VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

KNIHOVNA PRO ZPRACOVÁNÍ SOUBORŮ VE FORMÁTU EPUB

LIBRARY FOR PROCESSING OF EPUB FILES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ROMAN VÁLEK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MARTIN KRČMA**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/19448/2016/xvalek08

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Válek Roman**
Obor: Informační technologie
Téma: **Knihovna pro zpracování souborů ve formátu epub**
Library for Processing of epub Files
Kategorie: Softwarové inženýrství

Pokyny:

1. Prostudujte formát pro uložení elektronických knih epub ve verzi 3.0.
2. Navrhněte rozhraní knihovny pro zpracování souborů v tomto formátu. Knihovna bude umět soubory v tomto formátu vytvářet, číst a editovat v rámci logických celků daných všemi úrovněmi kapitol a odstavci. Bude podporovat práci s HTML tagovaným a netagovaným textem, zpracování a editaci formátem nabízených metadat, včetně nepovinných. Všechny identifikátory budou v anglickém jazyce.
3. Implementujte knihovnu v jazyce C++.
4. Vytvořte dokumentaci rozhraní knihovny ve formátech HTML a PDF. Dokumentace bude v angličtině.
5. Vytvořte demonstrační aplikaci provádějící převod formátu Markdown na formát epub.
6. Vyhodnoťte dosažené výsledky a navrhněte další rozvoj práce.

Literatura:

- Dle pokynů vedoucího.

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

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

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

Vedoucí: **Krčma Martin, Ing.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

prof. Ing. Lukáš Sekanina, Ph.D.
vedoucí ústavu

Abstrakt

Tato bakalářská práce se zabývá analýzou formátu EPUB určeného k elektronické publikaci knih a dokumentů v jeho specifikaci verze 3.0. Na základě získaných poznatků je navržena knihovna pro práci se soubory v tomto formátu a je popsána její implementace v jayzce C++. Práce dále popisuje rozhraní knihovny a její využití při implementaci aplikací pracujících s elektronickými publikacemi. Je popsána implementace jednoduché demonstrační aplikace pro převod dokumentu ve formátu Markdown do formátu EPUB.

Abstract

This bachelor's thesis deals with an analysis of the EPUB format version 3.0 serving for the electronic publication. On this basis it presents the library designed to work with files in the EPUB format and describes its implemantation. The thesis also deals with the library interface and its usage for applications processing the EPUB files. The simple demonstration application performing a conversion of the Markdown format to the EPUB format is presented.

Klíčová slova

Epub, C++, knihovna, Markdown, LipZip, TinyXML2

Keywords

Epub, C++, library, Markdown, LipZip, TinyXML2

Citace

VÁLEK, Roman. *Knihovna pro zpracování souborů ve formátu epub*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Martin Krčma

Knihovna pro zpracování souborů ve formátu epub

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Martina Krčmy. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Roman Válek
17. května 2017

Poděkování

Na tomto místě bych rád poděkoval Ing. Martinovi Krčmovi, za vedení bakalářské práce, odbornou pomoc a poskytnuté konzultace.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Knihy a jejich tvorba se od svých počátků během historie postupně rozvíjely. Od náboženských textů, přepisovaných mnichy v klášterech, kdy přepis trval celé týdny a gramotnosti dosahovalo nízké procento lidí, k rozmachu Gutenbergova knihtisku v 15. století, díky němuž tisk trval mnohonásobně kratší dobu a knihy byly dostupnější čím dál širší veřejnosti, až k moderním knihařským lisům. S nástupem internetu se začaly texty šířit elektronicky, nejdříve jen jako prostý text, následovaný textovými formáty jako jsou `doc` a `rtf` až k dnešním standardům.

V dnešní době jsou elektronické knihy a publikace na vzestupu. Vznikají nakladatelství a nové kanály pro publikování elektronických knih, jenž využívají možnosti jednodušší a rychlejší distribuce k uživatelům. Další výhodou těchto publikací je možnost mít veškerý obsah své knihovny kdykoliv a kdekoliv dostupný ve vhodném elektronickém zařízení. Tento formát dokumentů pomáhá také k udržování ekologické stability, kdy se snižuje spotřeba dřeva určeného pro papírnický průmysl.

Vzniká mnoho standardů pro uložení elektronických publikací, a s tím se rozšiřuje i základna zařízení, které jsou schopné tento obsah zobrazovat, od čteček elektronických knih přes mobilní telefony a tablety až po počítače. Tato práce se zabývá vytvořením knihovny pro jazyk C++ pracující se standardem Epub, který se v dnešní době řadí mezi nejrozšířenější.

V následující kapitole bude popsán formát Epub, jeho historie a další formáty pro ukládání elektronických knih. V kapitole 3 bude popsán návrh knihovny s rozložením tříd do logických celků společně s návrhem metod pro dělení kapitol a kontroly konzistence. Kapitola 4 se bude zaobírat implementací v jazyce C++ a použitím externích knihoven. V kapitole 5 bude popsáno aplikační rozhraní knihovny. V kapitol 6 bude popsána instalace knihovny a jejich závislostí. Kapitola 7 bude obsahovat popis jazyka Markdown a návrh demonstrační aplikace, vytvořené za pomoci této knihovny, sloužící k převodu tetu v jazyce Markdown na soubor Epub. V závěrečné kapitole bude shrnutí práce a návrhy pro další rozšíření.

Kapitola 2

Formát Epub

V této kapitole bude probrán standard `Epub`, v sekci 2.1 bude shrnuta historie formátu a v následujících sekcích bude probrána struktura, účel a význam souborů `mimetype`, `container.xml`, `toc.ncx`, `souboru .opf` a `textových souborů` v archivu Epub.

Epub je formát pro elektronické dokumenty a publikace založený na webových standartech. Byl vytvořen dle standartu organizace *International Digital Publishing Forum (IDPF)*[16] skládající se ze tří otevřených standardů *Open Publication Structure (OPS)*[24], *Open Packaging Format (OPF)*[23] a *Open Container Format (OCF)*[21].

2.1 Historie

`Epub` vychází z formátu známého jako *Open EBook Publication Structure (OEBPS)*[22]. OEBPS byl schválen v roce 1999 organizací Open eBook Forum, ze které se později stalo Mezinárodní digitální publikační fórum (IDPF). V roce 2005 začala práce na kontejnerovém formátu s jedním souborem OEBPS, který byl schválen IDPF jako Open Container Format v roce 2006. Paralelně začala práce na OEBPS 2.0 a byla schválena jako přejmenovaná verze *Epub 2.0*[5]. Epub soubor je ve skutečnosti přejmenovaný soubor .ZIP se specifickou adresářovou a souborovou strukturou.

V současné době je využívána specifikace Epub 3. Bill Kasdorf ve svých článcích [35, 36] popsal okolnosti vzniku nové specifikace, porovnal ji vůči její starší verzi a odhadl její dopad na průmysl elektronického publikování a růst souvisejícího ekosystému. Proces vytváření nové specifikace byl složitý a potýkal se s nedostatkem času, proto byl rozdělen na několik částí, které byly poté vytvářeny odděleně. [31] V předchozích letech vyšlo v zahraničí o nové specifikaci několik publikací od Matta Garrishe [32, 33, 34]. Na tuzemském trhu je rovněž dostupná česká kniha [38] určená především pro nakladatele, která pojednává o publikování ve formátu EPUB a MOBI.

2.2 Struktura archivu

`Open Container Format` (OCF) definuje adresářovou strukturu a její uložení v kontejneru zip. Na přiložených obrázcích je možné vidět jak mohou vypadat struktury archivu Epub. Kořenový adresář musí obsahovat složku `META-INF` a soubor `mimetype`. Ostatní soubory mohou být kdekoliv v adresáři, kromě adresáře `META-INF`, který obsahuje informace o metadatech knih. Může v něm být jeden nebo i více souborů, ale bude obsahovat minimálně soubor `container.xml`, jak je vidět na obrázku 2.1b.

Na druhou stranu je možné v archivu vytvořit organizovanou strukturu obsahu rozdělením souborů do složek. Jak lze vidět na obrázku 2.1a, všechny soubory, které nemají pevně dané umístění, jsou uloženy a dále rozřazeny ve složce `OEBPS`, do funkcionálně oddělených složek `images` obsahující obrázky obsažené v dokumentu, `styles` sloužící pro uložení souboru s kaskádovými styly a `xhtml` pro uložení textového obsahu dokumentu.

2.2.1 Soubor mimetype

Soubor mimetype je přečten jako první a oznamuje čtecímu zařízení v jakém formátu je dokument uložen. Tento soubor je jednotný pro všechny Epub soubory. Obsahuje deklaraci MIME.

```
application/epub+zip
```

MIME

Původně definován pro použití v e-mailových zprávách, použití se rozšířilo i do dalších protokolů. Typ media je složen z typu, podtypu a volitelných parametrů.

