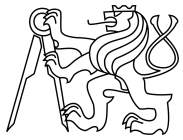
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Kytarový zpěvník pro Android
Student:	Gabriela Melingerová
Vedoucí:	Ing. Zdeněk Rybola
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Kytarový zpěvník je webová aplikace umožňující spravovat texty písniček s akordy, organizovat si je do zpěvníků a sdílet je mezi uživateli.
Cílem projektu je vyvinout mobilního klienta pro Android umožňujícího zobrazování a omezenou správu písniček a zpěvníků z webové aplikace skrze REST rozhraní.
Cílem této práce je realizovat backend aplikace, umožňující komunikaci s REST rozhraním webové aplikace, manipulaci s písničkami a zpěvníky, autentizaci uživatele, offline režim práce s daty a synchronizaci dat.

Dílčí úkoly v rámci řešení práce zahrnují:
1) Proveďte analýzu požadavků na mobilního klienta z pohledu backendu, analyzujte jednotlivé procesy s daty a práci v offline režimu.
2) Proveďte analýzu REST rozhraní webové aplikace.
3) Proveďte návrh backendu mobilního klienta pro Android.
4) Implementujte backend mobilního klienta.
5) Řešení řádně otestujte a zdokumentujte.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 1. února 2016

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

Bakalářská práce

Kytarový zpěvník pro Android

Gabriela Melingerová

Vedoucí práce: Ing. Zdeněk Rybola

16. května 2016

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 16. května 2016 .....

Odkaz na tuto práci

Melingerová, Gabriela. *Kytarový zpěvník pro Android.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Cílem této práce je realizovat backend aplikace, umožňující komunikaci s REST rozhraním webové aplikace, manipulaci s písničkami a zpěvníky, autentizaci uživatele, offline režim práce s daty a synchronizaci dat. Nejprve jsem se zaměřila na komunikaci mobilní aplikace s webovou aplikací pomocí REST rozhraní, aby aplikace byla schopná přijímat a odesílat potřebná data. Pro offline práci s daty jsem použila SQLite databázi, která je součástí operačního systému Android. Vytvořené řešení je schopné autentizace uživatele, manipulace s písničkami a zpěvníky, přidávání hodnocení a mnoha dalších funkcionalit, které poskytuje webová aplikace. Výsledek této práce umožňuje uživateli využívat mobilní aplikace Kytarový zpěvník.

Klíčová slova mobilní aplikace, analýza webové aplikace, Android, návrh, backend, kytarový zpěvník, Java, REST

Abstract

The aim of this thesis is to implement beckend applications which they can allow communication with REST interface of web application, handling songs and songbooks, user authentication, offline mode for working with data and data sychronization. First, I focus on communication with mobile application using REST interface. I wanted the application to be able to receive and send data. I used SQLite databases which is part of the Android OS to be able to work offline with data. Created solution is capable of authenticating users, handling songs and songbooks, adding rating and many other functionalities that web application provides. The result of this work allows user to use the mobile application Guitar songbook.

Keywords mobile application, analysis of web application, Android, design, backend, guitar songbook, Java, REST

Obsah

Seznam obrázků

Úvod

Používání mobilních aplikací se dnes stalo běžnou součástí našich každodenních životů. Mnoho lidí využívá mobilní aplikace pro různé účely. Například děti spouštějí aplikace za účelem zábavy, jiní lidé preferují aplikace, které jim usnadní organizaci jejich života.

Téma bakalářské práce jsem si zvolila z důvodu přínosu pro běžné uživatele, kteří chtějí mít své oblíbené písničky s akordy vždy při sobě a nechtějí sebou nosit papírové verze zpěvníků a písniček. Aplikace jim usnadní manipulaci s jejich oblíbenými zpěvníky a ještě jim ušetří místo v tašce.

V bakalářské práci se zabývám analýzou, návrhem a implementací aplikace Kytarový zpěvník. Konkrétně komunikací serveru s mobilní aplikací pomocí rozhraní REST.

V analýze se nejprve budu věnovat funkčním a nefunkčním požadavkům, dále se zaměřím na podobné aplikace, které se vyskytují na trhu. A v poslední řade se zmíním o rozhraní REST webové aplikace a obecně o REST technologii v Androidu. Dále potom v implementační části se budu zabývat rozšířením mobilní aplikace o různé části. Tedy implementací funkčních požadavků z analýzy, například, aby uživatel mohl přidat hodnocení ke konkrétní písničce a další funkcionality.

Tato práce navazuje na dvě bakalářské práce. První je bakalářská práce Jiřího Mantlíka, který se zabývá tvorbou webu, se kterým mobilní aplikace komunikuje. Druhou bakalářskou práci zpracovává Tomáš Havlíček, který se stará o prezentační vrstvu této aplikace.

Kapitola 1

Cíl práce

Cílem rešeršní části práce je analýza požadavků na mobilního klienta z pohledu backendu, analýza jednotlivých procesů s daty a práce v offline režimu, analýza existujících aplikací, analýza REST rozhraní webové aplikace a analýza REST rozhraní v Androidu.

Cílem praktické části je realizovat backend aplikace, umožňující komunikaci s REST rozhraním webové aplikace, manipulaci s písničkami a zpěvníky, autentizaci uživatele, offline režim práce s daty a synchronizaci dat.

Kapitola 2

Analýza

Analýza slouží k zanalyzování různých požadavků, ze kterých nám potom vyplynou funkcionality, které se snažíme naimplementovat. Kytarový zpěvník je aplikace, na které jsme se podílela v rámci předmětu Softwarový projekt 1 a Softwarový projekt 2. Spolu s týmem jsme zanalyzovali základní požadavky na aplikaci, jako jsou např.: zobrazování písní, vytváření zpěvníku, stažení písničky pro offline režim. Proto se během bakalářské práce soustředím na rozšíření funkcionalit této aplikace, neboť webový server poskytuje mnohem více možností, než které jsme implementovali.

2.1 Cílová skupina

Cílovou skupinou jsou lidé, kteří používají mobilní zařízení s operačním systémem Android a kteří chtějí využívat spravování písniček a zpěvníků pomocí mobilní aplikace, která komunikuje s webovou aplikací pomocí REST rozhraní. Cílová skupina není omezena věkem ani pohlavím. Stačí pouze schopnost umět využívat mobilní aplikace na uživatelské úrovni. Aplikace by měla hlavně zjednodušit manipulaci s daty, pokud uživatel ví, že mu bude chybět internetové připojení.

2.2 Funkční požadavky

2.2.1 Zapamatování si přihlášeného uživatele

Hlavním požadavkem na každou mobilní aplikaci, ke které se uživatel musí přihlašovat, je určitě zapamatování si přihlašovacích údajů. Uživatel tedy bude mít možnost se rozhodnout po vyplnění přihlašovacích údajů, jestli chce, aby si ho aplikace pamatovala či nikoli. Díky této funkcionalitě uživateli odpadá otravné opětované vyplňování svého přihlašovacího jména a hesla. Mobilní aplikace se snaží při znovuotevření navázat spojení s webovou aplikací pomocí údajů, které si o uživateli uložila.

