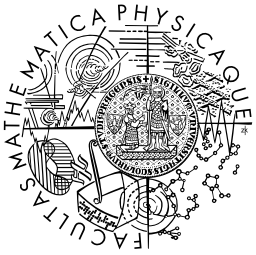


MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Matěj Mezera

Aplikace na vytváření kytarového zpěvníku z písní dostupných na webu

Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce: Mgr. Filip Kliber

Studijní program: Informatika

Praha 2024

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Chtěl bych poděkovat Mgr. Filipu Kliberovi za odborné vedení práce. Dále děkuji Mgr. Zuzaně Soudkové za pomoc při gramatické kontrole práce.

Název práce: Aplikace na vytváření kytarového zpěvníku z písní dostupných na webu

Autor: Matěj Mezera

Katedra: Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce: Mgr. Filip Kliber, Katedra distribuovaných a spolehlivých systémů

Abstrakt: Kytarový zpěvník je seznam textů písní s akordy. Většina kytaristů si vytváří vlastní zpěvníky, přičemž existující programy pro tvorbu kytarového zpěvníku nemusí pokrývat většinu funkcí, které od nich uživatel očekává. V této práci se zabýváme návrhem a implementací nové aplikace, která slouží k tvorbě kytarového zpěvníku. Aplikace se od ostatních liší tím, že umožňuje přidávat do zpěvníku písně dostupné na internetu. Písně se z webových stránek přidávají do zpěvníku pomocí techniky web scraping. Dalšími funkcemi aplikace jsou například úprava existujících písní, uložení zpěvníku pro pozdější úpravu či přidávání vlastních písní do zpěvníku. Výsledný zpěvník lze uložit v několika formátech, především ve formátu PDF.

Klíčová slova: zpěvník, aplikace, web, scraping

Title: Application for creating guitar songbooks from songs available on the web

Author: Matěj Mezera

Department: Department of Distributed and Dependable Systems

Supervisor: Mgr. Filip Kliber, Department of Distributed and Dependable Systems

Abstract: The guitar songbook is a list of song lyrics with chords. Most guitarists create their own songbooks, and the existing programs for guitar songbook creation may not cover most of the features that the user expects from them. In this thesis we are focusing on design and implementation of a new application for guitar songbook creation. The application differs from others in that it allows you to add songs available on the Internet to the songbook. Songs from websites are added to the songbook using the web scraping technique. Other functions of the application are for example editing existing songs, saving the songbook for later editing or adding your own songs to the songbook. The resulting songbook can be saved in several formats, especially in PDF format.

Keywords: songbook, application, web, scraping

Obsah

1 Úvod

1.1 Motivace

Stejně tak jako každý kytarista, jsem i já dospěl k rozhodnutí vytvořit si vlastní zpěvník v tištěné formě. Vzhledem k množství a různorodosti písní, které bych chtěl mít ve svém zpěvníku, mi bylo jasné, že jeho tvorba nebude jednoduchá. Při přemýšlení, jak zpěvník vytvořit, jsem ihned zavrhl metodu kopírování písní z již existujících tištěných zpěvníků, jelikož je téměř nereálné sehnat zpěvníky se všemi písněmi, o které mám zájem. Stahování písní z webu a jejich následné tištění na papír je časově velmi náročné, písně jsou navíc v různých textových formátech a takto vytvořený vlastní zpěvník by nebyl úplně pohledný. Proto jsem se rozhodl pro jiný přístup, a to vytvořit vlastní aplikaci, která by co nejvíce ulehčila a zkrátila tvorbu kytarového zpěvníku. Zároveň by tato aplikace sjednotila formátování písní. Navíc nově vytvořený zpěvník a jednotlivé písně by bylo možné v této aplikaci dále upravovat.

1.2 Pojmy z hudební nauky

Abychom mohli pracovat s pojmem „kytarový zpěvník“, je nutné pro potřebu této práce vysvětlit některé termíny z hudební nauky.

Zvuk je druh mechanického vlnění, konkrétně tlaková vlna. Zvuk se šíří vzduchem, či jiným prostředím jako kmitavý pohyb molekul. Lidské ucho u běžného člověka je schopné vnímat zvuky o frekvencích od 20 Hz do 20 kHz [1]. Zdrojem zvuku je chvění pružných těles, které se přenáší do okolního prostředí a vzbuzuje v něm zvukové vlnění [2]. Zdrojem zvuku je například chvějící se struna kytary, či chvějící se sloupec vzduchu uvnitř píšťaly. **Tóny**, též hudební zvuky, jsou zvuky, jejichž intenzita závisí periodicky na čase [3]. **Melodie**, též melodická linka, je posloupnost jednotlivých tónů daného pořadí, která je zapamatovatelná a rozpoznatelná [4]. **Harmonie** je jev nastávající při souzvuku několika tónů najednou [5]. **Akord** je souzvuk dvou a více různých tónů. Tónina akordu je tón, kterým lze reprezentovat a pojmenovat akord [6]. Existují pravidla pro určení tóniny akordu, ale pro tuto práci je důležité pouze to, že každý akord lze jednoznačně zapsat pomocí akordového zápisu, tzv. akordové značky, ve které je použita tónina akordu. Akordová značka tedy slouží pouze k zaznamenání harmonie písně, nikoliv k zaznamenání její melodické linky. **Transpozice** je posunutí všech tónů, či akordů o určitý interval [7]. Transpozice akordu mění pouze jeho tóninu.

1.3 Kytarový zpěvník v kontextu této práce

Pojem **kytarový zpěvník** bude v této práci znamenat soubor textů písní s příslušným akordovým zápisem jejich harmonie. Kytarový zpěvník slouží kytaristům, kteří vytvářejí harmonický a rytmický doprovod pro melodickou linku ve formě zpěvu nebo hry na jiný hudební nástroj, nikoliv kytaristům, kteří hrají melodickou linku.

1.4 Existující programy

V této sekci si představíme některé existující programy pro tvorbu kytarového zpěvníku. U jednotlivých programů si probereme jejich výhody a především jejich nevýhody, které jsou motivací pro tvorbu nové aplikace.

1.4.1 SongBook

SongBook [8] je multiplatformní offline aplikace od firmy LinkeSOFT sloužící k tvorbě zpěvníku. Aplikace je navržená pro operační systémy Windows, macOS, Android i IOS. V aplikaci lze prohlížet jednotlivé písně, které jsou organizovány do kategorií. Z těchto písní si lze vytvářet vlastní zpěvníky. Existující zpěvník je možné upravovat přidáváním, či odebíráním písní a jejich řazením. Písně lze upravovat pomocí transpozice, změny textu, či přidání a odebrání akordu. Danou píseň lze také zobrazit ve formátu PDF. Aplikace umožňuje zobrazit grafické znázornění jak hrát daný akord, a to pro několik různých hudebních nástrojů. Zároveň lze akordy upravovat, či si nadefinovat vlastní akord. Mezi další funkcionalitu aplikace *SongBook* patří například automatické přibližování a oddalování textu písně v závislosti na její délce, či automatické scrollování s možností nastavení doby scrollování a tempa.

Aplikace *SongBook* má mnoho výhod. Patří mezi ně hlavně její funkcionalita a fungování v režimu offline. Za nevýhody této aplikace lze považovat, že je placená, nelze v ní prohlížet libovolnou skladbu dohledatelnou na internetu a především to, že nevytváří zpěvník ve formátu dokumentu.

1.4.2 Písničky akordy

Písničky akordy [9] je webová aplikace, která umožňuje prohlížení velkého množství písní. Ty lze vyhledávat pomocí autora písně, názvu písně, či alba, ve kterém byla píseň vydána. Písně jsou zobrazovány ve formě textu s akordy nad jednotlivými řádky textu. Při kliknutí na akord se zobrazí zápis, který graficky popisuje, jak hrát daný akord na kytaře. Zobrazenou píseň lze transponovat z původní tóniny do jiné, a to kliknutím na jedno z tlačítek s tóninami, která se nacházejí nad písní. Aplikace také podporuje tisk dané písně, či její export ve formátu PDF. Po vytvoření uživatelského účtu a přihlášení do aplikace si lze vytvářet vlastní zpěvníky a stahovat je ve formátu PDF.

Mezi výhody této aplikace patří určitě její jednoduchost a snadné ovládání. Za nevýhody lze považovat prohlížení pouze těch písní, které aplikace podporuje, stahování zpěvníku až po přihlášení a také to, že nelze vytvářet a prohlížet vlastní píseň, či jinak upravovat již existující píseň.

1.4.3 Ostatní aplikace

Aplikací pro kytaristy existuje mnoho. Nejčastěji se jedná o webové a mobilní aplikace, které slouží k prohlížení písní s akordy. Takovým příkladem, kromě výše popisovaných, jsou webová aplikace *Supermusic* a mobilní aplikace *Songbook.pro*. Většině těchto aplikací ovšem chybí funkce pro úpravy písní a možnost přidávat do zpěvníku vlastní písně nebo písně, které jsou dostupné na internetu. Obvykle

aplikace slouží pouze k prohlížení písní, nikoliv k tvorbě kytarového zpěvníku ve formátu vhodném pro tisk.

1.5 Cíle práce

Hlavním cílem práce je vytvořit aplikaci pro tvorbu vlastního kytarového zpěvníku. Aplikace bude umět načíst písně z webových stránek, vložit je do zpěvníku a následně je upravovat. Aplikace dále bude podporovat přidávání vlastních písní do zpěvníku. Zpěvník bude možné uložit do paměti pro pozdější úpravy nebo jej uložit do zařízení ve formátu vhodném pro tisk.

1.6 Struktura práce

Ve druhé kapitole jsou představeny hlavní problémy při vytváření aplikace a jejich možná řešení. Kapitola 3 popisuje architekturu aplikace a implementaci jejích jednotlivých částí a stěžejní funkcionality. Čtvrtá kapitola obsahuje uživatelskou dokumentaci aplikace.

2 Analýza řešení

2.1 Alternativní implementace

Jedním z prvních problémů při tvorbě libovolné aplikace je volba její platformy, tedy je potřeba určit, zdali to bude webová, mobilní, či desktopová aplikace [10].

Výběr platformy zásadně ovlivňuje i výběr technologií především pro tvorbu frontendu aplikace, ale i jejího backendu. Frontend aplikace slouží pro komunikaci uživatele s interní logikou aplikace prostřednictvím různých tlačítek, textboxů apod. Backend aplikace zahrnuje interní funkcionalitu a logiku aplikace, ukládání dat a práci s nimi, přípravu dat pro uživatele apod.

2.1.1 Mobilní aplikace

Mobilní aplikace je program instalovatelný na chytré telefony a tablety. V současné době jsou dominantními operačními systémy pro mobilní zařízení iOS a Android.

Výhody mobilních aplikací

Mezi výhody mobilních aplikací patří používání funkcí mobilních zařízení, jako je například vytváření notifikací, či vibrace zařízení. Dále tyto aplikace mohou používat mnoho speciálních pohybů zařízením pro interakci s uživatelem, například dotek displeje, zatřesení zařízením, přejetí ze strany na stranu po displeji apod. Mobilní aplikace jsou také snadno dohledatelné v oficiálních obchodech s aplikacemi.

Nevýhody mobilních aplikací

Nevýhodou mobilních aplikací je například nutnost vytvořit dvě aplikace pro iOS a pro Android, či jiným způsobem vyřešit, aby aplikace fungovala na obou operačních systémech. Další nevýhodou je to, že pokud chceme, aby aplikace byla dostupná z oficiálního obchodu s mobilními aplikacemi (*Google Play* pro Android a *App Store* pro iOS), je nutné, aby splňovala podmínky, které si daná služba určuje. Vystavení aplikací na tyto služby je navíc zpoplatněné a při vydání nové verze je nejprve nutné potvrzení verze danou službou, což prodlužuje dobu mezi tím, kdy se nová verze vytvoří a tím, kdy se dostane k uživatelům.

2.1.2 Webová aplikace

Webová aplikace je program běžící na webovém serveru dostupný z webového prohlížeče. Technologie používané při tvorbě frontendu webových aplikací jsou HTML, CSS a JavaScript, které jsou obvykle používány s knihovnami, či frameworky, jako například React, Angular apod.

Výhody webových aplikací

Mezi hlavní výhody webových aplikací patří to, že jsou snadno dostupné skrze webové prohlížeče, a tedy není potřeba instalovat aplikaci do zařízení, z čehož vyplývá, že webové aplikace jsou multiplatformní. S tím je spojené i to, že update na novou verzi se provede automaticky a uživatel nemusí aplikaci aktualizovat. Další výhodou je, že webové aplikace nezabírají paměť zařízení a nemají vysoké nároky na jeho hardware.

Nevýhody webových aplikací

Mezi nevýhody webových aplikací patří to, že velká část funkcionality aplikace potřebuje připojení k internetu. Je-li aplikace nahraná na webovém serveru, pak je pro spuštění aplikace potřeba mít připojení k tomuto serveru. Další nevýhodou je například, že v některých webových prohlížečích se může aplikace zobrazovat, či fungovat nesprávně.

2.1.3 Desktopová aplikace

Desktopové aplikace jsou programy vyvinuté pro počítače. V současné době jsou nejpoužívanější počítače s operačními systémy Windows, Linux a MacOS.

Výhody desktopových aplikací

Mezi výhody desktopových aplikací patří například používání funkcionality operačních systémů, jmenovitě snadný přístup k souborům uloženým v paměti zařízení, interakce se systémovými nástroji apod.

Nevýhody desktopových aplikací

Mezi nevýhody desktopových aplikací patří rozdílný hardware a software počítačů, a tedy na některých počítačích nemusí aplikace běžet tak rychle jako na jiných a na některých se dokonce nemusí spustit. Další nevýhodou je nutnost vyřešit problém rozdílných operačních systémů počítačů, pro které bude aplikace vyvinuta.

