Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Generátor a parser formulářů recenzí příspěvků na konferenci TSD

Plzeň 2019 Vojtěch Danišík

Místo této strany bude zadání práce.

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.
V Bakalářské práci jsou použity názvy programových produktů, firem apod., které mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

V Plzni dne 1. dubna 2019

Vojtěch Danišík

Poděkování

Děkuji panu Ing. Kamilu Ekšteinovi Ph.D. za ochotu při vedení bakalářské práce a rady s jejím vypracováním.

Abstract

Generator and Parser of Submission Review Forms for the TSD Conference. The goal of this thesis is to create PHP module, which will be easily integrate into existing information system for managing TSD conference. First part of the thesis explains standard PDF format and forms created in PDF. Subsequently, there are described existing PHP libraries for generating and parsing PDF form of scientific contribution. Second part of the thesis focuses on the implementation of selected libraries into TSD conference web portal. The module was tested by conference system users and multiple PDF browsers were used. Test results are part of this thesis.

Abstrakt

Cílem bakalářské práce je vytvořit jednoduše integrovatelný PHP modul do již existujícího informačního systému pro správu konference TSD. První část práce důkladně vysvětluje standardní formát PDF a formuláře vytvořené v PDF. Následně jsou popsány existující PHP knihovny pro generování a zpracování PDF formuláře daného vědeckého příspěvku. Druhá část práce se věnuje implementaci vybraných knihoven do webového portálu konference TSD. Modul byl otestován uživateli konferenčního systému a bylo použito více PDF prohlížečů. Výsledky testování jsou součástí této práce.

Obsah

1 Úvod

TSD (**T**ext, **S**peech and **D**ialogue) je konference zabývající se problémy zpracování přirozeného jazyka. Mezi nejčastěji probíraná témata se řadí například rozpoznávání řeči, modelování řeči,textové korpusy, značkování textu a mnoho dalších. Konference se koná každý rok v září a místo konání se střídá mezi Brnem (pořadatelem je Fakulta informatiky Masarykovy Univerzity) a Plzní (pořadatelem je Fakulta aplikovaných věd Západočeské univerzity v Plzni). Tento rok bude konference organizována právě Západočeskou univerzitou, a poprvé se bude konat za hranicemi České Republiky, přesněji na Slovinsku ve městě Ljubljana.

Ke každé konferenci existuje webový portál vytvořený daným pořadatelem, na nějž jsou od uživatelů nahrávány vědecké příspěvky. Tyto příspěvky jsou poté hodnoceny recenzenty (převážně členy programového výboru) formou online formuláře a na základě konečného hodnocení jednotlivých parametrů a na doporučení recenzentů jsou tyto příspěvky schváleny organizátorem a mohou být prezentovány na konferenci, nebo jsou zamítnuty z důvodu nedostatečného hodnocení. Modul vytvářený autorem bude implementován do webového portálu organizovaný Fakultou aplikovaných věd.

Cílem této práce je prostudovat strukturu PDF formátu, který je pro vytváření editovacích formulářů nejvhodnější a byl vybrán zadávajícím jako standard, tak i funkcionalitu volně dostupných PHP knihoven pro generování a parsování PDF souborů obsahujících editovatelný formulář, aby existovala možnost ohodnocení daného vědeckého příspěvku i v místech, kde není dostupné internetové připojení, neboli off-line. Tento PDF soubor musí obsahovat hodnotící formulář se všemi hodnotícími parametry, text vědeckého příspěvku doplněný o vodoznak. Pro generování a parsování musí být použity výhradně knihovny v jazyce PHP, jelikož není vhodné využívat aplikace třetích stran spustitelné z terminálu. Modul musí být nezávislý na platformě a lze ho upravovat v jakémkoliv PDF prohlížeči nezávisle na verzi PDF. Před vytvořením modulu na testovací verzi webového portálu bude potřeba projít zdrojové soubory webového portálu pro seznámení s již existujícími funkcionalitami a zařadit do portálu i náš modul. Z dřívějších let je zde naimplementován totožný modul pro generování a parsování PDF souborů, bohužel tento modul nesplňuje veškeré body zadání právě z důvodu použití nevhodného parseru.

2 Formát PDF

Formát **PDF** (**P**ortable **D**ocument **F**ormat) je souborový formát vyvinutý společností Adobe v roce 1992. PDF formát byl vyvinut za účelem konzistentní prezentace dokumentů (spustitelné na více zařízeních a různých platformách). Díky konzistenci lze dosáhnout toho, že PDF soubor vytvořený a uložený v systému Windows bude zobrazen totožně na systémech Mac, na všech distribucích Linuxu nezávisle na použitém PDF prohlížeči (Adobe Reader, Foxit a další).

V PDF souboru lze uchovávat velice širokou škálu dat, včetně formátovaného textu, vektorové grafiky a rastrových obrazů, nebo například informace o rozložení, velikosti a tvaru stránky. Informace definující umístění jednotlivých položek (jsou zde zahrnuty i editovací objekty pro formuláře) na stránce jsou zde uloženy též. Do dokumentu lze ukládat i metadata. Metadata jsou informace uložené v hlavičce souboru a lze do nich uložit název dokumentu, autora dokumentu, předmět a klíčová slova. Je zde možnost uložit heslo, aby byl dokument přístupný pouze autorizovaným uživatelům. Všechny tyto informace jsou uloženy ve standardním formátu [5, 9].

2.1 Objekty

PDF Objekty jsou základním stavebním kamenem pro uchovávání dat v dokumentu. Množinou PDF objektů lze reprezentovat bitmapové a vektorové objekty, barevné prostory, text, fonty aj. [10].

2.1.1 Základní objekty

V PDF můžeme najít celkem 5 základních objektů:

- **Celá a reálná čísla** – Celá čísla jsou reprezentována jako jedno nebo více desetinných čísel z rozsahu 0..9 se znaménkem + nebo - před číslem. Reálné číslo je celé číslo rozšířené o desetinnou část s ideálně jedním desetinným číslem (reálná čísla nelze popsat exponenciálním způsobem). Přesnost a rozsah celých a reálných čísel je definován jednotlivými implementacemi PDF. V některých implementacích platí pravidlo které přetypuje celé číslo na reálné po přesáhnutí předem daného rozsahu.

- **Řetězce** – Řetězec je reprezentován jako množina po sobě jdoucích bytů vepsaných mezi jednoduché závorky. Jako příklad lze uvést: *(Hello, World!)*. Pro zobrazení zpětného lomítka a jednoduchých závorek je potřeba před tyto znaky přidat zpětné lomítko pro jejich správné zobrazení v dokumentu. V tabulce 2.1 lze vidět využití zpětného lomítka pro zobrazení odřádkovacích znaků:

Sekvence znaků	**Význam**
$\backslash n$	*Line feed (LF)*
$\backslash r$	*Carriage return (CR)*
$\backslash t$	*Tab*
$\backslash b$	*Backspace*

Tabulka 2.1: Odřádkovací sekvence znaků

  Řetězce můžou být reprezentovány i jako sekvence hexadecimálních čísel vložených mezi znaky $<$ a $>$.
  Jako příklad lze uvést: *<4F6EFF00> →0x4F, 0x6E, 0xFF, 0x00.*

- **Jména** – Jméno je reprezentováno jako sloučení zpětného lomítka a řetězce (př. */Jmeno*). Za jméno se pokládá i zpětné lomítko bez řetězce. Pokud bychom potřebovali nadefinovat v dokumentu jméno, jenž bude obsahovat mezery, musíme do řetězce přidat i sekvenci znaků **#20**, jelikož v ASCII tabulce je hexadecimální hodnota 20 vyjádřena jako prázdný znak. Jména jsou case-sensitive, proto */Jmeno* a */jmeno* jsou rozdílná jména. Jeho využití v PDF je prosté, slouží jako klíče ve slovnících a pro definice složitějších (vícehodnotových) objektů.