2.2.2 Hodnocení písniček a zpěvníků

Každý, kdo bude používat tuto aplikaci, by jistě ocenil možnost hodnotit přidané písně a zpěvníky jiných uživatelů. Uživatel pomocí aplikace Kytarový zpěvník bude schopen písemně okomentovat danou písničku či zpěvník a přidat až pětihvězdičkové hodnocení. Dále aplikace nabízí funkcionalitu editovat již odeslané hodnocení a také jeho smazání. Dále se nám při rozkliknutí detailu písničky zobrazí průměrné hodnocení písně.

2.2.3 Komentování písniček a zpěvníků

Při prohlížení písniček nebo zpěvníků bude uživatel schopen zobrazit si komentáře, pokud existují, od jiných uživatelů či od sebe samotného. Dále je bude smět sám tvořit, editovat a mazat. Nastane-li situace, že se uživatel ocitl bez internetového připojení, může přidávat poznámky k písničkám nebo zpěvníkům, které si dříve uložil offline. Aplikace mu opět dovoluje mazání a editování těchto poznámek. Poznámky, jak již název napovídá, slouží k uložení poznatků, na které během svého používání uživatel natrefil např. špatné akordy atd..

2.2.4 Přebírání písniček a zpěvníků

Novou funkcionalitou, kterou nabízí webová aplikace, je přebírání písniček a zpěvníků. Přebírání písniček a zpěvníků spočívá v tom, že uživatel přebere veřejnou písničku či zpěvník, ke kterému může přidávat soukromé tagy (označení pro uživatele, např. country). Kdykoliv autor převzaté písně či zpěvníků změní či smaže obsah, uživateli přijde notifikace, co se stalo. U písní je možnost zkopírovat píseň před tím, než ji autor upravil. Aplikace samozřejmě nabízí i možnost zrušení přebírání dané písně či zpěvníku.

2.2.5 Kopírování písniček a zpěvníků

Kopírování písniček a zpěvníků se velmi podobá požadavku přebírání s drobným rozdílem. Uživatel zkopíruje obsah písně či zpěvník a tím se vytvoří jeho vlastní výtvor, který v případě zpěvníku může editovat (přidávat písně, odebírat písně, přidávat tagy, odebírat tagy atd.). Opět je tu provázanost mezi původním obsahem a stávajícím, na kterou uživatele upozorní notifikace.

2.2.6 Notifikování

Dalším požadavkem na aplikaci je schopnost zobrazovat notifikace uživateli, které se týkají jeho obsahu. Notifikace slouží jako informační prvek pro uživatele. Jak už tu bylo naznačeno, jedná se o komentování, přebírání, kopírování a hodnocení. Aplikace uživateli dovolí zobrazit si všechny notifikace a označit konkrétní notifikaci či všechny notifikace jako přečtené.

2.2.7 Nabízení písniček vyhovujících přání

Uživatel může pomocí aplikace přidat přání. Přání se skládá z názvu písně a interpreta písně. Po přidání písně se uživateli zobrazí seznam písní, které jsou uložené na webové aplikaci. Jako první se mu zobrazí písně, které stoprocentně vyhovují zadanému přání, dále se zobrazí písně vyhovující názvem a naposled písně, které odpovídají zadanému interpretu.

2.2.8 Upravení pořadí písniček

Při editování zpěvníku na svém mobilním zařízení, kde se dají editovat například písničky, které budou tvořit obsah zpěvníku, je kladen požadavek, aby se dalo měnit pořadí písniček. To znamená, že jakákoli změna pořadí písniček ve zpěvníků se promítne do webové aplikace. Uživatel tedy bude schopen měnit pořadí jednotlivých písní dle vlastní libosti.

2.3 Nefunkční požadavky

2.3.1 API webové aplikace

Mobilní aplikace je úzce svázána s konkrétní webovou aplikací. Cokoli uživatel změní na webové aplikaci se okamžitě promítne na mobilní aplikaci a naopak. To samé platí o vývojáři webové aplikace. Nepatrná změna API webové aplikace má za následek špatnou funkčnost mobilní aplikace.

2.3.2 OS Android

Mobilní aplikace vyžaduje, aby klient vlastnil zařízení, které má OS Android s verzí 4.0 (API 15) a výš. Android poskytuje mnoho různých verzí operačního systému. Ze statistik, které můžeme vidět na obr 2.1, je patrné, že celkem 97,4% uživatelů mobilního aparátu s OS Android bude schopno naši aplikaci využívat.

2.3.3 SQLite

Aby aplikace byla schopná ukládat data offline, je zapotřebí databáze. V tomto případě jsem zvolila databázi SQLite, která je součástí OS Android. To znamená, že uživateli stačí pouhá instalace Kytarového zpěvníku a ničeho dalšího. SQLite je kompaktní knihovna, která implementuje SQLite databázový systém. Na rozdíl od jiných databází nemá rozdělené serverové procesy, zapisuje a čte přímo z disku. SQLite je uživatelům poskytnuta zdarma [1].

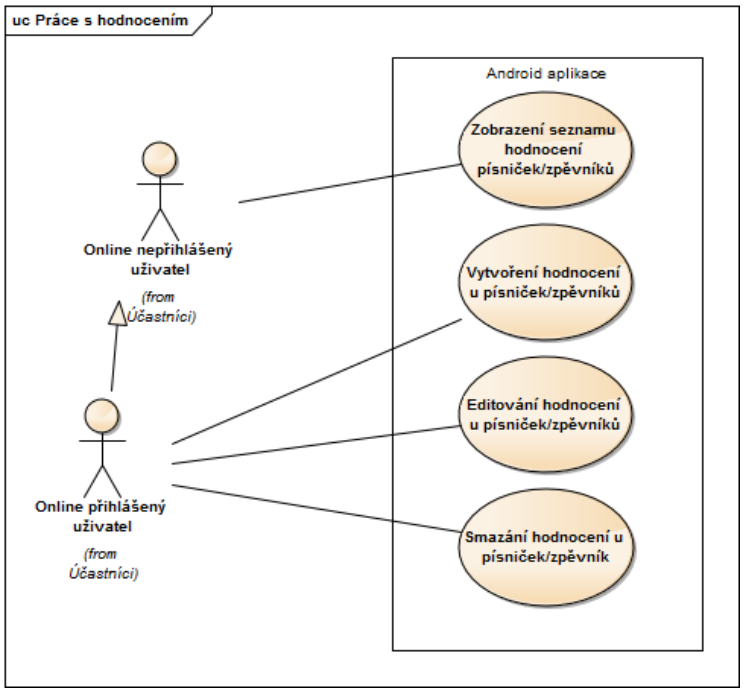
Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	2.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.2%
4.1.x	Jelly Bean	16	7.8%
4.2.x		17	10.5%
4.3		18	3.0%
4.4	KitKat	19	33.4%
5.0	Lollipop	21	16.4%
5.1		22	19.4%
6.0	Marshmallow	23	4.6%

Obrázek 2.1: Zastoupení jednotlivých verzí OS - zdroj [2]

2.4 Případy užití

Z analýzy požadavků nám vyplynuly tyto případy užití, které budou implementovány. Konkrétně se můžeme podívat na obr. 2.2, kde je vyzobrazena práce s hodnocením, a na obr. 2.3, kde vidíme případy užití týkající se notifikací.

