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

Bakalářská práce

Generátor a parser formulářů recenzí příspěvků na konferenci TSD

Plzeň 2017 Jakub Váverka

Místo této strany bude zadání práce.

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 23. dubna 2017

Jakub Váverka

Abstract

Generator and Parser of Submission Review Forms for the TSD Conference

The text of the abstract (in English). It contains the English translation of the thesis title and a short description of the thesis.

Abstrakt

Text abstraktu (česky). Obsahuje krátkou anotaci (cca 10 řádek) v češtině. Budete ji potřebovat i při vyplňování údajů o bakalářské práci ve STAGu. Český i anglický abstrakt by měly být na stejné stránce a měly by si obsahem co možná nejvíce odpovídat (samozřejmě není možný doslovný překlad!).

Obsah

1 Úvod

Cílem této práce je navrhnout a implementovat rozšíření již stávající webové stránky TSD. Text, speech and Dialogue, dále jen TSD , je konference, která probíhá každoročně. Tato konference se zabývá například rozpoznáváním, pochopením, generováním a obecně interakcí lidského jazyka s počítačem. Uživatelé mohou každoročně přihlásit své články do systému, tyto články jsou později ohodnoceny vybranými uživateli a buď schváleny nebo odmítnuty. Vyplnění tohoto hodnocení probíhalo doposud jen skrze online formulář. Mým úkolem je navrhnout způsob jak tento úkon uživatelům co nejvíce zpříjemnit a umožnit jim hodnocení těchto článků i bez přístupu k internetu. Zároveň musí být řešení dostatečně jednoduché, jelikož se nedá počítat s vysokou počítačovou zdatností daných uživatelů. Pro uživatele by bylo ideální pokud by mohl jednoduše stáhnout jediný soubor, který by obsahoval jak článek který mají hodnotit, tak i funkcionalitu na jeho ohodnocení.

Pokud bych zvolil formát, který je pro takové věci určený, bude pravděpodobně neznámý a musel bych po uživatelích požadovat instalaci dodatečných programů. Pokud bych naopak zvolil formát otevíratelný v textovém editoru, zmátl bych nezkušené uživatele a umožnil bych jim zničit strukturu potřebnou k zpětnému zjištění dat. Zlatou střední cestou se zdá být využití formátu pdf. Tento formát je sice velmi rozšířený ale poměrně uzavřený. Navíc se tento formát nepoužívá jen na textové soubory, a tak není snadno editovatelný. To bude představovat výzvu pro implementaci.

1.1 Textové formáty

TODO

1.2 Adobe

Adobe Systems je společnost, která byla založena Johnem Warnockem a Charlesem Geschkem v roce 1980. Tito zakladatelé byli do té doby částí výzkumného týmu ve společnosti Xerox. Jejich prvním produktem byl software nazývaný PostScript. PostScript je programovací jazyk, který dokáže tiskárně popsat rozvržení elektronické stránky. Toto byl v té době velký pokrok, který jim vydělal velké množství peněz. Firma Adobe na tento úspěch navázala mnoha dalšími produkty, mezi které patří například první volně

škálovatelný font, program na tvorbu maleb Adobe Illustrator a mimo jiné i Portable Document Format neboli pdf. [8] Pdf jako formát pro sdílení a tisknutí elektronických souborů byl zveřejněn v roce 1993. Velmi rychle rozdrtil konkurenci a stal se standardem, který každý zná a běžně využívá každý den. Pdf se postupně vyvíjelo, z první verze 1.0 je již momentálně verze 1.7 a touto nejnovější verzí se stalo pdf standardem ISO. Adobe vydalo program pojmenovaný Acrobat Reader, který slouží k čtení tohoto formátu. Tento program byl nejdříve placený, ale současně je zdarma a tvoří výbavu skoro každého počítače. To je jeden z důvodů proč je právě tento formát ideální pro mou práci. [4] Přestože Adobe umožňuje uživatelům zdarma používat software na čtení tohoto formátu, již neumožňuje tento formát zdarma editovat či jinak měnit. K takovým operacím slouží program s názvem Acrobat Acrobat. Licence na tento program musí být placena měsíčně. Navíc je Adobe Acrobat určen především pro klasické uživatele, kteří budou pracovat skrze grafické rozhraní. Tímto programem není možné vytvořit dokument skrze příkazovou řádku nebo jiným automatizovaným způsobem.