- **Boolean (pravdivostní) hodnoty** – Logické hodnoty **true**/**false** a vyskytuje se v jednotlivých záznamech ve slovníku jako příznak.

- **Hodnota null** – Nabývá hodnot *f* (free) nebo *n* (use) a vyjadřuje, zda je objekt vyobrazen v dokumentu.

2.1.2 Složené objekty

Složený objekt je takový objekt, který obsahuje seřazenou/neseřazenou množinu základních objektů i množinu složených objektů.

- **Pole** – Pole je v PDF reprezentováno jako seřazená množina základních i složených PDF objektů (v poli může být uložen například i slovník nebo pole) nezávisle na typech (v poli lze uchovávat například řetězec a číslo zároveň). Hodnoty pole jsou vloženy mezi znaky [a].

- **Slovníky** – Slovník se skládá z množiny dvou hodnot: klíče a hodnoty, pomocí kterých se slovník namapuje. Klíč je reprezentován pomocí **jména**, zatímco hodnota může být kterýkoliv PDF objekt, povoleny jsou i slovníky nebo pole. Slovníky jsou uloženy mezi znaky « a ».
- **Datové proudy** – Datové proudy slouží především pro uložení binárních dat a skoro ve všech případech jsou zkomprimovány různými kombinacemi algoritmů, které jsou popsány v kapitole 2.2, proto datové proudy musí být zároveň i nepřímým odkazem. Skládají se ze slovníků a části binárních dat. Slovník je využit pro ukládání parametrů binárních dat, jako například délka binárních dat aj.

2.1.3 Linkovací objekty

PDF objekty můžou být různě velké. Pokud je objekt až příliš veliký, pak jsou v kódu dokumentu využity nepřímé odkazy. Na obrázku 2.1 si lze všimnout využití nepřímých odkazů ve slovníku.

```
<<
/Resources 10 0 R <--- znak R reprezentuje nepřímý odkaz na objekt s ID 10 a gen. číslem 0
/Contents [4 0 R]
>>
```

Obrázek 2.1: Ukázka nepřímého odkazu

2.2 Komprese dat v PDF

PDF soubory moho být poměrně kompaktní, o mnoho menší než ekvivalentní postscriptové soubory. Tato vlastnost je dosažena nejen lepší strukturou dat, ale i díky kompresním algoritmům, které jsou velice efektivní. Typ komprese dat PDF souboru lze zjistit pomocí textového editoru, který dokáže zpracovat binární data, vyhledáním klíčového slova **/Filter**. Níže jsou popsány kompresní algoritmy využívané v PDF [3].

- **CCITT G3/G4** – Algoritmus je bezeztrátový a využívá se pro vykreslení černobílých obrázků.
- **JPEG** – JPEG algoritmus může být jak ztrátový, tak i bezeztrátový. V Acrobatu se využívá pouze ztrátový s 5 stupni komprese. Využívá se pro barevné a šedotónové obrázky.

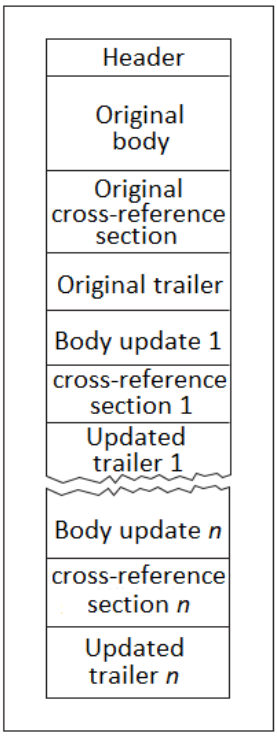
- **JPEG2000** – Rychlejší algoritmus na bázi JPEGu. Víceméně se nepoužívá, jelikož není kompatibilní se staršímy systémy a vysokými nároky na procesor.
- **Flate** – Bezeztrátový algoritmus, vychází z kompresních algoritmů LZ77 a Huffmanova kódování.
- **JBIG2** – Alternativní k CCITT. V Dnešní době se nevyužívá z důvodu pomalejší komprese než je u jeho protějšku.
- **LZW** – Komprimací LZW algoritmem lze dosáhnout až o polovinu menší velikosti díky komprimaci veškerého textu a operátorů v souboru.
- **RLE** – Bezeztrátový algoritmus pro vykreslování černobílých obrázků. Nahrazen efektivnějším algoritmem CCITT.
- **ZIP** – Bezeztrátový algoritmus, učinější než jeho protějšek LZW.

2.3 Vnitřní struktura PDF

Vnitřní reprezentace PDF souboru je rozdělena na sekce, které jsou znázorněny na obrázku 2.2.

Z obrázku lze vyčíst, že se zde vyskytují 4 hlavní sekce: *Header, Body, Cross-reference a trailer*. Díky jedné z vlastností PDF formátu se při úpravě souboru staré sekce neodstraní, místo toho se pouze na jeho konci vytvoří nové sekce [7].

- **Header** – Hlavička souboru je uložena na první řádce, obsahující primárně použitou verzi PDF.
- **Body** – V těle dokumentu jsou uložena veškerá data objektů reprezentující celý dokument. Objekty jsou referencovány v tabulce Cross-reference z důvodu rozprostření částí dat patřících k danému objektu po celé sekci.Pokud se v dokumentu vyskytuje jeden obrázek/zvukový záznam vícekrát než jednou, tak se poté všechny objekty reprezentující obrázky odkazují na jednu množinu dat [6].
- **Cross-reference table** – Jinak nazývána **xref** je tabulka obsahující reference na veškeré objekty uložené v těle a v kódu začíná řetězcem *xref*. Reference uložená v tabulce je reprezentována na 2 řádcích pomocí řetězce a skládá se z 5 částí o celkové velikosti 20 bytů včetně oddělovačů *CRLF*:



Obrázek 2.2: Interní struktura PDF souboru

```
%PDF-1.4%âãÐÓ <--- hlavička souboru
```

Obrázek 2.3: Ukázka hlavičky

```
4 0 obj     <--- start objektu
<</Filter /FlateDecode /Length 1882>>
stream      <--- start dat
xśíśßoÚ6□ĄąWżî□ °‡´ŘÂŠ□ő□□öÐ
iÐ□CÓĆŘ□Ö=ŘŽăx‰“ĹVšv ýŽ§□–Ďrě¸>#0□#’,Qüňî H~
(S’üµ-_ĽŃRk□□□lźËŁv+ß□´□\□·^□źj9□Ř÷(Nä}
endstream   <--- konec dat
endobj      <--- konec objektu
```

Obrázek 2.4: Ukázka dat objektu

– *Číslo objektu* – Jednoznačný číselný identifikátor objektu.
– *Počet subobjektů* – Počet částí daného objektu vyskytujícího se v dokumentu.
– *Začátek objektu* – Tvoří většinu řetězce (prvních 10 bytů) a určuje offset od začátku PDF dokumentu až po začátek daného objektu.
– *Generační číslo objektu* – Vyjadřuje jak často byl objekt vymazán při úpravě dokumentu.

