VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

EXTRAKCE DAT Z DOKUMENTŮ PDF

DATA EXTRACTION FROM PDF DOCUMENTS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MICHAL BARTOŠÁK**
AUTHOR

VEDOUCÍ PRÁCE **doc. Ing. RADEK BURGET, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

146957

Ústav: Ústav informačních systémů (UIFS)
Student: **Bartošák Michal**
Program: Informační technologie
Specializace: Informační technologie

Název: **Extrakce dat z dokumentů PDF**
Kategorie: Informační systémy
Akademický rok: 2022/23

Zadání:

1. Prostudujte dokumentový formát PDF a dostupné knihovny pro čtení a zpracování dokumentů v tomto formátu. Zaměřte se zejména na implementační platformu Java.
2. Seznamte se s požadavky zadavatele na extrakci datových položek z konkrétních PDF dokumentů.
3. Na základě analýzy požadavků a po konzultaci s vedoucím navrhněte architekturu rozšiřitelného nástroje pro identifikaci a extrakci konkrétních dat z PDF na základě uživatelem dodané specifikace.
4. Implementujte navržený nástroj pomocí vhodných technologií.
5. Proveďte testování na dostupných reálných dokumentech.
6. Zhodnoťte dosažené výsledky.

Literatura:

- Adobe Systems, Inc.: PDF Reference (sixth edition), Version 1.7, November 2006
- Dvořáček, Libor. Využití získávání znalostí pro data z PDF souborů. Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií.
- Oltmanová, Kristína. Statistická analýza dat z PDF souborů. Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií.

Při obhajobě semestrální části projektu je požadováno:
Body 1 až 3.

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

Vedoucí práce: **Burget Radek, doc. Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1.11.2022
Termín pro odevzdání: 10.5.2023
Datum schválení: 21.10.2022

Abstrakt

Práce se zaměřuje na extrakcí informací ze zdravotních záznamů ve formátu PDF, které byly vytvořeny srdečními stimulátory při pravidelné kontrole pacientů v nemocnici. Výsledkem této práce je desktopová aplikace v programovacím jazyce Java, která získává a analyzuje informace ze záznamů pomocí knihoven PDFBox a pdf2dom. Výstupem aplikace je CSV soubor, který reprezentuje získané hodnoty formou tabulky, a extrahované obrázky, které se ukládají do výstupní složky určené uživatelem. Testování aplikace na záznamech od tří různých společností prokázalo, že je extrakce záznamů velmi spolehlivá (celkové metriky přesnosti i úplnosti dosáhly téměř vždy 100 %), pokud jsou správně nastaveny její argumenty.

Abstract

The work focuses on extracting information from medical records saved in PDF format, which were created by heart pacemakers during regular patient monitoring in the hospital. The result of this work is a desktop application written in Java that retrieves and analyzes data from records using PDFBox and pdf2dom libraries. The output of the application is a CSV file, which represents the acquired values in table form, as well as extracted images that are saved to a user-defined output folder. Application testing on records from three different companies proved that record extraction is highly reliable (with overall precision and recall metrics reaching almost 100 % in every test), provided that the application arguments are correctly set.

Klíčová slova

PDF, extraktor, extrakce dat, zdravotní záznam, PDFBox, pdf2dom

Keywords

PDF, extractor, data extraction, medical record, PDFBox, pdf2dom

Citace

BARTOŠÁK, Michal. *Extrakce dat z dokumentů PDF*. Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. Radek Burget, Ph.D.

Extrakce dat z dokumentů PDF

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. Ing. Radka Burgeta, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Michal Bartošák
9. května 2023

Poděkování

Chtěl bych touto cestou poděkovat mému vedoucímu, panu doc. Ing. Radku Burgetovi Ph.D., za jeho vřelou ochotu a vstřícnost, kterou mi poskytl během této bakalářské práce. Rád bych také poděkoval panu doktoru MUDr. Milanu Sepšimu, Ph.D., za poskytnutí tématu této bakalářské práce a Davidovi Zouharovi za pomoc s korekturou gramatiky a formulací vět. Na závěr bych rád poděkoval svým rodičům, kteří mi umožnili klidné a ničím nerušené psaní, a svojí Anetce, která mi poskytla tu největší psychickou podporu, kterou mi snad může jenom člověk dát.

Obsah

Kapitola 1

Úvod

S rostoucím množstvím digitalizovaných informací se neustále zvyšuje i počet zařízení, která produkují elektronické dokumenty. Tyto dokumenty se často ukládají ve formátu PDF, který zajišťuje jejich přehledné a konzistentní zobrazení. Přehlednost se ale postupně vytrácí s narůstajícím počtem dokumentů, které se generují v průběhu let, a s narůstajícím počtem uživatelů, které tyto zařízení používají. Aby bylo možné extrahovat tak velké množství dat, například kvůli sledování stavu pacienta při pravidelných kontrolách v průběhu let, tak bylo za potřebí vytvořit aplikaci, která automatizuje jejich zpracování. S takovouto situací se potýkalo i oddělení interní medicíny a kardiologie ve Fakultní nemocnici Brno, které provádí implantace a následné kontroly stimulátorů srdce. Tyto stimulátory při každé kontrole vytvářejí záznamy ve formátu PDF, které obsahují široké spektrum informací o stavu pacienta a stavu zařízení. Z těchto informací si následně doktoři musí vybírat taková data, která jsou podle nich důležitá pro celkovou analýzu zdravotního stavu pacienta.

Cílem této práce je vytvoření efektivního nástroje pro extrakci dat, který zautomatizuje a zrychlí proces získávání informací z velkého množství záznamů, které mohou pocházet z odlišných zařízení a mohou mít různou strukturu. Tento nástroj tedy musí být konfigurovatelný pro libovolný typ dokumentů. Nástroj bude extrahované informace ukládat do výstupního souboru, který bude reprezentovat tabulku získaných dat (např. soubor ve formátu CSV). Výstupní soubor bude možné otevřít pomocí běžných tabulkových procesorů, jako je například Microsoft Excel nebo LibreOffice Calc. Kromě textových informací bude nástroj umožňovat i extrakci obrázků, které se budou ukládat do speficikované výstupní složky.

V kapitole 2 jsou detailně popsány jednotlivé části formátu PDF, včetně objektů, které jsou spojeny s tvorbou textu. Kapitola 3 provádí průzkum placených i neplacených knihoven pro čtení a zpracování dokumentů PDF, které jsou implementovány v programovacím jazyce Java. V kapitole 4 je představen návrh architektury extrakčního nástroje a kapitola 5 se podrobně zaměřuje na jeho implementaci (obě kapitoly rozebírají také režim ladění). Kapitola 6 se věnuje návrhu a implementaci grafického uživatelského rozhraní a výsledky testování extrakčního nástroje jsou prezentovány v kapitole 7. Veškeré dosažené poznatky jsou nakonec shrnuty v závěrečné kapitole 8.

Kapitola 2

Formát PDF

Kapitola se věnuje podrobnému rozboru jednotlivých částí formátu PDF. Na začátku se uvádí samotná definice formátu (sekce 2.1). Poté se analyzuje jeho syntaxe, která se dělí na:

- Objekty (sekce 2.2)
- Strukturu souboru (sekce 2.3)
- Strukturu dokumentu (sekce 2.4)
- Datové proudy (sekce 2.5)

V sekci datových proudů se navíc rozebírají i textové objekty a fonty, jelikož je na nich postavena celá extrakce. Všechny informace v této kapitole byly získány z oficiální dokumentace formátu PDF od společnosti Adobe [2], pokud nebude uvedeno jinak.

2.1 Definice

PDF je formát souboru vytvořený v roce 1993 společností Adobe Systems, který umožňuje jednotnou výměnu a zobrazení elektronických dokumentů, nezávisle na hardwaru, sofwaru nebo operačním systému. Zkratka PDF je odvozená z anglického názvu Portable Document Format (přenositelný formát dokumentů). Od roku 2008 se stal formát otevřeným standardem, který spravuje Mezinárodní organizace pro normalizaci (International Organization for Standardization, zkratka ISO) [1]. Ačkoliv je založený na jazyce PostScript, nejedná se o plnohodnotný jazyk, jelikož neobsahuje běžné prvky programovacích jazyků jako jsou například procedury, proměnné nebo řídící konstrukce. Formát umožňuje vkládat do souboru různý multimediální obsah (nejen text nebo obrázky, ale i videa, tabulky, animace a další multimediální prvky). Obsah souboru může být zapezpečen šifrováním nebo elektronickým podpisem (případně oběma metodami současně).

2.2 Základní objekty

Dokument PDF je datová struktura, která se skládá z několika základních datových objektů. Objekty se dělí na booleovské hodnoty, čísla, řetězce, jména, pole, slovníky, streamy a objekt null. Kromě základních objektů se v PDF může objevit i tzv. nepřímý objekt (*indirect object*), na který se mohou odkazovat ostatní objekty.

Booleovké hodnoty

Booleovské objekty jsou reprezentovány hodnotami `true` a `false`. Hodnoty se mohou použit jako prvky polí nebo jako záznamy ve slovnících. Dále se mohou vyskytovat i výpočetních funkcích kalkulačky PostScriptu jako výsledky booleovských a relačních operátorů nebo jako operandy podmíněných operátorů `if` a `else`.

Čísla

Numerické objekty se v PDF dělí na 2 typy: celá čísla a reálná čísla. Celé číslo (integer) se zapisuje jako číslo v desítkové soustavě s volitelným znaménkem:

```
+15 0 -84
```

V případě, že hodnota celého čísla překročí limit nastavený pro celá čísla, je hodnota převedena na reálné číslo.

Reálné číslo (real) se zapisuje jako číslo s volitelným znaménkem a desetinnou tečkou, která se může nacházet na libovolném místě:

```
+19.57 .25 -31.
```

Ve chvíli, kdy hodnota reálného čísla překročí limit nastavený pro reálná čísla, nastane chyba.

Řetězce

Řetězcové objekty se skládají z posloupnosti bajtů, které reprezentují celé čísla v rozsahu od 0 do 255. Objekty se ukládají do kompaktnější formy, než je forma objektů celých čísel. Existují 2 typy řetězcových objektů: literální řetězce a hexadecimální řetězce.

Literální řetězce se zapisují jako libovolná sekvence znaků, které se uvádí do kulatých závorek. V literálním řetězci se mohou vyskytovat jakékoliv znaky kromě zpětného lomítka a nevyvážených závorek. Ty se zaznamenávají pomocí únikového označení (zpětného lomítka). Vyvážené závorky mohou být uvedeny bez tohoto označení. Kromě znaků se v řetězci mohou vyskytovat i řídící sekvence. Následující příklady představují validní řetězce:

```
(Vysoké učení technické v Brně\n\tFakulta informačních technologií)
(Úspěšně ukončené bakalářské studium :-\).)
```

Pokud je řetězec příliš dlouhý, aby byl na jednom řádku, může být rozdělen do několika linií pomocí zpětného lomítka nebo značky konce řádku.

Hexadecimální řetězce slouží pro zápis binárních dat do souboru PDF. Zapisují se jako posloupnost hexadecimálních číslic do ostrých (úhlových) závorek:

```
<2AFF1C2D938D6D5F7A1BCA6120256F9D2E>
```

Každá dvojice definuje jeden bajt řetězce. Pokud je posloupnost lichá, předpokládá se, že poslední číslice je 0. Bílé znaky (např. mezera nebo tabulátor) jsou ignorovány.

Jména

Jmenné objekty jsou symboly bez vnitřní struktury (tzv. atomické symboly), které jsou jednoznačně definovány posloupností znaků. Pojem „jednoznačně definovaný“ znamená, že dva objekty jsou identické, pokud jsou složené ze stejné posloupnosti znaků. Jméno objektu

je uvozeno zpětným lomítkem, které není do samotného jména zahrnuto. Za lomítkem nesmí být bílé znaky ani oddělovače. Oddělovače jsou reprezentovány znaky: `(, ), <, >, [, ], {, }, /, %`. Rozlišují se malá a velká písmena. Následující příklady ukazují validní jména:

```
/Michal_Bartošák
/***Extrakce~dat~z~PDF***
```

Pro zapsání oddělovačů a bílých znaků se používá dvouciferný hexadecimální kód, před kterým je uveden znak mřížky (#):

```
/Bakalářská#20práce = Bakalářská práce
```

Hexadecimální kód se vnímá jako jeden znak.

Pole

Objekty typu pole jsou jednorozměrné kolekce po sobě jdoucích objektů, které se uvádějí do hranatých závorek. Tyto objekty mohou být i heterogenní, což znamená, že pole může obsahovat objekty různých typů. Pole může být zapsáno například tímto způsobem:

```
[/Michal#20Bartošák (FIT) 2023 true]
```

Vícerozměrné pole se vytváří použitím polí jako jednotlivých prvků, které se vloží do dalšího pole.

Slovníky

Slovníkové objekty jsou základními stavebními kameny dokumentu PDF. Používají se ke sběru a propojení charakteristik složitých objektů, jako je například font nebo stránka dokumentu. Jedná se o asociativní tabulky, které obsahují páry objektů nazývané jako záznamy. Záznam se skládá z dvojice klíč – hodnota, přičemž klíč představuje objekt jména a hodnota může představovat objekt libovolného typu. V případě, že je hodnota v záznamu `null`, se daný záznam interpretuje stejně, jako kdyby neexistoval. Pokud existují dva záznamy se shodnými klíči, hodnota klíčů není definována. Slovník je zapsán jako posloupnost záznamů ve dvojtých úhlových závorkách:

```
<< /Type /Font
   /Subtype /Type1
   /BaseFont /Helvetica
>>
```

Podle dohody záznam `Type` označuje typ objektu, který slovník popisuje, a záznam `Subtype` označuje specifickou podkategorii v případě, že je typ příliš obecný. Oba záznamy mají vždy jako hodnotu objekt jména.

Streamy

Stream objekty se skládají z posloupnosti bajtů stejně jako řetězce. Od řetězců se odlišují tím, že nemají omezenou délku a mohou se číst postupně. Z tohoto důvodu se streamy používají pro objekty s velkým množstvím dat, jako jsou například obrázky nebo popisy stránek. Stream se skládá ze slovníku, za kterým následuje sekvence bajtů, která je ohraničená klíčovými slovy `stream` a `endstream`. Stream se reprezentuje nepřímým objektem a slovník streamu se reprezentuje přímým objektem.

Klíčová slova musí být uvedena zvlášť na samostatném řádku:

```
slovník
stream
   sekvence bajtů
endstream
```

Bajty mohou být obsaženy i v externím souboru, který se definuje v rámci slovníku streamu. V takovém případě se jakékoliv bajty mezi klíčovými slovy ignorují.

Objekt null

Objekt null existuje v rámci dokumentu PDF pouze jednou a označuje se klíčovým slovem `null`. Jeho hodnota a typ se liší od jakéhokoliv jiného objektu. Pokud je hodnota v záznamu slovníku nastavena na `null`, tak se záznam interpretuje stejně, jako kdyby neexistoval. V případě, že se odkazuje na neexistující objekt (v rámci nepřímých objektů), je tento objekt vyhodnocen jako null.

Nepřímé objekty

Jakékoliv objekty se mohou odkazovat na tzv. nepřímé objekty. Nepřímé objekty vznikají z klasických objektů tak, že se objekt označí identifikátorem a ohraničí se klíčovými slovy `obj` a `endobj`. Identifikátor se skládá z čísla objektu (kladné celé číslo) a čísla generace (nezáporné celé číslo). Čísla dohromady jednoznačně identifikují objekt. Objekt si po celou dobu zachovává stejné čísla, i když se může jeho hodnota měnit. Následující příklad ukazuje validní definici nepřímého objektu, přičemž první číslo představuje číslo objektu a druhé číslo představuje číslo generace:

```
14 3 obj
    \Extraktor
endobj
```

Objekty se mohou na nepřímý objekt odkazovat prostřednictvím nepřímé reference, která se skládá z čísla objektu, čísla generace a klíčového písmena R:

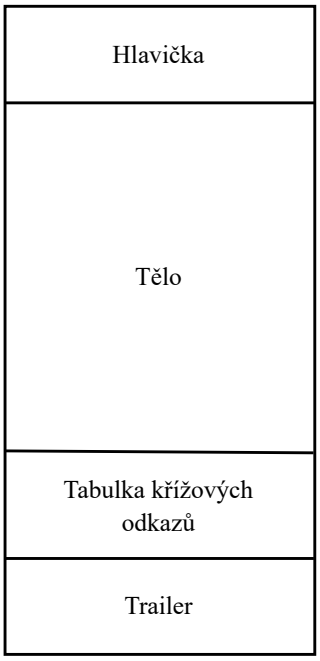
```
14 3 R
```

Nepřímé objekty mohou být uloženy i ve stream objektu. Odkazuje se na ně stejně, avšak jejich definice nezahrnuje klíčová slova `obj` a `endobj`.

2.3 Struktura souboru

Strukturu každého souboru PDF tvoří 4 prvky:

- Jednořádková hlavička, která specifikuje verzi PDF.
- Tělo, které je tvořeno objekty dokumentu PDF.
- Tabulka křížových odkazů, která obsahuje informace o nepřímých objektech.
- Trailer, který udává umístění tabulky křížových odkazů a speciálních objektů v těle souboru.



Obrázek 2.1: Ukázka struktury souboru PDF

Struktura může být později modifikována aktualizacemi, které přidávají další prvky na konec souboru.

Hlavička

První řádek souboru tvoří hlavička, která označuje verzi PDF odpovídající danému souboru. Hlavička souboru PDF verze 1.7 se zapisuje následovně:

```
%PDF-1.7
```

Hlavičku souboru lze přepsat pomocí záznamu `Version` v katalogovém slovníku dokumentu. Verze se poté mění pomocí inkrementální aktualizace. Starší hlavičky mohou být zpracovávány i aplikacemi, které podporují novější verze PDF.

Tělo

Tělo souboru se skládá z posloupnosti nepřímých objektů reprezentujích obsah dokumentu. Objekty představují jednotlivé komponenty, jako je například font nebo stránka. Součástí těla souboru mohou být i streamy, které obsahují posloupnosti nepřímých objektů.

Tabulka křížových odkazů

Tabulka křížových odkazů se skládá z jednotlivých řádků, které reprezentují nepřímé objekty včetně jejich umístění. Proto je možné přistupovat k těmto objektům tak, aby se nemusel procházet celý soubor.

Na začátku tabulka obsahuje pouze jednu sekci křížových odkazů, ke které se přídávají další sekce po každé aktualizaci souboru. Každá sekce začíná řádkem obsahujícím klíčové slovo `xref`. Pod tímto řádkem následuje jedna nebo více podsekcí v libovolném pořadí.

Každá podsekce začíná řádkem, který obsahuje číslo prvního objektu a číslo počtu záznamů v dané podsekci. Pokud se soubor nikdy neaktualizoval, sekce obsahuje pouze jednu podsekci, ve které se objekty číslují od čísla 0. Čísla objektů nesmí mít záznamy ve více podsekcích dané sekce. Následující příklad ukazuje řádek, který představuje podsekci obsahující tři objekty s čísly 12, 13 a 14:

```
12 3
```

Pod tímto řádkem následují samotné záznamy. Záznamy se zapisují na jeden řádek a mají velikost přesně 20 bajtů. Existují dva druhy záznamů: jeden pro využívané objekty a druhý pro objekty, které se smazaly a jsou nyní volné. Oba dva druhy záznamů mají podobný formát zápisu, který se rozlišuje klíčovými písmeny `n` (pro záznam využívaného objektu) a `f` (pro záznam volného objektu). Formát záznamu pro využívané objekty se značí následovně:

```
nnnnnnnnnn ggggg n eol
```

Písmena jsou definována takto:

- `nnnnnnnnnn` je desetimístný bajtový offset udávající počet bajtů od začátku souboru do začátku objektu
- `ggggg` je pětimístné číslo generace
- `n` je klíčové písmeno identifikující využívaný objekt
- `eol` je dvojznaková sekvence ukončení řádku

V případě, že číslo bajtového offsetu nebo číslo generace nemá daný počet míst, je toto číslo doplněno zleva nulami.