1.2.1 Formuláře

Formát pdf dokáže obsahovat i dynamický obsah v podobě vyplnitelných formulářů. Bezplatná verze programu Acrobat Reader však neumožňuje tvorbu takových souborů. K tvorbě formulářů uživatel potřebuje plnou placenou verzi Adobe Acrobat. Po vytvoření takového souboru jej může samozřejmě sdílet s jinými uživateli, kteří ji mohou vyplnit i bezplatnou verzí Acrobat Reader. [3] Formuláře firmy Adobe jsou velmi uzavřená záležitost. Samotná realizace formuláře lze provést několika způsoby. Během vývoje pdf formulářů, firma Adobe používala více technologií. Většina těchto technologií využívala servery k získání odpovědí od uživatelů. Tyto servery již byly odstaveny, takže se dané technologie dají využít jen ve velmi omezené míře. Navíc nejsou tyto systémy nastaveny na automatizaci a Adobe předpokládá, že bude autor formulářů odpovědi od uživatelů sbírat pomocí grafického programu Adobe Acrobat a dále je konvertovat třeba do tabulek Excel. Adobe se snaží tyto technologie navrhovat pro podnikatele a obchodníky. K jejich využití tedy uživatel nepotřebuje skoro žádné znalosti a většina funkcionality je před ním skrytá. Prvním programem na tvorbu formulářů od Adobe byl Acrobat Forms Data Format Toolkit, který má bezplatnou licenci.

1.2.2 Acrobat FDF Toolkit

FDF neboli Forms Data Format je formát navrhnutý firmou Adobe. Slouží k uložení dat vyplněných do formuláře v souboru pdf. Uživatel vyplní formulář a pak místo uložení celého souboru uloží odděleně soubor s koncovkou fdf, který obsahuje pouze identifikátory polí a uživatelův vstup. Tento malý soubor může uživatel odeslat na server, kde je díky znalosti předešlého souboru pdf možné zjistit co uživatel vyplnil. K největší úspoře dochází ve chvíli, kdy je soubor pdf velký, protože nemusíme zbytečně odesílat celý jeho obsah na server, ale stačí jen malý soubor s odpověďmi. Adobe se navíc pokusilo tímto formátem do jisté míry nasimulovat webové formuláře. Firma Adobe umožňuje zakoupení a instalaci serverového softwaru, který přijímá přes internet fdf soubory od uživatelů, kteří formulář vyplnili. Uživateli se v Adobe Readeru objeví možnost odeslání formuláře. Tento způsob se zdá velmi chytrý, Adobe od něj však upustilo a je udržován spíše kvůli zpětné kompatibilitě. Jsem přesvědčen o tom, že většina uživatelů o této vlastnosti pdf formulářů neví a jednou z důležitých vlastností, které by mělo mé řešení splňovat je intuitivnost, proto jsem od tohoto řešení upustil.

1.2.3 Acrobat LiveCycle Designer

Dalším vývojovým stádiem Adobe formulářů byl LiveCycle Designer. Cílem tohoto programu bylo vytvořit xml soubory, které jsou obaleny souborem pdf. Tyto formuláře se nazývají XFA forms. Designer umožňuje do takových souborů doprogramovat další funkcionalitu pomocí jazyka JavaScript, nebo skriptovacího jazyka FormCalc, který navrhlo samotné Adobe. Formuláře vytvořeny pomocí LiveCycle Designer však nejsou kompatibilní s jinými programy od firmy Adobe. Mohou být samozřejmě vyplněny, ale ne již změněny. LiveCycle Designer byl součástí programu Adobe Acrobat až do verze X. Po této verzi se Adobe rozhodlo prodávat LiveCycle Designer jako samostatný produkt. To způsobilo mnoho nepříjemností uživatelům, kteří o této změně nevěděli a aktualizovali svůj Adobe Acrobat z verze X na verzi XI, protože jim byl během instalace LiveCycle Designer odinstalován. Adobe neumožňuje verzi programu downgradovat a to znamenalo, že tito uživatelé již nemohli své formuláře nijak upravovat dokud si od firmy Adobe nezakoupili i samostatnou licenci na LiveCycle Designer. To samozřejmě vzbudilo vlnu odporu, a tak Adobe začalo umožňovat jednorázové získání licence LiveCycle Designer pro uživatele, kteří zakoupili produkt Adobe Acrobat před datem 15. října 2012. [2] [5]

1.2.4 FormsCentral

FormsCentral byl poslední pokus Adobe k ovládnutí trhu s online formuláři. Tento program se, i jako oba předchozí, soustředil na možnost vyplnění pdf formuláře přímo na webu a použití externích serverů na ukládání odpovědí. Při vytvoření firma Adobe očekávala, že bude o takovou službu mnohem větší zájem, než který ve skutečnosti vznikl. Adobe však pomocí FormsCentral nenabídlo uživatelům takové změny, za které by stálo na novou platformu přejít. Nezájem způsobil to, že byla i tato služba ukončena 28. července 2015. Ještě před ukončením se na trhu objevilo více dalších služeb, které nabízí naprosto stejné vlastnosti a dokonce umožňují i naimportovat již stávající tabulky z FormsCentral. [1]