– *Identifikátor využití* – Nabývá hodnot *f* (free) nebo *n* (use) a vyjadřuje, zda je objekt vyobrazen v dokumentu.

```
xref   <--- start tabulky
0 1    <--- ID objektu a počet subobjektů
0000000001 65535 f
31 1
0000423765 00000 n
```

Obrázek 2.5: Ukázka jednoduché xref tabulky

- **Trailer** – Trailer je seznam informací, ze kterých lze snadno zjistit například velikost nebo umístění xref tabulky. Trailer může obsahovat tyto elementy:
  - *Size* – Udává počet objektů referencovaných v xref tabulce.
  - *Prev* – Offset od začátku dokumentu k předchozí xref tabulce.
  - *Root* – Odkazuje na objekt obsahující informace ohledně katalogu xref tabulek.
  - *Encrypt* – Specifikuje komprimující algoritmus použití pro daný dokument.
  - *Info* – Obsahuje dodatečné informace ohledně katalogu xref tabulek.
  - *ID* – 2-bytový identifikátor PDF dokumentu.
  - *XrefStm* – Offset od začátku dokumentu až k dekódovanému xref streamu. Využívá se pouze u hybridně-referencovaných souborů pouze tehdy, kdy hledaný objekt není nalezen v xref tabulce (před tím, než se volá element *Prev*).

```
trailer         <--- start traileru
<<
/Size 742       <--- velikost xref tabulky
/Root 741 0 R   <--- odkaz na objekt odkazující na objekt katalogu xref tabulek
/Info 740 0 R   <--- odkaz na informační slovník xref tabulek
/ID [<009feb05c3e899ac1d26612f86bb56aa> <009feb05c3e899ac1d26612f86bb56aa>] --->
<--- indentifikátor souboru
>>
startxref
408764          <--- offset tabulky xref
%%EOF
```

Obrázek 2.6: Ukázka traileru

2.4 PDF formuláře

Pod pojmem formulář si lze představit dokumenty, které od svých uživatelů vyžadují vyplnění určitých údajů. Mezi nejznámější dokumenty lze například uvést daňová přiznání, oznamovací tiskopisy, dotazníky, složenky aj. Ruční vyplňování i jejich následné zpracování bývá obvykle pracné a zdlouhavé, proto je v dnešní době výhodnější využívat interaktivní elektronické formuláře. Základní výhoda těchto formulářů spočívá ve snažším vyplňování, zpracování lze jednoduše zautomatizovat a také se díky elektronické podobě zvedne úspora papíru a financí vynaložených na tisk formulářů. Mezi nejčastější formuláře, které lze potkat na internetu, jsou ve formátu HTML a lidé se s nima setkávají každodenně (ať už to jsou jednoduché přihlašovací formuláře stránek nebo různé dotazníky na určitá témata). Nevýhoda těchto formulářů je v jejich závislosti na internetovém připojení.

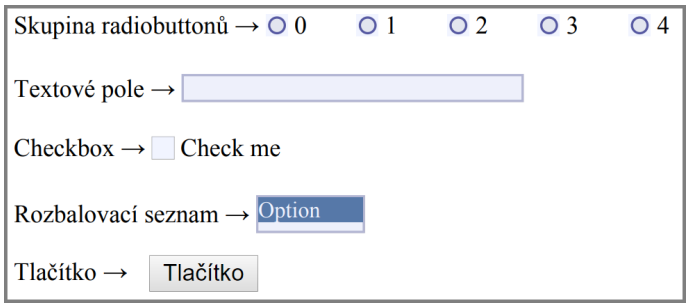
Proto firma Adobe přišla se svým řešením, interaktivním PDF formulářem, který lze vyplňovat kdekoliv nezávisle na internetovém připojení. Mezi další výhody PDF formulářů patří elektronický podpis (lze s ním potvrzovat smlouvy z domova), zabezpečení (dokument se otevře až po zadání správného hesla, neautorizovaným uživatelům je přístup zamítnut) aj. Tyto formuláře obsahují stejné interaktivní prvky jako mají HTML formuláře viz kapitola 2.4.1. Pro generování PDF formulářů lze využít kterýkoliv programovací jazyk, který podporuje práci s PDF soubory (například *PHP*, *Java*), produkty firmy Adobe (například *Adobe Acrobat*) nebo lze použít i nekomerční aplikace typu *TeX* nebo *pdfmarks*.

Tvorba formulářů je jedna věc, druhá věc je jejich zpracování (získání dat vyplněných od uživatele). Mezi nejznámější nástroje pro zpracování vyplněných dat patří určitě nástroj *FDF Toolkit* od firmy Adobe. Tento nástroj je zcela zdarma a umožňuje vytvářet orientovaná řešení pro zpracování dat v jazycích *C/C++*, *ActiveX*, *Java* a *Perl*. Jsou-li data odeslána v HTML, lze k jejich zpracování využít nástroje určené pro formáty *CGI*, *PHP* aj. [1].

2.4.1 Základní prvky

Jednotlivé formulářové prvky mohou mít přiřazeny nejrůznější atributy a jsou reprezentovány jako PDF objekty. Tyto atributy lze rozdělit do následujících skupin: **Vzhled** (definovaný vzhled prvku), **Akce** (po kliknutí na prvek se provede daná akce), **Formát** (typ fontu textu aj.), **Ověřování dat** (akceptovatelný formát vstupu) a **Výpočty** (matematické operace použité při práci se vstupy z jiných prvků) [2].

Ve formuláři se může vyskytovat až 7 různých prvků viz obrázek 2.7:



Obrázek 2.7: Základní prvky vyskytující se v PDF

- **Textové pole** – Slouží k vyplnění textu. Jako příklad lze uvést například klasický přihlašovací formulář, který obsahuje 2 textové pole, jedno pro zadání uživatelského jména a druhé (upravené, místo textu se zobrazují pouze speciální znaky pro zakrytí zadaného textu) pro zadání hesla. Při vytváření lze předvyplnit toto pole výchozím textem, lze omezit maximální počet znaků vkládaných do pole a jejich formát. Pole může být uzamčeno a může sloužit i jako informační položka.
- **Tlačítko** – Účel tohoto prvku je spouštění zvolených akcí, které se po kliknutí na tlačítko mají provést, tudíž se označují jako hlavní řídící prvek každého formuláře. Tlačítko se skládá převážně z ikonky a textu, případně mu může být nastaven externí obrázek.
- **Seznam** – Zobrazuje seznam položek v rolovacím okně, ze kterého lze současně označit jeden nebo více položek (s využitím klávesy *Shift* nebo *Ctrl*). Pro seznamy lze nastavit filtry, které budou seznam třídit podle předem daných parametrů a zobrazí položky na základě těchto filtrů.
- **Kombinované pole** – Kombinované pole je ve své podstatě seznam prvků, ale liší se ve výběru položek. V kombinovaném poli lze vybrat pouze jeden aktivní prvek, ostatní budou zakázány. Platí zde pravidla s tříděním pprvků odle filtrů.
- **Přepínací tlačítka** – Jinak označované jako **radio-buttony** je seznam tlačítek, ve kterém uživatel vybírá pouze jednu z nabízených hodnot.
- **Zaškrtávací pole** – Jedná se o indikační prvek umožňující současný výběr více položek. V odborném prostředí se označuje jako **Checkbox**.
- **Podpis** – Pomocí tohoto prvku lze do dokumentu vložit elektronický podpis.

3 Knihovny

V programování můžeme knihovnu definovat jako kolekci předem zkompilovaných procedur, funkcí (v objektovém programování i třídy a objekty), konstant a datové typy. Knihovna by měla být následně i dobře zdokumentována pro její snadnější zakomponování do již existujících modulů (při používání nezdokumentovaných knihoven se musí provádět takzvaný reverse engineering pro zjištění všech procedur a funkcí, nebo vyhledávat už hotová řešení na internetu).

Knihovny jsou z technického hlediska rozděleny do 2 skupin, které se následně rozdělují do 2 podskupin:

- **Rozdělení z hlediska způsobu propojení s programem:**
  - *Statická knihovna* – Zdrojový kód knihovny je v průběhu překládání zkopírován do výsledného programu pomocí kompilátoru. Největší výhoda statických knihoven spočívá v jistotě, že všechny potřebné knihovny budou přítomny ve výsledném programu, proto nikdy nemůže nastat situace nazvaná *dependency hell (DLL Hell)*, která značí nepřítomnost jedné nebo více knihoven, které jsou využívány jinou knihovnou, nebo také může značit nadbytečné závislosti knihoven, které nejsou ve výsledku využity.
  - *Dynamická knihovna* – Oproti statickým knihovnám, zdrojové kódy dynamických knihoven nejsou zakomponovány ve výsledném programu, ale pomocí linkeru jsou vytvořeny záznamy na funkce použité v programu, které jsou následně uloženy do tabulky symbolů vyskytující se ve výsledném programu.
- **Rozdělení z hlediska sdílení kódu mezi programy:**
  - *Sdílená knihovna* – Zdrojový kód sdílených knihoven je možné sdílet mezi více programy. Tímto způsobem jsou efektivně sníženy nároky na velikost operační paměti, protože úseky kódu využívané více procesory jsou uloženy ve sdílené paměti (namapovány do adresních prostorů všech procesů, které ji využívají).
  - *Nesdílená knihovna* – Nesdílené knihovny neumožňují sdílet úseky kódu více procesorům z důvodu kopírování kódu z knihoven do souborů při linkování souborů.

3.1 PHP Knihovny pro generování PDF

3.1.1 FPDF

Free PDF (zkráceně FPDF) je knihovna psána v jazyce PHP a slouží pro generování PDF souborů bez využití externích programů, jejíž zdrojové kódy jsou volně dostupné a lze je modifikovat pro potřeby uživatele. Díky volně dostupným zdrojovým kódům lze na oficiální stránce nalézt velice užitečná rozšíření této knihovny. Mezi hlavní funkcionality patří například automatické zalomování stránek, komprese stránek, hyperlinky a mnoho dalších. Bohužel zde nejdou vytvářet interaktivní formuláře, proto nelze tuto knihovnu použít pro vyvíjený modul.

3.1.2 dompdf

Knihovna **dompdf** má za úkol převést HTML kód do PDF souboru pomocí jazyku PHP. Své funkcionality dosáhne ve dvou případech: za pomoci externí knihovny *PDFlib* (placená), nebo pomocí třídy *R&OS CPDF* (sepsaná uživatelem *Wayne Munro*). Mezi hlavní funkcionality patří podpora 8/24/32 bitových obrázků (bitmapové a jpeg), externí CSS styly uložené na jiných stránkách/ftp, podpora atributů v HTML 4.0 verzi aj. Bohužel **dompdf** není vhodná pro vyvíjený modul z důvodu neschopnosti vytvářet interaktivní formuláře (za podmínky využití přiložené pomocné třídy místo knihovny PDFlib).

3.1.3 TCPDF

Knihovna **TCPDF** je open-source PHP knihovna sloužící pro práci s PDF soubory. Její vývoj odstartoval už v roce 2002 kdy vznikla jako odnož knihovny FPDF. Díky její rozmanitosti funkcí pro vytváření PDF souborů si jí oblíbilo mnoho uživatelů a je využívána i na mnoha webových portálech. Mezi hlavní funkcionality lze zařadit například podporu UTF-8 kódování, komprese stránek, vkládání zdrojových souborů, šifrování celého dokumentu, vkládání čárových kódů aj. Protože je psána pouze v jazyce PHP a nevyužívá žádné externí knihovny, pak ji lze brát jako vhodnou knihovnu pro vyvíjený modul.

3.1.4 HTML2FDPF

HTML2FPDF vychází z již existující knihovny FPDF a má za úkol převést HTML kód a vytvořit z něj PDF soubor. Bohužel tato knihovna už není dále

vyvíjena, ale stále funguje na všech verzích PHP. Mezi hlavní nevýhody lze zařadit nemožnost vytvářet interaktivní formuláře, proto ji nelze využít pro vyvíjený modul.

3.1.5 mPDF

Jedna z dalších PHP knihoven pro generování PDF souborů **mPDF** je velice chytrá knihovna obsahující množství funkcionalit. Byla vyvinuta z již existujících knihoven *FPDF* a *HTML2FPDF*, kdy převádí HTML kód a vytváří z něj PDF soubor se všemi HTML prkvy (až na výjimky jako například nemožnost zobrazit tlačítko). Oproti knihovnám, ze kterých *mPDF* vychází, je tvorba PDF výrazně pomalejší z důvodu unicode fontů (pokud jsou použity) a zároveň z důvodu procesního času při tvorbě souboru. Jako hlavní funkcionality se řadí vytváření interaktivních formulářů, UTF-8 kódování pro HTML kód, vkládání vodoznaku do stránek a mnoho dalšího. Z důvodu možnosti vytvářet interaktivní formuláře a nezávislosti na externích programech lze tuto knihovnu využít pro vyvíjený modul.

3.2 PHP Knihovny pro zpracování PDF

3.2.1 pdf-to-html

Knihovna **pdf-to-html** má za úkol překonvertovat veškerý obsah PDF souboru do HTML struktury, ze které lze snadno vyextrahovat obsah souboru a předat ho ke zpracování. Požadavky pro správné fungování této knihovny je mít v PHP konfiguraci mít povolen přístup k příkazové řádce systému a mít na serveru nainstalovaný *Poppler* (knihovna napsaná v jazyce C++ sloužící k renderování PDF dokumentů) [8]. Protože je tato knihovna závislá na knihovně (Poppler), pak ji nelze brát jako vhodnou pro vyvíjený modul.

3.2.2 TCPDF parser

TCPDF parser je součást knihovny **TCPDF** (viz 3.1.3), která se soustředí na zpracování PDF souboru. Pro svůj běh nepotřebuje žádné externí knihovny a je psána pouze v jazyce PHP, ale stále se nachází ve fázi vývoje a při jejím použití nemusíme vždy dojít ke správnému výsledku. Proto z tohoto důvodu není nejvhodnější pro vyvíjený modul a bude lepší se ohlédnout po jiné knihovně.

3.2.3 PDF Parser

PDF Parser je další z mnoha knihoven sloužících pro zpracování PDF souborů. Tato knihovna je založena na již existující knihovně **TCPDF parser**, která je navíc doplněna o nové funkcionality jako je například extrakce metadat a komprimovaných souborů aj. Na oficiálních stránkách lze najít demo verzi, která demonstruje funkčnost, kdy po nahrání kteréhokoliv PDF souboru se na stránkách zobrazí data extrahovaná z nahraného souboru. Vzhledem k tomu, že PDF Parser je plně vyvinutá knihovna využívaná na mnoha webových portálech pro zpracování PDF souborů, pak ji lze brát jako vhodnou knihovnu pro vyvíjený modul.

3.2.4 php-pdftk

Nástroj **PDF Toolkit** (zkráceně pdftk) je multiplatformní nástroj pro manipulaci s PDF soubory, který navazuje na starší verzi nástroje **iText library**. PDF Toolkit lze najít ve třech verzích. Mezi neplacené verze patří *PDFtk Server*, což je open-source tool v příkazové řádce a verze *PDFtk Free*, která je úplně zdarma), zatímco mezi placené verze patří verze *PDFtk Pro* (patří mezi proprietární software, jehož zdrojové soubory nejsou volně dostupné). Pomocí tohoto nástroje lze oddělovat/ spojovat/šifrovat PDF soubory, měnit jeho vlastnosti, metata, vyplňovat formuláře *FDF daty* (Forms Data Format) a mnoho dalších funkcionalit [4]. Díky rozsáhlé funkcionalitě byla vyvinuta knihovna v PHP s názvem **php-pdftk**, pomocí které lze využívat veškerou funkcionalitu tohoto nástroje v jazyce PHP. Bohužel díky závislosti na externím programu ji nelze brát jako vhodnou pro vyvíjený modul.

3.2.5 pdftotext

pdftotext je open-source nástroj spouštěný přes příkazovou řádku využívaný k převodu PDF souboru do prostého textu využívající knihovnu *Poppler*. Je volně dostupný v Linuxových distribucích (v některých distribucích je součástí systému), zatímco pro Windows ho nalézt jako součást programu *Xpdf*. Belgická firma *Spatie* vyvinula open-source PHP knihovnu využívající tento nástroj, aby byl dostupný i v jazyce PHP. Protože tato knihovna stejně jako **pdf-to-html** využívá Poppler, pak ji nelze brát jako vhodnou pro vyvíjený modul.

3.3 Závěr průzkumu

Autor této práce provedl rozsáhlý průzkum zaměřující se na volně dostupné PHP knihovny pro generování a zpracování PDF souborů. Co se týče PHP knihoven pro generování interaktivních formulářů, pak zde existují dvě velice slušné knihovny **mPDF** a **TCPDF**, které dokáží splnit veškeré požadavky zadávajícího. Proto při vývoji modulu budou použity obě dvě a následně bude vybrána ta nejvíce vyhovující zadání. U PHP knihoven zpracovávající PDF soubory to tak není, většina knihoven využívá pro svojí funkcionalitu externí programy/knihovny psané v jiném programovacím jazyku a jsou převážně spouštěny z příkazové řádky, což silně odporuje požadavkům zadávajícího. Jediná knihovna splňující tyto požadavky byla **PDF Parser**, proto bude použita při vývoji modulu.

4 Návrh modulu

4.1 Vzhled PDF dokumentu

Při návrhu výsledného vzhledu celého dokumentu je potřeba klást důraz hlavně na co nejpřesnější vyobrazení webového formuláře do PDF souboru. Vzhled webového formuláře je zobrazen na obrázku 4.1.

4.1.1 Záhlaví

Záhlaví dokumentu by mělo obsahovat jednoznačný identifikátor hodnotícího příspěvku doplněný o název vědeckého příspěvku, který bude hodnocen. Rozhodně by zde nemělo chybět ani logo konference TSD (ideálně ve vektorovém formátu). V případě, že název vědeckého příspěvku bude zasahovat do loga, tak bude název zkrácen na fixní velikost.

4.1.2 Titulek

Titulek dokumentu by měl uživateli jednoznačně říct, který vědecký příspěvek hodnotí (ideálně zobrazit název i identifikátor). V titulku by nemělo chybět ani jméno posuzovatele a doplňující informace ohledně vyplňování formuláře, případně ho informovat o nedostatcích a omezeních aktuálně vygenerovaného PDF dokumentu.

4.1.3 Formulář

Vzhled formuláře by se měl v ideálním případě shodovat s webovým formulářem. První část formuláře obsahuje stupnicové hodnocení, zatímco druhá je spíše slovní formou. Po konzultaci s vedoucím práce bylo usouzeno, že element **Combo box** reprezentující stupnicové hodnocení parametru bude nahrazen skupinou elementů **Radio button**. Důvod tohoto rozhodnutí spočíval v lepším zobrazení hodnot na stupnici, doplněn o neschopnosti PHP generátorů zobrazit pěkný **Combo box** s bezproblémovou funkcionalitou (špatná manipulace při vybírání hodnot). Pro uložení doplňujícího textu byl použit element **Text area** pro případné poznámky ohledně stavu a obsahu vědeckého příspěvku. Element popisující *Review state* a tlačítko *Save review* nebudou do formuláře vloženy, jejich funkcionalita není potřebná pro modulem vytvářený formulář.

4.1.4 Hodnocený vědecký příspěvek

Vygenerovaný dokument by měl hlavně sloužit pro vyplňování hodnotícího formuláře off-line a ideálně by měl obsahovat i veškerý obsah hodnoceného vědeckého příspěvku, aby měl posuzovatel možnost kdykoliv nahlédnout na jeho obsah. Tento příspěvek bude vložen na konec PDF souboru.

4.1.5 Vodoznak

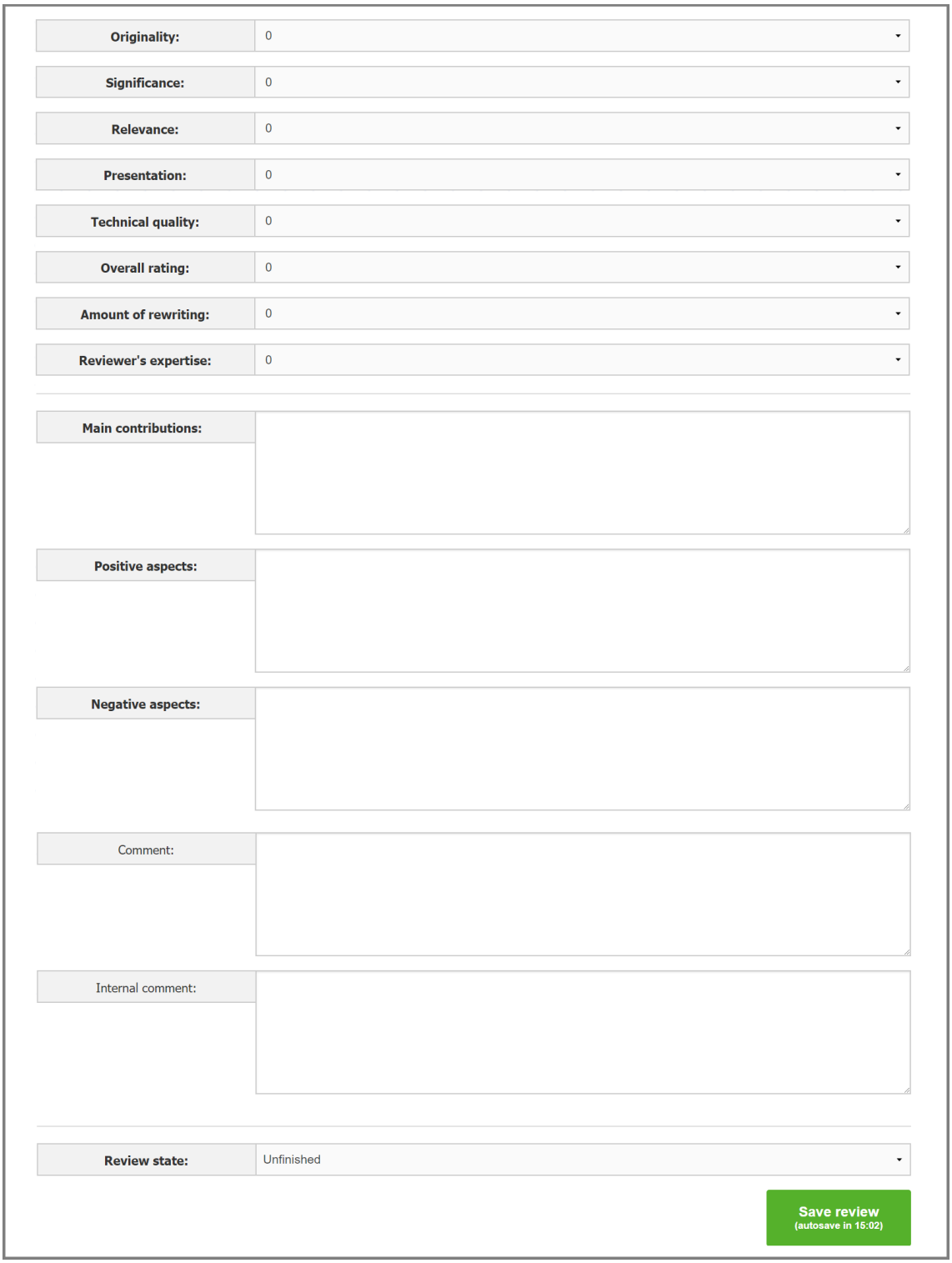
Ve světě se často stává, že se neoprávněně kopírují již hotová díla, která nejsou stále zaregistrovaná a mohou být případným zlodějem ukradnuta a vydána pod zlodějovo jménem. Proto je vhodné do celého dokumentu vložit vodoznak, který bude jasně říkat, že se jedná pouze o hodnotící soubor, nikoliv o plnohodnotné dílo.

4.1.6 Fonty

V PDF a PostScript prostředí se lze setkat s pojmem „14 standardních/-základních fontů“. Tento pojem byl odvozen ze standardních 13 PostScript fontů a vyjadřuje základní fonty používané při vytváření veškerých PDF souborů. Všechny základní fonty lze nalézt v tabulce 4.1.

Rodina fontů	**Fonty**
Times	Times-Roman Times-Italic Times-Bold Times-BoldItalic
Helvetica	Helvetica Helvetica-Oblique Helvetica-Bold Helvetica-BoldOblique
Courier	Courier Courier-Oblique Courier-Bold Courier-BoldOblique
Symbol	Symbol

Tabulka 4.1: Tabulka základních fontů pro PDF soubory



Obrázek 4.1: Webový formulář používaný pro hodnocení vědeckých příspěvků na portálu konference TSD

4.2 Hlavní funkce modulu

Vyvíjený modul musí být napsán stejným stylem jako je celý webový portál konferenčního systému TSD. Jelikož se na tomto portálu vyskytuje modul, který má stejnou funkcionalitu jako modul vyvíjený autorem bakalářské práce, tak je návrh funkcí jednodušší. Modul vyskytující se na portálu implementuje 3 důležité funkce, které zajišťují veškerou funkcionalitu i přes to, že pro zpracovávání PDF souborů je použit externí program *PDFtk*. Po konzultaci s vedoucím práce bylo rozhodnuto, že se původní názvy funkcí zachovají a budou pouze změněny jejich parametry. Autorův modul bude tedy ve výsledku obsahovat, stejně jako starý modul, 3 důležité funkce doplněny o pomocné funkce a konstanty. Všechny hlavní funkce jsou popsány níže.

4.2.1 Funkce pro generování

Pro generování PDF souboru byla navržena 1 funkce. Tato funkce má za úkol nejdříve nastavit veškeré fonty a styly pro vzhled dokumentu, následně využít vhodný generátor PDF souborů, který vytvoří všechny části dokumentu (vypsané v kapitole 4.1) a nabídne uživateli možnost stáhnout si výsledný hodnotící PDF soubor. Návrh hlavičky funkce viz 4.1.

```
function generate_offline_review_form($rid, $reviewer_name, $sid, $submission_name,
    $submission_filename)
```

Listing 4.1: Návrh hlavičky funkce pro generování PDF souboru

Popis vstupních parametrů funkce:

- **$rid** – ID hodnotícího příspěvku
- **$reviewer_name** – Celé jméno posuzovatele
- **$sid** – ID vědeckého příspěvku
- **$submission_name** – Celý název vědeckého příspěvku
- **$submission_filename** – Celý název PDF souboru vědeckého příspěvku

4.2.2 Funkce pro zpracování

Pro zpracování PDF souboru byly navrženy 2 funkce, kdy první z nich má za úkol nahrát celý soubor do konferenčního souboru. Po nahrání souboru

začne extrakce dat pomocí vhodných parserů a jejich zpracování. Následně se vše uloží do vhodných struktur a všechny extrahované hodnotící parametry se předají do následující funkce. Návrh hlavičky funkce viz 4.2.

```
function process_offline_review_form($rid, $sid, $revform_filename)
```

Listing 4.2: Návrh hlavičky funkce pro extrakci dat

Popis vstupních parametrů funkce:

- **$rid** – ID hodnotícího příspěvku
- **$sid** – ID vědeckého příspěvku
- **$revform_filename** – Celý název PDF souboru hodnotícího příspěvku

Druhá funkce pro zpracování PDF souboru má za úkol uložit již extrahované hodnotící parametry do databáze konferenčního systému. Návrh hlavičky funkce viz 4.3.

```
function upload_to_DB_offline_review_form($rid, $values)
```

Listing 4.3: Návrh hlavičky funkce pro uložení dat do databáze

Popis vstupních parametrů funkce:

- **$rid** – ID hodnotícího příspěvku
- **$values** – Seznam všech hodnotících parametrů

5 Implementace modulu

Při implementaci modulu bylo potřeba vybrat nejvhodnější knihovnu pro generování PDF souboru, kterou následně zaintegrovat i s předem vybraným parserem. Pro snadnější zaintegrování byly vytvořeny nové třídy.

5.1 Adresářová struktura modulu

Adresářová struktura modulu na serveru vypadá následovně:

- **config** – Adresář obsahující konfigurační soubor *configuration.xml*, ve kterém jsou uloženy často měněná data (rok konference, upozorňující info aj.).
- **css** – Adresář obsahující soubor *style.css* s kaskádovými styly pro konfiguraci vzhledu HTML kódu a výsledného dokumentu.
- **img** – Adresář obsahující obrázky použité v dokumentu.
- **lib** – Adresář obsahující zdrojové kódy knihoven třetích stran (generátor a parser).
- **src** – Adresář obsahující zdrojové kódy vytvořené autorem bakalářské práce.
- **how-to.txt** – Informační soubor, ve kterém se nachází implementační postupy (pro případné změny nebo rozšíření stávajícího modulu).
- **orlib.php** – Hlavní soubor modulu obsahující všechny 3 hlavní funkce modulu.

5.2 Implementované třídy

Pro modul bylo vytvořeno 7 tříd rozdělených do 4 souborů, které zajišťují veškerou pomocnou funkcionalitu (vytváření HTML kódu, konstanty aj.) při vytváření PDF souboru.

5.2.1 Výčtové typy

Výčtový typ (neboli **Enum**) je datový typ určený pro uložení konstant programu, kdy každé z těchto konstant je přiřazena jedna instance výčtu. Ve vytvářeném modulu byly použity 4 třídy jako výčtové typy.

Třída **Instruction** uchovává konstanty využité při generování titulku a informací ohledně vyplňování formuláře. Tyto konstanty reprezentují celkem 4 části dokumentu (Záhlaví, Titulek dokumentu, Jméno posuzovatele a Instrukční text pro vyplňování formuláře).

Třída **FormElements** slouží pouze pro rozlišení použitých objektů na základní prvky formuláře. Zde byly použity prvky *Radiobutton* a *Textové pole*.

Ve třídě **TextareaInfo** jsou uloženy veškeré konstanty pro hodnotící parametry, které jsou reprezentovány jako *Textové pole*, a pomocné funkce. Každý hodnotící parametr je zde reprezentován třemi konstantami (jednoznačný identifikátor, název a jeho popis). Dále se tu vyskytují 2 konstanty využité při vytváření formulářového prvku pomocí HTML kódu a následně i při jejich zpracovávání. Byla zde vytvořena i funkce *getNotNeededConstants()* pro získání nepovinných hodnotících parametrů.

Třída **RadiobuttonInfo** je téměř totožná s třídou **TextareaInfo** s tím rozdílem, že hodnotící parametry jsou reprezentovány jako *Radiobutton*.

5.2.2 HTMLElements

Hlavním důvodem vzniku této třídy byla snaha nevytvářet HTML kód přímo v hlavní funkci *generate_offline_review_form*, ale za použití nově vytvořených metod. Pomocí implementovaných metod lze vytvářet textová pole, radiobuttony a textové části dokumentu.

5.2.3 TextConversioner

V některých případech je název vědeckého příspěvku nebo jméno posuzovatele příliš dlouhé, a proto narušuje vzhled výsledného dokumentu. Narušení může nastat i při chybě programátora, pokud by byl instrukční text příliš rozsáhlý. Proto byla vytvořena třída **TextConversioner**, která má za úkol nejdříve zkontrolovat předaný text a porovnat ho se stanovenými konstantami určujícími maximální délku textu. Pokud je rozsah textu delší než stanovená délka, tak se následně vypočte potřebný font pro vykreslení celého textu pomocí vzorce (5.1), která se porovnává se stanovenými konstantami

určujícími minimální font.

$$newFontSize = \left\lfloor \frac{maxTextLength}{textLength} \cdot fontSize \right\rfloor \tag{5.1}$$

Za okolností, že vypočtený font je menší než předem stanovený minimální font, je text zkrácen na velikost vypočtenou pomocí vzorce (5.2) a doplněn třemi tečkami na jeho konci.

$$newLength = \left\lfloor \frac{oldFontSize}{minFontSize} \cdot textLength \right\rfloor \tag{5.2}$$

5.2.4 ConfigurationData

Třída načítá veškerý obsah konfiguračního souboru, který následně ukládá do svých proměnných. Data uložená v konfiguračním souboru slouží pro nastavení textu vodoznaku a jako informační text pro uživatele.