Formát záznamu volných objektů je téměř totožný se záznamem využívaných objektů, akorát se používá jiné klíčové písmeno a interpretuje se odlišně první část záznamu:

```
nnnnnnnnnn ggggg f eol
```

Písmena jsou definována následovně:

- `nnnnnnnnnn` je desetimístné číslo následujícího volného objektu
- `ggggg` je pětimístné číslo generace
- `f` je klíčové písmeno identifikující volný objekt
- `eol` je dvojznaková sekvence ukončení řádku

Záznamy volných objektů spolu v tabulce tvoří tzv. spojový seznam, jelikož každý záznam obsahuje číslo následujícího volného objektu. Hlavička spojového seznamu je reprezentována prvním záznamem, který má číslo objektu 0 a číslo generace 65 535. Ocásek spojového seznamu je reprezentovaný posledním záznamem, který odkazuje na první záznam (číslo objektu 0). Poslední záznam tedy uzavírá kruh spojového seznamu. Tabulka může obsahovat i další záznamy volných objektů, které se mohou odkazovat na první objekt a mohou mít číslo generace 65 535. Tyto záznamy však nejsou součástí spojového seznamu.

Všechny objekty (kromě objektu čísla 0) mají v tabulce na začátku číslo generace 0. Číslo se zvýší o 1 pokaždé, když se smaže nepřímý objekt odpovídající danému záznamu. Záznam se poté označí jako volný a přidá se do spojového seznamu volných objektů. Z toho vyplívá, že při každém opětovném použití daného záznamu bude mít záznam nové číslo generace. Jakmile číslo dosáhne hodnoty 65 535, daný záznam nebude znovu používán.

Tabulka křížových odkazů musí obsahovat záznamy pro všechna čísla objektů v rozmezí od 0 až po nejvyšší číslo objektu, i když se některá čísla v daném rozmezí nemusí v tomto souboru vyskytovat. Následující ukázka demonstruje sekci, která obsahuje 2 podsekce:

```
xref
0 4
0000000002 65535 f
0000000019 00000 n
0000000000 00004 f
0000001945 00000 n
8 2
0000008754 65535 n
0000008791 00000 f
```

V první podsekci jsou uvedeny 4 záznamy, které se vztahují k objektům s čísly 0, 1, 2 a 3. Tato podsekce má dva záznamy, které jsou využívány (číslo objektu 1 a 3), a dva záznamy, které jsou volné (číslo objektu 0 a 2). Objekt s číslem 2 byl v této podsekci smazán a další objekt vytvořený se stejným číslem bude mít hodnotu 4 jako hodnotu čísla generace. V druhé podsekci jsou uvedeny 2 záznamy, které se vztahují k objektům s čísly 8 a 9. Tato podsekce má jeden využívaný záznam (číslo objektu 8) a jeden volný záznam (číslo objektu 9).

Trailer

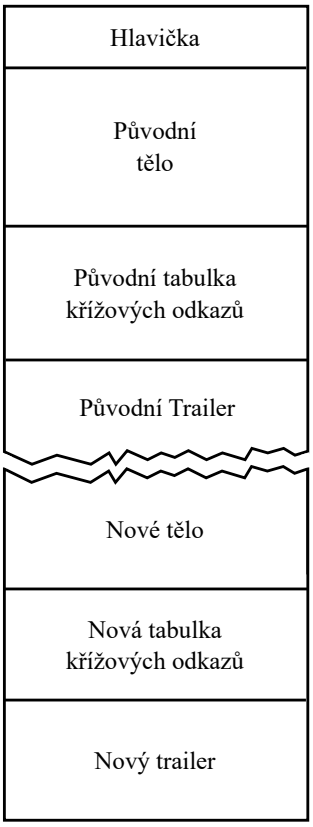
Trailer slouží k tomu, aby aplikace, které čtou PDF, mohly co nejrychleji najít tabulku křížových odkazů a speciální objekty. Jelikož čtou aplikace soubor od konce, bude v tomto směru probíhat i popis traileru.

Konec traileru tvoří značka konce souboru (`%%EOF`), která se nachází na posledním řádku. Před ní se nachází dva řádky, které se vztahují k bajtovému offsetu. První řádek pouze informuje o bajtovém offsetu pomocí klíčového slova `startxref`. Druhý řádek obsahuje hodnotu, která udává bajtový offset od začátku souboru do začátku poslední sekce křížových odkazů. Před těmito řádky se nachází slovník traileru. Slovník je uvozený klíčovým slovem `trailer`, za kterým následuje řada dvojic typu klíč – hodnota, které jsou ohraničené dvojitými úhlovými závorkami. Trailer má tedy následující strukturu:

```
trailer
<< klíč hodnota
   ...
>>
startxref
bajtový_offset
%%EOF
```

Inkrementální aktualizace

Strukturu souboru ovlivňují také inkrementální aktualizace, které přidávají změny na konec souboru, aniž by se musel celý soubor přepisovat. Hlavní výhodou je, že aktualizace rychle ukládají malé změny v rozsáhlém dokumentu. Při každé inkrementální aktualizaci se na konec souboru připojuje nový úsek, který obsahuje 3 části: první částí je upravené tělo obsahující nové nebo změněné objekty, druhou částí je nová sekce křížových odkazů a třetí částí je nový trailer:



Obrázek 2.2: Ukázka struktury aktualizovaného souboru PDF

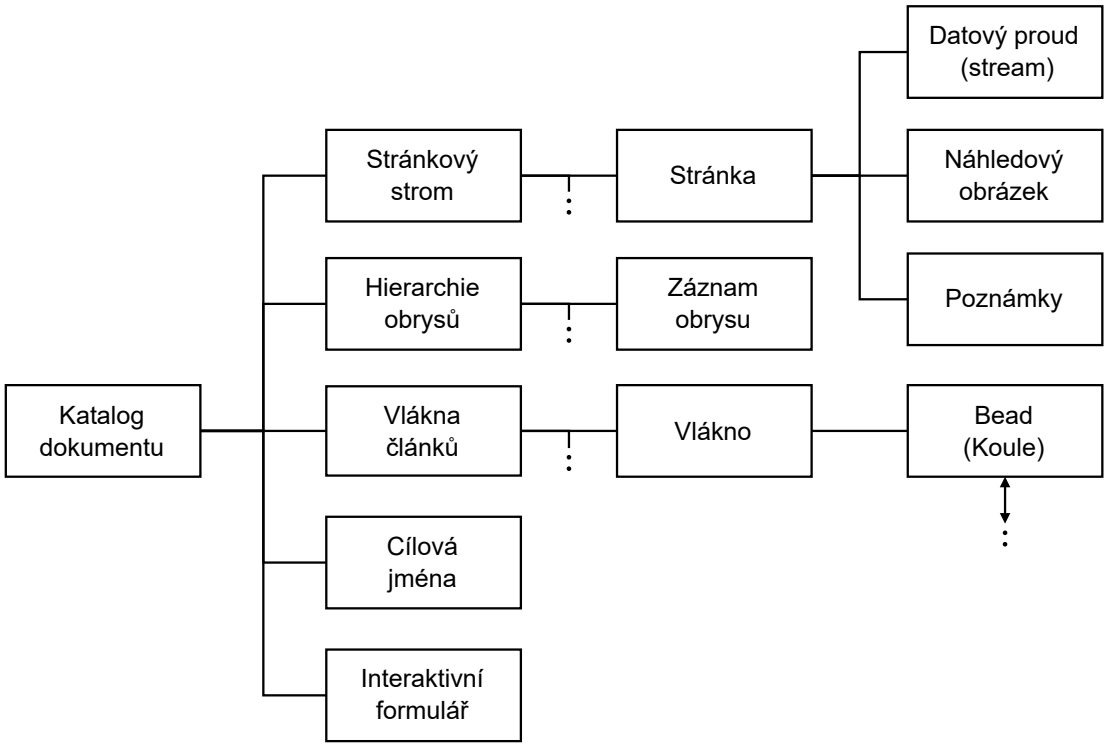
V nové sekci jsou uvedeny záznamy, které popisují změněné, nahrazené nebo smazané objekty. Smazané objekty jsou v souboru ponechány beze změny, akorát jsou jejich záznamy označeny jako volné. Pod novou sekcí se nachází nový trailer, který obsahuje kromě všech záznamů jeho předchůdce také záznam `Prev`, který udává umístění předchozí sekce křížových odkazů.

Jelikož aktualizace přidávají nové úseky na konec souboru, soubor může mít několik kopií objektů se stejným identifikátorem (číslo objektu a číslo generace). Aplikace, které čtou soubor PDF, řeší tuto situaci takovým způsobem, že si vytvářejí vlastní informace o křížových odkazech, aby v souboru přistupovaly k nejnovějším kopiím objektů.

Aktualizace dokáží změnit i verzi PDF. Verze se mění pomocí záznamu `Version` v katalogovém slovníku dokumentu, který přepíše původní verzi zapsanou v hlavičce souboru.

2.4 Struktura dokumentu

Dokument PDF tvoří hierarchie objektů, které se nacházejí v těle souboru. Většina objektů v této hierarchii jsou slovníky. Kořenem této hierarchie je katalogový slovník dokumentu. Každá stránka dokumentu je reprezentována stránkovým objektem, což je slovník, který obsahuje odkazy na obsah stánky a další atributy, jako je například náhledový obrázek. Tyto objekty jsou vzájemně propojeny v hierarchické struktuře, která se označuje jako stránkový strom. Strom je popsán pomocí nepřímého odkazu v katalogu dokumentu. Jednotlivé vztahy v rámci stromu jsou popsány pomocí záznamů slovníku, které obsahují nepřímé odkazy na další slovníky.



Obrázek 2.3: Ukázka struktury dokumentu PDF

Katalog dokumentu

Kořenem hierarchie objektů v dokumentu PDF je katalogový slovník, který se nachází prostřednictvím záznamu `Root` v traileru souboru. Katalog obsahuje odkazy na další objekty, které specifikují různé vlastnosti dokumentu, jako je například jeho obsah. Kromě toho obsahuje také informace o zobrazení dokumentu na obrazovce. Katalog musí obsahovat následující záznamy:

Klíč	Typ objektu	Hodnota
Type	jméno	Typ objektu, který tento slovník popisuje. Pokud se jedná o katalogový slovník, hodnota musí být `Catalog`.
Pages	slovník	Kořenový uzel stránkového stomu, který musí být popsán pomocí nepřímého odkazu.

Tabulka 2.1: Povinné záznamy katalogového slovníku

Volitelné záznamy jsou uvedeny v oficiální dokumentaci formátu PDF [2]. Následující příklad ukazuje validní katalogový objekt:

```
2 0 obj
   << /Type /Catalog
      /Pages 4 0 R
   >>
endobj
```

Stránkový strom

Stránkový strom je hierarchická struktura, přes kterou se přistupuje k jednotlivým stránkám dokumentu. Hlavním úkolem této struktury je definovat pořadí stránek. Stromová struktura umožňuje aplikacím, které čtou PDF, rychle otevřít dokument s několika tisícemi stránek, aniž se na to použilo velké množství paměti. Strom obsahuje 2 typů uzlů: vnitřní uzly (tzv. uzly stránkového stromu) a koncové uzly (tzv. stránkové objekty). Uzly mohou být libovolně uspořádány.

Uzly stránkového stromu

Uzly stránkového stromu jsou základními prvky struktury dokumentu PDF. Uzly vytvářejí hierarchickou strukturu, do které jsou organizovány jednotlivé stránky a objekty. Aby byly uzly platné, musí obsahovat tyto záznamy:

Klíč	Typ objektu	Hodnota
Type	jméno	Typ objektu, který tento slovník popisuje. Pokud se jedná o uzel stránkového stromu, hodnota musí být `Pages`.
Parent	slovník	Bezprostřední rodičovský uzel aktuálního uzlu, který musí být popsán pomocí nepřímého odkazu. Záznam se netýká kořenového uzlu.
Kids	pole	Pole nepřímých odkazů na bezprostřední potomky aktuálního uzlu. Potomky mohou být stránkové objekty nebo jiné uzly stránkového stromu.
Count	číslo (integer)	Počet koncových uzlů (stránkových objektů), které jsou potomky aktuálního uzlu v rámci stránkového stromu.

Tabulka 2.2: Povinné záznamy uzlů stránkového stromu

Struktura uzlů nemusí souviset s logickou strukturou dokumentu. Uzly tedy nemusí nutně reprezentovat jednotlivé části dokumentu PDF, jako jsou například kapitoly nebo sekce. Následující příklad zobrazuje strom pro dokument, který obsahuje jednu stránku:

```
1 0 obj
   << /Type /Pages
      /Kids [ 8 0 R ]
      /Count 1
   >>
endobj

8 0 obj
   << /Type /Page
      ...
   >>
endobj
```

Stránkové objekty

Koncovými uzly stromu jsou stránkové objekty. Objekty jsou reprezentovány slovníky, které specifikují vlastnosti stránek dokumentu. Stránkové objekty musí obsahovat následující záznamy:

Klíč	Typ objektu	Hodnota
Type	jméno	Typ objektu, který tento slovník popisuje. Pokud se jedná o stránkový objekt, hodnota musí být `Page`.
Parent	slovník	Bezprostřední rodičovský uzel aktuálního stránkového objektu, který musí být popsán pomocí nepřímého odkazu.
Resources	slovník	Slovník, který obsahuje prostředky požadované stránkou. Pokud stránka nevyžaduje žádné prostředky, hodnota záznamu by měla být prázdným slovníkem. Pokud je vynechán, tak se prostředky dědí od předka v hierarchii stránkového stromu. *Vlastnost lze dědit.*
MediaBox	obdélník	Obdélník vyjádřený v jednotkách výchozího uživatelského prostoru (v anglickém znění *default user space units*), který definuje hranice fyzického média (např. monitoru), na kterém je stránka určena k zobrazení nebo tisku. *Vlastnost lze dědit.*

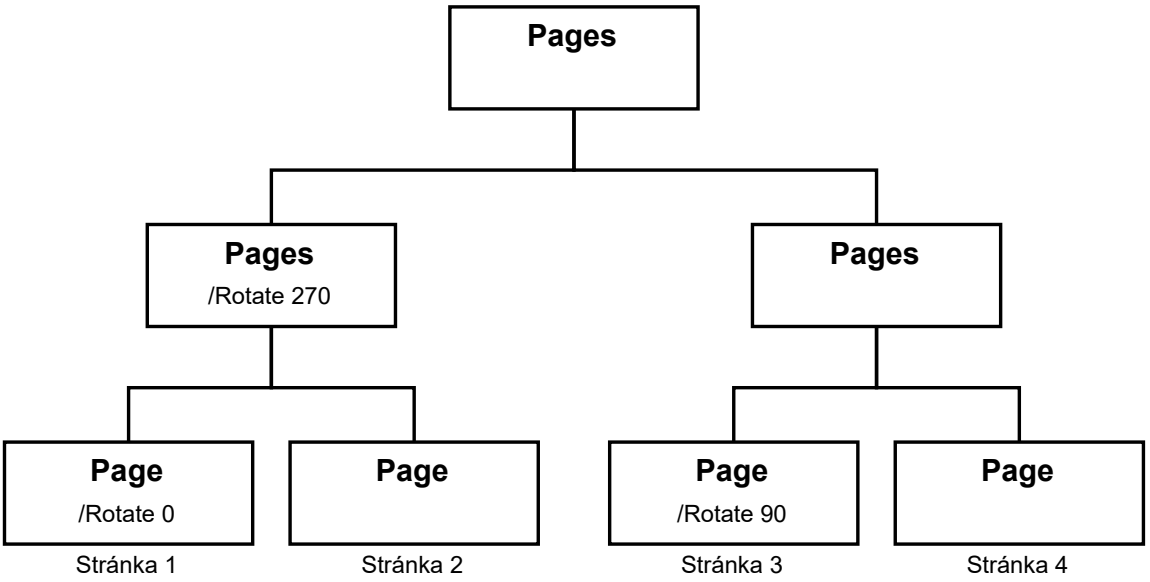
Tabulka 2.3: Povinné záznamy uzlů stránkového stromu

Volitelné záznamy jsou uvedeny v oficiální dokumentaci formátu PDF [2]. Následující příklad demonstruje základní definici stránkového objektu:

```
6 0 obj
   << /Type /Page
      /Parent 2 0 R
      /Resources << /Font << /F2 7 0 R
                             /F7 9 0 R
                             /F9 11 0 R
                          >>
                   /ProcSet [/PDF]
                >>
       /MediaBox [ 0 0 612 792 ]
       ...
   >>
endobj
```

Záznam `MediaBox` specifikuje, že je stránka určena k tisku na papír velikosti „Letter“. Slovník prostředků (specifikovaný jako přímý objekt) uvádí, že stránka používá 3 fonty, které jsou pojmenované „F2“, „F7“ a „F9“.

Některé vlastnosti stránky se mohou dědit. V případě, že jsou tyto vlastnosti vynechány, jsou jejich hodnoty zděděny od nadřazeného uzlu (povinná vlastnost se nahrazuje hodnotou nadřazeného uzlu a volitelná vlastnost se nahrazuje implicitní hodnotou). Vlastnost může být aplikována pro větší sadu stránek tím, že se specifikuje v nadřazeném uzlu a stránky, které mají danou vlastnost sdílet, jsou umístěny jako potomci tohoto uzlu. Následující obrázek zobrazuje dědičnost vlastností na volitelném atributu rotace stránky:



Obrázek 2.4: Ukázka dědičnosti vlastností

Obrázek zobrazuje strom, který má 4 stránky. Stránka 2 je otočená o 270 stupňů, jelikož tuto vlastnost zdědila od svého nadřazeného uzlu. Stránka 1 předpsala tento nadřazený uzel, tudíž se neotáčí (rotace o 0 stupňů). Stránka 3 otočena o 90 stupňů a stránka 4 má implicitní nastavení (neotáčí se).

Slovník jmen

Některé kategorie objektů se mohou v souboru PDF odkazovat podle jména místo odkazu na objekt. Aby se mohlo odkazovat na jméno, je potřeba vytvořit spojitost mezi jménem a objektem. Spojitost se vytváří pomocí slovníku jmen, který se nachází prostřednictvím záznamu `Names` v katalogu dokumentu. Každý záznam určuje kořenový prvek stromu jmen, který definuje jména pro konkrétní kategorii objektů. Všechny záznamy tohoto slovníku jsou volitelné.

2.5 Datové proudy

Datový proud je tzv. „stream objekt“, jehož data se skládají z posloupnosti instrukcí popisujících grafické prvky, které mají být vykresleny na stránce. Kromě toho slouží datový proud také k zahrnutí sekvence instrukcí jako samostatných grafických prvků (např. for-

muláře, vzory nebo určité fonty). Instrukce jsou reprezentovány ve formě objektů, které mají stejnou syntaxi jako zbytek dokumentu. Objekty se narozdíl od zbytku dokumentu interpretují a vykonávájí postupně.

Každá stránka je reprezentována jedním nebo více datovými proudy, které závisí na informacích obsažených ve slovníku zdrojů. Proudy se skládají z objektů, které označují operandy a operátory.

Operandy jsou přímé objekty libovolného datového typu, které musí být uvedeny před daným operátorem. Výjimku tvoří objekty datového proudu, nepřímé objekty a odkazy na objekty, které nesmí být použity jako operandy. Mezi operandy mohou patřit i slovníky, které jsou povolené pouze u některých specifických operátorů.

Operátory se mohou aplikovat v rámci datového proudu. Jedná se o klíčové slova, která specifikují nějakou akci, která má být provedena (např. vykreslení grafického tvaru na stránku). Klíčové slovo operátoru je odlišeno od jmenného objektu tak, že je před operátorem vynechán znak zpětného lomítka.

