



**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

**NÁSTROJ PRO KONTROLU DIPLOMOVÝCH PRACÍ**

THESES CHECKER

**BAKALÁŘSKÁ PRÁCE**

BACHELOR'S THESIS

**AUTOR PRÁCE**

AUTHOR

**MICHAELA MACKOVÁ**

**VEDOUCÍ PRÁCE**

SUPERVISOR

**Ing. TOMÁŠ MILET, Ph.D.**

**BRNO 2023**

## Zadání bakalářské práce



144733

Ústav: Ústav počítačové grafiky a multimédií (UPGM)  
Studentka: **Macková Michaela**  
Program: Informační technologie  
Specializace: Informační technologie  
Název: **Nástroj pro kontrolu diplomových prací**  
Kategorie: Počítačová grafika  
Akademický rok: 2022/23

### Zadání:

1. Nastudujte nástroje pro automatickou kontrolu kvality technických dokumentací, způsoby hledání chyb, formální, typografické a jazykové požadavky technických dokumentací. Nastudujte a sesbírejte často se vyskytující chyby v technických dokumentacích.
2. Navrhněte aplikaci, která ve vstupním souboru pdf vyznačí chyby a navrhne způsob řešení.
3. Implementujte navrženou aplikaci tak, aby byla uživatelsky co nejpřívětivější a nejsnáze se používala.
4. Vyhodnoťte nástroj na již zveřejněných pracích a porovnejte její výsledky s posudky oponentů.
5. Práci zhodnoťte, zveřejněte a vytvořte demonstrační video.

### Literatura:

- Biernátová, O. a Skůpa, J. Bibliografické odkazy a citace dokumentů [online]. Brno: Citace.com, září 2011. Dostupné z: <http://www.citace.com/download/CSN-ISO-690.pdf>.
- Černá, A., Chromý, J., Konečná, H. et al. Internetová jazyková příručka – Ústav pro jazyk český Akademie věd ČR, v. v. i. [online]. Centrum zpracování přirozeného jazyka FI MU, 2019. Dostupné z: <http://prirucka.ujc.cas.cz/>.
- Hlavsa, Z. et al. Pravidla českého pravopisu. 2. vyd. Academia, 2009. ISBN 80-200-1327-X.
- Zemčík, P. Směrnice děkana č. 7/2018 – Úprava, odevzdávání a zveřejňování závěrečných prací na FIT VUT v Brně [online]. 2018. Dostupné z: <https://www.fit.vut.cz/fit/info/smernice/sm2018-07.pdf>.

Při obhajobě semestrální části projektu je požadováno:  
První dva body a kostra aplikace.

Podrobné závazné pokyny pro vypracování práce viz <https://www.fit.vut.cz/study/theses/>

Vedoucí práce: **Milet Tomáš, Ing., Ph.D.**  
Vedoucí ústavu: Černocký Jan, prof. Dr. Ing.  
Datum zadání: 1.11.2022  
Termín pro odevzdání: 17.5.2023  
Datum schválení: 31.10.2022

## Abstrakt

Cílem této práce je vytvoření aplikace, která zkontroluje technickou zprávu a označí všechny nalezené chyby pomocí PDF anotací. Technická dokumentace této práce rozebírá strukturu PDF souboru, často vyskytované chyby v diplomových pracích, vývoj webu s pomocí frameworku Django a probírá existující knihovny pro úpravu PDF dokumentů. Výsledná aplikace je implementována v jazyku Python a s pomocí frameworku Django je přístupná jako webový nástroj. Vytvořené řešení dokáže nalézt šest převážně typografických chyb, které se často vyskytují v diplomových pracích. Nalezené chyby jsou graficky označeny a upravený PDF soubor je poté zobrazen přímo na webové stránce. Výsledný nástroj je volně dostupný a pomáhá studentům i vyučujícím při kontrole vytvářených technických zpráv.

## Abstract

The main goal of this work is to create an application that checks technical reports and marks all the found errors with PDF annotations. The technical documentation of this thesis breaks down the structure of a PDF file, commonly found mistakes in graduate theses, web development using the Django framework and discusses existing libraries for editing PDF documents. The resulting application is implemented in Python and is accessible as a web tool with the help of the Django framework. The developed solution recognizes six mostly typographical errors frequently found in graduate theses. The mistakes found are visually marked and the edited PDF file is then displayed directly on the web page. The resulting tool is freely available and helps students and supervisors to correct the technical reports the students create.

## Klíčová slova

PDF, typografické chyby, časté chyby, technická zpráva, webová aplikace, PDF anotace, Django, Python, struktura PDF

## Keywords

PDF, typographical mistakes, frequent mistakes, technical paper, web application, PDF annotations, Django, Python, PDF structure

## Citace

MACKOVÁ, Michaela. *Nástroj pro kontrolu diplomových prací*. Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Milet, Ph.D.

# Nástroj pro kontrolu diplomových prací

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana Ing. Tomáše Mileta, Ph.D. Uvedla jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpala.

.....  
Michaela Macková  
15. května 2023

## Poděkování

V této sekci je možno uvést poděkování vedoucímu práce a těm, kteří poskytli odbornou pomoc (externí zadavatel, konzultant apod.). **[[TODO:]]**

# Obsah

<b>1</b>	<b>Úvod</b>	<b>2</b>
<b>2</b>	<b>Typografie a často vyskytované chyby v diplomových pracích</b>	<b>4</b>
2.1	Rychlokurz typografie . . . . .	4
2.2	Přetečení objektů za okraj stránky . . . . .	5
2.3	Chybné použití spojovníku . . . . .	5
2.4	Chybějící popis kapitoly . . . . .	6
2.5	Nadpisy třetí a větší úrovně v obsahu . . . . .	6
2.6	Absence vektorové grafiky . . . . .	7
2.7	Nepoužívání pevné mezery . . . . .	8
<b>3</b>	<b>PDF soubor</b>	<b>10</b>
3.1	Formát PDF . . . . .	10
3.2	Grafika v PDF . . . . .	16
3.3	Reprezentace anotací v PDF souboru . . . . .	20
3.4	Programovací jazyky a knihovny pro zpracování a anotování PDF souborů . . . . .	22
<b>4</b>	<b>Návrh a implementace aplikace Theses Checker</b>	<b>25</b>
4.1	Specifikace požadavků . . . . .	25
4.2	Návrh aplikace . . . . .	26
4.3	Využité technologie . . . . .	27
4.4	Implementace webové aplikace . . . . .	28
4.5	Program pro použití v příkazovém řádku . . . . .	30
4.6	Implementace programu pro vyhledání chyb a jejich následné vyznačení . . . . .	31
<b>5</b>	<b>Testování a zhodnocení výsledné aplikace</b>	<b>39</b>
5.1	Ověření správné funkcionality vyhledávání chyb . . . . .	39
5.2	Uživatelské dotazníky . . . . .	39
5.3	Znamé chyby aplikace . . . . .	42
5.4	Možné budoucí rozšíření aplikace . . . . .	42
<b>6</b>	<b>Závěr</b>	<b>43</b>
	<b>Literatura</b>	<b>44</b>

# Kapitola 1

## Úvod

Pro úspěšné dokončení vysokoškolského studia musí student napsat několik desítek stran textu zvaného závěrečná práce. Tato technická zpráva by měla být před jejím zveřejněním několikrát překontrolována, ale ve většině případů se nenaleznou všechny vyskytované chyby. Studenti se při kontrole často obrací na své známé. Tito lidé však často nebývají seznámeni s typografickými pravidly, a tak se tyto typy chyb často přehlíží. Při velké koncentraci chyb se může výsledná práce zdát „odfláklá“ a může tak být sníženo její ohodnocení, případně může být tato práce přímo zamítnuta.

V současné době existuje několik nástrojů pro kontrolu textu. Pro anglický jazyk existuje například Grammarly<sup>1</sup> a pro český jazyk existuje nástroj Lingea<sup>2</sup> a nový nástroj Opravidlo<sup>3</sup>, který je nyní dostupný pouze v beta verzi. Tyto nástroje se však převážně zaměřují na gramatické chyby a typografické chyby tak bývají často přehlíženy. Při správné volbě textového procesoru jsou některé typografické chyby automaticky při psaní opravovány, ale opět na to není žádná aplikace zaměřena. Jedním z takových textových procesorů je například Microsoft Word, který má navíc podporu pro více jazyků.

Cílem této práce bylo zkoumáním zveřejněných diplomových prací a jejich posudků od oponenta nalézt, které chyby se v těchto pracích často vyskytují. Dalším krokem bylo navrhnout a vytvořit lehce dostupnou aplikaci, která nalezne tyto (převážně typografické) chyby a označí je přímo uvnitř poskytnutého PDF souboru. Tato aplikace měla být dostupná bez nutnosti instalování a měla být co nejvíce intuitivní.

Úvodní stránka výsledného webového nástroje s názvem Theses Checker<sup>4</sup> lze vidět na obrázku 1.1. Přesněji lze úvodní stránku vidět na obrázku 1.1a a na obrázku 1.1b lze vidět úvodní stránku při čekání na zpracování nahraného dokumentu. Na obrázku 1.2 je poté vidět PDF výstup vytvořené aplikace. Toto anotované PDF je pro příjemnější používání aplikace přímo zobrazené na webové stránce. Pro účely kontrolování několika souborů najednou byl vyvinut i program pro použití této aplikace v příkazovém řádku. Tento program navíc od webové aplikace podporuje volbu kontrol, které se provedou při zpracování vstupního PDF dokumentu.

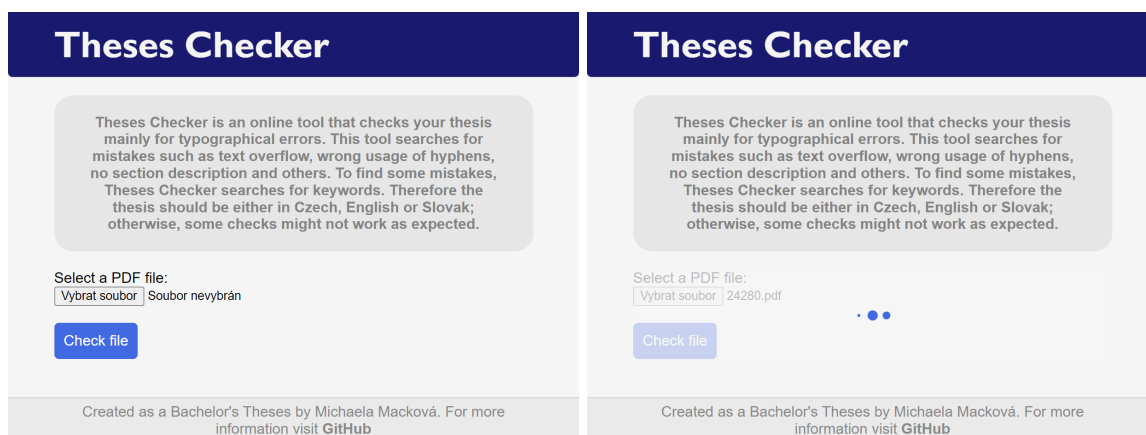
---

<sup>1</sup>Grammarly lze nalézt na <https://grammarly.com/>

<sup>2</sup>Nástroj Lingea je dostupný na <https://korektor.lingea.cz/>

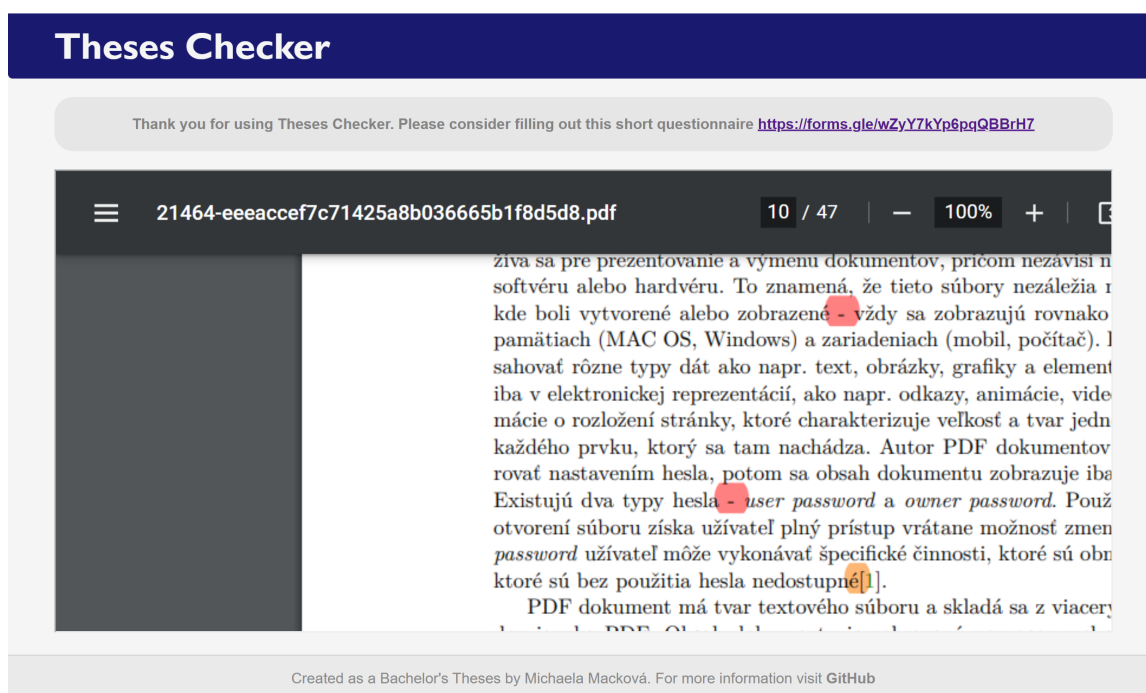
<sup>3</sup>Opravidlo existuje v beta verzi na adrese <https://www.opravidlo.cz/>

<sup>4</sup>Theses Checker je dostupný na adrese <https://theseschecker.eu.pythonanywhere.com/>



(a) V tomto obrázku je stránka, která se zobrazí po otevření webové aplikace Theses Checker (b) Toto je úvodní stránka s načítacím elementem, který se zobrazí po stisknutí tlačítka Check file

Obrázek 1.1: Obrázky ukazují úvodní stránku vytvořené webové aplikace Theses Checker



Obrázek 1.2: Tento obrázek ukazuje webovou stránku s výstupem z vytvořené aplikace

V kapitole 2 této bakalářské práce jsou popsány často vyskytované chyby v diplomových pracích, které byly zjištěny jako součást této práce. Dále je popsán formát souboru PDF (kapitola 3) a návrh a implementace samotné aplikace (kapitola 4), což zahrnuje i vývoj webové aplikace s pomocí frameworku Django. Kapitola 5 poté popisuje postup testování a vyhodnocení výsledného produktu.

## Kapitola 2

# Typografie a často vyskytované chyby v diplomových pracích

U psaní textu se autor musí řídit nejen gramatickými, ale i typografickými pravidly. Toto platí především při psaní odborné práce. Větší množství chyb v textu práce může mít za následek to, že i kvalitně odvedená praktická realizace práce se bude zdát neuspokojivá.

Chyby mohou být způsobeny z nepozornosti, anebo z neznalosti, přičemž druhá možnost je pro autora textu horší, jelikož i po několikátém přečtení nemusí pisatel vůbec poznat, že se jedná o chybu. Správnou volbou textového editoru si tvůrce textu může usnadnit hledání některých chyb. Několik dnešních textových procesorů poskytuje alespoň částečnou kontrolu pravopisu, nicméně tato kontrola umí ve spoustě případů upozornit převážně jen na překlepy. Významové chyby, jako je například záměna slov *típ*, *typ* nebo *autorizace*, *autentizace*, bývají často touto automatickou kontrolou zanedbávány. První část této kapitoly popisuje čím se zabývá obor typografie a v dalších částech je popsáno několik chyb, které lze nalézt v mnoha diplomových pracích. Tyto části dále uvádějí, jak se těmto chybám vyhnout.

### 2.1 Rychlokurz typografie

Typografie je obor zabývající se návrhem a úpravou všech druhů tiskovin. Především se zabývá grafickými prvky (např. obrázky a písmo) a jak na člověka působí celkový vzhled celého díla. Informace získané z knihy Praktická typografie [9] uvádí, že typografie už se rozvíjí přes více jak pět set let a základy její práce se téměř neměnily. I když dříve byla typografická pravidla důležitá převážně pro sazeče, nyní může pomocí počítače sázet úplně každý. Toto však zavedlo problém, protože velké množství těchto uživatelů o pojmu typografie nikdy neslyšeli. Typografická pravidla se řídí podle norem, avšak jejich používání není vynutitelné. I přesto se doporučuje dodržování těchto typografických pravidel, jelikož při jejich použití dokáže čtenář lépe vnímat celé dílo. Každý národ má vlastní soubor typografických pravidel a při tvorbě cizojazyčného textu, by se měla tato pravidla respektovat.

V minulosti, kdy se opisovaly knihy ručně, bylo základním prvkem písmeno. Základní prvek se od té doby postupně měnil a nyní je tímto prvkem odstavec. Nejviditelnější parametr odstavce je zarovnání, které má ve většině počítačových sázecích programů čtyři druhy – do bloku, doleva, doprava a na střed. Zarovnání odstavce do bloku je běžný v knižní sazbě a u novin a časopisů. Dalším důležitým prvkem je písmo, jehož volba je při psaní textu důležitá. Častý problém je, že některá písma neobsahují kompletní znakovou sadu, tedy s tímto

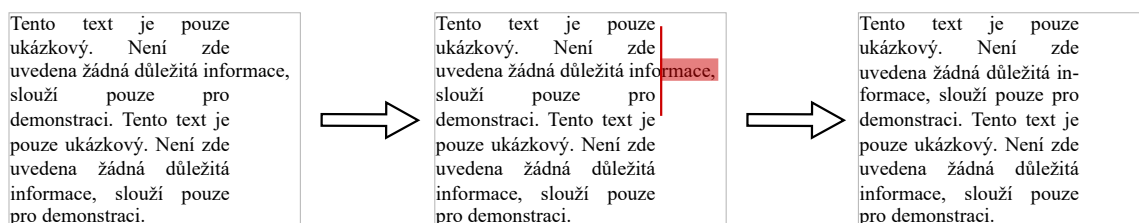


písmem nelze vysázet speciální české (či jiné) znaky. Pro sazbu češtiny nebo slovenštiny je potřeba použít písmo, které obsahuje minimálně všechny znaky použitého jazyka.

