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 — **MICHAELA MACKOVÁ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. TOMÁŠ MILET, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

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 pouštěla a používala.
4. Vyhodnotě 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:	10.5.2023
Datum schválení:	31.10.2022

Abstrakt

Cílem této práce je vytvoření aplikace, která zkontroluje technickou zprávu a označí všechny nalezené chyby pomocí PDF anotací. Aplikace je implementována v jazyku Python a 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 uživatelům při kontrolování textu své vytvářené technické zprávy.

Abstract

Do tohoto odstavce bude zapsán výtah (abstrakt) práce v anglickém jazyce. [[**TODO:**]]

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

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
28. dubna 2023

Poděkování

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

Obsah

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. Při velké koncentraci chyb se může výsledná práce zdát „odfláklá“ a může tak být sníženo její ohodnocení případně přímo zamítnuta.

V současné době existuje několik nástrojů pro kontrolu textu. Tyto nástroje se však převážně zaměřují na gramatické chyby a typografické chyby tak bývají často přehlíženy. Při správné volbě textového procesoru jsou některé chyby automaticky při psaní opravovány, ale opět na to není žádná aplikace zaměřena.

Cílem této práce je zkoumáním zveřejněných diplomových prací nalézt, které chyby se v nich 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 by měla být dostupná bez nutnosti instalování a měla by být co nejvíce intuitivní.

V kapitole ... **[[TODO: popis kapitol]]**

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.

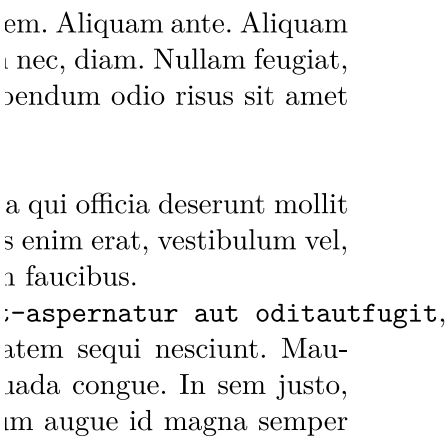
Chyby mohou být způsobeny z nepozornosti, anebo z neznalosti, přičemž druhá možnost je pro autora textu horší, jelikož i po několikátém přečtení nemusí pisatel vůbec poznat, že se jedná o chybu. Správnou volbou textového editoru si tvůrce textu může usnadnit hledání některých chyb. Několik dnešních textových 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 *tip, typ* nebo *autorizace, autentizace*, bývají často touto automatickou kontrolou zanedbávány. Další části této kapitoly popisují několik chyb, které lze nalézt v mnoha diplomových pracích a dále uvádějí, jak se těmto chybám vyhnout.

2.1 Rychlokurz typografie

[[TODO: text]]

2.2 Přetečení objektů za okraj stránky

Přetečení textu za okraj se nejčastěji vyskytuje, když student píše svou diplomovou práci s pomocí jazyka 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 napovědením možného zalomení 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: **Ukázka přetečení za okraj.** [[popisek + přetečení obrázku]]

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, uvedena v internetové příručce Ústavu pro jazyk český [1], 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 tehdy, 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 [7] 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] však uvádí, že je v korespondenci dovoleno znak minus (−) nahradit pomlčkou (–).

Podle článku [8] se tato chyba (naznačena na obrázku **??**) vyskytuje v textu kvůli absenci znaku pomlčky na klávesnici. Místo znaku pomlčky, který je při psaní textu pravděpodobně potřebný 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 LaTeX 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 `$$-$$`.

2.4 Chybějící popis kapitoly

I když je kapitola rozdělena na několik podkapitol, musí i samotná kapitola obsahovat úvod do této kapitoly. Blog [9] vysvětluje, že pokud není uveden popis mezi kapitolou a její podkapitolou, působí poté práce nedopracovaně. Tuto skutečnost lze vidět i na ukázce v obrázku **??**. V tomto místě se hodí napsat 1–2 odstavce, kde bude vysvětlené o čem daná kapitola je a co se v ní čtenář dozví.

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 **??**, 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 LaTeX tohoto lze dosáhnout příkazem `\subsection*{}`. Nadpisy čtvrté a větší úrovně už by se v diplomové práci neměly vůbec vyskytovat.