Většina operátorů se týká kreslení grafických prvků na stránce nebo určování parametrů, které ovlivňují kreslící operace. Vzhledem k tomu, že je extrakce zaměřena na práci s textem, budou následující sekce věnovány operátorům, které jsou spojeny s tvorbou textu.

Textové objekty

Textové objekty se skládají z operátorů, které mohou zobrazovat textové řětězce, přesouvat pozici v textu a nastavovat stav textu. V textovém objektu jsou definovány tři parametry, které se zachovávají pouze v rámci daného objektu:

- T_m neboli textová matice
- T_{lm} neboli textová řádková matice
- T_{rm} neboli textová vykreslovací matice, která je pouze mezivýsledkem, který kombinuje efekty parametrů stavu textu, textové matice (T_m) a aktuální transformační matice.

Textový objekt musí obsahovat následující záznamy:

Operandy	Operátor	Popis
—	**BT**	Operátor, který zahajuje textový objekt. Kromě toho také inicializuje textovou matici T_m a textovou řádkovou matici T_{lm}.
—	**ET**	Operátor, který ukončuje textový objekt a ruší textovou matici.

Tabulka 2.4: Povinné záznamy textového objektu

Textové objekty se nemohou vnořovat, tedy druhý operátor `BT` nesmí být umístěn před první operátor `ET`. Následující příklad ukazuje validní zápis textového objektu:

```
BT
    /F13 14 Tf
    360 720 Td
    (STAY FIT) Tj
ET
```

Operátory v ukázce jsou definovány následovně:

- `Tf` – nastavuje font a jeho velikost
- `Td` – nastavuje počáteční souřadnice vykreslování textu
- `Tj` – vykresluje textový řetězec

Text „STAY FIT“ je umístěný 10 palců od spodu stránky a 5 palců od levého okraje (palec je 1/72 jednotky výchozího uživatelského prostoru, která se používá pro souřadnice v PDF). Text se vykresluje pomocí fontu Helvetica (pod názvem „F13“) o velikosti 14 bodů.

Fonty

Fonty jsou v PDF reprezentovány slovníky, které specifikují jejich typ, PostScriptové jméno, kódování a informace, které se využívají k náhrazení písma ve chvíli, kdy není daný font k dispozici. Fonty mohou být vloženy také jako stream objekty do souboru PDF – lze tedy definovat vlastní fonty.

Ve slovníku jsou typy fontů rozlišeny na základě záznamu `Subtype`. Typy se dělí do dvou kategorii: jednoduché a kompozitní. Jako kompozitní typ je označovaný font Type 0. Ostatní typy jsou označované jako jednoduché. Speciálním typem jsou tzv. „CIDFonty“, které se nepoužívají přímo, ale jako součást fontu Type 0. Následující tabulka uvádí všechny typy fontů, které se vyskytují v PDF:

Typ	Podtyp	Popis
Type 0	**Type0**	Kompozitní font, který je složený z glyfů potomka CIDFontu.[1]
Type 1	**Type1**	Font, který definuje tvary glyfů pomocí technologie Type 1.
	MMType1	*Multi Master font*, rozšíření fontu Type 1, který umožňuje generování různých stylů písma z jednoho fontu.
Type 3	**Type3**	Font, který definuje glyfy pomocí streamů grafických operátorů.
TrueType	**TrueType**	Font, který je založený na formátu TrueType.
CIDFont	**CIDFontType0**	CIDFont, jehož popisy glyfů jsou založeny na technologii písma Type 1.
	CIDFontType2	CIDFont, jehož popisy glyfů jsou založeny na technologii písma TrueType.

Tabulka 2.5: Typy fontů v PDF

[1] Glyf je grafický tvar, který podléhá veškerým grafickým manipulacím, jako je například transformace souřadnic.

V souvislosti s fonty se objevuje pojem „standartních 14 fontů“. Jedná se o standartní fonty typu Type 1, které mají názvy:

Times–Roman	Helvetica	Courier	Symbol
Times–Bold	Helvetica–Bold	Courier–Bold	ZapfDingbats
Times–Italic	Helvetica–Oblique	Courier–Oblique	
Times–BoldItalic	Helvetica–BoldOblique	Courier–BoldOblique	

Tyto fonty (nebo vhodné náhradní fonty se stejnými metrikami) musí být vždy dostupné pro aplikaci, která čte PDF. Ostatní fonty mohou být nahrazeny.

Kapitola 3

Knihovny pro čtení a zpracování dokumentů PDF

Kapitola zkoumá placené i neplacené knihovny pro čtení a zpracování dokumentů PDF, které jsou implementované v programovacím jazyce Java. Nejprve se analyzují knihovny s otevřeným zdrojovým kódem (tvz. „open-source“ knihovny), které jsou dostupné zdarma. Následně se rozebírají knihovny, které mají placenou licenci. Veškeré informace v této kapitole byly získány z článku „Java PDF Library Comparison (Free & Paid Tools)“ od společnosti IronPDF [6]. Je tedy důležité brát v úvahu, že poskytnuté informace mohou být tendenční a mohou výrazně zvýhodňovat tuto knihovnu.

iTextPDF

iTextPDF je populární knihovna s otevřeným zdrojovým kódem, která umožňuje vývojářům vytvářet, manipulovat a extrahovat data z dokumentů PDF. Knihovna nabízí funkce pro práci s textem, obrázky, tabulkami a různými dalšími grafickými prvky. Navíc poskytuje podporu digitálních podpisů, šifrování a dalších bezpečnostních prvků. Knihovna je dostupná v placené i neplacené verzi a má velkou komunitu aktivních vývojářů, kteří se neustále podílejí na jejím vývoji.

Výhody:

- **Pokročilá práce s formuláři:** Knihovna nabízí funkce, které dokáží kvalitně pracovat s formuláři (včetně jejich vyplňování, extrakce a ověřování zadaných vstupů).
- **Možné přizpůsobení:** Knihovna poskytuje řadu možností pro přizpůsobení dokumentu takovým způsobem, aby vyhovoval potřebám uživatele.

Nevýhody:

- **Složité použití:** Knihova je velmi složitá pro začínající vývojáře.
- **Rozsáhlá knihovna:** Knihovna má širokou škálu funkcí, což znamená, že může být poněkud rozsáhlá (v některých případech to může negativně ovlivnit výkon a využití paměti) [6].
- **Komerční licence:** Ačkoliv je knihovna open-source, některé pokročilé funkce vyžadují komeční licenci.

Apache PDFBox

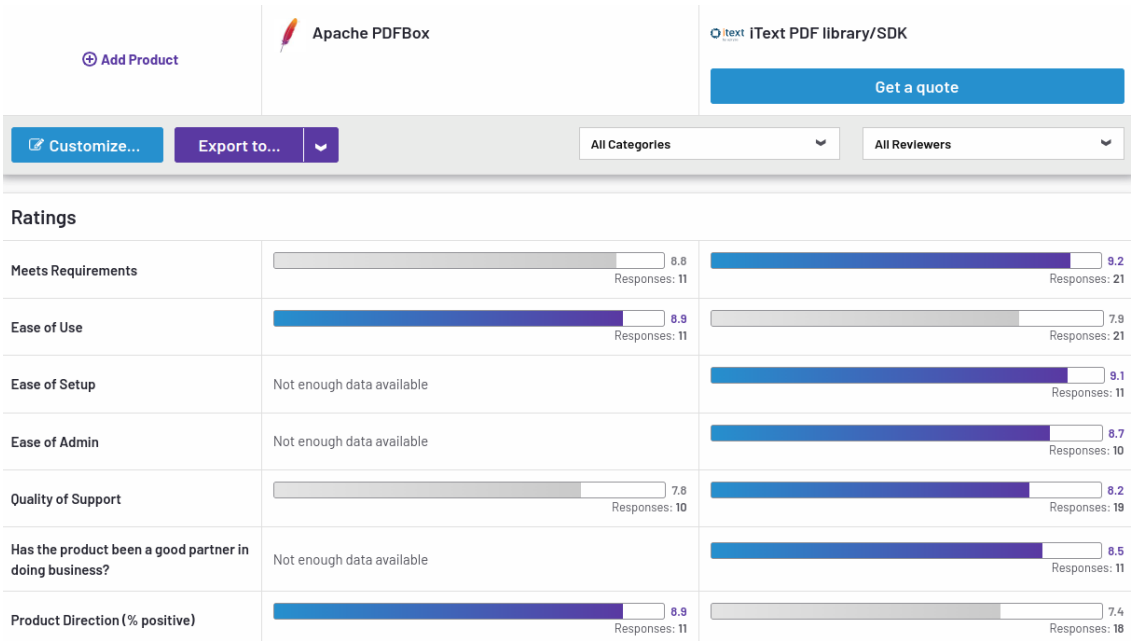
PDFBox je knihovna s otevřeným zdrojovým kódem, která se používá pro tvorbu, manipulaci a extrakci dat ze stránek dokumentu PDF. Knihovna poskytuje širokou škálu funkcí pro manipulaci s formátem PDF. Mezi takové manipulace může patřit například vytváření nových dokumentů, úprava stávajících, extrakce textu a obrázků, přidávání poznámek a záložek, slučování a dělení souborů a také šifrování a dešifrování obsahu. PDFBox je postavený na knihovně Portable Document Format (zkráceně PDF) vyvinuté společností Apache, která je rovněž napsána v jazyce Java a distribuována pod licencí Apache Licence 2.0.

Výhody:

- **Dostupnost:** Jelikož je knihovna volně dostupná, tak je vhodná pro vývojáře s omezeným rozpočtem.
- **Snadné použití:** Knihovna má poměrně jednoduché rozhraní (tzv. „API“), které je snadno použitelné i pro nové vývojáře.

Nevýhody:

- **Problémy s výkonem:** Knihova může být pomalá při zpracování velkých dokumentů nebo při provádění složitých operací [6].
- **Slabá dokumentace:** I když má knihovna vstřícnou a nápomocnou komunitu, její oficiální dokumentace není v některých oblastech dostatečně zpracovaná.
- **Omezená podpora:** Ačkoliv knihovna umožňuje používat formát PDF 2.0, její podpora nových funkcí zavedených v této verzi je omezená. Kromě toho má také omezenou podporu pro pokročilé funkce.



Obrázek 3.1: Porovnání hodnocení knihoven PDFBox a iText (podle vývojářů) [5]

Aspose.PDF

Aspose.PDF je knihovna s placenou licencí, která umožňuje vývojářům vytvářet, manipulovat a konvertovat dokumenty PDF v aplikacích vytvořených v jazyce Java. Knihovna nabízí rozsáhlé spektrum funkcí pro práci s PDF soubory, jako je například přidávání či odebírání stránek, úprava textu a obrázků, vyplňování formulářů, slučování a rozdělování dokumentů nebo přidávání bezpečnostních prvků. Aspose.PDF poskytuje jednoduché a intuitivní rozhraní pro tisk, které se snadno integruje do aplikací implementovaných v jazyce Java.

Výhody:

- **Multiplatformní knihovna:** Knihovna se může používat na různých operačních systémech, jako jsou Windows, macOS a Linux.
- **Pravidelné aktualizace:** Knihovna se pravidelně aktualizuje kvůli novému vylepšení a opravě chyb, takže mají vývojáři přístup k nejnovějším funkcím souvisejím s PDF.
- **Zákaznická podpora:** Společnost Aspose poskytuje kvalitní zákaznickou podporu prostřednictvím svých fór a emailu. Navíc poskytují vývojářům 30denní bezplatné zkušební období, které jim umožňuje zdarma vyzkoušet jejich knihovnu.

Nevýhody:

- **Vysoká paměťová náročnost:** Knihovna vyžaduje významné množství paměti pro zpracování dokumentů PDF ve srovnání s jinými knihovnami, což může představovat závažný problémem pro aplikace, které běží na systémech s omezeným množstvím paměti [6].
- **Omezená podpora ostatních formátů:** Ačkoliv knihovna podporuje různé vstupní a výstupní formáty (např. HTML nebo XML), její podpora ostatních formátů, které nejsou PDF, může být omezená ve srovnání s ostatními alternativami.
- **Problémy s kompatibilitou:** Knihovna nemusí být vždy kompatibilní se staršími verzemi programovacího jazyka Java. Vývojáři si tak buď musí aktualizovat verzi Javy nebo musí použít jinou alternativní knihovnu.

IronPDF

IronPDF je knihovna s placenou licencí, která umožňuje vývojářům vytvářet, upravovat a manipulovat s dokumenty PDF pomocí programovacího jazyka Java. Knihovna disponuje bohatým spektrem funkcionalit, které zahrnují vytváření nového dokumentu, slučování různých PDF souborů, přidávání textu a obrázků, extrahování dat a také převádění HTML na PDF. IronPDF je postavený na webovém prohlížeči Chromium, který mu umožňuje vykreslovat HTML a CSS do formátu PDF s vysokou přesností. To znamená, že vývojáři mohou použít tuto knihovnu k převodu složitých webových stránek s dynamickým obsahem a interaktivními prvky na dokumenty formátu PDF. Knihovna poskytuje rozsáhlou dokumentaci a podporu, která zahrnuje tutoriály, referenční API materiály a bázi znalostí (anglický výraz *knowledge base*).

Výhody:

- **Rozsáhlejší možnosti:** Knihovna nabízí rozsáhlejší spektrum funkcí pro tvorbu, manipulaci a úpravy dokumentů PDF oproti ostatním alternativám, včetně podpory různých PDF standardů a schopnosti převést HTML na PDF.
- **Vykreslování HTML do PDF:** Knihovna využívá jádro webového prohlížeče Chromium (tzv. „prohlížečový engine“), který jí umožňuje provádět vykreslování HTML a CSS do PDF s vysokou přesností, což je užitečné pro konverzi webových stránek do PDF.
- **Dokumentace a podpora:** IronPDF poskytuje rozsáhlou dokumentaci a podporu, které pomáhají vývojářům začít s používáním této knihovny a řešením případných problémů, se kterými se mohou setkat. Navíc nabízí vývojářům 30denní bezplatné zkušební období, které jim umožňuje vyzkoušet knihovnu zdarma se všemi funkcemi, které jsou v komerční licenci.

Nevýhody:

- **Komerční licence:** Knihovna vyžaduje placenou licenci k jejímu použití. To může představovat potenciální překážku pro vývojáře nebo organizace s omezeným rozpočtem.

Kapitola 4

Návrh architektury

Kapitola návrhu architektury se zaobírá částmi programu, které musely být navrženy před samotnou implementací. Na začátku kapitoly se provádí podrobný rozbor požadavků extrakce (sekce 4.1). Podle rozboru požadavků se navrhuje prostředek komunikace – jazyk, který má za úkol programu sdělit požadavky uživatele (sekce 4.2). Následně se navrhuje architektura programu (sekce 4.3). Některé části návrhu se rozdělují do samostatných sekcí, jelikož jsou značně rozsáhlé (sekce 4.4) nebo se vymykají základnímu návrhu architektury programu (sekce 4.5).

4.1 Požadavky zadavatele

Jedním z klíčových faktorů extrakce je seznámení se s požadavky zadavatele. Po prozkoumání těchto požadavků se určí, jakým způsobem bude extrakce probíhat. Zadavatelem této práce je oddělení kardiologie ve fakultní nemocnici Brno.

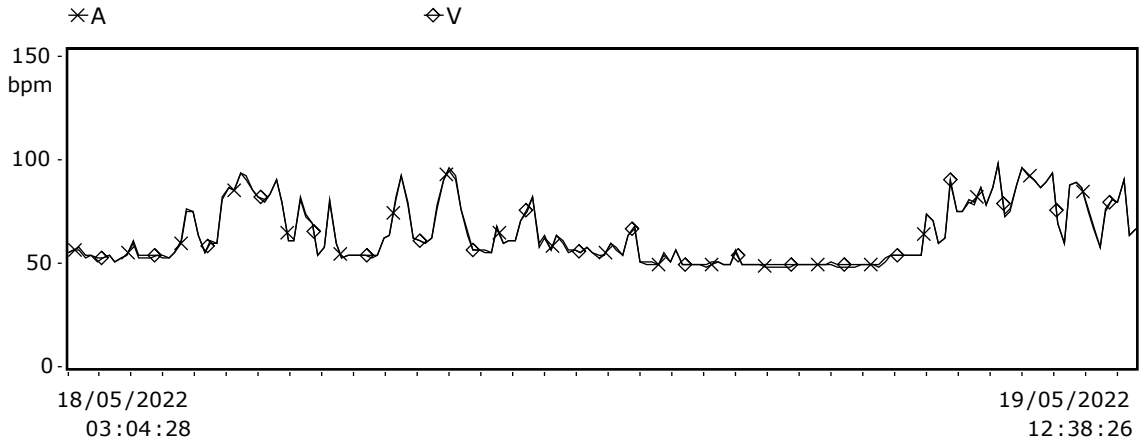
Oddělení kardiologie spolupracuje s firmami, které vyrábí srdeční stimulátory. Stimulátory při kontrole vytvářejí záznamy o stavu pacienta ve formátu PDF. Komplikace ale nastává v mnohotvárnosti formátu záznamu. Formát záznamu je odlišný nejen v rámci firem, ale i v rámci jedné firmy podle verze a typu zařízení. Proto se proces extrakce navrhl co nejobecněji, aby mohl probíhat nezávisle na formátu záznamu. Záznamy bývají značně rozsáhlé a mívají velké množství různě důležitých informací. S informacemi se dále nepracuje. Požadavkem zadavatele je tedy extrakce informací ze záznamů a jejich přehledné uložení.

Informace o stavu pacienta a stavu srdečního stimulátoru jsou zaznamenávány do tabulek a grafů. Tabulky jsou rozděleny do několika sloupců, přičemž první sloupec reprezentuje stav. Ostatní sloupce reprezentují hodnoty stavu podle parametrů měření. Stavy se často sdružují do skupin. Skupina se váže k určité sledované oblasti pacienta. Název skupiny je vyznačený tučně a stavy spadající pod tuto skupinu jsou mírně odsazeny od jejího názvu. Tabulky nemají viditelné ohraničení a jejich položky zpravidla obsahují text.

Leads Data	**Implant** N/R N/R N/R	**Previous Session**	**Most Recent**
Atrial			
Intrinsic Amplitude	N/R mV	6.8 mV	6.8 mV
Pace Impedance	N/R Ω	762 Ω	622 Ω
Pace Threshold	N/R V @ N/R ms	Auto 0.6 V @ 0.4 ms	Auto 0.7 V @ 0.4 ms
Ventricular			
Intrinsic Amplitude	N/R mV	>25.0 mV	19.9 mV @ 55 min^{-1}

Obrázek 4.1: Ukázka tabulky, která má šedě vyznačené záhlaví s tučnými nadpisy.

Grafy jsou vykreslovány jako vektor, který reprezentuje záznam pacientova stavu v čase. Většinou mají vyznačené osy a okolo sebe mívají jednoduchou textovou legendu.



Obrázek 4.2: Ukázka grafu srdečního rytmu během 24 hodin

Z požadavků zadavatele vyplývá, že je nutné rozdělit extrakci na 2 typy: extrakci textu a extrakci obrázku. Extrakce textu má ukládat extrahované hodnoty stavů z tabulek do souboru. Soubor má mít tyto hodnoty přehledně uložené (např. formou tabulky). Extrakce obrázku má ukládat extrahované grafy samostatně jako obrázky. Celý výstup pak má být uložený v rámci jednoho cílového adresáře.