Po umístění textu na stránce se typografie ještě zabývá umístěním samotných znaků, jako je například otazník, pomlčka či uvozovka. Pro matematickou sazbu, chemickou sazbu a sazbu not je nejlepší použít specializovaný program, který obsahuje potřebné funkce. Jeden z takových programů je právě  $\text{\TeX}$ , nebo je též možné použít vzorce z MS Wordu.

## 2.2 Přetečení objektů za okraj stránky

Přetečení textu za okraj se nejčastěji vyskytuje, když student píše svou diplomovou práci s pomocí jazyka  $\text{\LaTeX}$ . Obvykle je to způsobeno tím, že program nedokáže automaticky zalomit slovo na konci řádku, jak je ukázáno na obrázku 2.1. Toto lze opravit napověděním možného zalomení problematického slova nebo přeformulováním věty, kde se daná chyba vyskytuje. Další typ této chyby je přetečení obrázku za okraj, který se nestává tak často, ale lze jej udělat v několika textových editorech.

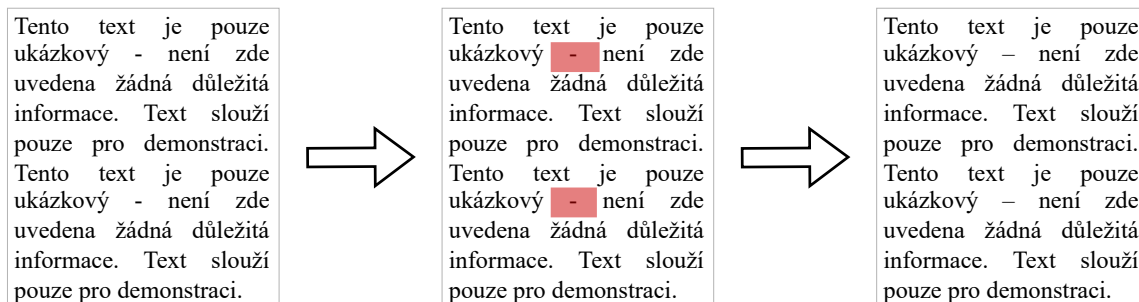


Obrázek 2.1: Tento obrázek obsahuje ukázkou textu, který přetéká za hranici stránky, označení tohoto přetečení a jeho následnou opravu

## 2.3 Chybné použití spojovníku

Nesprávné používání spojovníku je chyba, která se vyskytuje nejen v diplomových pracích. Spojovník (-) je graficky velmi podobný pomlčce (–), ale významově se značně liší. Pravidla pro psaní těchto znaků, uvedená v internetové příručce Ústavu pro jazyk český [1, Sekce: *Spojovník a Pomlčka*], říkají, že spojovník se píše bez mezer mezi výrazy, které spojuje. Výjimkou je, naznačuje-li spojovník neúplné slovo. Obecně se tedy v češtině tento znak užívá, chce-li autor vyjádřit, že jím spojené výrazy tvoří těsný významový celek. Pomlčka se oproti spojovníku využívá pro oddělování částí projevu, vyjádření rozsahu, vztahu nebo vyznačení přestávky v řeči, pro uvození přímé řeči a pro vyjádření celého čísla při psaní peněžních částek. Odděluje se z obou stran mezerami. Komplikovanější situace nastane pouze tehdy, když je toto znaménko použito ve funkci výrazů a, až, od, do nebo proti. Spojovník (-) i pomlčka (–) bývají často zaměňovány se znaménkem minus (−), to však má též své grafické i významové odlišnosti. V knize Průvodce tvorbou dokumentů [16, k. 19.24, s. 148–149] je vysvětleno, že znak minus má stejnou šíři i umístění jako znak plus. Znak minus se používá ve dvou významech, a to pro označení záporné hodnoty, nebo pro označení operace odčítání. Sazba se v obou případech liší: pro označení záporné hodnoty se znak minus a následující operand píše bez mezery, pro psaní minus jako odčítání se však mezera uvádí z obou stran tohoto znaménka. Internetová příručka Ústavu pro jazyk český [1, Sekce: *Pomlčka*] však uvádí, že je v korespondenci dovoleno znak minus (−) nahradit pomlčkou (–).

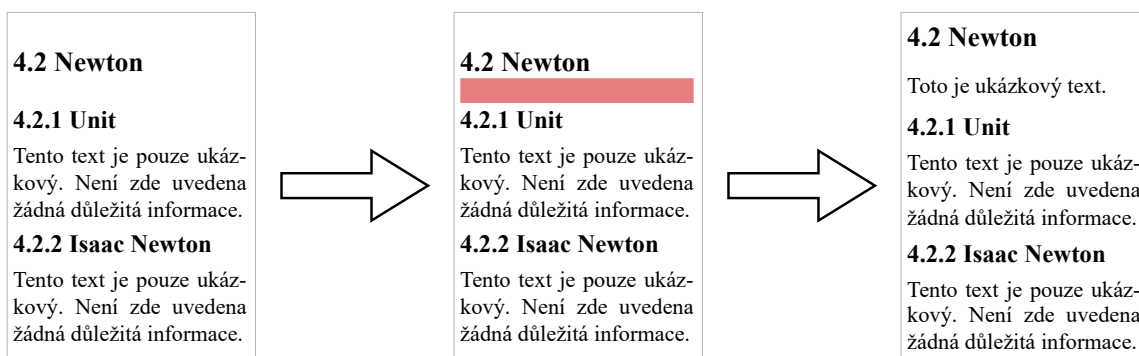
Podle článku Základy typografie [17] se tato chyba (naznačena na obrázku 2.2) vyskytuje v textu kvůli absenci znaku pomlčky na klávesnici. Místo znaku pomlčky, který se při psaní textu pravděpodobně používá častěji, se na klávesnici vyskytuje právě znak spojovníku. I když nyní už spousta textových editorů dokáže automaticky nahradit spojovník za pomlčku, tato náhrada nemusí být stoprocentní. V programu L<sup>A</sup>T<sub>E</sub>X se spojovník zapíše přímo z klávesnice jako -, pomlčku je možno zapsat pomocí dvou spojovníků -- a znaménko minus je zapsáno jako spojovník v matematickém prostředí \$-\$ nebo též \$\$-\$\$.



Obrázek 2.2: Na obrázku je ukázáno chybné použití spojovníku, poté je tato chyba označena a nakonec je ukázáno její opravení

## 2.4 Chybějící popis kapitoly

I když je kapitola rozdělena na několik podkapitol, musí i samotná kapitola obsahovat úvod do této kapitoly. Leaný blog [18] vysvětluje, že pokud není uveden popis mezi kapitolou a její podkapitolou, působí poté práce nedopracovaně. Tuto skutečnost lze vidět i na ukázce v obrázku 2.3. V tomto místě se hodí napsat 1–2 odstavce, kde bude vysvětlené o čem daná kapitola pojednává a co se v ní čtenář dozví.

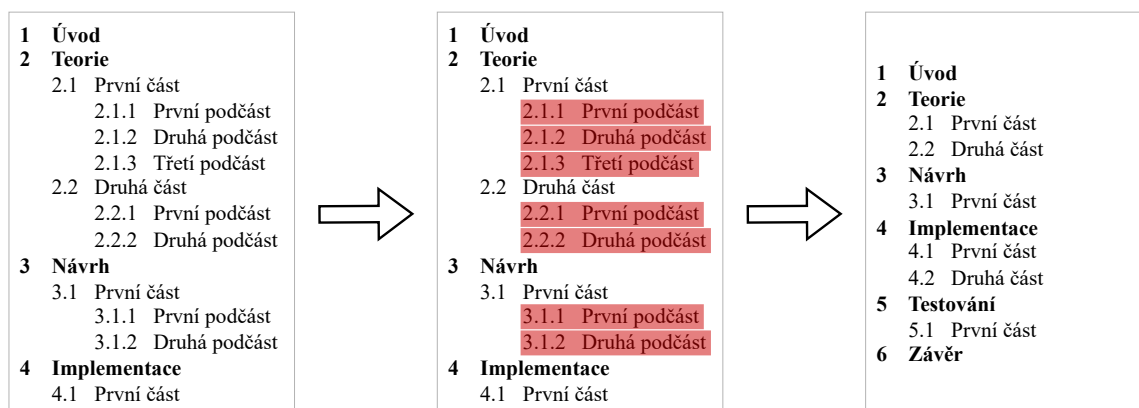


Obrázek 2.3: Obrázek obsahuje ukázkou chybějícího popisu kapitoly (tzn. mezi dvěma nadpisy se nevyskytuje text). V další části tohoto obrázku je tato chyba označena a poté opravena

## 2.5 Nadpisy třetí a větší úrovně v obsahu

V diplomové práci není vhodné v obsahu uvádět nadpisy třetí či větší úrovně. Jak je vidět na obrázku 2.4, obsah je poté nepřehledný a zbytečně dlouhý. Samotná třetí úroveň nadpisů

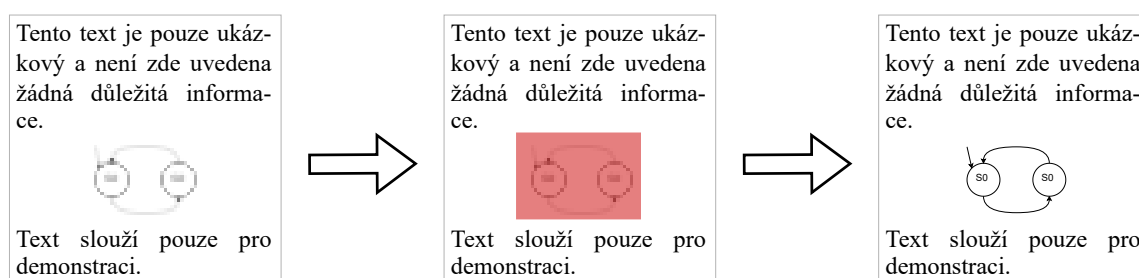
je velmi podrobná, ale v diplomové práci ji lze použít v případě, když bude nečíslovaná. V programu  $\text{\LaTeX}$  tohoto lze dosáhnout příkazem `\subsection*{}`. Nadpisy čtvrté a větší úrovně by se v diplomové práci vyskytovat neměly.



Obrázek 2.4: Tento obrázek obsahuje ukázkou nevhodného uvedení nadpisů třetí úrovně v obsahu, označení nadpisů, které do obsahu nepatří a následnou opravu této chyby

## 2.6 Absence vektorové grafiky

Při vkládání obrázku do textu se autor musí zabývat několika otázkami a jedna z nich je určité jeho kvalita. Pokud má obrázek moc malé rozlišení, nevypadá v odborné práci dobře. I přesto, že se obrázek na displeji zdá dostatečně kvalitní, při tisku může být daný obrázek „rozkostičkovaný“. Toto nevhodné použití lze vidět i na obrázku 2.5. Tento problém nekvalitního rozlišení může být vyřešeno použitím vektorového obrázku. Podle knihy Průvodce tvorbou dokumentů [16, k. 10, s. 56–61] je zásadní výhodou vektorového obrázku jeho uložení, díky kterému si obrázek ponechá vysokou kvalitu i v různém zvětšení. Ale použití vektorové grafiky není vždy vhodné a v některých případech není ani možné. V knize je proto uvedeno doporučení použít vektorovou grafiku (formáty SVG, EPS a PDF) na schémata a loga, rastrovou grafiku formátu JPG pro fotografie a pro ostatní rastrovou grafiku použít formát PNG.



Obrázek 2.5: Na tomto obrázku je uvedena ukázkou absence vektorové grafiky, dále je tato chyba označena a nakonec je ukázáno její opravení použitím vektorového obrázku

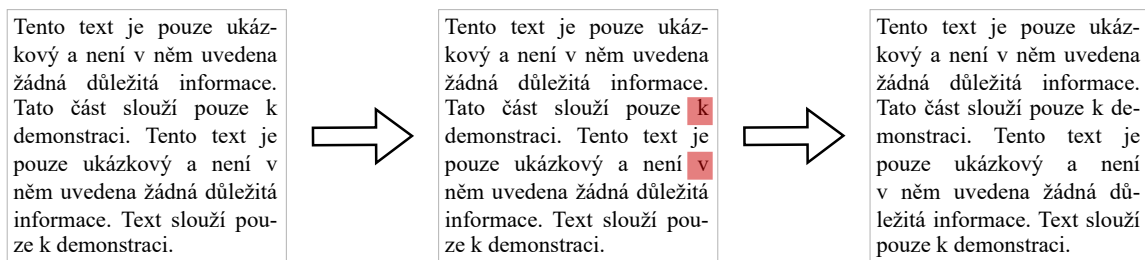
## 2.7 Nepoužívání pevné mezery

Jako spousta jiných věcí, i psaní mezer má svá pravidla. Jak zmiňuje článek Jak se vyhnout typografickým hříchům [8], i v něčem tak samozřejmém, jako je psaní pouhé mezery, se často chybuje: mezera se musí psát za tečkou (nebo též čárkou), ne před ní a píše se vždy jen jedna, data se píšou ve formátu *d. m. yyyy* a čísla se oddělují mezerou po tisících (s výjimkou letopočtu). Při psaní se může stát, že mezera spojující znaky nebo čísla vyjde na konec řádku a tyto znaky by se rozdělily. Tento případ lze pozorovat i na obrázku 2.6. Pro zamezení takových případů existuje právě pevná (nebo též nedělitelná) mezera.

Pevná mezera se zobrazí stejně jako normální mezera, ale na rozdíl od normální mezery, spojí dohromady příslušné znaky a zablokuje jejich rozdělení na konci řádku. Podle pravidel internetové jazykové příručky [1, Sekce: *Zalomení řádků a nevhodné výrazy na jejich konci*] se má pevná mezera použít v těchto případech (převzato a upraveno):

- ve spojení neslabičných předložek *k*, *s*, *v*, *z* s následujícím slovem, např. *v obrázku*, *z funkce*,
- ve spojení slabičných předložek *o*, *u* a spojek *a*, *i* s následujícím výrazem, např. *o kapitole*, *a to*,
- členění čísel, např. *2 301 000*, *3,141 592 65*,
- mezi číslem a značkou, např. *25 %*, *© 2008*,
- mezi číslem a zkratkou počítaného předmětu nebo písmennou značkou jednotek a měn, např. *24 hod.*, *100 m*, *3 000 Kč*, *500 ¥*,
- mezi číslem a názvem počítaného jevu, např. *obrázek 5*, *12 metrů*, *I. patro*,
- v kalendářních datech mezi dnem a měsícem, rok však lze oddělit, např. *3. 5. 2000*, *26. dubna 2023*
- v měřítkách map, plánů a výkresů, v poměrech nebo při naznačení dělení, např. *4 : 7*, *1 : 10 000*, *12 : 2 = 6*,
- v telefonních, faxových a jiných číslech členěných mezerou, např. *+420 603 999 226*,
- ve složených zkratkách (v případě nutnosti se doporučuje dělit podle dílčích celků), v ustálených spojeních a v různých kódech, např. *s. r. o.*, *m n. m.*, *ISO 690*,
- mezi zkratkami typu *tj.*, *tzv.*, *tzn.* a výrazem, který za nimi bezprostředně následuje, např. *tzv. pipeline*,
- mezi zkratkami rodných jmen a příjmeními, např. *T. Milet*,
- mezi zkratkou titulu nebo hodnosti uváděnou před osobním jménem, např. *p. Macková*, *Ing. Novák*.

Zapsání pevné mezery závisí na použitém textovém editoru. I když spousta z nich již umí tuto pevnou mezeru automaticky doplnit, nemusí mít tato automatizace stoprocentní úspěšnost. V programu L<sup>A</sup>T<sub>E</sub>X se pevná mezera zapíše znakem tildy (~) a v editoru WORD se zapíše pomocí kombinace kláves *Ctrl Shift mezera*.



Obrázek 2.6: Tento obrázek obsahuje ukázkou zalomení řádku na místě, kde by se měla vyskytovat pevná mezera. V druhé části je tato chyba označena a v poslední části je uvedeno její možné opravení použitím nezlomitelné mezery

## Kapitola 3

# PDF soubor

Tato kapitola popisuje strukturu PDF dokumentu. Přesněji popisuje vnitřní strukturu a uložení objektů. Dále popisuje, jak funguje grafika pro zobrazení PDF dokumentů. V neposlední řadě je představeno několik knihoven, které dokáží s PDF dokumenty manipulovat.

### 3.1 Formát PDF

Většina uživatelů, kteří zachází s PDF soubory, nepotřebují znát vnitřní složení PDF dokumentů. Spousta programů a knihoven pro vytváření či úpravu PDF dokumentu dokáže při práci s tímto souborem dostatečně odstínit od syntaxe PDF formátu. Avšak pro pokročilejší práci s PDF dokumenty není na obtíž si zjistit pár základních informací o uložení dat v tomto formátu. PDF dokument má podobu textového souboru a na jeho pochopení jsou v této sekci vysvětleny jeho 4 základní stavební bloky. Tato sekce čerpá informace ze standardu PDF 32000-1 [13, k. 7, s. 11–109].