- zapamatování přihlášení
- zobrazení seznamu hodnocení písniček/zpěvníků
- vytvoření hodnocení u písniček/zpěvníků
- editování hodnocení u písniček/zpěvníků
- smazání hodnocení u písniček/zpěvníků
- zobrazení seznamu komentářů písniček/zpěvníků
- vytvoření komentáře u písniček/zpěvníků
- editování komentáře u písniček/zpěvníků
- smazání komentáře u písniček/zpěvníků
- přebírání písně/zpěvníku



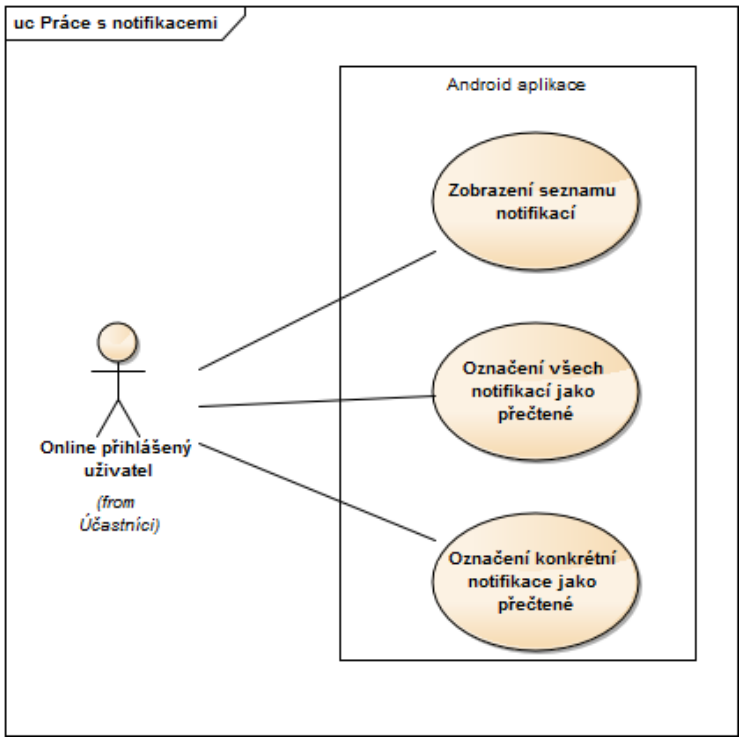
Obrázek 2.2: Případy užití - hodnocení

- zrušení přebírání písně/zpěvníku
- zahození staré verze písně
- kopírování písně/zpěvníku
- zobrazení seznamu notifikací
- označení všech notifikací jako přečtené
- označení konkrétní notifikace jako přečtené
- zobrazení seznamu písní vyhovujících přání
- upravení pořadí písní ve zpěvníku

2.5 Business procesy

2.5.1 Napsaní hodnocení ke konkrétní písni - online

Uživatel, který chce vytvořit hodnocení k písničce, která není jeho, si nejdříve spustí aplikaci a přihlásí se. Poté si pomocí aplikace vyhledá písničku, kterou chce ohodnotit a není jejím autorem. Nakonec vyplní slovní komentář a počet hvězdiček a přidá poznámku.



Obrázek 2.3: Případy užití - notifikace

2.5.2 Upravení pořadí písní ve zpěvníku - online

Uživatel, který chce upravit pořadí písní ve zpěvníku, musí být přihlášený na mobilní aplikaci. Poté specifikuje zpěvník, který chce upravovat. Následně si ve zpěvníku upraví pořadí jednotlivých písní a uloží změny, které vytvořil.

2.5.3 Přebírání konkrétní písně - online

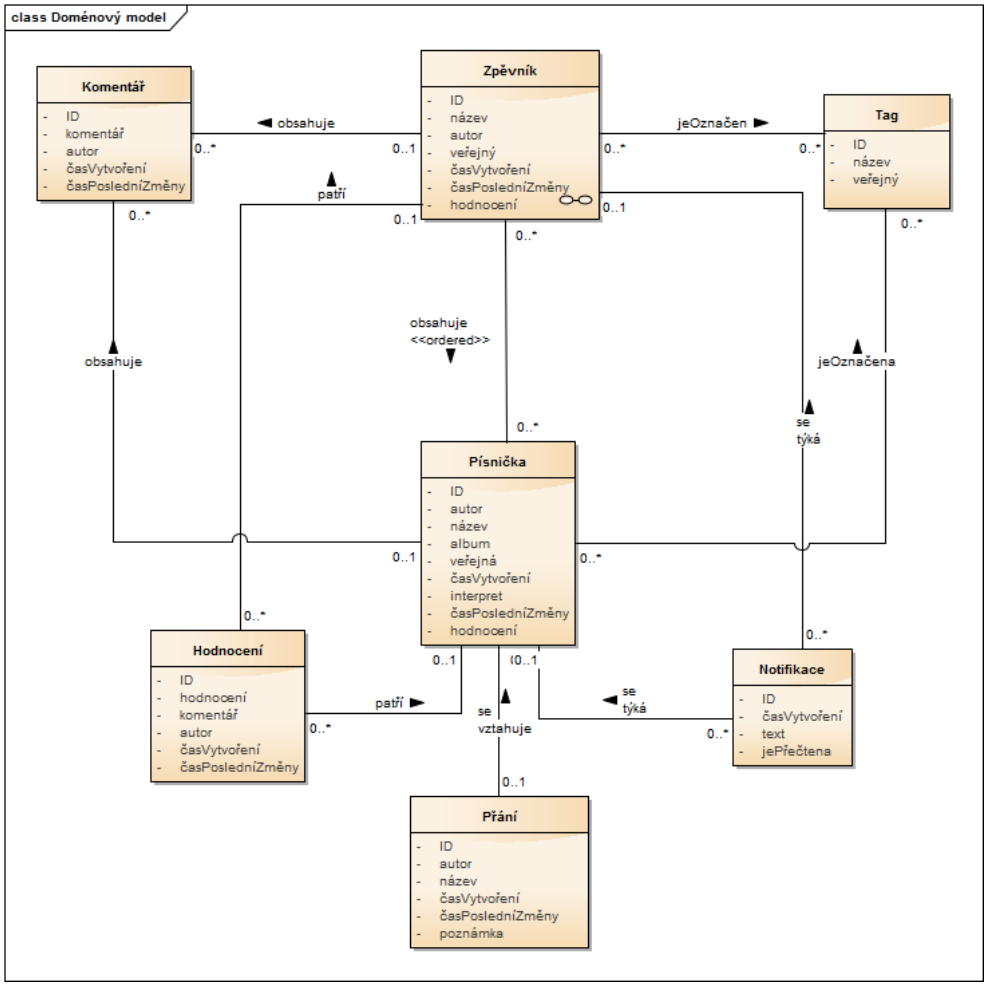
Uživatel, který chce přebrat konkrétní píseň, je přihlášený pomocí mobilní aplikace, kde specifikuje, kterou píseň chce přebrat. Následně si vybere možnost převzít píseň.

2.5.4 Editování konkrétní poznámky ke zpěvníku - offline

Uživatel, který chce editovat poznámku ke zpěvníku, si nejdříve spustí aplikaci. Poté si zobrazí svůj seznam offline zpěvníku a vyhledá si zpěvník, kde chce editovat konkrétní poznámku. Nakonec upraví slovní komentář a znovu odešle poznámku.

2.6 Doménový model

Doménový model má funkci zachytit jednotlivé entity, jejich atributy a vazby mezi dalšími entitami, které se budou vyskytovat v systému. Pomocí modelu se můžeme lépe zorientovat v problémové oblasti. Doménový model mnohdy slouží jako základ pro databázový model. V doménovém modelu k mobilní aplikaci jsou stěžejní tyto entity: zpěvník, písnička, tag, hodnocení, komentář, přání, notifikace. Výsledný doménový model si můžeme prohlédnout na obr. 2.4.