1.2.5 FormsCentral alternativy třetích stran

Firma Adobe samotná na svých stránkách, které se zabývají přerušením služby FormsCentral doporučuje, aby uživatele přešli na jednu z alternativ. Nejpopulárnější podle nich jsou Formstack, JotForm, Survey Monkey a WuFoo. Některé z těchto služeb umožňují hostování serveru na ukládání odpovědí zdarma, do určitého limitu. [1] Pro tuto práci by ale nebylo ideální, kdyby byla závislá na službách ze třetí strany, které by mohly postupem času skončit. Navíc se tyto služby soustředí na vyplňování formulářů online a to je již realizováno na webových stránkách TSD pomocí webového formuláře. Hlavním cílem této práce je navrhnout formuláře, které jdou vyplnit i bez připojení k internetu a následně je možné je přes webové stránky pouze nahrát. Proto se nebudu ve své práci těmito webovými servery nadále zabývat.

1.3 PDFLib

PDFlib je jeden z největších a nejúspěšnějších nástrojů pro vytváření a úpravu souborů ve formátu pdf. Jeho hlavním cílem je tvorba dynamických souborů, které obsahují data z databáze, pomocí webového serveru nebo libovolných serverových systémů. Je realizován jako knihovna pro PHP. Povedlo se jí zjednodušit práci se soubory pdf, a tak umožňuje aby se programátor soustředil na skládání komponent na stránku místo procházení komplikovaného formátu pdf. Knihovna PDFlib je na trhu již od roku 1997 a nabízí velký sortiment různých balíčků. Tato knihovna je placená. Nejlevnější je základní verze pro osobní počítač, která v současné době stojí €375. Pokud chce však uživatel využít tuto knihovnu na strojích IBM nebo Oracle zaplatí

za licenci až €3990 a to jen za základní verzi, která se soustředí především na vytváření souborů. Firma PDFlib GmbH která tuto knihovnu poskytuje nabízí i další pěkné služby. Patří mezi ně například PDI, je to knihovna která se soustředí na importování. Umožňuje tedy vzít již existující pdf dokument a přidat do něj další stránky, také umožňuje přidat záhlaví a zápatí nebo vložit vodoznak a jiné grafiky. Tato knihovna samozřejmě již nedokáže měnit stávající obsah nebo interagovat s dynamických obsahem. A to jsou právě vlastnosti, které jsou pro nás důležité. Další knihovna od této firmy se nazývá TET, což je zkratka pro Text and Image Extraction Toolkit. Jak již název naznačuje, tato knihovna se zabývá extrakcí textu, obrázků a metadat ze souborů formátu pdf. Obrázky jsou uloženy v běžných grafických formátech, zatím co texty tato knihovna ukládá jako speciální xml soubory, kterým se říká TETML. TET toho docílí za pomoci analytických algoritmů, které zjišťují umístění a formátování textu. Poslední knihovna od firmy PDFlib GmbH, která stojí za zmínku se jmenuje PLOP a zabývá se linearizací, optimalizací a ochranou pdf souborů. Linearizace vizuálně urychluje stažení pdf souboru, klientovi se nejdříve odešlou nejdůležitější informace a pdf se tedy zobrazí v kratším čase než kdyby bylo odesláno klasickým způsobem. Uživatel tedy může prohlížet dokument dříve, než má stažená všechna data. Zbytek dat se stáhne na pozadí. Optimalizace je u pdf velmi užitečná, protože dva stejné pdf soubory mohou mít naprosto rozdílné velikosti. PLOP dokáže takové soubory projít a smazat veškeré redundantní informace. Tím docílí menší velikosti souborů, aniž by snížil kvalitu. Co se zabezpečení týče mohou pdf soubory mít nastaveny zámky například na tisk nebo extrakci dat. Tento nástroj dokáže zámky odemknout a daný obsah zpřístupnit. Uživatel musí samozřejmě znát heslo nebo vlastnit certifikát jímž byl dokument uzavřen. PLOP dokáže navíc i projít poškozenou pdf strukturu a pokusit se jí opravit. S těmito dalšími knihovnami samozřejmě stoupá i cena licence. Firma PDFlib GmbH sice umožňuje stažení omezené verze PDFlib Lite 7, ale tato verze je spíše demonstrace toho, co dokáže plná verze. Navíc firma ukončila vývoj a PDFlib Lite 7 není od roku 2010 udržován.[10]

Knihovna je psaná v jazyce C, lze ji však použít v mnoha jazycích. Mezi ty nejznámější patří ANSI C, ANSI C++, Java, .NET, Perl, Python a PHP. Samozřejmě je potřeba před použitím knihovnu různě upravit nebo obalit, to je však již přehledně popsáno v dokumentaci. [11]