4.2 Návrh pseudojazyka Barty

Základním úkolem extrakce je zjištění konkrétních požadavků od uživatele. Požadavky musí mít pevně danou strukturu a musí se řídit jasně definovanými pravidly – toho lze dosáhnout vytvořením vlastního jazyka. Jelikož se jazyk skládá pouze z několika jednoduchých pravidel, jedná se spíše o takzvaný „pseudojazyk“. Pseudojazyk je citlivý na velikost písmen (*case-sensitive*) a není citlivý na počet mezer či tabulátorů (*whitespace-insensitive*). Pseudojazyk má za úkol sdělit:

- Jaké položky mají být extrahované?
- Které hodnoty položek mají být extrahované?
- Jak se tyto hodnoty mají po extrakci jmenovat?

Uživatel zapisuje informace prostřednictvím příkazů do příkazového souboru. Příkazový soubor je skupina příkazů, která slouží určení konkrétních informací pro extrakci. Soubor může být uložen jako textový dokument (přípona .txt) nebo jako soubor Barty (přípona .barty). Příkaz je jednořádková kompozice několika seřazených parametrů. Každý parametr má svůj oddělovač, který ho jednoznačně identifikuje. Parametry se dělí na 2 typy: textové a číselné. Textové parametry jsou uvedeny v uvozovkách. Číselné parametry mít uvozovky nesmí. Pokud se v názvu textového parametru vyskytuje uvozovka, píše se před ní zpětné lomítko (jedná se o tzv. „řídící sekvenci“). Podle parametrů se určí, zda se jedná o extrakci textu či extrakci obrázku. Následující výpis popisuje formální definici gramatiky pseudojazyka Barty pomocí rozvinuté Backusovy–Naurovy formy (EBNF):

```
(* Formální definice gramatiky pseudojazyka Barty *)

písmeno = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J"
        | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T"
        | "U" | "V" | "W" | "X" | "Y" | "Z" | "a" | "b" | "c" | "d"
        | "e" | "f" | "g" | "h" | "i" | "j" | "k" | "l" | "m" | "n"
        | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x"
        | "y" | "z" ;
číslice = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" ;
speciální-znak = "!" | '"' | "#" | "$" | "%" | "&" | "'" | "(" | ")"
               | "*" | "+" | "," | "-" | "." | "/" | ":" | ";" | "<"
               | "=" | ">" | "?" | "@" | "[" | "\" | "]" | "^" | "_"
               | "`" | "{" | "|" | "}" | "~" ;
mezera = { " " | "\t" } ;

text = ( písmeno | číslice | speciální-znak | mezera ) ,
       { písmeno | číslice | speciální-znak | mezera } ;

název = mezera , '"' , text , '"' , mezera ;
jméno = název ;
číslo = číslice , { číslice } ;

odkud = název ;
přes = název ;
kam = název ;

upřesňující-parametry = [ ">" , odkud ]
                      , { "-" , přes }
                      , [ "<" , kam ] ;
výskyt = číslo ;
proporce-obrázku = "{" , číslo , "," , [ " " ] , číslo , "}"
                 | "{" , číslo , "," , [ " " ] , číslo , ","
                 , [ " " ] , číslo , "," , [ " " ] , číslo , "}" ;

povinné-parametry = název , ":" , jméno , "[" , číslo , "]" ;
volitelné-parametry = upřesňující-parametry , [ výskyt ]
                    , [ proporce-obrázku ] ;

příkaz = povinné-parametry , volitelné-parametry ;
příkazový-soubor = { příkaz , "\n" } , [ příkaz ] ;
```

Výpis 4.1: Formální definice gramatiky pseudojazyka Barty v EBNF[1].

[1] EBNF je zapsána ve standardu ISO.

Z formální definice gramatiky vyplívá, že příkaz musí obsahovat povinné parametry. Povinnými parametry jsou:

- Parametr jméno
- Parametr název
- Parametr číslo

Parametr jméno označuje názvy sloupců extrahovaných hodnot ve výsledné tabulce. Pokud se jedná o extrakci obrázku, parametr slouží k pojmenování extrahovaného obrázku. Parametr název určuje stav, jehož hodnota má být extrahována (nezávisle na typu extrakce). Implicitně se vybírá první nalezený stav. Parametr číslo udává typ extrakce. V případě extrakce textu se zjišťuje i číslo sloupce, ze kterého má být daná hodnota extrahována. Ve chvíli, kdy nejsou tyto parametry uvedeny, nastane chyba. Syntaxe povinných parametrů vypadá následovně:

```
"jméno" : "název" [číslo]
```

Mezi jednotlivými parametry se nachází oddělovače. Oddělovače jsou reprezentovány znakem nebo závorkami. Znak dvojtečky reprezentuje oddělovač parametru název. Hranaté závorky reprezentují oddělovač parametru číslo. Parametr jméno jako jediný oddělovač nemá. Hodnotou parametru jméno a parametru název je libovolný text. Hodnotou parametru číslo je celé nezáporné číslo. Je-li číslo 0 hodnotou parametru, provede se extrakce obrázku. Je-li hodnotou parametru libovolné kladné číslo, provede se extrakce textu. V tomto případě navíc číslo interpretuje sloupec, jehož hodnota se má extrahovat. Pokud je hodnota čísla větší než 5, automaticky se při extrakci zobrazí varování (velmi pravděpodobně se jedná o chybu).

Jestliže se v dokumentu vyskytuje stav na vícero místech, nelze další výskyty tohoto stavu extrahovat. Proto byly vytvořeny takzvané „upřesňující parametry“, které mají za úkol specifikovat extrakci tak, aby se z dokumentu vybral správný stav. Parametry fungují jako zarážky (horní a dolní), nebo jako kontrola výskytu položek v okolí. Parametr odkud určuje počátek vyhledávání položky. Parametr přes udává konkrétní body, přes které musí vyhledávání projít. Parametr může být použit v příkazu vícekrát za sebou. Parametr kam určuje konec vyhledávání položky. Upřesňující parametry se vyskytují vždy za povinnými parametry. Stejně jako u povinných parametrů se nesmí prohazovat jejich pořadí. Syntaxe upřesňujících parametrů vypadá následovně:

```
> "odkud" - "přes" < "kam"
```

Upřesňující parametry mají také své oddělovače. Znak větší než reprezentuje oddělovač parametru odkud. Znak pomlčky reprezentuje oddělovač parametru přes. Znak menší než reprezentuje oddělovač parametru kam. Hodnotou parametrů je libovolný text nacházející se v dokumentu (nemusí jít o položku). Upřešňující parametry nejsou povinné a nejsou na sebe navzájem nijak vázané.

Nezřídka může nastat situace, kdy se daná tabulka nebo obrázek objeví v dokumentu vícekrát. Vyhledávání přes upřesňující parametry by bylo v tomto případě neefektivní a zbytečně zdlouhavé. Proto se zavedl parametr výskyt. Tento parametr se uvádí po upřesňujících parametrech (pokud existují). Parametr má jednoduchou syntaxi:

```
(výskyt)
```

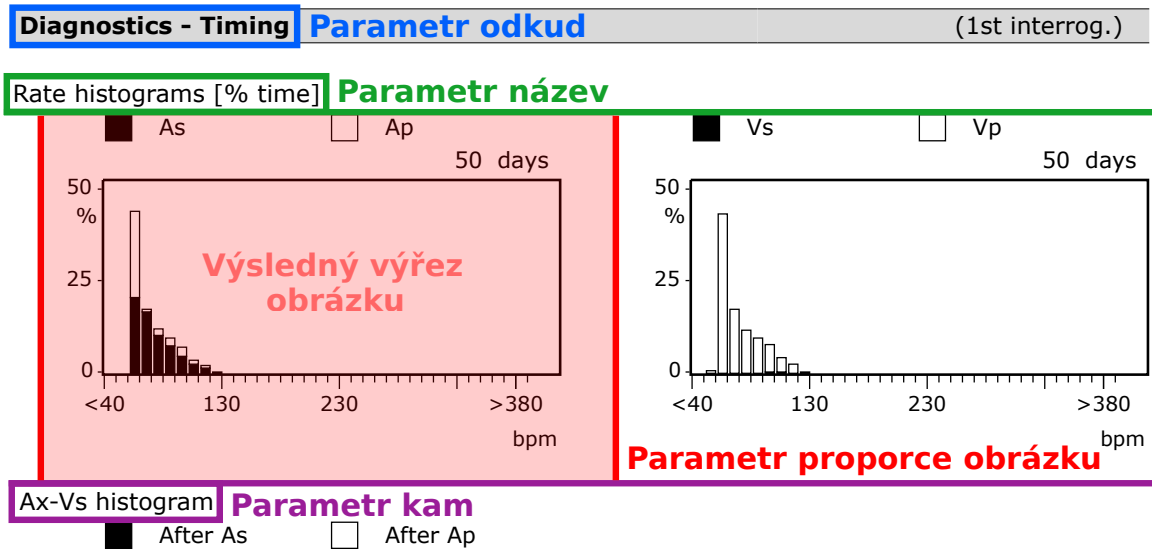
Parametr výskyt má jako svůj oddělovač kulaté závorky. Hodnotou parametru je libovolné přirozené číslo, které má určit počet výskytů hodnot odpovídajících vzoru parametrů daného příkazu. Vzor parametrů příkazu je podrobněji popsán v sekci 4.5.

V případě, že byla zvolena extrakce obrázku, je nutné získat dvě dvojice souřadnic (x, y). První dvojice souřadnic (x_1, y_1) představuje levý horní okraj obrázku. Souřadnice x_1 reprezentuje začátek stránky (tedy $x_1 = 0$) a souřadnice y_1 se získá ze souřadnic spodního okraje parametru název. Druhá dvojice souřadnic (x_2, y_2) představuje pravý spodní okraj obrázku. Souřadnice x_2 reprezentuje šířku stránky a souřadnice y_2 se získá ze souřadnic horního okraje parametru kam. Pokud v příkazu není uveden parametr kam, jako souřadnice y_2 se zvolí výška stránky. Oběma y souřadnicím bylo nastaveno vhodné odsazení od okraje textu tak, aby tento text nebyl v obrázku zachycen.

Jelikož je zachycení obrázku značně omezené, vytvořil se parametr proporce obrázku. Parametr má za úkol zúžit původní výběr obrázku (tzv. „oříznutí obrázku“). Parametr se skládá ze 2 (šířka, výška) nebo 4 (vlevo, vpravo, nahoře, dole) hodnot. Parametr není povinný a uvádí se vždy na konci příkazu. Syntaxe parametru je následující:

```
{výška, šířka} nebo {vlevo, vpravo, nahoře, dole}
```

Parametr proporce obrázku má jako svůj oddělovač složené závorky. Hodnoty parametru jsou celé čísla od 0 do 100, která jsou odděleny čárkou. Parametr se dvěma hodnotami reprezentuje procento výběru (hodnota 100 představuje původní výběr). S klesající hodnotou se výběr úží směrem do středu. Parametr se čtyřmi hodnotami reprezentuje proporce obrázku. Obrázek má horizontální a vertikální proporce. Proporce se vymezují dvěma hodnotami, které se nesmí křížit – hodnota vlevo musí být menší než hodnota vpravo a zároveň hodnota nahoře musí být menší než hodnota dole. Hodnota vlevo začíná na 0 a s rostoucí hodnotou se výběr posouvá doprava. Hodnota vpravo začíná na 100 a s klesající hodnotou se výběr posouvá doleva. Stejný princip funguje i u vertikálních proporcí.



Obrázek 4.3: Ukázka výběru obrázku příkazem:
`"Rate histogram": "Rate histograms [%time]" [0] > "Diagnostics - Timing" < "Ax-Vs histogram" {15, 50, 0, 100}`

Aplikace na případné chyby v příkazovém souboru upozorní v průběhu procesu extrakce.

4.3 Návrh kostry programu

Návrh kostry představuje esenciální část návrhu architektury programu. Dobrý návrh totiž umožnuje lépe pochopit podstatu jednotlivých částí programu. Pochopení koncepce návrhu následně vede ke kvalitnější implementaci programu. Návrh kostry programu tvoří:

- Syntaktická a sémantická kontrola argumentů
- Syntaktická a sémantická kontrola příkazového souboru
- Skenování vstupního adresáře a kontrola, zda adresář obsahuje záznamy formátu PDF
- Vytváření složek ve výstupním adresáři
- Proces extrakce a proces ladění
- Vytváření výstupních dokumentů

Kompozice z uvedených částí tvoří základní kostru programu. Některé části návrhu jsou značně rozsáhlé, proto jsou podrobněji rozebírány v dalších sekcích. V případě selhání jakékoliv části se chyba zapíše do výstupního dokumentu, který slouží k zaznamenávání chyb.

U většiny programů se nejprve kontrolují argumenty programu. Extrakční program byl původně navržený jako konzolová aplikace. Postupným vývojem se z něj stala desktopová aplikace s grafickým uživatelským rozhraním, proto se většina syntaktické a sémantické kontroly argumentů odehrává před samotným spuštěním procesu extrakce. Z tohoto důvodu nejsou v této sekci kontroly argumentů více rozebírány (jejich analýza probíhá v sekci 6.1).

Po kontrole argumentů probíhá kontrola příkazového souboru. Ověří se, zda příkazy v souboru odpovídají formální definici gramatiky pseudojazyka (tzv. „syntaktická kontrola“) a následně se ověří i základní sémantika příkazů (např. ověření sémantiky parametru proporce obrázku). Pokud obě ověření proběhnou úspěšně, pokračuje se další částí.

V další části se skenuje vstupní adresář se záznamy ve formátu PDF. Při povolení rekurzivního zanořování se skenují i podadresáře vstupního adresáře. Výsledkem skenování je uložení cest k těmto záznamům. Cesty k souborům jiného formátu nebudou uloženy. Jestliže je vstupní adresář prázdný, extrakce se ukončí s chybovým hlášením.

Před procesem extrakce se vytvoří složky ve výstupním adresáři. Pokud již složky ve výstupním adresáři existují, budou vymazány (včetně jejich obsahu) a nahrazeny nově vytvořenými složkami. Složky slouží k ukládání extrahovaných obrázků či k ukládání výstupu ladění.

Následuje proces extrakce, který má za úkol extrahovat informace ze záznamů pomocí instrukcí z příkazového souboru. Extrahované informace jsou uloženy do výstupních dokumentů. Ve chvíli, kdy je zapnutý režim ladění, se kromě procesu extrakce provádí i proces ladění. Proces ladění vytváří obrázky stránek záznamu, ze kterých proběhla extrakce. Kromě obrázku se vytváří v rámci každého záznamu textový soubor ukládající počet nalezených hodnot odpovídajících vzoru parametrů a počet nalezených hodnot parametru název.

Na konec se vytvoří výstupní dokumenty, do kterých se zapíší data získané z extrakce. Jedná se především o soubor formátu CSV, který reprezentuje přehlednou tabulku extrahovaných hodnot z tabulek záznamů. Dalším dokumentem je textový soubor se záznamy chyb, které se vyskytly během extrakce. Tento soubor se vytvoří právě tehdy, když při extrakci nastane chyba nebo varování.

4.4 Návrh procesu extrakce

Proces extrakce probíhá u každého záznamu, jehož cesta byla uložena v rámci skenování vstupního adresáře. Záznam se načte a získá se z něj veškerý text včetně jeho souřadnic. Podle těchto souřadnic se text záznamu seřadí. Nejprve proběhne řazení podle souřadnice y, kdy se text seřadí do řádků. Následně proběhne řazení těchto řádků podle souřadnice x.

Další fází procesu extrakce je takzvané „slepování slov“. Ačkoliv je text seřazený, nelze určit, který text je hodnota stavu a který text je stav (hodnoty stavu se hledají podle čísla sloupce). Z tohoto důvodu se vytvořilo slepování slov, které má za úkol spojit text sousloví a text vět mezerami. Pro správné slepování slov je nutné určit vhodnou mezeru mezi dvěma slovy. Mezera bývá v různých záznamech velmi odlišná a nelze ji jednoznačně určit, proto mezeru stanovuje uživatel.

Po procesu slepování slov dochází k extrakci. Před zpracováním prvního příkazu extrakce se vytvářejí podsložky ve výstupních složkách. Podsložky mají název daného záznamu. Do podsložek se ukládají extrahované obrázky a ladící výstupy. Extrakce probíhá na základě zpracování a následného vyhledávání jednotlivých hodnot odpovídajících vzoru parametrů příkazu. Hodnoty představují část záznamu. Vyhledávání u extrakce obrázku probíhá stejně jako vyhledávání u extrakce textu. Vyhledávání probíhá opakovaně z místa posledního nálezu ve chvíli, kdy je nataven parametr počtu výskytu. Po zpracování všech příkazů se extrahované hodnoty uloží a celý proces se opakuje, dokud nejsou extrahovány všechny záznamy.

4.5 Návrh režimu ladění

Kromě procesu extrakce může v programu probíhat i proces ladění. Cílem ladění je ukázat větší podrobnosti, které vedly k výběru konkrétní hodnoty stavu či obrázku. Výstup ladění obsahuje složky s názvy extrahovaných záznamů, které obsahují obrázky extrakce a textový soubor.

Obrázky reprezentují stránku záznamu, ze které byla daná informace extrahována. Obrázky jsou vytvářeny nezávisle na typu extrakce. Zachycují tedy nejen extrakci hodnoty stavu, ale i extrakci obrázku. Extrahovaná část stránky je v obrázku označena tučně červeným obdélníkem. V případě extrakce textu je navíc označena červeně i vybraná buňka v tabulce, aby nedocházelo k nesprávnému určení vybrané hodnoty stavu.

Mode		DDDR
Basic rate/UTR [bpm]		55/130
	A	V
Pulse amplitude [V]	1.6	1.4
Pulse width [ms]	0.4	0.4
Expected ERI		12 Y. 7 Mo.
Battery status		OK
Remaining battery capacity [%]		100

Obrázek 4.4: Ukázka části obrázku extrahované hodnoty z výstupu ladění

Textový soubor slouží k zaznamenávání podrobnějších informací o extrakci. Soubor se váže k jednomu záznamu a skládá se z úvodní hlavičky a výpisů. V úvodní hlavičce jsou vypsány základní informace o extrakci (např. jméno záznamu nebo jméno příkazového souboru). Pod

hlavičkou se nacházejí výpisy, které mají za úkol detailněji informovat o průběhu extrakce. Každý výpis se pojí s konkrétním příkazem a obsahuje:

- Číslo příkazu
- Název příkazu
- Parametry příkazu
- Počet výskytů hodnot odpovídajících vzoru parametrů příkazu
- Počet výskytů parametru název
- Typ extrakce

Číslo příkazu udává číslo řádku, na kterém je příkaz zaznamenaný v příkazovém souboru. Toto číslo se zapisuje do hranatých závorek. Název příkazu je pojmenovaný podle parametru název. Název slouží k označení výpisu. Parametry příkazu jsou obsaženy ve výpisu z důvodu uvedení celého příkazu. Parametry se uvádějí v závorce podle postupu hledání. Počet výskytů hodnot poskytuje informaci o nalezení částí textů, které odpovídají vzoru parametrů příkazu. Vzor je následující:

1. Hledá se parametr od
2. Hledá se parametr přes (při vícero parametrech se hledá zleva)
3. Hledá se parametr název
4. Hledá se parametr kam

Vzor se upravuje podle přítomnosti volitelných parametrů. Počet výskytů parametru název udává počet nalezených pojmů, které se shodují s hodnotou parametru název. Části výpisu s počty výskytů mají informovat uživatele o dalších možných přítomnostech daného stavu (parametr název) či hodnot odpovídajících vzoru parametrů příkazu, o kterých uživatel nemusel vědět. Poslední částí výpisu je typ extrakce, který oznamuje, zda se jedná o extrakci textu či obrázku.

```
----------------------------------------------------------------------
Debug log "debug.txt"
    Created: 29/03/2023 14:55:30
    PDF File: record.pdf (Path: /home/user/Extractor/Input/)
    Command File: command.barty (Path: /home/user/Extractor/Command/)
----------------------------------------------------------------------

[1] Command name "Amplitude" (Title: "Pulse amplitude", Number: "1")
        Phrase occurrence: 2
        Title occurrence: 6
        Type: Text search
```

Výpis 4.2: Ukázka textového souboru s úvodní hlavičkou a výpisem

Kapitola 5

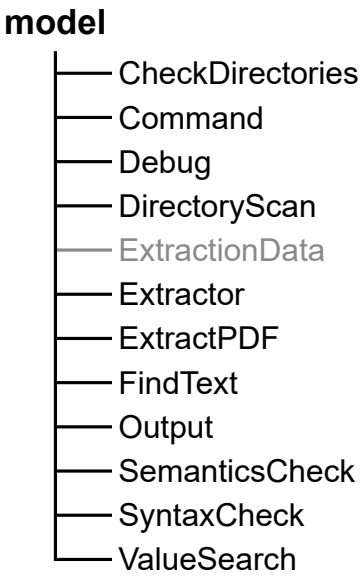
Implementace

Kapitola podrobně analyzuje implementaci extrakčního programu, který je napsaný v programovacím jazyce Java. Program je navržený jako desktopová aplikace, která uplatňuje návrhový vzor MVC (anglická zkratka pro *Model-View-Controller*). Vzhledem k tomu, že je hlavním úkolem extrakce dat, tak se bude kapitola zaměřovat především na modelovou část (ostatní části jsou podrobněji rozebírány v kapitole 6). V modelové části však není zahrnuta třída argumentů ExtractionData, která vznikla během implementace grafického uživatelského rozhraní (tzv. „GUI“).