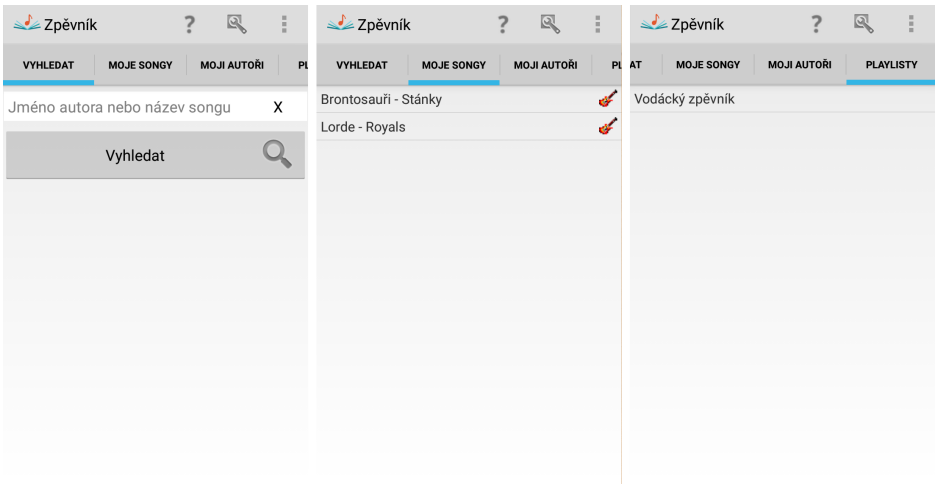


Obrázek 2.4: Doménový model

2.7 Podobné aplikace

Cílem bakalářské práce je vyvinout aplikaci, která je v určitých směrech lepší než již vydané aplikace. Proto důležitou částí analýzy je zaměření se na podobné aplikace, které jsou přístupné uživatelům. Z tohoto důvodu se soustředíme na konkurenční mobilní aplikace s OS Android, které si běžný uživatel může stáhnout na Google play zdarma. Jelikož se zabývám backend aplikacemi, soustředíme se spíš na funkcionality než na grafické uživatelské rozhraní.

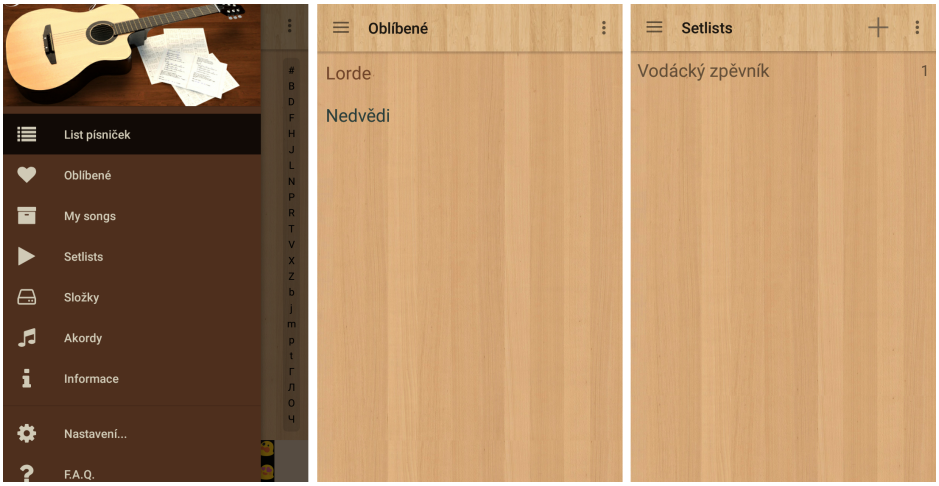
2.7.1 Zpěvník



Obrázek 2.5: Ukázka funkcí aplikace - Zpěvník - zdroj [3]

Zpěvník je jedna z prvních aplikací, které nám Google play zobrazí při specifikování požadavků. Primární funkcí aplikace je samozřejmě vyhledávání písniček, které uživatel požaduje. Aplikace je schopná vyhledávat podle jména autora a názvu písničky. Během vyhledávání si můžeme všimnout, že Zpěvník prohledává několik stránek a to například: pisnicky-akordy.cz, akordytexty.cz, zpevniky.wz.cz, supermusic.sk atd.. Výsledkem vyhledávání může být duplicitní obsah z různých stránek. Mobilní aplikace je schopná uložit písničku a zobrazit obsah bez přístupu na internet. Taky poskytuje možnost vytvoření playlistu, zařazení písničky do playlistu a editování písničky bez toho, aby se změnil obsah na serveru. Problém této aplikace je, že se uživatel nemůže přihlásit pod vlastním účtem a spravovat svůj obsah veřejně. Nemůže tedy například vytvořit playlist, který by uložil na server a který by ostatní uživatelé měli možnost prohlížet. Dále tu zcela chybí možnost hodnocení a komentování obsahu.

2.7.2 Písničky pro kytaru/Guitar Songs

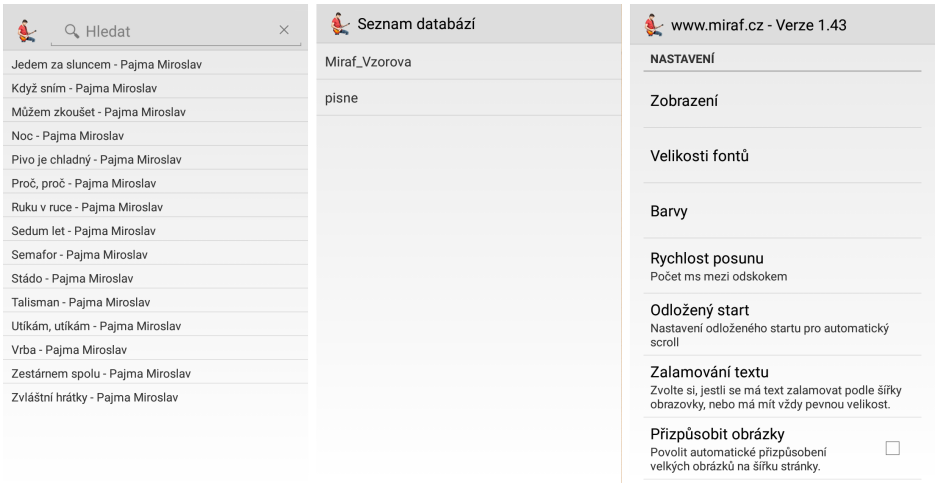


Obrázek 2.6: Ukázka funkcí aplikace - Písničky pro kytaru - zdroj [4]

Aplikace Písničky pro kytaru komunikuje se serverem en.guitarsongs.club. Je opět schopná vyhledávat dle jména autora a názvu písničky. Na rozdíl od Zpěvníku nám poskytuje možnost hodnotit jednotlivé písničky, ale pouze počtem hvězdiček, chybí slovní hodnocení. Oceňuji, že aplikace poskytuje funkci zobrazit nedávno přidané písničky na serveru. Díky této funkci může uživatel získat představu o nových písničkách. Uživatel, který si bude chtít vytvořit zpěvník skládající se z různých písní, má příležitost si vytvořit tzv. setlisty, které si pojmenuje a naplní dle libosti. Při prvním spuštění se všechny písničky stáhnou do databáze, takže i bez internetového připojení se zobrazí jejich obsah. To ovšem vede k tomu, že na mobilu budeme mít uloženy i některé písně/interprety, o které nestojíme. Aplikace tento problém řeší tím, že si jednotlivé písně můžeme smazat a interprety si můžeme skrýt.