2.1.4 Volba platformy a jazyka aplikace

Při uvažování nad platformou aplikace jsem se rozhodl, že cílovým zařízením aplikace nebude mobilní zařízení. Toto rozhodnutí jsem udělal z důvodu, že primárním účelem mobilní aplikace by, dle mého názoru, neměla být tvorba dokumentu vhodného pro tisk. Webová aplikace má hlavní nevýhodu v tom, že sebemenší úprava existujícího zpěvníku, jakou je například změna textu písně, či transpozice písně, by vyžadovala připojení k internetu. Z těchto důvodů jsem se rozhodl, že výsledná aplikace bude desktopová, která bude fungovat i v režimu offline, ovšem bez funkcionality přidání písně z webu. Tato funkcionalita bude dostupná pouze, pokud bude zařízení, na kterém aplikace běží, připojeno k internetu. Aplikace bude psána v jazyce C# za použití vhodných .NET knihoven. Grafické uživatelské rozhraní (frontend) aplikace bude vytvořený pomocí frameworku WPF.

Jazyk C#

C# [11] je objektově orientovaný programovací jazyk. Součástí jazyku C# je deaultní Garbage Collector, který se stará o mazání nedosažitelných objektů z paměti. Zdrojový kód jazyka C# je v prvním kroku zkompilován do Intermediate Language (IL) mezikódu, který se uloží v podobě assembly. Při spuštění kódu se assembly načte do Common Launguage Runtime (CLR), což je virtuální prostředí .NETu sloužící ke spuštění kódu. CLR provede Just-In-Time (JIT) kompilaci, která převede mezikód IL do strojového kódu, který se následně vykoná.

WPF

WPF [12] je framework pro tvorbu uživatelského rozhraní. WPF používá vektorový vykreslovací modul navržený tak, aby využíval moderní grafický hardware. Pomocí WPF lze vytvářet ovládací prvky, 2D i 3D objekty, animace a mnoho dalšího. Uživatelské rozhraní se vytváří za pomocí jazyka XAML. WPF je součástí rozhraní .NET a pro jeho tvorbu existuje v programu Visual Studio textový i grafický editor.

Volba jazyka aplikace

Jako jazyk aplikace jsem se rozhodl zvolit jazyk C#. Důvodem je, že C# je můj oblíbený jazyk a mám s ním největší zkušenosti, nikoliv protože by byl vhodnějším, nebo dokonce nejvhodnějším jazykem pro tvorbu této aplikace. Aplikaci by šlo stejně dobře napsat v jakémkoliv jiném programovacím jazyce, jako je například *Python*, *Java*, či *C++*.

Volba frameworku pro tvorbu uživatelského rozhraní

Uživatelské rozhraní bude vytvořeno pomocí frameworku WPF. Framework musí být kompatibilní s jazykem C#, a proto jsem volil mezi frameworky, které jsou součástí rozhraní .NET. Framework WPF jsem zvolil na základě svých subjektivních preferencí, nikoliv protože by byl pro tvorbu uživatelského rozhraní aplikace vhodnější než například framework *WinForms*, či *MAUI*.

2.2 Formát výstupu zpěvníku

Výstupem aplikace bude kytarový zpěvník, je však potřeba zvolit vhodný formát výstupu. Vzhledem k tomu, že jednou z motivací práce je vytvoření programu pro tvorbu kytarového zpěvníku do tištěné formy, je potřeba, aby formát výstupu byl vhodný pro tisk. Asi nejznámějšími formáty pro tisk souboru jsou formáty PDF a DOC i s jeho pozdější verzí DOCX.

2.2.1 PDF

PDF [13], neboli Portable Document Format, je formát dokumentu vytvořený firmou Adobe v 90. letech 20. století. Cílem bylo vytvořit jednotný formát, pro reprezentaci dokumentů, který bude nezávislý na hardwaru a softwaru zařízení. Do formátu PDF lze ukládat texty, obrázky, odkazy, formuláře a mnoho dalšího.

PDF soubor je posloupnost bytů, které se seskupují do tzv. tokenů popisujících jednotlivé objekty reprezentující uložená data v souboru. Každý PDF dokument se skládá z tzv. PDF hlavičky, těla souboru, cross-referenční tabulky a PDF patičky. Hlavička PDF dokumentu začíná unikátním identifikátorem souboru a verzí formátu PDF. Tělo souboru se skládá ze sekvencí bytů reprezentujících komponenty dokumentu. Cross-referenční tabulka obsahuje informace umožňující přístup k jednotlivým objektům bez čtení celého PDF souboru. PDF patička zajišťuje rychlé hledání cross-referenční tabulky a jiných speciálních objektů. Z tohoto důvodu by programy měly číst PDF soubory od konce.

2.2.2 DOC/DOCX

DOC [14] je formát souborů generovaných programem Microsoft Word nebo jiných textových dokumentů uložených jako jeden binární soubor. Soubor DOC může obsahovat texty, obrázky, různé druhy grafů a mnoho dalšího. Primárně soubor DOC sloužil pro tvorbu dokumentací, specifikací, manuálů a obdobných dokumentů. V průběhu let z DOC formátu vznikl nový formát DOCX.

DOCX [15] je formát Microsoft Word dokumentů, který byl představen v roce 2007 jako součást Microsoft Office. Zpočátku byl DOCX čistě binární soubor s daty, ale nyní se skládá z několika XML souborů uzavřených v ZIP adresáři. Soubory XML jsou rozděleny do dvou kategorií:

- XML soubory obsahující metadata dokumentu
- XML soubory s obsahem dokumentu

Nevýhodou formátu DOC a DOCX je, že jsou vytvořené společností Microsoft, tedy jsou primárně navržené pro počítače s operačními systémy od společnosti Microsoft. Například na operační systém Linux nelze nainstalovat program Microsoft Word, a proto je zobrazení DOC a DOCX složitější. Další nevýhodou těchto formátů je, že program Microsoft Word i jeho online verze jsou zpoplatněné.

2.3 Tvorba PDF

Vzhledem k tomu, že PDF formát lze snadno otevřít na téměř každém zařízení, rozhodl jsem se, že výstupem programu bude kytarový zpěvník ve formátu PDF. Vytvořit PDF soubor lze pomocí mnoha různých metod, které jsou použitelné pro libovolný programovací jazyk, či metod specifických pro jazyk C# a jiné .NET jazyky.

2.3.1 TeX

TeX [16] je jazyk pro vytváření dokumentů složený ze značkovacích a programovacích funkcí, který vytvořil Donald Knuth ze Stanfordské Univerzity. Výhodami TeXu je, že umí formátovat i složité matematické vzorce a je nezávislý na operačním systému. Formátování dokumentu se provádí snadno pomocí příkazů začínajících zpětným lomítkem.

pdfTeX [17] je rozšíření TeXu pro tvorbu PDF souboru přímo z TeXového zdrojového souboru. pdfTeX vytváří PDF soubor nebo soubor *.dvi*.

Výhodami použití jazyku TeX s rozšířením pdfTeX je formátování, které ale nebude u kytarového zpěvníku tak náročné. Nevýhodami jsou tvorba TeX dokumentu z kódu ve správné syntaxi a následné vytvoření PDF a jeho uložení.

Vzhledem k tomu, že kytarový zpěvník nevyžaduje žádný speciální formát odsazování a jiné složité úpravy, přijde mi použití TeX v kombinaci s pdfTeXem nevýhodné. Jako lepší varianta se mi jeví použití nějaké vhodné .NET knihovny pro tvorbu PDF.

2.3.2 PDFShrap

`PDFSharp` [18] je open sourceová .NET knihovna pro tvorbu a zpracování PDF dokumentů za běhu aplikace. Knihovna `PDFSharp` je kompatibilní se všemi .NET jazyky a pracuje s PDF soubory pomocí jednoduchých funkcí pro zapsání textu, vykreslení obrazce apod., ale neumí například automaticky zalamovat řádky.

Výhodou použití knihovny `PDFSharp` je její jednoduchost a snadné používání. Nevýhodou je formátování souboru PDF, které není tak pohledné a snadné, jako při použití jazyka TeX. Pro tvorbu kytarového zpěvníku mi osobně přijde knihovna `PDFSharp` dostačující a vhodná vzhledem k výběru jazyka pro backend aplikace (jazyk C#).

2.3.3 Jiné knihovny .NET

V .Net existuje mimo knihovny `PDFSharp` mnoho dalších, jak open sourceových, tak placených knihoven pro tvorbu dokumentů v PDF formátu. Jsou jimi například:

- `iTextSharp`
- `Pdfium`
- `Syncfusion PDF`
- `IronPDF`

Každá z těchto knihoven má své výhody i nevýhody a každá se hodí na něco trochu jiného viz [19]. Pro vytvoření kytarového zpěvníku by zajisté byla vhodná každá z těchto knihoven, ale nakonec jsem se rozhodl pro použití knihovny `PDFSharp`.

2.4 Parsování písní

Hlavní funkcionalitou, kterou se bude tato aplikace lišit, je možnost přidávání písní z webu do zpěvníku. Proto je nutné vymyslet, jak bude píseň získána z webu a jak se uloží do paměti aplikace, aby se s ní mohlo dále pracovat. Řešením tohoto problému je vhodná implementace techniky web scraping.

2.4.1 Web scraping

Technika web scraping [20] je proces získávání nestrukturovaných dat z internetu a jejich převedení do strukturovaného formátu, se kterým se lépe pracuje.

Tímto strukturovaným formátem může být například struktura, či třída v programovacím jazyce, formát JSON, formát XML, databázová tabulka apod. Web scraping se dá rozdělit na manuální a automatický.

Manuální web scraping

Manuální web scraping spočívá v ručním kopírování HTML kódu entit, ve kterých jsou uložena data, která chceme získat. Pro parsování malého množství dat je manuální web scraping vhodný, i když je potřeba vždy počítat se vznikem chyby zapříčiněné lidským faktorem. Pro programy načítající opakovaně data z webových stránek nebo načítající velké množství dat je ovšem manuální web scraping nepoužitelný.

Automatický web scraping

Automatický web scraping je založený na použití tzv. scrapovacího softwaru, též web scraper, který se stará o získávání dat z webu. Existuje mnoho metod automatického web scrapingu. Některé z nich si nyní popíšeme a vysvětlíme si jejich výhody a nevýhody.

RESTful API

RESTful API [21] je rozhraní pro dva počítačové systémy sloužící k bezpečnému předávání informací skrze internet. Application programming interface (API) je rozhraní definující pravidla komunikace ostatních programů s aplikací. Representational State Transfer (REST) je styl architektury API vynucující určité chování. Součástí API musí být veřejně dostupná dokumentace použití API. Komunikace klienta se serverem prostřednictvím REST API by měla vypadat takto:

1. Klient pošle dotaz splňující formát srozumitelný pro server popsaný v dokumentaci API.
2. Server ověří, zda má klient právo poslat daný dotaz.
3. Server interně zpracuje daný dotaz. Způsob zpracování dotazu není pro klienta známý.
4. Server pošle klientovi odpověď na dotaz obsahující informaci, zda byl dotaz úspěšně vyhodnocen a v případě, že dotaz byl úspěšně vyhodnocen, jsou součástí odpovědi i data, která si klient prostřednictvím dotazu vyžádal.

REST API dotaz by se měl skládat z těchto částí:

- jednoznačný identifikátor zdroje dat, většinou se používá URL tohoto zdroje
- metoda dotazu - metoda dotazuje je HTTP metoda dotazu, například *GET, POST, PUT, DELETE...*
- HTTP hlavička - HTTP hlavička se skládá z dat poslaných při dotazu a z parametrů sloužících k upřesnění dotazu

Odpověď serveru by se měla skládat ze tří částí:

- řádek s HTTP status kódem
- tělo odpovědi obsahují data dotazovaná klientem
- hlavička odpovědi obsahující metadata odpovědi, jako je například formát dat, kódování textu apod.

Status kód [22] je trojciferné číslo od 100 do 599 určující výsledek vyhodnocení dotazu. První číslice status kódu určuje typ odpovědi a celé číslo určuje konkrétní informaci. Například status kód 101 značí, že dotaz je vyhodnocován a nyní není k dispozici odpověď. Status kódy se podle první číslice dělí následovně:

- odpovědi začínající číslicí jedna sdělují informaci o průběhu zpracovávání dotazu
- odpovědi začínající číslicí dva znamenají, že dotaz byl úspěšně zpracován
- odpovědi začínající číslicí tři znamenají, že dotaz byl přesměrován a čeká se na jeho vyhodnocení
- odpovědi začínající číslicí čtyři znamenají chybu na straně klienta, obvykle způsobenou špatně položeným dotazem
- odpovědi začínající číslicí pět znamenají chybu na straně serveru vzniklou při zpracování dotazu.

REST API umožňuje snadnou rozšiřitelnost programů, jak serveru, tak klienta. Komunikace přes REST API je nezávislá na použití programovacího jazyka, tedy programy klienta a serveru mohou být psány v různých jazycích, ale přesto jsou spolu schopné komunikovat pomocí REST API. Použití REST API pro web scraping spočívá v získávání požadovaných dat pomocí veřejných metod API popsaných v dokumentaci.

Použití REST API by bylo optimální pro stahování písní z internetu, jenže většina webových aplikací pro prohlížení písní REST API nepoužívá. Proto je nutné použít jinou metodu automatického web scrapingu.

DOM

Document Object Model (DOM) [23] je rozhraní pro tvorbu strukturované reprezentace webové stránky ve formě objektů, se kterými se snadněji pracuje. DOM byl vyvinut a publikován organizací World Wide Web Consortium (W3C). Strukturovaná reprezentace dat vzniklá při použití DOM je uložena v tzv. DOM stromu. Tento strom se skládá z vrcholů, což jsou komponenty webové stránky, ze kterých se skládá webový dokument. Vrcholem je například i text tlačítka. Element je vrchol odpovídající HTML tagu. Vrcholy jsou propojené v závislosti na tom, jak jsou rozložené na stránce. Pokud je vrchol uvnitř jiného, bude se v DOM stromu nacházet v jeho podstromu.

DOM umožňuje přístup k libovolnému vrcholu stránky, jeho odstranění, či nahrazení pomocí předem předpřipravených metod. DOM je schopný vyhledat konkrétní HTML element pomocí jeho identifikátoru, či více HTML elementů pomocí tříd, které reprezentuje, či CSS selektoru, který splňují. DOM je také

schopný nalézt všechny potomky vrcholu, všechny přímé potomky, všechny vrcholy splňující CSS selector, či implementující stejné třídy.

Výhodou použití DOM při web scrapingu je jeho snadné používání a také to, že pro mnohé programovací jazyky existují knihovny pro převedení HTML stránky do její reprezentace pomocí DOM stromu.