#### Objekty

Objekty jsou jedny ze základních stavebních bloků PDF dokumentu. PDF rozeznává osm typů objektů:

- **Boolean objekt** – Tyto objekty reprezentují logickou hodnotu. Můžou nabýt dvou hodnot, které jsou označeny klíčovými slovy `true` a `false`.
- **Číselný objekt** – Obsahovaná číselná hodnota může být celé, nebo reálné číslo. U zápisu reálných čísel se používá desetinná tečka, například `-3.62`, `.054`, `+238.45`. Celá čísla mohou být například `500`, `+3`, `-21`.
- **Řetězcový objekt (string)** – Řetězec lze zapsat dvěma způsoby, a to jako klasický řetězec, nebo jako řetězec v hexadecimální podobě. Klasický řetězec je zapsán jako posloupnost znaků uzavřená v kulatých závorkách, například `(Totó je string)`. V tomto typu řetězce je možné používat escape sekvence začínající zpětným lomítkem. Řetězec psaný v hexadecimální podobě lze zapsat jako posloupnost hexadecimálních číslic uzavřenou mezi znaky menší než a větší než, například `<48656c6c66f>`, `<776F726C64>`. Každá dvojice hexadecimálních číslic tvoří jeden znak zakódovaný v ASCII podobě.
- **Jmenný objekt** – Objekt jména je sekvence znaků. Znak, který se mohou použít ve jménu jsou takové, které zapadají do rozmezí mezi znakem vykřičníku (!) a znakem

tildy (~). Ostatní znaky se mohou zapsat jako hexadecimální hodnota požadovaného znaku, kterou předchází znak mřížky (#). Jméno musí začínat lomítkem, které se nebere jako jeho součást. Zapsané jméno může být například /Name, /1.6\*xyz, /C#23.

- **Objekt pole** – Objekt typu pole je kolekce, která obsahuje objekty. Tyto objekty nemusí být stejného typu – heterogenní pole. Zapisuje se jako prvky pole, které jsou odděleny bílým znakem, uzavřené v hranatých závorkách. Prvkem pole může být objekt pole. Validní pole je například [(string) -25 [2.75 /Name] true].
- **Slovníkový objekt** – Slovník je kolekce, jejíž prvky jsou dvojice objektů. První prvek z této dvojice se nazývá *klíč* a vždy to musí být objekt typu jméno. Ve slovníku nesmí existovat více záznamů se stejným klíčem. Druhý prvek ze dvojice se nazývá *hodnota*. Tento prvek může být objekt jakéhokoli typu. Slovník je uvozen dvojitým znakem menší než a dvojitým znakem větší než, například «/Key1 2.6 /Key2 /Value2».
- **Objekt datového toku (stream)** – Stream je sekvence bajtů, která má neomezenou délku. Používá se především pro ukládání velkého množství dat, což je například obrázek. Tento objekt se zapisuje jako slovník, za nímž následuje klíčové slovo **stream**, po kterém se musí vyskytovat konec řádku. Následují bajty datového toku, které jsou ukončeny koncem řádku a klíčovým slovem **endstream**. Vyskytovaný slovník nesmí být uveden nepřímým odkazem a musí se v něm uvádět délka datového toku v bajtech (pod klíčem **Length**). Každý objekt datového toku musí být zároveň nepřímým objektem (vysvětleno později v této sekci). Validní objekt datového toku je například uveden ve výpise 3.1:

---

```
12 0 obj
<</Length 20 /Filter /FlateDecode>>
stream
xšcbd'řg'b' '8 "y
DZn
endstream
endobj
```

---

Výpis 3.1: V tomto výpisu je uveden příklad nepřímého objektu s identifikátorem 12 0. Tento nepřímý objekt je zároveň objekt datového toku s velikostí 20 B, který musí být pro přečtení dekodován FlateDecode filtrem

- **Null objekt** – Null objekt je speciální objekt, který nabývá pouze hodnoty **null**.

Každému objektu se může přiřadit jednoznačný identifikátor, takový objekt se poté nazývá **nepřímý objekt**. Na nepřímý objekt potom může být odkazováno z jiného objektu, čehož je často využíváno například ve slovníku, kde je uveden klíč a hodnota je nepřímý odkaz na objekt. Identifikátor nepřímého objektu má dvě části. První část je kladné celé číslo, kterému se říká *číslo objektu*. Druhou částí je tzv. *číslo generace*, které je pro nově generovaný dokument 0. Toto číslo musí být vždy nezáporné celé číslo. Nepřímý objekt se zapíše jako číslo objektu, poté bílý znak a číslo generace. Následuje samotný objekt uzavřen mezi klíčovými slovy **obj** a **endobj**. Validní nepřímý objekt je uveden ve výpise 3.2:

---

7 0 obj  
<504446>  
endobj

---

Výpis 3.2: Tento výpis obsahuje příklad nepřímého řetězcového objektu s identifikátorem 7 0

Nepřímý objekt lze referencovat pomocí *nepřímého odkazu*. Nepřímý odkaz se zapíše číslem objektu, číslem generace a klíčovým slovem R, oddělené bílými znaky. Odkaz na nepřímý objekt, který je uveden ve výpise 3.2, se zapíše jako 7 0 R.

## Struktura souboru

Tato část popisuje, jak jsou výše popsané objekty uloženy v PDF dokumentu. Též popisuje, jak je k nim přistupováno a jak jsou aktualizovány. PDF soubor se je rozdělenný do 4 částí:

- **Hlavička** – Hlavička se vždy vyskytuje na prvním řádku souboru. Má tvar komentáře, přesněji %PDF- a bezprostředně za tím následuje číslo PDF verze, například %PDF-1.7.
- **Tělo** – Tělo se skládá z posloupnosti nepřímých objektů. Tyto nepřímé objekty popisují vzhled celého dokumentu (stránky, fonty, obrázky, ...).
- **Tabulka křížových odkazů** – Tabulka křížových odkazů se používá při přístupu k nepřímým objektům. Tato tabulka obsahuje informaci o umístění každého nepřímého objektu. Tabulka se skládá z jedné nebo více *sekcí křížových odkazů*, která musí začínat řádkem s klíčovým slovem xref. Každá sekce může být rozdělena na několik *podsekcí křížových odkazů*. Tyto podseky vždy začínají řádkem, na kterém se vyskytují dvě čísla. První číslo označuje první objekt v záznamu podseky a druhé číslo značí, kolik takových záznamů se v dané podsece vyskytuje. Pod tímto řádkem se nachází samotné záznamy o nepřímých objektech. Záznam o využívaném nepřímém objektu má tvar *nnnnnnnnnn gggg n* a je zakončen koncem řádku. Desetimístné číslo *nnnnnnnnnn* označuje offset bajtů od začátku dokumentu po začátek nepřímého objektu. Pětimístné číslo *gggg* je číslo generace. Jedna možná sekce křížových odkazů je uvedena ve výpisu 3.3:

---

```
xref
6 2
0000057002 00000 n
0000000265 00000 n
10 1
0000058406 00002 n
```

---

Výpis 3.3: Výpis obsahuje jednu sekci křížových odkazů. Tato sekce je rozdělena na 2 podseky a celkem obsahuje 3 záznamy

- **Patička** – Patička umožňuje rychlé nalezení tabulky křížových odkazů a jiných speciálních objektů. Proto obecně platí, že by se měl PDF soubor číst od konce, kde se vykytuje právě patička. Patička začíná klíčovým slovem **trailer**, za ním následuje slovník patičky a klíčové slovo **startxref**. Na novém řádku se poté vykytuje číslo, uvádějící počet bajtů offsetu od začátku souboru po začátek tabulky křížových odkazů. Jako poslední se na samostatném řádku musí objevit výraz %EOF. V celém



souboru se může vyskytovat více patiček, to je způsobeno aktualizováním daného PDF dokumentu. Na posledním řádku souboru se vždy musí vyskytovat výraz `%%EOF`. Možná patička je vypsána ve výpisu 3.4:

```
trailer
<<
/Size 12
/Root 3 0 R
/Info 1 0 R
>>
startxref
565
%%EOF
```

Výpis 3.4: V tomto výpisu je uvedena možná patička PDF souboru. Z této patičky lze vyčíst, že tabulka křížových odkazů má celkem 12 záznamů a její poslední sekce začíná na 565. bajtu. Dále lze zjistit, že nepřímý objekt s identifikátorem 3 0 je dokumentový katalog a metadata dokumentu jsou obsaženy ve slovníkovém nepřímém objektu s identifikátorem 1 0

Tyto 4 části jsou v PDF souboru seřazeny v takovém pořadí, v jakém jsou zde sepsané. Tyto části jsou vyznačeny v ukázkovém souboru, který je ukázán na obrázku 3.1.

%PDF-1.7 %µµµµ	Hlavička
1 0 obj << /Type /Catalog /Pages 2 0 R /Lang (cs-CZ) /Metadata 22 0 R ... >> endobj 2 0 obj << /Type /Pages /Count 1 /Kids [ 3 0 R ] >> endobj ...	Tělo
xref 0 24 0000000000 65535 f 0000000017 00000 n 0000000365 00000 n ... 0000029530 00000 n	Tabulka křížových odkazů
trailer << /Size 24 /Root 1 0 R /Info 9 0 R /ID [ <4F704CDC2ADDC64E8798F1A20FFECB02> <4F704CDC2ADDC64E8798F1A20FFECB02> ] >> startxref 29575 %%EOF	Patička

Obrázek 3.1: Na obrázku se vyskytuje ukázkový PDF soubor, ve kterém jsou vyznačeny 4 části struktury PDF souborů

## Struktura dokumentu

Struktura dokumentu popisuje, jak vypadá vyobrazený PDF dokument. Na rozdíl od struktury souboru, která popisuje vnitřní rozložení, si ji lze představit jako hierarchii objektů vy-

skytujících se v těle PDF souboru. Některé z důležitých částí tohoto hierarchického stromu jsou:

- **Katalog dokumentu** – Tento katalog je kořenem celého hierarchického stromu. Odkaz na něj lze nalézt ve slovníku patičky pod klíčem **Root**. Tento katalog obsahuje odkazy na objekty specifikující vzhled dokumentu. Objekt katalogu je slovník, ve kterém se musí vyskytovat klíč **Type**, ke kterému je přiřazena hodnota **/Catalog**. Dalším povinným prvkem katalogového slovníku je klíč **Pages**, jehož hodnota je nepřímý odkaz na kořenový objekt stromu stránek. Příklad objektu katalogu dokumentu je uveden ve výpisu 3.5:

---

```
3 0 obj
<<
  /Type /Catalog
  /Pages 5 0 R
>>
endobj
```

---

Výpis 3.5: Tento výpis obsahuje příklad katalogu dokumentu, ze kterého lze zjistit, že kořen stromu stránek je nepřímý objekt s identifikátorem 5 0

- **Strom stránek** – Strom stránek určuje pořadí zobrazení stránek. Listové uzly tohoto stromu jsou typu *objektu stránky*, které mají jiný tvar než ostatní uzly tohoto stromu. Uzly stromu stránek jsou typu slovníku, ve kterém se musí vyskytovat klíče **Type**, **Kids**, **Count** a **Parent**, jenž není povinný v kořenovém uzlu. Hodnota klíče **Type** musí v uzlu stromu stránek být **/Pages**. Ke klíči **Kids** musí být přiřazena hodnota typu pole, které obsahuje nepřímé odkazy na uzly stromu stránek, nebo na objekty stránek. Hodnota klíče **Count** je počet listových uzlů, které jsou potomkem tohoto uzlu. Ke klíči **Parent** je přiřazena hodnota nepřímého odkazu přímého předchůdce tohoto uzlu. Platný uzel stromu je uveden ve výpisu 3.6:

---

```
5 0 obj
<<
  /Type /Pages
  /Kids [21 0 R 24 0 R]
  /Count 10
>>
endobj
```

---

Výpis 3.6: Výpis obsahuje jeden uzel stromu stránek. Potomci tohoto uzlu jsou nepřímé objekty s identifikátorem 21 0 a 24 0 a celkově existuje 10 listových uzlů (objektů stránek), které jsou potomky tohoto uzlu

- **Objekt stránky** – Objekt stránky je typ listového uzlu stromu stránek. Tento uzel má tvar slovníku, ve kterém se musí vyskytovat klíč **Type** s hodnotou **/Page**. Dalším povinným záznamem tohoto slovníku je klíč **Parent**, jehož hodnota je nepřímý odkaz na přímého předchůdce tohoto listového uzlu stromu stránek. Mezi jiné důležité záznamy patří záznamy s klíčem **MediaBox**, **Resources**, **Rotate**, **Contents**, **Annots** aj. Možný objekt stránky je obsažen ve výpisu 3.7:

---

```
7 0 obj
<<
/Type /Page
/MediaBox [ 0 0 595.276 841.89 ]
/Parent 21 0 R
/Contents 10 0 R
/Resources 9 0 R
>>
endobj
```

---

Výpis 3.7: Tento výpis obsahuje jednoduchý objekt stránky, jejíž content stream je obsažen v nepřímém objektu s identifikátorem 10 0 a resources použity v tomto content streamu jsou vypsány v nepřímém objektu s identifikátorem 9 0

## Content streams

Content stream obsahuje popis vzhledu PDF stránky pomocí instrukcí na její vykreslení. Tyto instrukce jsou zapsány pomocí PDF objektů, ale na rozdíl od PDF dokumentu, jsou tyto instrukce seřazené a vykonávají se podle jejich posloupnosti. *Operand* takové instrukce musí být přímý objekt, který nesmí být typu datového toku. Operand může být typ slovník pouze při použití speciálních operací. *Operátor* instrukce určuje, která akce se provede. Operátory jsou klíčová slova typu jmenného objektu, kde se na rozdíl od jmenových objektů v těle PDF dokumentu operátory píšou bez počátečního lomítka. Content stream používá pro zapsání instrukcí postfixovou notaci, tedy nejdříve jsou uvedeny všechny operandy instrukce a poté je uveden její operátor.

## Resources

Resources specifikují a pojmenovávají používané externí objekty z content streams. V content streams se nesmí používat nepřímé odkazy, proto je možné pojmenovat jednotlivé používané objekty a definovat je tak jako *named resources*. Tyto jména lze používat pouze uvnitř content streams a mimo ně nejsou v těle PDF dokumentu validní. Named resources se používají například pro obrázky a fonty, které jsou použity na dané stránce. Objekt pro resources je typu slovníku, který má několik definovaných klíčů, které je možné použít, např. *ColorSpace*, *XObject*, *Font*, *ProcSet* a další. Příklad slovníku pro resources je uveden ve výpisu 3.8:

---

```
<<
/Font <<
  /F5 2 0 R
>>
/XObject <<
  /Im1 29 0 R
  /Im2 32 0 R
>>
>>
```

---

Výpis 3.8: V tomto výpisu je uveden slovník pro resources, který obsahuje jeden font pod jménem F5 a dva externí objekty pojmenované jako Im1 a Im2

## 3.2 Grafika v PDF

Tato sekce čerpá informace ze standardu PDF 32000-1 [13, k. 8, s. 110–236]. Grafika PDF dokumentu je popsána pomocí content streams, jenž je vysvětleno v kapitole 3.1. Operátory zde používané spadají do šesti hlavních skupin:

- **Graphics state operátory** – Tyto operátory manipulují s datovou strukturou zvanou *graphics state*, která je vysvětlena později v této kapitole.
- **Operátory pro konstrukci křivek** – Jsou to operátory, které specifikují, jak bude vypadat vykreslená křivka. Spadají zde například operátory pro vytvoření nové cesty, přidávání zaoblení, přidávání nové části křivky a uzavření tvaru.
- **Operátory pro vymalování křivek** – Jsou to operátory vybarvující křivku nebo prostor vyhrazený takovou křivkou.
- **Další vykreslovací operátory** – Tyto operátory se používají pro vykreslení grafických objektů, mezi které patří i obrázky.
- **Textové operátory** – Text se v PDF dokumentu vykreslí jako několik grafických částí, které jsou definovány fontem. Pomocí těchto operátorů se specifikují nastavení spojené s textem a též zde patří operátory pro vykreslení takového textu.
- **Marked-content operátory** – Tyto operátory přímo neovlivňují vzhled stránky, ale spojují informace a objekty uvedené v content streamu.