2.2.2 Soubor container.xml

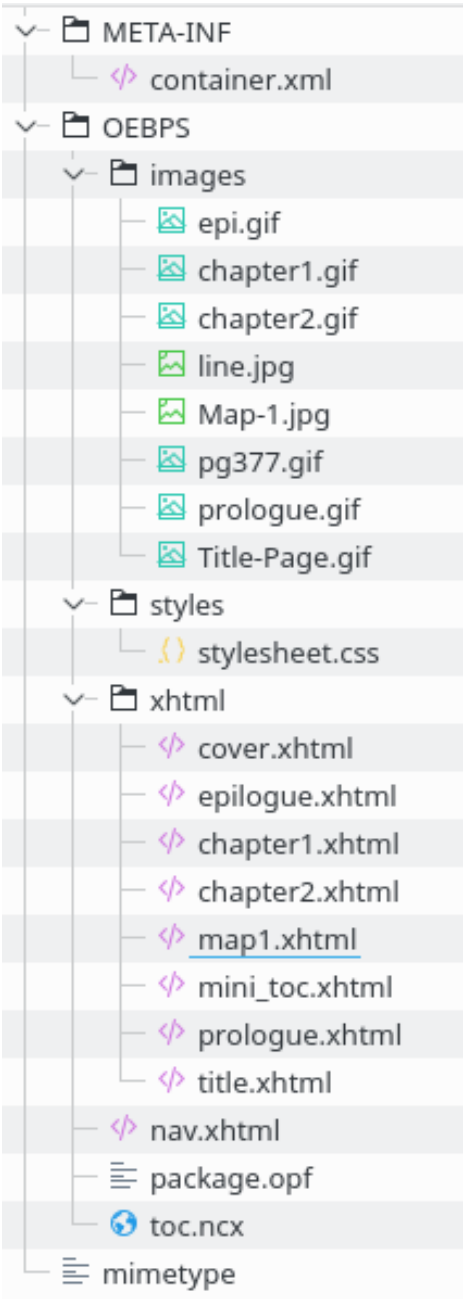
Soubor je uložen ve složce META-INF. Je to soubor obsahující data ve formátu XML reprezentující seznam souborů `opf`. Díky této informaci je čtecí zařízení schopno lokalizovat opf soubor a získat více informací o publikaci. Strukturu souboru opf ilustruje Schéma 2.1. Soubor musí obsahovat jeden a více záznamů o opf souborech.

```
1 <xml version="1.0">
2     <container version="1.0"
3                xmlns="urn:oasis:names:tc:opendocument:xmlns:container">
4         <rootfiles>
5             <rootfile full-path="content.opf"
6                       media-type="application/oebps-package+xml"/>
7         </rootfiles>
8     </container>
9 </xml>
```

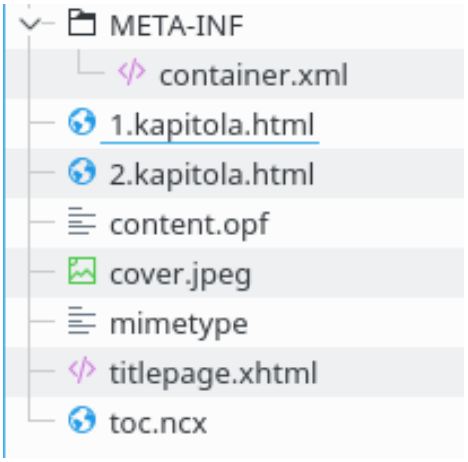
Schéma 2.1: Ukázka souboru container.xml

2.2.3 Soubor .opf

Podobně jako ostatní soubory je i tento soubor uložen ve formátu XML. Tento soubor obsahuje primární zdroj informací o publikaci. Nejvyšším elementem je `package`, který ve svých atributech obsahuje položky `version` udávající informaci o verzi specifikace Epub,v níž je publikace uložena a dále pak `unique-identifier` sloužící jako unikátní identifikátor, podle kterého čtecí zařízení určují duplicitní záznamy o dokumentech ve své databázi. Soubor rovněž obsahuje tři klíčové elementy nesoucí další informace. Těmito elementy jsou `Metadata`, `Manifest` a `Spine`, které budou popsány v dalším textu.



(a) Ukázka strukturovaného obsahu



(b) Ukázka nestrukturovaného obsahu

Obrázek 2.1: Struktury obsahu Epub archivu

Metadata

Tento element uchovává informace o obsahu publikace, které jsou uloženy ve vnořených elementech XML. Epub vyžaduje jen elementy `title`, `identifier`, `language`, které nesou informace o názvu publikace, její unikátní identifikátor a informaci o jazyku. Ostatní elementy jako `creator`, `contributor`, `date`, `description` a další jsou volitelné.

```
<metadata xmlns:dc="http://purl.org/dc/elements/1.1/">
<dc:title id="title">Název dokumentu</dc:title>
<dc:identifier id="uuid-id">urn:isbn:1234567890</dc:identifier>
<dc:language>cs</dc:language>
<dc:creator id="creator">Autor</dc:creator>
<description>Popis dokumentu</description>
```

Schéma 2.2: Ukázka souboru container.xml

Manifest

Archiv formátu Epub může kromě souborů se samotným textovým obsahem zahrnovat i soubory s kaskádovými styly, fonty, obrázky, videi a dalšími soubory, které rozšiřují původní text publikace o další obsah a možnosti vizuálního formátování. Seznam těchto souborů je pak uložen v elementu `manifest`, jehož každý vnořený element `item` odkazuje na konkrétní soubor s obsahem a přiděluje mu jednoznačný identifikátor, který je platný v rámci celého archivu. Rovněž určuje typ souborů, podle kterého čtecí zařízení určí způsob zpracování. Ilustrace těchto informací je na schématu 2.3

```
<item href="cover.jpeg" id="cover" media-type="image/jpeg"/>
<item href="stylesheet.css" id="css" media-type="text/css"/>
<item href="titlepage.xhtml" id="titlepage"
                             media-type="application/xhtml+xml"/>
<item href="toc.ncx" id="ncx" media-type="application/x-dtbncx+xml"/>
<item href="1.kapitola.html" id="html1"
                             media-type="application/xhtml+xml"/>
```

Schéma 2.3: Zápis různých zdrojů v elementu manifest

Spine

Seznam souborů v elementu `manifest` obecně neurčuje pořadí, v jakém budou čtenáři zobrazeny. Pro určení pořadí zobrazení textového obsahu proto soubor opf obsahuje element `Spine`, jehož položky svým pořadím v elementu určují pořadí souborů. Položky se odkazují na soubory skrze jejich identifikátor deklarovaný v elementech `manifest`. Mohou se odkazovat pouze na soubory typu `application/xhtml+xml` nebo `application/x-dtbook+xml`. Ukázka elementu `Spine` a jeho položek je na Schématu 2.4.

2.2.4 soubor toc.ncx

Zkratka ToC znamená `Table of content`. Má obdobný význam jako element `Spine` v souboru .opf, ale na rozdíl od něj neurčuje pořadí zobrazení textových souborů, ale obsahuje

```
<itemref idref="titlepage"/>
<itemref idref="html1"/>
<itemref idref="html2"/>
```

Schéma 2.4: Element spine

hierarchickou strukturu publikace, která dovoluje uživateli přímo přecházet mezi kapitolami, podkapitolami a sekcemi dokumentu. Slouží jako rozcestník obsahu publikace pro jejího čtenáře, zatímco informace v souboru .opf slouží jako rozcestník pro zobrazovací zařízení, která zároveň určuje i pořadí procházení souborů.

Skládá se ze tří základních částí. `Head`, `docTitle` obsahující název publikace a `NavMap` obsahující obsah publikace. `NavMap` obsahuje elementy `navPoint` určující pozice v dokumentu, na které je možné přejít. `NavPoint` má jako atributy položky `id` a `playOrder` určující pořadí zobrazení, dále `NavPoint` se skládá z elementů `navLabel` jména záznamu, jenž se zobrazí čtenáři a `content` odkazující na soubor. Odkazovat se mohou na celý soubor *kniha.html* nebo na jeho části s použitím html záložky #. Pokud se element odkazuje na část html souboru, je nutné, aby v souboru byla nastavena záložka, která bude souhlasit se záložkou v `.NCX` souboru. Obsah souboru toc je ilustrován na schématu 2.5

```
<navMap>
<navPoint id="titulní strana" playOrder="1">
<navLabel><text>název knihy</text></navLabel>
<content src="kniha.html"/>
</navPoint>
<navPoint id="předmluva" playOrder="2">
<navlabel><text>předmluva</text></navLabel>
<content src="kniha.html#predmluva"/>
</navPoint>
...
</navMap>
```

Schéma 2.5: Soubor toc.ncx

2.2.5 Textové soubory

Textové soubory jsou uloženy v podobě `html` nebo `xhtml` souborů. Soubory mají podobu html souboru obsahující html hlavičku, elementy `head` který je složen z informací o názvu publikace, použité znakové sadě a cesty ke zdrojovému souboru kaskádových stylů a `body` obsahující zobrazovaný text formátovaný pomocí html značek a kaskádových stylů. Výjimku tvoří soubory s titulní stranou a soubory obsahující různé přílohy, kde se často v elementu `body` vyskytují pouze `html` odkazy na soubory s obrázky nebo jiným vizuálním obsahem.

2.3 Aplikace

Jelikož je Epub v tuto chvíli nejrozšířenějším formátem elektronických knih, rozšiřuje se pro něj i podpora ve čtecích zařízeních, mobilních aplikacích a desktopových programech. V této sekci 2.3.1 bude popsáno několik zástupců z uvedených odvětví čtení elektronických

knih a v sekci 2.3.2 budou popsány programy sloužící ke konverzi souborů z a do formátu Epub.

2.3.1 Prohlížeče elektronických knih

Tato sekce se bude skládat ze dvou částí, kde v první budou popsány čtecí zařízení a ve druhé části aplikace pro čtení elektronických knih, ve které budou popsány nejznámější desktopové programy ke kterým po rozšíření chytrých zařízení přibyly i jejich mobilní alternativy.

Jedním z nejznámějších desktopových programů je `FBReader`[8], který byl vyvinut v roce 2005 Nikolayem Pultsinem. Výhodou této čtečky je její multiplatformní využití na všech zařízeních s OS Windows, Linux a OS X včetně mobilní platformy Android. Tato čtečka podporuje kromě formátu Epub i další nejznámější formáty jako `.mobi` a `pdf`.

Mezi další rozšířené aplikace patří `Cool Reader`[2]. Aplikace podporuje širokou škálu formátu Epub(bez DRM ochrany),fb2, doc, txt, rtf, html, chm, tcr, pdb, prc, mobi (bez DRM ochrany), pml. Stejně jako předchozí aplikace je i `Cool Reader` multiplatformní s podporou Windows, Linux a Android.

2.3.2 Konvertory

Calibre[1] primárně slouží jako knihovna pro uchovávání elektronických knih s vloženou jednoduchou čtečkou, kromě toho obsahuje funkce pro převod do mnoha formátů s možností editace základních metadat.
Sigil [29] je multiplatformní Epub editor vyvinut Strahinjarem Markovićem v roce 2009 jako školní projekt, ze kterého se stal základ pro bakalářskou a následně diplomovou práci. Kromě editace obsahuje i možnost náhledu výsledné knihy před jejím vygenerováním.