Nevýhodou je, že pro parsování dat je nutné mít znalost struktury webové stránky. Například je nutné vědět, ve kterém HTML elementu jsou data uložena.

Regulární výrazy

Regulární výraz (Regex) je formule popisující posloupnosti písmen splňujících určitou vlastnost. Vlastností může být například to, že posloupnosti začínají stejným písmenem, obsahují pouze čísla a mnoho dalšího. Regulární výrazy slouží k rychlému vyhledávání těchto posloupností v textu, a proto mohou být použity pro získávání dat z webových stránek.

Výhodou regulárních výrazů je jejich snadné použití a poměrně rychlé vyhledávání v textu stránky. Nevýhodou je, že regexy pro hledání mohou být občas komplikované a při drobné změně, například stylu stránky, se může stát, že regex, který dříve fungoval pro nalezení konkrétních dat, je nyní nefunkční.

Headless Browsers

Headless browser [24] je webový prohlížeč, který nepoužívá uživatelské rozhraní. Tyto prohlížeče například nezobrazují ikony, obrázky, či tlačítka. Headless browsery, stejně jako běžné prohlížeče, umožňují navigaci mezi jednotlivými stránkami, stahování a nahrávání obsahu na stránku a mnoho dalších věcí. Tímto se headless browsery liší od prostého stažení webové stránky, jelikož například pro kliknutí na tlačítko je potřeba staženou stránku otevřít v prohlížeči. Headless browsery jsou schopné získat data z HTML elementů stránky, ale i data, která nejsou viditelná v HTML kódu stránky. Prohlížeče načtou stránku, vyrendrují z ní HTML kód a nad tímto kódem spustí JavaScript. Příkladem headless browserů je *Puppeteer*, či *Selenium*.

Výhodou použití headless browserů při parsování webové stránky je rychlost načtení této stránky. Oproti běžným prohlížečům jsou mnohem rychlejší, jelikož nenačítají prvky uživatelského rozhraní, jejichž vykreslování zabírá obvykle mnoho času.

Nevýhodou použití headless browserů je, že načtení webové stránky zabírá téměř stejné množství paměti jako při použití běžného vyhledávače. Další nevýhodou je, že existují webové stránky, které neumožňují prohlížení jejich obsahu pomocí headless browserů.

2.5 Přidávání nových parserů

Jak již bylo zmíněno, aplikace bude načítat písně z jednotlivých webových stránek. Přirozeně tedy nastává problém, jak jednoduše přidávat parsery pro nové webové stránky. Rozhodně by bylo vhodné, aby přidání nového parseru bylo možné s co nejmenším zásahem do kódu aplikace.

2.5.1 Dependency injection

Dependency injection [25] je technika používaná v programování sloužící k odstranění závislostí mezi objekty. Objekt A je závislý na objektu B (objekt B je závislostí objektu A), pokud objekt A používá metody objektu B. Cílem této techniky je zlepšit rozšiřitelnost kódu takovým způsobem, že bude možné upravit závislosti objektů bez úpravy objektů samotných.

Jak funguje dependency injection?

Máme *službu*, kterou používá *klient*, tedy *klient* je závislý na *službě*. Tuto závislost chceme odstranit. Vytvoříme *rozhraní* definující chování *služby*. Každá *služba*, kterou bude *klient* používat, bude muset implementovat *rozhraní*. Vytvoříme objekt, tzv. *injektor*, který bude sloužit k vytvoření instance *služby* a předání této instance *klientovi*. Tímto se odstraní závislost *klienta* na *službě*. V principu dependecy injektion nemusí používat *rozhraní*, ale jeho použití je ve většině případů vhodné [26].

Využití dependency injection při přidávání nového parseru

V této aplikaci bude vhodné, aby objekt načítající písně do zpěvníku nebyl závislý na jednotlivých parserech webových stránek. Proto použití dependency injection spolu s rozhraním by bylo vhodné nejen pro odstranění závislosti, ale i pro snadné rozšiřování programu o nové parsery.

2.5.2 Plugins

Plugin je software rozšiřující funkcionalitu existujícího programu bez nutnosti úpravy původního kódu. Pomocí pluginů lze například prohlížet videa a fotky, či otvírat soubory přímo ve webovém prohlížeči.

Plugin [27] aplikace komunikuje s aplikací pomocí API a skrze metody API rozšiřuje funkcionalitu původní aplikace.

Výhodou používání pluginů je tedy to, že se snadno nainstalují a nedochází ke změně původního kódu.

3 Implementace

V této kapitole si popíšeme architekturu aplikace, jednotlivé komponenty aplikace a implementaci zajímavých a stěžejních funkcionalit aplikace.

3.1 Architektura aplikace

Aplikace je rozdělena do čtyř komponent, z nichž každá je obsažena v jednom podadresáři.

- Adresář *API* obsahuje knihovny definující nové objekty a rozhraní používaná v aplikaci.
- Adresář *SongbookMaker* obsahuje knihovny specifické pro aplikaci SongbookMaker, tedy například knihovny s implementacemi konkrétních parserů webových stránek.
- Adresář *GUI* obsahuje projekt uživatelského rozhraní aplikace.
- Adresář *Tests* obsahuje projekty s testy k aplikaci.

3.1.1 Závislosti aplikace

Všechny knihovny, které aplikace používá, jsou dostupné z balíčkového systému NuGet. Použitými knihovnami jsou:

- `HtmlAgilityPack`, která se používá při webscrapingu
- `PDFSharp`, která se používá pro tvorbu PDF dokumentu obsahujícího zpěvník
- `Newtsoft.Json`, která se používá pro práci se zpěvníkem ve formátu JSON

3.1.2 Sestavení aplikace

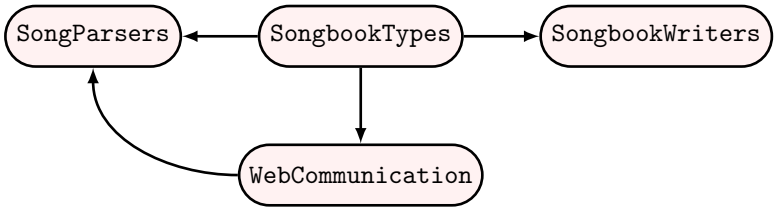
Aplikace byla vyvíjena ve vývojovém prostředí Visual Studio. Cílovým frameworkem aplikace je `net6.0-windows`. Projektem pro spuštění je WPF projekt `GUI`. Aplikace se sestaví do podadresáře adresáře *bin* v projektu `GUI`. Tento výsledný podadresář obsahující sestavenou aplikaci (soubor *SongbookMaker.exe*) bude distribuován uživateli.

3.1.3 Spuštění aplikace

Aplikace je distribuována jako adresář binárních souborů. Pro spuštění aplikace je potřeba spustit soubor *SongbookMaker.exe*. Spuštění tohoto souboru ovšem vyžaduje, aby měl uživatel nainstalovaný .NET verze 6.0, či jiné verze kompatibilní s architekturou aplikace. Aplikaci je možné spustit pouze na počítačích s operačním systémem windows. Aplikace není multiplatformní, jelikož uživatelské rozhraní vytvořené ve framework WPF není multiplatformní.

3.2 API

API je komponenta aplikace obsahující knihovny s datovými typy a rozhraními, které aplikace používá. Tato komponenta je navržena tak, aby mohla být použita pro tvorbu aplikací zabívajících se podobnou problematikou. API obsahuje čtyři knihovny, mezi kterými existují závislosti (obrázek 3.1).



Obrázek 3.1 Závislosti knihoven API

3.2.1 SongbookTypes

`SongbookTypes` je knihovna tříd C# a zároveň jmenný prostor obsahující základní datové typy reprezentující akordy, písně a kytarový zpěvník. Nyní si podrobněji probereme jednotlivé objekty obsažené v tomto jmenném prostoru.

Key

Enum `Key` reprezentuje tóninu akordu, která se používá při tvorbě jeho akordové značky. Různých tónin je celkem dvanáct a jsou to C, Cis, D, Dis, E, F, Fis, G, Gis, A, B, H.

Type

Enum `Type` popisuje „druh“ akordu. Pojmem druh akordu myslíme, zdali je akord durový, mollový, apod. V současné verzi podporuje aplikace tyto druhy akordu:

- Dur (durový)
- Moll (mollový)
- Maj
- Sus
- Add

Chord

Struktura `Chord` slouží k reprezentaci akordu. Struktura má tři vlastnosti:

- `Key` typu `Key`, ve které je uložena tónina akordu.
- `Type` typu `Type`, ve které je uložen „druh“ akordu.

- `ScaleDegree` typu `int`, ve které je uložena informace popisující specifickou vlastnost akordu z hlediska hudební nauky. V aplikaci je nutné mít tuto informaci uloženou pro každý akord, jelikož se používá při tvorbě akordové značky, a to tak, že ve vlastnosti `ScaleDegree` je uloženo číslo uvedené na konci akordové značky. Pokud je ve `ScaleDegree` uložena hodnota 0, akordová značka nekončí žádnou číslicí.

Pro lepší pochopení si ukážeme, jak se v aplikaci uloží „mollový akord A se sedmičkou“, tedy akord s akordovou značkou Ami7. Akord je v tónině A, tedy do vlastnosti `Key` se přiřadí enum `Key` odpovídající tónině A. Akord je mollový, tedy do vlastnosi `Type` se přiřadí enum `Type` odpovídající „druhu“ akrodu Moll. Akord je se sedmičkou, tedy do vlastnosti `ScaleDegree` se přiřadí číslo 7.

Metody třídy Chord

Třída `Chord` obsahuje metodu `Transpose` s návratovou hodnotou typu `Chord` a parametrem typu `int`. Tato metoda vrátí nový akord, který z původního vznikne transpozicí o daný počet půltónů (předán parametrem). Třída obsahuje přepsanou metodu `ToString`, která vrátí textovou reprezentaci akordové značky akordu.

Třída `Chord` má dva konstruktory. Prvnímu konstruktoru se v parametrech předají všechny potřebné objekty, které se uloží do odpovídajících vlastností akordu. Druhý konstruktor vytvoří akord z akordové značky (typu `string`). Při tvorbě akordu z akordové značky se používají tyto metody:

- `GetChordKey`, která obdrží část akordové značky a vrátí příslušnou tóninu akordu
- `GetChordType`, která obdrží část akordové značky a vrátí příslušný druh akordu

Song

Třída `Song` reprezentuje píseň. Tato třída má pět vlastnosti:

- `SongName` typu `string`, ve které je uložen název písně.
- `SongAuthor` typu `string`, ve které je uloženo jméno autora písně.
- `Chords` typu `List<Chord>`, ve které jsou uloženy jednotlivé akordy písně v pořadí, v jakém se nacházejí v písni. Vlastnost `Chords` může obsahovat prázdný seznam v případě, kdy chceme uložit pouze píseň s textem. Uložení akordů do separátního seznamu umožňuje snadnou transpozici písně.
- `Text` typu `List<string>`, ve které jsou uložena jednotlivá slova textu ve stejném pořadí jako se vyskytují v písni. Při vypisování textu se jednotlivá slova oddělují mezerou, proto by se slova měla do vlastnosti `Text` ukládat bez ohraničujících mezer.
- `Capo` typu `int`, ve které je uložena informace na kolikátý pražec kytary se má upevnit kapodastr[1]. Výchozí hodnota vlastnosti `Capo` je 0, což znamená, že kapodastr se neupevňuje na žádný pražec kytary.

[1]kapodastr je zařízení, které se upevní na pražec kytary, a tím zvýší tóninu kytary o stejný počet půltónů jako je pořadové číslo pražce

Třída `Song` používá dvě speciální konstanty typu `string`, které se vkládají do vlastnosti `Text` jako speciální slova a při vypisování jsou nahrazeny speciálními znaky. První konstantou je `NEW_LINE_SIGN` s hodnotou `"$NEW_LINE$"`, která se při vypisování písně nahradí znakem pro zalomení. Druhou konstantou je `CHORD_SIGN` s hodnotou `"$CHORD$"`, která se do seznamu slov textu vloží vždy místo akordu. Při vypisování písně se tato konstanta nahradí odpovídajícím akordem uloženým v seznamu akordů ve vlastnosti `Chords`. Konstanta `NEW_LINE_SIGN` se používá při tvorbě PDF souboru obsahujícího kytarový zpěvník. Konstanta `"$CHORD$"` se používá při převodu zpěvníku do formátu PDF, TXT a v dalších částech aplikace. Pro správnou funkci aplikace je nutné, aby uložené písně splňovaly tento formát.

Metody třídy Song

Třída `Song` obsahuje metodu `Trim`, která odstraní prázdné řádky na začátku i na konci písně. Tato metoda se volá v konstruktoru třídy. Kromě této metody obsahuje třída `Song` dvě přetížení metody `Transpose`. První s parametrem typu `int` transponuje píseň o příslušný počet půltónů (uložený v parametru). Druhé přetížení metody s parametrem typu `Key` transponuje píseň do tóniny uložené v parametru.

Songbook

Třída `Songbook` reprezentuje kytarový zpěvník. Třída má dvě vlastnosti:

- `Title` typu `string`, ve které je uložen název kytarového zpěvníku.
- `Songs` typu `List<Song>`, ve které jsou uloženy instance třídy `Song` reprezentující jednotlivé písně ve stejném pořadí v jakém se budou nacházet ve zpěvníku.

Metody třídy Songbook

Třída `Songbook` má dvě metody pro přidávání písně do zpěvníku. První je metoda `AddSong`, která vloží píseň na konec zpěvníku. Druhou je metoda `AddSongAtPosition`, která vloží píseň do zpěvníku na konkrétní pozici (pozice jsou indexovány od nuly). Dále třída obsahuje tyto metody pro řazení písní:

- `OrderSongsByNameDesc`, která seřadí písně podle názvu sestupně (písně se stejným názvem jsou řazené podle jména autora)
- `OrderSongsByNameAsc`, která seřadí písně podle názvu vzestupně (písně se stejným názvem jsou řazené podle jména autora)
- `OrderSongsByAuthorDesc`, která seřadí písně podle jména autora sestupně (písně se stejným autorem jsou řazené podle názvu)
- `OrderSongsByAuthorAsc`, která seřadí písně podle jména autora vzestupně (písně se stejným autorem jsou řazené podle názvu)

Výjimky

Kromě výše zmiňovaných objektů obsahuje knihovna také tyto výjimky:

- `UnknownTypeException`, která se vyhodí při vytváření nového akordu z akordové značky se špatně zadaným druhem akordu.
- `UnknownKeyException`, která se vyhodí při vytváření akordu z akordové značky se špatně zapsanou tóninou.
- `UnknownChordException`, která se vyhodí, pokud selže vytváření akordu z akordové značky.
- `SongAlreadyInSongbookException`, která se vyhodí, pokud se pokusíme přidat do zpěvníku píseň, kterou již obsahuje.

