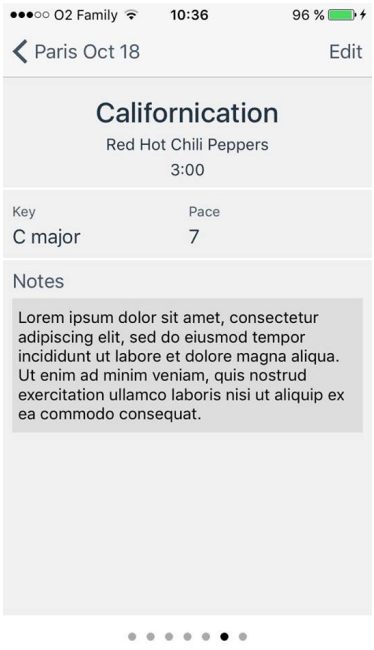
1. Vytvořte píseň s následujícími atributy:
   - název: Highway to Hell
   - autor: AC/DC
   - délka: 2min 20sec
   - tempo: 250BPM
   - tónina: C-minor
2. K vytvořené písni napište libovolnou poznámku.
3. Vytvořte druhou píseň s následujícími atributy:
   - název: Stairway to Heaven
   - autor: Led Zepellin
   - délka: 1min 30sec
   - tempo: 250BPM
   - tónina: D-major
4. Vytvořte playlist se jménem *Můj koncert*, který bude obsahovat obě vytvořené písně.
5. Otevřete nový playlist a změňte pořadí písní.
6. Sdílejte tento playlist s uživatelem s e-mailem: *mail@mail.com*.
7. Uvnitř playlistu nastavte tempo písně se jménem Highway to Hell na 100BPM a její tóninu na F-minor tak, aby původní podoba písně zůstala nezměněna.
8. Vytvořte nový playlist s názvem *Nový koncert*, při vytváření playlistu vytvořte novou píseň se jménem *Red Hot Chili Peppers* a zařaďte ji do tohoto playlistu.
9. Zobrazte playlist s názvem *Nový koncert* a přidejte do něj všechny nahrané písně.
10. Zobrazte detail písně uvnitř playlistu *Nový koncert* a bez návratu do přehledu playlistu zjistěte délku všech v playlistu obsažených písní.
11. Z playlistu *Nový koncert* smažte všechny písně.
12. Smažte všechny písně a playlisty z aplikace.

Příloha B

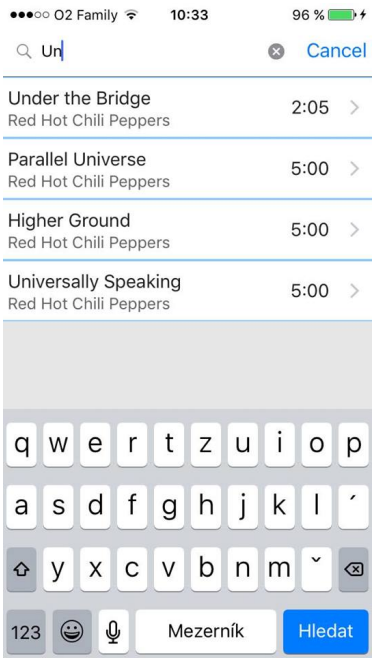
Výsledná podoba aplikace



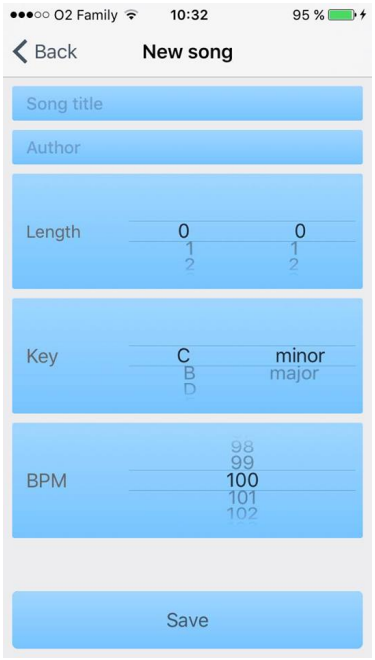
Obrázek B.1: Přehled všech písní.



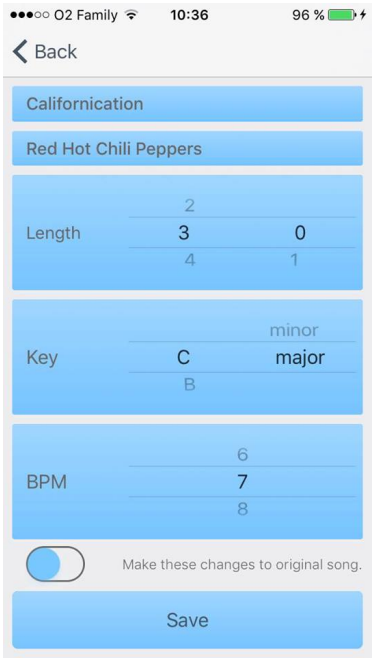
Obrázek B.2: Zobrazení detailu písně.



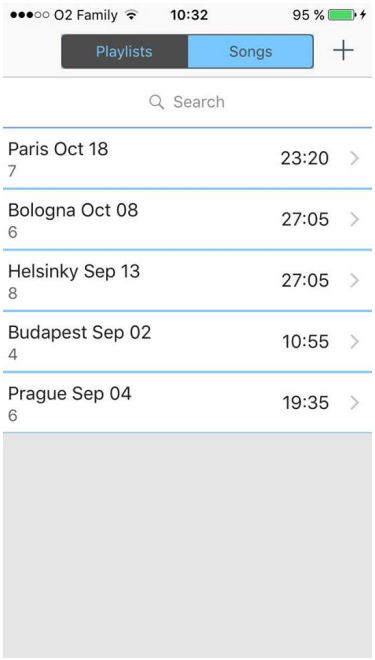
Obrázek B.3: Přehled písní po využití vyhledávání.