2.3.3 Existující knihovny a srovnání

Výše zmíněné aplikace využívají svou vlastní implementaci práce se soubory formátu Epub. V současné době již ale existuje několik knihoven pro práci s tímto formátem napsaných v různých jazycích. V jazyce python bylo vytvořeno několik knihoven. Nejstarší je knihovna *pipy.epub* [26] pracující s formátem Epub verze 2. Verze 3 je podporována knihovnou *EbookLib* [4], která dovoluje i konverzi do formátu MOBI. Další jednoduchou knihovnou je *pypub* [27]. Všechny tyto knihovny se zaměřují na jednoduchou práci s formátem. Umožňují otevření a načtení existujícího souboru stejně jako vytváření nové publikace. Všechny knihovny nabízejí práci s kapitolami, knihovna EbookLib nabízí i práci s metadaty a strukturálními soubory. Podobné možnosti má i knihovna *publib* [6] napsaná v jazyce Java.

V jazyce C++ doposud žádná volně dostupná a otevřená knihovna napsána nebyla a výše zmíněné knihovny pro tento jazyk postrádají wrappery (rozhraní umožňující přístup k rutinám těchto knihoven z metod napsaných v jazyce C++). Pro zjednodušení práce s formátem Epub v prostředí jazyka C++ je tedy výhodné vytvoření knihovny v tomto jazyce napsané. Výše zmíněné knihovny rovněž postrádají některé funkce, jako je jednoduchá kontrola konzistence souborů. Tato funkcionalita by byla užitečná vzhledem k existenci nekonzistentních souborů vzniklých nevhodnou konverzí z jiných formátů. Následkem konverzí také vznikají dokumenty postrádající strukturu a dělení na kapitoly, což značně ztěžuje čtenářovu orientaci a navigaci v textu. Proto by bylo vhodné rovněž implementovat funkci umožňující jednoduché dělení dokumentu na kapitoly podle různých klíčů. Ne všechny z uvedených knihoven rovněž podporují hlubší práci s formátem, tedy editaci metadat a přímou manipulaci s daty strukturálních souborů TOC a opf. Takovýto přístup k vnitřním

datům je však potřebný mimo jiné i pro opravu nekonzistentních či jinak poškozených souborů. V navrhované knihovně proto budou implementovány všechny tyto funkce.

2.4 Jiné formáty

Kromě formátu Epub existuje řada dalších formátů pro ukládání elektronických publikací. Jedním z nejrozšířenějších je *AZW/MOBI* [20] vyvíjený a používaný společností Amazon (původním tvůrcem byla společnost Mobipocket, od níž Amazon formát odkoupil). Tento formát podporuje mnoho pokročilých funkcí včetně DRM (Digital Rights Management), práce s fonty a grafikou, komprese a dalších. Jeho nevýhodou je, že je podporován především zařízeními od své mateřské společnosti, tedy čtečkami Amazon Kindle.

Dalším formátem, dnes již zastaralým, je *eReader/PDB* [7]. Jeho výhodou je široká kompatibilita, nejvýraznější nevýhodou je pak absence podpory pro formátování textu. Dalším formátem je *FictionBook/FB2* [9], který je založený na XML, nepodporuje ale DRM. V minulosti se rovněž hojně používaly formáty společnosti Microsoft - *DOC*, *RTF* a *LIT* [3, 28, 18].

Jedním z nejrozšířenějších formátů vůbec je také formát *PDF* [25], který má velmi širokou podporu napříč platformami, aplikacemi a zařízeními. Pro čtení na zařízeních s menším displejem, jako jsou mobilní telefony, tablety a čtečky elektronických knih ale není příliš vhodný kvůli svému fixnímu formátování dokumentu na stranách. Vyjma uvedených formátů existují desítky dalších [10], jejich rozšíření je ale nízké. Mnohé mají také jen speciální využití.

Kapitola 3

Návrh

V předchozí kapitole byl popsán standard Epub. V této kapitole bude vytvořen návrh knihovny, s rozložením tříd do logických celků, které budou v souladu se standardem Epub souboru. Kapitola bude členěna do jednotlivých částí podle programových tříd implementujících knihovnu, která je předmětem této práce.

V sekci 3.1 bude probrána třída `Book`, která zastřešuje práci s nejvyššími metodami knihovny. Sekce 3.2 bude pojednávat o třídě starající se o rozbalování a následné zabalení souborů do archivu Epub. V dalších sekcích budou popsány další třídy zpracovávající jednotlivé soubory v archivu formátu Epub. popis jednotlivých sekcí bude doplněn o diagramy tříd dle standardu UML2 [37].

3.1 Třída Book

Jak bylo popsáno v úvodu, takto sekce se zaměří na třídu `Book`.Tato třída je výchozím bodem zpracování souboru a zároveň vstupním bodem navrhovaného API. Třída se stará o veškerou práci s archivem a následné zpracování datových souborů od jejich parsování, po jejich opětovné ukládání a generování nových souborů. Rovněž v sobě zahrnuje vnořené třídy, které slouží k uchovávání a zpracování sémanticky příbuzných dat vyčtených z datových souborů. Data jsou v těchto třídách a jejich vnořených třídách uchovávaná hierarchickým způsobem odpovídajícím původní hierarchii souborů. Rozhraní, které třída Book a jejich vnořené třídy nabízejí tak slouží ke strukturovanému přístupu k datům a jejich zpracování. Třída `Book` rovněž provádí kontroly konzistence uložených dat a jejich vztahů.

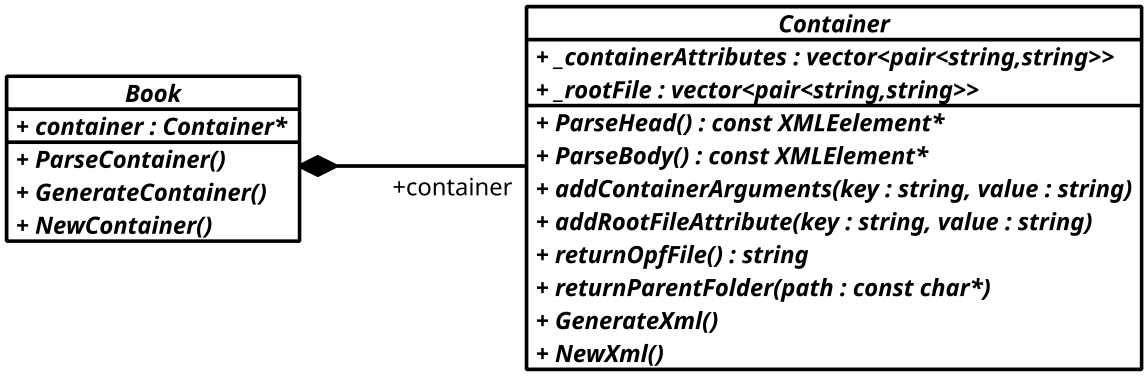
3.2 Zpracování Epub souboru

Třída obsahuje metody, pro extrakci souborů z archivu a jejich uložení do pracovního adresáře, kde budou prováděny všechny editace. Po ukončení práce se soubory třída vygeneruje nový archiv s editovanými daty. Během extrakce se názvy souborů ukládají do seznamu, který bude později použit pro testování konzistence souborů. Testování konzistence bude probráno v dalších sekcích.

3.3 Zpracovaní souboru container

Zpracování souboru `container` je použita stejnojmenná třída `container`. Třída obsahuje datové struktury typu `vector` se spárovanými hodnotami `string` pro uchování argumentů

a hodnot hlavičky kořenového elementu a seznamu opf souborů. Seznam opf souborů je dále použit jako zdroj souborů při zpracovávání .opf souborů, kterému bude věnována další sekce.



Obrázek 3.1: Diagram Třídy Container

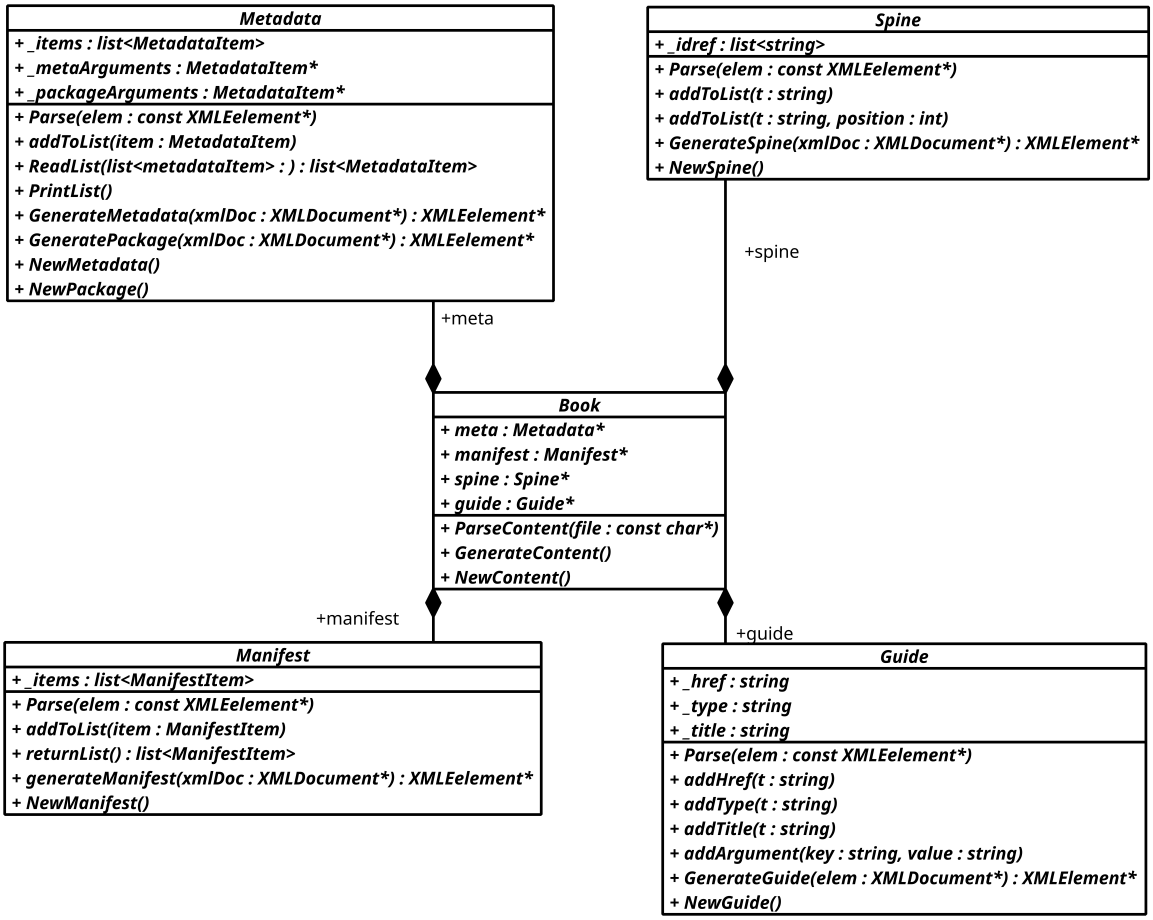
3.4 Zpracování opf souboru

Soubor se skládá ze čtyř částí a proto je i návrh rozdělen do čtyř tříd `Metadata`, `Manifest`, `Spine` a `Guide`, které reflektují strukturu elementů v souboru .opf uchovávající sémanticky různá data jak bylo popsáno v předcházející kapitole. Toto rozvržení struktury tříd umožňuje využít rekurzivní přístup při zpracování čtených elementů, stejně jako při jejich generování. Tento způsob práce s elementy je výhodný pro svou jednoduchost přirozené zachování struktury dokumentu při práci s jeho jednotlivými datovými strukturami. V logice přístupu se tento princip přibližuje principům zpracování XML dokumentů pomocí metody DOM. Každá z těchto tříd uchovává data z elementu odpovídajícího jména. Tyto třídy obsahují hlavní metody `New` pro vytvoření prázdné šablony, `Parse` sloužící ke zpracování existujícího elementu a `Generate` k následnému vytvoření souboru .opf, jak bylo popsáno výše, tyto metody využívají rekurzivního zanořování.