3.2.2 SongParsers

`SongParsers` je základní knihovna pro práci s parsery písní z webových stránek. Knihovna obsahuje rozhraní `ISongParser`, které musí implementovat každý parser písní z webové stránky použitý v aplikaci. Kromě tohoto rozhraní obsahuje knihovna také statickou třídu `ParsersGetter`, která zajišťuje načtení všech parserů aplikace. Implementaci a použití těchto objektů si podrobně probereme v této kapitole v sekci Parsery písní z webové stránky.

3.2.3 SongbookWriters

`SongbookWriters` je knihovna obsahující rozhraní `ISongbookWriter`, které musí implementovat každý objekt sloužící k ukládání zpěvníku do souboru daného formátu. Dále knihovna obsahuje statickou třídu `ISongbookWriterExtensions` rozšiřující toto rozhraní a statickou třídu `SongbookWritersGetter`, která zajišťuje načtení všech parserů aplikace. Implementaci a použití těchto objektů si podrobně probereme v této kapitole v sekci Formát výstupu.

3.2.4 WebCommunication

`WebCommunication` je knihovna, která zajišťuje získání objektu reprezentujícího DOM (Document Object Model) stránky a usnadňuje práci s tímto objektem. Knihovna využívá objekty a funkce, které jsou součástí knihovny `HtmlAgilityPack`. Knihovna obsahuje tři objekty:

- statickou třídu `WebCommunicator`
- statickou třídu `HtmlNodeExtensions`
- výjimku `ConnectionErrorException`

Knihovnu `WebCommunication` používají parsery, obsažené v současné verzi aplikace, pro převod stránky do DOM. Nové parsery písní z webových stránek mohou používat libovolnou metodu webscrapingu, jelikož rozhraní `ISongParser` nevynucuje použití třídy `WebCommunicator` ani použití knihovny `HtmlAgilityPack`.

WebCommunicator

`WebCommunicator` je statická třída sloužící k získání objektu, který reprezentuje DOM webové stránky. Třída objekt získává pomocí statické metody `GetHtmlDocumentFromUrl`. Tato metoda vytvoří z webové stránky instanci objektu `HtmlDocument`, který je součástí knihovny `HtmlAgilityPack`. Metoda má jeden parametr typu `string`, ve kterém je uloženo URL webové stránky. Objekt `HtmlDocument` reprezentuje DOM webové stránky. Metoda funguje tak, že vytvoří instanci třídy `HtmlWeb`, která je součástí knihovny `HtmlAgilityPack`, a zkusí na ní zavolat metodu `Load`, které jako parametr předá URL webové stránky. Pokud tato metoda uspěje, vrátí instanci třídy `HtmlDocument`, kterou následně vrátí i metoda `GetHtmlDocumentFromUrl`. Pokud neuspěje, vyhodí se výjimka `ConnectionErrorException`.

HtmlNodeExtensions

`HtmlNodeExtensions` je statická třída rozšiřující třídu `HtmlNode`, která je součástí knihovny `HtmlAgilityPack` a reprezentuje DOM stránky. Tato statická třída rozšiřuje třídu `HtmlNode` o následující metody:

- `GetAllElementWithClass`, která vrátí seznam instancí třídy `HtmlNode` reprezentující všechny vrcholy DOM stromu, které jsou atributy daného typu a třídy a jsou potomky původního vrcholu.
- `ContainsAttributeWithClass`, která vrátí informaci, zda existuje vrchol DOM stromu reprezentující atribut daného typu a dané třídy, který je potomkem původního vrcholu.
- `GetFirstElementWithClass`, která vrátí první nalezený vrchol DOM stromu reprezentující atribut daného typu a dané třídy, který je potomkem původního vrcholu.

3.3 SongbookMaker

SongbookMaker je komponenta aplikace obsahující knihovny specifické pro vytvořenou aplikaci. Adresář *SongbookMaker* obsahuje dva speciální podadresáře:

- *OutputWriters* obsahující tři knihovny s objekty, které slouží k uložení zpěvníku do souboru daného formátu. Každá z těchto knihoven obsahuje třídu `SongbookWriter` implementující rozhraní `ISongbookWriter`. Třída `SongbookWriter` reprezentuje objekt sloužící k uložení zpěvníku do souboru.
- *SongParsers* obsahující dvě knihovny s parsery písní. Obě tyto knihovny obsahují jedinou třídu `SongParser` reprezentující parser písní z webové stránky. Třída `SongParser` implementuje rozhraní `ISongParser`. `SongParser` získává data z webové stránky pomocí techniky web scraping, konkrétně pomocí převodu stránky do DOM. Třída převádí stránku do DOM pomocí knihovny `WebCommunicator`. Pro práci s objektem reprezentujícím DOM stránky používá třída knihovnu `HtmlAgilityPack` a její rozšíření obsažená v knihovně `WebCommunicator`.

3.3.1 PDFWriter

`PDFWriter` je knihovna, která slouží k tvorbě a ukládání kytarového zpěvníku ve formátu PDF.

SongbookWriter

Tvorbu kytarového zpěvník ve formátu PDF zajišťuje třída `SongbookWriter`. Třída `SongbookWriter` používá funkce a objekty obsažené v knihovně `PDFSharp`. Soubor PDF je reprezentován objektem `PdfDocument`, který je součástí knihovny `PDFSharp`. Kromě vlastností a metod, které vynucuje rozhraní `ISongbookWriter`, obsahuje třída `SongbookWriter` mnoho metod pro tvorbu PDF souboru a pro práci s ním. Příkladem těchto metod jsou:

- `DrawImage`, která vykreslý obrázek na konkrétní pozici na stránce.
- `Merge`, která sloučí dvě instance třídy `PdfDocument` do jedné.
- `WriteContentsPage`, která z písní a seznamu jejich počátečních stran vytvoří stránku s obsahem.

ColumnState

Součástí knihovny `PDFWriter` je také třída `ColumnState`. Tato třída je používaná při tvorbě obsahu kytarového zpěvníku. Třída má tři vlastnosti:

- `X` typu `int`, ve které je uložena x-ová souřadnice řádku obsahu.
- `BaseY` typu `int`, ve které je uložena vzdálenost prvního řádku od horního okraje stránky. Tato hodnota není uložena v konstantě, jelikož pro první stránku obsahu je hodnota větší než pro stránky ostatní.
- `SecondColumn` typu `bool`, ve které je hodnota `true`, pokud vytváříme obsah do druhého sloupce na stránce.

Tvorba PDF

Tvorbu kytarového zpěvníku ve formátu PDF zajišťuje třída `SongbookWriter`, která je obsažena v knihovně `PDFWriter`. Tato třída používá k tvorbě PDF knihovnu `PDFSharp`, která je dostupná z balíčkového systému NuGet. Tvorba PDF probíhá tak, že se nejprve vytvoří první PDF soubor obsahující titulní stranu. Poté se vytvoří druhý PDF soubor obsahující stránky s jednotlivými písněmi a závěrečnou stránku s akordy. Současně s vytvářením druhého souboru se spočítají offsety písní a závěrečné stránky. Podle těchto offsetů se vytvoří stránka s obsahem v prvním souboru. Dalším krokem je sloučení prvního a druhého souboru do jednoho výsledného souboru PDF. Slučování probíhá tak, že stránky druhého souboru se připojí za stránky prvního souboru. Na závěr se očíslují stránky výsledného souboru. Hlavičky stránek souborů se vytvářejí v jazyce, který byl nastavený v aplikaci v okamžiku tvorby souboru.

Titulní strana podporuje pouze název zpěvníku, který lze vypsat na jeden řádek. Pokud je název zpěvníku moc dlouhý, PDF soubor se nevytvoří a aplikace upozorní uživatele, že se zpěvník ve formátu PDF neuložil z důvodu dlouhého názvu zpěvníku.

Tvorba stránky s písní

Při tvorbě stránky se nejprve vypíše hlavička s údaji o písni a poté se vypíše text s akordy. Text písně se vypisuje po slovech, která se oddělují mezerou. Slova textu se načítají z vlastnosti `Text` třídy `Song`. Pokud je aktuálním slovem řetězec uložený v konstantě `Song.CHORD_SIGN`, vypíše se do horního indexu odpovídající akord (načtený z vlastnosti `Chords` třídy `Song`). Pokud aktuální slovo odpovídá řetězci uloženému v konstantě `Song.NEW_LINE_SIGN`, zalomíme řádek. Během vypisování textu si ukládáme offset konce posledního řádku a pro každé nové slovo překontrolujeme, zda lze vypsat celé na poslední řádek. Pokud ne, zalomíme řádek a pokračujeme na řádku novém. Kromě offsetu konce posledního řádku si udržujeme i offset posledního řádku, abychom v případě potřeby přešli na stránku novou.

Zápis textu do PDF souboru zajišťuje třída `XGraphics` (obsažená v knihovně `PDFSharp`) pomocí metody `DrawString`. Tato metoda nepodporuje zalomení řádku při vypisování řetězce obsahujícího znak nové řádky (`'\n'`). Z tohoto důvodu byla zavedena konstanta `Song.NEW_LINE_SIGN`, pomocí které se snáze určuje, kdy je třeba řádek zalomit.

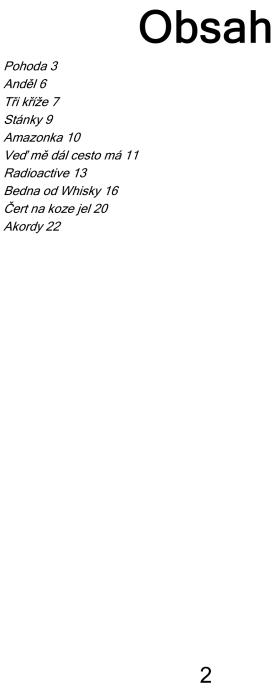
Tvorba obsahu

Obsah se vytváří do dvou sloupců a při jeho tvorbě vypisujeme na řádek pouze název písně. Je-li název písně příliš dlouhý, zkrátíme jej. Při tvorbě obsahu si opět udržujeme offset řádku, abychom věděli, kdy přejít do druhého sloupce a kdy vypisovat obsah na další stránku.

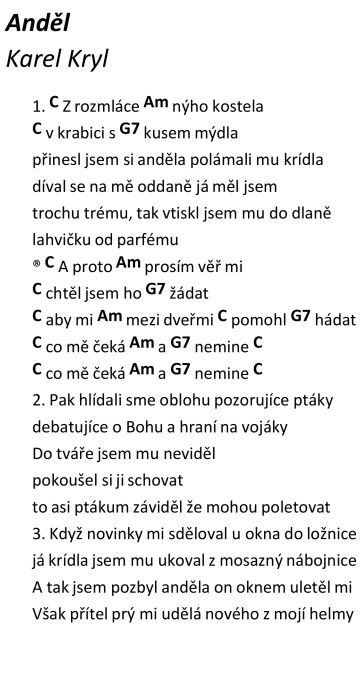
Struktura výsledného PDF souboru

Výsledný soubor PDF obsahující kytarový zpěvník se skládá z titulní strany (obrázek 3.3 (a)), obsahu (obrázek 3.4 (b)), stránek s jednotlivými písněmi a ze závěrečné strany obsahující užitečné kytarové akordy (obrázek 3.6 (d)). Titulní strana obsahuje název zpěvníku a obrázek kytary. Všechny stránky, kromě titulní strany, jsou očíslované. První stránka obsahu (první očíslovaná stránka) má číslo 2. Stránka obsahující konktrétní píseň (obrázek 3.5 (c)) se skládá z hlavičky a textu písně s akordy. Hlavička obsahuje název písně, název autora a informaci, na který pražec se upevňuje kapodastr. Pokud informace o kapodastru není uvedena, píseň se hraje bez použití kapodastru.

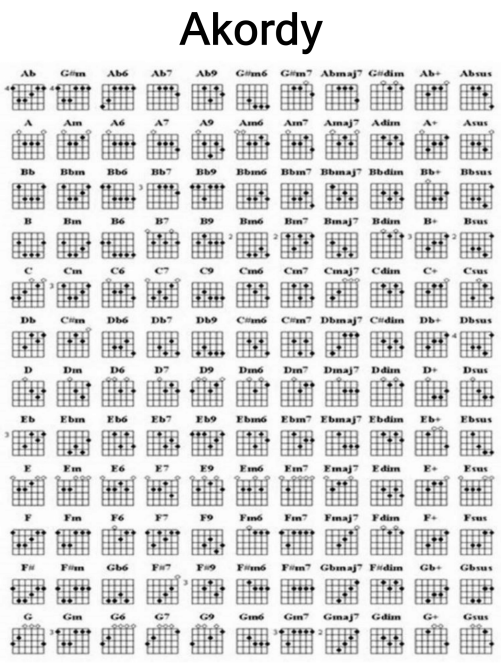
(a) Titulní strana zpěvníku



(b) Strana zpěvníku s obsahem



(c) Strana zpěvníku s písní



(d) Závěrečná strana zpěvníku

Obrázek 3.2 Stránky výsledného PDF souboru

3.3.2 JSONWriter

`JSONWriter` je knihovna, která slouží pro tvorbu a ukládání kytarového zpěvníku ve formátu JSON. Knihovna obsahuje pouze třídu `SongbookWriter`, která má

pouze vlastnosti a metody vynucené rozhraním `ISongbookWriter`. Třída používá pro vytvoření objektu JSON knihovnu `Newtsoft.Json`, konkrétně její statickou třídu `JsonConvert`. Tvorba souboru JSON obsahujícího zpěvník probíhá tak, že na statické třídě `JsonConvert` se zavolá statická metoda `SerializeObject` s parametrem, kterým je objekt reprezentující zpěvník (instance třídy `Songbook`). Metoda vrátí textový řetězec, ve kterém je uložen příslušný JSON. Tento řetězec se uloží do uživatelem zvoleného souboru.