1.4 FPDF

FPDF je třída určena pro programovací jazyk PHP. Umožňuje generování pdf souborů pomocí čistého PHP, to znamená bez použití knihovny PDFlib, která je psaná v jazyce C. F v názvu znamená free. Tato třída je dostupná naprosto zdarma a může být využívána a upravována podle jakýkoliv potřeb. Knihovna FPDF je lehčí verzí knihovny TCPDF, obě dokážou podobné věci, ale TCPDF je mnohem robustnější. Protože je knihovna TCPDF postavena na knihovně FPDF, syntaxe kódu pro generování stránek se moc neliší. [6][12]

1.4.1 FPDI

FPDI je kolekce PHP tříd, které umožňují vývojářům číst stránky z již existujícího pdf dokumentu. Její využití je také zdarma. Ke svému běhu potřebuje knihovnu FPDF. [7]

1.5 mPDF

Knihovna mPDF je rozšíření předchozí knihovny FPDF. Především se soustředí na generování pdf souborů z html. Na rozdíl od svého předchůdce dokáže tato knihovna již použít kódování UTF-8. mPDF také dokáže aplikovat na dané html kaskádové styly, a tak se přiblížit vzhledu, který by html kód měl, pokud by byl zobrazen v internetovém prohlížeči. Tyto vlastnoti jsou vykoupeny větší velikostí souborů a delším časem, který je potřeba pro generování souborů. [9]

1.6 DOMPDF

DOMPDF se velmi podobná knihovně mPDF, ale na rozdíl od mPDF, která vychází z FPDF, tato knihovna potřebuje ke své funkci nainstalovaný nástroj PDFlib. Pokud uživatel tento nástroj nemá, může využít pro vytváření dokumentů přiloženou třídu R& OS CPDF. Generování dokumentů za pomoci této třídy je pomalejší a více paměťově náročné, ale je tím zaručeno že DOMPDF nebude závislá na žádné externí knihovně. Jelikož není v dokumentaci této knihovny žádná zmínka o kódování, předpokládám, že neumí kódovat UTF-8, proto považuji předchozí knihovnu za nadřazenější a k DOMPDF bych se uchýlil jen ve chvíli, kdy by mPDF nedokázalo vykreslit můj obsah podle představ. [**?**]

1.7 Snappy

Další způsob pro generování pdf souborů v programovacím jazyce PHP je obalová knihovna Snappy. Tato knihovna umožňuje snadné využití konzolové aplikace wkhtmltopdf. Nástroj wkhtmltopdf, jak již název napovídá, se využívá k převádění html na pdf. Knihovna Snappy jej tedy obaluje, aby bylo možno tento nástroj snadno použít i v jazyce PHP. Programu stačí předat url a výstupem je soubor, který vypadá identicky jako stránka, která se na daném url nachází.[13] [15]

1.8 TCPDF

TCPDF je PHP třída určena pro generování pdf dokumentů a dodnes je nejpoužívanější. Je to hlavně díky tomu, že je využívána ve mnoha populárních PHP aplikacích. TCPDF původně vznikla jako odvětví třídy FPDF, ale postupem času přibylo množství funkcí a dnes je skoro kompletně přepsána. Díky tomu, že je tak bohatě využívána, vzniklo na internetu velké množství návodů a příkladů, které budou při vývoji aplikací, které tuto třídu využívají, velmi užitečné. TCPDF pro generování pdf dokumentů nepotřebuje žádné externí knihovny. Automaticky vytváří dokumenty ve správných formátech a následně umožňuje si tyto formáty libovolně přizpůsobit a definovat vlastní okraje nebo například jednotky měření. Třída TCPDF zvládá generovat dokumenty v kódování UTF-8. Součástí této třídy jsou i nástroje pro zpětné importování existujících pdf dokumentů. Bohužel jsou stále ve vývoji a nemusíme jejich použitím dojít ke kýženým výsledkům. [14] [12]

1.9 PDF Parser

Pokud bychom chtěli importovat již existující pdf dokumenty, lepší cestou by byla PHP knihovna PDF Parser. Je to opět rozšíření, postavené na knihovně TCPDF, obohacené o další funkce. PDF Parser lze dokonce vyzkoušet online. Stačí na stránky PDF Parseru nahrát vaše pdf a okamžitě se vám v textovém souboru zobrazí co všechno byl schopný PDF Parser z vašeho dokumentu získat. [**?**]

1.10 PDFToolkit

PDFtk je jednoduchý grafický nástroj na rychlé spojování a rozdělování pdf dokumentů. Je naprosto zdarma. Obsahuje však i nástroj PDFtk Server,

který lze spustit z příkazové řádky. Mimo spojování a rozdělování má tento nástroj mnoho dalších užitečných funkcí, mezi které patří i extrakce dat. PDFtk dokáže vypsat pdf metadata, ale i obecně datová pole, která jsou v pdf uložena. [**?**]