2.6 Absence vektorové grafiky

Při vkládání obrázku do textu se autor musí zabývat několika otázkami a jedna z nich je určitě 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 **??**. Tento problém kvalitního rozlišení nám může vyřešit použití vektorového obrázku. Podle knihy [7] 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.

2.7 Nepoužívání pevné mezery

Jako spousta jiných věcí i psaní mezer má svá pravidla. Jak zmiňuje článek [2], 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 **??**. 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] 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 LaTeX se pevná mezera zapíše znakem tildy (~) a v editoru WORD se zapíše pomocí kombinace kláves *Ctrl Shift mezera*.

2.8 Použití nesprávných uvozovek

Kapitola 3

PDF soubor

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 [4].

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 můžou 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 `(Toto 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 `<48656c6c6f>`, `<776F726C64>`. Každá dvojice hexadecimálních číslic tvoří jeden znak zakódovaný v ASCII podobě.
- **Jmenný objekt** – Objekt jména je sekvence znaků. Znaky, které 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
xścbd‘ŕg‘b‘‘8 "y
DZn
endstream
endobj
```

Výpis 3.1: TODO:

- **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 například:

```
7 0 obj
   <504446>
endobj
```

Výpis 3.2: TODO:

Nepřímý objekt lze referencovat pomocí *nepřímého odkazu*. Nepřímý odkaz se zapíše číslem objektu, číslem generace a klíčovým slovem `R`, oddělené bílými znaky. Odkaz na výše uvedený nepřímý objet se zapíše jako `7 0 R`.

Struktura souboru

Tato část popisuje, jak jsou výše popsané objekty uložené v PDF dokumentu. Též popisuje, jak je k nim přistupováno a jak jsou aktualizovány. PDF soubor se je rozdělen 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 podsekce vždy začínají řádkem, na kterém se vyskytují dvě čísla. První číslo označuje první objekt v záznamu podsekce a druhé číslo značí, kolik takových záznamů se v dané podsekci vyskytuje. Pod tímto řádkem se nachází samotné záznamy o nepřímých objektech. Záznam o využívaném nepřímém objektu má tvar *nnnnnnnnnn ggggg* **`n`** a je zakončen koncem řádku. Desetimístné číslo *nnnnnnnnnn* označuje offset bajtů od začátku dokumentu, po začátek nepřímého objektu. Pětimístné číslo *ggggg* je číslo generace. Jedna sekce křížových odkazů, rozdělena na 2 podsekce, celkově obsahující 3 záznamy může vypadat například:

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

Výpis 3.3: TODO:

[[od verze 1.5 může být ve streamu]]

- **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`. Patička může vypadat následovně:

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

Výpis 3.4: TODO:

[[od verze 1.5 může být slovník ve streamu]]

Struktura dokumentu

Struktura dokumentu popisuje, jak vypadá vyobrazený dokument. Lze si ji představit jako hierarchii objektů vyskytujících 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. Objekt katalogu dokumentu může být například:

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

Výpis 3.5: TODO:

- **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. Uzel stromu stránek může být například:

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

Výpis 3.6: TODO:

- **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` (popsáno dále v této kapitole), `Rotate`, `Contents` (popsáno dále v této kapitole), `Annots` aj. Jednoduchý objekt stránky může vypadat následovně:

```
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: TODO:

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 na rozdíl od PDF dokumentu se 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 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. Slovník pro resources, který obsahuje font pod jménem `F5` a dva externí objekty pojmenované jako `Im1` a `Im2`, může vypadat následovně:

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

Výpis 3.8: TODO:

[[F5, Im1 a Im2 správné zvýraznění? (emph vs texttt vs none)]]

3.2 Grafika v PDF

Tato sekce čerpá informace ze standardu PDF 32000-1 [4]. 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** – 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 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).

$$M_T = \begin{pmatrix} a & b & 0 \\ c & d & 0 \\ d & f & 1 \end{pmatrix} \tag{3.1}$$

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.