3.3.3 PlainTextWriter

`PlainTextWriter` je knihovna, která slouží pro tvorbu a ukládání kytarového zpěvníku ve formátu TXT. Knihovna obsahuje pouze třídu `SongbookWriter`. Tato třída zajišťuje ukládání kytarového zpěvníku ve formátu TXT. Třída obsahuje, kromě vlastností a metod vynucených rozhraním `ISongbookWriter`, metody pro tvorbu jednotlivých částí souboru TXT. Těmito metodami jsou:

- `SongTextToString` s návratovou hodnotou typu `string` a parametrem typu `Song`
- `SongToString` s návratovou hodnotou typu `string` a parametrem typu `Song`
- `SongbookToString` s návratovou hodnotou typu `string` a parametrem typu `Songbook`

Tvorba TXT souboru

TXT soubor se vytváří tak, že nejprve se převede zpěvník do textového řetězce pomocí metody `SongbookToString` a poté se textový řetězec uloží do souboru pomocí .NET třídy `StreamWriter`. Metoda `SongbookToString` nejdříve vytvoří řetězec obsahující hlavičku zpěvníku s názvem souboru a poté pro každou píseň vytvoří pomocí metody `SongToString` textový řetězec, kterým prodlouží řetězec reprezentující zpěvník. Řetězec reprezentující píseň vytváří metoda `SongToString` tak, že nejprve vytvoří hlavičku písně obsahující název písně, jméno autora a pražec, na který se umístí kapodastr (pokud je zadán). Poté se vytvoří text písně s akordy pomocí metody `SongTextToString`. Metoda `SongTextToString` vytvoří řetězec ze slov textu uložených ve vlastnosti `Text` třídy `Song` tak, že je skládá za sebe a odděluje je mezerou. Stejně jako při tvorbě PDF se slovo, které odpovídá hodnotě uložené v konstantě `Song.CHORD_SIGN` nahradí odpovídajícím akordem (načteným z vlastnosti `Chords` třídy `Song`) a místo slova odpovídajícího hodnotě uložené v konstantě `Song.NEW_LINE_SIGN` se zalomí řádek. Akordy jsou při vypisování ohraničeny hranatými závorkami. Hlavičky zpěvníku a jednotlivých písní se vytvoří v jazyce, který byl nastavený v aplikaci v okamžiku tvorby souboru.

3.3.4 ChordsAndTabsParser

`ChordsAndTabsParser` je knihovna obsahující parser písní z webové stránky *Chords and Tabs* [28]. Třída `SongParser` reprezentující tento parser obsahuje, kromě vlastností a metody vynucených rozhraním `ISongParser`, metody pro načtení DOMu písně a práci s ním. Těmito metodami jsou například:

- `GetSongPageDocument`, která získá objekt reprezentující DOM stránky s písní.
- `GetChordsAndUpdateText`, která vytvoří seznam akordů písně a upraví DOM.
- `ParseSongText`, která získá text písně.

Uložení písně na stránce

Většina písní je uložena na stránce s URL: `https://www.chords-and-tabs.net/song/name/$songAuthor-$songName`, kde `$songAuthor` je jméno autora, `$songName` je název písně, ve kterých jsou oddělovači pomlčky místo mezer. Píseň je na webové stránce obsažena v HTML elementu `<div>` s identifikátorem `snippet-snippetXteView`. Text písně i s akordy je obsažený v několika HTML elementech `<div>` patřících do třídy `song`. Akordová značka každého akordu je obsažena ve vlastním HTML elementu `<sup>`. Uložení písně na stránce bylo získáno vlastní analýzou několika písní, jelikož stránka *Chords and Tabs* neobsahuje informaci o tom, jak jsou písně na stránce uloženy.

Získávání písně z webové stránky

Získávání písně probíhá tak, že nejprve se z názvu písně a jména autora vytvoří URL stránky obsahující píseň. Poté se otestuje, zda stránka s tímto URL obsahuje píseň a pokud ne, pokusíme se píseň na stránce vyhledat, a tím získat nové URL. Jakmile máme URL písně, získáme akordy písně ze všech HTML elementů `<sup>`, které jsou potomky elementu `<div>` s identifikátorem `snippet-snippetXteView`. Současně s tvorbou akordů se upraví objekt obsahující DOM stránky tak, že všechny HTML elementy `<sup>`, ze kterých jsme získali akordy, nahradíme řetězcem uloženým v konstantě `CHORD_SIGN` třídy `Song`. Z vlastnosti `InnerText` upraveného elementu `<div>` s identifikátorem `snippet-snippetXteView` získáme již snadno text písně. Pokud selže získání DOM stránky pomocí knihovny, parser vyhodí výjimku `ConnectionErrorException`. Pokud hledání písně selže z jiného důvodu, parser vyhodí výjimku `SongNotFoundException`.

3.3.5 PisnickyAkordyParser

`PisnickyAkordyParser` je knihovna obsahující parser písní z webové stránky *Písničky Akordy* [9]. Třída `SongParser` reprezentující tento parser obsahuje, kromě vlastností a metody vynucených rozhraním `ISongParser`, metody pro načtení DOMu stránky obsahující píseň a práci s ním. Těmito metodami jsou například:

- `GetSongPageDocument`, která získá objekt reprezentující DOM stránky s písní.
- `GetChordsAndUpdateInnerHtml`, která vytvoří seznam akordů písně z objektu reprezentujícího DOM stránky a upraví tento objekt.
- `ParseSongText`, která získá text písně.

Uložení písně na stránce

Písně jsou uložené na stránce s URL:
`https://pisnicky-akordy.cz/$songAuthor-$songName`, kde stejně jako u předchozího parseru je `$songAuthor` jméno autora a `$songName` název písně, ve kterých jsou oddělovači pomlčky místo mezer. Text písně s akordy je na stránce uložen v HTML elementu `<div>` s identifikátorem `songtext`. V elementu `<el>` je uložený řádek s akordy, které se zobrazují nad následujícím řádkem. Každý akord je uložen v HTML elementu `<a>` (akordová značka je uložena ve vlastnosti `InnerText`), který je přímým potomkem HTML elementu `<span>` třídy `akord`. Stejně jako u předchozího parseru nejsou tyto informace získané z důvěryhodného zdroje, ale jsou získané vlastní analýzou několika písní.

Získávání písně z webové stránky

Získávání písně probíhá tak, že nejprve získáme objekt reprezentující DOM stránky s písní. Poté parser vytvoří seznam akordů písně a všechny HTML elementy `<span>` třídy `akord` nahradí řetězcem uloženým v konstantě `CHORD_SIGN` třídy `Song`. Po tomto nahrazení je ve vlastnosti `InnerText` elementu `<div>` s identifikátorem `songtext` uložen text písně a řádky s akordy (místo jednotlivých akordů je uložena hodnota `Song.CHORD_SIGN`). V případě, že po řádku s akordy následuje neprázný řádek textu, je potřeba tyto dva řádky sloučit do jednoho. Z tohoto důvodu je nutné si pro každý akord předpočítat offset na řádku a jakmile při zpracování řádku s textem dosáhneme offsetu libovolného akordu, zpracujeme tento akord a pokračujeme ve zpracovávání řádku s textem. Akordy, které mají offset vyšší než je délka řádku, uložíme na konec řádku. Pokud selže získání DOM stránky pomocí knihovny `WebCommunicator`, vyhodí parser výjimku `ConnectionErrorException`. Pokud hledání písně selže z jiného důvodu, parser vyhodí výjimku `SongNotFoundException`.

3.3.6 JSONReader

`JSONReader` je knihovna, která slouží k načítání kytarového zpěvníku do aplikace ze souboru formátu JSON. Načtení zpěvníku zajišťuje statická třída `Reader` pomocí statické metoda `LoadSongbook` s návratovou hodnotou typu `Songbook` a s parametrem typu `string`. V parametru metody je uložena absolutní cesta k souboru, ze kterého načítáme zpěvník. Tato metoda funguje tak, že nejprve načte obsah souboru do proměnné typu `string` a poté zavolá statickou generickou metodu `DeserializeObject<Songbook>` na statické třídě `JsonConvert`, která je součástí knihovny `Newtsoft.Json`. Tato metoda se volá s parametrem, kterým je proměnná s obsahem souboru. Metoda vrátí instanci třídy `Songbook`. Na závěr otestujeme, zda zpěvník vrácený metodou je validní. Validace kontroluje, zda metoda nevrátila hodnotu `null` a zda zpěvník vrácený metodou má přiřazený název a seznam písní (může být prázdný).

Kromě třídy `Reader` obsahuje knihovna výjimku `SongbookLoadException`, která se vyhodí, pokud selže validace zpěvníku.

3.4 Uživatelské rozhraní

Uživatelské rozhraní aplikace je obsaženo v adresáři *GUI*. Uživatelské rozhraní je vytvořeno ve frameworku WPF.

3.4.1 Okno uživatelského rozhraní

Uživatelské rozhraní je tvořeno jedním oknem, které je reprezentováno pomocí třídy `MainWindow`, která je potomkem třídy `System.Windows.Window`. Ovládací prvky tohoto okna se překreslují v závislosti na uživatelem vyvolaných událostech.

Rozložení ovládacích prvků okna

Všechny prvky okna jsou potomky prvku `Viewbox`, který zajišťuje změnu velikosti ovládacích prvků při změně velikosti okna. Přímým potomkem tohoto prvku je prvek `Grid`, který zajišťuje rozdělení okna na řádek s hlavičkou okna a na řádek s tělem okna. Hlavička okna je tvořena prvkem `Grid`, který zajišťuje uspořádání prvků hlavičky do čtyř sloupců. Tělo okna je tvořené šesti prvky `Grid`, které obsahují specifické ovládací prvky uspořádané ve mřížce. Vždy pouze jeden z těchto prvků se zobrazuje v okně.

Překreslování těla okna

Hlavička okna je zobrazena po celou dobu používání aplikace, ale ovládací prvky těla okna je třeba překreslovat. Z tohoto důvodu obsahuje uživatelské rozhraní enum `ApplicationState`, který reprezentuje stavy, ve kterých se může aplikace nacházet. V každém z těchto stavů obsahuje tělo okna jiné ovládací prvky. Aplikace se může nacházet v následujících šesti stavech:

- `MainMenu`, ve kterém se zobrazuje hlavní menu aplikace.
- `PreviewSongbook`, ve kterém lze prohlížet a upravovat zpěvník.
- `LoadSongbook`, ve kterém lze načítat zpěvníky uložené v paměti aplikace.
- `ExportSongbook`, ve kterém lze ukládat zpěvník do zařízení jako soubor v určitém formátu, například PDF.
- `AddSong`, ve kterém lze přidat novou píseň do zpěvníku.
- `UpdateSong`, ve kterém lze upravovat píseň obsaženou ve zpěvníku.

Překreslování těla okna probíhá tak, že se vždy zobrazuje prvek `Grid` příslušný aktuálnímu stavu aplikace a ostatní jsou skryté. O změnu stavu aplikace a překreslení okna se stará metoda `SetState`, které se v parametru předá enum `ApplicationState` reprezentující nový stav aplikace. Metoda skryje všechny prvky `Grid` těla okna a zobrazí prvek `Grid` odpovídající novému stavu. Prvek `Grid` se skryje tak, že se do vlastnosti `Visibility` přiřadí enum `Visibility.Hidden` a do vlastnosti `IsEnabled` se přiřadí hodnota `false`. Zobrazení prvku `Grid` v okně se zajistí přiřazením hodnoty `true` do vlastnosti `IsEnabled` a přiřazením enumu `Visibility.Visible` do vlastnosti `Visibility`.

3.4.2 GUI.NewObjects

`GUI.NewObjects` je jmenný prostor obsahující vlastní objekty používané v okně uživatelského rozhraní. Jmenný prostor obsahuje objekty reprezentující nová tlačítka, které jsou potomky třídy `Button`. Těmito objekty jsou:

- třída `ExportButton`, která reprezentuje tlačítko pro uložení souboru obsahujícího zpěvník do zařízení. Toto tlačítko je rozšířené o proměnnou `outputWriter` typu `ISongbookWriter` a readonly vlastnost `FileType` typu `string`. Po stisknutí tlačítka se zobrazí prohlížeč souborů windows, ve kterém si uživatel zvolí, kam chce uložit soubor obsahující kytarový zpěvník a poté se zavolá metoda `SaveSongbook` na proměnné `outputWriter`, která soubor uloží.

- třída `SongUpdateButton`, která reprezentuje tlačítko pro úpravu písně. Toto tlačítko je rozšířené o vlastnost `Row` typu `int`, ve které je uloženo pořadové číslo řádku, na kterém se píseň zobrazuje (při prohlížení zpěvníku). Řádky jsou číslovány od nuly a pořadové číslo řádku je shodné s pořadovým číslem písně ve zpěvníku.

- třída `SongRemoveButton`, která reprezentuje tlačítko pro odebrání písně. Toto tlačítko je rozšířené o vlastnost `Row` typu `int`, ve které je, stejně jako ve vlastnosti `Row` tlačítka `SongUpdateButton`, uloženo pořadové číslo řádku.

- třída `LoadButton`, která reprezentuje tlačítko pro načtení zpěvníku z paměti aplikace. Toto tlačítko je rozšířené o vlastnost `Path` typu `string` obsahující absolutní cestu k souboru (formátu JSON), který obsahuje kytarový zpěvník. Stisknutí tohoto tlačítka načte kytarový zpěvník ze souboru do aplikace.

Kromě objektů reprezentujících nová tlačítka, obsahuje jmenný prostory tyto objekty:

- Enum `ApplicationState`, který reprezentuje stav aplikace, který se využívá při překreslování těla okna

- Třída `SongRowGrid`, která reprezentuje řádek s písní zobrazený při prohlížení zpěvníku. Třída si udržuje pořadové číslo řádku, na kterém je zobrazena (řádky číslujeme od 0) ve vlastnosti `Row` typu `int`. Třída je potomkem třídy `Grid` a obsahuje tři ovládací prvky uspořádané do mřížky. Těmito ovládacími prvky jsou popisek obsahující základní informace o písni, tlačítko `SongUpdateButton` sloužící k úpravě písně a tlačítko `SongRemoveButton` sloužící k odstranění písně ze zpěvníku.