5.3 Generátor

Generátor by měl být při vytváření PDF dokumentu rychlý, vykreslit co nejpřesněji prvky webového formuláře do vygenerovaného dokumentu a nebýt implementačně náročný.

5.3.1 TCPDF x mPDF

Při analyzování dostupných PHP knihoven pro generování PDF souborů byly zjištěny 2 vyhovující knihovny, které můžou potencionálně splňovat potřebnou funkcionalitu, bohužel pouze 1 může být použit do vyvíjeného modulu. Po vytvoření jednoduchého souboru obsahující základní formulářové prvky bylo rozhodnuto, že knihovna **mPDF** bude použita pro generování PDF souborů. Důvody této volby jsou popsány níže.

Jeden z důležitých faktorů lze označit skoro kompletní podporu *CSS3* (Cascading Style Sheets 3) u **mPDF**, díky čemuž lze dosáhnout perfektního nastavení stylů pro jednotlivé objekty v dokumentu, zatímco **TCPDF** nepodporuje značné množství CSS parametrů (například parametr určující šířku vnějšího okraje prvku) a pro dosažení obdobného výsledku je zapotřebí značné množství jiných parametrů definující styl prvku.

Důležitým faktorem při generování PDF je rychlost generování a paměťová náročnost. V tabulce 5.1 lze vidět porovnání knihoven pro 2 PDF soubory, kdy komplexní PDF obsahovalo hlavně CSS styly, zatímco v dlouhém PDF byla vytvořena tabulka s více jak tisíci záznamy.

Název	**Komplexní PDF**		**Dlouhé PDF**	
	Paměť [MB]	**Čas [ms]**	**Paměť [MB]**	**Čas [ms]**
TCPDF (v6.2.13)	74	35944	2,3	96350
mPDF (v7.1.6)	14	11316	22,5	4120

Tabulka 5.1: Tabulka časové náročnosti a využité paměti při generování

Posledním a zároveň rozhodujícím důležitým faktorem je psaní PHP kódu pro vykreslování obsahu, kdy při psaní kódu u **mPDF** se využívá minimum funkcí pro nastavení parametrů PDF souboru (jako jsou například metadata), zatímco veškeré zobrazené elementy a text jsou psány v HTML stylu, který je snadno manipulovatelný a lze měnit parametry jednotlivých elementů (hodnota této vlastnosti bude oceněna hlavně u parseru). U **TCPDF** se zobrazovaný obsah vkládá pomocí předem vytvořených funkcí, kdy v některých případech tyto funkce obsahují mnoho parametrů, které si uživatel jen tak nezapamatuje a vždy bude potřebovat patřičnou dokumentaci pro správné použití (to bude zabírat mnoho času při vyvíjení nových modulů).

Na závěr průzkumu lze říci, že ve většině případů je vhodné využít pro generování PDF souborů knihovnu **mPDF**. Pokud by bylo nutné vytvořit dokument například ve stylu knihy (nulové využití CSS stylů a potřeba kvalitního vysázení textu), pak je lepší využít knihovnu **TCPDF**.

5.3.2 Popis vytvoření dokumentu

Na samotném začátku generování jsou vytvořeny veškeré proměnné vytvořených tříd (u proměnné třídy **ConfigurationData** proběhne i načtení dat z konfiguračního xml souboru). Dále jsou vytvořeny proměnné reprezentující název vytvořeného dokumentu a info ohledně nahrání vyplněného dokumentu do webového portálu konference TSD. Před samotným začátkem generování je do modulu importováno CSS nastavení pro vzhled celého dokumentu.

V první části generování probíhá přiřazení CSS stylů pro jednotlivé HTML položky (použité fonty, velikosti fontů aj.), které je doprovázeno vytvořením záhlaví pro celý dokument. Pro celý dokument byl použit font *Helvetica*, pouze u pár výjimek je použit font *Times New Roman* (pro titulek dokumentu a veškerý text se stylem *Bold*). Do záhlaví byl vložen identifikátor hodnotícího příspěvku doplněn o název hodnoceného vědeckého příspěvku (ten je případně zkrácen na určitou délku pokud nesplňuje limity nastavené ve třídě **TextConversioner**) a logem konference TSD.

V druhé části generování proběhlo vložení vodoznaku do celého doku-

mentu, uložení jednoznačného identifikátoru jak hodnoceného vědeckého příspěvku, tak i hodnotícího příspěvku do metadat dokumentu. Na první stránce dokumentu je vykreslen titulek s identifikátorem hodnoceného vědeckého příspěvku, název hodnoceného vědeckého příspěvku (případně zkrácen stejně jako u záhlaví) a jméno posuzovatele, doplněno o doprovodný text při vyplňování hodnotícího formuláře. Pod tímto textem je vykreslena první část hodnotícího formuláře, který obsahuje 8 skupin radio buttonů a 1 textové pole.

Ve třetí a poslední části probíhá vykreslování druhé části hodnotícího formuláře obsahující 4 textová pole, kde 2 poslední jsou nepovinná. Za hodnotícím formulářem je vložen kompletně celý hodnocený vědecký příspěvek, kdy jeho obsah se uloží do modulu a následně stránku po stránce je přidáván do generovaného PDF dokumentu.

5.3.3 Chyby v mPDF

Při vytváření dokumentu byly nalezeny 2 chyby znemožňující úplné vykreslení celého dokumentu. Níže jsou tyto chyby popsány i s jejich řešením.

První chyba byla zjištěna na úplném začátku implementace generátoru, kdy při vkládání textových polí do formuláře se po přeložení kódu nevytvořil žádný dokument. Při zkoumání zdrojového kódu knihovny a vytvoření pár testovacích dokumentů bylo žjištěno, že knihovna neumožňuje vytvořit textové pole pokud se při jeho vytvoření nezadá vkládaný text. Proto bylo nutné, aby se poupravil kód knihovny, konkrétněji ve vykreslování textového pole. Aby bylo možno takto upravovat zdrojový kód knihovny, tak nesmí být knihovna pod licencí, nebo alespoň poď licencí dovolující úpravy (například *GNU General Public License* verze 2, pod kterou je licencována i mPDF). Pro vyřešení tohoto problému byl přidán mechanismus, který při vytváření prázdného textového pole přidá znak „**a**“ (viz 5.1) a posléze je v knihovně při vykreslování textového pole tento znak odstraněn (nemá vliv na jakýkoliv jiný znak či slova, pouze na znak „**a**“, viz 5.2).

```
if($textarea_text == '') $textarea_text = 'a';
```

Listing 5.1: Přiřazení znaku „**a**“ jako text textového pole (HTMLElements.php)

```
if (isset($objattr['text']) && $objattr['text'] != 'a') {
        $texto = $objattr['text'];
}
else $texto = '';
```

Listing 5.2: Odstranění znaku „**a**“ z textového pole (Mpdf.php)

Druhá chyba byla nalezena při testování zkracování délky textu titulku, pokud překročí nastavenou mez. Protože v aktuální verzi PHP je z neznámých příčin chyba, která zapříčiňuje špatné ukládání některých znaků v kódování UTF-8, které by se měli přenášet i do vygenerovaného dokumentu. Bohužel generování dokumentu neprobíhalo správně, protože knihovna mPDF tyto znaky nerozpoznala a proto výsledek generování vždy skončil chybou. Ze všech vyzkoušených možností (změnit kódování textu titulku, nahrazení neplatných znaků prázdnými aj.) fungovala pouze jedna, a to nastavení atributu **ignore_invalid_utf8** na *true* u proměnné třídy *Mpdf* (viz 5.3).

```
$mpdf->ignore_invalid_utf8 = true;
```

Listing 5.3: Nastavení atributu **ignore_invalid_utf8** (orlib.php)

5.4 Parser

Z analýzy knihoven pro zpracování PDF dokumentů splnil nutné požadavky pouze **PDF Parser**.Při implementování parseru bylo zjištěno, že jednotlivé PDF prohlížeče při uložení PDF dokumentu využívají jiné komprimační metody, využívají novější verze PDF pro nové funkce a některé prohlížeče ukládají objekty v dokumentu na více místech (duplikace, jednou komprimovaně, jednou nekomprimovaně).