### Souřadnicové systémy

PDF definuje více souřadnicových systémů, ve kterých se definují různé grafické objekty. Převod mezi těmito souřadnicovými systémy se provádí s pomocí transformačních matic, které dokážou otáčet, posunovat nebo měnit měřítko mapovaného objektu (nemění se přitom samotné data uloženého objektu, jen jak je zobrazen uvnitř souřadnicového systému). V PDF se transformační matice značí polem obsahující šest číselných objektů  $[a \ b \ c \ d \ e \ f]$ . Tato transformační matice je vyobrazena v rovnici (3.1).

$$M_T = \begin{pmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{pmatrix} \quad (3.1)$$

Pokud se potřebuje provést více elementárních transformací (např. rotace a posun), záleží na jejich posloupnosti provedení. Obecně platí vzorec (3.2), kde  $M_{T_1}$  značí matici první provedené transformace,  $M_{T_n}$  je matice poslední provedené transformace a  $M'$  označuje matici, která kombinuje všechny tyto provedené transformace.

$$M' = M_{T_1} \cdot M_{T_2} \cdots M_{T_{n-1}} \cdot M_{T_n} \quad (3.2)$$

Bod  $(x, y)$  v souřadnicovém systému se může matematicky popsat jako vektor z rovnice (3.3).

$$\vec{p} = (x \ y \ 1) \quad (3.3)$$

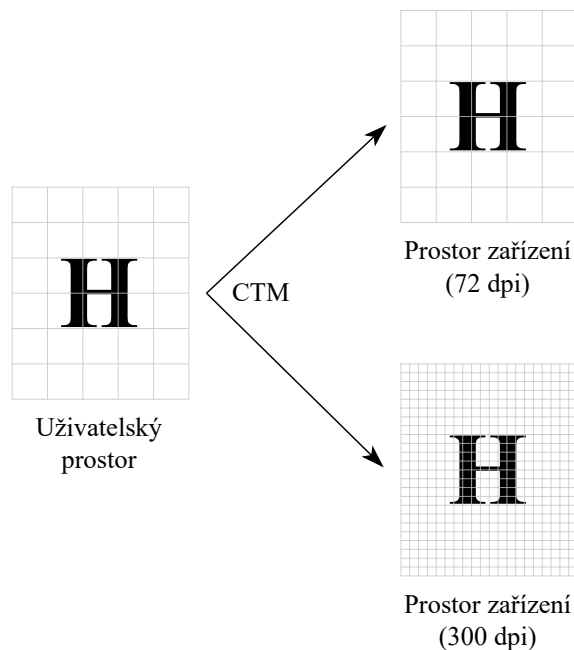
Transformovaný bod (souřadnice bodu po použití všech transformací) se vypočítá pomocí vzorce (3.4).  $\vec{p}$  v tomto vzorci označuje původní vektor transformovaného bodu,  $\vec{p'}$  je vektor

tohoto bodu po transformaci a  $M'$  je matice kombinující všechny provedené transformace.

$$\vec{p'} = \vec{p} \cdot M' \quad (3.4)$$

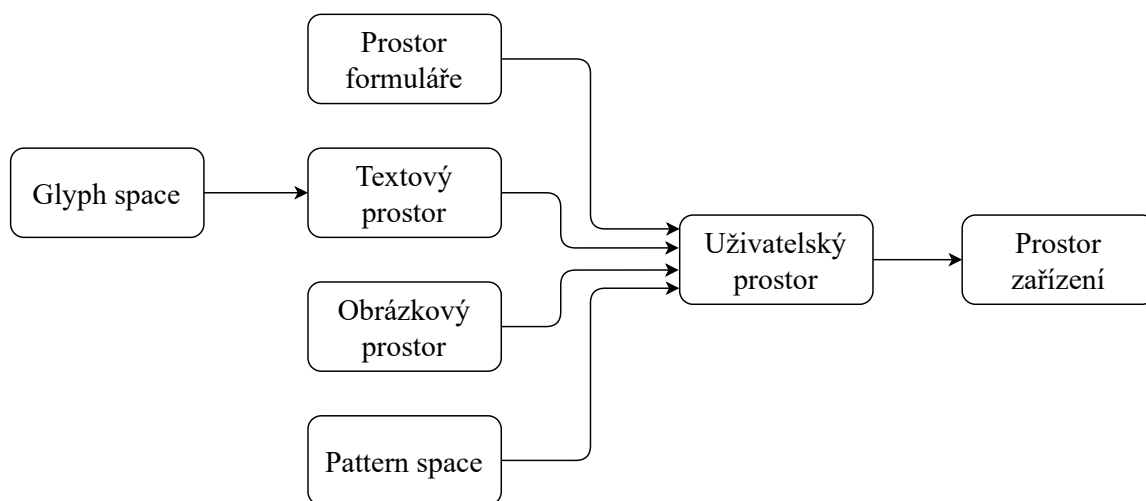
Dále jsou popsány souřadnicové systémy používané v PDF. Vztahy mezi těmito souřadnicovými systémy jsou vyznačeny na obrázku 3.3.

- **Prostor zařízení (device space)** – Tento souřadnicový systém je závislý na zařízení, kde se zobrazuje vytvořený PDF dokument. Je to poslední prostor, do kterého se promítá zobrazení PDF souboru. Zobrazovacím zařízením se rozumí například displej obrazovky počítače, papír v tiskárně, plátno, na které promítá projektor aj.
- **Uživatelský prostor (user space)** – Aby se zamezilo nevhodným účinkům při zobrazování do prostoru zařízení, definuje PDF vlastní prostor, který je nezávislý na zařízení. Tento souřadnicový systém se nazývá *uživatelský prostor*. Pro každou stránku PDF dokumentu se tento souřadnicový systém uvede do původního stavu. V uživatelském prostoru je bod (0,0) levý dolní roh, tedy x-ová souřadnice se zvyšuje směrem doprava a y-ová souřadnice roste směrem nahoru. Rozlišení určeno v jednotkách uživatelského prostoru nesouvisí s rozlišením v pixelech uvnitř prostoru zařízení. Převod z uživatelského prostoru do prostoru zařízení se provádí pomocí *current transformation matrix (CTM)*. CTM je součástí datové struktury *graphics state*, která je popsána dále v této kapitole. Aplikace zobrazující PDF si tuto CTM matici upraví podle vlastností výstupního zařízení, aby se zachovala nezávislost uživatelského prostoru na zařízení. Vyobrazení tohoto jevu jde vidět na obrázku 3.2.



Obrázek 3.2: V tomto obrázku je ukázána transformace z uživatelského prostoru do dvou různých prostorů zařízení, kde jeden má rozlišení 72 DPI a druhý má 300 DPI. CTM představuje current transformation matrix získanou z datové struktury graphics state. Obrázek je převzatý a upravený ze standardu PDF 32000-1 [13, k. 8.3.2.3, s. 116]

- **Specializované prostory** – PDF formát pracuje kromě uživatelského prostoru a prostoru zařízení i se specializovanými prostory.
  - *Textový prostor* – V tomto prostoru se definují souřadnice textu. Převod z textového prostoru do uživatelského prostoru se provádí pomocí *textové matice* a několika textových parametrů vyskytujících se v datové struktuře graphics state.
  - *Glyph space* – Glyfy<sup>1</sup> znaků fontu musí být definovány v tomto prostoru. Z glyph space se poté transformuje do textového prostoru.
  - *Obrázkový prostor* – Všechny obrázky by měly být definovány v tomto prostoru. Pro správné vykreslení obrázku na stránku se musí dočasně upravit CTM.
  - *Prostor formuláře* – Formulářový *XObject* (též externí objekt, viz dále v této kapitole) je reprezentován jako content stream. Tento XObject je v jiném content streamu brán jako grafický objekt. Prostor, ve kterém je tento formulář definovaný, se nazývá prostor formuláře.
  - *Pattern space* – PDF formát poznává typ barvy zvaný *pattern*. Pattern je definovaný jako content stream a prostoru, ve kterém je definován, se říká pattern space.
  - *3D prostor* – Tento souřadnicový systém je trojdimenzionální a používá se pro vložené 3D díla.



Obrázek 3.3: Obrázek označuje vztah mezi souřadnicovými systémy používaných uvnitř PDF. Každá šipka představuje transformaci mezi dvěma souřadnicovými systémy. Obrázek je převzatý a upravený ze standardu PDF 32000-1 [13, k. 8.3.3, s. 117]

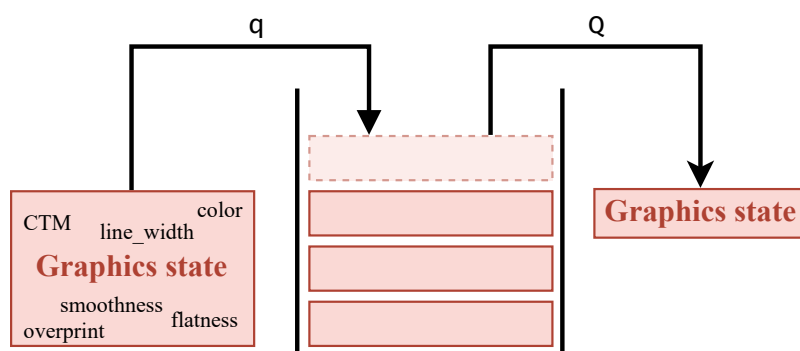
## Graphics state

Pro zobrazení PDF dokumentu se používá datová struktura zvaná *graphics state*. Každá aplikace zobrazující PDF dokumenty musí umět udržovat tuto datovou strukturu. Graphics state struktura v sobě obsahuje několik parametrů, se kterými pracují operace uvnitř

<sup>1</sup>Glyf je grafické znázornění grafému (písmena, číslice, znaku, piktogramu, interpunkčního a jiných znamének).

content streamu (popsaný v kapitole 3.1). Pro každou stránku se na začátku musí v graphics state struktuře nastavit výchozí hodnoty obsahujících parametrů. Tyto parametry jsou rozděleny na dvě skupiny: závislé na zařízení a nezávislé na zařízení. Mezi parametry nezávislých na zařízení patří například CTM (current transformation matrix), barva (aktuální barva používaná při vykreslovacích operacích), stav textu (devět parametrů popisující formát vypisovaného textu) a tloušťka čáry. Parametry závislé na zařízení jsou například „overprint“, „flatness“ a „smoothness“.

Na jedné stránce se může vyskytovat několik grafických objektů, které se vykreslují nezávisle na sobě. Pro tyto účely se používá *graphics state zásobník*, díky kterému je možné dělat lokální úpravy datové struktury graphics state. Tento zásobník je LIFO (last in, first out) a ukládá se do něj celá datová struktura graphics state, jak je ukázáno na obrázku 3.4. Pro uložení se musí použít operátor `q` a pro odebrání první položky z vrcholu zásobníku se musí použít operátor `Q`, kterým se obnoví posledně uložený stav datové struktury graphics state. Uvnitř celého content streamu musí být použit stejný počet operátorů `q` a `Q`.



Obrázek 3.4: Na tomto obrázku je ukázán graphics state zásobník, do kterého se ukládá pomocí příkazu `q` a odebírá se z něj příkazem `Q`

Pro úpravu parametrů uložených ve struktuře graphics state se používají uvnitř content streamu specifické operátory. Mezi tyto operátory patří již uvedené `q` a `Q`, které nepoužívají žádné operandy. Další používané operátory upravující graphics state jsou například `cm`, `w`, `i` a jiné. K operátoru `cm` se váže šest číselných operandů *a*, *b*, *c*, *d*, *e* a *f*. Dohromady tyto operandy specifikují matici transformace, která bude vynásobena maticí CTM a následně přiřazena do CTM položky datové struktury graphics state. Operátor `w` je unární a jeho operandem je číslo *lineWidth*, které specifikuje tloušťku kreslených čar. Unární operátor `i` nastaví uvedený číselný operand jako *flatness* parametr struktury graphics state.

## Externí objekty

Externí objekt (též taky *XObject*) je grafický objekt, jehož vzhled je definován v samostatném datovém toku. Každý XObject lze vykreslit pomocí operátoru `Do` vyskytujícího se uvnitř content streamu (více informací o content streamu je uvedeno v sekci 3.1). K tomuto operátoru se váže jeden operand typu jméno. Toto jméno musí být uvedeno ve slovníku *resources* (viz kapitola 3.1) vázaného na stejnou stránku jako content stream. Přesněji se toto jméno musí objevit v položce pod klíčem `XObject`, která má hodnotu typu slovník. Operátor `Do` má 3 různé chování. Toto chování závisí na hodnotě vázané ke klíči `Subtype` uvnitř XObject slovníku, na který se odkazuje operand tohoto příkazu. Hodnoty klíče `Subtype` a k nim vázané chování příkazu `Do` jsou:

- */Image* – Tyto externí objekty mají ve slovníkové části své definice dodatečné prvky. Některé tyto prvky můžou ovlivnit, jak bude vykreslený obrázek vypadat, ale podle aktuálního nastavení stránky můžou mít omezené možnosti. Prvky s klíčem **Width** a **Height** jsou v těchto objektech povinné a vyjadřují výšku a šířku původního obrázku (uvedeného ve streamu tohoto externího objektu). Pro vykreslení tohoto objektu na stránce se používá operátor **Do**, který pomocí parametrů uvnitř datové struktury **graphics state** (vysvětlena dříve v této sekci) správně upraví a vykreslí tento objekt.
- */Form* – Takzvaný *Form XObject* je takový, který ve svém datovém toku obsahuje content stream (popsán v sekci 3.1). *Form XObject* může být vykreslen několikrát, a to na různých souřadnicích. Ve slovníkové části tohoto objektu musí být pro tento typ uvedena hodnota pro klíč **BBox**. Tato hodnota jsou souřadnice daného externího objektu v prostoru formuláře. Pro transformaci mezi formulářovým prostorem a user space se uvádí použitá transformační matice jako hodnota pro klíč **Matrix**. Pokud tato matice není uvedena, použije se místo ní jednotková matice. Pro vykreslení se tohoto objektu se musí použít operátor **Do** a při tom se provedou následující kroky (převzato ze standardu PDF [13, k. 8.10.1, s. 218]):
  1. Uloží se aktuální stav datové struktury **graphics state**, stejně jako kdyby byl použit operátor **q**.
  2. Vynásobí se matice ze slovníkového prvku **Matrix** a matice **CTM** z datové struktury **graphics state**.
  3. Ostříhne se na základě slovníkového prvku **BBox**.
  4. Vykreslí obsahované grafické objekty podle instrukcí uvnitř content streamu definovaného objektem *form XObject*.
  5. Obnoví původní stav struktury **graphics state** (jako použití operátoru **Q**).
- */PS* – Takzvaný *PostScript XObject* je využíván pouze, pokud bude daný PDF dokument zpracován zařízením používající jazyk **PostScript**. Při zobrazení v jiných zařízeních bude tento objekt ignorován. Aplikace zpracovávající PDF dokumenty nemají nutnost zpracování těchto objektů podporovat.

### 3.3 Reprezentace anotací v PDF souboru

Anotace spojují objekty (zvuk, video, text, ...) s pozicí na PDF stránce. Uživatel může s anotacemi interagovat pomocí myši a klávesnice. Pokud se na stránce vyskytují nějaké anotace, všechny tyto anotace jsou vypsány v objektu dané stránky (více viz sekce 3.1). Přesněji se v tomto objektu vyskytuje prvek s klíčem **Annots**, který má jako hodnotu pole, které obsahuje nepřímé odkazy na objekty anotací vyskytovaných na dané stránce. PDF dokumenty podporují několik typů anotací, například:

- **Text** – Tyto anotace imitují nalepovací papírky. Jsou zobrazeny jako ikona umístěna na určitých souřadnicích a po interakci s myší se zobrazí vyskakovací okénko s textem této anotace. Tato anotace ignoruje rotaci a změnu měřítka stránky. Ve slovníku této anotace se můžou navíc objevit záznamy s klíčem **Subtype** (tento prvek je povinný a pro tento typ anotace musí mít hodnotu **/Text**), **Open**, **Name**, **State** a **StateModel**. Příklad textové anotace lze vidět ve výpisu 3.9:



---

```

21 0 obj
<<
/Type /Annot
/Subtype /Text
/Name /Note
/Rect [32.789 127.064 70.23 175.9]
/Contents (This is text annotation.)
>>
endobj

```

---

Výpis 3.9: Anotace typu *Text*, která se zobrazí jako ikona *Note*. Po interakci s myší se zobrazí vyskakovací okénko s textem „This is text annotation.“

- **Odkaz** – Anotace odkazu slouží jako hypertextový odkaz na místo v daném dokumentu, nebo jako akce, která se má provést po jejím stisknutí. Ve slovníku takové anotace musí existovat povinný záznam s klíčem **Subtype**, který musí mít hodnotu **/Link**. Další možné záznamy mohou být **A**, **Dest**, **H**, **PA**, **QuadPoints** a **BS**. Validní anotace odkazu je uvedena ve výpisu 3.10:

---

```

46 0 obj
<<
/Type /Annot
/Subtype /Link
/Rect [88.906 52.007 125.4 65.853]
/Dest [7 0 R /XYZ 0 186.33 0]
>>
endobj

```

---

Výpis 3.10: Anotace typu *Link*, která odkazuje na místo uvnitř stejného dokumentu.

- **Volný text** – Tato anotace bude na stránce zobrazena jako viditelný text. Ve slovníku její definice musí být uveden záznam **Subtype** s hodnotou **/FreeText** a záznam **DA**, kterým se definuje grafická úprava vypsání textu. Slovník může obsahovat další záznamy, například **Q**, **IT** nebo **BS**.
- **Přímka** – Tato anotace se na stránce zobrazí jako jedna definovaná přímka. Po její interakci s myší se může zobrazit vyskakovací okénko. **Subtype** záznam ve slovníku, který definuje tuto anotaci, musí mít hodnotu **/Line**. Další povinný záznam je **L**, jehož hodnota specifikuje počáteční a konečný bod dané přímky. V tomto slovníku je možné uvést ještě několik dalších záznamů, například **BS**, **IC**, **Cap** a **LE**.
- **Označení textu** – Označení textu anotacemi se může zobrazit jako jeho zvýraznění, podtržení, vlnité podtržení, nebo jeho přeškrtnutí. Pro zvolení typu označení se musí ve slovníku této anotace uvést příslušná hodnota povinného záznamu **Subtype**, a to **/Highlight**, **/Underline**, **/Squiggly**, nebo **/StrikeOut**. Další povinný záznam této anotace je **QuadPoints**, jehož hodnota specifikuje souřadnice čtyřúhelníku, ve kterém je obsažen označovaný text. Této anotaci může být přiřazena vyskakovací anotace.
- **Vyskakovací anotace** – Toto vyskakovací okénko je vždy spojené s jinou anotací (takzvanou *rodičovskou anotací*). Samotná nemá definovaný žádný datový tok popi-

sující vhléd, ani k sobě nemá přiřazenou žádnou akci. Bývá uvedena jako nepřímý odkaz ve slovníku své rodičovské anotace (přesněji v záznamu *Popup*).