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

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

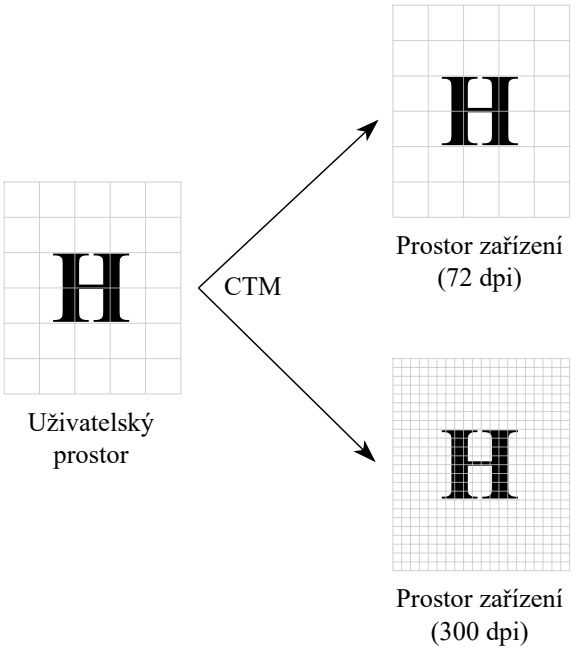
$$P = \begin{pmatrix} x & y & 1 \end{pmatrix} \tag{3.3}$$

Transformovaný bod (souřadnice bodu po použití všech transformací) se vypočítá pomocí vzorce (3.4). P v tomto vzorci označuje původní vektor transformovaného bodu, P' je vektor tohoto bodu po transformaci a M' je matice kombinující všechny provedené transformace.

$$P' = P \cdot M' \tag{3.4}$$

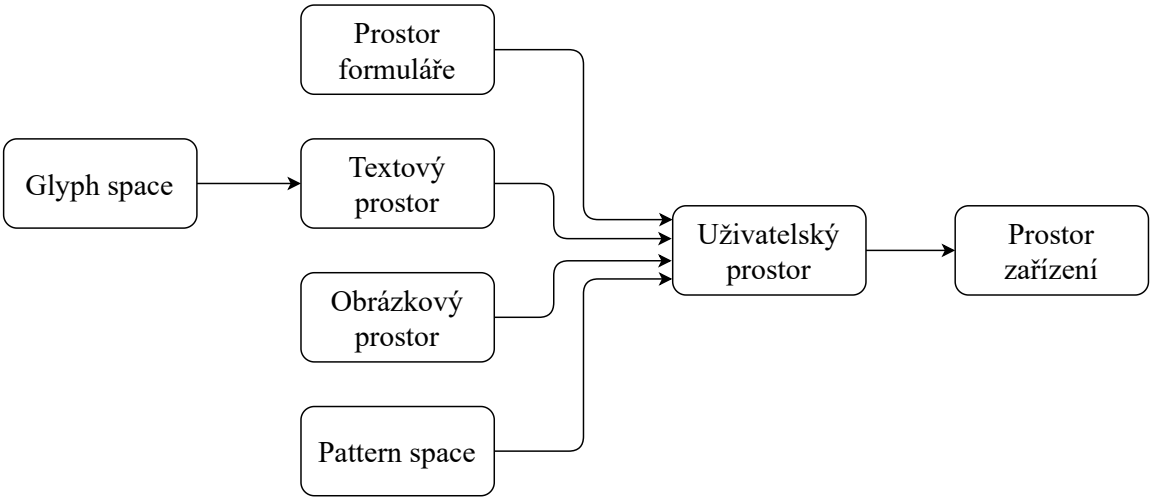
Dále jsou popsány souřadnicové systémy používané v PDF. Vztahy mezi těmito souřadnicovými systémy jsou vyznačeny na obrázku 3.2.

- **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. Převedení z uživatelského prostoru do prostoru zařízení se provádí pomocí *current transformation matrix (CTM)*. CTM je součástí datové struktury *graphics state*, která je popsáná 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 3.1.



Obrázek 3.1: [[**TODO: popisek**]] Inspirován obrázkem ze standardu PDF 32000-1 [4]

- **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 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 3.2: Obrázek označuje vztah mezi souřadnicovými systémy používaných uvnitř PDF. Každá šipka představuje transformaci mezi dvěma souřadnicovými systémy. Inspirováno obrázkem ze standardu PDF 32000-1 [4]

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 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. 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`.

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,* 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 číselná operand jako *flatness* parametr struktury graphics state.