- Třída `LoadSongbookRow`, která reprezentuje řádek, který se zobrazuje pro každý zpěvník uložený v paměti aplikace. Třída obsahuje tři ovládací prvky uspořádané do mřížky. Těmito ovládacími prvky jsou popisek obsahující název zpěvníku, tlačítko `LoadButton` sloužící k načtení zpěvníku a tlačítko sloužící ke smazání zpěvníku z paměti aplikace. Třída si udržuje absolutní cestu k souboru (formátu JSON), který obsahuje kytarový zpěvník, ve vlastnosti `Path` typu `string`.

- Třída `PlaceHolderTextBox`, která reprezentuje textové pole obsahující výchozí text, který po kliknutí do tohoto pole zmizí. Třída je potomkem třídy `TextBox` a je rozšířená o vlastnost `DefaultText` typu `string` obsahující výchozí text. Mizení a objevování původního textu je řešené pomocí události `GetFocus`, která je způsobena kliknutím na textové pole a události `LostFocus`, která je způsobena kliknutím mimo textové pole. Při události `GetFocus` se ověří, zda text v textovém poli odpovídá textu ve vlastnosti `DefaultText` a pokud ano, smaže se obsah textového pole. Při události `LostFocus` se ověří, zda je textové pole prázdné a pokud ano, uloží se do něj text uložený ve vlastnosti `DefaultText`.

3.4.3 Jazyk uživatelského rozhraní

Uživatelské rozhraní aplikace je multijazyčné. Aktuální verze aplikace obsahuje české a anglické uživatelské rozhraní. Jazyk uživatelského rozhraní je možné měnit pomocí tlačítek obsažených v hlavičce okna. Výchozí jazyk aplikace se nastavuje podle hodnoty `CurrentUICulture` aktuálního vlákna. Odpovídá-li hodnota `CurrentUICulture` češtině, nastaví se jazyk uživatelského rozhraní na češtinu, jinak se nastaví na agličtinu.

Vícejazyčnost uživatelského rozhraní je řešená pomocí řetězcových konstant uložených v souborech prostředků. Těmito soubory jsou *Resources.resx* pro anglické konstanty a *Resources.cs.resx* pro české konstanty.

Změna jazyka uživatelského rozhraní se provádí tak, že se do vlastnosti `CurrentUICulture` aktuálního vlákna přiřadí `CultureInfo` odpovídající zvolenému jazyku.

3.4.4 Vlastní řazení písní

Uživatelské rozhraní umožňuje vlastní řazení písní pomocí metody drag and drop. Řazení pomocí metody drag and drop je umožněno pouze tehdy, je-li ve vlastnosti `SelectedItem` objektu `OrderComboBox` (typu `ComboBox`) uložen objekt `OwnOrderComboBoxItem` (typu `ComboBoxItem`). Řazení písní je implementováno tak, že při vyvolání události `MouseMove` na objektu `SongRowGrid` se zjistí, zdali je řazení umožněno. Je-li řazení umožněno, změní se kurzor nad popiskem písně a pokud je navíc stisknuté levé tlačítko myši, vyvolá se na objektu `SongRowGrid` událost `DragOver`. Během události `DragOver` se kontroluje, zdali se přetahovaný objekt neposunul na jiný řádek. Pokud ano, píseň se přesune ve zpěvníku na novou pozici a vytvoří se nové řádky s písněmi, které se zobrazí v okně. Řádky s písněmi jsou v okně součástí objektu `SongsScrollViewer` (typu `ScrollViewer`). Během události `DragOver` se kontroluje, zdali je nutné posunout obsah objektu `SongsScrollViewer` pro zobrazení dalších písní.

3.4.5 Ukládání zpěvníků do paměti aplikace

Aplikace umí uložit zpěvník do paměti tak, aby bylo možné jej později načíst a dále s ním v aplikaci pracovat. Ukládání zpěvníku funguje tak, že se zpěvník převede do formátu JSON, pomocí třídy `SongbookWriter` nacházející se v knihovně `JSONWriter`. Soubor se uloží do adresáře *SavedSongbooks*, který se nachází ve

stejném adresáři jako soubor *SongbookMaker.exe*, pod stejným názvem jako je název zpěvníku, který je v něm uložen. Pokud adresář neexistuje, vytvoří se při ukládání zpěvníku.

Načítání uložených zpěvníků funguje tak, že každý soubor s příponou *.json* nacházející se v adresáři *SavedSongbooks* se aplikace pokusí převést na instanci třídy `Songbook`. Pokud lze soubor převést, vytvoří se v aplikaci instance objektu `LoadSongbookRow` a následně se tato instance zobrazí v okně aplikace.

Správné načtení zpěvníku zajišťuje třída `Reader`, která je součástí knihovny `JSONReader`. Odstranění zpěvníku se provede tak, že se smaže odpovídající soubor v adresáři *SavedSongbooks*.

3.5 Tests

Adresář *Tests* obsahuje testové projekty aplikace. Součástí aktuální verze aplikace jsou dva testové projekty `ChordTests` a `ParsersTests`.

3.5.1 ChordTests

`ChordTests` je testový projekt obsahující Unit testy ke třídě `Chord`. Projekt obsahuje dvě testové třídy:

- `StringConstructorTests` testující správnou tvorbu akordu z akordové značky, kterou zajišťuje konstruktor třídy `Chord` s parametrem typu `string`.
- `ChordTransposeTests` testující správnost transpozici akordu, kterou zajišťuje metoda `Transpose` třídy `Chord`.

3.5.2 ParsersTests

`ParsersTests` je testový projekt obsahující Unit testy k jednotlivým parserům písní z webových stránek. Projekt pro každý parser obsahuje testovou třídu (třída `ChordsAndTabsParserTests` a třída `PisnickyAkordyParserTests`), která testuje, zda načtení existující písně parserem vrátí píseň se správným jménem autora a názvem písně. Třída také kontroluje, zda pro neexistující píseň vyhodí parser správnou výjimku. Třída nekontroluje text písně načtené parserem ani akordy, které se v ní objevují.

3.6 Parsery písní z webové stránky

Hlavní funkcionalitou této aplikace, kterou se liší od ostatních aplikací pro tvorbu kytarového zpěvníku, je možnost přidávat do zpěvníku písně dostupné na webových stránkách. O načtení písně z konkrétní webové stránky do aplikace se stará část aplikace, které v této sekci budeme říkat parser. V této sekci si vysvětlíme, jak je tato funkcionalita zpracována, jak funguje dynamické načítání jednotlivých parserů, jak funguje načítání písně z webu a jak aplikaci rozšířit o nový parser.

3.6.1 ISongParser

`ISongParser` je rozhraní, které je součástí knihovny `SongParsers` a musí jej implementovat každý parser používaný aplikací. Toto rozhraní od každého parseru vynucuje, aby implementoval metodu `GetSong` a měl tyto dvě veřejné vlastnosti:

- `ParserURL` typu `string`, která je readonly a má v sobě uložené URL webové stránky, ze které parser získává písně.
- `Priority` typu `int`, která určuje pořadí, v jakém bude parser použit při vyhledávání konkrétní písně.

Návratová hodnota metody `GetSong` je typu `Song` a reprezentuje píseň získanou z webové stránky. Metoda má dva parametry typu `string`, ve kterých je uloženo jméno autora a název písně. Tato metoda by měla prohledat webovou stránku, ze které parser získává písně a pokud nalezne hledanou píseň, měla by ji vrátit jako instanci třídy `Song`. Pokud hledanou píseň nenalezne, měla by vyhodit výjimku `SongNotFoundException`, což ovšem pomocí rozhraní nelze vynutit.

3.6.2 Dynamické načítání parserů

Aplikace používá parsery písní z webových stránek. Tyto parsery jsou obsaženy v knihovnách a aby s nimi mohla aplikace dále pracovat, je potřeba parsery do aplikace načíst. O dynamické načítání parserů se stará statická třída `ParsersGetter`, která je obsažena v knihovně `SongParsers` a na které existuje veřejná statická metoda `GetAllSongParsers` s návratovou hodnotou `List<ISongParser>`. Tato metoda má jeden vstupní parametr typu `string` obsahující absolutní cestu k adresáři s knihovnami jednotlivých parserů. Metoda funguje tak, že rekurzivně prohledá celý adresář, najde všechny knihovny, které se v adresáři nachází a pro každou knihovnu zjistí, jestli obsahuje objekt implementující rozhraní `IParser`. Pokud takový objekt najde, vytvoří pomocí třídy `Activator`, která je součástí .NET knihovny `System.Runtime`, jeho instanci a tu přidá do seznamu parserů. Nakonec vrátí seznam instancí nalezených parserů setřízený podle jejich priority.

3.6.3 Načtení písně z webu

Pro načtení písně z webu je nutné, aby uživatel přesně zadal název písně i jméno autora. Aplikace postupně pro každý parser (setřízené podle priority) zkusí zavolat metodu `GetSong` s parametry typu `string`, do kterých uloží název písně a jméno autora. Dokud jsme nevyzkoušeli všechny parsery nebo dokud metoda `GetSong` neuspěje a vyhodí výjimku `SongNotFoundException`, voláme metodu na dalším parseru v pořadí. Pokud žádný z parserů neuspěl, vyhodíme výjimku `SongNotFoundException`, kterou aplikace dále zpracovává. Pokud některý z parserů uspěl, tedy vrátil instanci třídy `Song` reprezentující hledanou píseň nalezenou na webové stránce, vrátíme nalezenou píseň a na dalších parserech v pořadí již metodu nevoláme.

3.6.4 Rozšiřitelnost aplikace o nové parsery

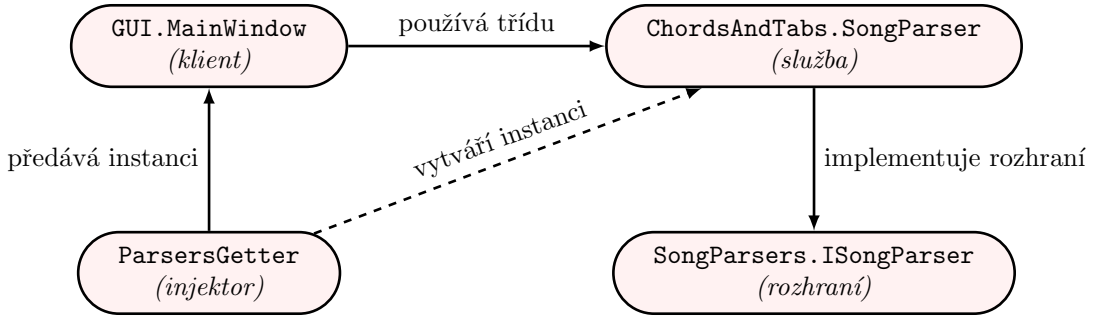
Rozšíření aplikace o nové parsery je poměrně snadné díky implementaci techniky dependency injection. Pro přidání nového parseru je nutné, aby tento nový

parser implementoval rozhraní `ISongParser`, abychom ho mohli používat v aplikaci. Druhou podmínkou je, že parser musí být součástí knihovny jazyka C# s příponou *.dll*. Poslední podmínkou je, že knihovna s novým parserem se musí nacházet ve stejném adresáři jako ostatní binární soubory aplikace (především soubor *SongbookMaker.exe*). Poslední dvě podmínky zajišťují, že při načítání parserů metodou `GetAllSongParsers` se bude prohledávat i tato nová knihovna. Při implementaci nového parseru je nutné, aby metoda `GetSong` vyhazovala výjimku `SongNotFoundException`.

Rozšiřitelnost o nové parsery byla navržena tak, že nový parser nemusí získávat písně z webu, ale může například získávat písně z databáze. Je však nutné, aby parser splňoval podmínky uvedené výše.

3.6.5 Implementace dependency injection pro parsery

Již jsme zmínili, že jsme použili techniku dependency injection, abychom se zbavili závislosti třídy pracující s parsery na samotných parserech. Na obrázku 3.3 je znázorněno, jak byla technika dependency injection implementována. Naším klientem je třída `GUI.MainWindow`, která přímo používá parsery. Naší službou jsou tedy jednotlivé parsery. Pro demonstraci jsme na obrázku 3.3 zvolili jediný parser, a to parser webové stránky *Chords and Tabs* [28], ale pro všechny ostatní parsery funguje dependency injection stejně. Rozhodli jsme se použít rozhraní pro parsery, kterým je `ISongParser`. Naším injektorem je statická třída `ParsersGetter`, která pomocí statické metody `GetAllSongParsers` nalezne všechny parsery, vytvoří jejich instance a přidá je do seznamu, který přiřadí do vlastnosti třídy `GUI.MainWindow`.



Obrázek 3.3 Dependency injection pro parsery

3.7 Formát výstupu

Další funkcionalitou, ve které se aplikace liší od většiny ostatních aplikací zaměřujících se na podobnou problematiku, je že umí uložit zpěvník ve formátu vhodném pro tisk, konkrétně ve formátu PDF. Aplikace kromě výstupu ve formátu PDF umožňuje navíc uložit zpěvník ve formátech TXT nebo JSON. Aplikaci lze snadno rozšířit, aby podporovala ukládání zpěvníku i v jiných formátech. Nyní si popíšeme, jak se převádí objekt reprezentující kytarový zpěvník do jednotlivých formátů, co musí splňovat objekty zajišťující tento převod a jak rozšířit aplikaci o nový formát výstupu. Pro jednodušší popisování bude pro tuto sekci pojem

„writer“ znamenat objekt, který ukládá zpěvník do výstupního souboru s předem daným formátem. Tento pojem budeme v sekci vymezovat uvozovkami a skloňovat pokud to bude nutné.

3.7.1 ISongbookWriter

`ISongbookWriter` je rozhraní obsažené v knihovně `SongbookWriters`, které musí splňovat každý „writer“. Rozhraní vynucuje, že každý „writer“ má následující dvě vlastnosti:

- readonly vlastnost `Type` typu `string`, ve které je uložený název formátu výstupního souboru.
- readonly vlastnost `Suffix` typu `string`, ve které je uložena přípona výsledného výstupního souboru. Součástí přípony uložené ve vlastnosti je počáteční tečka.