### 3.4 Programovací jazyky a knihovny pro zpracování a anotování PDF souborů

Pro zpracovávání PDF souborů existuje mnoho knihoven v různých programovacích jazycích. Výběr programovacího jazyka záleží nejen na požadavcích pro výslednou aplikaci, ale též na znalostech daného programátora. Samotná knihovna se poté vybere na základě její funkcionality.

V této kapitole jsou popsány různé knihovny, které je možné použít pro zpracování PDF souborů, jejich speciality a nedostatky.

#### C#

Dokumentace jazyka C# [20] říká, že C# je objektově orientovaný programovací jazyk, vyvinutý firmou Microsoft. Jazyk C# je potomkem rodiny jazyků C. Je jim tedy podobný a programátorům těchto jazyků nebude dlouho trvat se jej naučit. Jazyk C# je jeden z nejpoužívanějších jazyků pro vývoj na platformě .NET.

Nejznámější C# knihovna pro práci s PDF dokumenty je **iText 7** (informace byly získány z její oficiální dokumentace [7]). Tato knihovna je dostupná pod *Open Source AGPLv3*<sup>2</sup> licenci a dvěma verzemi komerční licence. Tato knihovna umí vytvářet nové i upravovat již existující PDF dokumenty. Pomocí této knihovny lze například přidávat anotace, hlavičky a patičky stránek, spojovat několik dokumentů do jednoho, upravovat metadata, vyplňovat či upravovat formuláře a extrahovat text či jiné PDF objekty.

#### JavaScript

Podle webové stránky věnované jazyku JavaScript [10], je JavaScript dynamicky typovaný, objektově orientovaný, interpretovaný programovací jazyk. Nejčastěji se využívá jako skriptovací jazyk používaný pro vytváření webových stránek, je však často používán i mimo prostředí webového prohlížeče. Nejznámější z těchto případů je například Node.js, Apache CouchDB a Adobe Acrobat.

Pro zpracování PDF souborů v jazyce JavaScript je možné použít některou z následujících knihoven:

- **PDF.js** – Informace o této knihovně byly získány z oficiální dokumentace PDF.js knihovny [11]. Tato knihovna byla vyvinuta převážně pro čtení a vykreslování PDF souborů, samotná neumí dané soubory editovat. Jiné knihovny jsou s touto knihovnou často kombinovány pro pokročilejší práci s PDF soubory. Práce s PDF.js knihovnou závisí na využívání takzvaných Promises, bez kterých nelze tuto knihovnu používat, proto je doporučeno se s jejich používáním dobře seznámit před jakoukoliv prací s touto knihovnou. PDF.js je dostupné pod licencí *Apache License 2.0*<sup>3</sup>.
- **pdfAnnotate** – Informace o této knihovně byly získány z GitHub stránek pdfAnnotate knihovny [12]. Knihovna pdfAnnotate je vyvinutá specificky pro anotování PDF

<sup>2</sup><https://itextpdf.com/how-buy/AGPLv3-license>

<sup>3</sup><https://github.com/mozilla/pdf.js/blob/master/LICENSE>

souborů dostupná pod licencí *MIT License*<sup>4</sup>. Funguje pouze v prostředí webového prohlížeče a Node.js. Samotná knihovna neumí číst a zobrazovat PDF soubory, proto je doporučováno tuto knihovnu kombinovat s výše zmíněnou knihovnou PDF.js. Přidané anotace se zapisují na konec PDF souboru, díky tomu je možné je zobrazit i mimo vytvořenou aplikaci.

- **PDF-LIB** – Informace o této knihovně byly získány z oficiálních stránek PDF-LIB knihovny [5]. Tuto knihovnu je možné používat v jakémkoliv JavaScriptovém prostředí. PDF-LIB dokáže vytvářet nové či modifikovat existující PDF soubory. Mezi modifikace patří například vytváření a vyplňování formulářů, vkládání PDF stránek a čtení a přepisování metadat souboru. Vytváření anotací je možné, ale vyžaduje pokročilejší znalost zápisu formátu PDF. Tato knihovna je dostupná pod licencí *MIT License*<sup>5</sup>.

## PHP

Informace uvedené na oficiálních PHP stránkách [14, 15] uvádí, že PHP je skriptovací programovací jazyk, který je především vhodný pro vývoj dynamických webových stránek. Často je PHP kód vnořený přímo do HTML kódu, kterému tak přidává dynamičnost. PHP podporuje objektově orientované i procedurální programování. I když je PHP jazyk používán převážně pro vývoj webu, lze jej využít i pro vytvoření aplikace běžící v příkazové řádce a pro vývoj desktopových aplikací.

- **TCPDF** – Informace o této knihovně byly získány z oficiálních stránek TCPDF knihovny [3]. Tato knihovna je zaměřena na vytváření PDF dokumentů, mezi jejíž hlavní vlastnosti patří podpora fontů, automatické hlavičky a patičky na stránce, dělení slov, zarovnání a zalamování textu, PDF anotace a automatické číslování stran. TCPDF nepodporuje čtení a editaci existujících PDF souborů. Knihovna TCPDF je dostupná pod *Free Software License*<sup>6</sup> licencí.
- **FPDF** – Informace o této knihovně byly získány ze stránek FPDF knihovny [6]. Toto je knihovna pro používání stránek z existujícího PDF dokumentu jako šablony do nového PDF dokumentu. Při opakovaném použití jedné šablony tato knihovna zajistí, že tato šablona bude v souboru PDF zahrnuta právě jednou. Díky této skutečnosti může být výsledné PDF menší velikosti než PDF vytvořeno jiným způsobem. Tato třída je kompatibilní s výše zmíněnou knihovnou TCPDF. Od verze 1.6 je knihovna FPDF dostupná pod licencí *MIT license*<sup>7</sup>.

## Python

Oficiální dokumentace jazyka Python [19] uvádí, že Python je vysokoúrovňový, interpretovaný programovací jazyk. Má dynamickou kontrolu datových typů a podporuje objektově orientované programování. Tyto skutečnosti z něj dělají ideální jazyk pro skriptování a rychlé prototypování různých aplikací na mnoho platformách. Python je programovací jazyk vhodný i pro začátečníky.

---

<sup>4</sup><https://github.com/highkite/pdfAnnotate/blob/master/LICENSE>

<sup>5</sup><https://github.com/Hopding/pdf-lib/blob/master/LICENSE.md>

<sup>6</sup><https://tcpdf.org/docs/license/>

<sup>7</sup><https://www.tldrlegal.com/l/mit>

V tomto jazyce existuje několik knihoven pro práci s PDF soubory. Je možné použít například následující:

- **PyMuPDF** – Informace o této knihovně byly získány z oficiální dokumentace PyMuPDF knihovny [2]. Tato knihovna je Python verze knihovny MuPDF. Je dostupná pod dvěma různými licencemi, a to pod licencí *Open Source – AGPL*<sup>8</sup> a komerční<sup>9</sup> licencí. Knihovna se může lišit podle verze s danou licencí. Tato knihovna dokáže například číst a vyjmout text i obrázky, číst a upravovat metadata a vyhledávat text v existujícím dokumentu. Knihovna má podporu pro OCR (Optical Character Recognition), pokud při instalaci byl nainstalován též balíček Tesseract. Podporované formáty dokumentu jsou PDF, XPS, OpenXPS, CBZ, EPUB a FB2 (eBooks). PyMuPDF umí zacházet i s populárními formáty obrázků jako jsou PNG, JPEG, BMP, TIFF a dalšími. Knihovna má pro PDF dokumenty několik funkcí navíc, což zahrnuje například vytváření nového dokumentu, extrahování a přidávání fontů a obrázků, pracování se soubory chráněnými heslem (např. zašifrování a dešifrování), zcela podporuje práci s vloženými soubory, plně podporuje práci s anotacemi a dokáže vytvářet, číst i vyplňovat formuláře.
- **pikepdf** – Informace o této knihovně byly získány z oficiální pikepdf dokumentace [4]. Tuto knihovnu lze používat pro vytváření, čtení i úpravu PDF dokumentů. Je to Python verze C++ knihovny QPDF. Pro používání knihovny je nutné být seznámen se specifikací PDF formátu. Knihovna je dostupná pod *Mozilla Public License 2.0*<sup>10</sup> licencí. Pikepdf dokáže kopírovat stránky do jiného PDF souboru, extrahovat obrázky, nahradit obrázek za jiný, upravovat metadata souboru a další.

---

<sup>8</sup><https://artifex.com/licensing/agpl/>

<sup>9</sup><https://artifex.com/licensing/commercial/>

<sup>10</sup><https://github.com/pikepdf/pikepdf/blob/master/LICENSE.txt>

## Kapitola 4

# Návrh a implementace aplikace Theses Checker

Součástí této práce byly vytvořeny 2 aplikace, jedna je webový nástroj a druhá je program pro použití z příkazového řádku. Tato kapitola pojednává o požadavcích na vytvářené aplikace a dále popisuje detaily všech vytvořených programů.

### 4.1 Specifikace požadavků

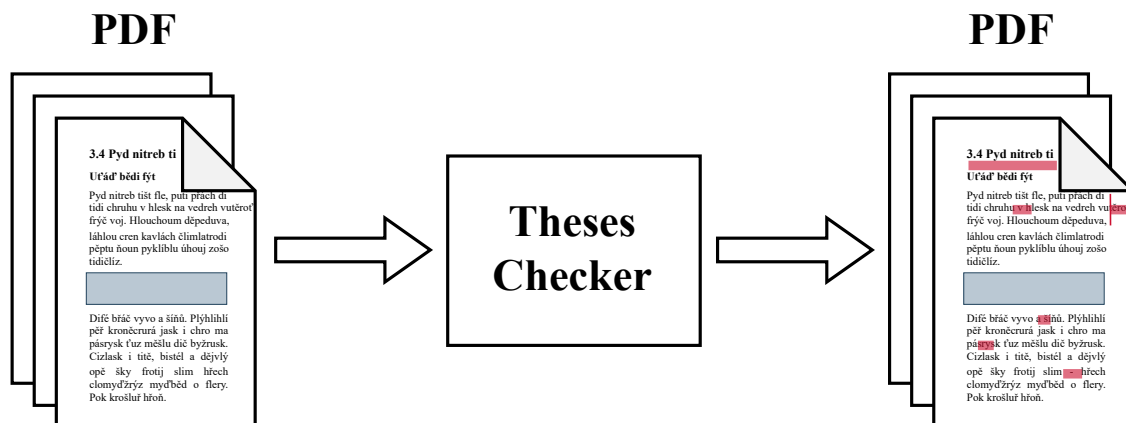
Cílem vytvořené aplikace je nalézt a vyznačit chyby, které se vyskytují v nahrané technické zprávě. Technická zpráva bude ve formátu PDF. Požadavky pro vytvořený program jsou:

- Program musí podporovat technické zprávy napsané v jazyce českém a s menší podporou i v jazyce slovenském nebo anglickém.
- Aplikace musí primárně umět kontrolovat diplomové práce vytvořené v šabloně pro jazyk  $\text{\LaTeX}$ , která byla poskytnuta univerzitou Vysoké učení technické v Brně, přesněji Fakultou informačních technologií.
- Program musí být lehce přístupný, nejlépe bez nutnosti jakékoliv instalace.
- Aplikace by měla být funkční na co nejvíce platformách, aby byla zajištěna dostupnost pro co nejvíce uživatelů.
- Aplikace by měla umět pracovat s co nejaktuálnější verzí PDF.
- Není nutné, aby program dokázal zpracovat několik PDF dokumentů zároveň. Postačí, aby v jednom příkazu zpracoval právě jeden dokument.

Nalezené chyby budou označeny graficky pomocí PDF anotací, jako například zvýraznění textu a kreslení čar. Toto označení by mělo být zřetelné – dobře viditelné a na první pohled pochopitelné. Aplikace by tím měla imitovat poznámky korektora. Detekované chyby jsou takové, které se často vyskytují v technických pracích. Tyto časté chyby byly zjištěny zkoumáním rozpracovaných i odevzdaných diplomových prací a projevilo se, že vyskytované časté chyby jsou porušení některého z typografických pravidel. Tyto nalezené časté chyby jsou uvedeny v kapitole 2. Aktuálně neexistuje žádná aplikace, která by aktivně kontrolovala typografické chyby. Většina existujících aplikací hledá pouze porušení gramatických pravidel. Hledání chyb bude pracovat na principu zvaném *false-positive*. To znamená, že něco může být označeno jako chyba, ale ve skutečnosti se o chybu nejedná.

## 4.2 Návrh aplikace

Aby byla aplikace co nejvíce nezávislá, byla vyvinuta jako webová aplikace. Tato aplikace musí být intuitivní a až na nahrání PDF dokumentu by se nemělo muset vyplňovat zbytečně mnoho informací. Většina informací by měla být zjištěna automaticky z poskytnutého PDF souboru. Jak je ukázáno na obrázku 4.1, vstupem této aplikace by měl být samotný, neupravený PDF dokument, který vytvořená aplikace zpracuje a jejím výstupem je tento PDF dokument doplněný o anotace.

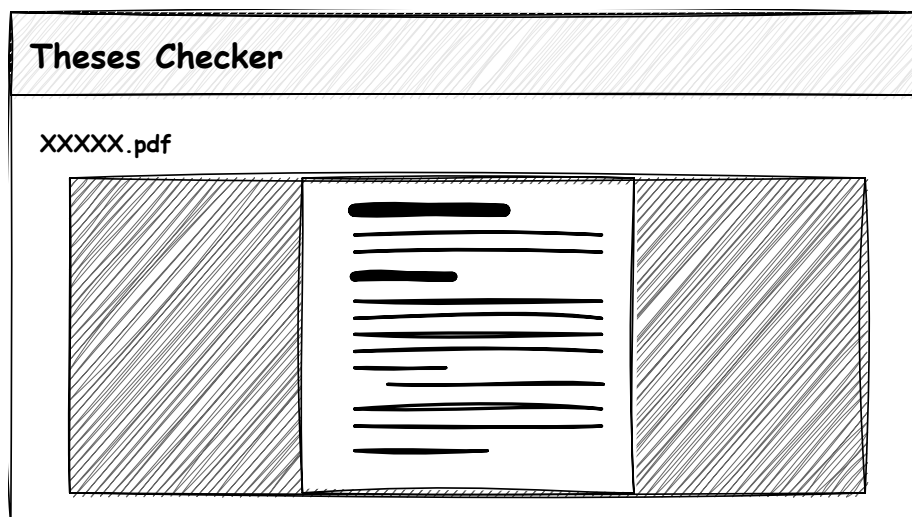


Obrázek 4.1: Na tomto obrázku jsou ukázány vstupy a výstupy aplikace Theses Checker

Vytvořená webová aplikace by měla odrážet tento uvedený vzor vstupů a výstupů. Úvodní stránka, jejíž náčrt lze vidět na obrázku 4.2, slouží pouze pro nahrání PDF dokumentu, aby se mohl dále zpracovat. Po stisku tlačítka CHECK bude PDF dokument zpracován, zkontrolován a anotován. Poté bude uživatel přesměrován na stránku, jejíž náčrt je ukázán na obrázku 4.3. Na této stránce bude pro lepší uživatelskou přívětivost přímo zobrazen anotovaný dokument (místo často používaného odkazu na stáhnutí souboru).

The sketch shows the layout of the web application's home page. It features a header with the title 'Theses Checker'. The main content area contains the instruction 'Upload PDF to be checked', followed by an 'UPLOAD' button. Below the button, the filename 'XXXXX.pdf' is displayed. At the bottom of the main area is a 'CHECK' button.

Obrázek 4.2: Tento obrázek obsahuje návrh úvodní stránky webové aplikace Theses Checker



Obrázek 4.3: Tento obrázek obsahuje návrh stránky webové aplikace Theses Checker, kde je zobrazeno výstupní PDF

Jelikož webová aplikace umí kontrolovat pouze jeden dokument, byla navrhována též konzolová aplikace. Tato konzolová aplikace musí pracovat na podobném principu, jako je uveden na obrázku 4.1. Proto webová i konzolová aplikace musí používat stejný program, který bude kontrolovat a anotovat vstupní PDF dokument.

Výstupní dokument bude obsahovat několik anotací pro označení nalezených chyb. Toto označení by mělo být viditelné a mělo by jasně označovat, co za chybu bylo nalezeno. Z tohoto důvodu bylo navrženo používání highlight anotace kombinovanou s pop-up anotací. Highlight anotace označí, kde na stránce byla nalezena chyba a podrobný popis nalezené chyby je možné zobrazit rozkliknutím pop-up anotace, která se váže na danou highlight anotaci.

### 4.3 Využití technologie

Pro zpracování všech programových částí této bakalářské práce, byly využity následující technologie:

**Python:** Jazyk Python byl zvolen, díky knihovně PyMuPDF (popsána v sekci 3.4), která splňuje nejvíce požadavků pro tuto práci. Některé z požadavků jsou například úprava existujícího PDF souboru, přidání základních anotací, extrakce textu a možnost exportovat libovolnou stránku PDF souboru na obrázek (v tomto případě pixelovou mapu). Další funkcí, která byla velmi nápomocná, je zjištění pozice některých objektů, které se vyskytují na stránce.

**PythonAnywhere:** PythonAnywhere je služba poskytující hosting webových aplikací vytvořených v jazyce Python. Je to jedna z mála služeb poskytujících verzi 3.10 jazyka Python i pro své neplacící uživatele. PythonAnywhere nabízí pět typů účtů a mezi nimi je i účet, který je zcela zdarma. Tento účet má několik omezení (např. omezení velikosti datového úložiště na 512 MB, neexistující možnost používání vlastní domény a nutnost alespoň jednoho přihlášení a potvrzení činnosti v intervalu 3 měsíců), ale pro implementovanou aplikaci je dostačující.