2.7.3 Miraf SongBook

Miraf SongBook nabízí změnit URL adresy internetové databáze, ze které čerpá data. Implicitně písničky získává z miraf.cz/pisne.amsb. Také nabízí stažení internetové databáze pro offline použití. Při stažení databáze pro offline použití chybí příležitost promazat její obsah. Aplikace standardně vyhledává dle interpreta či názvu písně, bohužel nelze vytvářet zpěvníky, hodnocení, komentáře. Chybí tu úplně kontakt s dalšími uživateli, je to hlavně kvůli tomu, že uživatel si sám externě vytvoří internetovou databázi, kterou si naplní sám a tu pak využívá. Navíc tu neexistuje žádný účet, který by propojoval klienta mobilního zařízení s účtem konkrétního serveru. Dalším nedostatkem je ab-



Obrázek 2.7: Ukázka funkcí aplikace - Miraf SongBook - zdroj [5]

sence detailu písně tak jako u předchozích aplikací, které nenabízí podrobné informace o písni, např.: album, rok vydání, hodnocení atd..

Kapitola 3

Technologie

V této kapitole se nejprve zaměříme na OS Android, SQLite a REST. REST je taková architektura, která umožňuje programátorovi posílat data serveru přes HTTP protokol. Pro úspěšné používání RESTu je nutné znát základy HTTP protokolu.

3.1 OS Android

Android je operační systém, který můžeme dnes najít na mobilních zařízeních, tabletech, ale dokonce je i součástí Google Glass. Základ OS Android tvoří Linuxové jádro. Android je vyvíjen společností Google. Pro vývojáře, kteří chtějí tvořit aplikace pro mobilní zařízení s tímto operačním systémem, je velmi výhodné používat Android Studio. Android Studio je vývojové prostředí, které zjednodušuje vývoj aplikací a je plně zdarma k dispozici. Instalace Android Studia vyžaduje JDK, neboť aplikace pro OS Android jsou programovány pomocí jazyka Java. Jako každý jiný operační systém má Android mnoho verzí. Poslední verze má jméno Marshmallow a byla vydána 5. října 2015.

3.2 SQLite

Jak již tu bylo zmíněno, SQLite databáze je plně podporována OS Android, protože je součástí OS Android. Databáze se vytváří pomocí SQLiteOpenHelperu, kde se nadefinují jednotlivé entity, jejich atributy a integritní omezení (primární klíč, nesmí zůstat nevyplněné atd.). Manipulace s databází je velmi uživatelsky přívětivá. Jak již název napovídá, používá se pro komunikaci s databází SQL jazyk. Nejpoužívanější SQL příkazy s jejich použitím:

- `CREATE jmeno_tabulky (atribut INTEGER PRIMARY KEY);` - Vytvoří tabulku s atributem, který bude primárním klíčem s datovám typem integer.

- `SELECT * FROM jmeno_tabulky;` - Zobrazí všechny atributy dané tabulky.
- `DROP TABLE IF EXISTS jmeno_tabulky;` - Smaže zadanou tabulku, pokud existuje.

Příkaz CREATE se uplatňuje pro vytvoření tabulek, které budou tvořit databázi, kterou si navrhneme. SELECTem se v databázi nejčastěji vyhledává, ale může sloužit i k zisku určitého atributu, který je vstupem do nějaké funkce. DROP zase umožňuje smazat vybrané tabulky a tím smazat i celou databázi.

3.3 HTTP protokol

HTTP protokol slouží ke komunikaci programů přes celý web. Nejčastěji se využívá komunikace mezi webovými prohlížeči a webovými servery.

3.3.1 HTTP metody

HTTP protokol využívá několik metod, každý HTTP požadavek obsahuje metodu. Nejznámější jsou tyto metody:

- GET - Metoda slouží k získání určitých dat ze serveru ke klientovi.
- PUT - Pomocí PUT je klient schopen uložit data na server.
- POST - Metoda, která je schopna zaktualizovat či uložit data na serveru.
- DELETE - Tato metoda zase umí vymazat data, která jsou uložena na serveru.

Uživatel proto pošle jeden z těchto HTTP požadavků serveru. Server uživateli odpoví tak, že mu odešle HTTP odpověď.

3.3.2 HTTP status kód

Každá HTTP odpověď obsahuje tzv. status code. Status kód je číslo, které má tři cifry a které nám prozradí, jak náš HTTP požadavek obstál u serveru. Status kód, který začíná číslem 2, znamená, že HTTP požadavek byl úspěšně vykonán. Naproti tomu status kódy, které začínají číslem 4, značí chybu na straně klienta. Naopak jestli nastala chyba na straně serveru, vrátí se nám status kód s číslem 5 na začátku. Mezi nejčastější status kódy patří:

- 200 OK - Znamená, že všechno dopadlo úspěšně.
- 404 NOT FOUND - Data, která byla požadována, nebyla nalezena.
- 503 SERVICE UNAVAILABLE - Servis je momentálně nedostupný.

3.3.3 HTTP hlavička

HTTP hlavičky se rozdělují do několika skupin:

- Obecné hlavičky - Obsahují univerzální informace o zprávě, můžou se objevit jak v odpovědi tak i v dotazu.
- Hlavičky dotazu - Poskytují více informací o dotazu.
- Hlavičky odpovědi - Poskytují více informací o odpovědi.
- Hlavičky těla - Popisují délku těla a obsah.

Každá hlavička má jednoduchou syntaxi [6].

3.4 REST Android

REST v Androidu je podporován knihovnou Volley. Knihovna Volley je knihovna, která je poskytnuta společností Google a která velmi ulehčuje vývoj Androidích aplikací, které potřebují síťování. Nabízí podporu prioritizaci požadavků, více souběžných připojení k síti, plánování požadavků, pokročilé možnosti paměťového cashování. Důležitá je podpora pro zpracování JSON odpovědí, které nám vrací API webové aplikace [7]. JSON je formát, který není závislý na platformě a slouží pro výměnu dat na webu.

- JSONString – textový řetězec
- JSONNumber – číslo (celočíselné nebo reálné, včetně zápisu s exponentem)
- JSONBoolean – logická hodnota
- JSONNull – hodnota null
- JSONArray – pole
- JSONObject - objekt

Nás nejvíce zajímají JSONObject a JSONArray, které posíláme webové aplikaci a které nám zase webová aplikace vrací. Odpověď, která nám přijde, je nutné převést na konkrétní datový objekt, který je pak dále zpracován.

3.5 REST rozhraní webové aplikace

Dokumentace webové aplikace nám říká, jakou HTTP metodu a na jakou adresu ji máme odeslat. Také se dozvíme, jaký status kód nám vrátí při úspěšném přijetí. Jedná-li se o metodu POST, je dokumentace doplněna o strukturu požadavku, kterou musíme odeslat skrze JSONObject či JSONArray. Například,

chceme-li vypsat všechny notifikace, odešleme metodu GET s /notifications. Při správném zpracování nám webová aplikace vrátí JSONArray, kde se budou nacházet JSONObject, které se vztahují k notifikacím, a status kód 200, který jak již víme, znamená, že vše proběhlo v naprostém pořádku.