Kapitola nejprve rozebírá strukturu a tok programu (sekce 5.1). Následně se zaměřuje na zpracování příkazového souboru (sekce 5.2) a zpracování jednotlivých záznamů o stavu pacienta (sekce 5.3). Poté se věnuje procesu vyhledávání (sekce 5.4) a nakonec se zabývá procesem ladění programu (sekce 5.5).

5.1 Struktura a tok programu

Základní struktura programu je založena na modelu, který se skládá z několika tříd:



Obrázek 5.1: Struktura modelové části programu (návrhový vzor MVC)

Každá třída reprezentuje určitou část procesu extrakce. Třídy se nejprve instanciují (vytváří se objekty) a poté se volají metody těchto instancí. Metody se volají v určitém pořadí.

Na začátku se volá metoda třídy Extractor, která původně sloužila k získávání argumentů ze vstupu (před implementací grafického uživatelského rozhraní). Nyní se používá k zapsání potenciálních chybových zpráv do záznamového souboru `log.txt`. Následně se volá metoda třídy ExtractPDF, která tvoří základní kostru procesu extrakce. Metoda má za úkol zavolat další metody, které:

- Zpracovávají příkazový soubor
- Získávají záznamy ze vstupního adresáře
- Vytvářejí výstupní složky
- Zpracovávají získané záznamy
- Vyhledávají hodnoty a obrázky
- Vytvářejí soubory ve výstupním adresáři

Nejdříve se volají melody tříd, které zpracovávají příkazový soubor. Soubor se zpracovává ve třídách SyntaxCheck a SemanticsCheck. Metoda třídy SyntaxCheck kontroluje, zda soubor odpovídá formální gramatice pseudojazyka Barty. Navíc si v průběhu kontroly ukládá jednotlivé parametry příkazu do objektu Command (instance třídy Command), který obsahuje atributy pro ukládání parametrů. Objekty se poté ukládají do seznamu `commandList`, kde každý objekt reprezenzuje jeden příkaz. Po úspěšném dokončení syntaktické analýzy se volá metoda třídy SemanticsCheck, která kontroluje, zda příkazy splňují veškeré sémantické požadavky (požadavky jsou vypsány v sekci 5.2).

V případě, že jsou požadavky sémantické analýzy splněny, se volá metoda třídy DirectoryScan. Tato metoda má za úkol získat jednotlivé cesty k záznamům, které se nacházejí ve vstupním adresáři. Záznamem může být libovolný soubor, který je ve formátu PDF. Pokud je aktivováno rekurzivní zanořování (viz sekce 6.1), tak se cesty hledají i v podadresářích vstupního adresáře. Nalezené cesty se ukládají do seznamu `pdfList`, který je dále analyzován metodami třídy pro zpracování záznamů. Po uložení cest se volá metoda třídy CheckDirectories, která odstraňuje původní výstupní složky a vytváří nové (pokud jsou potřeba). Složky jsou součátí výsledku extrakce, který se ukládá do výstupního adresáře. Jedná se o složky Image (pro extrahované obrázky) a Debug (pro obrázky procesu ladění). Kromě těchto složek (a jejich souborů) obsahuje výstup extrakce i další soubory, které jsou uvedeny níže v této sekci.

Po vytvoření výstupních složek se volá metoda třídy ValueSearch, která zajišťuje postupné zpracování záznamů, ve kterých se následně vyhledávají požadované hodnoty a obrázky. Nejprve se každý záznam zpracuje pomocí metody třídy FindText. Tato metoda má za úkol získat všechny textová data, která jsou obsažena v daném záznamu. Textová data se získávají pomocí knihovny Pdf2Dom [4], která je nástavbou knihovny Apache PDFBox [3]. Knihovna ukládá získaná data do seznamu `textElements`, který slouží k vyhledávání konkrétních hodnot. Data obsahují kromě samotného textu i souřadnice PDF, které určují jejich polohu v záznamu. Podle těchto souřadnic se může nalézt přesná poloha obrázku, který se má extrahovat. Zpracování záznamu končí ve chvíli, kdy jsou získána veškerá data, která jsou následně seřazena v seznamu `textElements` na základě jejich souřadnic.

Po zpracování záznamu se pokračuje v metodě třídy ValueSearch, která má za úkol vyhledávat požadované hodnoty a obrázky. Vyhledávání probíhá podle příkazů, které jsou

umístěny v seznamu `commandList`. Každý příkaz obsahuje parametry, které se vyhledávají podle vzoru parametrů příkazu (vzor je definován v sekci 4.5) v seznamu `searchList` (seřazený seznam textElements). Pokud se naleznou všechny parametry a identifikuje se hledaná hodnota, tato hodnota se uloží do seznamu `valuesList`. V případě, že se jedná o extrakci obrázku, se hledaný obrázek vystřihne ze záznamu pomocí souřadnic textových dat, které odpovídají parametrům příkazu (extrakce je podrobněji popsána v sekci 5.4). Obrázek se uloží do složky, která je pojmenováná podle extrahovaného záznamu. Tato složka je součástí výstupní složky Image.

Na závěr se ve výstupním adresáři vytvoří soubory, které představují výsledek extrakce. Soubor `result.csv` reprezentuje tabulku ve formátu CSV, která obsahuje extrahované hodnoty ze seznamu `resultList`. Hodnoty jsou v tabulce odděleny specifickým znakem (tzv. „oddělovačem“), který se vybírá v okně aplikace (viz sekce 6.1). Kromě výsledné tabulky se ve výstupu vytváří i soubor `log.txt`, který obsahuje záznamy chyb a varování, které nastaly během extrakce. Problémy se zapisují do řetězce `errorLog`, který se po ukončení extrakce vypisuje do výstupního souboru. Extrakce může být ukončena předčasně, pokud se během jejího průběhu vyskytne závažná chyba.

5.2 Zpracování příkazového souboru

První fází procesu extrakce je zpracování příkazového souboru. Příkazový soubor se zpracovává tak, že se syntakticky a sémanticky analyzují jeho příkazy. Příkazy se během syntaktické analýzy ukládají do seznamu `commandList`, který se následně využívá při vyhledávání hodnot a obrázků (sekce 5.4). Pokud některá analýza nalezne v příkazovém souboru chybu, extrakce se ukončí.

Během syntaktické analýzy se kontroluje, zda soubor odpovídá formální gramatice pseudojazyka Barty (pseudojazyk je definován v sekci 4.2). Kontrola probíhá tak, že se soubor čte postupně po řádcích pomocí metody `readLine()` třídy BufferReader. Metoda v každé iteraci načte řádek do řetězce `commandRow`, který se následně zkoumá v cyklu while. Cyklus analyzuje řetězec po jednotlivých znacích pomocí přepínače switch, který interpretuje znaky podle stavových proměnných. Stavové proměnné reprezentují část příkazu, která se v danou chvíli analyzuje v řetězci. Proměnná `stateType` identifikuje typ parametru, který se aktuálně zpracovává. Proměnná může být nastavena na jednu z následujících hodnot: „name“, „title“, „number“, „from“, „through“, „to“, „appearance“ nebo „imageProportions“. Proměnná tedy obsahuje hodnotu, která odpovídá konkrétnímu typu parametru. Kromě samotného typu je nutné specifikovat i jeho konkrétní část, která se aktuálně zpracovává. K tomuto účelu se používá proměnná `stateCode`, která může nabývat těchto hodnot:

- *quote* (parametry: jméno, název, odkud, přes, kam)
- *sequence* (stejné parametry jako u quote)
- *integer* (parametry: číslo, výskyt a proporce obrázku)
- *separator* (všechny parametry kromě parametru název)
- *space* (parametr proporce obrázku)
- *error* (všechny parametry)

Hodnota „quote“ je nastavena u všech parametrů, které obsahují text. Ve chvíli, kdy je nastavena tato hodnota, se očekává, že se na dané pozici v řetězci nachází počáteční uvozovka

(případně bílé znaky, dokud se nenarazí na uvozovku). Pokud se na této pozici vyskytuje jiný znak, nastane syntaktická chyba. Chyba se signalizuje nastavením hodnoty error, která se v další iteraci vyhodnotí tak, že se ukončí proces zpracování příkazového souboru i proces extrakce. Následně se chyba zapíše do záznamu souboru log.txt, který bude obsahovat číslo řádku a sloupce, na kterých se daná chyba v příkazovém souboru vyskytla. Chyba se zpracovává stejným způsobem i u dalších hodnot.

Hodnota „sequence“ se používá k získání textových parametrů, které se ukládají po jednotlivých znacích do řetězce `name`. Řetězec se po nalezení koncové uvozovky přiřadí k příslušnému atributu objektu Command (viz sekce 5.1), který se určí podle hodnoty proměnné `stateType`. Samotný objekt se inicializuje před zpracováním daného řádku a po jeho zpracování se přidává do seznamu `commandList`. Tento postup je shodný i pro hodnotu integer, která se používá k získání číselných parametrů.

Hodnota „separator“ označuje typ parametru, který se aktuálně zpracovává. Každý parametr (kromě parametru název) má svůj vlastní oddělovač, který ho jednoznačně identifikuje. Oddělovač se hledá po ukončení předchozího parametru, který se ukončuje uvozovkou nebo jiným znakem (s výjimkou prvního parametru, u kterého se oddělovač nehledá). Po nalezení oddělovače se přiřadí hodnota true do booleovské proměnné, která přísluší danému parametru. Každý parametr má svoji booleovskou proměnnou, která se inicializuje před analýzou daného řádku na hodnotu false a během analýzy se nastavuje na hodnotu true (pokud se našel její oddělovač). Hodnota slouží k ověření, zda se daný parametr uložil do atributu objektu Command. Pokud by se tedy napsal pouze oddělovač, za kterým by nenásledovala žádná hodnota, analýza by vyhodnotila příkaz jako chybný. Kromě toho je nutné dodržet pořadí parametrů, které je definováno předem (definice v sekci 4.2). V případě, že se toto pořadí nedodrží, nastává chyba. Pokud se některý nepovinný parametr vynechá a ostatní parametry jsou uvedeny ve správném pořadí, příkaz je platný.

Hodnota „space“ se využívá k určení polohy mezer, které se mohou vyskytovat v rámci parametru proporce obrázku. Parametr může mít mezeru za každou čárkou, která odděluje jeho jednotlivé hodnoty. Mezera je volitelná, takže pokud není uvedena, nemá to žádný vliv na vyhodnocení daného parametru.

Po dokončení syntaktické analýzy následuje kontrola sémantiky, která ověřuje, zda příkazy (reprezentované objekty Command v seznamu commandList) splňují všechny sémantické požadavky. Požadavky jsou následující:

- Hodnota parametru výskytu musí být minimálně 1.
- Hodnota parametru číslo musí být minimálně 0.
- Parametr proporce obrázku nesmí být uveden u příkazu, který extrahuje text (určení typu extrakce je popsáno v sekci 4.2).
- Parametr proporce obrázku musí mít 2 nebo 4 hodnoty.
- Hodnoty parametru proporce obrázku musí být celá čísla v rozmezí od 0 do 100.
- Parametr proporce obrázku, který má 4 hodnoty, nesmí mít překřížené vertikální ani horizontální proporce hodnot (křížení jsou podrobněji popsáno v sekci 4.2).

Pokud je hodnota parametru výskytu větší než 3 nebo pokud je hodnota parametru číslo větší než 5, zapíše se varování do souboru `log.txt`. Varování slouží pouze k upozornění na neobvykle vysoké hodnoty parametrů (nedochází tedy ke změně průběhu procesu extrakce).

5.3 Zpracování záznamů

Druhá fáze procesu extrakce zahrnuje zpracování záznamů, které obsahují informace o zdravotním stavu pacientů. Záznamy se zpracovávají samostatně pomocí metod třídy FindText, která je odvozená od třídy PDFBoxTree. Tato třída je součástí knihovny Pdf2Dom [4], která využívá knihovnu PDFBox [3]. Zpracování záznamů probíhá v několika krocích:

1. Ze záznamu se získá veškerý text i s jeho souřadnicemi, který se uloží do seznamu `textElements`.
2. Text se v tomto seznamu seřadí.
3. V seznamu se sloučí slova, která spolu významově souvisí (tzv. „slepování slov“).

Nejprve se ze záznamu získá veškerý text, který se zpracovává po jednotlivých stránkách. Před zpracováním každé stránky se zavolá metoda `startNewPage()`, která získá rozměry dané stránky prostřednictvím záznamu MediaBox (viz sekce 2.4). Rozměry se mezi sebou prohazují ve chvíli, kdy se zjistí, že je daná stránka otočená. Kromě rozměrů se zjišťuje také výškový offset, který slouží k určení polohy jednotlivých textových objektů. Offset se při každém volání metody aktualizuje tak, že se proměnná `nextPageYOffset` inkrementuje o výšku dané stránky. Proměnná se společně s rozměry stránky ukládá do atributů objektu Page, který se na konci metody přidává do seznamu `pageList`. Tento seznam se následně používá k určení polohy obrázků (viz sekce 5.4 a sekce 5.5).

V průběhu analýzy stránky se volá metoda `renderText()`, která zpracovává nalezené slova. Každé slovo je reprezentováno textovým boxem, který má 4 souřadnice: levou a pravou souřadnici x (začátek a konec slova) a horní a dolní souřadnici y (horní a dolní mez slova). Tyto souřadnice slouží pro určení polohy textu v záznamu.

Obrázek 5.2: Ukázka textového boxu slova Sensitivity

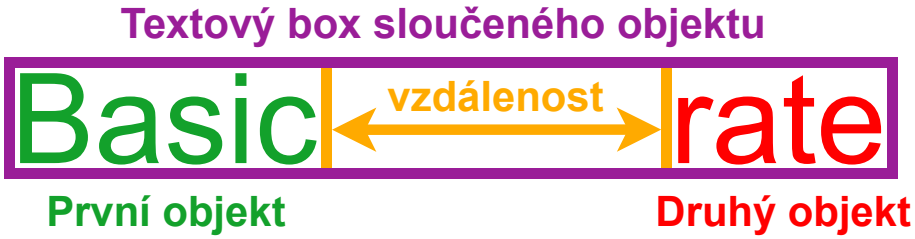
Textový box se společně s daným slovem ukládají do atributů objektu textElement, který se na konci metody přidává do seznamu `textElements`. Seznam reprezentuje daný záznam, ve kterém se následně vyhledávají požadované hodnoty pomocí instrukcí ze seznamu `commandList` (viz sekce 5.4). Pokud se v záznamu nenajde žádný text, vypíše se varování do souboru `log.txt`.

Následně se texty seřazují podle jejich souřadnic. Řazení probíhá pomocí metody `sort()` třídy Collections, která využívá metodu `compareTo()` pro seřazení textových objektů v seznamu textElements. Pro použití této metody je nutné, aby byl textový objekt implementován rozhraním Collections a zároveň měl definovanou tuto metodu. Metoda má za úkol seřadit objekty podle souřadnice y. K tomuto účelu se používají tzv. středové souřadnice, které se vypočítávají následovně:

`dolní souřadnice y / 2`

Řazení probíhá tak, že se nejprve vypočítají středové souřadnice porovnávaných objektů (vždy se v metodě porovnávají právě 2 objekty). Tyto souřadnice se následně odečtou a podle tohoto výsledku se porovnávané objekty seřadí (pokud je výsledek kladný, tak je první souřadnice větší a naopak). Pokud jsou středové souřadnice textových objektů shodné (texty byly na jednom řádku), tak se tyto objekty seřadí podle levé souřadnice x (princip řazení je stejný jako u souřadnice y). Seřazené objekty se nakonec uloží do seznamu `sortedList`.

Poslední fází zpracování záznamů je tzv. „slepování slov“, které se provádí v metodě `sortAndClassify()`. Metoda v každé iteraci porovnává dva textové objekty ze seznamu `sortedList`, u kterých kontroluje jejich vzájemnou vzdálenost. Vzdálenost se vypočítá rozdílem pravé souřadnice x prvního objektu a levé souřadnice x druhého objektu. Pokud se oba textové objekty nachází na jednom řádku a jejich vzdálenost je menší nebo rovna hodnotě, kterou určil uživatel (viz sekce 6.1), pak se tyto objekty slučují. Objekty se slučují tak, že se informace z druhého objektu přiřadí do atributů prvního objektu a druhý objekt se následně zahodí. První objekt má po sloučení upravený atribut textu, který obsahuje obě slova objektů spojená mezerou, a novou pravou souřadnici x získanou od druhého objektu. Sloučený objekt se porovnává v další iteraci s novým objektem, který je na druhé pozici. Pokud se objekty nemohou sloučit (ať už jsou od sebe příliš vzdálené nebo je každý na jiném řádku), tak se první objekt uloží do seznamu `row`, který se na konci řádku uloží do seznamu `searchList`, a druhý objekt se posune na místo prvního objektu. Na tomto místě se v další iteraci prorovnává s novým objektem, který je umístěný na druhé pozici. Tento proces probíhá do té doby, dokud se nezpracuje poslední objekt.



Obrázek 5.3: Ukázka slučování textových objektů (s výpočtem jejich vzdálenosti)

5.4 Vyhledávání hodnot a obrázků

Třetí a zároveň poslední fází procesu extrakce je vyhledávání hodnot a obrázků. Vyhledávání probíhá v metodě `searchList()`, která má za úkol získat hodnoty nebo polohy obrázků z jednotlivých záznamů. Záznamy jsou postupně reprezentovány seznamem `searchList`, do kterého se v každé iteraci vnějšího cyklu vyhledávání ukládá nový záznam. Záznamy se extrahují prostřednictvím příkazů, které jsou uloženy v seznamu `commandList`. Příkazy obsahují sadu instrukcí, podle kterých se mají dané hodnoty vyhledávat. Instrukce jsou reprezentovány objekty Command, které se postupně zpracovávají ve vnitřním cyklu vyhledávání. U každého objektu se nejprve zkontroluje, zda obsahuje parametr výskyt. Tento parametr určuje, kolikrát se dané vyhledávání bude provádět. Pokud se vyhledávání provádí vícekrát, tak se pokračuje z místa posledního nálezu. V případě, že se tento parametr v objektu nevyskytuje, se hodnota vyhledává pouze jednou.

Po stanovení počtu vyhledávání následuje validace parametrů a získání samotné hodnoty v metodě `getValue()`. Validace ověřuje, zda se všechny parametry uvedené v objektu Command vyskytují v daném záznamu. Validace probíhá podle předem definovaného vzoru (vzor je definovaný v sekci 4.5), který se kontroluje pomocí cyklu *while*. Cyklus prověřuje jednotlivé parametry pomocí přepínače *switch*, který vybírá své varianty na základě příkazových proměnných. Příkazové proměnné reprezentují validovaný parametr a jeho aktuální stav (nalezený/nenalezený/nepoužitý). Proměnná `commandType` specifikuje parametr, který se právě validuje. Proměnná může nabývat jedné z následujících hodnot: „from“, „through“, „title“, „number“ nebo „to“. Proměnná tedy obsahuje hodnotu, které odpovídá danému parametru. Kromě samotného parametru je nutné určit i jeho stav. K tomuto účelu se používá proměnná `commandCode`, která může být nastavena na jednu z těchto hodnot: „found“, „used“ nebo „error“.

Hodnota „found“ je nastavena právě tehdy, když je metoda `itemNotFound()` úspěšná ve validaci daného parametru. Parametr se validuje tím, že se nalezne v seznamu `searchList`. Tento seznam se prohledává postupně po jednotlivých řádcích, na kterých se kontroluje každé slovo. Kontrola probíhá prostřednictvím dvou proměnných: proměnné `listIndex`, která reprezentuje prohledávané řádky, a proměnné `itemIndex`, která reprezentuje jednotlivá slova na daném řádku. Aby se zachovaly hodnoty proměnných (kvůli validaci dalších parametrů), tak jsou proměnné součástí atributů hlavní třídy searchValue. Hodnoty se zachovávají z toho důvodu, aby se vyhledávání provádělo z místa, kde se nalezl předchozí parametr. V případě, že se jedná o parametr název nebo o parametr kam, se navíc provádí i uložení textových objektů, které odpovídají těmto parametrům. Textové objekty se ukládají do atributů objektu Command, který je využívá v případě extrakce obrázku (kvůli souřadnicím jejich textových boxů).

Ve chvíli, kdy se jakýkoliv parametr nedokáže validovat, se do příkazové proměnné nastavuje hodnota „error“. Hodnota způsobí, že se do souboru `log.txt` zapíše chyba validace odpovídající danému parametru. Poté se vyhledávání ukončí a chybová hodnota se přidá do seznamu `valuesList`, který reprezentuje hodnoty nalezené v daném záznamu. Následně se začne zpracovávat nový příkaz (popř. i nový záznam).

Před validací volitelného parametru se ověří, zda je daný parametr obsažený v atributu objektu Command. Pokud se parametr v atributu nenachází, přiřadí se do příkazové proměnné hodnota „unused“.

Speciální případ představuje parametr číslo, který se nevaliduje. Parametr slouží pouze k vyhledání konkrétní hodnoty v metodě `searchValue()`, která se provádí pouze v případě extrakce textu (při extrakci obrázku se hodnota nevyhledává a příkazová proměnná se nastavuje na hodnotu „unused“). Metoda vyhledává hodnotu, která je na stejném řádku jako je parametr jméno (proměnná `listIndex` tedy zůstává stejná). Hodnota se získává tak, že se v každé iteraci vyhledávácího cyklu odečítá hodnota parametru číslo, dokud není nulová. Následně se nalezená hodnota uloží do řetězce `value`, který se vrací jako návratová hodnota metody `getValue()`. Kromě nalezené hodnoty se ukládá i její textový objekt, jehož souřadnice se mohou použít v režimu ladění. Pokud je hodnota parametru větší než počet položek, které se mají na daném řádku vyhledat, vrací se hodnota „value not found“. Tato hodnota se zpracovává stejným způsobem jako hodnota „error“ (chybový záznam se zapíše do souboru `log.txt` a vyhledávání se ukončí).

Pokud se jedná o extrakci textu, tak se návratová hodnota z metody `getValue()` přidá do seznamu `valuesList`. Tento seznam se po zpracování všech příkazů uloží do seznamu `resultList`, který reprezentuje výsledky extrakce jednotlivých záznamů. V případě, že jde o extrakci obrázku, se nejprve ověří, zda není návratová hodnota metody `getValue()`

„error“, a následně se získá obrázek, který se extrahuje v metodě `makeImage()`. Obrázek se extrahuje následujícím způsobem:

1. Prostřednictvím souřadnic textových objektů se získá poloha obrázku.
2. Tato poloha se upraví podle parametru proporce obrázku (pokud ho příkaz obsahuje).
3. Vytvoří se obrázek stránky, ze které se má daný obrázek extrahovat.
4. Podle polohy se z obrázku vyjme pouze část, která se má extrahovat.
5. Vyjmutá část se po kontrole názvu uloží do složky, která je umístěna ve složce Image.

Nejprve se určí poloha obrázku (princip v sekci 4.2). Poloha se určí pomocí souřadnic textových objektů, které se uložily do atributů objektu Command, a pomocí souřadnic objektu Page (viz sekce 5.3), který obsahuje informace o dané stránce, na které se obrázek nachází. Objekt Page se ze seznamu `pageList` vybírá pomocí atributu textového objektu, který obsahuje informaci o čísle stránky, na které se daný objekt vyskytuje.

Následně se poloha upravuje podle parametru proporce obrázku (princip v sekci 4.2). Poloha se upravuje podle počtu hodnot, které daný parametr obsahuje. Pokud parametr obsahuje 2 hodnoty, tak se vertikální i horizontální poloha zkrátí o tolik procent, kolik je uvedeno v tomto parametru. Pokud parametr obsahuje 4 hodnoty, tak se poloha obrázku zkrátí podle jednotlivých hodnot.

Poté se vytvoří obrázek stránky pomocí metody `renderImageWithDPI()`, která je součástí třídy PDFRenderer. Tato metoda vykresluje danou stránku s rozlišením DPI (anglická zkratka pro *Dots Per Inch*, v překladu „počet obrazových bodů na palec“), která se ukládá jako obrázek do objektu třídy BufferedImage. Jelikož má obrázek jiné jednotky než mají souřadnice polohy, tak se musí souřadnice vynásobit konstantou, která tento rozdíl eliminuje. Konstanta se vypočítá podílem rozměru obrázku a rozměru stránky, který se počítá jak pro horizontální (podíl šířek), tak pro vertikální (podíl výšek) souřadnice. Přepočítané souřadnice se následně používají k vyjmutí části obrázku, která odpovídá jejich poloze. To se provádí v metodě `getSubimage()`, která je součástí třídy ImageBuffer. Vyjmutý obrázek se následně ukládá.

Před uložením obrázku se kontroluje jeho název v metodě `checkFileSemantics()`, která má za úkol nahradit veškeré znaky, které daný operační systém v názvu nepodporuje. Tyto znaky se převedou do textové podoby, která je ohraničená podržítkem a znakem dolaru (např. znak $*$ se nahrazuje v názvu jako `_$asterisk$_`). Kromě znaků se kontrolují také vyhrazené názvy, před které se vkládá předpona `RESERVED_` (např. ve windows se název COM1 nahradí názvem `RESERVED_COM1`).

Nakonec se obrázek ve formátu PNG uloží do složky pomocí metody `write()`, která je součástí třídy ImageIO. Složka je umístěna v adresáři Image a je pojmenována podle aktuálně zpracovávaného záznamu. Složka je vytvořena před zpracováním daného záznamu pouze v případě, že se má v záznamu extrahovat alespoň jeden obrázek.

5.5 Režim ladění

Pokud extrakce neproběhla podle očekávání, může se aktivovat režim ladění. Režim se aktivuje pomocí tlačítka „Debug Mode“ v grafickém uživatelském rozhraní, které je popsáno v kapitole 6. V případě, že je tento režim aktivován, se kromě samotného procesu extrakce provádí i zjišťování informací o jeho průběhu, které se ukládají do výstupní složky Debug.

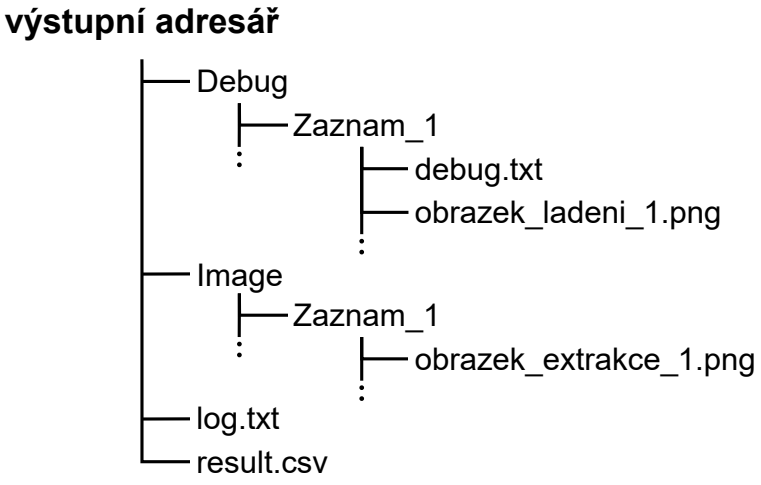
Zjišťují se následující informace: 1. Kolikrát se v záznamu u každého příkazu vyskytuje vzor parametrů a parametr název. 2. Na kterém místě se v záznamu získaly extrahované hodnoty a obrázky. Informace se získávají pomocí metod třídy Debug, které se volají na konci zpracování každého příkazu.

Nejprve se volá metoda `numberOfOccurrences()`, která má za úkol získat informace zmiňované v prvním bodu. Metoda zjišťuje počet výskytů vzoru parametrů (vzor je popsán v sekci 4.5 a validován v sekci 5.4) prostřednictvím volání metody `getValue()`, která validuje parametry do té doby, než se validace nezdaří (vrátí se hodnota „error“ nebo hodnota „value not found“). Před každým jejím voláním se inkrementuje počítadlo `phraseOccurrence`, které zaznamenává počet použití této metody. Následně se zjišťuje počet výskytu parametru název, který se vyhledává v seznamu `searchList`. Seznam se prohledává postupně a při každém nálezu tohoto parametru se zvyší počítadlo `titleOccurence`. Počítadla se společně s parametry příkazu ukládají do řetězce `debugLog`, který se na konci zpracování daného záznamu zapisuje do souboru `debug.txt` (soubor je popsán v sekci 4.5).

Následně se volá metoda `imageOfSelectedSection()`, která vytváří snímky stránek s označením míst, na kterých se hodnoty a obrázky extrahovaly. Snímky se vytvářejí pouze v případě, že byla extrakce úspěšná (výsledkem extrakce tedy není hodnota „error“). Ve chvíli, kdy se provádí extrakce obrázku, se zavolá metoda `makeImage()` (metoda je popsána v sekci 5.4), která v režimu lazení využívá polohu ořezu obrázku k vykreslení červeného obdélníku pro zvýraznění míst extrakce. Tento obdélník se vytváří metodou `drawRect()`, která je součástí třídy Graphics2D.

Pokud se jedná o extrakci textu, tak se zvýraznění daného místa provádí obdobným způsobem jako v metodě `makeImage()`, akorát se pro zajištění větší přehlednosti označuje celý řádek od začátku až po vybranou hodnotu (levá souřadnice x se získá z parametru název a ostatní souřadnice se získají z vybrané hodnoty). Pokud se hodnota nenalezne (tzv. hodnota „value not found“), tak se řádek zvýrazňuje až po jeho poslední položku. Kromě zvýraznění extrahovaného místa se navíc vyznačuje i vybraná hodnota, která se vybarvuje světle červenou barvou (ukázka v sekci 4.5).

Snímky se po extrakci společně se souborem `debug.txt` ukládají do složky, která je umístěna v adresáři Debug. Tato složka je pojmenována podle zpracovávaného záznamu. V následujícím obrázku je ilustrován možný výsledek extrakce spolu s výstupem ladění:



Obrázek 5.4: Ukázka výstupu extrakce s aktivovaným režimem ladění

Kapitola 6

Grafické uživatelské rozhraní

Kapitola rozebírá grafické uživatelské rozhraní, které bylo přidáno jako nádstavba k původní konzolové aplikaci, aby mohl uživatel jednodušeji zadávat vstupní argumenty programu. V důstedku této nádstavby se program přetvořil na desktopovou aplikaci, která využívá návrhový vzor MVC (anglická zkratka pro *Model-View-Controller*). Tato kapitola se zaměřuje především na vzhledovou a kontrolní část návrhu, který slouží k interakci s uživatelem. V modelové části se analyzuje pouze třída ExtractionData, která shromažďuje argumenty programu (ostatní třídy modelu jsou popsány v kapitole 5). Kapitola se nejprve věnuje návrhu grafického uživatelského rozhraní (sekce 6.1) a následně se zaobírá jeho samotnou implementací (sekce 6.2).

6.1 Návrh

Základem návrhu grafického uživatelského rozhraní je získání vstupních argumentů, které se získávají prostřednictvím interakce s uživatelem. Uživatel si volí argumenty podle daného typu záznamu a osobních preferencí. Aby se uživateli ulehčila práce, tak se zavedl systém na zapamatování nastavené konfigurace argumentů, jelikož se některé extrakce mohou provádět opakovaně. Zapamatování probíhá tak, že se po každém spuštění procesu extrakce vytvoří konfigurační soubor `setup.properties`, který obsahuje veškeré informace o vytvořené konfiguraci. Tento soubor se ukládá do vstupní složky, ze které byla provedena extrakce. Pokud již v dané složce soubor existuje, tak se pouze přepíše.

Kromě získávání argumentů se v návrhu řeší také vzhled aplikace, který je založený na kombinaci odstínů červené a zlaté barvy, a její uživatelské rozhraní, které se skládá z hlavního menu, nastavení konfigurace, procesu extrakce a tutoriálu. Každá z těchto částí je podrobněji popsána níže.

Hlavní menu

Hlavní menu obsahuje 4 tlačítka, které primárně slouží k volbě způsobu tvorby konfigurace. Jedná se o:

- Tlačítko „New Setup“, které umožňuje vytvořit novou konfiguraci.
- Tlačítko „Load Setup“, které umožňuje načíst existující konfiguraci.
- Tlačítko „Last Setup“, které umožňuje načíst poslední konfiguraci.
- Tlačítko „Tutorial“, které umožňuje spustit návod na používání této aplikace.

Tlačítko „Load Setup“ slouží k načtení existující konfigurace ze souboru `setup.properties`, který se vybírá prostřednictvím uživatelského rozhraní knihovny FileChooser, a tlačítko „Last Setup“ umožňuje načíst poslední konfiguraci nezávisle na tom, zda byla uložena do tohoto souboru (informace o poslední konfiguraci se ukládají jinam).

Obrázek 6.1: Ukázka hlavního menu aplikace

Nastavení konfigurace

Nastavení konfigurace probíhá v 5 oknech, které slouží k výběru jednotlivých argumentů:

1. Okno „Choose input directory“ umožňuje selekci vstupní složky, ze které se budou extahovat záznamy. Složka se vybírá pomocí tlačítka Browse, které spouští uživatelské rozhraní knihovny DirectoryChooser. Jakmile se vybere daná složka, tak se zobrazí její cesta ve světle žlutém textovém poli, které se nachází vedle tlačítka Browse. Pokud se vytváří nová konfigurace, tak se jako výchozí složka automaticky zvolí adresář, ve kterém se nachází spuštěný program (tzv. „uživatelův pracovní adresář“). Kromě výběru vstupní složky se také volí možnost „Recursive Nesting“, která povoluje extrakci záznamů i v podsložkách vstupní složky (tato možnost je aktivována implicitně).

2. Okno „Choose output directory“ slouží k výběru výstupní složky, do které se bude vkládat výstup extrakce. Složka se vybírá stejným způsobem, jako se vybírala v prvním okně. Kromě výstupní složky se volí možnost „Debug mode“, která aktivuje režim ladění (návrh režimu je popsán v sekci 4.5 a jeho implementace je popsána v sekci 5.5).

3. Okno „Choose command file“ umožňuje selekci příkazového souboru, který definuje, jaké hodnoty mají být extrahovány. Příkazový soubor se vybírá stejným způsobem, jako se vybíraly předchozí dvě složky.

4. Okno „Choose values spacing“ slouží k určení optimální vzdálenosti mezi jednotlivými sloupci v tabulkách záznamů. Podle této vzdálenosti se následně slepují slova v procesu zpracování záznamů (sekce 5.3). Vzdálenost se může zvolit ze dvou přednastavených možností (možnost „Small“ pro malý rozestup a možnost „Large“ pro velký rozestup) nebo se může zvolit vlastní vzdálenost (možnost „Own“), která musí být reálným číslem v rozmezí od 0 do 100.

5. Okno „Choose CSV separator“ umožňuje vybrat znak, který se aplikuje jako oddělovač hodnot v tabulce result.csv. V tomto okně se může zvolit jeden z následujících znaků: **;** (středník), **,** (čárka) nebo **␣** (mezera).

Kromě části s výběrem argumentů obsahuje každé okno i navigační tlačítka, která slouží k přecházení mezi jednotlivými okny. Jedná se o:

- Tlačítko „Previous“, které umožňuje přejít do předchozího okna konfigurace.
- Tlačítko „Next“, které umožňuje přejít do následujícího okna konfigurace.
- Tlačítko „Extract“, které spouští proces extrakce.

Tlačítko „Previous“ je v prvním okně nahrazeno tlačítkem „Menu“, které slouží k návratu do hlavního menu. Tlačítko „Next“ funguje pouze v případě, že je vyplněn argument daného okna. Kromě toho není k dispozici v posledním okně konfigurace. Tlačítko „Extract“ funguje pouze tehdy, když jsou vyplněny všechny argumenty konfigurace. Toto tlačítko slouží k urychlení nastavení konfigurace ve chvíli, kdy má uživatel aktivovaný režim ladění (tlačítko „Last Setup“) nebo kdy je zvolena existující konfigurace (tlačítko „Load Setup“).

Obrázek 6.2: Ukázka prvního okna konfigurace

Proces extrakce

Proces extrakce slouží k zobrazení aktuálního stavu získávání dat. Před zahájením samotného procesu se zapíše nastavení konfigurace do souboru `setup.properties`, který se ukládá do vybrané vstupní složky. Následně se během procesu aktualizuje indikátor postupu s jeho popiskem, které se mění na základě aktuální fáze extrakce. Popisek se nastavuje na jednu z těchto zpráv: Inicializating (*Inicializace*), Checking syntax (*Syntaktická kontrola*), Checking semantics (*Sémantická kontrola*), Scanning directories (*Skenování adresářů*), Creating output directories (*Vytváření výstupních adresářů*), Searching for values (*Vyhledávání hodnot*), Creating result file (*Vytváření souboru result.csv*) a Creating log file (*Vytváření souboru log.txt*). Po úspěšné extrakci se popisek nastaví na zprávu Extraction completed (*Extrakce dokončena*). Kromě popisku se také mění barva indikátoru, která se nastavuje podle výsledku extrakce. Barvy se rozlišují následujícím způsobem:

- Červená – pokud se extrakce nezdařila
- Oranžová – pokud se extrakce podařila s chybami nebo varováními
- Zelená – pokud se extrakce podařila bez chyb a varování

Společně s vybarvením indikátoru se také ukáže příslušná zpráva, která se zobrazí pomocí dialogového okna třídy Alert. Pokud se extrakce nezdaří nebo se provede s chybami, tak se tyto chyby zapíšou do souboru `log.txt`, který se ukládá do zvoleného výstupního adresáře. V případě, že dojde k nesprávnému zadání argumentů při extrakci velkého množství souborů (extrakce malého množství souborů proběhne rychleji, než stihne uživatel zareagovat), se může použít tlačítko „Cancel“, které slouží k zastavení procesu extrakce.

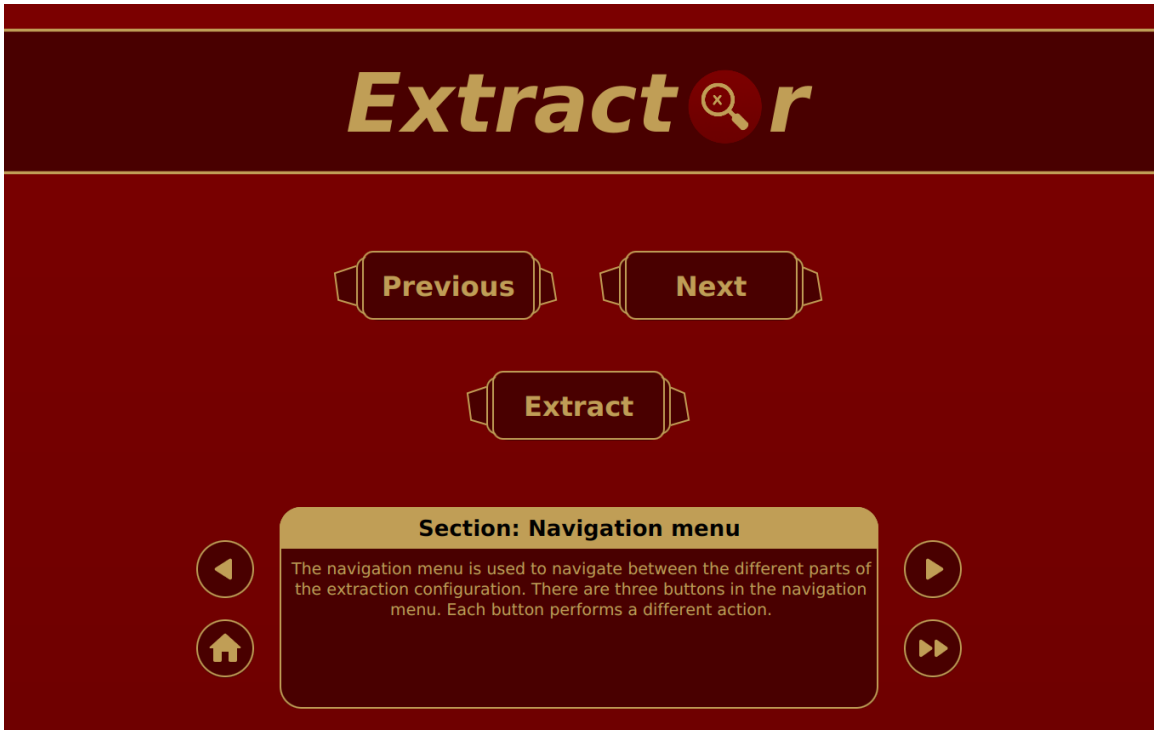
Obrázek 6.3: Ukázka okna procesu extrakce

Tutoriál

Tutoriál poskytuje nápovědy, které mají za úkol vysvětlit používání této aplikace. Každá nápověda obsahuje vizualizaci konkrétní části okna aplikace a její popis v rámci nápovědního dialogu. Tyto nápovědy jsou rozděleny do sekcí, které postupně popisují jednotlivé části daného okna aplikace. Všechny sekce začínají s označením „Section:“, které je uvedeno před názvem nápovědního dialogu. Kromě názvu a nápovědy obsahuje dialog také 4 tlačítka:

- Tlačítko levé šipky, které umožňuje přejít do předchozího okna nápovědy.
- Tlačítko pravé šipky, které umožňuje přejít do následujícího okna nápovědy.
- Tlačítko domečku, které umožňuje přejít do hlavního menu.
- Tlačítko dvojité pravé šipky, které umožňuje přejít do další sekce tutoriálu.

Kromě tlačítek se může použít i levá a pravá šipka na klávesnici, které umožňují přechod mezi jednotlivými okny nápovědy.



Obrázek 6.4: Ukázka okna tutoriálu

6.2 Implementace

Grafické uživatelské rozhraní je implementováno pomocí platformy JavaFX [7], která vykresluje jednotlivá okna pomocí těchto prvků:

- stage (*okno*), která představuje hlavní okno aplikace (vždy je pouze jeden);
- scene (*scéna*), která představuje scénu tohoto okna (může jich být více);
- node (*uzel*), který představuje jednotlivé grafické prvky této scény.

Scény v oknech se mohou měnit pomocí metody `load()`, která je součástí třídy FXMLLoader. Tato metoda načte novou scénu uloženou v jiném FXML souboru, pomocí které se překreslí stávající obsah daného okna. Kromě vykreslování oken se v implementaci řeší také architektura platformy, která je postavená na návrhovém vzoru MVC (viz úvod do kapitoly 6). Návrhový vzor tvoří 3 části:

- Model, který řeší logiku aplikace a zpracování dat.
- View, který řeší prezentaci dat.
- Controller, který řeší uživatelskou interakci a práci s daty.

Tyto části jsou v aplikaci reprezentovány jednotlivými složkami (model, controller a extractor_gui), které obsahují soubory pro danou část návrhu.

Vzhledová část návrhu se v rámci platformy JavaFX vytváří pomocí FXML souborů, které obsahují grafické prvky zapsané v syntaxi jazyka XML. Tyto prvky, kterými jsou například tlačítka, kontejnery, textová pole a mnoho dalších, mohou být vizuálně upraveny pomocí jazyka CSS, který definuje jejich vzhled. Prvky se vytvářejí a modifikují pomocí vizuálního nástroje SceneBuilder[1], který umožňuje tvořit uživatelské rozhraní bez toho, aniž by se musel psát kód (tzv. technologie „drag & drop“). Před samotným vytvářením grafických prvků je nutné, aby každý FXML soubor obsahoval odkaz na kontrolní část (tzv. kontrolér), která s těmito prvky pracuje.

Kontrolní části návrhu spouští metody (tzv. listenery), které umožňují obsluhovat události vyvolané uživatelem. Těmito událostmi mohou být například kliknutí na tlačítko nebo stiknutí klávesy, které vyvolají překreslení scény (tzv. „přechod“ do jiného okna). Metody musí být označeny značkou `@FXML`, která deklaruje jejich propojení s grafickými prvky v příslušném FXML souboru. Kromě nich musí obsahovat značku také grafické prvky, se kterými se má v kontroléru pracovat. Práce s těmito prvky probíhá tak, že se například kontroluje, která možnost se označila (pomocí výběrových tlačítek, tzv. radio buttonů), nebo se zjišťuje, zda je povoleno nějaké nastavení (prostřednictvím zaškrtávacích políček, tzv. checkboxů). Tyto prvky se v kontroléru pojmenovávají podle názvu jejich identifikačního atributu `fx:id`, který je uvedený v souboru FXML. Kromě grafických prvků se také pracuje s knihovnami, které vytvářejí uživatelské rozhraní pro výběr složek a souborů. Pro tento účel se využívají knihovny FileChooser a DirectoryChooser, které jako výsledek vrací zvolený soubor nebo složku. Z těchto objektů se získává jejich cesta, která se vyplní do textového pole v daném okně a poté se uloží do instance třídy ExtractionData, která se nachází v modelové části.

Modelová část návrhu má za úkol provést extrakci záznamů. Vzhledem k tomu, že se celý proces extrakce rozebírá v kapitole 5 (kapitola „Implementace“), tak se tento popis zaměří pouze na třídu ExtrakcionData, která se doposud neprobírala. Třída ExtractionData slouží k ukládání argumentů, které se vybírají prostřednictvím tlačítek v jednotlivých oknech nastavení konfigurace (viz sekce 6.1). Ve chvíli, kdy se zmáčkne tlačítko „Extract“, se tyto argumenty zapíší do konfiguračního souboru `setup.properties`, který se vytvoří pomocí knihovny Properties. Tento soubor je možné následně načíst prostřednictvím tlačítka „Load Setup“ v hlavním menu. Kromě toho se argumenty ukládají do míst, ke kterým nemá běžný uživatel přístup. Jedná se lokální uložiště, do kterých se konfigurace ukládá pomocí knihovny Preferences. Tato konfigurace se následně využívá při načítání posledního nastavení prostřednictím tlačítka „Last Setup“ v hlavním menu. Pokud se načte soubor, který obsahuje neplatné cesty (např. cestu ke smazané složce), vypíše se varování pomocí knihovny Alert a tyto cesty se následně nahradí implicitními, pokud je to možné.

[1] https://www.oracle.com/java/technologies/javase/javafxscenebuilder-info.html

Kapitola 7

Testování

Kapitola testování má za úkol zhodnotit kvalitu extrakce. Hodnocení se provádí tak, že se u každé zdravotnické společnosti testuje, do jaké míry se výsledky extrakce shodují s informacemi, které se měly extrahovat. Toto testování se provádí v rámci těchto společností:

- Biotronik (sekce 7.1)
- Boston Scientific (sekce 7.2)
- Medtronic (sekce 7.3)

U každé společnosti se nejprve provede testování jednoho příkazu na velkém množství souborů, poté se provede testování více příkazů na menším množství souborů a nakonec se provede testování jednoho příkazu na extrakci obrázků. Každé testování se následně vyhodnotí pomocí metrik přesnosti a úplnosti (z anglického výrazu *Precision and Recall*), které se využívají pro posouzení kvality výsledků. Obě metriky se následně převádí na procentuální hodnoty tak, že se vynásobí konstantou 100.

Metrika přesnosti vyjadřuje, do jaké míry se shodují výsledky s hodnotami, které se měly extrahovat. Tato metrika se určuje pomocí termínu „pravdivě pozitivní“ (z anglického výrazu *True Positive*) a termínu „falešně pozitivní“ (z anglického výrazu *False Positive*). Termín „pravdivě pozitivní“ označuje případ, kdy se extrahovaná hodnota shoduje s očekávaným výsledkem, zatímco termín „falešně pozitivní“ označuje případ, kdy se extrahovaná hodnota neshoduje s očekávaným výsledkem. Tyto termíny se používají k výpočtu přesnosti následovně:

$$Přesnost = \frac{pravdivě\ pozitivní}{pravdivě\ pozitivní + falešně\ pozitivní}$$

Přesnost tedy udává poměr mezi počtem správně extrahovaných hodnot a celkovým počtem extrahovaných hodnot.

Metrika úplnosti udává, do jaké míry se úspěšně extrahovaly výsledky. Tato metrika se určuje pomocí pojmu „pravdivě pozitivní“ a pojmu „falešně negativní“ (z anglického výrazu *False Negative*). Pojem „pravdivě pozitivní“ představuje situaci, kdy se hodnota úspěšně extrahovala. Oproti tomu pojem „falešně negativní“ představuje situaci, kdy se tato hodnota nedokázala extrahovat, přestože se v záznamu vyskytuje. Tyto pojmy se používají k výpočtu úplnosti následovně:

$$Úplnost = \frac{pravdivě\ pozitivní}{pravdivě\ pozitivní + falešně\ negativní}$$

Úplnost tedy vyjadřuje poměr počtu extrahovaných hodnot k celkovému počtu hodnot, které se měly extrahovat.

7.1 Společnost Biotronik

Společnost Biotronik zapisuje veškeré stavy pacienta a srdečního stimulátoru do jednoho záznamu. Záznam zahrnuje jak textové hodnoty stavů, tak i vektorové obrázky grafů měření (viz sekce 4.1). Každý záznam se v rámci testování zpracovával s použitím hodnoty 10,5 jako argumentu optimální vzdálenosti (možnost „Large“, viz sekce 6.1). Kromě toho byly všechny záznamy ve stejném formátu a každý z nich patřil jinému pacientovi.

Nejprve se testoval příkaz, který měl za úkol nalézt hodnotu stavu „Basic Rate“ ve 20 záznamech. Vzhledem k tomu, že se v každém záznamu podařilo extrahovat hodnotu, se v tomto testování neobjevily žádné falešně negativní výsledky. Kromě toho byla každá hodnota určena správně, takže všechny výsledky byly pravdivě pozitivní (falešně pozitivní výsledek tedy nebyl žádný). To znamená, že přesnost i úplnost tohoto testování dosáhla 100 %. Aby se přesnost i úplnost narušila, bylo potřeba změnit optimální vzdálenost na hodnotu 3,16 nebo menší (obě metriky v tomto případě dosahovaly 0 %).

Následně se testovalo 5 příkazů, které měly nalézt hodnoty ve 13 záznamech. Jednalo se o hodnoty stavů „Number S1“, „Onset“, „Manufacturer“, „Mode“ (u testování impedance) a „Shock impedance“. V tomto testování se vyskytla chyba. Pokud daný stav neobsahoval hodnotu, tak se výsledek neidentifikoval správně (místo hodnoty „value not found“ se našla jiná hodnota v daném záznamu). Proto se v rámci tohoto testování objevily 4 falešně pozitivní výsledky, které snížili metriku přesnosti. Testování tedy v tomto případě dosáhlo pouze přesnosti 93,85 %. Jelikož se neobjevil žádný falešně negativní výsledek, úplnost opět dosáhla 100 %. Poté, co se upravil algoritmus na výběr hodnot, se přesnost při stejném postupu zvýšila na 100 %. Aby se tato přesnost udržela, musela se optimální vzdálenost pohybovat v intervalu hodnot od 3,17 do 66,55. Pokud byla hodnota menší, tak se narušily obě metriky (viz předchozí test). Pokud byla hodnota větší, tak se snížila přesnost na 87,69 %, a při překročení hodnoty 72,27 se přesnost snížila na pouhých 73,85 %. Úplnost při zvyšování zůstávala stejná.

Nakonec se testoval příkaz, který měl extrahovat obrázek. Jednalo se o obrázek „Long term rate trend“, který se extrahoval ve 13 záznamech. Jelikož se podařilo správně extrahovat všechny obrázky (tj. všechny výsledky extrakce byly pravdivě pozitivní), tak bylo u tohoto testování dosaženo 100 % přesnosti i úplnosti.

Testování potvrdilo, že je extrakce záznamů od společnosti Biotronik vysoce spolehlivá. Celkem se extrahovalo 98 informací, z nichž bylo 85 hodnot a 13 obrázků. Po zprůměrování všech hodnot byla přesnost testování (bez úpravy algoritmu) 97,95 % a úplnost testování dosáhla 100 %.

7.2 Společnost Boston Scientific

Společnost Boston Scientific vytváří až 11 různých typů záznamů, které se generují na základě provedeného vyšetření daného pacienta. Každému pacientovi se vygeneruje jiné množství záznamů, které mohou být různého typu. Záznamy obsahují jak textové hodnoty stavů, tak i vektorové obrázky grafů měření (viz sekce 4.1). Záznamy se při testování zpracovávaly s použitím hodnoty 3,25 jako argumentu optimální vzdálenosti (možnost „Small“,

viz sekce 6.1). Kromě toho každý záznam patřil jinému pacientovi a při každém testování se zpracovával jiný typ záznamů, aby se extrakce nezaměřila pouze na jeden specifický typ.

Nejprve se testoval příkaz, který měl za úkol nalézt hodnotu stavu „Approximate time to explant:“. Tato hodnota se vyhledávala v 50 záznamech, které byly typu „Quick Notes“. Hodnota se našla a správně určila u všech 50 záznamů, což znamená, že přesnost i úplnost dosáhly 100 %. Aby přesnost i úplnost byly stoprocentní, musela se optimální vzdálenost pohybovat v intervalu hodnot od 2,78 do 28,8. Pokud byla vzdálenost nižší, tak se snížila úplnost na 14 % (přesnost zůstala stejná). Při hodnotě nižší než 2,08 se nenašel žádný záznam, tudíž obě metriky byly nulové. Pokud byla vzdálenost vyšší, tak se snížila úplnost na 86 % (přesnost byla stoprocetní), a při hodnotě vyšší než 48,0 se úplnost snížila na pouhé 2 % (našla se správně pouze jedna hodnota). Při překročení hodnoty 73,73 nebyla nalezena žádná platná hodnota, tudíž byly obě metriky nulové.

Poté se testovalo 5 příkazů, které měly nalézt hodnoty ve 36 záznamech. Jednalo se o hodnoty stavů: „Last Programmed“, „Noise Response“, „Lower Rate Limit“, „Accelerometer“ a „Sleep Duration“. Vzhledem k tomu, že se našly a správně určily všechny hodnoty, dosáhla přesnost i úplnost 100 %. Aby metriky zůstaly stoprocentní, tak se musel argument optimální vzdálenosti pohybovat v intervalu hodnot od 3,0 do 29,43. V případě, že se nastavila optimální vzdálenost na hodnotu 2 a nižší, se úplnost snížila na 20 % (přesnost zůstala stejná). Ve chvíli, kdy se vzdálenost nastavila na hodnotu 50, se přesnost snížila na 85,31 % a úplnost na 98,05 %. Pokud se nastavila vzdálenost na její maximální hodnotu, přesnost dosahovala 51,82 % a úplnost 62,28 %.

Nakonec se testoval příkaz, který měl extrahoval obrázek měření. Obrázek se extrahoval ze 17 záznamů, které byly typu „Episode Detail Egram“. Jelikož se podařilo správně extrahovat všechny obrázky, tak dosáhla metrika přesnosti i úplnosti 100 %.

Testování prokázalo, že je extrakce záznamů od společnosti Boston Scientific velmi spolehlivá, pokud je vhodně zvolena hodnota argumentu optimální vzdálenosti. Během testování se extrahovalo 247 informací, z toho bylo 230 hodnot a 17 obrázků. Vzhledem k tomu, že byly všechny informace také úspěšně indentifikovány, bylo dosaženo maximální celkové přesnosti i úplnosti, tedy 100 %.

7.3 Společnost Medtronic

Společnost Medtronic vytváří několik typů záznamů, které se generují podle daného vyšetření u každého pacienta. Pacient může mít i několik záznamů, které jsou buď stejného nebo odlišného typu. Záznamy zahrnují jak textové hodnoty stavů, tak i vektorové obrázky grafů měření (viz sekce 4.1). Každý záznam v rámci testování patřil jinému pacientovi a během extrakce se zpracovával s použitím hodnoty 3,25 jako argumetu optimální vzdálenosti (možnost „Small“, viz sekce 6.1). Navíc byly všechny záznamy typu „Quick Look II“, jelikož ostatní typy záznamů neobsahovaly dostatečný vzorek pro testování.

Nejprve se testoval příkaz, který měl za úkol nalézt hodnotu stavu „Acute Phase Completed“ ve 40 záznamech. Jelikož se podařilo najít správnou hodnotu ve všech záznamech, tak byla přesnost i úplnost tohoto testování 100 %. Aby metriky zůstaly stoprocentní, musel se argument optimální vzdálenosti pohybovat v intervalu hodnot od 2,3 do 12,06. Pokud byla vzdálenost nižší, tak byly obě metriky nulové (nenašla se žádná hodnota). Pokud byla vzdálenost vyšší, tak se úplnost začala postupně snižovat: po překročení hodnoty 12,06 se úplnost snížila na 67,5 %, při vzdálenosti 50 na 30% a při maximální vzdálenosti na pouhých 12,5 %. Po celou dobu, co se úplnost snižovala, zůstávala přesnost stoprocentní (všechny extrahované hodnoty se vždy identifikovaly správně).

Následně se testovalo 5 příkazů, které měly nalézt hodnoty ve 20 záznamech. Jednalo se o hodnoty stavů: „Measured P/ R Wave“, „Mode Switch“, „Rate Adaptive AV“, „Rate Response“ a „Capture Threshold“. Vzhledem k tomu, že se hodnoty našly a správně identifikovaly u všech záznamů, dosáhla přesnost i úplnost 100 %. Aby zůstaly metriky stoprocentní, musela se optimální vzdálenost pohybovat v intervalu hodnot od 2,51 do 5,09. Pokud byla vzdálenost nižší, tak se úplnost snížila na 40 % (přesnost zůstala stejná). Při hodnotě nižší než 2,24 byly obě metriky nulové (nedokázala se extrahovat žádná hodnota). V případě, že byla vzdálenost vyšší, úplnost klesla na 84% (přesnost zůstala stoprocentní). Ve chvíli, kdy se vzdálenost nastavila na hodnotu 50, obě metriky klesly na 80,95 %. (ze 100 hodnot se extrahovalo 16 falešně pozitivních a 16 falešně negativních výsledků). Pokud se nastavila maximální vzdálenost, tak se nenašel žádný záznam – obě metriky byly nulové.