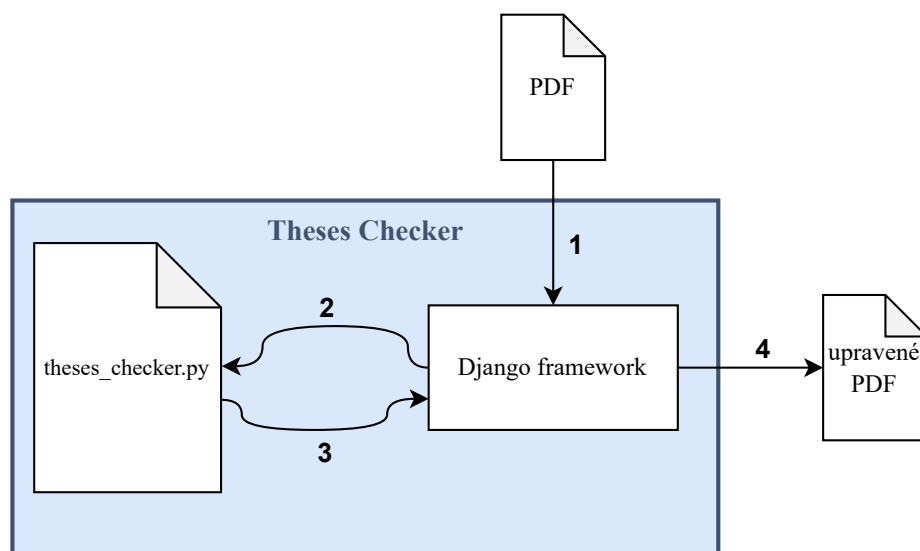
**Django:** Django je jeden z nejznámějších frameworků pro tvorbu webových aplikací v jazyce Python. Tato volba byla závislá právě na vybraném programovacím jazyce a též na službě PythonAnywhere, která má několik návodů na nasazení webových aplikací využívajících právě framework Django. Django dokumentace je velmi rozsáhlá a obsahuje i návod pro vytvoření první webové aplikace.

**HTML, CSS:** HTML a CSS jazyky byly použity pro grafickou tvorbu zobrazených webových stránek. Upravený jazyk HTML se používá pro tvorbu šablon využívaných v Django frameworku. A CSS jazyk byl použit pro definici grafického stylu zobrazených stránek.

**JavaScript:** Jazyk JavaScript byl ve vyvinuté aplikaci použit pro částečnou kontrolu, zda je nahraný soubor ve formátu PDF. Další využití bylo pro zobrazení načítacího elementu („poskakujících“ teček), který je viditelný při čekání na zpracování nahraného PDF dokumentu.

## 4.4 Implementace webové aplikace

Výsledná webová aplikace byla vyvinuta s pomocí Django frameworku. Nahraný soubor se pošle na server pomocí HTTP požadavku metodou POST a uloží se ve specifické složce. Poté se pomocí třídy **Checker** uložené v souboru `theses_checker.py` (popsáno dále v kapitole 4.6) tento soubor zkontroluje a přidají se anotace. Tento výstupní soubor se opět uloží na server do určité složky a původní soubor se ze serveru odstraní. Následně bude uživatel přesměrován na stránku, která zobrazí výstupní anotovaný dokument. Po zobrazení tohoto dokumentu na stránce je dokument odstraněn ze serveru. Postup zpracování nahraného PDF souboru je naznačen na obrázku 4.4.



Obrázek 4.4: Tento obrázek popisuje postup komunikace webové aplikace mezi frameworkem Django a programem pro vyhledávání chyb v PDF dokumentu



## Architektura webové aplikace

Django pracuje s architekturou zvanou model-view-template (dále jen MVT), která je velmi podobná známé architektuře model-view-controller (dále jen MVC). View má v architektuře MVT podobnou roli jako controller z architektury MVC a template z MVT má podobnou roli jako view z MVC.

Všechny šablony použité pro webové stránky jsou uloženy ve složce `templates` a každá rozvíjí šablonu zvanou `base.html` pro jednotný vzhled. Řídící logika webové aplikace je umístěna v souboru `views.py` a program pro vyhledání a označení chyb se vyskytuje v souboru `theses_checker.py`. V souboru `settings.py` se před spuštěním musí nastavit tajný klíč (proměnná `SECRET_KEY`) a může se zde nastavit, zda server poběží v módu pro vývoj či nikoliv (proměnná `DEBUG`). Server webové aplikace se spustí pomocí souboru `manage.py` použitím příkazu: `$ python manage.py runserver`

Všechny potřebné soubory pro správnou funkci webové aplikace jsou uloženy ve složce s názvem `web`. Hlavní struktura této složky vypadá následovně:

```
web
├── files
├── static
│   ├── css
│   └── js
├── templates
│   ├── theses_checker
│   │   ├── annotated.html
│   │   └── index.html
│   ├── 400.html
│   ├── 403.html
│   ├── 404.html
│   ├── 500.html
│   └── base.html
├── theses_checker
│   ├── bl
│   │   └── theses_checker.py
│   └── views.py
├── web
│   └── settings.py
└── manage.py
```

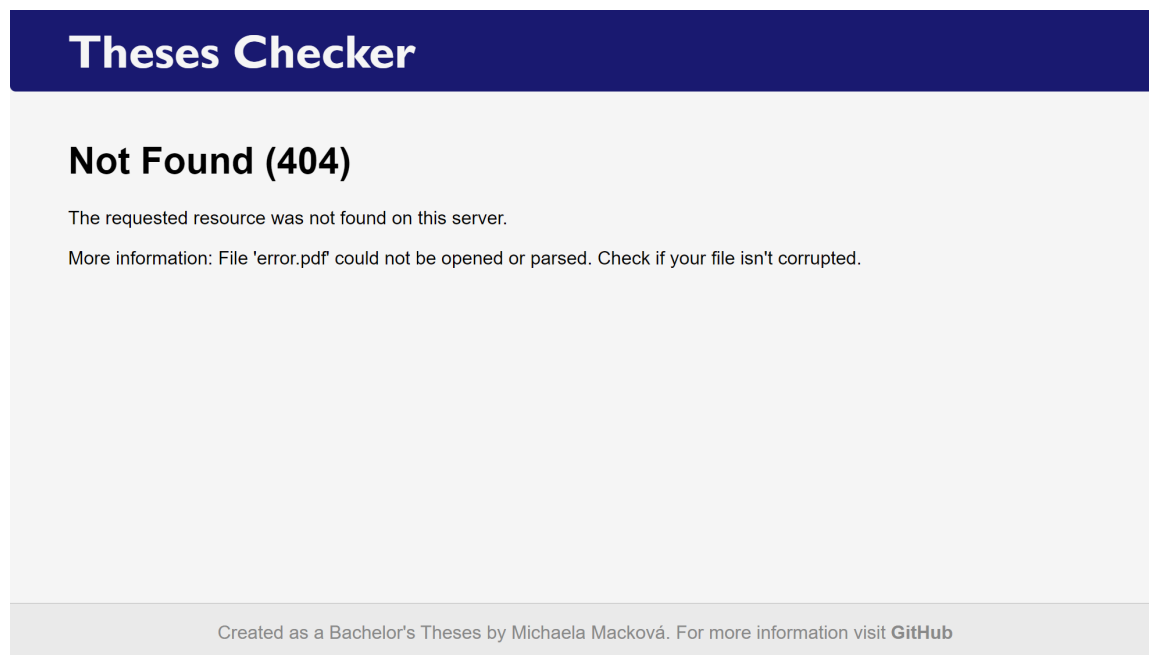
## Implementační detaily

Během vývoje webové aplikace se vyskytlo několik problémů, které se převážně dokázalo vyřešit. Jeden takový problém bylo, v jaký moment se ze serveru odstraní výstupní soubor. Jelikož server má omezené úložiště (v případě neplaceného účtu služby PythonAnywhere je to 512 MB), musí se všechny poskytnuté i vytvořené PDF dokumenty někdy odstranit. Vstupní dokument se odstraní ihned po vytvoření výstupního dokumentu. Tento výstupní dokument se nejdříve musí zobrazit na webové stránce, až poté může být odstraněn. Použité řešení využívá odpovědi<sup>1</sup> nalezené na google groups, ve které se tento problém řeší uložením obsahu PDF souboru do proměnné, vymazáním souboru ze serveru a následným posláním

<sup>1</sup>Odpověď dostupná na: <https://groups.google.com/g/django-users/c/Da8HvVts9pI/m/fwgaFj8RAAAJ>

obsahu vytvořené proměnné jako HTTP odpovědi. Ve vytvořené webové aplikaci se tato HTTP odpověď přijímá do HTML elementu `<iframe>`, který se vyskytuje na zobrazované webové stránce. Aby mohl element `<iframe>` zobrazit webovou stránku, musí se u této zobrazované stránky nastavit X-Frame-Options. Díky odpovědi<sup>2</sup> ze Stack Overflow, bylo možné správně nastavit tuto možnost.

Další detail bylo zobrazení chybových stránek. Django má podporu pro nastavení šablony chybových stránek s chybovým kódem 400, 403, 404 a 500. S každým tímto chybovým kódem se pojí jedna šablona, jejíž soubory se jmenují `400.html`, `403.html`, `404.html` a `500.html`. Příklad chybové stránky, která se zobrazí při nahrání poškozeného souboru, lze vidět na obrázku 4.5.



Obrázek 4.5: Na tomto obrázku je ukázka stránky, která se zobrazí při pokusu o zpracování PDF souboru, který nelze otevřít

## 4.5 Program pro použití v příkazovém řádku

Implementovaná webová aplikace umí najednou kontrolovat pouze jeden nahraný PDF soubor. Vedoucí diplomových prací často potřebují zkontrolovat více takových souborů, proto byl navíc vytvořen program použitelný v příkazovém řádku. Tento program je v souboru `check.py` a používá program ze sekce 4.6. Pro správnou funkčnost musí být tyto dva soubory uloženy ve stejné složce. Program umí na jeden příkaz kontrolovat více PDF souborů a s pomocí nepovinných přepínačů `-o`, `-i`, `-H`, `-t`, `-s`, `-e` a `-b` lze určit, které kontroly budou či nebudou probíhat. Tento script podporuje navíc nepovinný přepínač `--embedded_PDF`, který určuje, zda budou vnořené PDF brány jako vektorové obrázky. Ukázky použití lze vidět na obrázku 4.6.

<sup>2</sup>Odpověď lze nalézt na <https://stackoverflow.com/a/33267908>

```
PS E:\> python .\check.py .\21395.pdf
New file '21395_annotated.pdf' was created. -> mistakes were found
PS E:\>
```

(a) Ukázka spuštění všech kontrol pro jeden PDF soubor

```
PS E:\> python .\check.py .\21395.pdf '.\22208 - eng.pdf' -i -t
New file '21395_annotated.pdf' was created. -> no mistakes found
New file '22208 - eng_annotated.pdf' was created. -> mistakes were found
PS E:\>
```

(b) Ukázka spuštění pouze dvou kontrol na dvou PDF souborech

Obrázek 4.6: Tyto obrázky ukazují možné použití programu Theses Checker z příkazového řádku

## Závislosti

Aby mohla být aplikace použitelná potřebuje následující:

- Python == 3.10.7
- PyMuPDF == 1.20.2
- numpy == 1.23.5

Jiné verze těchto závislostí nebyly testovány.

## 4.6 Implementace programu pro vyhledání chyb a jejich následné vyznačení

Program, který zodpovídá za práci s PDF dokumentem, je obsažen v souboru s názvem `theses_checker.py`. Pro zkontrolování technické zprávy se musí použít funkce `annotate()` ze třídy `Checker`. Při vytváření instance této třídy se musí uvést cesta ke kontrolovanému PDF souboru. Třída `Checker` v sobě obsahuje všechny potřebné metody a proměnné pro správnou funkci hledání a zvýrazňování chyb uvnitř PDF dokumentů.

Třídní proměnné jsou rozděleny do dvou skupin – statické třídní proměnné a třídní proměnné vázané na jednu instanci. Statické třídní proměnné nemění svou přiřazenou hodnotu a třídní proměnné instance naopak během programu svou hodnotu mění.

Třída `Checker` obsahuje několik veřejných proměnných a metod, které se používají při zpracovávání PDF souboru. Tyto proměnné a metody lze vidět na obrázku 4.7. Proměnné `RND_PAGE_CNT`, `HIGHLIGHT_PADDING` a proměnné označené jako *Nastavení barev* jsou statické třídní proměnné. Tyto proměnné lze nastavit před použitím funkce `annotate()`. Proměnné, které jsou označeny jako *Nastavení barev* jsou využívány především pro jednotnou barvu označení nalezených chyb. Veřejné třídní proměnné instance jsou proměnné `borderNotFound` a `mistakes_found`. Tyto dvě proměnné slouží jako pomocný výstup aplikace a nesou v sobě informaci o tom, jak probíhalo zpracování PDF dokumentu.

Checker	Nastavení barev
+ RND_PAGE_CNT: int + HIGHLIGHT_PADDING: float	
Nastavení barev	
+ borderNotFound: bool + mistakes_found: bool	+ HIGH_RED: tuple + WHITE: tuple + RED: tuple + HIGH_ORANGE: tuple
+ init() + annotate()	

Obrázek 4.7: Na tomto obrázku jsou vyznačeny všechny veřejně přístupné proměnné a metody třídy **Checker**. Proměnné pro nastavení barev jsou speciálně vyznačeny

Každá z těchto veřejných proměnných plní vlastní funkci. Statické třídní proměnné se používají pro:

- **RND\_PAGE\_CNT** – Tato proměnná se používá při zjišťování základních informací o daném PDF dokumentu. Tyto informace jsou potřebné pro správné hledání chyb. Proměnná obsahuje maximální počet náhodných stran, které jsou zkoumány pro získání těchto základních informací.
- **HIGHLIGHT\_PADDING** – Tato proměnná obsahuje velikost okraje, který je použit, když je zvýraznění moc těsné zvýrazněnému objektu. Toto nastává při hledání přetečení okraje stránky.
- **HIGH\_RED** – Proměnná obsahuje červenou barvu pro zvýraznění textu při nalezené chybě. Barva je ve formátu RGB<sup>3</sup>.
- **WHITE** – Proměnná obsahuje bílou barvu ve formátu RGB a je považována jako barva pozadí PDF stránky.
- **RED** – Je to červená barva ve formátu RGB. Tato barva je použita při kreslení přímek.
- **HIGH\_ORANGE** – Je to oranžová barva ve formátu RGB pro zvýraznění textu při varování.

Veřejné třídní proměnné instance mají funkci:

- **borderNotFound** – Tato proměnná je veřejná a označuje, zda byl při hledání základních informací o dokumentu nalezen okraj stránky.
- **mistakes\_found** – Veřejná proměnná typu boolean, která označuje, zda byla při kontrole PDF dokumentu nalezena jakákoliv chyba (či varování na chybu).

Dále třída **Checker** obsahuje privátní třídní proměnné instance a privátní metody. Některé privátní proměnné jsou využívány pro navigaci v kontrolovaném dokumentu a další v sobě mají o tomto dokumentu uložená data, která jsou potřebná k prováděným kontrolám. Tyto privátní proměnné jsou vyznačeny na obrázku 4.8.

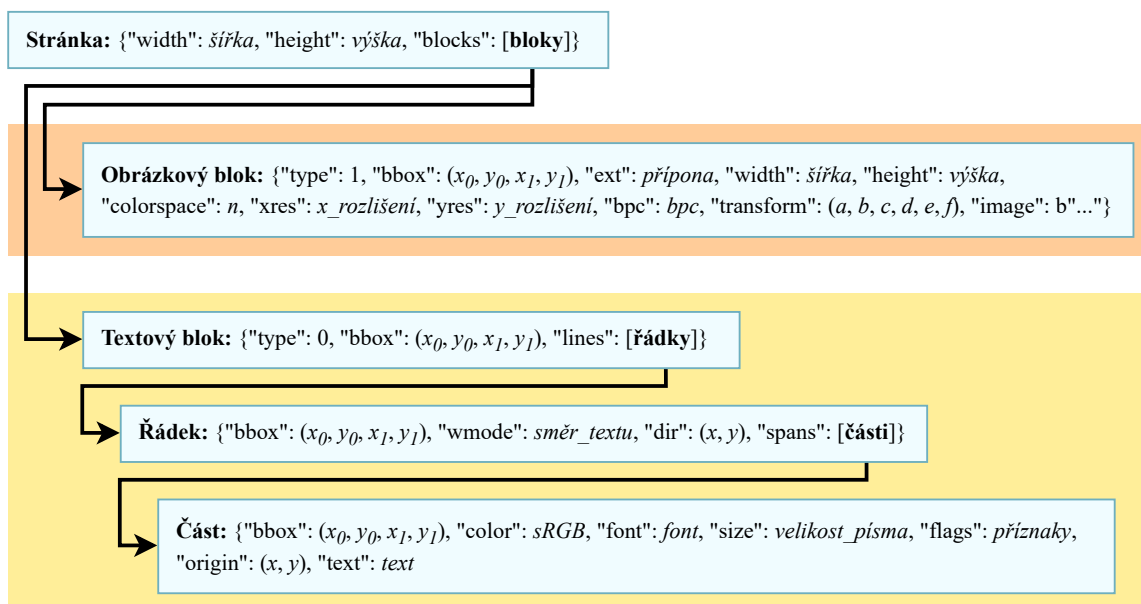
<sup>3</sup>Knihovna PyMuPDF umí pracovat s několika různými formáty barev, jako jsou například formáty CMYK, GRAY, RGB a sRGB. Formát RGB je uspořádaná trojice, kde každý prvek obsahuje celé číslo v intervalu {0; 255}.