Rozhraní `ISongbookWriter` vynucuje navíc metodu `SaveSongbookWithFullPath` bez návratové hodnoty. Tato metoda se volá se dvěma argumenty. První argument je typu `Songbook` a obsahuje zpěvník, který chceme uložit. Druhý argument je typu `string` a obsahuje absolutní cestu k souboru, kam má být zpěvník uložen. Tato metoda vytvoří výstupní soubor se zpěvníkem a ten uloží do paměti zařízení podle cesty předané v parametru.

3.7.2 ISongbookWriterExtension

Třída `ISongbookWriterExtensions` je součástí knihovny `SongbookWriters` a rozšiřuje rozhraní `ISongbookWriter` o metodu `SaveSongbook` bez návratové hodnoty. Tato metoda, nikoliv metoda `SaveSongbookWithFullPath`, se volá ve chvíli, kdy chceme uložit zpěvník do výstupního souboru daného formátu. Metoda má stejné parametry jako výše zmíněná metoda `SaveSongbookWithFullPath`. Metoda `SaveSongbook` funguje tak, že zkontroluje, zda cesta předaná parametrem má správnou příponu a pokud tomu tak není, přidá správnou příponu na konec cesty. Nakonec zavolá metodu `SaveSongbookWithFullPath` se stejným prvním parametrem typu `Songbook` a s druhým parametrem, kterým je cesta zakončená správnou příponou.

3.7.3 Dynamické načítání Writerů

Aby byla aplikace schopná ukládat zpěvník ve všech formátech, které jsou implementovány, je potřeba, aby si při spuštění dynamicky načetla všechny „writery“. Dynamické načtení „writerů“ z knihoven do aplikace zajišťuje statická třída `SongbookWritersGetter`, pomocí statické metody `GetAllSongbookWriters` s návratovým typem `List<ISongbookWriter>`. Tato metoda funguje stejně jako metoda `GetAllSongParsers` zmíněná v sekci Parsery písní z webové stránky, konkrétněji v její podsekci Dynamické načítání parserů. Jediným rozdílem mezi těmito metodami je rozhraní, které implementují instance vrácené v seznamu. Třída `SongbookWritersGetter` je součástí knihovny `SongbookWriters`.

Použití jedné generické metody

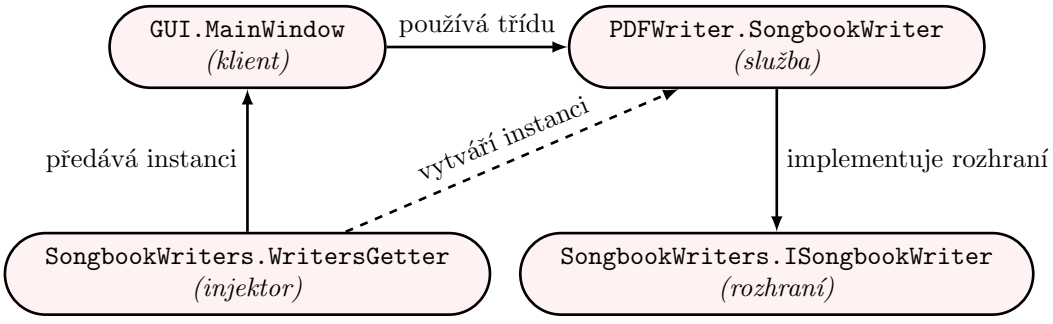
Metody `GetAllSongParsers` a `GetAllSongbookWriters` jsou implementovány až na rozhraní stejně. Nabízí se tedy otázka, proč nenaimplementujeme jednu generickou metodu, která by vracela seznam instancí všech objektů implementujících dané rozhraní. Pro dynamické načtení parserů by se zavolala tato metoda s generickým parametrem `ISongParser` a pro dynamické načtení „writerů“ by se zavolala metoda s generickým parametrem `ISongbookWriter`. Použití jedné generické metody jsme zavrhli, jelikož načítání parserů i writerů lze snadno rozšířit, což by mohlo vynutit změnu metody pro dynamické načítání. Tato změna by mohla vést k rozdělení generické metody na dvě metody, a proto jsme se rozhodli metody oddělit již nyní.

3.7.4 Rozšiřitelnost aplikace o nové formáty výstupu

Rozšíření aplikace o nové „writery“ funguje velmi podobně jako rozšíření o nové parsery, jelikož dynamické načítání parserů a „writerů“ funguje stejně. Nový „writer“ musí implementovat rozhraní `ISongbookWriter` a musí být součástí C# knihovny s příponou *.dll*. Dále je potřeba, aby knihovna obsahující „writer“ byla obsažena v adresáři s ostatními binárními soubory aplikace, především se souborem *SongbookMaker.exe*.

3.7.5 Implementace dependency injection pro writery

Přidání nového „writeru“ je poměrně snadné, jelikož jsme vhodně implementovali techniku dependency injection. Na obrázku 3.4 je znázorněna implementace techniky dependency injection pro odstranění závislosti třídy `GUI.MainWindow` (*klient*) na „writeru“ `PDFWriter.SongbookWriter` (*služba*). Roli *injektoru* zde plní třída `SongbookWritersGetter`.



Obrázek 3.4 Dependency injection pro „writery“

4 Uživatelská dokumentace

V této sekci si představíme jednotlivá okna aplikace s jejich ovládacími prvky.

Záhlaví okna

Všechna okna aplikace mají stejná záhlaví. Záhlaví obsahuje popisek s názvem aplikace a tři tlačítka:

- tlačítko s ikonou domečku, jehož stisknutí přesune uživatele do hlavního menu
- tlačítko s ikonou anglické vlajky, jehož stisknutí nastaví angličtinu jako jazyk aplikace
- tlačítko s ikonou české vlajky, jehož stisknutí nastaví češtinu jako jazyk aplikace

4.1 Hlavní menu

Obrázek 4.1 Hlavní menu aplikace

Hlavní menu se zobrazí při spuštění aplikace a slouží k přesunutí uživatele do dalších oken aplikace. Hlavní menu se skládá ze tří tlačítek:

- *Vytvořit zpěvník*, jehož stisknutí přesune uživatele do okna pro prohlížení zpěvníku
- *Uložené zpěvníky*, jehož stisknutí přesune uživatele do okna pro načtení zpěvníku
- *Ukončit aplikaci*, jehož stisknutí zavře aplikaci

4.2 Okno pro prohlížení zpěvníku

Obrázek 4.2 Okno pro prohlížení zpěvníku

Okno slouží k zobrazení kytarového zpěvníku. V okně jsou zobrazeny jednotlivé písně v pořadí, ve kterém jsou uloženy ve zpěvníku. Okno obsahuje v horní části textové pole sloužící k zadání názvu zpěvníku a „rozbalovací seznam“, pomocí kterého lze upravovat pořadí písní ve zpěvníku. „Rozbalovací seznam“ obsahuje tyto volby:

- *Vlastní řazení*, která umožňuje řazení písní přetažením písně na jinou pozici
- *Řadit dle názvu* ↑, která po zvolení seřadí písně sestupně dle názvu (písně se stejným názvem seřadí sestupně dle jména autora)
- *Řadit dle názvu* ↓, která po zvolení seřadí písně vzestupně dle názvu (písně se stejným názvem seřadí vzestupně dle jména autora)
- *Řadit dle autora* ↑, která po zvolení seřadí písně sestupně dle jména autora (písně se stejným autorem seřadí sestupně dle názvu)
- *Řadit dle autora* ↓, která po zvolení seřadí písně vzestupně dle jména autora (písně se stejným autorem seřadí vzestupně dle názvu)

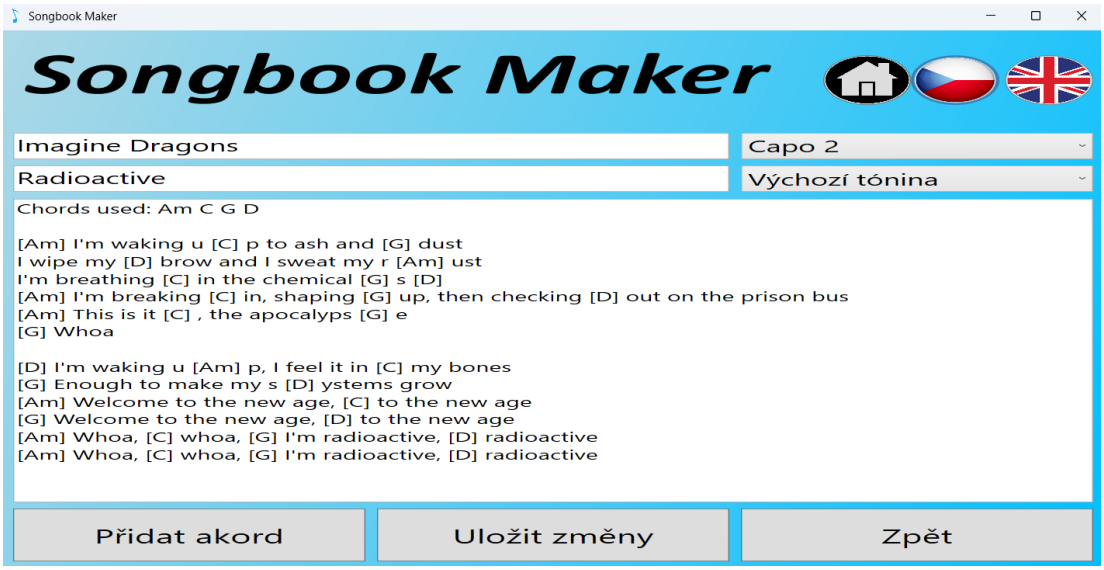
Pro každou píseň obsaženou ve zpěvníku se zobrazí řádek, který se skládá z popisku písně a dvou tlačítek. Popisek písně obsahuje název písně, jméno autora, tóninu písně a pražec, na který se umisťuje kapodastr (pokud je zadán). Pokud se v popisku nezobrazuje tónina, znamená to, že píseň neobsahuje akordy. Tlačítka obsažená v řádku jsou:

- *Upravit píseň*, které po stisknutí přesune uživatele do okna pro úpravu písně, ve kterém je možné upravit píseň odpovídající tlačítku
- *Odebrat*, které po stisknutí odebere píseň ze zpěvníku

Kromě výše uvedených ovládacích prvků obsahuje okno tato tlačítka:

- *Přidat píseň*, které po stisknutí přesune uživatele do okna pro přidání písně.
- *Uložit název*, které po stisknutí uloží jako název zpěvníku text zobrazený v textovém poli (pro zadávání názvu zpěvníku).
- *Vymazat zpěvník*, které po stisknutí vymaže všechny písně ve zpěvníku a současně vymaže i jeho název.
- *Exportovat zpěvník*, které po stisknutí přesune uživatele do okna pro export zpěvníku. Aby byl uživatel přesunut, je potřeba nejprve uložit název zpěvníku.
- *Uložit zpěvník*, které po stisknutí uloží zpěvník do paměti aplikace. Aby byl zpěvník uložen, je potřeba nejprve uložit jeho název.
- *Zpět*, které po stisknutí přesune uživatele do hlavního menu.

4.3 Okno pro úpravu písně



Obrázek 4.3 Okno pro úpravu písně

Okno slouží k úpravě písně, která se již nachází ve zpěvníku. Okno obsahuje tři textová pole:

- textové pole obsahující jméno autora, které se nachází nejvýše v okně
- textové pole obsahující název písně, které se nachází přímo pod textovým polem obsahujícím název autora
- textové pole obsahující text písně s akordy, které je ze všech tří největší

Změnou textu jednotlivých textových polí lze upravovat píseň. Text (v textovém poli obsahujícím text písně s akordy) ohraničený hranatými závorkami je vnímán jako akordová značka. Každý takový text bude při ukládání změn a při transpozici písně převeden na akord. Pokud převedení selže, program upozorní uživatele a změny neprovede. Změny způsobené přepsáním textových polí se projeví ve zpěvníku po stisknutí tlačítka *Uložit změny*.

Kromě úpravy textu písně, akordů písně, názvu písně a jména autora lze v tomto okně změnit tóninu písně a nastavit pražec, na který se má umístit kapodastr. I pro tyto změny platí, že se ve zpěvníku projeví až po kliknutí na tlačítko *Uložit změny*. Změny se provádí pomocí dvou „rozbalovacích seznamů“:

- vrchní „rozbalovací seznam“ slouží k nastavení pražce, na který se má umístit kapodastr.
- spodní „rozbalovací seznam“ slouží ke změně tóniny písně. Změna tóniny se ihned projeví v textovém poli s textem písně.

Kromě textových polí a „rozbalovacích seznamů“ uvedených výše, obsahuje okno tato tři tlačítka:

- *Přidat akord*, které po stisknutí zobrazí okno pro přidání akordu
- *Uložit změny*, které po stisknutí uloží změny písně do zpěvníku a přesune uživatele do okna pro prohlížení zpěvníku
- *Zpět*, které zahodí provedené změny a přesune uživatele do okna pro prohlížení zpěvníku

4.3.1 Okno pro přidání akordu

Obrázek 4.4 Okno pro přidání akordu

Okno slouží k vložení akordu do písně. Okno se zobrazuje v popředí okna pro úpravu písně nebo okna pro přidání písně. Okno v pozadí je zablokované, dokud

se okno pro přidání akordu nezavře. Součástí okna je popisek s textem „Zadejte akord“ a tři ovládací prvky:

- tlačítko *Zavřít*, které zavře okno, a tím odblokuje okno v pozadí
- textové pole sloužící k zadání akordové značky vkládaného akordu
- tlačítko *Přidat akord*, které vloží do textu písně uživatelem zadanou značku akordu z textového pole, zavře okno a odblokuje okno v pozadí

4.4 Okno pro přidání písně

Obrázek 4.5 Okno pro přidání písně

Okno slouží pro přidání nové písně do zpěvníku. Do zpěvníku lze přidat vlastní píseň, či píseň získanou z internetu. Okno má velice podobný vzhled jako okno pro úpravu písně. Liší se pouze tlačítky, která se nacházejí ve spodní části okna. Okno pro přidání písně používá tato tlačítka:

- *Smazat píseň*, po jehož stisknutí se vymaže aktuálně rozpracovaná píseň a okno se vrátí do stejného stavu, v jakém bylo těsně po otevření
- *Načíst píseň z webu*, které slouží k nahrání písně z webu
- *Přidat akord*, které má stejnou funkcionalitu jako tlačítko stejného názvu v okně pro úpravu písně
- *Uložit píseň*, které po stisknutí vytvoří píseň z hodnot uložených v textových polích a „rozbalovacích seznamech“, přidá ji na konec zpěvníku a přesune uživatele do okna pro prohlížení zpěvníku
- *Zpět*, které po stisknutí smaže změny vytvořené v okně a přesune uživatele do okna pro prohlížení zpěvníku

4.4.1 Získání nové písně

Načítání písně z internetu

Pro správné načtení písně z internetu je nutné mít správně zadaný název písně a jméno jejího autora. Jsou-li tyto hodnoty správně zadány, píseň se načte stisknutím tlačítka *Načíst píseň z webu*. Pokud načtení písně proběhlo bez chyb, text písně i s akordy se objeví v textovém poli. Pokud při načítání písně nastala chyba, program upozorní uživatele. Při načítání písně z webu je nutné, aby byly název písně i jméno autora zadány přesně, například prohození křestního jména a příjmení autora může vést k tomu, že se píseň nenačte. Načtenou píseň lze dále upravovat pomocí výše zmíněných ovládacích prvků.