Kapitola 4

Návrh

V návrhové části práce se soustředíme na architekturu, kde výběr typu architektury je velice důležitý a může ovlivnit celé spektrum aplikace. Dále se seznámíme s databázovým modelem, který aplikace používá pro uložení dat, když uživatel nemá přístup na internet. A nakonec diagram nasazení nám přiblíží, co vše je potřeba pro to, aby vše fungovalo, jak má.

4.1 Architektura

Architektura je důležitou součástí návrhu každého informačního systému. Čím lépe je navržena, tím lépe se dá aplikace rozšiřovat o různé nové funkcionality a lépe se aplikace udržuje. Máme různé druhy architektury, např.: jednovrstvou, dvouvrstvou (která se rozděluje na prezentační a datovou vrstvu) a třívrstvou. Třívrstvá architektura se dělí na tři vrstvy: prezentační, business a datová. Třívrstvé architektury jsou vhodné pro rozsáhlejší aplikace. Z tohoto důvodu jsme ji zvolili v rámci předmětu Softwarový projekt. Zvolená architektura je navíc striktní. Striktní architektura je architektura, kde závislost proudí směrem dolů a pouze o jednu úroveň. To znamená, že prezentační vrstva nemůže přímo komunikovat s datovou, ale musí ke komunikaci využít business vrstvu. V rámci bakalářské práce mám na starosti business a datovou vrstvu.

4.1.1 Business vrstva

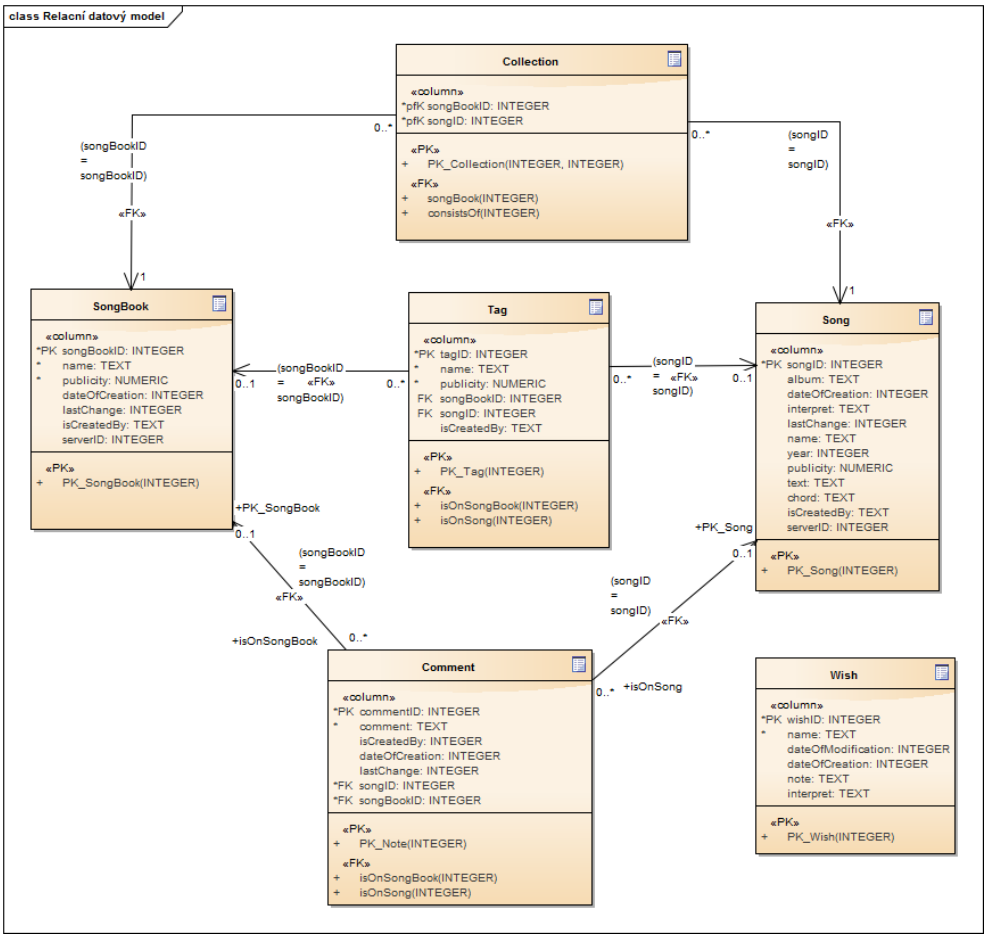
Výhoda třívrstvé architektury je ta, že business logika není závislá na způsobu uložení dat. Business vrstva se stará o logiku, výpočty a zpracovávání dat.

4.1.2 Datová vrstva

Datová vrstva řeší nejčastěji ukládání, načítání dat z databáze a získání dat z webové aplikace, které poté vrací business vrstvě.

4.2 Databázový model

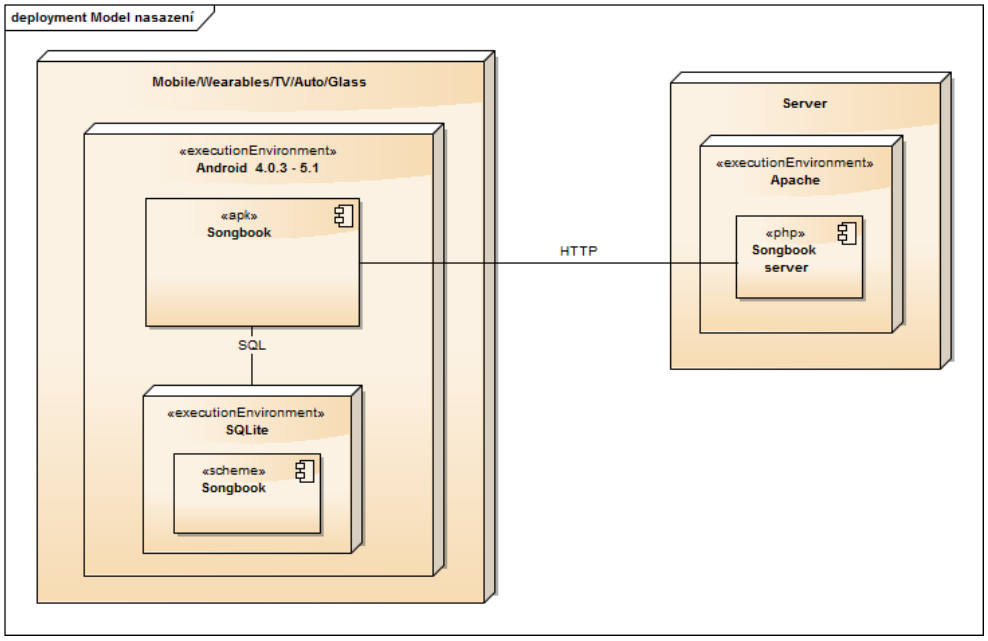
Databázový model slouží k představě o uložených entitách a vazbami mezi nimi. Databáze v aplikaci má funkci takovou, aby ukládala písničky, zpěvníky a poznámky v offline režimu, tedy aby uživatel měl k těmto datům kdykoli přístup bez nutnosti být připojen k mobilnímu internetu. Můžete si všimnout, že ve vztahu M:N mezi zpěvníky a písničkami (jeden konkrétní zpěvník může obsahovat více písniček a jedna konkrétní písnička se může nacházet ve více zpěvnících), byla provedena dekompozice, neboť binární notace ji vyžaduje. Dekompozice je rozdělení vztahu M:N pomocí entity, v tomto případě se jedná o entitu collection. Tím jsme docílili vzniku dvou vztahů 1:N. Databázový model, který se týká Kytarového zpěvníku můžeme vidět na obr. 4.1.