Kromě těchto metod třídy obsahují třídy, které nejsou v diagramu popsány. Třída `Metadata` obsahuje seznam objektů `MetadataItem`, kde každý objekt obsahuje hodnotu a atributy jednoho elementu metadat. Třída `Manifest` zahrnuje seznam objektů `Manifestitem`, obsahující hodnoty `href` s cestou a názvem souboru obsaženého v archivu, `id` daného souboru, se kterým se dále pracuje a `media-type` udávající typ datového média. Všechny tyto metody jsou zastřešeny metodami ve třídě `Book`, jenž slouží k načtení souboru a následnému rozdělení na logické celky, které je možné delegovat ke zpracování konkrétním metodám daných tříd.

3.5 Zpracování souboru toc.ncx

Jak bylo popsáno výše, toc soubor zobrazuje hierarchickou strukturu dokumentu a slouží k rychlé navigaci v dokumentu. Soubor je složen ze tří části, které byly reflektovány při návrhu tříd `TocNcx`, `TocHead`, `TocNavMap` a využívají podpůrných tříd `TocMetaItem` sloužící pro ukládání metadat z hlavičkového elementu a `TocItem` využívaný k uložení jednotlivých položek obsahu dokumentu obsahující `id` dané sekce, `playOrder` udávající pořadí v jakém jsou kapitoly řazeny v dokumentu, `text` název dané kapitoly a `src` odkaz na soubor nebo

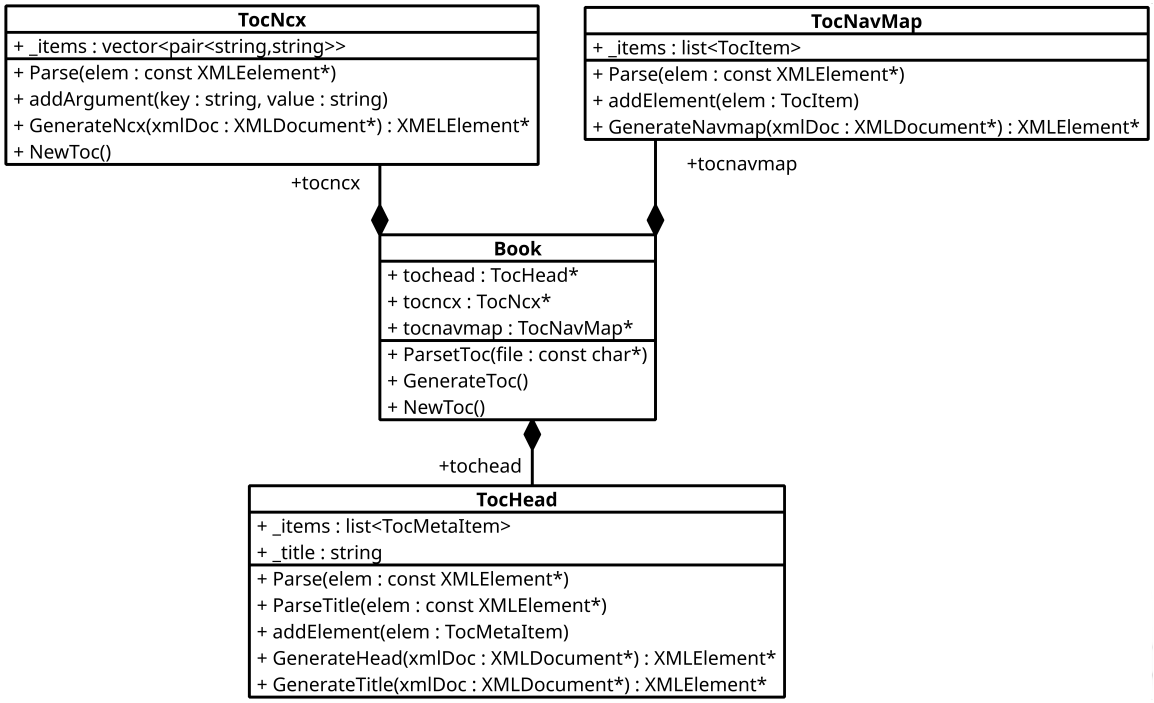


Obrázek 3.2: Diagram tříd určených ke zpracování dat ze souboru opf.

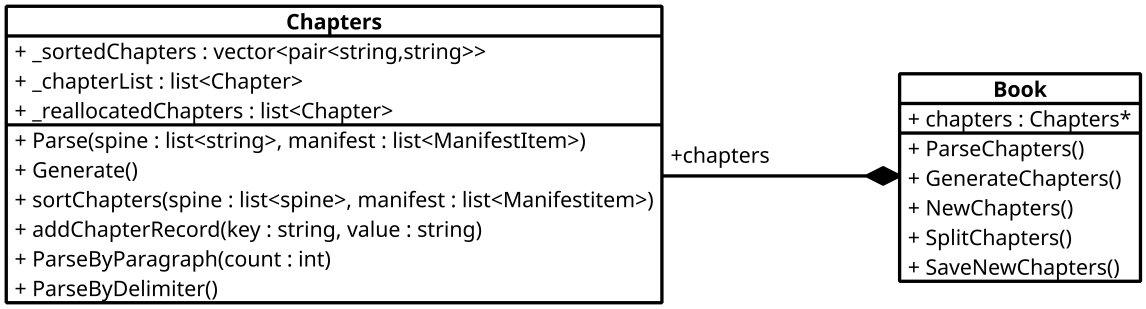
část souboru s daným obsahem. Stejně jako další třídy na této úrovni, i třídy pro práci s toc souborem obsahují metody `Parse`, `Generate` a `New`, jenž jsou zastřešeny metodami z třídy `Book`, které rozdělí soubor na jednotlivé elementy a ty následně předá ke zpracování konkrétním metodám. Každá z uvedených tříd pracuje se stejnojmenným elementem v souboru.

3.6 Zpracování textového obsahu

Soubory s textovým obsahem jsou strukturovány do podoby `html` souboru. Z toho důvodu je i návrh zpracování obsahu rozdělen do částí podle elementů v souboru `head` a `body`, které jsou zpracovávány pomocí metod v tříd `Chapters`. Třída `Chapters` slouží ke zpracovávání textového obsahu a udržování konzistence nad všemi textovými soubory. O udržení dat z jednotlivých souborů se starají pomocné třídy `Chapter`, `ChapterHead` a `ChapterBody` jenž uchovávají veškeré informace o daném souboru, data z hlavičky a těla spolu s názvem souboru.



Obrázek 3.3: Diagram třídy Toc.



Obrázek 3.4: Diagram třídy Chapters.

3.7 Automatické dělení souboru do kapitol

Z důvodů automatického převodu nejrůznějších textových formátů do formátu Epub vzniká nekonzistence takto vytvořených souborů, jenž obvykle vytváří jeden nebo několik málo souborů s textovým obsahem a jeden element v souboru `toc` s odkazem na začátek publikace.

V této sekci bude probrán návrh funkčnosti automatického dělení kapitol implementovaného v knihovně. Tato funkčnost byla do návrhu zahrnuta z důvodu rozdělení textu z jednoho nebo několika málo souborů, bez jakékoliv obsahové struktury, do samostatných souborů s vnitřní obsahovou strukturou v souboru `toc`.

V dalších částech sekce bude probrán způsob načítání textového obsahu, způsob vyhledávání řetězců v textu pro dělení obsahu do kapitol s následným ukládáním do připravených struktur a nahrazení původní struktury dokumentu.

V rámci knihovny byly implementovány dvě metody pro rozdělení kapitol `ParseByDelimiter` dělicí textový obsah podle oddělovače a `ParseByKeyWord` dělící obsah dle klíčového slova. Obě metody jsou dále popsány v sekcích 3.7.2.

3.7.1 Načítání a zpracování obsahu

V této části budou probrány obecné kroky, které využívají obě metody pro dělení kapitol. Rozdíly mezi metodami budou popsány níže v sekci 3.7.2.

Textový obsah nemusí ležet jen v jednom souboru a proto je jako první úkon načtení obsahu do jedné textové struktury s vyjímkou souboru `titlepage`. Soubor `titlepage` obsahuje často jen přebal publikace nebo informace nevztahující se k samotnému obsahu, proto je tento soubor zachován v původní podobě.

Poté co je textový obsah načten do struktury, jsou použity regulární výrazy pro vyhledání a rozdělení obsahu na jednotlivé kapitoly.

Ve chvíli kdy je pomocí regulárního výrazu nalezena shoda, je daná část textu oddělena a uložena do připraveného objektu, obsahující hlavičkové data s nově vygenerovaným jménem souboru. Takto proběhne vyhledávání a dělení nad celým zbylým textovým obsahem. Objekty s textem nijak nezasahují do původní struktury dat, proto je možné využívat dělící metody s různými parametry nad původními daty a tím dosáhnout nejoptimálnějšího řešení.