5.4.1 Popis zpracování dokumentu

Zpracování dokumentu začíná ihned po jeho nahrání do webového portálu konferenčního systému, kdy se veškerá raw data předají do PDF Parseru. Před samotnou extrakcí dat jsou pomocí TCPDF parseru (který je součástí PDF Parseru) raw data rozdělena na objekty pomocí *traileru* a *xref tabulky*. Následně jsou objekty dekódovány a předány PDF Parseru, který s nimi dále pracuje. Ihned po předání jsou tyto objekty dále zpracovávány podle specifických znaků, které se vyskytují v datech (například v objektu *Slovník* se na samém začátku vyskytují znaky „«“). Zároveň se kontroluje název objektu, podle kterého lze zjistit o jaký typ HTML prvku se jedná. Každý prvek formuláře má přiřazen jednoznačný název. Jako příklad lze uvést textové pole, které má při generování přiřazen název „textareaID“, kde ID je jednoznačný identifikátor textového pole, který je deklarován ve výčtovém typu **TextareaInfo**. Pokud je název totožný se specifickým názvem jakéhokoliv HTML prvku, které jsou v modulu naimplementovány, tak jsou potom ihned uloženy do struktury, která je po dokončení parsování poslána do modulu.

Po zpracování celého dokumentu jsou požadované objekty roztříděny na základě jejich jednoznačných identifikátorů (čísla na konci slovního identifikátoru, například „*textarea**0***“, kde **0** popisuje v modulu hodnotící parametr *Originality*). Před uložením hodnot jsou tyto hodnotící parametry testovány, jestli jsou náležitě vyplněny (platí pouze pro povinné položky). Zpracování probíhá pro každý základní prvek formuláře samostatně (ideálně pomocí cyklu a switche). V případě že všechna povinná pole jsou vyplněna a neproběhla žádná chyba ve zpracování, jsou všechny hodnotící parametry uloženy do databáze webového portálu konference TSD.

Při nahrávání dokumentu může dojít k několika chybám, kterých se posuzovatel může dopustit, a proto jsou náležitě ošetřeny.

- **Neplatné PDF** – Posuzovatel při nahrávání dokumentu zvolí nevalidní PDF dokument (nevygenerovaný webovým portálem).
- **Neplatný identifikátor hodnotícího příspěvku** – Posuzovatel může při nahrávání zvolit hodnotící PDF dokument patřící k jinému hodnocení vědeckého příspěvku (posuzovaní identifikátoru hodnotícího příspěvku a vědeckého příspěvku).
- **Nevyplněné požadované parametry** – Nahrávaný PDF dokument obsahuje nevyplněné požadované hodnotící parametry. Tyto parametry jsou vypsány v chybové hlášce zobrazené po pokusu nahrát PDF dokument do webového portálu.
- **Databázové chyby** – Při ukládání dat do databáze webového portálu může dojít k neočekávané chybě, která zapříčiní nesprávné uložení dat.
- **Uzavřené hodnocení příspěvku** – Když posuzovatel nahraje hodnotící PDF dokument do webového portálu, zatímco hodnocení vědeckého příspěvku je uzavřeno administrátorem webového portálu.

5.4.2 Extrakce formulářových prvků z předpřipravených dat

TCPDF parser částečně extrahuje veškeré PDF objekty do svých struktur, které jsou následně poslány do PDF Parseru, který je pomocí svých funkcí a tříd roztřídí na základě datového typu vyplněného obsahu, jako je například prostý text, datum nebo číselná hodnota. Příklad struktury jednotlivých objektů lze vidět na obrázku 5.1. Tento obrázek ukazuje pouze část struktury, která je mnohem rozsáhlejší a každým krokem parseru se rozkládá na menší kousky.

```
Smalot\PdfParser\Header Object ( ...
   [T] => Smalot\PdfParser\Element\ElementString Object (
      [document:protected] => [value:protected] => group0 )
   [V] => Smalot\PdfParser\Element\ElementName Object (
      [document:protected] => [value:protected] => 2
... )
```

Obrázek 5.1: Část předpřipraveného PDF objektu

Pro extrahování dat a zjištění typu formulářového prvku byla vyvinuta metoda *extractElement* viz 5.4.

```
protected function extractElement($header) {
   $elementKey = $header->getElements()['T'];
   if ($elementKey != null) {
      if (strpos($elementKey->getContent(), 'group') !== false) $type = 'groups';
      else if (strpos($elementKey->getContent(), 'textarea') !== false) $type = 'textareas';

      if ($type != null) {
         $key = $elementKey->getContent();
         $elementValue = $header->getElements()['V'];
         if ($elementValue != null) {
            $value = $elementValue->getContent();
         }
      }
   }
}
```

Listing 5.4: Funkční kód pro uložení formulářových prvků z PDF objektů

Na samém začátku kód kontroluje, zda je aktuálně zpracovávaný objekt pojmenován. To je zjištěno na základě indexu „**T**“ (T - Type) v poli elementů. Pokud název existuje, zjišťuje se zda se jedná o textové pole nebo skupinu radio buttonů. Za předpokladu, že typ objektu je validní, je kontrolována hodnota na základě indexu „**V**“ (V - Value) v poli elementů. Jakmile objekt obsahuje hodnotu, jsou do parseru vráceny všechny tři hodnoty (název formulářového prvku, typ formulářového prvku a jeho hodnota), které parser uloží do pole všech extrahovaných prvků.

5.5 Výsledný vzhled PDF formuláře

5.6 Technické požadavky

6 Rozšiřitelnost modulu

7 Ověření kvality software

8 Závěr

Literatura

[1] *PDF formuláře: obecný úvod* [online]. 2002. [cit. 2002/04/25]. Dostupné z: `http://www.grafika.cz/rubriky/pdf---adobe-acrobat/pdf-formulare-obecny-uvod-130460cz`.

[2] *PDF formuláře: Popis formulářových prvků* [online]. 2002. [cit. 2002/05/15]. Dostupné z: `http://www.grafika.cz/rubriky/pdf---adobe-acrobat/pdf-formulare-popis-formularovych-prvku-130502cz`.

[3] *Compression in PDF files* [online]. Prepressure, 2017. [cit. 2017/01/05]. Dostupné z: `https://www.prepressure.com/pdf/basics/compression`.

[4] *PHP PDFTK* [online]. 2017. [cit. 2017/12/17]. Dostupné z: `https://www.drupal.org/project/phppdftk`.

[5] Christenssion, P. *PDF Definition* [online]. TechTerms. Sharpened Productions, 2018. [cit. 2018/04/05]. The Tech Terms Dictionary. Dostupné z: `https://techterms.com/definition/pdf`.

[6] King, J. *Introduction to the Insides of PDF* [online]. Adobe, 2005. [cit. 2005/04/29]. Dostupné z: `https://www.adobe.com/technology/pdfs/presentations/KingPDFTutorial.pdf`.

[7] Lukan, D. *PDF File Format: Basic Structure* [online]. InfoSec, 2018. Dostupné z: `https://resources.infosecinstitute.com/pdf-file-format-basic-structure`.

[8] Nikolaev, A. *PHP PDF to HTML: Convert PDF to HTML using Poppler* [online]. 2018. [cit. 2018/06/29]. Dostupné z: `https://www.phpclasses.org/package/9423-PHP-Convert-PDF-to-HTML-using-Poppler.html`.

[9] Rouse, M. *Portable Document Format (PDF)* [online]. TechTarget, 2010. [cit. 2010/05/20]. Dostupné z: `https://whatis.techtarget.com/definition/Portable-Document-Format-PDF`.

[10] Whitington, J. *PDF Explained, Chapter 3. File Structure.* O'Reilly Media, Inc., 2011. ISBN 9781449310028.