1.11 Závěr průzkumu

Došel jsem k závěru, že vytváření dynamických pdf dokumentů není v dnešní době problém. Existuje obrovské množství knihoven, tříd a nástrojů, které takovou funkcionalitu poskytují. Mnohem větší problém bude vytváření dokumentů, které by mohl uživatel snadno doplňovat. Technologie, které umožňují uživatelům vyplňování formulářů, se během existence formátu pdf neustále měnily. A protože je tento formát mířený na úředníky a obchodníky, kteří neradi aktualizují své zavedené systémy, jsou tyto technologie zpětně kompatibilní. Může se tedy stát, že dva naprosto identické dokumenty s formulářem, mohou být interně realizovány naprosto rozdílným způsobem. Navíc nebylo nad takovými formuláři uvažováno jako nad něčím, co se bude po vyplnění následně nahrávat přes internet do databáze. Spíše šlo o ulehčení vyplnění před vytisknutím. Další možností bylo odesílání dat skrze Acrobat Reader za použití speciálních serverů na kolekci odpovědí. Kvůli tomu má program Acrobat Reader překvapivě rozsáhlé množství funkcí. Dokáže například provést javascirpt kód, který byl do pdf dokumentu dopsán. Díky tomu se dokáže připojit a upravovat obsah databáze. V tomto smyslu se Acrobat Reader přibližuje spíše internetovému prohlížeči, než jen hloupému průzkumníku pro pdf soubory. Moje první kroky budou směřovat k vytvoření vyplnitelných formulářů ve formátu pdf. Následně se pokusím z těchto souborů vyextrahovat obsah, který do něj uživatelé vyplnili.

2 Implementace

TODO

3 Dokumentace

Můj konečný modul je realizován jako malá knihovna v jazyce php. Veškerý kód je rozdělen do dvou funkcí, jedna se zabývá generováním pdf dokumentu a druhá jeho zpracováním. Také se v této knihovně nahoře nachází překrytá definice třídy z knihovny TCPDF.

3.1 Třída PdfWithHeader

Při vytváření pdf dokumentů pomocí knihovny TCPDF, se nejdříve volá konstruktor třídy TCPDF. Danému objektu se poté nastavují parametry a vlastnosti pomocí jeho funkcí. Pokud však chceme změnit záhlaví nebo zápatí, je zapotřebí překrýt funkce Header a Footer. V našem případě není zapotřebí zápatí, proto překrývám jen funkci Header. Informace v záhlaví jsou při každé generaci jiné. Toto dynamické záhlaví je ve tvaru ID:Name. Potřebujeme proto naší překryté třídě nějakým způsobem předat tyto dvě informace. Rozhodl jsem se k tomu použít konstruktor a tak jsem stávající konstruktor přetížil a přidal jsem 2 parametry - Name a ID. Tyto parametry se při volání konstruktoru uloží do vytvářeného objektu, podle návrhového vzoru přepravka a jsou poté zpřístupněny z funkce Header.

3.1.1 Funkce Header

```
public function Header() {
   $limit = 50;
   if(mb_strlen($this->name, 'UTF-8') > $limit){
      $this->name = mb_substr($this->name, 0, $limit,
         'UTF-8')."... ";
   }
   $this->SetFont('freesans', 'B', 20);
   $firstWordWidth = TCPDF::GetStringWidth($this->id ." : ",
      'freesans', 'B', 20, false);
   $this->Cell($firstWordWidth, 0, $this->id ." : ", 0, 0, 'L', 0,
      '', 0, false, 'L', 'C');

   $this->SetFont('freesans', 'B', 15);
   $this->Cell(0, 0, $this->name , 0, 0, 'L', 0, '', 0, false,
      'L', 'C');
```

```
    $this->Line(5, $this->y + 4, $this->w - 5, $this->y + 4);
}
```

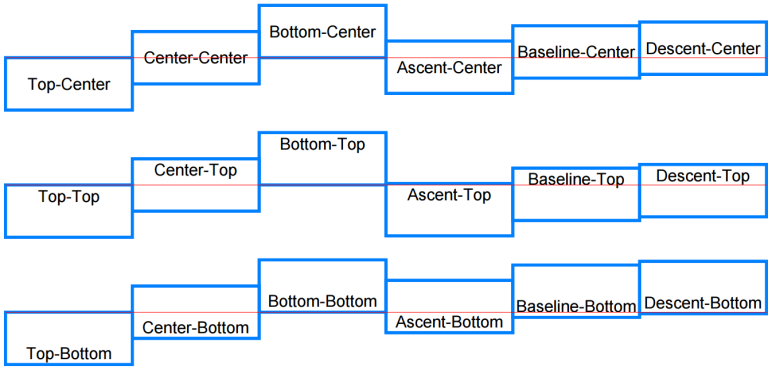
V této funkci nejdříve dojde k oříznutí textu, často se totiž stává, že je název článku velice dlouhý. Text v záhlaví je vykreslován pomocí funkce Cell, tato funkce text nezalamuje, to znamená, že pokud bychom nezkontrolovali jeho délku, došlo by po vykreslení k tomu, že by text pokračoval i mimo stránku. Veškeré objekty, které jsou vykreslovány pomocí knihovny TCPDF se vykreslují na současnou pozici ukazatele. Tento pomyslný ukazatel se poté postupně posouvá. Pokud chceme jeho pozici nastavit, slouží k tomu funkce setX a setY, pokud jsou tyto funkce použity se zápornými čísly, počítá se vzdálenost od pravé nebo spodní strany. Samotná funkce Cell má několik parametrů.