Ve chvíli kdy je vygenerována vhodná struktura dat je použita metoda `SaveNewChapters`, která nahradí původní obsah objektů `Chapter` nově vytvořenými objekty a upraví záznamy v souborech `.opf` a `toc`. Po dokončení je možné přistoupit ke generování nového souboru Epub.

Ve výsledném pohledu je vytvořen nový soubor Epub obsahující rozdělený text, spolu s nimi jsou upraveny i záznamy `manifest` a `spine` v souboru `.opf` obsahující nové položky s atributy obsahující název souboru a jeho id. A také je upraven obsah souboru `toc`. To jak soubor toc vypadal před dělením kapitol je znázorněno na schématu 3.1 a následně po provedení dělení na schématu3.2. Jak lze vidět soubor `titlepage.xhtml` byl ponechán v původní podobě, jak bylo vysvětleno dříve.

```
1  <navMap>
2    <navPoint id="u4f24aa22-cae2-414b-99b4-11addce6028a" playOrder="1">
3      <navLabel>
4        <text>Spustit</text>
5      </navLabel>
6      <content src="titlepage.xhtml"/>
7    </navPoint>
8    <navPoint id="u4f24aa22-cae2-414b-99b4-458ddbc8aac9" playOrder="2">
9      <navLabel>
10       <text>Kniha</text>
11     </navLabel>
12     <content src="kniha.html"/>
13   </navPoint>
14 </navMap>
```

Schéma 3.1: Soubor toc před rozdělením kapitol

```
<navMap>
    <navPoint id="u4f24aa22-cae2-414b-99b4-11addce6028a" PlayOrder="1">
          <navLabel>
              <text>Spustit</text>
          </navLabel>
          <content src="titlepage.xhtml"/>
      </navPoint>
      <navPoint id="12338131538448643834" PlayOrder="2">
          <navLabel>
              <text>1.kapitola</text>
          </navLabel>
          <content src="html1.html"/>
      </navPoint>
      <navPoint id="12338131538448643835" PlayOrder="3">
          <navLabel>
              <text>2.kapitola</text>
          </navLabel>
          <content src="html2.html"/>
      </navPoint>
      ...
</navMap>
```

Schéma 3.2: Soubor toc po rozdělení kapitol

3.7.2 Metody ParseByDelimiter a ParseByKeyWord

V předcházející sekci byl popsán obecný postup pro dělení kapitol. V této sekci budou popsány přístupy k regulárním výrazům a způsob vyhledávání v textu pomocí zadaných regulárních výrazů.

Metoda `ParseByDelimiter` má jako argumenty `delimiter`, podle kterého se bude následný text dělit a kladný číselný údaj `count` udávající kolik částí rozděleného textu se má spojit do výsledného textu. Jako oddělovač je očekávána otevírací html značka s volitelnými atributy a uvozena závorkami, v případě absence závorek, které jsou do regulárního výrazu přidány. Pokud nejsou zadány žádné parametry, jsou použity výchozí parametry pro dělení textu dle html značky `<p>` sloužící k uvozování odstavců a proměnná `count` s hodnotou 10, jinak řečeno bude text rozdělen do souborů po 10 odstavcích.

Oproti tomu metoda `ParseByKeyWord` očekává jako oddělovací argument `keyWord` textový řetězec obsažený v textu a stejně jako předcházející metoda očekává kladný číselný údaj `count`. Jak bylo řečeno, jako oddělovač slouží textový řetězec, ke kterému je v regulárním výrazu přidána předcházející otevírací html značka a následující ukončující značku, aby byla udržena konzistence souboru. Obdobně jako u metody `ParseByKeyWord` obsahuje i tato metoda implicitní hodnoty argumentů. V případě neuvedení parametrů je použit vyhledávací regulární výraz obsahující řetězce kapitola a chapter s hodnotou proměnně `count` 1.

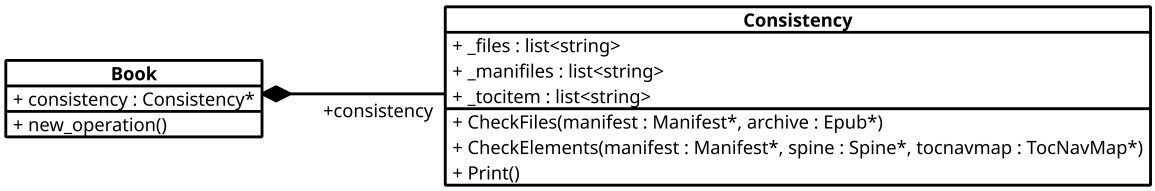
3.8 Konzistence souborů

V předcházejících části byl popsán návrh zpracování archivu, souborů a zastřešující třídy `Book`. V dnešní době mohou být soubory Epub editovány a generovány nejrůznějšími programy, které byly popsány v sekci 2.3.2 nebo mohou být upravovány samotným uživatelem. Z toho důvodu je do knihovny zahrnuta základní kontrola konzistence souborů v archivu Epub implementována ve třídě `Cosistency`.

Třída `Consistency` byla navržena tak, aby kontrolovala jak existenci souborů v archivu, tak i základní povinné elementy, které soubory musí obsahovat.

Kontrola souborů obsažených v archivu probíhá pomocí metody `checkFiles`, která porovnává záznamy v elementu `manifest` se seznamem souborů získaných při rozbalování archivu do pracovního adresáře. Spolu s tím je provedena kontrola souborů, jenž dle specifikace mají udanou pevnou cestu ve struktuře archivu, soubor `mimetype` je uložen kořenu archivu a soubor `container.xml` je obsažen ve složce `META-INF`.

Kromě kontroly uložení souborů v archivu je prováděna i kontrola obsahu jednotlivých souborů a vyhledávání nekonzistentních elementů. Tuto kontrolu provádí metoda `checkElements`, která v elementech `manifest` a `spine` provádí porovnání atributů `id` kde vyhledává rozdílné nebo chybějící záznamy. Spolu s tím provádí i kontrolu záznamů v elementu `manifest` spolu s obsahem `NavMap` elementu v souboru `toc`, kde kontroluje zda se elementy `NavPoint` neodkazují na neexistující soubory. Výstupem této kontroly je výpis na standardní výstup.



Obrázek 3.5: Diagram třídy Consistency.

Kapitola 4

Implementace

V předcházející kapitole byl popsán návrh implementace knihovny pro Epub. V této kapitole budou popsány externí knihovny použity při implementaci. Jedná se o knihovny `LibZip`[17] pro práci s archivem a `TinyXML2.`[30] sloužící pro zpracovávání XML souborů. Rovněž budou popsány jednotlivé funkce a jejich použití v knihovně. Knihona byla vyvíjena v jazyce C++ a testována na operačním systému Linux Mint 18.1 64-bitové architektury. Při vývoji byly využity nástroje a technologie dostupné v této linuxové distribuci. Knihovna byla vyvíjena primárně pro linuxové operační systémy.

4.1 LibZip

Tato kapitola bude pojednávat o knihovně `libzip`. Knihovna je open source projekt pro práci s archivy zip. Knihovna slouží pro čtení, zápis a vytváření archivu zip. Při zápisu knihovna dovoluje nahrazování současných souborů nebo přidávání nových. Knihovna je napsána v jazyce C a je dostupná v linuxových repozitářích v balíku `lipzip-dev`. Autory knihovny jsou Dieter Baron a Thomas Klausner.

Pro práci se soubory Epub, byly použity ve třídě `EPUB` funkce z knihovny libzip pro otevření archivu, načtení souborů do pracovního adresáře a zpětné uložení souborů do archivu. Klíčovou funkcí pro načítání souborů z archivu je `zip_fread`, která načítá sekvence bytů do bufferu, z kterého se pomocí stadartní metody `fwrite` ukládají do souborů v pracovním adresáři. Další metody použité pro extrakci dat z archivu jsou popsány v tabulce 4.1.

Funkce	**Popis**
`zip_open`	Sloužící pro otevření archivu, funkce vrací ukazatel na strukturu zip, použitou pro manipulaci s archivem.
`zip_get_num_entries`	Vrací počet souborů v archivu.
`zip_stat_index`	Získává informace o souboru v archivu na pozici index.
`zip_fread`	Načítá byty ze souboru v archivu do bufferu, z kterého se zapisují do souboru v pracovním adresáři.
`zip_fclose`	Funkce uzavře archiv a uvolní alokovanou paměť.

Tabulka 4.1: Funkce použité v metodě `openZip`

Obdobně jako při načítání souborů z archivu je i při ukládání nejdříve funkcí `zip_souce_file` vytvořen zdroj dat ze souboru a výstupem je struktura `zip_source`, která je následně přidána metodě `zip_file_add`, která v archivu vytvoří nový soubor a vrátí index souboru v

Funkce	Popis
`zip_source_file`	Slouží k vytvoření zdroje dat pro archiv ze souboru.
`zip_file_add`	Funkce přidá soubor do archivu a vrací index nového souboru v archivu.
`zip_close`	Uzavře archiv a uvolní alokovanou paměť. Pokud jsou v archivu provedeny jakékoliv změny, jsou nejdříve zapsány.

Tabulka 4.2: Funkce použité v metodě `createZip`

archivu. Výčet všech funkcí použitých pro vytváření archivu je v tabulce 4.2.

4.2 TinyXML2

TinyXML2 je jednoduchý parser XML dokumentů, který z načteného souboru vytváří DOM strukturu, kterou je možné číst, upravovat a následně ukládat. Autorem knihovny je Lee Thomason.

Knihovna TinyXML2 byla v knihovně použita pro zpracování, procházení a vytvoření všech XML souborů v archivu. Při zpracovávání i vytváření souborů byly použity objekty `XMLDocument`, `XMLELement` a `XMLAttribute` a jejich metody, které jsou popsány v tabulkách 4.3, 4.4 a 4.5.

Třída	Popis
`XMLDocument`	K sobě váže veškerou funkcionalitu. Může být uložen, nahrán nebo vytištěn na obrazovku.
`XMLElement`	Je kontejnerová třída. Obsahuje hodnotu, jméno elementu, dále může obsahovat další elementy, text, komentáře a volitelný počet atributů.
`XMLAttribute`	Je obsažen ve třídě `XMLElement` a je složen z páru klíč hodnota, každý atribut musí být ve třídě unikátní.