Obrázek 4.1: Databázový model použitý v aplikaci

4.3 Diagram nasazení

Diagram nasazení slouží k popsání fyzického rozvržení systému. Z diagramu můžeme vidět, který software běží na jakém hardwaru. Z obrázku 4.2 vyplývá, že prostředím pro běh (execution environment) aplikace Kytarový zpěvník je OS Android konkrétní verze a zařízení na kterém tato aplikace poběží je hardware (mobil, tablet atd.). Dále lze vypozorovat, že součástí OS Android je SQLite, které má v sobě schéma databáze, se kterým aplikace komunikuje pomocí jazyka SQL. Důležitým prvkem diagramu je také komunikace aplikace, konkrétně datové vrstvy, s webovou aplikací pomocí HTTP.



Obrázek 4.2: Diagram nazazení

Kapitola 5

Realizace

V této kapitole se nebudeme věnovat detailně všem implementovaným funkcionalitám (pouze si je shrneme), ale soustředíme se na části aplikace, které se věnují komentářům a zaslouží si naši pozornost. Budeme vycházet z poznatků, které jsme získali během analýzy a návrhové části.

5.1 Knihovna Volley

Jak již zde bylo zmíněno, pro komunikaci s webovou aplikací využívám knihovnu Volley. Pomocí této knihovny přijímám data z webové aplikace a také data webové aplikaci odesílám.

5.2 Parser

Webová aplikace nám vrací JSONObject či JSONArray a je tedy důležité převést tyto datové typy na skutečné objekty, které využívá mobilní aplikace. Proto jsem implementovala jednotlivé třídy, které se vztahují k jednotlivým datovým objektům z mobilní aplikace a které umějí z datových typů (JSONObject a JSONArray) vytvořit konkrétní objekty.

```
public static DComment parseCommentFromJson(JSONObject
    commentObject) throws JSONException
{
  int ID = Integer.parseInt(commentObject.getString("id"
      ));
  String comment = checkForNull(commentObject.getString(
      "comment"));
  if (comment != null)
    comment = EncodingConverter.convert(comment);
```

```
    String author = checkForNull(commentObject.optString("
        username", null));
    if (author != null)
      author = EncodingConverter.convert(author);

    Calendar dateOfCreation = parseDateFromDB(checkForNull
        (commentObject.optString("created", null)));
    Calendar lastChange = parseDateFromDB(checkForNull(
        commentObject.optString("modified", null)));

    return new DComment(ID, comment, author,
        dateOfCreation, lastChange);
}
```

Jelikož aplikace také pracuje s databází, musí být schopna převádět i cursory, které databáze vrací jako odpověď na dotazy, na datové objekty aplikace.

```
public static DComment parseCommentFromDB(Cursor cursor)
{
    if (cursor==null) return null;
    DComment comment = new DComment();
    comment.setID(cursor.getInt(cursor.getColumnIndex(
        DatabaseHelper.KEY_COMMENTID)));
    comment.setComment(cursor.getString(cursor.
        getColumnIndex(DatabaseHelper.KEY_COMMENT)));
    comment.setAuthor(cursor.getString(cursor.
        getColumnIndex(DatabaseHelper.KEY_ISCREATEDBY)));
    comment.setDateOfCreation(parseDateFromDB(cursor.
        getString(cursor.getColumnIndex(DatabaseHelper.
        KEY_DATEOFCREATION))));
    comment.setLastChange(parseDateFromDB(cursor.getString
        (cursor.getColumnIndex(DatabaseHelper.
        KEY_DATEOFCREATION))));
    return comment;
}
```

5.3 Naimplementované funkce

V implementační části se mi podařilo naimplementovat všechny požadavky, které jsme si určili:

- zapamatování přihlášení
- zobrazení seznamu hodnocení písniček/zpěvníků

- vytvoření hodnocení u písniček/zpěvníků
- editování hodnocení u písniček/zpěvníků
- smazání hodnocení u písniček/zpěvníků
- zobrazení seznamu komentářů písniček/zpěvníků
- vytvoření komentáře u písniček/zpěvníků
- editování komentáře u písniček/zpěvníků
- smazání komentáře u písniček/zpěvníků
- přebírání písně/zpěvníku
- zrušení přebírání písně/zpěvníku
- zahození staré verze písně
- kopírování písně/zpěvníku
- zobrazení seznamu notifikací
- označení všech notifikací jako přečtené
- označení konkrétní notifikace jako přečtené
- zobrazení seznamu písní vyhovujících přání
- upravení pořadí písní ve zpěvníku

5.4 Komentáře

5.4.1 Zobrazení komentářů k písničce na webu

Aby se zobrazily komentáře uživateli ke specifikované písničce, tak musí nejprve prezentační vrstva zavolat business vrstvu, konkrétně třídu CommentManager, která zavolá datovou vrstvu. CommentDAO se podle parametru type rozhodne, zda-li má zavolat třídu, která se stará o databázové metody, či má zavolat třídu, která má na starosti REST k webové aplikaci. V RESTu třída pošle restRequest, který vrátí odpověď typu JSONArray, z kterého je nutné získat datový objekt Comment. Ten se získá pomocí parsovače. Dále datová vrstva vrátí list s komentáři k dané písničce, který si business vrtsva převede na list s business objekty, které předá prezentační vrstvě.

Kapitola 6

Testování

Poslední důležitou kapitolou je testování, které slouží k ověření, že vše, co jsme naimplementovali, funguje bez problémů. To znamená, že nám mobilní aplikace vrací to, co očekáváme, a že nikde nenastala chyba.

6.1 Unit testy

Unit testy slouží k otestování dílčích částí programu. V testování jsem se nejvíce soustředila na nové funkcionality, které se týkají komentářů a hodnocení. V rámci datové vrstvy testuji, že databáze a webová aplikace vrací správné hodnoty. V business vrstvě testuji správnost konverze z datového objektu na business objekt. Abych získala přehled o tom, kolik jsme metod pokryla testováním, využívám JaCoCo knihovnu.

6.1.1 Komentáře - databáze

V rámci komentářů se věnuji testování všech metod na datové vrstvě, které jsem naimplementovala. Jedná se o metody: addComment, getComments, deleteComment a editComment. Pro otestování databázové metody si nejprve naplním testovací databázi daty, abych mohla otestovat jednotlivé metody. Dále jen zavolám metodu konkrétní třídy, která má na starosti databázové rozhraní ke komentářům. Níže můžete vidět kód pro testování metody getComments.

```
public void testGetCommentsDB() throws
    InterruptedException, ExecutionException,
    JSONException
{

   List<DComment> comments = dbCommentDAO.getComments(
      songID, isSong);
```

```
    for (DComment dComment : comments)
     {
      assertEquals(dComment.getID(), ID);
      assertEquals(dComment.getAuthor(), authorDB);
      assertEquals(dComment.getComment(), commentDB);
      assertEquals(dComment.getDateOfCreation(),
          dateOfCreationDB);
      assertEquals(dComment.getLastChange(),
          lastChangeDB);
     }
  }
```

Na obr. 6.1 můžeme vidět, že jsem ve všech vyjmenovaných metodách dosáhla hodnot pohybujících se mezi 80 až 100 %. Je to proto, že netestuji komentáře, které se vážou ke zpěvníkům.