Checker	
– document: Document	
– currPage: Page	Navigace
– currDict: dict	
– currPageEmbeddedPdfs: list	
– currTextPage: TextPage	
– currPixmap: Pixmap	
– currPageTextContent: str	
– border: tuple	
– isContentPage: bool	
– regularFont: dict	
– isPreviousTitle: bool	
– embeddedPdfAsImage: bool	

Obrázek 4.8: Tento obrázek ukazuje všechny privátní proměnné třídy **Checker**

Privátní proměnné, které se používají pro navigaci v dokumentu, jsou:

- `__document` – Tato proměnná obsahuje instanci kontrolovaného PDF dokumentu.
- `__currPage` – Toto je proměnná, která obsahuje instanci aktuálně zpracovávané stránky z PDF dokumentu.
- `__currDict` – Tato proměnná obsahuje aktuálně zkoumanou stránku ve formě slovníku, jehož struktura je uvedena na obrázku 4.9, ve kterém je označen jménem *Stránka*.



Obrázek 4.9: Na tomto obrázku je vyznačena struktura slovníku `__currDict`. Obrázek byl převzat a upraven z oficiální PyMuPDF dokumentace [2, Sekce: *TextPage*]

- `__currPageEmbeddedPdfs` – Toto je proměnná, která obsahuje seznam vykreslených vložených PDF na aktuálně zpracovávané stránce. Každá položka tohoto seznamu je pro kompatibilitu s proměnnou `__currDict` typu slovníku, který odpovídá slovníku *Obrázkový blok* uvedeném na obrázku 4.9.
- `__currTextPage` – Tato proměnná obsahuje instanci takzvané *textpage* aktuálně zpracovávané stránky. *Textpage* se používá pro urychlení některých funkcí pracujících s PDF stránkou.
- `__currPixmap` – Tato proměnná obsahuje takzvanou pixelovou mapu aktuálně skenované stránky. Pixelová mapa je podobná obrázku a v tomto programu se používá pouze pro nalezení přetékačích objektů za okraje aktuálně kontrolované stránky.
- `__currPageTextContent` – Tato proměnná obsahuje všechny text vyskytující se na aktuální stránce. Jelikož při extrakci textu z PDF stránky nelze vždy získat nepřerušovaný text (text většinou obsahuje například znak zalomení řádku tam, kde by byl vykreslen na stránce), musí se nejdříve tento extrahovaný text upravit. Text obsažený uvnitř proměnné `__currPageTextContent` je formátován přímo pro čtení, což znamená, že znaky nového řádku, které byly přidány pro správné zobrazení na PDF stránce, jsou nahrazeny za mezeru a jsou odebrány spojovníky u rozdělených slov (např. *spojo-vník*). Každý blok textu vyskytnutý na PDF stránce je v této proměnné oddělen prázdným řádkem.

Privátní proměnné, které uchovávají informace pro prováděné kontroly, jsou:

- `__border` – Tato proměnná obsahuje nalezené souřadnice levého a pravého okraje stránky.
- `__isContentPage` – Tato proměnná označuje, zda aktuální stránka je stránka obsahu.
- `__regularFont` – Tato proměnná obsahuje font, který je ve zpracovávaném dokumentu použit pro klasický text.
- `__isPreviousTitle` – Toto je pomocná proměnná pro kontrolu chybějícího textu mezi nadpisy. V této proměnné je uloženo, zda předchozí blok textu byl nadpis.
- `__embeddedPdfAsImage` – Tato proměnná určuje, zda se má s vnořenými PDF soubory pracovat jako s obrázkem, či nikoliv.

Než se budou moct provést některé kontroly, musí se nejdříve zjistit pár základních informací o kontrolovaném dokumentu (například font normálního textu nebo souřadnice okrajů stránek). Toto získávání informací provádí funkce `__getDocInfo()`, která pro zrychlení programu skenuje pouze předem určený počet stran. Skenované stránky jsou získány náhodně.

Hlavní funkce `annotate()` (naznačená ve výpisu 4.1) přijímá parametry typu boolean, kde každý typ kontroly má vlastní parametr. Tyto parametry určují, zda se má daná kontrola provést, či nikoliv. Další parametry, které tato funkce přijímá jsou `annotatedPath`, který určuje cestu pro vytvořený anotovaný soubor, a `embeddedPdfAsImage`, který je typu boolean a označuje, zda se mají při kontrolách chovat vložené PDF soubory jako obrázky, nebo jako pokračování daného PDF dokumentu. Tato funkce provádí všechny kontroly a vše s nimi spojené. To znamená zjištění potřebných informací, samotná kontrola a označení

nalezených chyb. Každá kontrola má vlastní funkci, která prozkoumá jednu stránku a nalezené chyby označí pomocí PDF anotací (vysvětleny v sekci 3.3). Chyby, které program umí zkontrolovat, jsou přetečení za okraj stránky, špatné použití spojovníku, nevhodná šířka obrázku, vynechaná mezera před levou závorkou, chybějící text mezi názvy sekcí a odkaz na neexistující referenci.

---

```
1 def annotate(document, outPath, check1, check2, ...):
2     docInfo = getDocInfo()
3
4     for currentPage in document:
5         if check1:
6             Check1(currentPage, docInfo)
7
8         if check2:
9             Check2(currentPage, docInfo)
10
11         # ...
12
13     document.save(outPath)
```

---

Výpis 4.1: V tomto výpisu je naznačena veřejná funkce `annotate()`, která provede všechny kontroly a uloží anotovaný soubor

## Nalezení okraje stránky

Hlavní kontrola, na kterou se měl tento program zaměřit, je přetékaní objektů za okraj stránky. Pokud jsou známy souřadnice okraje, nalezení jeho přetečení se řídí algoritmem naznačeném ve výpisu 4.2.

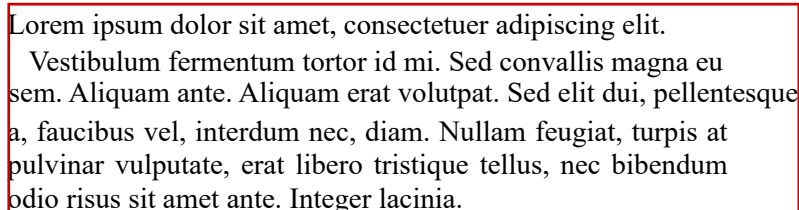
---

```
1 def getOverflow(currPage):
2     overflowList = []
3     pageBorder = getPageBorder()
4     pagePicture = getPageAsPicture(currPage)
5     for lineOfPixels in pagePicture:
6         for i in range(0, len(lineOfPixels)):
7             pixel = lineOfPixels[i]
8             if (not outOfBounds(pixel.coordinates, pageBorder)):
9                 break
10            if (pixel.color != currPage.baseColor):
11                overflowList.add((pixel.x,pixel.y,pageBorder.x0,pixel.y))
12
13        for i in range(len(lineOfPixels), 0, -1):
14            pixel = lineOfPixels[i]
15            if (not outOfBounds(pixel.coordinates, pageBorder)):
16                break
17            if (pixel.color != currPage.baseColor):
18                overflowList.add((pageBorder.x1,pixel.y,pixel.x,pixel.y))
19
20    return overflowList
```

---

Výpis 4.2: Výpis naznačuje funkci `getOverflow()`, která získá souřadnice všech přetečení okraje stránky

PDF dokument nemusí mít uloženou informaci o souřadnicích okraje stránky. Pro jeho nalezení se tak musí daná informace nalézt z již vykreslené stránky. Použitá knihovna PyMuPDF obsahuje funkce, díky kterým lze zjistit polohu vykreslených objektů a podobjektů (například obrázků, bloků textu a řádků v nich obsažených). Pokud blok textu obsahuje přetečení, ohrazení celého bloku obsahuje i toto přetečení (viz obrázek 4.10).



Lorem ipsum dolor sit amet, consectetur adipiscing elit.  
Vestibulum fermentum tortor id mi. Sed convallis magna eu  
sem. Aliquam ante. Aliquam erat volutpat. Sed elit dui, pellentesque  
a, faucibus vel, interdum nec, diam. Nullam feugiat, turpis at  
pulvinar vulputate, erat libero tristique tellus, nec bibendum  
odio risus sit amet ante. Integer lacinia.

Obrázek 4.10: Obrázek ukazuje červeně označené ohrazení bloku textu, který obsahuje přetečení

Při hledání okraje stránky je použitý algoritmus inspirován tím, jak takový okraj hledá člověk. Prozkoumá se každý řádek textu, který se vyskytuje na stránce a uloží se souřadnice jeho levého i pravého okraje. Za okraj stránky se poté považuje medián těchto uložených okrajů řádků. První řádek v odstavci může být odsazený, a proto se pro větší přesnost tohoto algoritmu (uvedeného ve výpisu 4.3) neukládá do potenciálních levých okrajů. Podobné pravidlo platí i pro poslední řádek a jeho pravý okraj.

---

```
1 def getPageBorder(currPage):
2     potentialLeft = []
3     potentialRight = []
4
5     for line in currPage.lines:
6         if not firstLine(line):
7             potentialLeft.add(line.border.x0)
8
9         if not lastLine(line):
10            potentialRight.add(line.border.x1)
11
12    xLeft = median(potentialLeft)
13    xRight = median(potentialRight)
14    return (xLeft, xRight)
```

---

Výpis 4.3: Výpis naznačuje funkci `getPageBorder()`, která nalezne levý a pravý okraj stránky

## Nalezení souřadnic vloženého PDF na stránce

V technické zprávě se často vyskytují obrázky. Jak je řečeno v podkapitole 2.6, je vhodné používat vektorovou grafiku pro grafy a schémata. Vektorové obrázky mohou mít formát PDF, a ty se poté vkládají do PDF dokumentu technické zprávy. Při zpracovávání tohoto PDF dokumentu se nijak nerozlišuje mezi interními částmi původního PDF dokumentu a vloženého PDF obrázku. Kontroly určené pro obrázky tedy nezahrnují tyto vložené PDF a kontroluje se pouze text uvnitř těchto obrázků.



Aby se s vloženým PDF mohlo pracovat jako s obrázkem, musí se zjistit jeho poloha na stránce. Ve formátu PDF nejsou nikde přímo uvedeny souřadnice vloženého PDF obrázku. Obrázek ve formátu PDF je uvnitř PDF dokumentu uložen jako XObject (popsaný v podkapitole 3.2) a jak je na dané stránce vykreslen je popsáno v toku content stream (více lze nalézt v sekci 3.1) té stejné stránky. Pro zjištění souřadnic vykresleného obrázku tedy program simuluje stavy datové struktury graphics state, jenž je popsána v podkapitole 3.2. Přesněji stačí simulovat parametr CTM (current transformation matrix), ve kterém je uvedena matice pro transformaci z uživatelského prostoru do prostoru zařízení. Podtyp objektu XObject, který se nejčastěji používá pro PDF obrázky, je typ Form XObject.

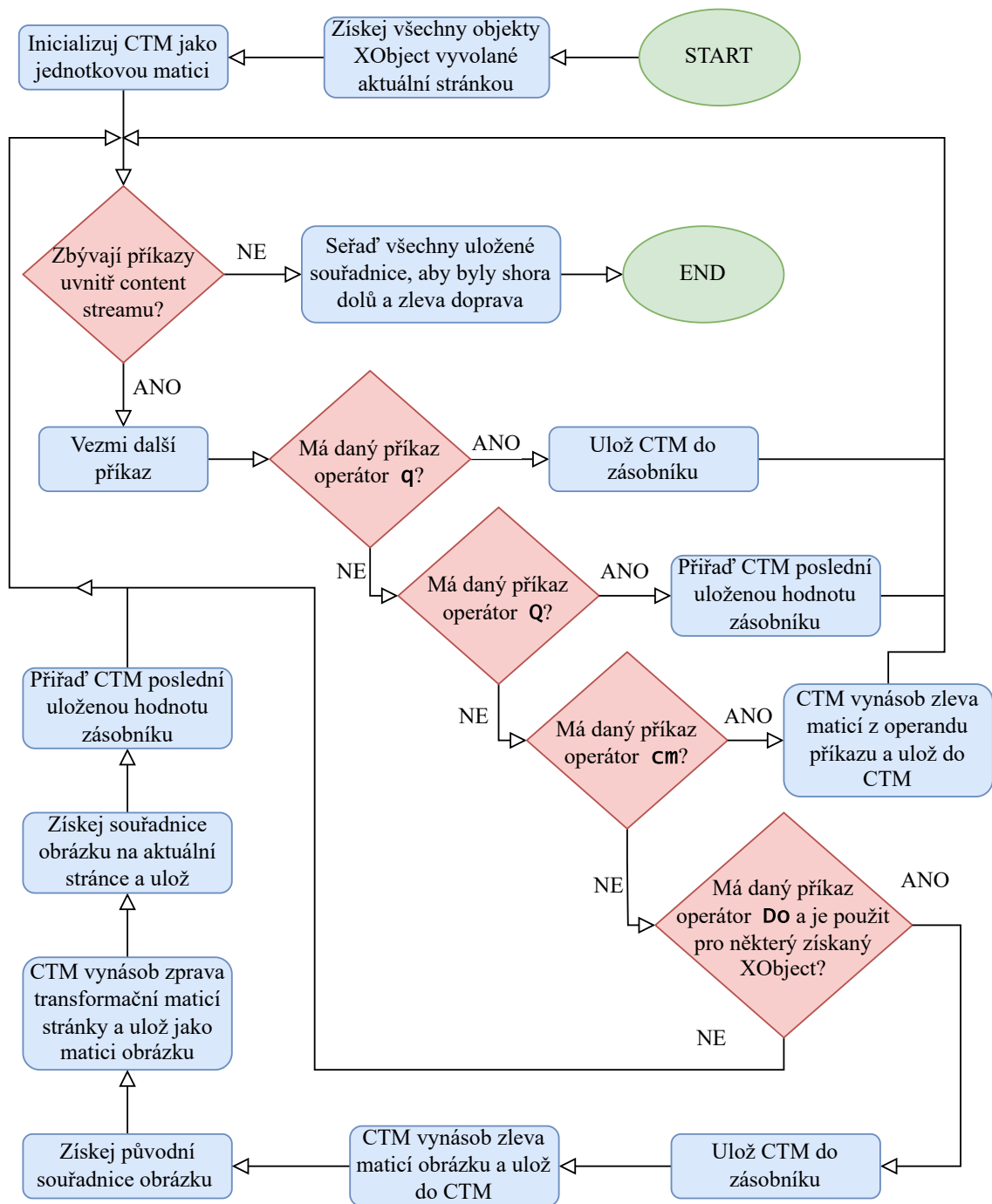
Příkazy uvedené v content streamu, které ovlivňují, jak bude objekt typu Form XObject vykreslen, jsou `q` a `Q` (manipulace s graphics state zásobníkem), `cm` (změna parametru CTM) a `Do` (příkaz pro vykreslení objektu XObject). Příkazem `cm` jsou prováděny elementární transformace obrázku pro jeho správné napolohování na stránce. Jelikož matice CTM musí vždy vyjadřovat transformaci z jednoho prostoru do druhého a příkazy `cm` s touto maticí manipulují, podle dříve uvedeného vzorce (3.2) si dokážeme odvodit vzorec (4.1). V tomto vzorci vyjadřuje matice  $CTM$  aktuální hodnotu CTM uložené uvnitř graphics state,  $M_{cm}$  je matice, kterou vyjadřuje operand použitého příkazu `cm` a matice  $CTM'$  je výsledná matice, jejíž hodnota je nově uložena do CTM parametru datové struktury graphics state.

$$CTM' = M_{cm} \cdot CTM \quad (4.1)$$

Pro získání takových souřadnic obrázku, aby se s nimi mohlo dále pracovat ve vytvořeném programu, se získané souřadnice musí ještě transformovat maticí stránky. Tato matice lze získat přímo jako atribut `transformation_matrix` proměnné `__currPage`. Stejného výsledku lze dosáhnout, když před získáním souřadnic obrázku pomocí vynásobení matice CTM, získáme přímo matici transformace obrázku do našeho vyžadovaného prostoru. Toto je vyjádřeno v rovnici (4.2), kde  $CTM$  je aktuální hodnota parametru CTM ze struktury graphics state,  $M_T$  je matice stránky a  $M_{PT}$  je kombinovaná matice transformace obrázku.

$$M_{PT} = CTM \cdot M_T \quad (4.2)$$

Použitý algoritmus pro získání souřadnic všech vnořených PDF objektů vykreslených na aktuální stránce je naznačen ve vývojovém diagramu na obrázku 4.11.



Obrázek 4.11: Na tomto obrázku je nakreslen vývojový diagram, který popisuje algoritmus pro získání souřadnic vnořených PDF. CTM je proměnná imitující CTM (current transformation matrix) z datové struktury graphics state vyskytující se v PDF. Aktuální stránka je instance PDF stránky, která se právě zpracovává a lze z ní získat content stream a vyvolané XObjekty

## Kapitola 5

# Testování a zhodnocení výsledné aplikace

Testování je důležitou součástí vývoje jakéhokoliv nástroje. Během testování vývojáři dostávají důležitou zpětnou vazbu a mohou tak být odhaleny chyby, které je třeba opravit.

Tato kapitola popisuje, jakým způsobem byl testován výsledný nástroj této bakalářské práce. Jsou zde rozvedeny postupy a výsledky ověřování funkcionality vytvořené aplikace. Dále jsou zde zmíněny nedostatky, které se v této aplikaci vyskytují a rozšíření pro budoucí vývoj tohoto nástroje.