Funkce Cell a její parametry

```
$pdf->Cell(w, h, txt , border, ln, align, fill, link, stretch,
    ignore_min_height, calign, valign);
```

První dva parametry odpovídají šířce a výšce buňky, která bude vykreslena. Základní hodnoty těchto parametrů jsou 0. Buňka s takovými parametry bude zabírat minimální výšku, do které se její obsah vejde a maximální šířku. Dalším parametrem je samozřejmě text, který má buňka obsahovat. Poté následuje okraj. Tento parametr může nabývat číselných hodnot 0 a 1. Pokud je zvolena 1, je okraj buňky zvýrazněn černým okrajem. Tato funkce je zvlášť užitečná při tvoření nových dokumentů, jelikož umožňuje zobrazení skutečného rozsahu dané buňky. Také je možné použít písmenné hodnoty L,T,R,B, které zobrazí okraj jen na jedné straně buňky. Písmena odpovídají anglickým názvům stran - left, top, right a bottom. Parametr ln může nabývat hodnot 0 a 1. Tato možnost určuje, zda bude po vytisknutí buňky odřádkováno. Zvolení 1 je stejné jako zvolení 0 a okamžité použití funkce Ln. Další parametr align slouží k zarovnání textu. Tento parametr může nabývat hodnot L,C,R,J. Text bude tedy zarovnán vlevo, vycentrován a nebo zarovnán vpravo. Písmeno J znamená justify a skrývá se pod ním takzvané zarovnání do bloku. Parametr fill určuje zda má být pozadí buňky průhledné a nebo vyplněné. My budeme ve všech případech potřebovat, aby bylo průhledné a tak bude nastaveno na false. Pokud bychom potřebovali použít buňku jako hypertextový odkaz, použijeme k tomu parametr link. Lze do něj dosadit URL a nebo identifikátor objektu vytvořeného funkcí AddLink. Parametr stretch určuje chování textu v buňce a může nabývat pěti hod-

not. Pro hodnotu 0 je text zobrazen ve své velikosti a nedochází k žádnému zkreslení. Pokud je zadána hodnota 1, tak se text horizontálně zkrátí pokud by se do dané buňky nevešel. Při hodnotě 2, se text přizpůsobí horizontálně velikosti buňky i pokud by byl kratší. Pokud je zvolena hodnota 3 je umožněno knihovně TCPDF přizpůsobit i mezery mezi písmeny, tak aby se text do buňky vešel a při hodnotě 4 jsou tyto mezery zvětšeny tak aby text zabíral celý prostor buňky. Parametr ignore_min_hight lze použít, pokud chceme kreslit buňky menší než je jejich minimální výška, která je definována v knihovně TCPDF. Tuto možnost také nebudeme potřebovat, takže ji necháme nastavenu na hodnotu false. Parametry calign a valign oba určují vertikální zarovnání. První určuje vertikální zarovnání buňky a druhý textu, který buňka obsahuje. Parametr calign může nabývat hodnot T,C,B,A,L,D a parametr valign jen hodnot T,C,B. Všechny možné kombinace, kterých se dá docílit jsou přehledně zobrazeny v obrázku 3.1.



Obrázek 3.1: Příklad vertikálního zarovnání buňky

3.2 Generování dokumentu

Generování dokumentu probíhá ve funkci pojmenované generate_offline_review_form. Tato funkce má 6 parametrů.

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

3.2.1 Parametry

Parametr rid je identifikační číslo recenze. Toto číslo je použito k propojení dat z pdf dokumentu a příslušným záznamem recenze v databázi. Jak již název napovídá reviewer_name je parametr obsahující celé jméno recenzenta, který daný článek hodnotí. Nejedná se tedy o autora článku, ale většinou o uživatele, který daný dokument stahuje. Parametr sid je identifikační číslo článku. Stejně jako každá recenze, tak má i článek své unikátní identifikační číslo. Při nahrávání souboru dochází ke kontrole tohoto čísla, pokud by soubor kontrolou neprošel, znamenalo by to problém v konzistenci dat. K této chybě by mohlo dojít pokud by se například změnil obsah databáze poté co byl soubor stažen a před tím než byl do systému opět nahrán. Parametr submission_name je celý název článku, který daná recenze hodnotí. Pokud je tento název moc dlouhý, dojde ke zkrácení a zbytek textu bude nahrazen třemi tečkami. Parametr submission_filename je celá cesta k originálnímu dokumentu článku ve formátu pdf. Pokud by se v dané cestě nenacházel soubor ve formátu pdf, dojde k vygenerování pouze formuláře. Poslední parametr review_html_footer slouží k definici vlastního zápatí poslední stránky. Obsah této proměnné musí být řetězec obsahující html kód.