DBCommentDAO

Element	Missed Instructions	Cov.	Missed Branches	Cov.
editComment(int, DComment, Boolean)		80%		50%
getComments(int, Boolean)		84%		50%
addComment(int, DComment, Boolean)		92%		50%
deleteComment(int)		100%		n/a
DBCommentDAO()		100%		n/a
setTestingDB()		100%		n/a
finalize()		100%		n/a
Total	31 of 258	88%	3 of 6	50%

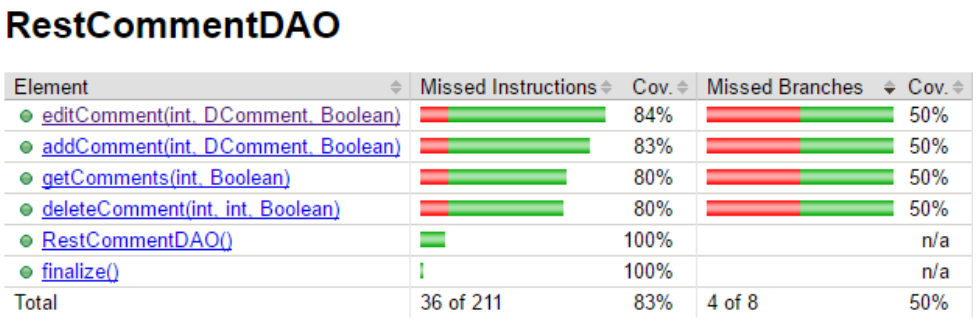
Obrázek 6.1: Pokrytí kódu v rámci databázových metod ke komentářům

6.1.2 Komentáře - REST

Samé metody podrobuji testování, která se týkají webové aplikace. Opět se ujišťuji, zda mobilní aplikace správně komunikuje s webovou aplikací. Nejprve si vytvořím komentář, který následně odešlu webové aplikaci. Tím získám skutečné ID odeslaného komentáře, které využiji během testování editace a mazání. Testováním pokrývám 80 až 84 % kódu, který se týká REST metod ke komentářům. Je to opět způsobeno tím, že netestuji komentování zpěvníků. Výsledky si můžeme prohledout na obr. 6.2.

6.1.3 Konverze z datového objektu na business

V business vrstvě testuji, zda se datový objekt, který dostane Manager, je schopen se převést na business objekt. Každý business objekt má v sobě implementovanou metodu convert, díky které je schopen konverze z datového



Obrázek 6.2: Pokrytí kódu v rámci RESTových metod ke komentářům

objektu na business. V testování pak kontroluji, zda se jednotlivé atributy datového objektu shodují s atributy právě převedeného business objektu.

```
public void testConvertingDComment()
{
    int ID = 3;
    String comment = "Super pisnicka.";
    String author = "sojka";
    Calendar dateOfCreation = Calendar.getInstance();
    Calendar lastChange = Calendar.getInstance();

    DComment dComment = new DComment(ID, comment, author
        , dateOfCreation, lastChange);

    Comment convertedComment = Comment.convert(
        dComment);

    assertEquals(dComment.getID(),
        convertedComment.getID());
    assertEquals(dComment.getComment(),
        convertedComment.getComment());
    assertEquals(dComment.getAuthor(),
        convertedComment.getAuthor());
    assertEquals(dComment.getDateOfCreation(),
        convertedComment.getDateOfCreation());
    assertEquals(dComment.getLastChange(),
        convertedComment.getLastChange());
}
```

6.2 Uživatelské testy

Pomocí uživatelský testů si samotný uživatel aplikace může ověřit, jestli aplikace běží v pořádku. Uživatel následuje jednotlivé scénáře, díky kterým si zkontroluje, zda vše funguje tak, jak má.

6.2.1 Přidání a mazání komentáře u písničky

1. Zobrazte si seznam písniček.
2. Klikněte na konkrétní písničku.
3. Vyplňte komentář.
4. Přidejte k ní komentář.
5. Ověřte si, že písnička obsahuje nově přidaný komentář.
6. Odeberte komentář, který jste přidali.
7. Ověřte si, že se již komentář nezobrazuje u písničky.

6.2.2 Editování komentáře u písničky

1. Zobrazte si seznam písniček.
2. Klikněte na konkrétní písničku.
3. Vyplňte komentář.
4. Přidejte k ní komentář.
5. Ověřte si, že písnička obsahuje nově přidaný komentář.
6. Klikněte na editaci komentáře.
7. Editujte komentář, který jste přidali.
8. Ověřte si, že se zobrazuje editovaný komentář u písničky.

6.2.3 Přidání a mazání hodnocení u písničky

1. Zobrazte si seznam veřejných písniček.
2. Klikněte na konkrétní písničku, kterou jste nevytvořil.
3. Zkontrolujte, že konkrétní písničku jste již nehodnotil.
4. Přidejte k ní hodnocení, vyplňte text a počet hvězdiček
5. Ověřte si, že písnička obsahuje nově přidané hodnocení.

6. Odeberte hodnocení, které jste právě přidali.
7. Ověřte si, že se již hodnocení nezobrazuje u písničky.

Závěr

Hlavním cílem bakalářské práce bylo vytvoření aplikace a analýza nových funkčních požadavků, o které se měla aplikace rozšířit. Součástí práce byla i implementace těchto požadavků a následné otestování.

Vytvořená aplikace splňuje vše, co jsem si ve své práci stanovila.

V budoucnosti by bylo možné aplikaci rozšířit o další funkcionality, které bude nabízet webová aplikace, se kterou Kytarový zpěvník komunikuje.

Literatura

[1] SQLite: *SQLite About [online]*. [cit. 2016-04-28]. Dostupné z: `https://www.sqlite.org/about.html`

[2] Google: *Android Developers - Dashboards [online]*. [cit. 2016-04-24]. Dostupné z: `http://developer.android.com/about/dashboards/index.html`

[3] Hovorka, K.: Zpěvník. [cit. 2016-04-29]. Dostupné z: `https://play.google.com/store/apps/details?id=eu.karelhovorka.zpevnik`

[4] Suslov, A.: Guitar Songs. [cit. 2016-04-29]. Dostupné z: `https://play.google.com/store/apps/details?id=ru.subprogram.guitarsongs.vint`

[5] Miraf: Miraf Songbook. [cit. 2016-04-29]. Dostupné z: `https://play.google.com/store/apps/details?id=cz.miraf.mirafsongbook01`

[6] David Gourley, M. S., Brian Totty: *HTTP: The Definitive Guide*. O'Reilly Media, třetí vydání, ISBN 978-1-56592-509-0.

[7] Google: *Android Developers - Volley [online]*. [cit. 2016-05-4]. Dostupné z: `https://developer.android.com/training/volley/index.html`

Příloha **A**

Seznam použitých zkratek

REST Representational State Transfer

HTTP Hyper Text Transfer Protokol

JSON JavaScript Object Notation

JDK Java SE Development Kit

JaCoCo Java Code Coverage

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `document` ..... dokumentace
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF