
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**
**BRNO UNIVERSITY OF TECHNOLOGY**

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**

**ÚSTAV INFORMAČNÍCH SYSTÉMŮ**
**DEPARTMENT OF INFORMATION SYSTEMS**


# AUTOMATICKÝ PŘEVOD SLAJDŮ Z FORMÁTU OPEN-DOCUMENT DO REVEAL.JS

**AUTOMATIC SLIDES TRANSLATION FROM OPENDOCUMENT TO REAVEAL.JS**

**BAKALÁŘSKÁ PRÁCE**
**BACHELOR'S THESIS**


**AUTOR PRÁCE** **FILIP SKALICKÝ**
**AUTHOR**

**VEDOUCÍ PRÁCE** **LIBOR POLČÁK, Ing., Ph.D.**
**SUPERVISOR**



**BRNO 2019**

**Vysoké učení technické v Brně**
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

# Zadání bakalářské práce


Student: **Skalický Filip**
Program: Informační technologie
Název: **Automatický převod slajdů z formátu OpenDocument do Reveal.js**
**Automatic Slides Translation from OpenDocument to Reaveal.js**
Kategorie: Informační systémy

Zadání:

1. Seznamte se s formátem OpenDocument, zaměřte se na datový formát vznikající v programu LibreOffice Impress.
2. Seznamte se s frameworkem Reveal.js a použitým značkovacím jazykem (markdown).
3. Navrhněte vhodný způsob automatického převodů slajdů z formátu OpenDocument do značkovacího jazyka Reveal.js (markdown).
4. Návrh implementujte.
5. Otestujte implementaci na vhodné testovací sadě včetně přednášek dodaných vedoucím práce.
6. Práci vyhodnoťte a navrhněte možná zlepšení.

Literatura:

- D. Tidwell. XSLT, O'Reilly Media; Second edition Edition, 2008. ISBN 978-0596527211
- "ISO/IEC 26300-1:2015 - Information technology - Open Document Format for Office Applications (OpenDocument) v1.2 - Part 1: OpenDocument Schema".

Pro udělení zápočtu za první semestr je požadováno:

- Vývoj aplikace bude probíhat iterativně metodologií *Feature-driven development*. Student bude vedoucímu práce předkládat funkční aplikaci ve 2-týdeních vývojových cyklech, každý bude zaměřený na konkrétní prvky prezentace. Pro udělení zápočtu je nutné demonstrovat převod alespoň jedné přednášky předmětu WAP použité v roce 2018, tedy mít rozpracované body 1-4 zadání.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Polčák Libor, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 29. října 2018

# Abstrakt

Práce řeší problematiku automatického převodu prezentace ve formátu OpenDocument do značkovacího jazyka Markdown, který se následně prezentuje pomocí frameworku Reveal.js. Výsledná aplikace pomocí úvodního nastavení, převede danou prezentaci do jazyka Markdown a následně vypíše informace o převodu. Prezentaci je pak možné pomocí Reaveal.js spustit. Obsahem práce je popis implementace aplikace společně s komplikacemi, které při převodu nastaly a jejich následné řešení. Aplikace je testována převodem vybraných prezentací.



# Abstract

The thesis deals with the issue of automatic conversion of OpenDocument presentation into Markdown language, which is then presented using Reveal.js framework. The application uses the initial setup, converts the presentation into Markdown, and then outputs the conversion information. The presentation can then be run using Reaveal.js. The content of this thesis is the description of the implementation of the application, describtion of the complications during the conversion and their solution. The application is tested by converting selected presentations.



# Klíčová slova

OpenDocument, prezentace, XML, Reveal.js, Markdown, XSLT, Extensible Stylesheet Language Transformations

# Keywords

OpenDocument, presentation, XML, Reveal.js, Markdown, XSLT, Extensible Stylesheet Language Transformations



# Citace

SKALICKÝ, Filip. *Automatický převod slajdů z formátu OpenDocument do Reveal.js*. Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Libor Polčák, Ing., Ph.D.

# Automatický převod slajdů z formátu OpenDocument do Reveal.js

## Prohlášení

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

.....
Filip Skalický
15. května 2019

## Poděkování

Velice rád bych poděkoval vedoucímu práce Ing. Liborovi Polčákovi, Ph.D za jeho podporu, cenné rady a připomínky při plnění práce.

# Obsah

# Kapitola 1

# Úvod

Cílem mé bakalářské práce je navrhnout vhodný způsob automatického převodu prezentace z formátu OpenDocument do značkovacího jazyka Markdown, který bude následně prezentován pomocí frameworku Reveal.js.

Ve své práci se zabývám způsoby zápisu jednotlivých prvků prezentace a jejich uložení do výsledného datového formátu, dále frameworkem Reveal.js a jeho možnostmi při tvoření prezentací, přičemž mapuji možnosti práce s tímto frameworkem s důrazem na využití značkovacího jazyka Markdown.

Práce poskytuje podrobný popis jednotlivých kroků převodu a navrhuje možná řešení komplikací, které mohou v průběhu implementace nastat. Výsledná implementace je pak otestována na testovací sadě přednášek, jejíž součástí jsou i prezentace vytvořené do předmětu WAP – Internetové aplikace.

Teoretická část práce je východiskem pro část praktickou. Cílem praktické části je vytvořit aplikaci, která ulehčí převod a následné úpravy při tvorbě nových výukových materiálů.

# Kapitola 2

# OpenDocument

Tato kapitola popisuje formát OpenDocument. Popisuje uspořádání částí dokumentu a informace, které obsahují. Zvláště se zaměřuje na soubor `content.xml`, v kterém je zapsán hlavní obsah dokumentu.

## 2.1 Formát OpenDocument

Formát souboru OpenDocument je otevřený formát pro ukládání dokumentů [1]. Byl vytvořen sdružením OASIS a je založený na formátech ZIP a XML. Od roku 2006 je tento formát standardizován Mezinárodní organizací pro normalizaci. Specifikace formátu je volně přístupná a umožňuje tak formátu být použitý libovolnými aplikacemi bez omezení [6].

Přípona dokumentu se liší v závislosti na typu dokumentu, který představuje.

- textový dokument – přípona `.odt`
- tabulka – přípona `.ods`
- prezentace – přípona `.odp`
- grafika – přípona `.odg`
- matematický výraz – přípona `.odf`
- databáze – přípona `.odb`
- obrázek – přípona `.odi`
- graf – přípona `.odc`
- hlavní textový dokument – přípona `.odm`

Soubor ve formátu OpenDocument je buď jeden XML soubor, který obsahuje všechny náležitosti, nebo častěji komprimovaný soubor formátu ZIP, který obsahuje řadu souborů a adresářů s obsahem. Jsou to tyto soubory a adresáře [1]:

- soubory
  - content.xml – obsahuje obsah samotného dokumentu
  - meta.xml – obsahuje metadata dokumentu, např. autora, datum modifikace, jazyk atd.

  - settings.xml – obsahuje nastavení prostředí, např. pozici kurzoru, přiblížení atd.
  - styles.xml – informace o stylování znaků, odstavců, stránek, rámů nebo listů
  - mimetype – informace o typu dokumentu
- adresáře
  - META-INF/ – informace o souborech obsažených v archivu
  - Thumbnails/ – obsahuje náhledy dokumentu
  - adresář obsahující obrázky obsažené v prezentaci, obvykle adresář Pictures/, standard OpenDocument nespecifikuje přesné umístění obrázků

## 2.2 XML

Informace popsané v této podkapitole jsou čerpány z knihy XML pro každého: podrobný průvodce [5]. eXtensible Markup Language (zkráceně XML) je značkovací jazyk sloužící pro přehledné uchování dat a případné další využití. Při tvorbě XML dokumentu je nutné dodržovat několik pravidel. Celý dokument je tvořen elementy, které jsou do sebe zanořeny. Elementy se značí do tzv. tagů. Každý element tvoří počáteční a ukončovací tag. Tag je jméno elementu obaleno znaky `<` (zepředu) a `>` (vzadu). Ukončovací tag má navíc po znaku `<` znak `/`. Mezi počátečním a ukončovacím tagem se nachází samotný obsah elementu (text, další elementy). Ukázka zápisu jednoduchého XML elementu se nachází ve výpisu 2.1.

```
<element>Toto je obsah elementu.</element>
```

Výpis 2.1: Ukázka zápisu XML elementu

V celém dokumentu tak musí být sobě si odpovídající dvojice počátečních a ukončovacích tagů. Jedinou výjimku tvoří elementy, které nemají žádný obsah. Ty lze zapsat pomocí zkráceného zápisu a použít tak pouze jeden tag. Ukázka zápisu ukončení XML elementu bez obsahu se nachází ve výpisu 2.2.

```
<element />
```

Výpis 2.2: Ukázka ukončení XML elementu bez obsahu

V dokumentu není povoleno, aby se počáteční a ukončovací tagy více elementů křížily. Celý dokument musí mít pouze jediný kořenový element.

Každý element může obsahovat atributy, které upřesní význam elementu. Atributy se zapisují do počátečního tagu a oddělují se mezerou. Obsah atributu se musí zabalit do uvozovek nebo apostrofů. Výpis 2.3 ukazuje zápis XML elementu s atributy.

```
<element zabezpeceni="tajne" autor="FS">informace</element>
```

Výpis 2.3: Ukázka zápisu XML elementu s atributy

Protože se znaky `<` a `>` používají k oddělení tagů od ostatního textu, je nutné zamezit jejich výskytu v textu. Proto se tyto znaky kódují pomocí tzv. znakových entit. Znak `<` se tak v textu zapíše jako `<` a znak `>` jako `>`. Pro zapsání těchto znakových entit potřebujeme znak `&` a proto se v textu zapisuje entitou `&`. V případě, že je potřeba do obsahu atributu zapsat apostrof i uvozovky, je nutné je zapsat pomocí entit `&apos;` a `&quot;`.

XML dokument může obsahovat elementy a atributy z různých modulů. Např. XSLT styl, který se zapisuje jako XML, může obsahovat HTML značky. Jména elementů a atributů v těchto modulech se mohou překrývat a proto je nutné rozlišit, do kterého modulu

element nebo atribut patří, aby nedocházelo ke konfliktům. Pro rozlišení elementů a atributů se stejnými jmény slouží jmenné prostory [3]. Jména elementů a atributů se skládají ze dvou částí – z prefixu jmenného prostoru a lokálního názvu oddělených dvojtečkou. Prefix jmenného prostoru je zkratkou unikátní adresy (URI), která identifikuje každý jmenný prostor.

### Abstraktní model dokumentu

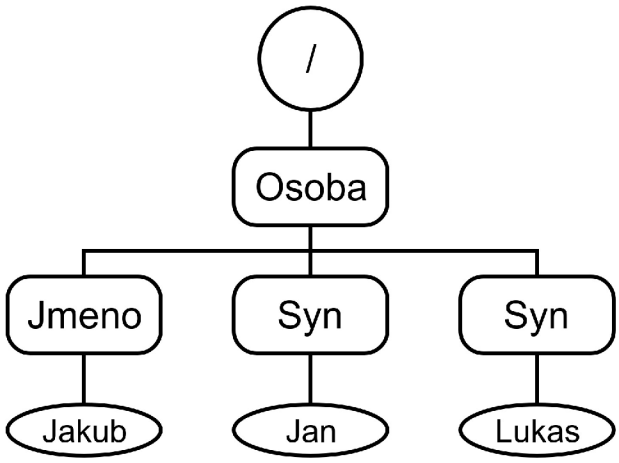
Abstraktní model dokumentu je model dokumentu vytvořený obvykle v paměti z fyzické reprezentace dokumentu. V abstraktním modelu je celý dokument XML reprezentován jako strom obsahující uzly několika typů. [4, Kapitola 2. XPath]

- kořenový uzel – jako jediný uzel neodpovídá žádnému elementu v dokumentu. Jeho potomkem je kořenový element dokumentu.
- element – každý element v dokumentu tvoří odpovídající uzel v abstraktním modelu. Jednotlivé uzly jsou do sebe zanořeny stejně, jako jsou do sebe zanořeny elementy v dokumentu. Dětmi elementu mohou být další elementy, textové elementy, komentáře. Ke každému uzlu mohou být připojeny jeho atributy a jmenné prostory.
- atribut – každý atribut je spojen s odpovídajícím uzlem. Jedná se o listový uzel, který se nepovažuje za potomka daného elementu.
- textový uzel – textový uzel odpovídá textovému obsahu elementu. Jedná se o listový uzel, který nemůže mít děti.

Na obrázku 2.1 je graficky znázorněn abstraktní model XML dokumentu ve výpisu 2.4.

```
<osoba>
  <jmeno>Jakub</jmno>
  <syn>Jan</syn>
  <syn>Lukas</syn>
</osoba>
```

Výpis 2.4: Jednoduchý XML dokument pro převod na abstraktní model




Obrázek 2.1: Ukázka grafického znázornění abstraktního modelu dokumentu

## 2.3 Struktura dokumentu content.xml pro prezentace

Obsah této podkapitoly je čerpán ze standardu OpenDocument [1].

Soubor `content.xml` obsahuje hlavní obsah prezentace. Jedná se o XML soubor, který má přesně danou strukturu. Kořen tvoří element `<office:document-content>`, který uchovává obsah v elementu `<office:body>` a automatické styly dokumentu v elementu `<office:automatic-styles>`. Dětmi tohoto elementu jsou elementy `<style:style>`, které už obsahují hodnoty nastavení jednotlivých vlastností stylů stránky, tabulky, odstavce atd. Element `<office:body>` obsahuje jednoho přímého potomka v závislosti na druhu celého dokumentu. Soubor prezentace obsahuje element `<office:presentation>`, ten už obsahuje elementy jednotlivých slidů prezentace, tedy `<draw:page>`.

Nejčastějšími dětmi elementu `<draw:page>` jsou elementy `<draw:frame>`, `<draw:custom-shape>` nebo např. `<draw:line>`. Tyto elementy obsahují informace o velikosti a umístění v dokumentu. Jejich potomky jsou elementy značící přesnou strukturu dokumentu, jako např. odstavce `<text:p>`, část textu `<text:span>`, zalomení textu `<text:line-break>`, obrázek `<draw:image>`, seznam `<text:list>`, položka seznamu `<text:list-item>` nebo tabulka `<table:table>`.

V následujícím příkladu (výpis 2.5) je uvedená zjednodušená ukázka obsahu souboru content.xml. Ukázka znázorňuje zanoření jednotlivých elementů a nastavení atributů, které jsou pro následující obsah práce důležité. Hodnoty jednotlivých atributů slouží k demonstraci.

```
<?xml version="1.0" encoding="UTF-8"?>
<office:document-content>
    <office:automatic-styles>
        <style:style style:name="name">
            <style:graphic-properties
                svg:stroke-color="#000000" draw:fill="none"
                draw:textarea-horizontal-align="center"
                fo:padding-left="0.25cm"/>
            <style:paragraph-properties
                fo:border-bottom="1.11pt solid #000000"/>
        </style:style>
    </office:automatic-styles>
    <office:body>
        <office:presentation>
            <draw:page draw:name="page1" draw:style-name="gr9">
                <draw:frame draw:name="Nadpis 1" svg:width="22.859cm"
                    svg:height="6.6cm" svg:x="1.499cm" svg:y="4.724cm">
                    <draw:text-box>
                        <text:p text:style-name="P1">
                            <text:span text:style-name="T1">Nadpis</text:span>
                        </text:p>
                    </draw:text-box>
                </draw:frame>
            </draw:page>
        </office:presentation>
    </office:body>
</office:document-content>
```

Výpis 2.5: zjednodušená ukázka obsahu souboru content.xml

# Kapitola 3

# Reaveal.js

Reveal.js je jednou z javascriptových knihoven pro tvorbu prezentací [8]. Základ prezentace tvoří HTML dokument s vlastním obsahem, který doplňují styly napsané v CSS. To umožňuje zobrazit slidy pouze za použití internetového prohlížeče, bez nutnosti instalovat speciální software. Tato knihovna nabízí několik předdefinovaných nastavení a funkcí, které usnadní její rychlé a efektivní použití. Rozsah těchto funkcí je ale výrazně menší než u běžných kancelářských programů, jako jsou například Microsoft Powerpoint nebo LibreOffice Impress. Uživatel je ale schopný rychle vytvořit zajímavou prezentaci bez složitého nastavování. Velkou výhodou Reveal.js je podpora Markdown, tedy značkovacího jazyka sloužícího k převodu prostého textu na formátovaný text většinou ve formě HTML. Mezi další zajímavé funkce lze zařadit možnost exportu prezentací do formátu PDF, možnost vytváření poznámek pro přednášejícího, které se zobrazí pouze na monitoru přednášejícího, nebo javascriptové API sloužící k rozšíření funkčnosti.

## 3.1 Základní použití

Informace pro tvorbu této podkapitoly byly čerpány z dokumentace Reveal.js [2].

Základ prezentace tvoří HTML dokument. V jeho hlavičce se importují využívané styly, v patičce se pak načítají skripty a provádí se inicializace prostředí. Obsah každého ze slajdů je obalen v HTML tagu `<section></section>`. Posloupnost zanoření jednotlivých značek musí být – značka `.reveal` obsahující značku `.slides`, která obsahuje značku `section`. Tyto slajdy jsou pak v prezentaci řazeny horizontálně za sebou. V případě, že přednášející chce vytvořit slajdy řazené vertikálně, stačí tag `<section>` umístit do jiného tagu `<section>`. Tímto způsobem může být například oddělena samostatná kapitola prezentace.

### Konfigurace

Na konci dokumentu je potřeba provést inicializaci Reveal.js, při které si autor prezentace může nastavit její chování. Všechny konfigurační hodnoty jsou volitelné a přednastavené, a proto je potřeba je uvést pouze v případě, když se požadované nastavení liší od výchozího nastavení. Uživatel si tak může zvolit vzhled, způsoby ovládání a zobrazování stavu. Vybrané hodnoty nastavení lze najít v tabulce 4.1. Všechny možnosti nastavení lze najít v dokumentaci [2].

Tabulka 3.1: Tabulka konfigurace Reveal.js

| Název | Výchozí hodnota | Popis |
|---|---|---|
| **controls** | true | Zobrazit navigační šipky |
| **controlsLayout** | 'bottom-right' | Definuje umístění navigačních šipek |
| **slideNumber** | false | Zobrazit číslo slidu |
| **hash** | false | Jedinečná url pro každý slide |
| **keyboard** | true | Umožní ovládat prezentaci pomocí klávesnice |
| **center** | true | Vertikální zarovnání slidů na střed |
| **overview** | true | Umožní zobrazit přehled slidů prezentace |
| **transition** | 'slide' | Typ přechodu mezi slidy |
| **autoSlide** | 0 | Automatický posun mezi slidy |

### Export do PDF

Reaveal.js umožňuje exportovat prezentaci do formátu PDF. Využívá k tomu vlastní CSS kód, který vloží obsah slidů na jednostlivé stránky. Použití této funkce vyžaduje použití internetového prohlížeče Google Chrome nebo Chromium a načítání prezentace z webového serveru. Při inicializaci prezentace je možné nastavit počet slidů na stránku pomocí atributu `pdfMaxPagesPerSlide`. Speciální styl `css/print/pdf.css` je nutné importovat v hlavičce dokumentu, následně už stačí v okně prohlížeče pro tisk určit požadované nastavení stránky a uložit dokument do PDF.

## 3.2 Markdown

Obsah jednotlivých slidů se v základní konfiguraci zapisuje v HTML značkách. Přidáním rozšiřujícího skriptu je možné psát obsah prezentace pomocí Markdown. Ke každému slidu (`<section>`), kde je tento zápis použit, je nutné přidat atribut `data-markdown`. Obsah v Markdown značkách se pomocí rozšíření převede na HTML, které je zobrazeno. Toto rozšíření je vytvořeno za základě data-markdown[1] a upraveno tak, aby podporovalo Github Flavored Markdown[2].

### Externí Markdown

Markdown umožňuje načítat obsah ze samostatných souborů. Pro správné načtení dat ze souboru je nutné nastavit několik informací v HTML souboru. Atribut `data-markdown` HTML značky `<section>` určuje cestu k souboru s obsahem. Tento atribut je povinný. Atribut
`data-separator` definuje regulární výraz pro značky oddělující horizontální slidy. Atribut `data-separator-vertical` definuje regulární výraz pro značky oddělující vertikální slidy. Pomocí atributu `data-charset` lze určit, kterou znakovou sadu použít k načtení externího souboru.

Načítání externího Markdown umožňuje přiřadit atributy jednotlivým elementům a slidům ve výsledném dokumentu. Atribut `data-element-attributes` značky `<section>`

[1] data-markdown – https://gist.github.com/paulirish/1343518
[2] Flavored Markdown – https://github.github.com/gfm/

v HTML souboru definuje regulární výraz zápisu atributů elementu. Následující část Markdown (výpis 3.1) ukazuje nastavení třídy (class) jednotlivých položek seznamu ve výsledném dokumentu, když je atribut `data-element-attributes` v HTML souboru nastaven na hodnotu `{_\s*?([^}]+?)}`

```
- <!-- {_class="trida1 trida2"} --> Lorem ipsum dolor sit.
- <!-- {_class="trida1 trida3"} --> In rutrum. Morbi leo mi non.
```

Výpis 3.1: ukázka nastavení atributu třídy jednotlivým položkám seznamu

Ve výsledném dokumentu se tak vytvoří seznam o dvou položkách, kde první položka bude mít atribut `class` s hodnotou `trida1 trida2` a druhá položka bude mít atribut `class` s hodnotou `trida1 trida3`. Toto nastavení je obaleno značkami pro HTML komentář, zabrání se tak špatnému zobrazení textu v případě chyby.

Podmínkou pro načítání obsahu ze samostatného souboru je, aby reveal.js byl spuštěn na webovém serveru [2].

### 3.2.1 Základní značky

Markdown je odlehčený značkovací jazyk, který slouží k jednoduchému formátování textu a jeho následnému převedení do HTML. Značky jsou umístěny přímo v textu. Další zápisy a použití jsou k nalezení v dokumentaci [7].

#### Nadpisy

Stejně jako v HTML existuje šest úrovní nadpisů, tak i v Markdown lze zapsat šest úrovní nadpisů. Značka pro nadpis je znak `#`. Pro každou další úroveň se přidává další znak `#`. Příklady zápisu jednotlivých nadpisů jsou ukázány ve výpisu 3.2.

```
# H1
## H2
### H3
#### H4
##### H5
###### H6
```

Výpis 3.2: ukázka zápisu nadpisů v Markdown

#### Seznamy

Seznam tvoří jedna a více položek. Každá z nich začíná na novém řádku, který uvozuje jeden ze znaků `-` a `+`. Jeden seznam tvoří řádky začínající stejným znakem. Číslovaný seznam lze vytvořit tak, že každou položku bude uvozovat číslo. Položky se číslují postupně od čísla první položky seznamu (na dalších číslech nezáleží). Zanožení seznamů lze vytvořit za pomoci odsazení jednotlivých seznamů. Výpis 3.3 ukazuje zápis seznamů v Markdown a jejich zanoření.

```
- polozka seznamu 1
- polozka seznamu 2
    1. zanoreni
    2. zanoreni 2
- polozka seznamu 3
```

Výpis 3.3: ukázka zápisu seznamu v Markdown a jejich zanoření

### Tabulky

V Markdown se každý řádek tabulky zapisuje na samostatný řádek zdrojového kódu. Jednotlivé buňky jsou odděleny svislítkem (znakem |). První řádek tabulky značí záhlaví tabulky a vždy po něm následuje řádek určující počet buněk v řádcích. Ten je tvořen posloupností pomlček (musí být alespoň tři za sebou) a oddělovači buněk (svislítko). Po řádku určující počet buněk mohou následovat další řádky tabulky. Výpis 3.4 ukazuje zápis tabulky v Markdown.

```
head1 | head2 | head 3
----- | ----- | ------
col 1 | col 2 | col 3
col 4 | col 5 | col 6
```

Výpis 3.4: ukázka zápisu tabulky v Markdown

### Odkazy

Odkazy lze zapsat kdekoliv v textu. Do hranatých závorek se zapíše text odkazu následovaný URL obalenou do kulatých závorek. Tento zápis ukazuje příklad ve výpisu 3.5.

```
Vice informaci naleznete [zde](http://www.fit.vutbr.cz)
```

Výpis 3.5: ukázka zápisu odkazu v Markdown

Druhý způsob pro zapsání odkazů je pomocí reference. Místo kulatých závorek s cílovou URL lze vložit referenční odkaz v hranatých závorkách. Na jiném místě zdrojového textu je nutné tento referenční odkaz definovat. Ten se definuje umístěním do hranatých závorek a následovaný cílovou URL. Tento způsob tak umožňuje odkazovat se na jednu URL z více míst zdrojového textu. Při změně URL se odkaz změní na všech místech použití. Tento zápis ukazuje příklad ve výpisu 3.6. [7]

```
[Odkaz pomoci reference][referencni odkaz]

[referencni odkaz]:http://www.fit.vutbr.cz
```

Výpis 3.6: ukázka zápisu odkazu v Markdown pomocí reference

Třetí možností zapsání odkazu v Markdown je pouhé umístění URL v textu. Ta se automaticky označí jako odkaz. Tento způsob však nefunguje vždy, protože správnost označení záleží na tvaru vložené URL.

### Obrázky

V Markdown se obrázek zapíše pomocí vykřičníku, za kterým v hranatých závorkách následuje alternativní text, pro případ, že by se obrázek nepodařilo načíst. Za tímto alternativním

textem následuje v kulatých závorkách URL obrázku. V závorce s URL lze zapsat další text, který odpovídá atributu `title` u tagu `<img>` v HTMl. Příklad tohoto zápisu ukazuje výpis 3.7.

```
![alt text](img/obrazek.jpg "Title")
```

Výpis 3.7: ukázka zápisu obrázku v Markdown

Stejně jako odkaz, jde i obrázek zapsat pomocí reference. Místo kulaté závorky se zapíše hranatá závorka s referenčním odkazem. Na jiném místě ve zdrojovém textu se pak tento referenční odkaz definuje. Tento zápis ukazuje příklad ve výpisu 3.8.

```
![alt text][logo]
[logo]:(img/obrazek.jpg "Title")
```

Výpis 3.8: ukázka zápisu obrázku v Markdown pomocí reference

### Kód a zvýraznění syntaxe

Značka kódu je v Markdown tvořena znakem zpětného apostrofu. Pro vytvoření řádkového elementu stačí text obalit jedním zpětným apostrofem. Pro delší blok kódu se text obalí třemi zpětnými apostrofy zpředu a třemi vzadu. Ve výpisu 3.9 lze nalést příklad řádkového a blokového elementu kódu.

```
Inline ‘code‘ example.

’’’javascript
var s~= "JavaScript syntax highlighting";
alert(s);
’’’
```

Výpis 3.9: ukázka zápisu kódu v Markdown

V případě blokové značky kódu (’’’) lze v některých případech využít zvýraznění syntaxe zadaného jazyka. Zvýrazňování syntaxe však není součástí Markdown, ale závisí na použitém interpretu. Proto v některých případech nemusí být dostupné. Název použitého jazyka se zapíše za úvodní apostrofy a následující kód se zvýrazní podle něj. [7]

# Kapitola 4

# XSLT

XSLT neboli eXtensible Stylesheet Language Transformations je transformace sloužící k převodu souborů ve formátu XML do jiného formátu, nejčastěji do HTML nebo jiného XML souboru. Transformaci provádí XSLT procesor, který pomocí XSL šablony transformuje zdrojový kód do výsledné podoby. [4]

Mezi nejznámější open source procesory na zpracování XML pomocí XSLT patří: [4, Podkapitola 1.3 Dostupné implementace]

- Saxon[1]
- libxslt/xsltproc[2]
- Xalan[3]

## 4.1 XPath

Informace uvedené v této části kapitoly jsou čerpány z [4, Kapitola 2. XPath].

XPath je jazyk určený pro zápis výrazu, který přesně určuje hodnotu, jeden uzel nebo množinu uzlů v XML dokumentu.

Každý výraz pro výběr uzlů se skládá z jedné nebo více částí oddělených znakem `/`. Každá část umožňuje určit osu průchodu a vybírat jen některé uzly na základě jejich typu nebo jména a dále je filtrovat pomocí podmínek testujících určité vlastnosti jednotlivých uzlů. Výraz se vyhodnocuje zleva doprava a postupně se tak upřesňuje výsledek.

V tabulce 4.1 se nachází příklady výrazů XPath společně s popisem výběru.

### Osy průchodu

V každé části výrazu je nutné určit osu průchodu po jednotlivých uzlech. Osa průchodu se zapisuje pomocí identifikátoru osy následovaného dvěma dvojtečkami. Vybrané a častěji používané osy jsou:

- child:: – všechny děti aktuálního uzlu, ve stejném pořadí jako jsou v dokumentu
- descendant:: – všechny potomci aktuálního uzlu, ve stejném pořadí jako jsou v dokumentu

[1] Saxon – http://saxon.sourceforge.net/
[2] libxslt/xsltproc – http://xmlsoft.org/libxslt/
[3] Xalan – https://xml.apache.org/xalan-j/

- descendant-or-self:: – aktuální uzel a všechny jeho potomci, ve stejném pořadí jako v dokumentu
- following-sibling:: – všechny následující uzly, které jsou sourozenci aktuálního uzlu, ve stejném pořadí jako jsou v dokumentu
- parent:: – rodič aktuálního uzlu
- self:: – aktuální uzel
- attribute:: – uzly odpovídající atributům aktuálního uzlu, pořadí není určeno

#### Test typu a jména uzlu

Za označením osy průchodu je možné testovat typ nebo jméno vybraného uzlu. `text()` vybere pouze textové uzly a `node()` vybere všechny uzly bez ohledu na jejich typ. Jméno vybraného uzlu se testuje zapsáním požadovaného jména. Pro uzel s libovolným názvem lze použít znak `*`.

#### Predikáty

Na konci každé části výrazu lze zapsat jeden nebo více predikátů. Každý predikát je zapsaný uvnitř hranatých závorek a vyhodnocuje se pro každý uzel, který odpovídá současnému výrazu. Predikát může být vyhodnocený jako pravdivý nebo nepravdivý. V případě, že je vyhodnocený jako nepravdivý, uzel je z výběru vyloučen. Více predikátů se vyhodnocuje zleva doprava. Predikát může být:

- číslo – jako pravdivý se vybere pouze uzel, který má pozici při průchodu stromem stejnou jako je dané číslo. Uzly se číslují od jedničky.
- XPath výraz – daný výraz se vyhodnotí, v případě, že je vrácená množina uzlů neprázdná, tak je predikát pravdivý
- ostatní výrazy – výrazy se vyhodnotí, výsledná hodnota je i výsledná hodnota predikátu

#### Začátek prohledávání

Výraz může prohledávat strom uzlů z několika míst:

- od aktuálního uzlu – výraz používá relativní cestu, proto výraz nezačíná znakem `/`. Aktuální uzel se mění v závislosti na šabloně a způsobu procházení dokumentu
- od kořenového uzlu – výraz používá absolutní cestu a začíná tak znakem `/`. Strom bude prohledán od kořenového uzlu bez ohledu na aktuální uzel.
- z libovolného přesně určeného uzlu – XSLT umožňuje vybrat konkrétní uzel pomocí jeho identifikátoru, a to pomocí funkce `id()`. Tato funkce může být použita pouze na začátku výrazu.

### Zkrácený zápis

Pro velmi často používané konstrukce existují zkrácené zápisy.

- v případě neuvedení osy průchodu se jako aktuální nastavení považuje průchod po ose `child::`
- v případě potřeby nastavení osy průchodu na `attribute::` lze použít znak `@`
- aktuální uzel lze odkazovat tečkou (`.`) místo `self::node()`
- rodičovský uzel lze odkazovat dvěma tečkami (`..`) místo `parent::node()`
- Při prohledávání potomků do libovolné hloubky lze použít `//` místo `/descendant-or-self::node()/`

Tabulka 4.1: Tabulka příkladů XPath

| **XPath** | **Popis** |
|---|---|
| `./katalog` | vybere všechny elementy `katalog`, které jsou dětmi aktuálního uzlu |
| `katalog` | vybere všechny elementy `katalog`, které jsou dětmi aktuálního uzlu |
| `/katalog` | vybere kořenový element `katalog` |
| `//katalog` | vybere všechny elementy `katalog` |
| `@href` | Vybere atribut `@href` aktuálního uzlu |
| `*/katalog` | Vybere všechny elementy `katalog`, které jsou vnoučaty (dětmi dětí) aktuálního uzlu |
| `/katalog/polozka[1]` | Vybere první element `polozka`, který je dítětem elementu `katalog`, kořenového elementu dokumentu |
| `/katalog/*[href='index']` | Vybere každý element, který je dítětem kořenového elementu `katalog`, pokud obsahuje jako dítě element `href` s textem `index`. |
| `..` | Vybere rodičovský element aktuálního uzlu |
| `self::polozka` | Vybere aktuální uzel, pokud je to element se jménem `polozka` |

## 4.2 XSLT styl

Informace uvedené v této části kapitoly jsou čerpány z [4, Kapitola 3. Způsob zpracování stylu].

XSLT procesor si na začátku své práce načte vstupní XML soubor a vytvoří si jeho stromovou strukturu. Tento strom pak postupně prochází v pořadí, v jakém je uveden v dokumentu (procházení do hloubky). Pro každý procházený uzel se pokusí najít šablonu, podle které uzel zpracuje. Potomci uzlu, který už byl zpracovaný podle šablony, už dále nejsou zpracováváni, pokud to není v šabloně definováno. V případě, že danému uzlu nevyhovuje žádná šablona ve stylu, je aplikována jedna ze zabudovaných šablon.

XSLT styl je soubor ve formátu XML. Kořenový element `<xsl:stylesheet>` obsahuje jednotlivé šablony `<xsl:template>`. Obecný tvar každé šablony je ukázán ve výpisu 4.1.

```
<xsl:template match="vzor">
   telo sablony
</xsl:template>
```

Výpis 4.1: obecný tvar šablony v XSLT

Atribut `match` slouží k zapsání XPath výrazu, pomocí kterého lze zjistit zda zkoumaný uzel výrazu vyhovuje. Tento výraz Xpath je však limitovaný pouze na osy průchodu pro přechod na potomky.

Tělo šablony pak přesně definuje postup zpracování daného uzlu a následný výstup. Používají se v něm další konstrukce XSLT, ale i elementy výstupu (prostý text, HTML tagy atd.). Popis vybraných elementů je popsán v tabulce 4.2. Pokud chceme aby se zpracovávaly i děti daného uzlu, je nutné v šabloně použít instrukci `<xsl:apply-templates>`. Při použití této instrukce se začnou hledat šablony pro všechny děti daného uzlu. Pro hledání šablony jen pro část dokumentu lze přidat atribut `select`, který obsahuje XPath výraz a vybere tak určitou množinu uzlů, pro které bude hledána správná šablona. Příklad tohoto zápisu lze nalézt ve výpisu 4.2.

```
<xsl:apply-templates select="polozka"/>
```

Výpis 4.2: příklad zápisu atributu select u elementu xsl:apply-templates

XSLT styl musí obsahovat alespoň jednu šablonu na zpracování kořenového elementu. Minimální zobrazení této šablony je ukázáno ve výpisu 4.3.

```
<xsl:template match="/"></xsl:template>
```

Výpis 4.3: ukázka povinné šablony v XSLT stylu

### Priorita šablon

V rozsáhlých stylech se může stát, že pro jeden uzel existuje více možných šablon. XSLT procesor, tak musí zvolit jednu z nich, kterou použije. Každá šablona tak má svou prioritu a procesor vybere šablonu s nejvyšší prioritou. Priorita lze nastavit manuálně a to přidáním atributu `priority` s číselnou hodnotou. Při nenastavení priority se procesor řídí pravidly:

- prioritu -0.5 dostanou šablony s výrazem obsahujícím znaky `*` a `@*`
- prioritu 0 dostanou šablony testující pouze název uzlu
- ostatní šablony dostanou prioritu 0.5

Tabulka 4.2: Tabulka vybraných instrukcí v XSLT

| **Instrukce** | **Popis** |
|---|---|
| `<xsl:text>` | zapsání textu na výstup |
| `<xsl:template>` | instrukce šablony<br>Atributy:<br>• match – XPath výraz uzlů, na které se šablona aplikuje<br>• priority – celé číslo, priorita vybrání šablony<br>• name – jméno šablony |
| `<xsl:value-of>` | Vypsání hodnoty proměnné nebo výrazu<br>Atributy:<br>• select – výraz XPath nebo název proměnné |
| `<xsl:variable>` | definuje lokální nebo globální proměnnou<br>Atributy:<br>• name – jméno proměnné<br>• select – výraz, hodnota proměnné |
| `<xsl:apply-templates>` | volání šablony pro aktuální uzel nebo pro jeho děti (potomky)<br>Atributy:<br>• select – výraz, množina uzlů, pro které se šablona volá |
| `<xsl:call-template>` | Volání pojmenované šablony<br>Atributy:<br>• name – jméno volané šablony |
| `<xsl:param>` | deklarace parametru<br>Atributy:<br>• name – jméno parametru<br>• select – XPath, výchozí hodnota parametru |
| `<xsl:with-param>` | volání `<xsl:apply-templates>` nebo `<xsl:call-template>` s určitým parametrem<br>Atributy:<br>• name – jméno parametru<br>• select – XPath, hodnota parametru |
| `<xsl:choose>` | podmíněná instrukce |
| `<xsl:when>` | dítě instrukce `<xsl:choose>`, obsahuje podmínku, pokud je pravdivá, tak se provede tělo<br>Atributy:<br>• test – podmínka k testování |
| `<xsl:otherwise>` | výchozí akce v `<xsl:choose>` |

# Kapitola 5

# Návrh aplikace

Cílem této bakalářské práce je navrhnutí a následná implementace aplikace, která automaticky převede danou vstupní prezentaci ve formátu .odp do Reaveal.js. Obsah prezentace by se měl převést do zápisu v Markdown, aby tak ulehčil případné další úpravy v prezentaci. Výhodou tohoto zobrazení je, že text i formátování se nachází na stejném místě a je lehce čitelné i člověku. Samotný text ve formátu Markdown je uložen v samostatném souboru, což zvyšuje přehlednost projektu a umožňuje jednodušší změnu v obsahu prezentace pouhým nahrazením celého souboru.

Grafická podobnost mezi vstupní a výslednou prezentací není tak důležitá jako obsah a jeho členění, které musí být zachováno. Aplikace tak zkopíruje text prezentace a převede pouze část z definovaných stylů, které slouží k základnímu zvýraznění textů.

## 5.1 Rozbalení archivu

Prezentace ve formátu .odp je komprimovaný archiv ve formátu .zip. Proto první krok při zpracování musí být dekomprimování tohoto archivu. Vznikne tak přesně daná struktura souborů, které mohou být jednotlivě zpracovány.

V případě ručního zpracování je možné využít systémové nástroje pro rozbalení archivu. Při automatickém zpracování lze využít nástroje pro rozbalení pomocí příkazové řádky. Mezi nejznámější se řadí `unzip` a `7zip`.

## 5.2 Převedení obsahu

Nejdůležitějším souborem v rozbaleném archivu je `content.xml`. Tento soubor ve formátu XML obsahuje samotný obsah prezentace společně s generovanými styly.

Každý slide v prezentaci je reprezentován elementem `<draw:page>`. Aplikace převede obsah každého slidu do Markdown a jednotlivé slidy oddělí značkou pro oddělení slidů v Reveal.js.

Elementy, které jsou dětmi elementu `<draw:page>`, mají nastaveny atributy `svg:width` a `svg:height`, které značí velikost (šířku a výšku) daného bloku na stránce, a atributy `svg:x` a `svg:y`, značící umístění bloku na stránce (konkrétně levého horního rohu bloku). Pořadí těchto elementů v XML není dáno, protože se výsledná pozice na stránce určuje pomocí těchto atributů. Při převodu do Markdown je proto potřeba jednotlivé elementy seřadit za sebe, aby pořadí textů bylo zachováno.

Konstrukce zapsané v Markdown jsou ve výsledném dokumentu zobrazeny postupně za sebou. Jejich vzájemná pozice nelze v samotném Markdown ovlivnit. Problém tak může nastat v případě, že ve vstupní prezentaci nejsou elementy pouze za sebou. V případě, že se nějakým způsobem překrývají nebo jsou umístěny vedle sebe, nelze v Markdown tuto pozici napodobit. Výsledný obsah se tak přeskládá pod sebe, což může význam slidu změnit, a proto by na tuto událost měl být uživatel upozorněn.

V těchto jednotlivých blocích, které jsou dětmi elementu `<draw:page>`, se nachází samotný obsah. Nejčastěji se jedná o text, obrázky atd. V případě textu se jedná o element `<text:p>`, který odpovídá novému odstavci. Ten může obsahovat libovolné množství elementů `<text:span>`, `<text:line-break>`, `<text:list>` atd. Struktura těchto elementů je velmi variabilní a závisí na aplikaci nebo osobě, která prezentaci vytváří.

Zápis elementů v Markdown je velmi jednoduchý, ale také omezený zápisem přímo v textu. Některé konstrukce tak neumožňují zapsání dalších konstrukcí uvnitř sebe sama. Tato skutečnost je podle mého názoru největším problémem převodu obsahu prezentace do Markdown. V prezentaci tak lze vytvořit několik případů, kdy na výstupu nelze zapsat stejnou konstrukci.

Příkladem tohoto problému může být element `<table:table>`, který obsahuje elementy `<table:table-cell>`. Tyto elementy znázorňují jednotlivou buňku tabulky a mohou obsahovat jakýkoli další element, např. seznam položek. Při prozkoumání zápisu tabulky a seznamu v Markdown lze nalézt velký nedostatek. Řádek tabulky se píše vždy na jeden řádek zdrojového kódu, kde jednotlivé buňky jsou odděleny znakem `|`. Na druhé straně jednotlivé položky v seznamu jsou od sebe odděleny novým řádkem. V případě výskytu seznamu v tabulce tak vzniká problém, jak zapsat element, který ze své definice musí být na více řádcích, do jednoho jediného řádku.

## 5.3 Převedení stylů

Hlavním cílem výsledné aplikace je převedení prezentace z hlediska správnosti obsahu, nikoliv z hlediska přesné grafické podobnosti. Tato vlastnost by byla velmi obtížná na zpracování, protože samotný Markdown nepodporuje nastavení stylu elementům. Prezentace v Reveal.js lze navíc spustit v různých internetových prohlížečích s různými velikostmi rozlišení. Proto všechna nastavení stylů z prezentace nelze použít.

Některá nastavení stylů však lze zachovat. Jedná se o nastavení textu. Jednotlivá nastavení zvýraznění částí textu pomocí změny barvy, velikosti nebo druhu fontu písma jsou možné vytvořit pomocí CSS.

Elementy v `content.xml` mohou mít nastaveny atributy `draw:text-style-name`, `text:style-name` a `table:style-name`. Tyto atributy obsahují identifikátor stylu, který je definován v elementu `<office:automatic-styles>`. Jednotliví potomci tohoto elementu obsahují definici stylu – jeho jméno a parametry stylu, které nastavuje. Z těchto parametrů jsem vybral několik základních, které se obvykle využívají k zvýraznění textu a které alespoň částečně dokáží napodobit styl původní prezentace. Vybral jsem tedy barvu pozadí, barvu písma, velikost písma, font písma, zarovnání textu a výšku řádku. Nastavení těchto atributů se tak převádí i do výsledné prezentace.

## 5.4 Kopírování obrázků

Pokud prezentace obsahuje nějaké obrázky, tak jsou v archivu uloženy ve adresáři `Pictures/`. V souboru `content.xml` s obsahem prezentace se nachází elementy `<draw:image>`, které značí, že daný blok obsahuje obrázek. Tento element obsahuje atribut `xlink:href`, který odkazuje do složky `Pictures/` na odpovídající obrázek. Aplikace proto překopíruje obrázky na potřebné místo a nastaví správnou cestu k nim v generovaném Markdown.

## 5.5 Úpravy podle konfigurace

Některé vlastnosti pro převod je nutné určit předem, protože je aplikace nemůže sama získat. Jedná se hlavně o nastavení značek v Markdown. Nejdůležitější nastavenou značkou je oddělovač jednotlivých slidů v Markdown. V některých případech by uživateli mohla být užitečná možnost nastavit si i ostatní Markdown znaky, jako např. jak se budou zapisovat nadpisy, seznamy.

Dalšími vlastnostmi převodu je samotná adresa vstupního souboru a jméno souboru, do kterého se má výsledný převedený obsah uložit.

Posledním druhem nastavení je možnost změnit úplně nebo alespoň částečně obsah jednotlivých slidů prezentace. Toto nastavení umožňuje odstranit určený slide, nahradit jeho obsah jiným obsahem než je v prezentaci. Také umožňuje odstranit z výsledného Markdown nastavení stylů.

Speciálním nastavením je pak převedení slidu přímo do HTML. Tato funkce částečně nahrazuje neschopnost v Markdown pozicovat elementy. Při převodu slidu do HTML se nabízí možnost jednotlivé elementy umístit na přesné místo a ve správné velikosti, a vytvořit tak obsah více graficky podobný vstupní prezentaci.

# Kapitola 6

# Implementace

Vytvořená aplikace `odptorevealjs` je napsaná v jazyce Python. Hlavní script `convert.py` postupně spouští jednotlivé podprogramy, které zpracovávají jednotlivé části převodu. Pro první zpracování XML souborů s obsahem a nastavením stylů se využívá XSLT procesor. Pro stylování obsahu prezentace se generuje kód v CSS.

Aplikace je tvořena několika soubory, které jsou strukturovány do více adresářů podle významu:

- `css/` – obsahuje soubory CSS, nastavení stylů prezentace, tiskový styl, grafická témata Reveal.js
- `js/` – obsahuje javascript pro správné fungování Reveal.js
- `lib/` – obsahuje další soubory potřebné pro správné fungování Reveal.js jako je nastavení stylů, fonty, javascript
- `markdown/` – obsahuje javascriptové soubory, které zajišťují převod markdown do HTML
- `plugin/` – obsahuje soubory pro základní rozšíření Reveal.js jako např. export do pdf, zvýraznění syntaxe (highlight) atd.
- `py/` – jednotlivé scripty v jazyce python, jsou volány postupně z convert.py, každý upraví část obsahu
- `xslt/` – obsahuje xslt styly pro převod jednotlivého obsahu

## 6.1 Základní práce z odptorevealjs

Tato sekce obsahuje základní popis, jak pracovat s aplikací `odptorevealjs`.

### Konfigurační soubor

Konfigurační soubor je soubor ve formátu json, který slouží k nastavení převodu prezentace. Musí obsahovat cestu ke vstupní prezentaci, která je zapsána v objektu s názvem `input` a cestu k výstupnímu souboru, který je zapsán v objektu s názvem `output`. Příklad konfiguračního souboru lze nalézt ve výpisu 6.1.

```
{
  "presentation":{
    "input": "cesta/prezentation.odp",
    "output": "cesta/presentation.md",
    "separator": "++++"
  }
}
```

Výpis 6.1: Ukázka nastavení konfiguračního souboru

### Nastavení prezentace

#### Objekt input

Objekt `input` je povinným parametrem v konfiguračním souboru. Jeho hodnota musí být validní cesta k souboru `.odp`. V případě, že cesta není validní nebo objekt neexistuje, převod skončí s chybou.

#### Objekt output

Objekt `output` je povinným parametrem v konfiguračním souboru. Jeho hodnota značí cestu k souboru, ve kterém bude uložený výsledný Markdown.

#### Objekt separator

Objekt `separator` je volitelný parametr v konfiguračním souboru. Jeho hodnota je jednořádkový řetězec, kterým budou ve výsledném Markdown odděleny horizontální slidy. Jeho výchozí hodnota je ++++.

### Nastavení konkrétního slidu

Konfigurační soubor může obsahovat nastavení převodu jednotlivých slidů ze vstupní prezentace. Tímto nastavením lze daný slide z výsledného Markdown odstranit nebo změnit jeho obsah na definovanou hodnotu. Tato část konfiguračního souboru umožňuje nastavit odstranění stylování jednotlivých prvků na daném slidu nebo změnit generovnání Markdown na generování HTML. Výchozí nastavení pro každý slide je ukázáno ve výpisu 6.2, kde je toto nastavení aplikováno pro první slide v prezentaci.

Nastavení `delete` a `content` je užitečné v případě, že se převádí více prezentací pomocí jednoho konfiguračního souboru a každá prezentace má stejné rozložení slidů. Např. na začátku každé prezentace je úvodní slide. Pomocí nastavení `content` lze nastavit jeho obsah přímo při převodu prezentace. Uživateli se zlehčí následná úprava výsledného Markdown. V případě, že např. chce uživatel spojit více prezentací do sebe, úvodní slidy přednášky nepotřebuje. Pomocí nastavení `delete` lze tyto slidy odstranit.

```
{
    "0":{
        "delete": false,
        "content": false,
        "absolute": false,
        "style": true
    }
}
```

Výpis 6.2: Ukázka výchozího nastavení konfigurace prvního slidu prezentace

**Objekt značící slide**

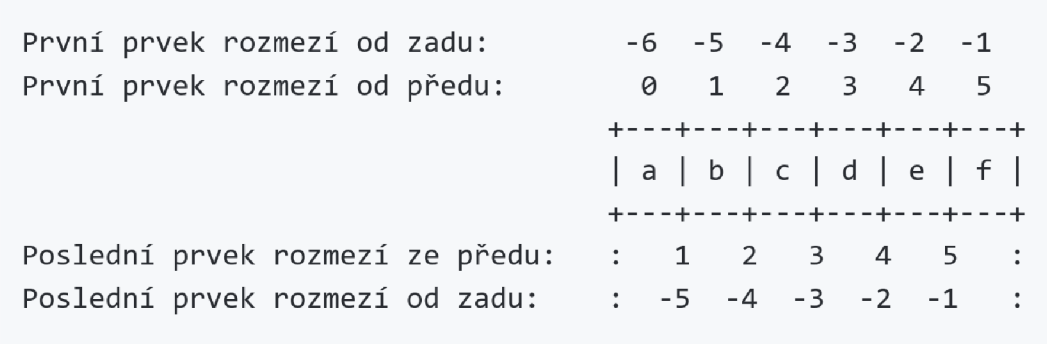
Objekt značící slide se určuje pomocí pořadí, ve kterém se v prezentaci nachází. Jednotlivé slidy se číslují od nuly, proto objekt se jménem `0` určuje vlastnosti prvního slidu v prezentaci. Indentifikátory slidů mohou obsahovat pouze celá čísla. V případě použití indetifikátoru, který se v prezentaci nenachází, se jeho nastavení ignoruje. Nastavení se ignoruje i v případě chybně zapsaného identifikátoru.

V případě potřeby lze využít záporný index pro indexování od posledního slidu prezentace. Pro nastavení více slidů současně lze jednotlivé indexy slidů oddělit čárkou. Jednotlivá nastavení se tak aplikují na všechny tyto slidy. V konfiguračním souboru lze nastavit parametry více slidů pomocí rozmezí. To se definuje čísly prvního a posledního slidu oddělených dvojtečkou (vynechání čísla značí začátek nebo konec slidů prezentace). Způsoby nastavení více slidů a rozmezí slidů nejde kombinovat.

Ukázky způsobů indexování se nachází ve výpisu 6.3 a na obrázku 6.1.

```
{
    "0":{
        "content": false
    },
    "-2":{
        "content": false
    },
    "0, -1, 2":{
        "content": false
    },
    ":2":{
        "content": false
    },
    "1:-1":{
        "content": false
    },
    ":":{
        "content": false
    }
}
```

Výpis 6.3: Ukázka způsobů nastavování konfigurace jednotlivým slidům




Obrázek 6.1: Ukázka indexování jednotlivých slidů v konfiguraci slidů

**Objekt delete**
Objekt `delete` je volitelným parametrem nastavení slidu. Jeho výchozí hodnota je `false`. Nastavením tohoto objektu na `true` se daný slide odstraní z výsledného Markdown. V případě nastavení na `true` se další nastavení slidu neprojeví. Ukázka nastavení tohoto objektu lze najít ve výpisu 6.4.

```
{
    "0":{
            "delete": true
    }
}
```

Výpis 6.4: Ukázka smazání slidu v konfiguraci

**Objekt content**
Objekt `content` je volitelným parametrem nastavení slidu, který slouží k přepsání obsahu daného slidu. Jeho výchozí hodnota je `false`. Pro přepsání obsahu slidu je nutné, aby nově nastavená hodnota byla řetězec. Ukázku nastavení lze najít ve výpisu 6.5.

```
{
    "0":{
            "content": ""
    },
    "1":{
            "content": "Nov obsah"
    }
}
```

Výpis 6.5: Ukázka změny obsahu slidu v konfiguraci

**Objekt absolute**
Objekt `absolute` je volitelným parametrem nastavení slidu, který slouží k převodu daného slidu do HTML. Jeho výchozí hodnotou je `false`. Pro aktivaci převodu do HTML je nutné nastavit hodnotu na `true`. Ukázka nastavení lze najít ve výpisu 6.6.

```
{
    "0":{
            "absolute": true
    }
}
```

Výpis 6.6: Ukázka nastavení převodu do HTML slidu v konfiguraci

**Objekt style**
Objekt `style` je volitelným parametrem nastavení slidu, který určuje, zda daný slide obsahuje parametry `class` pro stylování pomocí CSS. Výchozí hodnota je `true`. Pro smazání stylů na daném slidu je nutné nastavit hodnotu na `false`. Ukázku nastavení lze najít ve výpisu 6.7.

```
{
    "0":{
            "style": false
    }
}
```

Výpis 6.7: Ukázka odstranění stylů pro slide v konfiguraci

### Nastavení značek Markdown

Konfigurační soubor může obsahovat nastavení jednotlivých prvků (nadpisy, seznam) v Markdown. Nastavení se provádí pomocí objektu `characters`, který obsahuje nastavení jednotlivých značek. Toto nastavení umožňuje určit styl (úvodní znak) položek seznamu a určit úroveň jednotlivých nadpisů, popř. vypsat jednotlivé prvky bez úvodního znaku (odstavec v Markdown). Ukázku výchozího nastavení lze najít ve výpisu 6.8.

```
{
    "characters":{
        "list": "-",
        "heading1": "#",
        "heading2": "##"
    }
}
```

Výpis 6.8: Ukázka výchozího nastavení markdown značek v konfiguraci

**Objekt list**
Objekt `list` je volitelný parametr nastavení prezentace, který určuje znak nebo sekvenci znaků, kterými bude začínat každý prvek seznamu ve vygenerované prezentaci. Výchozí hodnota tohoto objektu je znak pomlčky (`-`).

**Objekt heading1**
Objekt `heading1` je volitelný parametr nastavení prezentace, který určuje znak nebo sekvenci znaků, kterými bude začínat každý nadpis první úrovně ve vygenerované prezentaci. Výchozí hodnota tohoto objektu je znak .

**Objekt heading2**
Objekt `heading2` je volitelný parametr nastavení prezentace, který určuje znak nebo sek-

venci znaků, kterými bude začínat každý nadpis druhé úrovně ve vygenerované prezentaci. Výchozí hodnota tohoto objektu je znak .

### Spuštění převodu

Převod prezentace zajišťuje script `convert.py`, který je napsaný v jazyce Python verze 3. Jediným a zároveň volitelným parametrem scriptu je validní cesta ke konfiguračnímu souboru. Výchozím nastavením pro konfigurační soubor je hodnota `convert.conf`, tedy případ, kdy je konfigurační soubor `convert.conf` uložen ve stejném adresáři jako script `conver.py`. Možné způsoby spuštění převodu lze nalézt ve výpisu 6.9.

```
python3 convert.py

python3 convert.py cesta/convert.conf

python3 convert.py --help
```

Výpis 6.9: Ukázka spuštění převodu

### Vygenerované soubory

Výsledkem převodu je několik souborů:

1. Markdown s obsahem prezentace. Umístění podle konfiguračního souboru.
2. CSS soubor umístěný v "css/generated.css". Obsahuje nastavení stylů z prezentace
3. Obrázky využité v prezentaci umístěné v "Pictures/"

## 6.2 Convert.py

Soubor `convert.py` je jádrem aplikace. Script nejdříve zkontroluje nastavení v konfiguračním souboru a následně začne spouštět jednotlivé podprogramy, které zpracovávají příslušnou část převodu. O každé akci je uživatel informován pomocí výpisu na standardní výstup. V případě nalezení chyby je program ukončen s výpisem příslušného problému.

## 6.3 Dočasné soubory

Prvním krokem, který aplikace provede, je vytvoření dočasné složky `tmp/`, do které je následně rozbalen archiv vstupní prezentace. Pokud v aktuálním adresáři tento adresář již existuje, je smazán včetně jeho obsahu.

Samotné rozbalení je provedeno voláním programu `unzip`, které pomocí parametru `-d` umožňuje zvolit, do kterého adresáře se daný archiv (prezentace) rozbalí. Protože obvyklým chováním programu `unzip` je i vypisování informací o prováděném rozbalování, je nutné ho vypnout pomocí parametru `-q`. Výpis informací o převodu prezentace nebude rušen podrobným výpisem o stavu rozbalení.

Na konci každého převodu je dočasný adresář `tmp/` odstraněn i s celým obsahem.

## 6.4 configuration.py

Soubor `configuration.py` je spuštěn na začátku převodu po rozbalení prezentace. Script provádí zpracování konfiguračního souboru. Načte hodnotu oddělovače slidů (v konfiguračním souboru hodnota parametru `separator`.) a vytvoří jednoduchý regulární výraz, který se následně využije pro úpravy vygenerovaných souborů. Tento nový údaj uloží do nového konfiguračního souboru v dočasném adresáři `tmp/`.

## 6.5 Generování Markdown

Dalším krokem je převod obsahu `content.xml` do Markdown pomocí XSLT stylu v souboru `xstl/presentation-new.xslt`. Do souboru `presentation-unsorted.md` v dočasném adresáři `tml/` se uloží výsledek převodu.

Pro převod jsem zvolil využití knihovny `libxslt`. Jedná se o XSLT knihovnu napsanou v jazyce C, která je uvolněná pod licencí MIT. Součástí této knihovny je nástroj příkazové řádky `xsltproc`. Ten je volán scriptem `convert.py` s parametrem `-o` pro nastavení výstupu a parametrem `-stringparam`, který umožní vložení parametru do XSLT stylu. Do stylu je vložen parametr `oddelovac`, který má hodnotu oddělovače jednotlivých slidů v Markdown, tedy hodnotu parametru `separator` v konfiguračním souboru.

XSLT styl je tvořen šablonami pro jednotlivé elementy `content.xml`. Šablony negenerují čistý Markdown, ale Markdown doplněný o různé značky, které jsou zpracovávány dalšími scripty. Tyto značky umožňují různé kontroly vygenerovaného kódu. Po zpracování příslušnou částí programu jsou tyto značky odstraněny. Značky, které jsou společně s Markdown generovány jsou:

- `rozmery <!-- {_class="hide rozmery" } -->` – obsahuje rozměry a umístění elementu na slidy, využívá se pro určení vzájemné pozice mezi elementy na stránce
- `!--text--!` – obsahuje celý text elementu, využívá se ke kontrole obsahu elementu a jejich dělení do menších částí
- `&lt;!-- {_class="hide pozor" data-id="text:p" } -->` – vypíše se v šabloně, když mohl nastat problém s výpisem textu

XSLT styl obsahuje výchozí šablony, které jsou využity v případě, že pro aktuální element neexistuje jiná šablona. Tyto šablony vypíší obsah daného elementu. Použití těchto šablon je nežádoucí, protože by vypisovaly údaje, které do výstupu nepatří. Z tohoto důvodu mají všechny šablony ve stylu `presentation-new.xstl` nastavený atribut `priority` na hodnotu 3. Obecná šablona `<xsl:template match="*" priority="2"></xsl:template>` je také součástí stylu. Ta s aktuálním elementem nijak nepracuje a ukončí prohledávání potomků. Šablona odpovídá všem elementů, ale protože má atribut `priority` nastaven na hodnotu 2, je tato šablona využita až v případě, že neexistuje jiná šablona s prioritou 3. Tato obecná šablona se ale využije dříve než výchozí šablony stylu, a tím zabrání vypsání nechtěných informací.

Většina elementů v Markdown začíná na novém řádku. V případě seznamu se ale může stát, že jednotlivé položky jsou zanořeny do více úrovní. V tomto případě se odsazení zapisuje pomocí několika mezer na začátku řádku. Všechny položky se stejným odsazením jsou na stejné úrovni zanoření. Proto se do každé šablony předává parametr `odsazeni`, který značí úroveň zanoření. Na začátku je nastaven na nulu, ale v případě, že využije šablona

pro element `<text:list-item>`, tak všechny jeho potomci jsou zpracováváni s parametrem odsazení o jedna větší. Před samotným výpisem hodnoty v šabloně se volá speciální šablona `<xsl:template name="for">`. Ta umístí na výstup tolik znaků (v tomto případě mezer), které jsou volány v parametru `stop`. Šablona také umožňuje nastavit znak nebo sekvenci znaků, které jsou vloženy při jednom cyklu. Nastavení znaku se provádí voláním této šablony s parametrem `znak`.

Mezi jednotlivými šablonami pro zpracování obsahu se předává několik parametrů. Jedním z nich je parametr se jménem `styl`. Jedná se o hodnoty atributů `text:style-name`, `table:style-name` a `draw:text-style-name` všech jeho nadřazených elementů, které jsou odděleny mezerou. Tento atribut se předává až k listovým elementům v XML, kde je dán na výstup jako atribut třídy. Protože se jednotlivé hodnoty atributů přidávají za sebe, tak první z nich má nejvíce obecné vlastnosti a poslední z nich určuje vlastnosti nejvíce konkrétní pro daný prvek.

Dalším parametrem, který se mezi jednotlivými šablonami předává je `nadrazeny`. Tento parametr slouží pro šablonu k informaci, jestli je její rodič element nadpisu, podnadpisu, seznam nebo jiný element. Podle toho, který element je rodič, se chová zalomení řádku. V případě nadpisu nebo podnadpisu se při novém odstavci nebo zalomení řádku generuje druhý element v Markdown uvozený znakem nadpisu nebo podnadpisu. V případě zalomení řádku v seznamu se vygeneruje do Markdown element `<span class="newline"> </span>`. Pomocí CSS, které je nastavené pro třídu `newline`, se text zalomí na další řádek, i když v Markdown je vše na jednom řádku.

## 6.6 Generování HTML

Konfigurační soubor umožňuje nastavit pro jednotlivé slidy položku `absolute`. V případě nastavení na hodnotu `true` se pro daný slide nevygeneruje Markdown, ale vygeneruje se HTML kód. Výhodou tohoto kódu je, že má stejnou strukturu jako původní XML a umožňuje tak zapsat konstrukce, které by jinak zapsat nešly. Jednotlivé elementy jsou pozicovány absolutně, tzn. že na výsledném slidu jsou na stejném místě jako v převáděné prezentaci.

Zpracování konfiguračního souboru a rozhodování o nastavení jednotlivých slidů probíhá na konci převodu. Převod všech slidů do HTML je tak spuštěn dříve a posouzení, který druh vygenerovaného kódu bude použit, se rozhodne později.

Pro převod se použije XSLT styl v souboru `tmp/presentation-new-absolute.xslt`. Spuštění převodu se provádí voláním příkazu `xsltproc` jako u generování Markdown. Výsledek převodu se ale ukládá do souboru `tmp/presentation-unsorted-absolute.md`.

## 6.7 Kontrola textu

Vygenerovaný Markdown je následně kontrolován, zda obsahuje celý text. K tomu se využijí značky, které jsou v souboru `tmp/presentation-unsorted.md` k tomu určené. Jedná se o značky `<!-- {_class="hide pozor" data-id="text:p" } -->`, kde text atributu `data-id` obsahuje jméno elementu, v kterém by se mohla nacházet chyba.

V případě nalezení značky je vytvořen soubor `error.json` v dočasné složce. Do tohoto souboru je vložena poznámka o čísle slidu a upozornění, aby uživatel zkontroloval obsah na daném slidu. Tento soubor se pak využívá pro ukládání jiných chyb a upozornění v průběhu převodu.

Z kontrolovaného textu jsou speciální značky odstraněny a výsledný Markdown uložen do souboru `tmp/presentation-loaderror.md`.

## 6.8 Zpracování tabulek

V souboru `content.xml` jsou tabulky reprezentovány elementem `<table:table>`. Ten nejdříve obsahuje elementy `<table:table-column>` značící sloupec tabulky a jeho vlastnosti a následují elementy `<table:table-row>`, které již obsahují jednotlivé buňky s obsahem. XSLT šablona pro tabulku tak nejdříve zpracuje elementy sloupce pro zjištění jejich počtu a až následně zpracuje obsah jednotlivých buněk tabulky. Tabulka tak na výstupu není správně zapsaná, protože první řádek tabulky obsahuje zápis např. `---|---`, který značí počet sloupců v tabulce, a v Markdown má být zapsaný až na druhém řádku.

Script `table.py` načte obsah souboru `tmp/presentation-loaderror.md` a pomocí regulárního výrazu vyhledá všechny tabulky. V nich následně prohodí první a druhý řádek, aby daná tabulka byla správně zapsána.

## 6.9 Generování stylu

Pomocí XSLT stylu `style.xslt` uloženého ve složce `xslt/` vygeneruje do adresáře `css/` nový soubor `generated.css`. Ten obsahuje CSS kód pro nastavení vzhledu elementů prezentace.

V případě, že element v souboru `content.xml` má speciální vlastnosti stylu, je mu přiřazen atribut `draw:text-style-name`, `text:style-name` nebo `table:style-name`. Jejich hodnota odkazuje na elementy `<style:style>`, které obsahují nastavení jednotlivých vlastností. XSLT styl prohledá všechny tyto elementy v souboru `content.xml` a pro každý nalezený element zkontroluje všechny jeho atributy. V případě, že se jméno atributu shoduje s jedním z definovaných v šabloně `<xsl:template match="*" mode="atribut">`, je tento atribut převeden do CSS vlastnosti tomu odpovídající. Tyto vlastnosti jsou nastaveny elementům, jejichž třída je shodná s hodnotou atributu elementu.

Každému elementu, který má nastavený atribut `text:style-name`, `table:style-name` a `draw:text-style-name`, je potřeba v Markdown přiřadit třídu dané hodnoty, aby se vlastnosti tohoto stylu aplikovaly. Umístěním `<!-- {_class="trida"} -->` za daný prvek, kde `trida` je seznam všech tříd prvku oddělených mezerou, probíhá nastavení třídy prvku v Markdown. Při převodu do HTML je příslušnému elementu přiřazena třída pomocí atributu `class`.

## 6.10 Řazení stylu

Jednotlivé elementy v `content.xml` mohou mít nastaveny atributy `draw:text-style-name`, `text:style-name` a `table:style-name`. Hodnoty těchto atributů tak tvoří strom jako samotné elementy, kde stejná vlastnost potomka má vyšší váhu než hodnota vlastnosti rodiče. Jednotlivé prvky v Markdown mají jednotlivé třídy seřazeny podle daného stromu, avšak vygenerované CSS v souboru css/generated.css není seřazeno. Jeho pravidla se aplikují podle seřazení tříd v tomto souboru a ne podle umístění dané třídy ve stromu. Proto je nutné obsah souboru `css/generated.css` seřadit tak, aby nastavení tříd šlo ve stejném pořadí jako v daném stromu.

Řazení nastavení tříd v tomto souboru provádí script `style.py`. Ten si v první fázi načte obsah souboru `generated.css` do vlastní struktury. Následně prochází soubor s vygenerovaným Markdown a hledá pomocí regulárního výrazu nastavení třídy nějakému elementu. V případě, že nalezne nastavení třídy, vezme jeho hodnotu, která je zpravidla tvořena více jmény tříd, a tyto třídy vloží do svého seznamu tak, aby se pořadí tříd neměnilo. Např. v Markdown jsou dva elementy. První má třídy nastavené na `h1` a `h2` a druhý na `h3` a `h1`. Druhé hodnoty se v XML nacházely dále od kořene, a proto jsou důležitější. Ve správném CSS stylu musí být nejdříve definována třída `h3`, pak `h1` a nakonec `h2`. Script zpracuje první element tak, že jeho třídy nechá seřazené tak jak jsou. Podle druhého pravidla je však třída `h3` obecnější než `h1`, a proto se vloží před třídu `h1`, která se již v seřazeném seznamu nachází. Vygenerováním tříd v nově seřazeném pořadí je zajištěno uplatnění CSS vlastností, které jsou konkrétnější.

## 6.11 Kopírování obrázků

V případě, že rozbalená prezentace obsahuje obrázky, nachází se v adresáři `Pictures/`. Všechny tyto obrázky jsou v prezentaci využity, a proto stačí zkopírovat celý adresář mimo dočasný adresář, v kterém je umístěn.

Ve vygenerov,aném Markdown je u obrázků nutné správně zapsat cestu. Pokud má nový adresář stejné jméno jako původní tzn. `Pictures/`, nemusí se adresa obrázků upravovat a stačí ji tak pouze zkopírovat do výsledného Markdown.

Ke kopírování této složky jsem využil operaci `copytree` z modulu `shutil`. Ta kopíruje zadaný adresář rekurzivně.

## 6.12 Kontrola překrývání

Každý blok v prezentaci má přesně danou pozici a velikost na slidu. Zatímco v Markdown se řadí bloky (jednotlivé elementy) pod sebe, v prezentaci se mohou libovolně překrývat. Proto je nutné na tyto případy uživatele upozornit.

O kontrolu překrývání jednotlivých bloků se stará script `position.py`. Ten si načte soubor s Markdown a rozdělí ho po jednotlivých slidech. Na každém slidu hledá pomocí regulárního výrazu speciální značku `rozmery <!-- {_class="hide rozmery" } -->`. Ta obsahuje atributy `width`, `height`, `x`, `y` a vztahuje se na následující Markdown až do další značky nebo konce slidu. Jednotlivé bloky jsou následně seřazeny podle atributu `y` a za pomoci atributu `height` je dopočítáno, zda se bloky překrývají. V kladném případě se pomocí atributů `x` a `width` spočítá, zda se prvky opravdu překrývají nebo jsou umístěny pouze vedle sebe.

Při překrývání se prvků nebo když jsou prvky vedle sebe se do souboru `tmp/errors.json` uloží pro daný slide záznam, který uživatele upozorní, aby se danému slidu věnoval a zkontroloval tak jeho obsah.

## 6.13 Mezery mezi  elementy

Již upravený Markdown v souboru `tmp/presentation-spaces.md` je následně kontrolován, zda obsahuje správný text s důrazem na mezery mezi jednotlivými částmi textu, které jsou obaleny inline značkou. Pro kontrolu se využívají speciální značky `!--text--!` vytvořené při generování Markdown. Obsah této značky (`text`) je tvořen textem celého bloku ze

zdrojového XML. Tento text se porovnává s obsahem jednotlivých inline značek. V případě špatného rozložení mezer se výsledný text upraví pomocí správného textu ve značce.

Z kontrolovaného textu jsou speciální značky odstraněny a výsledný Markdown uložen do souboru `tmp/presentation-span.md`.

V první části vývoje aplikace se jednotlivé inline elementy zapisovaly v Markdown jako kód pomocí zpětných apostrofů. U rozsáhlejších prezentací se ale stávalo, že se vygenerovaný Markdown stal z důvodu nadměrného množství apostrofů nečitelný. Proto byl do aplikace doplněn script `span.py`, který pomocí regulárního výrazu načte všechny inline elementy zapsané pomocí zpětných apostrofů a místo nich obalí obsah HTML značkou `<span>`.

## 6.14 Prázdné řádky a znaky Markdown

Protože použitím XSLT šablon a následujících úprav Markdown vznikl text s velkým množstvím prázdných řádků, byl do aplikace přidán script `row.py`, který pomocí regulárního výrazu hledá prázdné řádky. Nalezené řádky smaže a výsledný Markdown uloží do souboru `tmp/presentation.md`. Výsledný text není roztahaný a dá se lépe číst.

Při vývoji aplikace vznikl požadavek na možnost změnit značky generovaného Markdown. Pomocí tohoto nastavení lze měnit znak, který uvozuje zápis seznamu (`-`), nadpisu () a podnadpisu (). Nastavení se nachází v konfiguračním souboru v prvku `characters`, který obsahuje položku `list` pro určení znaku nebo sekvence znaků uvozující položku seznamu, položku `heading1` určující znak nebo sekvenci znaků pro zápis nadpisu a položku `heading2` určující znak nebo sekvenci znaků pro zápis podnadpisu.

Pomocí tohoto nastavení lze např. změnit úroveň všech nadpisů v prezentaci přímo v nastavení převodu, což může uživateli ulehčit následnou práci s vygenerovaným Markdown.

## 6.15 Zpracování konfiguračního souboru

V konfiguračním souboru je možné využít více způsobů zápisu nastavení jednotlivých slidů. Slide jde vybrat pomocí kladného (počítáno od začátku prezentace) nebo záporného (počítáno od posledního slidu prezentace) indexu. Skupina slidů lze zapsat jako seznam čísel slidů oddělený čárkou nebo rozmezí dvou indexů.

Tyto hromadné zápisy je nutné zpracovat, aby bylo přímo jasné, kterých slidů se nastavení týká. Proto se pomocí scriptu `configuration-later.py` zpracuje konfigurační soubor a vytvoří se tak upravený konfigurační soubor `tmp/convert.conf`, který obsahuje nastavení konkrétního slidu pouze za pomoci kladného indexu (zruší vícenásobnou volbu nastavení). Tato konfigurace se následně využívá při složení výsledného souboru.

## 6.16 Výsledný soubor

Na konci převodu prezentace se spouští script `composition.py`. Ten pomocí nastavení jednotlivých slidů v souboru `tmp/convert.conf` vytvoří soubor `presentation.md` v aktuálním pracovním adresáři. Script si načte obsah souboru `tmp/presentation.md` obsahující upravený obsah prezentace v Markdown a soubor `tmp/presentation-unsorted-absolute.md`, který obsahuje vygenerované HTML pro celou prezentaci.

Script prochází každý slide prezentace a pomocí obsahu konfiguračního souboru určí, který obsah dá na výstup. V případě nastavení `delete` na `true` se slide vynechá. V případě nastavení `content` se obsah daného slidu nahradí zadaným obsahem. V případě nastavení

`absolute` se do výsledného souboru uloží buď obsah slidu v Markdown nebo obsah slidu v HTML. Poslední možností nastavení je nastavení `style`, které smaže všechna označení tříd ve zdrojovém kódu. Prezentace se tak stává nestylovanou.

## 6.17 Výpis chyb a varování

Na závěr celého převodu je vypsán na standardní výstup přehled o celém převodu pomocí scriptu `errors.py`. Pro každý slide prezentace jsou ze souboru `tmp/errors.json` vypsána všechna varování. V případě, že pro slide nevznikla žádná upozornění, je daný slide vypsaný jako zpracovaný bez problémů.

# Kapitola 7

# Testování

Při testování aplikace jsem se zaměřil na 2 kritéria. Jedním je, zda výsledná prezentace obsahuje text, který je v původní převáděné prezentaci. Toto kritérium je velmi důležité, protože uživatel předpokládá převod celého textu a v případě, že by se nějaký text nepřevedl, musel by převod kontrolovat více podrobně. Druhým kritériem je grafická podobnost vzorové prezentace s převedeným výsledkem. Aplikace se nezaměřuje na přesnou podobnost převedené prezentace s původní, ale základní druhy zvýraznění textu převést dokáže. V některých případech je tato vlastnost převodu zbytečná, protože uživateli jde o převedení obsahu.

Testování jsem provedl na prezentacích z různých zdrojů:

- vlastní prezentace – reprezentující základní použití prezentace, testování jednoduchých prvků
- prezentace do předmětu WAP – složitější prezentace, testování složitých konstrukcí
- prezentace dostupné na webu – vybrané prezentace z internetu, představují prezentace, na kterých nebyla aplikace testována v průběhu jejího vývoje

Výsledný přehled testů se nachází v tabulce 7.1.

## 7.1 Vlastní prezentace

Tyto prezentace byly vytvořeny pro testování aplikace v průběhu implementace a obsahují pouze základní prvky.

### 1-zaklad/zaklad.odp

Prezentace `zaklad.odp` v adresáři `1-zaklad/` obsahuje pouze čtyři slidy. Jejich obsahem jsou základní prvky v prezentacích – nadpisy, text odstavce, seznam, obrázek a tabulka. Nejedná se o složité konstrukce, a proto převod proběhl bez komplikací. Grafická podobnost je bez výraznějších rozdílů, proto bych tento převod ohodnotil 100%.

### 2-obhajoba/obhajoba.odp

Prezentace `obhajoba.odp` v adresáři `2-obhajoba/` obsahuje dvanáct slidů a byla použita při obhajobě semestrálního projektu, na který tato práce navazuje. Prezentace obsahuje

základní prvky se základním stylováním nadpisů. Při převodu jsou vypsána varování pro pět slidů s informací, že se některé prvky na slidu překrývají. Po prozkoumání je vidět, že se jedná o prvky, u kterých se nepřekrývá obsah, ale pouze jejich okraje. Text prezentace se převedl celý, grafická podobnost je až na drobné rozdíly stejná.

## 7.2 WAP prezentace

Tyto prezentace byly vytvořeny pro předmět WAP – Internetové aplikace. Cílem praktické části této práce je vytvořit aplikaci, která pomůže s převodem a následnou úpravou právě těchto prezentací.

### 1-konektivita/WAP01-Konektivita.odp

Prezentace `WAP01-Konektivita.odp` v adresáři `1-konektivita/` obsahuje šedesát tři slidů a obsahuje látku do předmětu WAP. Při převodu všech slidů do Markdown vypíše aplikace mnoho varování o překrývání se prvků, ale také několik o možnosti chybějícího textu, který musí uživatel zkontrolovat sám. Chybějící text je způsoben uložením samotného textu a dalších prvků do jednoho nadřazeného prvku. Převod by proběhl bez problémů, když by text byl uložen v samostatném prvku. Protože se v prezentaci objevuje mnoho obrázků, které se překrývají, aby dosáhly správného zobrazení, je nutné u některých slidů zapnout generování do HTML. Tuto situaci lze vidět již na slidu č.4, kde je obrázek znázorňující více vrstev komunikace překrytý bílým obdélníkem, aby byla zobrazena pouze první vrstva. Celkem na jedenácti slidech program nedokázal vypsat správně text, na všechny nedostatky ale upozornil, a tak uživatel může nedostatek napravit, proto je hodnocení převedeného textu 85%. Grafická podobnost se liší nejvíce u slidů, které obsahují výpis kódu, který byl na některých místech špatně zarovnán, proto je hodnocení 75%.

### 2-WWW/WAP02-WWW.odp

Prezentace `WAP02-WWW.odp` v adresáři `2-WWW/` obsahuje osmdesát čtyři slidů a obsahuje látku do předmětu WAP. Slidy jsou tvořeny převážně textem, a proto se při výpisu převodu častěji objevuje varování pro kontrolu obsahu slidu. Celkem na dvacet tři slidů program upozornil uživatele, aby je zkontroloval, protože nezvládl vygenerovat obsah. Dlouhé texty jsou hůře formátované a velikost písma se často mění. Hodnocení převodu textu je tak 75% a hodnocení grafické podobnosti je 60%

### 3-XSLT/WAP04b-XPath-XSLT.odp

Prezentace `WAP04b-XPath-XSLT.odp` v adresáři `3-XSLT/` obsahuje šedesát sedm slidů a obsahuje látku do předmětu WAP. V této prezentaci se nachází velké množství seznamů a ukázek kódu. Celkem na devět slidů program upozornil, aby je uživatel zkontroloval, jedná se převážně o slidy s ukázkami kódu, kde je formátování náročnější. Protože rozdíly v textech jsou malé, tak hodnocení převodu textu je 92%. U grafické podobnosti jsou nejproblematičtější výpisy kódu se špatným zarovnáním a několik grafů, kterých není v prezentaci mnoho, proto hodnocení grafické podobnosti je 85%.

### 4-HTML/WAP05a-HTML.odp

Prezentace `WAP05a-HTML.odp` v adresáři `4-HTML/` obsahuje sto čtyřicet sedm slidů a obsahuje látku do předmětu WAP. V této prezentaci se nachází hlavně texty a ukázky kódu. Celkem na dvacet pět slidů program upozornil, aby je uživatel zkontroloval. Na daných slidech se vždy nachází jedna odrážka nebo odstavec, ve kterém chybí část textu, proto je hodnocení převodu textu 85%. Po grafické stránce jsou zvýrazněny části textu, které mají být zvýrazněné, ale v příkladech kódu se vygenerovalo špatné zarovnání, proto je hodnocení 75%.

### 5-DOM/WAP05c-HTMLDOM.odp

Prezentace `WAP05c-HTMLDOM.odp` v adresáři `5-DOM/` obsahuje osmnáct slidů a obsahuje látku do předmětu WAP. V této prezentaci se nachází převážně výpisy textu a schémata. Celkem na osm slidů byl uživatel upozorněn, aby provedl kontrolu. Toto velké množství je způsobeno způsobem uložení daného slidu v XML. Šablona pro zpracování prezentace nedokáže zpracovat prvky, které mají jako své přímé potomky textové i netextové uzly. V případě, že má pouze textové nebo pouze netextové, tak šablona pracuje správně. Uživatel je ale o tomto nedostatku informován a může dané slidy opravit sám, proto je hodnocení převedeného textu 50%. Při zaměření se na správně převedené slidy lze vyčíslit grafickou podobnost na 70%.

### 6-services/WAP09-Services.odp

Prezentace `WAP09-Services.odp` v adresáři `6-services/` obsahuje padesát devět slidů a obsahuje látku do předmětu WAP. Prezentace obsahuje převážně seznamy, výpisy kódu a obrázky. Celkem na patnáct slidů byl uživatel upozorněn, aby provedl kontrolu textu na slidu. Jednalo se převážně o výpisy kódu, proto převod textu je ohodnocen 72%. Grafickou podobnost lze vyčíslit na 70%.

## 7.3 Prezentace dostuplné na webu

### 1-web/file_example_ODP_1MB6.odp

Prezentace `file_example_ODP_1MB.odp` v adresáři `1-web/` obsahuje čtyři slidy a je volně dostupná na webu[1]. Prezentace obsahuje ukázku textu, grafu, tabulky a obrázku. Aplikace nedokáže převést graf, a proto je z převodu vynechán, vše ostatní se převede. Převod textu je tak ohodnocen 80% a grafická podobnost je 65%.

### 2-web/tuesday_d6.odp

Prezentace `tuesday_d6.odp` v adresáři `2-web/` obsahuje třicet sedm slidů a je volně dostupná na webu[2]. Prezentace obsahuje větší množství obrázků a bloků s kódem. Proto je vhodnější využít generování do HTML, které dokáže obrázky pozicovat. Převod bloků kódů aplikace nezvládne, a proto je uživatel upozorněn, aby dané slidy zkontroloval a následně upravil. Z grafické podobnosti jsou slidy hůře podobné z důvodu střídající se velikosti textu a špatného zarovnání. Převod textu je hodnocen 55% a grafická podobnost 55%.

[1] Dostupné na `https://file-examples.com/wp-content/uploads/2017/10/file_example_ODP_1MB.odp`
[2] Dostupné na `https://www.openoffice.org/marketing/ooocon2006/presentations/tuesday_d6.odp`

Tabulka 7.1: Přehled výsledků jednotlivých testů

| **Prezentace** | **Slidů** | **Převedený text** | **Grafická podobnost** |
|---|---|---|---|
| zaklad.odp | 4 | 100% | 100% |
| obhajoba.odp | 12 | 100% | 100% |
| WAP01-Konektivita.odp | 63 | 85% | 75% |
| WAP02-WWW.odp | 84 | 75% | 60% |
| WAP04b-XPath-XSLT.odp | 67 | 92% | 85% |
| WAP05a-HTML.odp | 147 | 85% | 75% |
| WAP05c-HTMLDOM.odp | 18 | 50% | 70% |
| WAP09-Services.odp | 59 | 72% | 70% |
| file_example_ODP_1MB.odp | 4 | 80% | 65% |
| tuesday_d6.odp | 37 | 55% | 55% |

# Kapitola 8

# Závěr

V první části práce byly shrnuty základní rysy formátu OpenDocument se zaměřením na prezentace uložené v tomto formátu. Byla popsána struktura dokumentu a významy jednotlivých částí. Dále byl představen prezentační framework Reveal.js, který umožňuje načítání obsahu prezentací zapsaného v jazyce Markdown. V teoretické části byly také představeny transformace XSLT, které umožňují zpracovávat XML soubory. Všechny tyto informace byly zapsány s ohledem na další využití v této práci. Jednotlivé kapitoly nepopisují kompletní problematiku, ale popisují její část pro pochopení praktické části práce.

Praktickou částí práce bylo vytvoření aplikace, která automaticky převede prezentaci ve formátu OpenDocument do jazyka Markdown. Ten bude následně načten do prezentace pomocí Reveal.js. Důraz byl kladen na správné převedení textů prezentace. Grafická podobnost výsledné prezentace s původní je pouze částečná a není jí kladen takový důraz.

Základem vytvořené aplikace je script v jazyce Python, který provádí jednotlivé kroky převodu. Nejdůležitějším krokem převodu je použití XSLT stylu, který slouží k převodu XML s obsahem původní prezentace. Největší překážkou při tvorbě aplikace byl převod obsahu do Markdown, protože některé konstrukce převést nelze. Řešením tohoto problému bylo přidání možnosti generovat obsah jednotlivých slidů do HTML, které část těchto nedostatků odstraní. Aplikace není schopná převést složitější objekty v prezentaci, jako např. grafy, schémata nebo animace.

Podnětem ke vzniku této aplikace bylo praktické využití při převodu přednášek do předmětu WAP – Internetové aplikace. Aplikace byla využita k převodu těchto prezentací a následné úpravě do aktualizované podoby.

Z testování aplikace vyplynulo, že je schopná převést většinu textového obsahu do Markdown. Na konstrukce, které nedokáže převést upozorní uživatele, aby obsah daného slidu upravil ručně. Důvodem, že aplikace není schopná obsah převést, je složitý zápis obsahu prezentace v XML. Variabilita zápisu je veliká a XSLT styl v aplikaci nedokáže zpracovat všechny tyto konstrukce.

Z vizuální stránky prezentace se převádí pouze styl částí textu, které jsou zvýrazněny oproti okolí. Do výsledné prezentace se tak pomocí CSS zobrazí pouze zvýraznění textu pomocí vlastností fontu a rozdílné barevnosti různých částí.

Možným pokračováním této práce by bylo rozšířit převod o prvky prezentace, které nejsou podporovány, jako jsou např. grafy nebo schémata. Aplikace by mohla být rozšířena o podporu animací a postupného zobrazování textu v Reveal.js. Upravením XSLT stylu by bylo možné se vyhnout výpisům, že aplikace nedokáže daný text převést.

V případě úprav této aplikace nebo práce na podobném zadání, bych doporučil převod do Markdown pouze pro jednoduché konstrukce, v případě složitějších konstrukcí je velmi

náročné Markdown použít. V případě, že to okolnosti dovolí, je výhodnější složité konstrukce generovat v HTML.

Nevýhodou využití Markdown je postupné umísťování jednotlivých prvků pouze pod sebe. V případě nutnosti umístit dva nebo více prvků vedle sebe je nemožné použít pouze jazyk Markdown.

# Literatura


[1] Brauer, M.: *Open Document Format for Office Applications (OpenDocument) v1.1.* Únor 2007, [Online; navštíveno 7.12.2018].
URL http://docs.oasis-open.org/office/v1.1/OS/OpenDocument-v1.1-html/OpenDocument-v1.1.html

[2] Hattab, H. E.: *reveal.js.* [Online; navštíveno 30.11.2018].
URL https://github.com/hakimel/reveal.js

[3] Kosek, J.: *Jmenné prostory.* [Online; navštíveno 4.5.2019].
URL https://www.kosek.cz/vyuka/4iz238/slidy/xsd/foil18.html

[4] Kosek, J.: *XSLT v příkladech.* [Online; navštíveno 7.12.2018].
URL https://www.kosek.cz/xml/xslt/index.html

[5] Kosek, J.: *XML pro každého: podrobný průvodce.* Grada, 2000, ISBN 80-7169-860-1.

[6] Krčmář, P.: *ISO/IEC 26300? OpenDocument.* [Online; navštíveno 12.03.2019].
URL https://www.root.cz/clanky/iso-iec-26300-opendocument/

[7] Pritchard, A.: *Markdown Cheatsheet.* [Online; navštíveno 5.12.2018].
URL https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet

[8] Štrauch, A.: *Reveal.js: efektní prezentace pomocí HTML.* [Online; navštíveno 25.02.2019].
URL https://www.root.cz/clanky/reveal-js-efektni-prezentace-pomoci-html/