Tabulka 4.3: Třídy použité pro zpracování XML souborů

Metoda	Popis
`LoadFile`	Metoda třídy `XMLDocument`, která načítá soubor XML.
`FirstChildElement`	Vrací `XMLElement` obsahující prvního potomka, nebo potomka elementu s daným jménem.
`NexSiblingElement`	Vrací prvního pravého sourozence aktuálního elementu.
`FirstAttribute`	Vrací první atribut aktuálního elementu.

Tabulka 4.4: Metody použité pro načtení souborů a procházení strukturou DOM

4.3 C++

Tato část bude popisovat jazyk `C++`, ve kterém je napsána tato knihovna. Jazyk `C++` je programovací jazyk vyvinut v `Bellových laboratořích AT&T` a je rozšířením jazyka C. Jazyk `C++` patří mezi nejrozšířenější programovací jazyky.

Metoda	Popis
`NewElement`	Vytvoří nový element spjatý s dokumentem.
`InsertFirstChild`	Přidá element jako prvního potomka.
`InserLastChild`	Obdoba `InsertFirstChild` s tím rozdílem, že vloží element jako posledního potomka.
`SetAttribute`	Nastaví atribut elementu jako pár klíč hodnota.
`SaveFile`	Uloží strukturu DOM do souboru.

Tabulka 4.5: Metody použité pro generování a ukládání souborů

4.4 GCC

`GNU Compiler Collection` zkráceně `GCC`[11] je sada kompilátorů podporující řadu programovacích jazyků jako je `C`, `C++`, `Fortran`. Kompilátor slouží pro překlad zdrojových kódů zapsaných v programovacím jazyce do jiného jazyka nejčastěji do strojového kódu.

Pro překlad kódu knihovny byl použit překladač `g++` ve verzi 5.4.0. Který slouží jako kompilátor jazyka `C++` v sadě `GCC`.

4.5 GIT

`GIT`[12] je široce rozšířený verzovací systém vyvinut pro účely vývoje Linuxového jádra `Linusem Torvaldsem` v roce 2005. V dnešní době je šířen pod licenci `GPL`[15], jedná se tedy o svobodný software a na základě původního projektu vznikla řada nových odnoží z nichž je nejznámější `GitHub`[13]. V rámci této práce pro účely verzování byl použit projekt `GitLab`[14].

Kapitola 5

Rozhraní knihovny

V předcházejících kapitolách byl popsán návrh a implementace knihovny. Tato kapitola popisuje její aplikační rozhraní (API). Popisuje tedy metody určené pro použití uživatelem knihovny při implementaci aplikace pracující s formátem Epub. Nezaměřuje se však detailně na strukturu knihovny, ta byla popsána v kapitole 3, ani na detailní popis implementace. Ten je přítomen v dokumentaci na přiloženém CD. V druhé části kapitoly jsou popsány příklady užití těchto metod pro několik standardních scénářů použití navržené knihovny.

5.1 Epub

Tato sekce bude popisovat metody pro práci s archivem Epub, z hlediska jeho rozbalení a opětovného sbalení. Obě metody pro rozbalení a sbalení jsou uvedené v tabulce 5.1. Tyto metody jsou využity v metodách `OpenBook` a `GenerateBook` ve třídě `Book`.

Metoda	Popis
`OpenZip`	Metoda slouží pro extrakci vstupního souboru do pracovní složky. Jako jediný parametr vyžaduje cestu k souboru `epub`.
`CreateZip`	Metoda, která vytvoří nový archiv Epub ze souborů v pracovním adresáři. Jako parametr vyžaduje jméno výsledného souboru.

Tabulka 5.1: Tabulka metod třídy Epub sloužící pro práci s archivem.

5.2 Book

Třída `Book` zastřešuje všechny třídy v knihovně jak ilustruje třídni diagram . Třída `Book` zajišťuje činnost vztahující se k práci s archívem a zpracování souborů, také se stará o práci s kapitolami a metadaty.

5.2.1 Metadata

Třída `Metadata` obsahuje informace o publikaci vyčtené ze souboru `opf`. Metody pracující s nejdůležitějšími metadaty zahrnujícími název publikace, jméno autora a jazyk jsou popsány v tabulce 5.4.

Metoda	Popis
`OpenBook`	Metoda slouží pro extrakci vstupního souboru do pracovní složky a provedení zpracování souborů do připravených struktur.
`GenerateBook`	Metoda, která vytvoří nový archiv Epub ze souborů v pracovním adresáři.
`NewBook`	Metoda slouží pro naplnění struktur hlavičkovými daty.
`SetTitle`	Metoda zapisuje nový název dokumentu do struktur.
`GetTitle`	Metoda vrací string obsahující název dokumentu.
`CheckConsistency`	Metoda provede základní kontrolu konzistence obsahu archivu Epub.

Tabulka 5.2: Tabulka metod třídy Book pro práci se soubory v pracovním adresáři.

Metoda	Popis
`SplitChapter`	Metoda rozděluje textový obsah v seznamu kapitol, podle zadaných parametrů.
`ReturnDividedChapters`	Vrátí seznam nově rozdělených kapitol v seznamu objektu chapter.
`SaveNewChapters`	Přepíše původní struktury kapitol a nahradí je obsahem rozdělených kapitol ze seznamu.
`AddChapter`	Metoda slouží pro přidání nové kapitoly do struktur. Implicitně je kapitola přidána na konec dokumentu.
`RemoveChapter`	Metoda slouží k odstranění kapitoly a všech jejích záznamů ze všech struktur.
`RenameChapter`	Metoda změní název kapitoly v souboru `toc`.

Tabulka 5.3: tabulka metod třídy Book pro zpracování obsahu.

5.2.2 Manifest

Třída `Manifest` sdružuje objekty `ManifestItem` do seznamu, stejně jak tomu je ve standardu Epub. Tabulka 5.5 popisuje práci nad seznamem a tabulka 5.6 popisuje práci s jednotlivými položkami objektu `ManifestItem`.

5.2.3 TocNavMap

Obdobně jako `Manifest` sdružuje objekty do seznamu, tak i třída `TocNavMap` obsahuje seznam objektů třídy `TocItem` obsahující informace o kapitolách. Tabulka 5.7 zobrazuje metody pro práci nad seznamem a tabulka 5.8 práci s objekty seznamu.

5.3 Práce s knihovnou

V této části budou popsány dva příklady využití knihovny, za pomoci metod popsaných v této kapitole. V první sekci 5.3.1 bude ilustrován příklad pro dělení existujícího archivu do kapitol a ve druhé sekci5.3.2 bude popsán postup vytvoření nového archivu a jeho naplnění kapitolami a nastavení jména publikace.

Metoda	Popis
`returnTitle`	Metoda vrací řetězec typu string s názvem dokumentu obsaženým v metadatech souboru `opf`.
`returnAuthor`	Metoda vrací řetězec typu string se jménem autora, obsaženým v metadatech souboru `opf`.
`returnLanguage`	Metoda vrací řetězec typu string s jazykem, ve kterém je dokument napsán.
`setTitle`	Metoda změní název dokumentu v metadatech souboru `opf`.
`setCreator`	Metoda změní jméno autora v metadatech souboru `opf`.
`setLanguage`	Metoda změní jazyk dokumentu v metadatech souboru `opf`.

Tabulka 5.4: Tabulka metod třídy Metadata.

Metoda	Popis
`addToList`	Metoda přidá objekt ManifestItem do seznamu elementů.
`returnList`	metoda vrátí seznam elementů ManifestItem.

Tabulka 5.5: Tabulka metod třídy Manifest.

5.3.1 Použití metody pro dělení kapitol

První příklad popisuje otevření existujícího dokumentu, použití metody pro dělení kapitol a následného vytvoření nového archivu.

Pro extrakci a zpracování souborů z archivu Epub je použita metoda `OpenBook`, která v sobě sdružuje metody `OpenZip` sloužící pro otevření archivu a extrakci souborů do pracovního adresáře a sadu metod `Parse`, zpracovávající jednotlivé soubory dle jejich funkce do připravených struktur. Pro kontrolu obsahu jsou vypsány informace o názvu dokumentu a jeho autorovi.

Následně je použita metoda `SpliChaptersbyDelimiter` pro dělení obsahu dokumentu na kapitoly. Metoda nemá zadané žádné parametry, proto jsou použity výchozí hodnoty dělení, kdy je použita jako oddělovač html značka `<p>` a části jsou sdružovány po 10 odstavcích. Ve chvíli kdy jsou kapitoly rozděleny, je možné metodou `SaveNewChapters` přepsat stávající rozdělení kapitol na nově vytvořené pomocí předcházející metody.

Jako poslední krok je provedení generování obsahu metodou `GenerateBook`, jenž ze struktur vygeneruje soubory v pracovním adresáři a následně tyto soubory zabalí do výsledného souboru Epub.

5.3.2 Vytvoření nové knihy

Druhý příklad popisuje vytvoření šablony, vložení několika kapitol a následné vytvoření archivu.

Vytvoření nové šablony pak probíhá pomocí metody `NewBook`, kdy metoda naplní struktury nejdůležitějšími daty pro vytvoření archivu. Následně jsou metodami `setTitle`, `setAuthor` a `setLanguage` uložena základní metadata o dokumentu.

V tuto chvíli šablona obsahuje pouze název dokumentu a jméno autora, proto jsou pomocí opakovaného volání metod `addChapter` a `RenameChapter` do dokumentu vloženy kapitoly a pojmenovány. První tři kapitoly jsou postupně vkládány na konec dokumentu, ale kapitola *úvod*, obsahuje jako druhý argument pozici, kam má být vložena.

Metoda	Popis
`getHref`	Metoda vrátí cestu k souboru z vnitřní proměnné.
`getId`	Metoda vrátí id souboru z vnitřní proměnné.
`getMediaType`	Metoda vrátí typ media souboru z vnitřní proměnné.
`setHref`	Metoda uloží cestu k souboru do vnitřní proměnné.
`setId`	Metoda uloží id souboru do vnitřní proměnné.
`setMediaType`	Metoda uloží typ media souboru do vnitřní proměnné.

Tabulka 5.6: Tabulka metod třídy `ManifestItem`.

Metoda	Popis
`addElement`	Metoda přidá nový záznam to seznamu elementů.
`returnList`	Metoda vrátí seznam elementů `TocItem`.