Externí objekty

Externí objekt (též taky *XObject*) ... [[**TODO: obecný popis**]]. Každý XObject lze vykreslit pomocí operátoru `Do` vyskytujícího se uvnitř content streamu. 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, která má hodnotu typu slovník, pod klíčem `XObject`. 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. Prvky s klíčem `Width` a `Height` jsou v těchto objektech povinné. Tyto prvky můžou ovlivnit jak bude vykreslený obrázek vypadat, ale podle aktuálního nastavení stránky můžou mít omezené možnosti. [[**Do**]]
- `/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 [4]):
  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. `BBox` [[**clips**]]
  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. [[**`Annotations` klíč ve slovníku stránky (viz sekce 3.1)**]] 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` 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 vypsaného 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 popisující vhled, 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#

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. [11]

Nejznámější C# knihovna pro práci s PDF dokumenty je **iText 7**[1]. Tato knihovna je dostupná pod *Open Source AGPLv3*[2] licencí a dvěma verzemi komerční licence. **[[popsat funkce knihovny iText 7]]**

JavaScript

JavaScript je dynamicky typovaný, objektově orientovaný, interpretovaný programovací jazyk. Nejčastěji se využívá jako skriptovací jazyk používaný pro vytváření webových stránek, je však často používaný i mimo prostředí webového prohlížeče. Nejznámější z těchto případů je například Node.js, Apache CouchDB a Adobe Acrobat. [3]

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

- **PDF.js**[3] – 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 doporučené se s jejich používáním dobře seznámit před jakoukoliv prací s touto knihovnou. PDF.js je dostupné pod licencí *Apache License 2.0*[4].

- **pdfAnnotate**[5] – Je knihovna vyvinutá specificky pro anotování PDF souborů dostupná pod licencí *MIT License*[6]. Funguje pouze v prostředí webového prohlížeče

[1] https://kb.itextpdf.com/home
[2] https://itextpdf.com/how-buy/AGPLv3-license
[3] http://mozilla.github.io/pdf.js/getting_started/
[4] https://github.com/mozilla/pdf.js/blob/master/LICENSE
[5] https://github.com/highkite/pdfAnnotate
[6] https://github.com/highkite/pdfAnnotate/blob/master/LICENSE

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 a díky tomu je možné je zobrazit i mimo vytvořenou aplikaci.

- **PDF-LIB**[7] – 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*[8].

PHP

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. [5], [6]

- **TCPDF**[9] – Tato knihovna je zaměřena na vytváření PDF dokumentů, mezi její 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*[10] licencí.

- **FPDI**[11] – Je knihovna pro používání stránek z existujícího PDF dokumentu jako šablonu do nového PDF dokumentu. Při opakovaném použití jedné šablony tato knihovna zajistí, že 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 FPDI dostupná pod licencí *MIT license*[12].

Python

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. [10]

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

[7] https://pdf-lib.js.org/
[8] https://github.com/Hopding/pdf-lib/blob/master/LICENSE.md
[9] https://tcpdf.org/
[10] https://tcpdf.org/docs/license/
[11] https://www.setasign.com/products/fpdi/about/
[12] https://www.tldrlegal.com/l/mit

- **PyMuPDF**[13] – Je Python verze MuPDF. Je dostupná pod dvěma různými licencemi, a to pod licencí *Open Source – AGPL*[14] a komerční[15] 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 je nainstalován též 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. **[[vlastnosti pouze pro PDF – anotovat, vyplňovat formuláře, vytvoření nového]] [[přístup z příkazové řádky]]**

- **pikepdf**[16] – Tuto knihovnu lze používat pro vytváření, čtení i úpravu PDF dokumentů. Je to Python verze C++ knihovny QPDF. Pro požívání knihovny je nutné být seznámen se specifikací PDF formátu. Knihovna je dostupná pod *Mozilla Public License 2.0*[17] licencí. Pikepdf dokáže kopírovat stránky do jiného PDF souboru, extrahovat obrázky, nahradit obrázek za jiný, upravovat metadata souboru a další.

[13] https://pymupdf.readthedocs.io/en/latest/
[14] https://artifex.com/licensing/agpl/
[15] https://artifex.com/licensing/commercial/
[16] https://pikepdf.readthedocs.io/en/latest/
[17] https://github.com/pikepdf/pikepdf/blob/master/LICENSE.txt

Kapitola 4

Návrh a implementace aplikace

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. 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 LaTeX, která byla poskytnuta univerzitou Vysoké učení technické v Brně, přesněji Fakultou informačních technologií. Aplikace musí být lehce přístupná, nejlépe bez nutnosti jakékoliv instalace. A měla by být funkční na co nejvíce platformách, aby byla zajištěna dostupnost pro co nejvíce uživatelů.