Aplikace načítá písně pouze ze dvou webových stránek a proto může docházet k tomu, že načtení písně selže z důvodu, že ani jedna webová stránka neobsahuje hledanou píseň.

Tvorba vlastní písně

V okně je možné vytvořit vlastní píseň. Vlastní píseň vytvoříme zadáním názvu písně, jména autora a textu písně do příslušných textových polí. Akordy do písně lze přidat stisknutím tlačítka *Přidat akord* nebo zadáním akordové značky ohraničené hranatými závorkami přímo do textu písně.

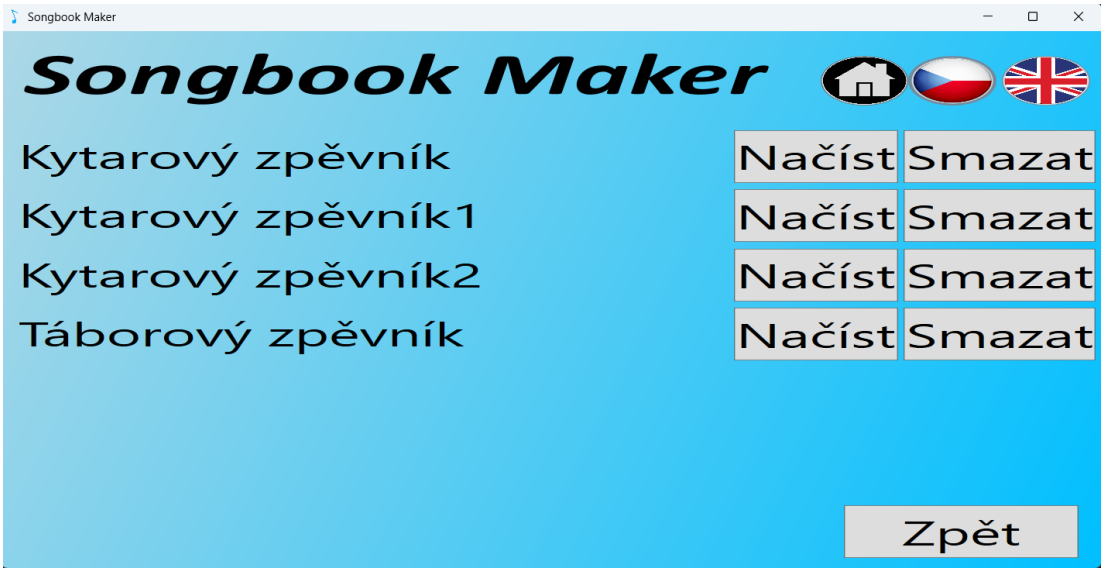
4.5 Okno pro export zpěvníku

Obrázek 4.6 Okno pro export zpěvníku

Okno slouží pro uložení souboru, který obsahuje kytarový zpěvník, do zařízení. V aplikaci lze kytarový zpěvník uložit jako soubor ve formátu PDF, JSON nebo TXT. Okno pro každý podporovaný formát obsahuje tlačítko, po jehož stisknutí se objeví dialogové okno windows pro ukládání souborů, ve kterém si uživatel může zvolit místo uložení souboru. Po potvrzení této volby se soubor uloží na uživatelem

zvolené místo v zařízení. Okno obsahuje tlačítko *Zpět*, které po stisknutí přesune uživatele do okna pro prohlížení zpěvníku.

4.6 Okno pro načtení zpěvníku



Obrázek 4.7 Okno pro načtení zpěvníku

V tomto okně se zobrazují zpěvníky uložené v aplikaci. Jednotlivé zpěvníky lze odstranit z paměti aplikace kliknutím na tlačítko *Smazat*, či načíst do aplikace kliknutím na tlačítko *Načíst*. Načtení zpěvníku přesune uživatele do okna pro prohlížení zpěvníku, ve kterém se zobrazí nově načtený zpěvník. Okno obsahuje tlačítko *Zpět*, které po stisknutí přesune uživatele do hlavního menu.

5 Závěr

V této práci jsme navrhli a následně implementovali aplikaci pro tvorbu kytarového zpěvníku. Tato aplikace se od ostatních dostupných aplikací liší tím, že umožňuje úpravu a následné uložení písní dostupných z webových stránek. Aplikace umožňuje vytvořit z kytarového zpěvníku soubor PDF, což velmi zjednodušuje převedení kytarového zpěvníku do tištěné podoby.

K aplikaci jsme vytvořili uživatelské rozhraní umožňující úpravu jednotlivých písní zpěvníku, tvorbu vlastních písní, načítání a ukládání písní do zpěvníku a úpravu samotného zpěvníku.

5.1 Vhodná rozšíření

5.1.1 Databáze písní

Aplikace umožňuje uživateli načíst písně ze dvou webových stránek. Kromě rozšíření o načítání písní z více webových stránek by bylo vhodné rozšířit aplikaci o možnost načítání písní z vlastního databázového serveru. Do databáze by se ukládaly ve vhodném formátu písně načtené z webu, které nejsou uložené v databázi. Při načítání písně by se nejprve vyhledala píseň v databázi a až poté na webových stránkách. Vytvoření databáze by zkrátilo čas načítání písně a také by usnadnilo rozšíření aplikace o vyhledávání písní dle názvu písně, jména autora, či úryvku textu.

5.1.2 Tvorba vlastních akordů

Aplikace podporuje známé a často používané druhy akordů, ale například méně známý akord s akodovou značkou `E/#G` aplikace nepodporuje. Proto by bylo vhodné rozšířit, či změnit implementaci třídy `Chord` tak, aby umožnila rozpoznávat více akordů, či vytvořit vlastní akord a k němu příslušnou akordovou značku.

5.1.3 Přidání podpory pro taby písní

Texty písní v aplikaci obsahují pouze akordové značky, které zaznamenávají harmonii písně. Mnoho písní obsahuje kytarová sóla (melodické linky hrané na kytaru), která bývají zaznamenávána pomocí tzv. „tabů“. Vhodným rozšířením aplikace by bylo přidání podpory pro tvorbu tabů, vkládání tabů do písně, úpravu tabů a vhodné zobrazení tabů ve výsledném PDF souboru (i v souborech ostatních formátů).

5.1.4 Formát výstupu HTML

Mezi rozšíření bez nutnosti úpravy kódu patří rozšíření aplikace o nové formáty výstupu kytarového zpěvníku. Vhodným formátem výstupu by byla například HTML stránka. Stránka by obsahovala název zpěvníku a list odkazů na stránky obsahující jednotlivé písně. Stránky obsahující jednotlivé písně by kromě zobrazení

základních údajů o písni umožňovaly transponovat píseň, nastavit pražec, na který se má upevnit kapodastr, či autoscroll písně s nastavitelnou rychlostí.

5.2 Osobní zhodnocení

Aplikace obsahuje části, se kterými jsem spokojený, ale i části, které bych nyní implementoval jiným způsobem.

Jsem spokojen s objektovým návrhem aplikace a rozdělením aplikace na jednotlivé komponenty, díky čemuž lze aplikaci poměrně snadno rozšířit o novou funkcionalitu. Dále jsem spokojen se vzhledem a funkcionalitou uživatelského rozhraní a se vzhledem výstupního PDF souboru.

Na druhou stranu nejsem spokojený s tvorbou PDF souboru a myslím si, že rozšíření PDF souboru, například o taby k písním, či obrázky by vedlo k velkému zásahu do kódu knihovny `PDFWriter`. Proto bych nyní při tvorbě PDF souboru použil více funkcionality knihovny `PDFSharp`, či zvolil jinou vhodnější knihovnu.

Závěrem je třeba říci, že práce splňuje cíle vytyčené v sekci Cíle práce a osobně jsem s její finální podobou velice spokojený.

Literatura

1. William Moebs Samuel J. Ling, Jeff Sanny. *University Physics Volume 1*. Houston, Texas: OpenStax, 2016. Volume 1. Dostupné také z: https://openstax.org/books/university-physics-volume-1/pages/17-introduction.
2. Reichl, J.; Všetička, M. *Zdroje zvuku* [online]. [cit. 2024-03-08]. Dostupné z: http://fyzika.jreichl.com/main.article/view/173-zdroje-zvuku.
3. Reichl, J.; Všetička, M. *Základní dělení zvuků* [online]. [cit. 2024-03-08]. Dostupné z: http://fyzika.jreichl.com/main.article/view/186-zakladni-deleni-zvuku.
4. Theory, Hello Music. *What Is Melody In Music? A Complete Guide* [online]. [cit. 2024-03-08]. Dostupné z: https://hellomusictheory.com/learn/melody.
5. Theory, Hello Music. *What Is Harmony In Music? A Complete Guide* [online]. [cit. 2024-03-08]. Dostupné z: https://hellomusictheory.com/learn/harmony.
6. Theory, Hello Music. *What Is The Root Of A Chord?* [online]. [cit. 2024-03-08]. Dostupné z: https://hellomusictheory.com/learn/root-of-a-chord.
7. Theory, Hello Music. *How To Transpose Music: A Step By Step Guide* [online]. [cit. 2024-03-08]. Dostupné z: https://hellomusictheory.com/learn/how-to-transpose-music.
8. LinkeSOFT. *SongBook Windows PCs 5.4.2 Songs and Chords* [online]. [cit. 2024-03-13]. Dostupné z: https://linkesoft.com/songbook/windows.html.
9. Vlastimil, Vašek. *Písničky s akordy* [online]. [cit. 2024-03-13]. Dostupné z: https://pisnicky-akordy.cz.
10. Janda, Paweł. *Mobile app, web app, desktop app: know the difference!* [online]. [cit. 2024-03-16]. Dostupné z: https://mobitouch.net/blog/mobile-app-web-app-desktop-app-know-the-difference.
11. Microsoft. *A tour of the C# language* [online]. [cit. 2024-03-18]. Dostupné z: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp.
12. adegeo. *What is Windows Presentation Foundation - WPF .NET | Microsoft Learn* [online]. [cit. 2024-04-02]. Dostupné z: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/overview.
13. Ltd, Openize Pty. *PDF File Format - What is a PDF file?* [online]. [cit. 2024-03-09]. Dostupné z: https://docs.fileformat.com/pdf.
14. Ltd, Openize Pty. *What is a DOC file?* [online]. [cit. 2024-03-09]. Dostupné z: https://docs.fileformat.com/word-processing/doc.
15. Ltd, Openize Pty. *What is a DOCX file?* [online]. [cit. 2024-03-09]. Dostupné z: https://docs.fileformat.com/word-processing/docx.

16. Ltd, Openize Pty. *What is a TEX File?* [online]. [cit. 2024-03-11]. Dostupné z: https://docs.fileformat.com/page-description-language/tex.

17. Thành, Hàn Thê´ et al. *The pdfTEX user manual*. 2024. Ver. 1.2. Dostupné také z: https://mirrors.nic.cz/tex-archive/systems/doc/pdftex/manual/pdftex-a.pdf.

18. Software GmbH, empira. *PDFsharp & MigraDoc* [online]. [cit. 2024-03-11]. Dostupné z: https://pdfsharp.net/Overview.ashx.

19. Wazir, Zeeshan. *The 5 C# PDF Libraries Every Developer Mostly Use* [online]. [cit. 2024-03-11]. Dostupné z: https://dev.to/xeshan6981/the-5-c-pdf-libraries-every-developer-mostly-use-1g7i.

20. Lotfi, Chaimaa; Srinivasan, Swetha; Ertz, Myriam; Latrous, Imen. Web Scraping Techniques and Applications: A Literature Review. In: 2021, s. 381–394. isbn 9789391842086. Dostupné z doi: 10.52458/978-93-91842-08-6-38.

21. Services, Amazon Web. *What is a RESTful API?* [online]. [cit. 2024-03-15]. Dostupné z: https://aws.amazon.com/what-is/restful-api.

22. mozilla.org. *HTTP response status codes* [online]. [cit. 2024-03-15]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status.

23. mozilla.org. *Document Object Model (DOM)* [online]. [cit. 2024-03-15]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model.

24. Karatas, Gulbahar. *What Is a Headless Browser and Its Applications? in 2024* [online]. [cit. 2024-03-15]. Dostupné z: https://research.aimultiple.com/headless-browser.

25. Tutorials, Dot Net. *Dependency Injection Design Pattern in C#* [online]. [cit. 2024-03-13]. Dostupné z: https://dotnettutorials.net/lesson/dependency-injection-design-pattern-csharp/?utm_content=cmp-true.

26. Stackify. *Design Patterns Explained – Dependency Injection with Code Examples* [online]. [cit. 2024-03-13]. Dostupné z: https://stackify.com/dependency-injection.

27. Inc., IONOS. *What is a plug-in and what is it used for?* [online]. [cit. 2024-03-13]. Dostupné z: https://www.ionos.com/digitalguide/server/know-how/what-is-a-plug-in.

28. *Chords and tabs* [online]. [cit. 2024-03-22]. Dostupné z: https://www.chords-and-tabs.net.