Tabulka 5.7: Tabulka metod třídy `TocNavMap`.

```
1  Book book;
2  book.OpenBook("kniha.epub");
3
4  cout << book.metadata.returnTitle() << endl;
5  cout << book.metadata.returnAuthor() << endl;
6
7  book.SplitChaptersByDelimiter();
8
9  book.SaveNewChapters();
10
11 book.Generate("kniha1.epub");
```

Schéma 5.1: Ukázka otevření souboru `kniha.epub` s použitím metody pro rozdělení kapitol a uložení výsledné struktury do nového souboru `kniha1.epub`.

Posledním krokem je vytvoření nových soubor v pracovním adresáři a vytvoření výsledného archivu epub za pomoci metody `GenerateBook`.

Metoda	Popis
`getSrc`	Metoda vrátí cestu k souboru z vnitřní proměnné.
`getId`	Metoda vrátí id kapitoly z vnitřní proměnné.
`getText`	Metoda vrátí název kapitoly z vnitřní proměnné.
`setSrc`	Metoda uloží cestu k souboru do vnitřní proměnné.
`setId`	Metoda uloží id kapitoly do vnitřní proměnné.
`setText`	Metoda uloží název kapitoly do vnitřní proměnné.

Tabulka 5.8: Tabulka metod třídy ManifestItem.

```
1  Book book;
2  book.NewBook();
3
4  book.setTitle("Název dokumentu");
5  book.setAuthor("Autor dokumentu");
6  book.setLanguage("cs");
7
8  cout << book.metadata.returnTitle() << endl;
9  cout << book.metadata.returnAuthor() << endl;
10
11 book.addChapter("Text první kapitoly");
12 book.RenameChapter("Popis problematiky",1);
13
14 book.addChapter("Text druhé kapitoly");
15 book.RenameChapter("Návrh řešení",1);
16
17 book.addChapter("Text třetí kapitoly");
18 book.RenameChapter("Testování",1);
19
20 book.addChapter("Úvod",1);
21 book.RenameChapter("Úvod",1);
22
23 book.Generate("kniha1.epub");
24
```

Schéma 5.2: Ukázka vytvoření nového dokumentu a vložení několika kapitol.

Kapitola 6

Instalace

V této kapitole bude popsáno získání zdrojových kódu, instalace knihovny `ZipLib` určenou pro práci s archívy, vložení knihovny Epub do vlastního projektu a následný překlad do spustitelné podoby.

Zdrojové kódy, této práce jsou dostupné z repozitáře verzovacího systému `Gitlab`, jenž je založen na projektu `Git`. Zdrojové kódy je možno přímo stáhnout z webové stránky[1] nebo použití linuxového terminalového příkazu v bashi:

```
git clone  https://gitlab.com/gomess/Bakalarka
```

Po získání zdrojových kódů je nutné stáhnout `dev` balíček, který obsahuje hlavičkové soubory knihovny `LipZip`. Tuto knihovnu lze stáhnout pomocí linuxové aplikace Správce softwaru nebo linuxovým terminálovým příkazem v bashi:

```
sudo apt-get install libzip-dev
```

Ve chvíli kdy jsou splněny všechny závislosti, je možné přistoupit k překladu programu pomocí vestavěného programu `make`. Program `make` přeloží knihovnu spolu s demonstrační aplikací `demo.cpp` do spustitelného souboru `DemonstracniAplikace`.

Pro práci s knihovnou pod operačním systémem `Windows` je nejdříve nutné zkompilovat knihovnu `LibZip` a přidat podporu pro tvar cest k souborům používaným v `OS Windows`. Kompilace může proběhnout pomocí stejných nástrojů a ekvivalentních postupů.

[1] https://gitlab.com/gomess/Bakalarka

Kapitola 7

Demonstrační program

V minulých kapitolách byl popsán návrh a implementace knihovny pro práci s formátem Epub, která se v této kapitole využije pro převod textu v podobě `Markdown` do formátu Epub. V sekci 7.1 bude popsán jazyk Markdown a jeho syntaxe, v další části bude ukázán návrh implementace za pomocí knihovny.

7.1 Jazyk Markdown

Markdown [19] je odlehčený značkovací jazyk, který slouží pro úpravu prostého textu do formátu html za pomoci stejnojmenného nástroje. Markdown je často používán k formátování `readme` souborů, psaní zpráv v internetových diskuzích a vytváření formátovaného textu za pomocí libovolného editoru prostého textu. Jazyk využívá jednoduché formátovací značky k vyznačení nadpisů, seznamů a vkládaní odkazů, obrázků a dalších.

7.1.1 Syntaxe jazyka

Markdown je uzpůsoben k tomu aby byl snadno čitelný a snadno napsatelný. Za tímto účelem je syntaxe zcela tvořena interpunkčními znaky, které byly vybrány tak aby znázorňovaly to, co znamenají.

Nadpisy jsou uvozeny jedním až šesti znaky # odpovídající html značkám `<h1>` až `<h6>`. Pro formátování vlastností textu se používají znaky *, _, ~, ‘ určené pro formátování tučného písma, kurzívy, přeškrtnutý text a bezpatkového písma, přičemž znaky * a _ zastávají stejnou funkci. Ukázka základních značkovacích příkazů je v tabulce 7.1. Ilustrace použití značek je vyobrazeno na schématech 7.1 a 7.2. Kromě formátování textu, lze v jazyce Markdown vytvářet seznamy za pomocí znaků *, - a + sloužící pro vytváření nečíslovaných seznamů nebo kladných číselných hodnot pro vytváření číslovaných seznamů.

7.2 Návrh aplikace

Návrh demonstrační aplikace bude rozdělen do částí, které se skládají z načítání zdrojového souboru a nahrazení značek markdown za html značky, následné vložení do vytvořené šablony pro dokument, vytvoření struktury kapitol a konečně zabalení do kontejneru Epub.

Pro otevření vstupního souboru je použita knihovní funkce `open` třídy `ifstream`, která soubor otevře a zpřístupní jej pro čtení. Soubor je následně načítán po řádcích, ve kterých jsou za pomoci sady regulárních výrazů nahrazovány značky jazyka Markdown za formátovací značky html. Po zpracování veškerého obsahu vstupního souboru je vytvořena nová

Markdown	html značky	Výsledný text
`**tučný text**`	tučný text	**tučný text**
`__tučný text__`	tučný text	**tučný text**
`*kurzíva*`	tučný text	*kurzíva*
`_kurzíva_`	kurzíva	*kurzíva*
`‘text v bezpatkovém fontu‘`	<code>text v bezpatkovém fontu</code>	`text v bezpatkovém fontu`
`~~přeškrtnutý text ~~`	~~přeškrtnutý text~~	~~přeškrtnutý text~~

Tabulka 7.1: Ukázka značkovacích příkazů.

instance třídy `Book`, následně s využitím metody `NewBook` jsou základní struktury naplněny hlavičkovými daty a tím je připravena prázdná šablona, pro vložení textového obsahu.

Ve chvíli kdy je šablona připravena, je vložení formátovaného textu do struktury uskutečněno metodou `addChapter`. Metoda vytvoří nový objekt `Chapter`, který naplní textovým obsahem a vloží záznamy do objektů `manifest`, `spine` a `toc`.

Nicméně v tuto chvíli je dokument koncipován do jednoho souboru, bez možnosti navigace. Tuto situaci řeší metoda `SplitChapters` s argumentem `<h3>`, která vyhledá v textu všechny nadpisy `<h3>` a podle nich vytvoří nové rozložení kapitol. Následně metoda `SaveNewChapters` nahradí původní rozložení kapitol a nahradí záznamy ve strukturách obsahující odkazy na soubory.

Na závěr je použita metoda `GenerateBook` s argumentem určující jméno výsledného souboru. Metoda z objektů vytvoří v pracovním adresáři výsledné soubory, které následně zabalí do archivu daného jména.

Na schématech jsou ukázky textu v jazyce Markdown 7.1 a tento text následně převedený do formátu html 7.2.

```
### Hlavní nadpis

Ukázka textu v jazyce Markdown, kde je použit **tučný text**
spolu s _kurzívou_ a ‘bezpatkovým písmem‘, tyto značku lze
zanořovat do sebe **tučný text s \_kurzívou\_**.

### **První podnadpis**
Značky pro stylizaci písma lze použít i v nadpisech.

V další části je použit oddělovač řádků, použit k oddělení
od nečíslovaného seznamu
---

- první položka
- druhá položka
- třetí položka
```

Schéma 7.1: Ukázka vstupního souboru ve v jazyce Markdown.

```
<h3>Hlavní nadpis</h3>

Ukázka textu v jazyce Markdown, kde je použit <b>tučný text</b>
spolu s <i>kurzívou</i> a <code>bezpatkovým písmem</code>,
tyto značku lze zanořovat do sebe <b>tučný text s <i>kurzívou</i></b>.

<h3><b>První podnadpis</b></h3>
Značky pro stylizaci písma lze použít i v nadpisech.

V další části je použit oddělovač řádků, použit k oddělení
od nečíslovaného seznamu<hr/>

<ul>
<li>první položka</li>
<li>druhá položka</li>
<li>třetí položka</li>
</ul>
```

Schéma 7.2: Ukázka zpracovaného souboru v jazyce HTML.

Kapitola 8

Závěr

Cílem této práce bylo navrhnout a poté implementovat knihovnu pro práci s formátem Epub. Pro dosažení tohoto cíle bylo potřeba prostudovat a analyzovat specifikaci formátu a také řadu publikací v tomto formátu uložených. Mnoho dostupných publikací v tomto formátu totiž vzniká konverzí z jiných formátů, čímž dochází k nekonzistenci, kterou bylo nutno v návrhu a fungování knihovny zohlednit. V návrhu byla rovněž zohledněna vnitřní struktura souborů tohoto formátu tak, aby knihovna přirozeně tuto strukturu dodržovala uspořádáním svých implementačních tříd. Tímto krokem byla zajištěna jednak jednoduchost načítání dat ze souborů i jejich ukládání. Zároveň tím je také umožněn přístup k vnitřním datům formátu až do nejnižší úrovně. Knihovna tak umožňuje editaci dat popisujících strukturu dokumentu a jeho metadat. Tím se odlišuje od jiných knihoven, které takto detailní způsob manipulace s vnitřními daty nenabízejí.