Nakonec se testoval příkaz, který měl extrahoval obrázek. Jednalo se o obrázek „Ventricular“, který se extrahoval v 15 záznamech. Jelikož se podařilo správně extrahovat všechny obrázky, tak dostáhla metrika přesnosti i úplnosti 100 %.

Během testování se extrahovalo celkem 155 informací, z nichž bylo 140 hodnot a 15 obrázků. Všechny informace byly nejen správně extrahovány, ale i správně indentifikovány. Proto obě celkové metriky dosáhly 100 %. Testování tedy prokázalo, že je extrakce záznamů od společnosti Medtronic spolehlivá, pokud je správně nastavena hodnota optimální vzdálenosti.

7.4 Souhrn výsledků testování

Následující tabulka přehledně shrnuje výsledky testování extrakce záznamů od jednotlivých společností, kde čísla testování odpovídají pořadí, v jakém byly dané testy provedeny:

Společnost	Testování	Počet dokumentů	Přesnost	Úplnost
Biotronik	**1**	20	100 %	100 %
	2	13	93,85 %	100 %
	3	13	100 %	100 %
Boston Scientific	**1**	50	100 %	100 %
	2	36	100 %	100 %
	3	17	100 %	100 %
Medtronic	**1**	40	100 %	100 %
	2	20	100 %	100 %
	3	15	100 %	100 %

Tabulka 7.1: Výsledky testování

Kapitola 8

Závěr

Cílem této práce bylo vytvořit extrakční nástroj, který zautomatizuje proces vyhledávání informací ve zdravotních záznamech PDF, které vytvořily srdeční stimulátory při kontrole pacienta v nemocnici. Tento nástroj byl úspěšně implementován jako desktopová aplikace, která během procesu extrakce ukládá získané obrázky do specifikované výstupní složky a na konci zapisuje extrahované data do výstupního souboru CSV. Výstupní soubor reprezentuje tyto data formou tabulky, kterou lze otevřít pomocí běžných tabulkových procesorů, jako je například Microsoft Excel.

Pro úspěšnou implementaci extrakčního nástroje bylo nejprve nutné pochopit formát PDF, aby bylo možné nalézt co nejefektivnější způsob získávání dat. Poté se provedl průzkum dostupných knihoven, které se používají pro čtení a zpracování dokumentů v tomto formátu. Na základě tohoto průzkumu byla zvolena knihovna Apache PDFBox, která se následně použila spolu s knihovnou Pdf2Dom pro extrakci a analýzu dat. Dále byl vytvořen návrh extrakčního nástroje, který byl v zápětí implementován pomocí programovacího jazyka Java. Jelikož byl tento nástroj původně navržený jako konzolová aplikace, tak bylo přidáno i grafické uživatelské rozhraní, aby bylo používání aplikace co nejjednodušší. Výsledná aplikace byla nakonec úspěšně otestována na několika záznamech, které pocházely od tří různých společností.

Testování aplikace prokázalo, že je extrakce záznamů velice spolehlivá, pokud je správně nastaveno určení optimální vzdálenosti mezi jednotlivými slovy. Testování záznamů vybraných společností ukázalo, že společnost Biotronik dosáhla celkové přesnosti 97,95 % a úplnosti 100 %, zatímco společnosti Boston Scientific i Medtronic dosáhly maximálních hodnot obou metrik, tedy 100 %. Tato vysoká úspěšnost testování je především výsledkem rozsáhlých možností parametrů příkazu a flexibilního určení vzdálenosti.

Do budoucna by mohla být aplikace rozšířena o vertikální režim extrakce, který by umožnil snadněji získat hodnoty z tabulek, které jsou uspořádány svisle. Kromě toho by se mohla do budoucna vyvinout i desktopová aplikace, která by usnadnila tvorbu příkazového souboru (například pomocí formulářů, do kterých by se zadávaly jednotlivé parametry příkazu). Tento soubor by bylo možné v aplikaci nejen vytvořit, ale také načíst i editovat. Dalším z možných způsobů rozšíření aplikace by mohla být implementace regulárních výrazů, které by umožnily vyhledání hodnot bez ohledu na velikost písmen u parametrů příkazu. Nakonec by bylo užitečné implementovat možnost extrahování více hodnot jedním příkazem, například pomocí substituce.

Literatura

[1] Adobe. What does PDF mean? *Adobe* [online]. 15. března 2006 [cit. 2023-04-13]. Dostupné z: https://www.adobe.com/acrobat/about-adobe-pdf.html.

[2] Adobe. *PDF Reference, sixth edition: Adobe Portable Document Format version 1.7* [online]. 1. vyd. Adobe Systems Incorporated, Říjen 2006 [cit. 2023-04-02]. 1310 s. Dostupné z: https://opensource.adobe.com/dc-acrobat-sdk-docs/pdfstandards/pdfreference1.7old.pdf.

[3] Apache. *Apache PDFBox 2.0.0 API* [online]. 19. března 2016 [cit. 2023-05-9]. Dostupné z: https://pdfbox.apache.org/docs/2.0.0/javadocs/.

[4] Burget, R. *Pdf2Dom* [online]. 7. října 2022 [cit. 2023-05-9]. Dostupné z: https://cssbox.sourceforge.net/pdf2dom/.

[5] G2. Compare Apache PDFBox and iText PDF library/SDK. *G2* [online]. 25. července 2021 [cit. 2023-04-27]. Dostupné z: https://www.g2.com/compare/apache-pdfbox-vs-itext-pdf-library-sdk.

[6] IronPDF. Java PDF Library Comparison (Free & Paid Tools). *IronPDF* [online]. 23. února 2023 [cit. 2023-04-27]. Dostupné z: https://ironpdf.com/java/blog/java-pdf-tools/java-pdf-library-comparison/.

[7] Oracle. *JavaFX 20* [online]. 21. března 2023 [cit. 2023-05-9]. Dostupné z: https://openjfx.io/javadoc/20/.

Příloha A

Obsah paměťového média

Příloha má za úkol popsat obsah paměťového média, které zahrnuje tyto soubory a složky:

- **Externí moduly/** – externí moduly, které jsou nutné pro správné spuštění balíčku JAR
- **Instalační balíčky/** – instalačními balíčky, které umožňují nainstalovat extrakční aplikaci na operační systémy Windows a Linux
- **Přeložené řešení/** – přeložené řešení ve formě komprimovaného archivu JAR
- **Technická zpráva/** – zdrojové kódy technické zprávy (jazyk LaTeX) a její přeložená verze ve formátu PDF
- **Zdrojové kódy/** – zdrojové kódy extrakčního programu (jazyk Java)
- **README.md** – popis paměťového média, návod ke spuštění balíčku JAR

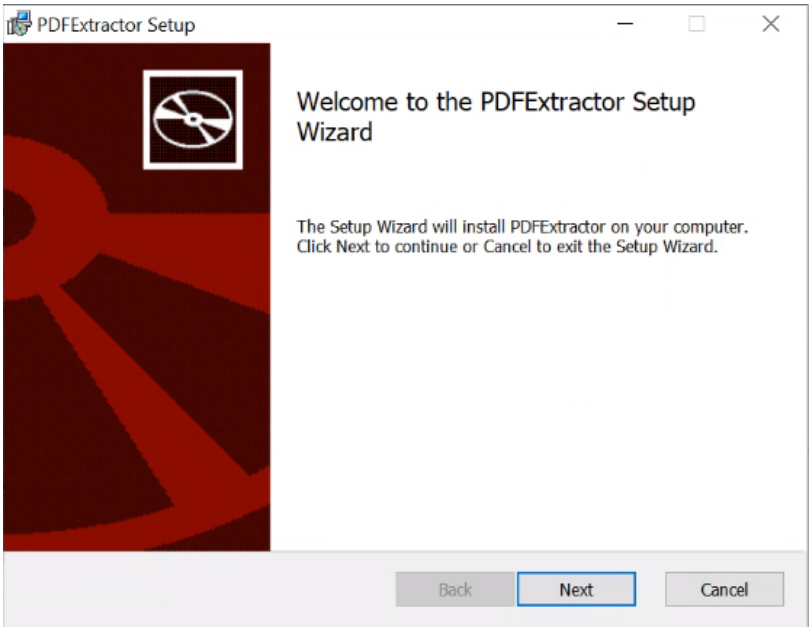
Příloha B

Návod k instalaci aplikace

Příloha má za úkol podrobně vysvětlit postup instalace aplikace pomocí instalačních balíčků, které jsou k dispozici pro operační systémy Windows a Linux. Návod je tedy rozdělený do sekcí podle daného operačního systému.

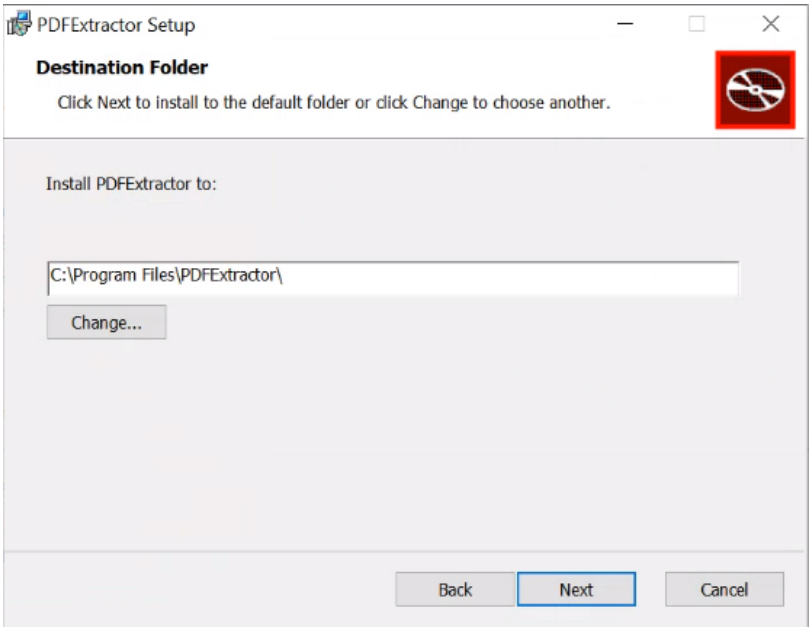
Windows

V adresáři „Instalační balíčky“ se nachází složka „Windows (.msi)“, která obsahuje instalační balíček pro operační systém Windows. Po spuštění tohoto balíčku se zobrazí průvodce instalací, který se ovládá pomocí navigačních tlačítek v dolním panelu aplikace. Tlačítko „Next“ slouží k přechodu na další krok průvodce, tlačítko „Back“ umožňuje přechod do předchozího kroku průvodce a tlačítko „Cancel“ zruší proces instalace a ukončí průvodce. Úvodní okno průvodce instalací slouží k uvítání uživatele a k poskytnutí základních informací o procesu instalace:



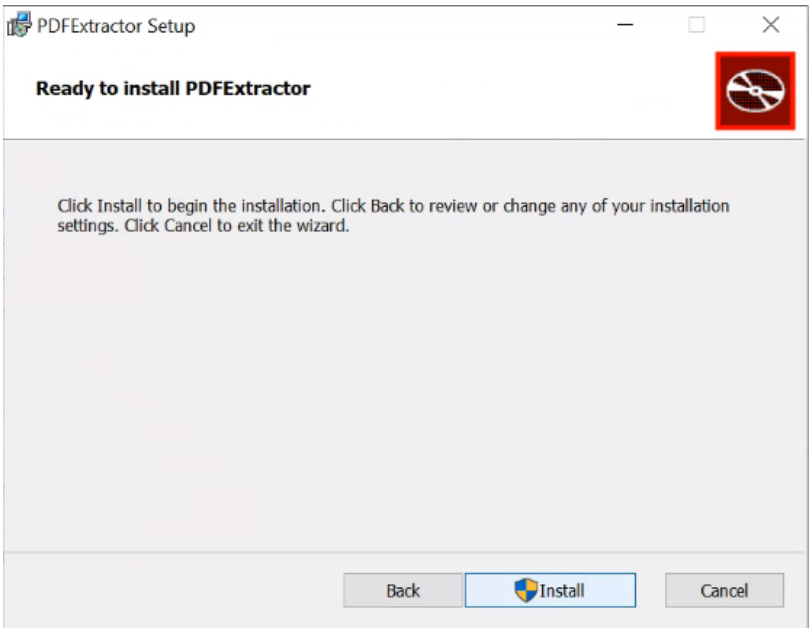
Obrázek B.1: Úvodní okno instalace aplikace

Po stiknutí tlačítka „Next“ se zobrazí následující krok průvodce, ve kterém si může uživatel vybrat adresář, do kterého se aplikace nainstaluje:



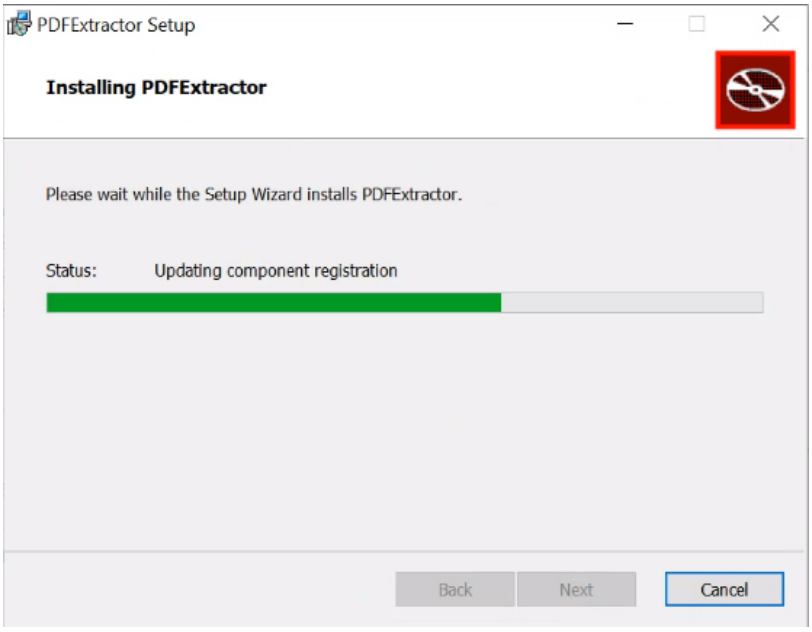
Obrázek B.2: Výběr cílového adresáře

V dalším kroku průvodce se objeví okno, které slouží k potvrzení instalace aplikace:



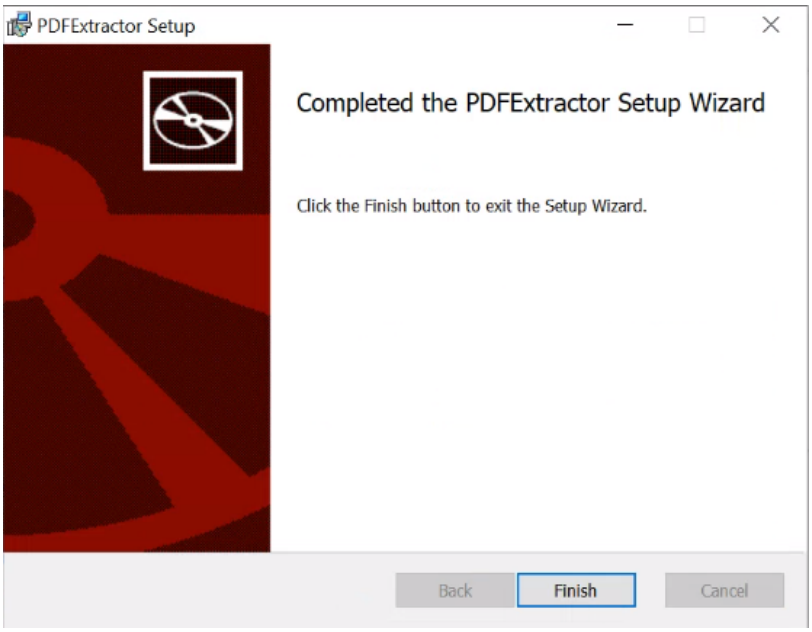
Obrázek B.3: Potvrzení instalace aplikace

Po kliknutí na tlačítko „Install“ se zobrazí okno, které má za úkol ukázat průběh instalace aplikace:



Obrázek B.4: Průběh instalace aplikace

Po nainstalování aplikace se nakonec zobrazí okno, které potvrzuje dokončenou instalaci:



Obrázek B.5: Dokončení instalace aplikace

Po stiknutí tlačítka „Finish“ se průvodce ukončí a aplikace je tedy úspěšně nainstalována. Aplikaci je možné najít na ploše nebo v nabídce „Start“. Tuto aplikaci lze odinstalovat běžným způsobem pomocí možnosti „Programy a funkce“.

Linux

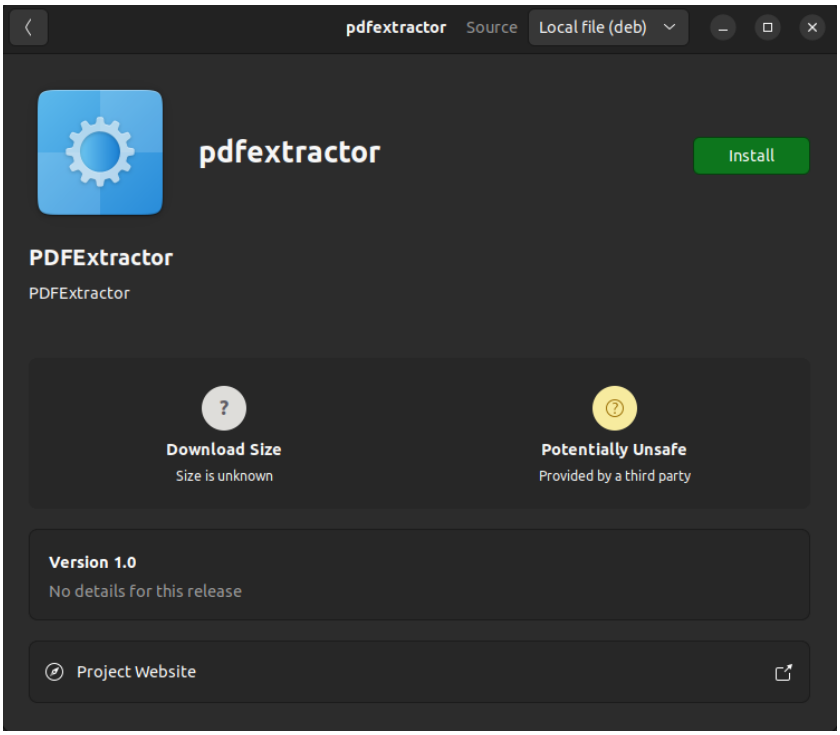
V adresáři „Instalační balíčky“ se nachází složka „Linux (.deb)“, která obsahuje instalační balíček pro operační systém Linux. Instalaci tohoto balíčku lze provést přes terminál pomocí dvojice příkazů:

```
sudo dpkg -i ./pdfextractor_1.0.deb && sudo apt-get install -f
```

nebo příkazu:

```
sudo apt install ./pdfextractor_1.0.deb
```

Další možností instalace balíčku je přes aplikaci Software Install, ve které se produkt nainstaluje po kliknutí na zelené tlačítko „Install“.



Obrázek B.6: Instalace produktu v aplikaci Software Install

Nainstalovanou aplikaci je možné najít v nabídce „Zobrazit aplikace“. Tuto aplikaci lze odinstalovat přes terminál pomocí příkazu:

```
sudo apt purge pdfextractor
```

nebo pomocí aplikace Software Install, která umožňuje odinstalovat produkt pomocí červeného tlačítka s ikonou bílého koše, které se nachází na místě, kde bylo původně umístěno tlačíko „Install“ před instalací produktu.