3.2.2 Generování

Funkce pro generování dokumentu využívá hned několik nástrojů. K vytváření nových dokumentů je použita knihovna TCPDF, s již existujícími dokumenty je manipulováno pomocí nástroje PDFToolkit a formulář je navrhnut pomocí nástroje Adobe FormsCentral. Ve funkci jsou nejprve definovány názvy všech dočasných souborů, které budou následně využity. Poté dojde ke kontrole typu souboru článku. Pokud se jedná o pdf, tak dojde ke sloučení již existujícího souboru s formulářem a daného pdf souboru s článkem. Naopak pokud se o pdf soubor nejedná, tak bude kód funkce pokračovat pouze se samotným formulářem. Dále je zapotřebí přidat záhlaví s informacemi o recenzi. K tomu je využita již výše zmíněná třída PdfWithHeader. Vytvoříme tedy jednu stránku s žádaným záhlavím a poté tento dokument vykreslíme. Máme tedy formulář s článkem v jednom dokumentu a záhlaví v druhém. Ke spojení použijeme nástroj PDFToolkit, který umožňuje prokládání již existujících dokumentů. Proložení je docíleno tímto příkazem.

```
pdftk withoutHeader.pdf multistamp header.pdf output combined.pdf
```

Následně se pomocí knihovny TCPDF vytvoří další pdf soubor obsahující pouze hlavní nadpis. Tento soubor bude proložen již existujícím pdf soubo-

rem obsahujícím formulář, článek a teď i záhlaví. Jelikož chceme aby nástroj PDFToolkit takto spojil jen první stránku, na které se nadpis nachází, je nutné aby byl dokument s nadpisem ukončen prázdnou stránkou. PDFToolkit totiž při prokládání postupuje po stránkách a pokud má dokument stránek méně, dojde k opakování poslední strany. Kdyby měl tedy soubor s nadpisem jen jedinou stránku, byl by tento nadpis proložen všemi stranami výsledného dokumentu, stejně jako tomu bylo u záhlaví. V této chvíli jsme získali dokument s formulářem, připojeným článek, záhlavím a dynamickým nadpisem. Od finálního vzhledu nás dělí jen zápatí na poslední stránce. Opět tedy vytvoříme dokument pomocí knihovny TCPDF. Tentokrát bude obsahovat dvě prázdné stránky, třetí stránku obsahující zápatí a ukončovací prázdnou stránku. Prolnutím získáme již kompletní dokument se všemi informacemi. Tomuto dokumentu je ještě zapotřebí upravit metadata. Nejdříve tedy použijeme PDFToolkit na extrakci současných dat do souboru, k těmto datům poté přidáme identifikační čísla článku a recenze a tímto příkazem aktualizujeme metadata.

```
pdftk withoutMeta.pdf update_info metafile.txt output withMeta.pdf
```

Výsledný soubor je již finální a tak ho odešleme prohlížeči ke stažení.

3.3 Zpracování dokumentu

Zpracování dokumentu je realizováno touto funkcí s 3 parametry.

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

3.3.1 Parametry

Prví dva parametry, rid a sid, obsahují identifikační čísla článku a recenze. Jsou zde pouze pro kontrolu zda uživatel nahrává správný soubor na správné adrese webové stránky. Třetím parametrem je cesta k pdf dokumentu s který byl nahrán na server.

3.3.2 Zpracování

Při zpracování dojde nejprve k extrakci identifikačních čísel článku a recenze ze souboru. Tyto hodnoty jsou následně porovnány s čísly, které byly zadány jako parametry funkce pro zpracování. Pokud se liší číslo recenze, pravděpodobně se uživatel snaží pouze nahrát dokument s recenzí k jinému článku

než zvolil ve webové aplikaci. Pokud se však liší číslo článku, jedná se o velkou chybu konzistence. Nikdy by se totiž nemělo změnit identifikační číslo článku, bez změny čísla recenze. Po této kontrole jsou extrahována pomocí nástroje PDFToolkit datová pole. Výstup s informacemi o datových polích může vypadat například takto:

```
---
FieldType: Button
FieldName: Originality &#8211; Rate how origina_ir*LeqwssF5KR7JiEO01FA
FieldFlags: 49152
FieldValue: 1
FieldJustification : Left
FieldStateOption: 0
FieldStateOption: 1
FieldStateOption: 10
FieldStateOption: 2
FieldStateOption: 3
FieldStateOption: 4
FieldStateOption: 5
FieldStateOption: 6
FieldStateOption: 7
FieldStateOption: 8
FieldStateOption: 9
FieldStateOption: Off
---
FieldType: Text
FieldName: Internal comment (optional) &#8211; _rKkqBCjQ4mH7ygxrjhFaFw
FieldNameAlt: Internal comment (optional) &#8211; An internal message for the organizers:
FieldFlags: 4096
FieldValue: Text kter&#253; je obsa&#382;en v textov&#233;m poli Intern&#237; koment&#225;&#345;.
FieldJustification : Left
---
FieldType: Text
FieldName: fc-int01-generateAppearances
FieldFlags: 5
FieldJustification : Left
```