Knihovna kromě otevírání a načítání souborů a analýzy jejich struktury a obsahu dovoluje také data editovat a ukládat je zpět do výchozího souboru. Tato editace zahrnuje přidávání, odebírání a přejmenování kapitol, manipulaci s metadaty a strukturou dokumentu. Knihovna také pomocí svých funkcí nabízí možnost vytvoření úplně nové publikace. Dále je knihovnu možné použít pro kontrolu konzistence souborů, čímž se knihovna také liší od některých knihoven dostupných pro jiné programovací jazyky, které takovou kontrolu nenabízejí. Rozšířením jsou rovněž funkce pro dělení souboru do kapitol. Tyto funkce dovolují rozdělení souborů, které postrádají vnitřní členění a navigaci, do kapitol podle HTML značek nebo klíčových slov specifikovaných uživatelem. Tím umožňují alespoň částečnou nápravu souborů vzniklých nevhodnou konverzí z jiných formátů.

Kromě návrhu a implementace knihovny je v práci také popsané její rozhraní a jeho využití demonstrováno na několika příkladech a také na vytvořené demonstrační aplikace provádějící převod textu ve formátu Markdown do formátu Epub. Dále je také popsána instalace knihovny a kroky nutné pro umožnění její funkčnosti.

Možnosti dalšího rozvoje práce jsou široké. Samotný standard Epub se nadále vyvíjí, přibývají do něj nové prvky a vlastnosti, které bude potřeba v budoucích verzích knihovny reflektovat. Kromě rozšiřování podpory nových standardů jsou zde i další možnosti. Je možné přidat funkce podporující práci s dalšími pomocnými soubory v archivu Epub, jako jsou soubory s kaskádovými styly, fonty, obrázky a další. Rovněž je možné rozšířit možnosti kontroly konzistence a přidat další možnosti automatického dělení dokumentu do kapitol.

Literatura

[1] *Calibre*. [Online; navštíveno 5.5.2017].
URL https://calibre-ebook.com/

[2] *Cool Reader*. [Online; navštíveno 5.5.2017].
URL https://play.google.com/store/apps/details?id=org.coolreader

[3] *DOC*. [Online; navštíveno 1.5.2017].
URL https://msdn.microsoft.com/en-us/library/office/cc313153(v=office.12).aspx

[4] *EbookLib*. [Online; navštíveno 1.5.2017].
URL https://pypi.python.org/pypi/EbookLib/0.16

[5] *Epub 2*. [Online; navštíveno 25.4.2017].
URL http://idpf.org/epub/201

[6] *Epublib*. [Online; navštíveno 1.5.2017].
URL http://www.siegmann.nl/epublib

[7] *eReader*. [Online; navštíveno 1.5.2017].
URL https://wiki.mobileread.com/wiki/PDB

[8] *FBReader*. [Online; navštíveno 5.5.2017].
URL https://fbreader.org/

[9] *FictionBook*. [Online; navštíveno 1.5.2017].
URL https://wiki.mobileread.com/wiki/FB2

[10] *Formáty elektronických knih*. [Online; navštíveno 1.5.2017].
URL https://wiki.mobileread.com/wiki/E-book_formats

[11] *GCC*. [Online; navštíveno 8.5.2017].
URL https://gcc.gnu.org/

[12] *GIT*. [Online; navštíveno 8.5.2017].
URL https://git-scm.com/

[13] *GitHub*. [Online; navštíveno 8.5.2017].
URL https://github.com/

[14] *GitLab*. [Online; navštíveno 8.5.2017].
URL https://gitlab.com/

[15] *GPL*. [Online; navštíveno 8.5.2017].
URL https://www.gnu.org/licenses/gpl-3.0.en.html

[16] *International Digital Publishing Forum*. [Online; navštíveno 25.4.2017].
URL http://idpf.org/

[17] *lipzip*. [Online; navštíveno 1.5.2017].
URL https://nih.at/libzip/

[18] *LIT*. [Online; navštíveno 1.5.2017].
URL https://wiki.mobileread.com/wiki/LIT

[19] *Markdown*. [Online; navštíveno 1.5.2017].
URL https://daringfireball.net/projects/markdown/

[20] *MOBI*. [Online; navštíveno 1.5.2017].
URL https://wiki.mobileread.com/wiki/MOBI

[21] *Open Container Format*. [Online; navštíveno 25.4.2017].
URL http://www.idpf.org/epub/31/spec/epub-ocf.html

[22] *Open EBook Publication Structure*. [Online; navštíveno 25.4.2017].
URL https://www.loc.gov/preservation/digital/formats/fdd/fdd000171.shtml

[23] *Open Packaging Format*. [Online; navštíveno 25.4.2017].
URL http://www.idpf.org/epub/20/spec/OPF_2.0.1_draft.htm

[24] *Open Publication Structure*. [Online; navštíveno 25.4.2017].
URL http://www.idpf.org/epub/20/spec/OPS_2.0.1_draft.htm

[25] *PDF*. [Online; navštíveno 1.5.2017].
URL http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/pdf_reference_1-7.pdf

[26] *Pypi.epub*. [Online; navštíveno 1.5.2017].
URL https://pypi.python.org/pypi/epub/

[27] *Pypub*. [Online; navštíveno 1.5.2017].
URL http://pypub.readthedocs.io/en/latest/index.html

[28] *RTF*. [Online; navštíveno 1.5.2017].
URL https://www.microsoft.com/en-us/download/details.aspx?id=10725

[29] *Sigil*. [Online; navštíveno 5.5.2017].
URL https://sigil-ebook.com/

[30] *tinyXML2*. [Online; navštíveno 1.5.2017].
URL http://www.grinninglizard.com/tinyxml2/

[31] Fahlgren, K.: Fighting Complexity in EPUB 3: Modularization and Delegation. Journal of Electronic Publishing, ročník 14, č. 1, 2011.

[32] Garrish, M.: What is EPUB 3? O'Reilly Media, 2011, ISBN 9781449317713.
URL https://books.google.cz/books?id=yM6SYIRnFpMC

[33] Garrish, M.: Accessible EPUB 3. O'Reilly Media, 2012, ISBN 9781449329327.
URL https://books.google.cz/books?id=hqKFrZHdAc8C

[34] Garrish, M.; Gylling, M.: EPUB 3 Best Practices: Optimize Your Digital Books. O'Reilly Media, 2013, ISBN 9781449329150.
URL https://books.google.cz/books?id=z0zpvR5QNvsC

[35] Kasdorf, B.: EPUB 3: (Not Your Father's EPUB). Information Standards Quarterly, ročník 23, č. 2, Spring 2011: s. 4–11, copyright - Copyright National Information Standards Organization Spring 2011; Document feature - Illustrations; Tables; ; Last updated - 2015-09-04.
URL https://search.proquest.com/docview/1709389423?accountid=17115

[36] Kasdorf, B.: Key Issue: EPUB 3's coming of age. Insights, ročník 26, č. 2, 07 2013: s. 210–213, copyright - Copyright United Kingdom Serials Group (UKSG) Jul 2013; Last updated - 2013-08-03.
URL https://search.proquest.com/docview/1416280103?accountid=17115

[37] Neustadt, I.; Arlow, J.: UML 2 a unifikovaný proces vývoje aplikací, kapitola Notace třídy v jazyce UML. 2016, ISBN 9788025142059, s. 152–163.
URL https://books.google.cz/books?id=zBHqCwAAQBAJ

[38] Pistorius, V.; Kočička, P.: Jak se dělá e-kniha: příprava elektronických publikací ve formátech EPUB a MOBI. Pistorius & Olšanská, 2015, ISBN 9788087855157.
URL https://books.google.cz/books?id=aX-PrgEACAAJ

Přílohy

Seznam příloh

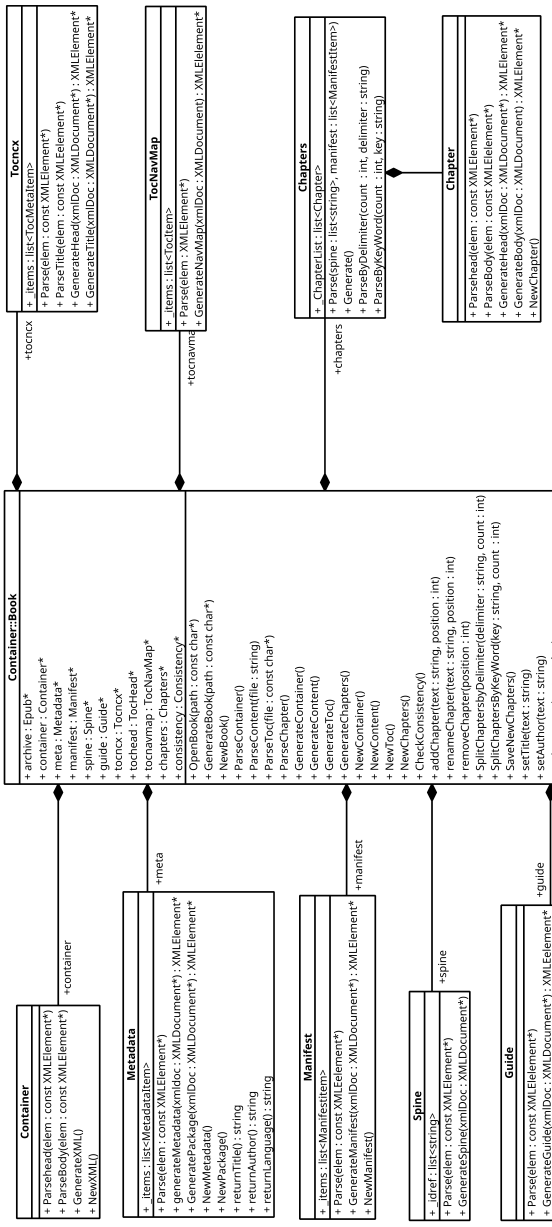
Příloha A

Obsah CD

- `doc` — složka s dokumentací
- `src` — složka se zdrojovými kódy
- `tex` — složka se zdrojovými kódy pro vygenerování technické zprávy
- `Technická zpráva.pdf` — elektronické verze technické zprávy
- `Obsah_cd.txt`
- `instalace.txt` — Návod k instalaci virtuálního systému s aplikací
- `VirtualBox.exe` — aplikace pro práci s virtuálním operačním systémem
- `image.ova` — virtuální disk s operačním systémem

Příloha B

Třídní diagram



Obrázek B.1: Třídní diagram knihovny.