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. 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.

4.2 Návrh aplikace

[[webová aplikace, nákresy webovky, princip nahrát -> zpracovat -> ukázat, cílová skupina?,]]

4.3 Využité technologie

Python

Django

HTML, CSS

JavaScript

4.4 Program 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 funkce 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 jsou využívány především pro jednotnou barvu označení nalezených chyb. Jsou to proměnné:

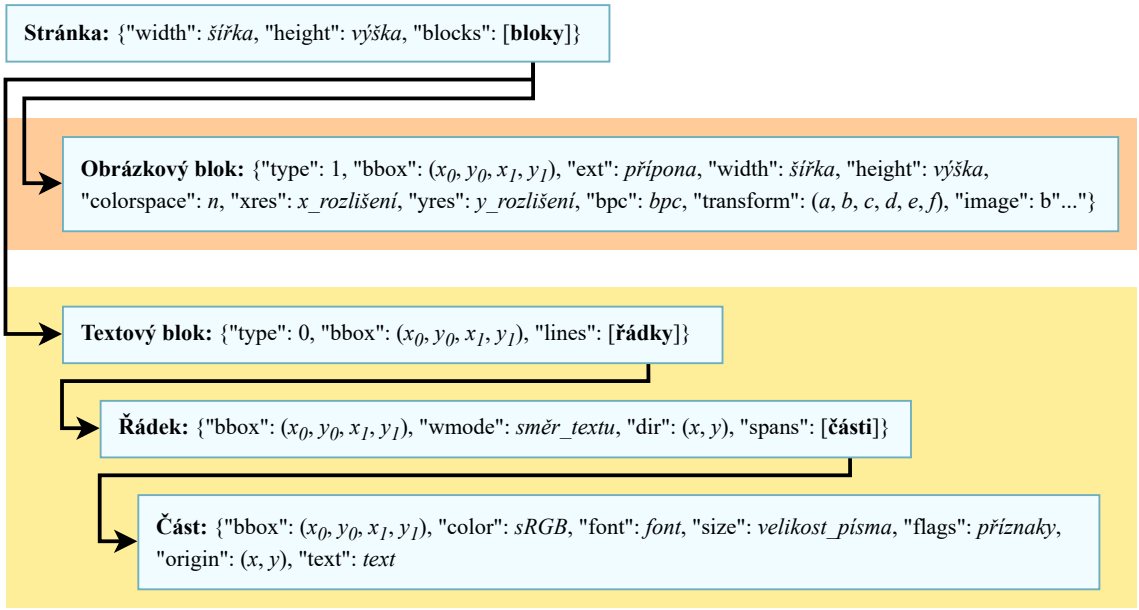
- `RND_PAGE_CNT` – Tato proměnná se používá při zjišťování základních informací o dodaném PDF dokumentu. Tyto informace jsou potřebné pro správné hledání chyb. Proměnná obsahuje maximální počet náhodných stran, které jsou zkoumány, pro získání těchto základních informací.
- `HIGH_RED` – Proměnná obsahuje červenou barvu pro zvýraznění textu při nalezené chybě. Barva je ve formátu RGB[1].
- `HIGH_ORANGE` – Je to oranžová barva ve formátu RGB pro zvýraznění textu při varování.
- `HIGHLIGHT_PADDING` – Tato proměnná obsahuje velikost okraje, který je použit, když je zvýraznění moc těsné zvýrazněnému objektu. Toto nastává při hledání přetečení okraje stránky.
- `RED` – Je to červená barva ve formátu RGB. Tato barva je použita při kreslení přímek.
- `WHITE` – Proměnná obsahuje bílou barvu ve formátu RGB.

Třídní proměnné instance během programu mění svou hodnotu. Některé jsou využívány pro navigaci v kontrolovaném dokumentu a další v sobě mají uložená data o tomto dokumentu, která jsou potřebná k prováděným kontrolám. Některé z důležitých proměnných jsou:

- `mistakes_found` – Veřejná proměnná typu boolean, která označuje, zda byla při kontrole PDF dokumentu nalezena jakákoliv chyba (či varování na chybu).
- `borderNotFound` – Tato proměnná je veřejná a označuje zda byl při hledání základních informací o dokumentu nalezen okraj stránky.
- `__document` – Tato privátní proměnná obsahuje instanci kontrolovaného PDF dokumentu.

[1] Formát RGB – Uspořádaná trojice, kde každý prvek obsahuje celé číslo v intervalu $< 0; 255 >$.

- `__currPage` – Privátní proměnná, která obsahuje instanci aktuálně zpracovávané stránky z PDF dokumentu.
- `__currTextPage` – Tato privátní proměnná obsahuje instanci takzvané *textpage* aktuálně zpracovávané stránky. Textpage se používá pro urychlení některých funkcí pracující s PDF stránkou.
- `__currPixmap` – Tato privátní proměnná obsahuje pixelovou mapu aktuálně skenované stránky. Pixelová mapa je podobná obrázku a v tomto programu se používá pouze pro nalezení přetékajících objektů za okraje aktuálně kontrolované stránky.
- `__currDict` – Tato privátní proměnná obsahuje aktuálně zkoumanou stránku ve formě slovníku, jehož struktura je uvedena na obrázku 4.1.



Obrázek 4.1: TODO: https://pymupdf.readthedocs.io/en/latest/textpage.html

- `__currPageEmbeddedPdfs` – Privátní proměnná, která obsahuje seznam vykreslených vložených PDF na aktuálně zpracovávané stránce. Každá položka tohoto seznamu je pro kompatibilitu s proměnnou `__currDict` typu slovník, který odpovídá slovníku *Image block* uvedeném na obrázku 4.1.
- `__currPageTextContent` – Tato privátní proměnná obsahuje všechen 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 obsažen uvnitř proměnné `__currPageTextContent` je formátován přímo pro čtení (falešné nové řádky nahrazeny za mezeru a jsou odebrány spojovníky u rozdělených slov např. *spojo-vník*), kde každý blok je oddělen prázdným řádkem.[[**TODO: samotná podsekce?, proměnná obsahuje i text z vložených pdf**]]

Než se budou moct provézt 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()` 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).

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

Výpis 4.1: TODO:

Extrahovaný text ve formátu pro čtení

[[proč existuje proměnná `__currPageTextContent` a na co se to potom používá]]

Nalezení okraje stránky

[[pseudokód, popsat princip, blbne u textových objektů (tabulka, listing, grafy?)]]

Nalezení souřadnic vloženého PDF na stránce

[[pseudokód, popsat princip, funkčí pouze pro Form XObjects]]

4.5 Doplňující program pro použití v příkazovém řádku

[[je to něco navíc, musí se instalovat python a pymupdf, jak to nainstalovat, aby to fungovalo, ukázka použití, proč to vůbec vzniklo (hromadná oprava)]]

4.6 Implementace webové aplikace

Architektura webové aplikace

[[MVC?, kde jsou jaké soubory,]]

Implementační detaily

[[error views, vymazání anotovaného souboru,]]

4.7 Ukázka použití vytvořené webové aplikace

Kapitola 5

Testování a zhodnocení výsledné aplikace

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[1].

Nejdříve bylo zapotřebí nalézt, které chyby se nejčastěji vyskytují v závěrečných pracích. Poté ...

[[TODO: vylepšení: proměnná `__currPageTextContent` nebude obsahovat z embedded pdfs]]

[1] 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] 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.

[3] 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.

[4] 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.

[5] PHP Group. What is PHP? *Php* [online]. [cit. 26. února 2023]. Dostupné z: https://www.php.net/manual/en/intro-whatis.php.

[6] PHP Group. What can PHP do? *Php* [online]. [cit. 2. března 2023]. Dostupné z: https://www.php.net/manual/en/intro-whatcando.php.

[7] Rybička, J., Čačková, P. a Přichystal, J. *Průvodce tvorbou dokumentů*. 1. vyd. Bučovice: Martin Stříž, 2011. ISBN 978-80-87106-43-3.

[8] 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/.

[9] 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.

[10] The Python Tutorial. *Python documentation* [online]. 26. února 2023 [cit. 27. února 2023]. Dostupné z: https://docs.python.org/3/tutorial/.

[11] 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/.