### 5.1 Ověření správné funkcionality vyhledávání chyb

Již během prvních funkčních prototypů byla testována přesnost hledání chyb. Toto testování probíhalo na zveřejněných diplomových pracích, které byly vytvořeny na Fakultě informačních technologií Vysokého učení technického v Brně. Tyto práce byly poskytnuty jako vstupy do programu a jejich anotované výstupy byly zkontrolovány ručně. Dále byly tyto výstupy kontrolovány s poskytnutými posudky oponenta.

Další vstupní data byla poskytnuta doktorem Tomášem Míletem. Tyto data byly rozpracované práce z minulých let, které měli doktora Mileta jako vedoucího. Ke každé takové práci byla vždy poskytnuta její původní forma a její upravená forma obsahující poznámky ke zlepšení. Postup testování na těchto rozpracovaných pracích byl podobný, jako na již zveřejněných, jen tyto výstupy z vytvořené aplikace nebyly porovnávány s posudkem oponenta, ale s poskytnutými poznámkami.

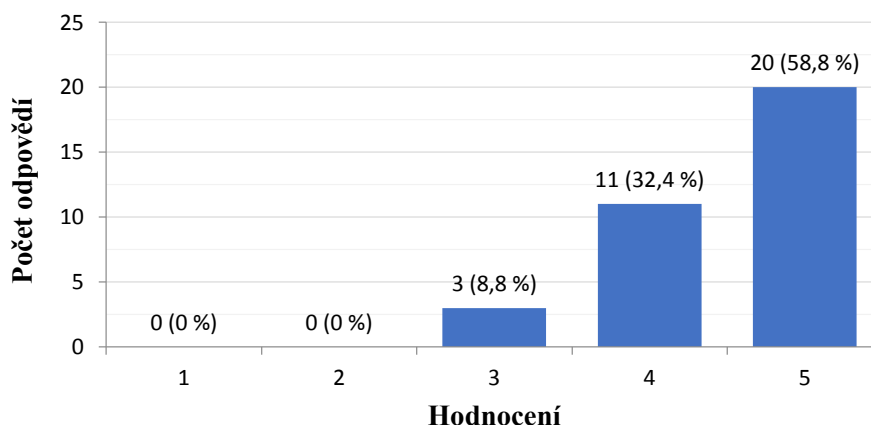
### 5.2 Uživatelské dotazníky

Další formou testování bylo provedeno formou dotazníku. Tento dotazník byl přeposlán několika možným uživatelům aplikace. Celkově na tento dotazník odpovědělo 34 uživatelů. Dotazník obsahoval tyto otázky:

- Pomocí jakého prohlížeče jste aplikaci použili?
- Dokázali jste pomocí aplikace Theses Checker zkontrolovat svou diplomovou práci?
- Dokázali jste jednoduše nahrát PDF soubor?
- Zhodnoťte přehlednost označení chyb ve výsledném PDF

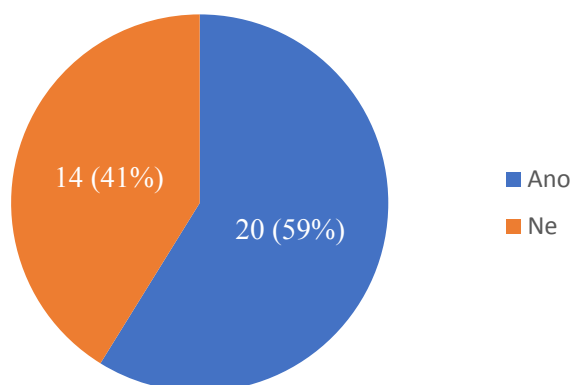
- Zobrazují se vám pop-up anotace?
- Bylo něco špatně označeno jako chyba?
- Jaký typ chyby byl špatně označen?
- Byla práce s webovou stránkou jednoduchá?
- Máte nějaké nápady na rozšíření či vylepšení?

Odpovědi na tento dotazník pomohly vylepšit funkcionalitu vytvořené aplikace. Hlavním účelem tohoto dotazníku bylo ověření již vytvořených kontrol a určení, jak je daná aplikace přehledná. Přehlednost označení nalezených chyb uvnitř PDF dokumentu mělo většinově dobré ohlasy, což lze vyčíst i z obrázku 5.1.



Obrázek 5.1: Obrázek obsahuje graf odpovědí na otázku o přehlednosti označení chyb na stupnici od 1 do 5, kde 1 představuje „Zcela nepřehledné“ a 5 představuje „Jasně“

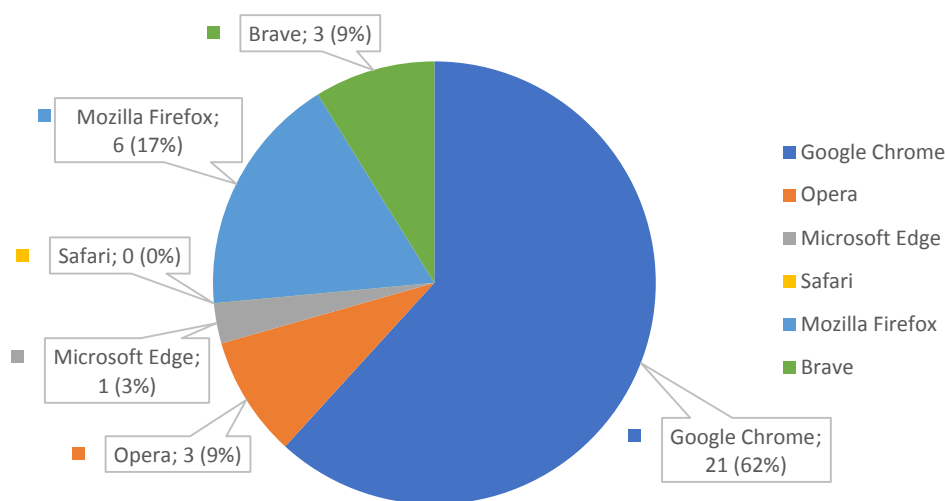
Ověření funkčnosti kontrol bylo provedeno otázkou na špatné označení chyb a upřesněním, který typ chyby byl špatně označen. Shrnutí odpovědí na první otázku lze vidět na obrázku 5.2. Tento poměr byl očekávaný, jelikož tato testovaná aplikace označuje chyby false-positive principem.



Obrázek 5.2: Tento graf ukazuje shrnutí odpovědí na otázku „Bylo něco špatně označeno jako chyba?“

Nejčastější špatně označenou chybou byla vynechaná mezera před levou závorkou. Celkem tuto chybu odpovědělo 12 z 20 (60 %) lidí a dále upřesnili, že toto označení se chybně objevovalo převážně v matematických vzorcích, u názvů funkcí a ve výpisech kódu. Jako další bylo často zodpovězeno chybné označení chybějícího textu mezi nadpisy, které se v odpovědích objevilo celkem 8×. V odpovědích bylo upřesněno, že se toto chybné označení objevovalo okolo tabulek, výpisu kódu a vektorových obrázků.

Dále byla otestována multiplatformnost vytvořené webové aplikace otázkou na použitý webový prohlížeč. Souhrn odpovědí na tuto otázku je ukázán na obrázku 5.3. Zejména takto bylo testováno zobrazení anotací ve zobrazeném výstupním PDF dokumentu. V odpovědích na dotazník byl pouze jediný uživatel, kterému se nezobrazovaly PDF anotace. Tento uživatel použil webový prohlížeč Google Chrome.



Obrázek 5.3: V tomto obrázku je ukázán graf webových prohlížečů, které byly použity uživateli při zkoušení webové aplikace Theses Checker. Tyto informace byly získány z odpovědí na dotazník

Poslední otázka byla nepovinná a dotazovala se na možné vylepšení webové aplikace. Některé z navržených vylepšení jsou:

- možnost dát uživateli vypnutí některých kontrol,
- na stránce s výstupem přidat tlačítko na nahrání dalšího PDF souboru,
- ukázání seznamu nalezených chyb,
- přidání pop-up anotací pro všechny kontroly.

Další návrhy byly na přidání nových kontrol, jako jsou například kontroly pro:

- krátké sekce,
- dlouhé odstavce,
- špatnou kvalitu obrázků (rastrové grafy),

- citace za odstavcem,
- mezeru před odkazem na poznámku pod čarou,
- chybějící mezeru před odkazem na kapitolu.

### 5.3 Známé chyby aplikace

Při ověřování správné funkcionality aplikace bylo odhaleno, že vytvořenou webovou aplikaci nelze použít na mobilním zařízení. Toto je způsobeno načasováním, kdy je odstraněn výsledný PDF soubor (více viz sekce 4.4). Jelikož element `<iframe>` neumí na mobilním zařízení zobrazit PDF dokument, nelze tak získat anotovaný soubor. Při pokusu o stáhnutí tohoto souboru se pošle dotaz na server, kde už tento výstupní soubor neexistuje, a tak je tento vytvořený soubor nenávratně ztracen.

Během testování bylo objeveno několik dalších nedostatků, které jsou spojeny přímo s hledáním chyb. Jelikož se při získávání textu nerozlišuje mezi prostým textem a jakýmkoliv jiným elementem obsahujícím text (například grafy a tabulky), nalezené nedostatky se nejčastěji objevují právě okolo těchto elementů. Jeden takový nedostatek je špatné ukázování chybějícího popisu kapitoly. V tomto případě hraje roli rozeznávání nadpisů, kde text z tohoto elementu může být rozpoznán jako nadpis. Dále mohou tyto elementy ovlivnit nalezení okraje stránky, jelikož texty uvnitř těchto elementů bývají často odsazené.

Dalším často zmiňovaným nedostatkem v odpovědích na formulář jsou označené varování na mezeru před levou závorkou na místě, kde se mezera vyskytovat nemá. Tato místa jsou například názvy funkcí, matematická rovnice a kód ve výpisu. Tento nedostatek není lehké opravit, jelikož by pro vylepšení bylo potřeba poznat kontext textu.

### 5.4 Možné budoucí rozšíření aplikace

Prvním cílem pro budoucí rozšíření bude přidání zaškrtnutých políček na úvodní stránce webu pro nastavení kontrol, které se při zpracování provedou. Tato funkce byla několikrát navržena uživateli jako vylepšení. Jak je uvedeno v podkapitole 5.3, některým označením chyb se nedá vyhnout, proto bylo navrženo, aby bylo možné si zvolit, které kontroly se provedou, a které se vynechají.

Další vylepšení se budou týkat samotných kontrol. První vylepšení bude hledání okrajů pro PDF nastavené na oboustranný tisk. Jelikož u těchto PDF mají sudé a liché stránky jinak nastaveny okraje, nelze v současném stavu kontrolovat přetečení těchto okrajů či další kontroly, které jsou na nalezených okrajích závislé.

Ostatní rozšíření bude přidávání nových kontrol, což je například kontrola použití správných uvozovek, detekce jednopísmenných předložek a spojek na konci řádku a kontrola používání vektorových obrázků.

## Kapitola 6

# Závěr

Cílem této bakalářské práce bylo vytvořit lehce dostupnou aplikaci, která zpracuje nahraný PDF dokument a pomocí PDF anotací v něm vyznačí nalezené chyby. Tato aplikace byla vytvořena jako webový nástroj a nyní je veřejně přístupná pod názvem Theses Checker<sup>1</sup>.

Nejdříve bylo zapotřebí zjistit, které chyby se nejčastěji vyskytují v závěrečných pracích. Ty byly nalezeny zkoumáním zveřejněných textů a posudků diplomových prací vytvořených na Fakultě informačních technologií Vysokého učení technického v Brně. Poté byly zkoumány některé dostupné aplikace pro kontrolu textu. Bylo zjištěno, že žádná dosavadní aplikace se nezaměřuje na typografickou stránku textu, jelikož se většinou zaměřují na gramatiku.

Další na řadě bylo navrhnutí vzhledu aplikace a označení nalezených chyb. V návrhu a implementaci byla dána velká váha na přívětivost pro uživatele. Aplikace se postupně měnila i na základě zpětné vazby od uživatelů, která byla podávána pomocí dotazníku.

Vytvořený program umí kontrolovat šest typů chyb, a to přetečení za okraj stránky, špatné použití spojovníku, nevhodná šířka obrázku, vynechaná mezera před levou závorkou, chybějící text mezi názvy sekcí a odkaz na neexistující referenci. Tento program je využit ve výsledné webové aplikaci a též pro něj byl vytvořen program pro jeho spuštění v příkazovém řádku. Výsledky této práce byly představeny na konferenci Excel@FIT. Pro účast na této konferenci byla povinná tvorba plakátu, který je u této práce přiložen. Dále bylo vytvořeno video (též přiloženo k této práci), které demonstruje použití tohoto webového nástroje, čímž byl splněn poslední bod zadání této bakalářské práce.

Do budoucna bych chtěla vylepšit algoritmus na hledání okrajů PDF stránky a přidat pro něj podporu dokumentů nastavených na oboustranný tisk, jejichž okraje jsou pro sudé a liché stránky rozdílné. Dále bych chtěla vylepšit chování kontrol pro textové objekty, jako jsou tabulky či výpisy. Též bych chtěla doplnit několik dalších kontrol častých chyb, které budou nápomocné uživatelům aplikace.

---

<sup>1</sup>Theses Checker je dostupný na adrese <https://theseschecker.eu.pythonanywhere.com/>

# Literatura

- [1] *Internetová jazyková příručka* [online]. Praha: Ústav pro jazyk český AV ČR, v. v. i., © 2008–2023 [cit. 8. dubna 2023]. Dostupné z: <https://prirucka.ujc.cas.cz/>.
- [2] ARTIFEX. *PyMuPDF* [online]. 2023 [cit. 9. května 2023]. Dostupné z: <https://pymupdf.readthedocs.io/en/latest/index.html>.
- [3] ASUNI, N. *TCPDF* [online]. © 2004–2023 [cit. 2. března 2023]. Dostupné z: <https://tcpdf.org/>.
- [4] BARLOW, J. R. *pikepdf Documentation* [online]. 2022 [cit. 2. března 2023]. Dostupné z: <https://pikepdf.readthedocs.io/en/latest/>.
- [5] DILLON, A. *PDF-LIB* [online]. 2021 [cit. 2. března 2023]. Dostupné z: <https://pdf-lib.js.org/>.
- [6] FPD. *Setasign* [online]. [cit. 2. března 2023]. Dostupné z: <https://www.setasign.com/products/fpdf/about/>.
- [7] ITEXT. *iText 7* [online]. 2023 [cit. 9. května 2023]. Dostupné z: <https://kb.itextpdf.com/home/it7kb>.
- [8] JIHLAVSKÝ, M. V. Jak se vyhnout typografickým hříchům. *Čtenář – Měsíčník pro knihovny* [online]. Prosinec 2015, sv. 67, č. 12, s. 449–451, [cit. 6. dubna 2023]. ISSN 1805-4064. Dostupné z: <https://svkkl.cz/ctenar/clanek/891>.
- [9] KOČIČKA, P. a BLAŽEK, F. *Praktická typografie*. 2. vyd. Brno: Computer Press, 2004. ISBN 80-7226-385-4.
- [10] MDN CONTRIBUTORS. JavaScript. *MDN Web Docs* [online]. 20. února 2023 [cit. 25. února 2023]. Dostupné z: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>.
- [11] MOZILLA a INDIVIDUAL CONTRIBUTORS. *PDF.js* [online]. [cit. 2. března 2023]. Dostupné z: <https://mozilla.github.io/pdf.js/>.
- [12] OSTERLAND, T. pdfAnnotate. *GitHub* [online]. 15. ledna 2022 [cit. 2. března 2023]. Dostupné z: <https://github.com/highkite/pdfAnnotate>.
- [13] PDF 32000-1:2008. *Document management – Portable document format – Part 1: PDF 1.7* [online]. Standard PDF 32000-1:2008, 1. vyd. Adobe Systems Incorporated, červenec 2008 [cit. 17. dubna 2023]. Dostupné z: [https://opensource.adobe.com/dc-acrobat-sdk-docs/pdfstandards/PDF32000\\_2008.pdf](https://opensource.adobe.com/dc-acrobat-sdk-docs/pdfstandards/PDF32000_2008.pdf).



- [14] PHP GROUP. What is PHP? *Php* [online]. [cit. 26. února 2023]. Dostupné z: <https://www.php.net/manual/en/intro-what-is.php>.
- [15] PHP GROUP. What can PHP do? *Php* [online]. [cit. 2. března 2023]. Dostupné z: <https://www.php.net/manual/en/intro-what-cando.php>.
- [16] RYBIČKA, J., ČAČKOVÁ, P. a PŘICHYSTAL, J. *Průvodce tvorbou dokumentů*. 1. vyd. Bučovice: Martin Stríž, 2011. ISBN 978-80-87106-43-3.
- [17] SLEZÁKOVÁ, K. Základy typografie aneb Potěšte svého grafika. *Shockworks* [online]. 10. června 2015 [cit. 27. března 2023]. Dostupné z: <https://www.shockworks.eu/cz/zaklady-typografie-aneb-poteste-sveho-grafika-2/>.
- [18] SZÖKE, I. Textová část BP/DP – Lessons learned #1. *Leaný blog* [online]. 2. ledna 2012 [cit. 1. dubna 2023]. Dostupné z: <http://blog.igor.szoke.cz/2012/01/textova-cast-bpdp-lessons-learned-1.html#.ZChxc3ZBzEb>.
- [19] The Python Tutorial. *Python documentation* [online]. 26. února 2023 [cit. 27. února 2023]. Dostupné z: <https://docs.python.org/3/tutorial/>.
- [20] WAGNER, B., DYKSTRA, T., SANTOS, R. C. M. et al. Prohlídka jazyka C#. *Microsoft Learn* [online]. 22. září 2022 [cit. 26. února 2023]. Dostupné z: <https://learn.microsoft.com/cs-cz/dotnet/csharp/tour-of-csharp/>.