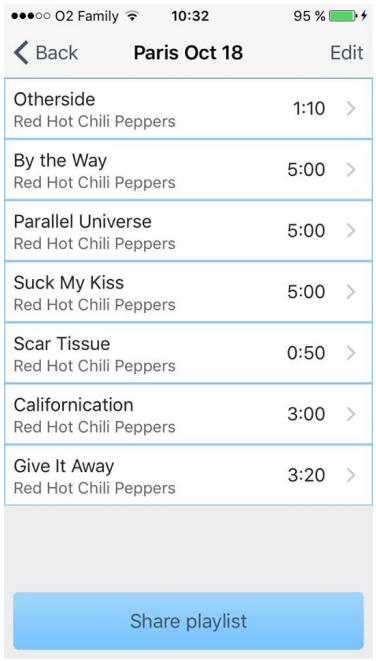
Obrázek B.4: Vytváření nové písně.



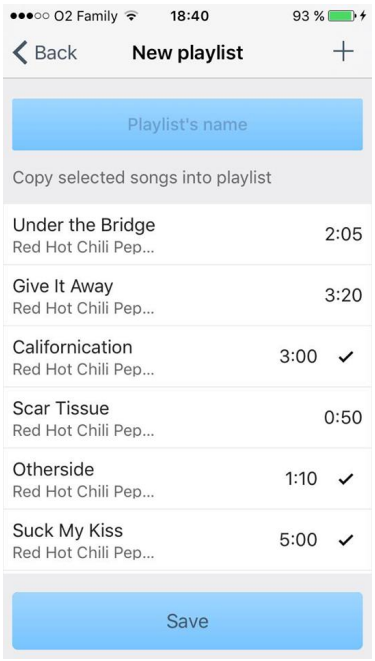
Obrázek B.5: Editace existující písně.



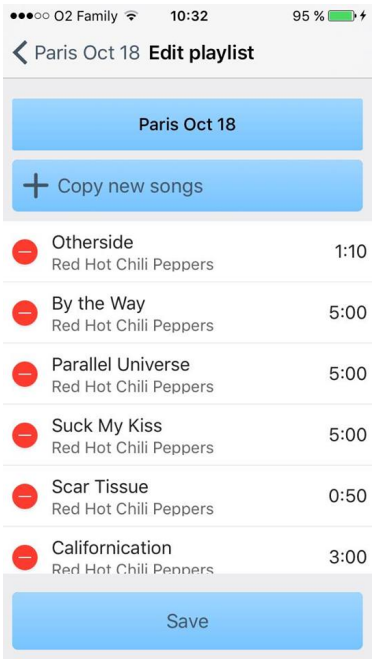
Obrázek B.6: Přehled všech playlistů.



Obrázek B.7: Zobrazení detailu playlistu.

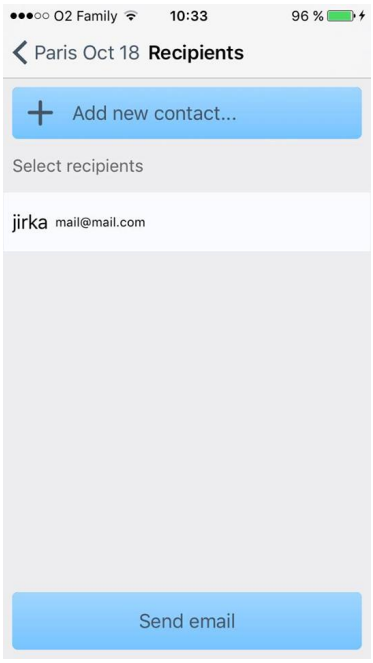


Obrázek B.8: Vytváření nového playlistu.

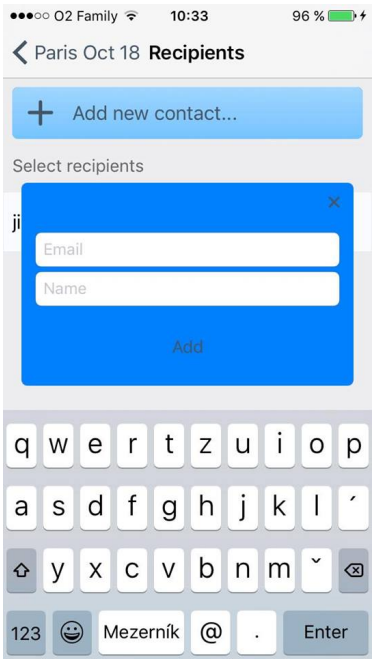


Obrázek B.9: Editace existujícího playlistu.

Obrázek B.10: Přidání písní do existujícího playlistu.



Obrázek B.11: Obrazovka s adresáty pro sdílení.



Obrázek B.12: Přidání nového adresáta.

Příloha C

Obsah CD

- **/TZ** – Technická zpráva ve formátu PDF
- **/TZ/src** – zdrojové soubory technické zprávy v LaTeX
- **/Playlists/src** – projekt programu Xcode se zdrojovými soubory
- **/VideoShort** – krátké prezentační video
- **/VideoLong** – delší video, sloužící jako návod k používaní aplikace
- **/Plakat** – plakát k aplikaci