Jednotlivá datová pole jsou oddělena sérií tří pomlček. A informace o datových polí jsou rozdělena pod klíčová slova jako FieldType, FieldName a FieldValue. Dojde tedy k parsování a výsledné pole hodnot je předáno další funkci, která komunikuje s databází.

3.3.3 Aktualizace hodnot v databázi

Poslední funkce, která je při zpracování dokumentu použita se nazývá upload_to_DB_offline_rev
Tato funkce nejdříve zkontroluje zda jsou dodané hodnoty korektní. Uživatel se může pokusit nahrát dokument s recenzí, která byla již označena jako uzavřená, proto je nejdřív získán současný stav recenze, kterou se snažíme aktualizovat. Pokud je stav vyhovující dojde k nahrání všech hodnot do databáze.

4 Rozšiřitelnost

Ve složce se zdrojovým kódem je umístěn vyplnitelný formulář, který je používán pro generování dokumentů. Pokud je tento dokument nahrazen jiným formulářem vytvořeným pomoci programu Adobe FormsCentral, budou výsledné dokumenty obsahovat tento upravený formulář. Z toho důvodu složka obsahuje i soubor s koncovkou .fcdt, jedná se totiž o šablonu, pomocí které byl první formulář vytvořen. Stačí tedy tento soubor otevřít pomocí Adobe FormsCentral, provést změny a vygenerovat nový formulářový dokument ve formátu pdf. Při větších úpravách se však nevyhneme zásahům do kódu. Pokud se například změní počet polí, je nutné aktualizovat příkaz, který nahrává hodnoty do databáze. Veškerá práce s databází se nachází ve funkci upload_to_DB_offline_review_form. Také je nutné si uvědomit, že informační zápatí formuláře bude vždy vykresleno na třetí straně dokumentu. Pokud se tedy počet stránek formuláře změní, je nutné ubrat popřípadě přidat prázdné stránky z generovaného dokumentu, obsahujícího zápatí. Druhý přístup, na který jsem bohužel přišel moc pozdě je generování celého formuláře pomocí knihovny TCPDF. Když jsem při prvotním průzkumu tuto možnost zkoušel, tak jsem nedokázal najít způsob jak získat data z formulářů vytvořených tímto způsobem, to se mi podařilo až s formulářem vytvořeným pomocí Adobe FormsCentral. Navíc jsem si v té době naplno neuvědomoval, jak bude můj modul v realitě využit. Až při testování jsem přišel na to, že PDFToolkit dokáže zjistit obsah i datových polí vytvořených knihovnou TCPDF. Jsem přesvědčen o tom, že tento přístup je korektnější, proto se v této části pokusím vysvětlit postup, jak navrhnout vyplnitelné formuláře pomocí knihovny TCPDF

4.1 Formuláře v TCPDF

Tvorba formulářů s knihovnou TCPDF je opravdu jednoduchá. Velmi se podobá tvorbě klasických statických dokumentů. Nejprve je nutné vytvořit objekt TCPDF, do kterého můžeme následně začít vykreslovat buňky s textem.

```
$pdf = new TCPDF();
$pdf->AddPage();
$pdf->SetFont('helvetica', 'BI', 18);
$pdf->Cell(0, 5, 'Priklad formulare', 0, 1, 'C');
```

```
$pdf->Ln(10);
```

Funkce Cell vykreslí na současnou pozici text Příklad formuláře a poté dojde k odřádkování pomocí funkce Ln. Pomocí knihovny TCPDF je dále možné vytvořit textová pole, přepínače, zatrhávací políčka nebo například tlačítka. Všem těmto prvkům je přiřazeno jméno, díky kterému je možné s obsahem manipulovat pomocí JavaScriptu. To lze například použít k validaci ze strany uživatele. Pokud bychom chtěli přizpůsobit vzhled formuláře, můžeme k tomu použít funkci setFormDefaultProp, která nám umožní definovat vlastní okraje nebo například barvu textových polí.

```
$pdf->setFormDefaultProp(array('lineWidth'=>1,
    'borderStyle'=>'solid', 'fillColor'=>array(255, 255, 200),
    'strokeColor'=>array(255, 128, 128)));
```

Dále do dokumentu přidáme formulářové prvky. Pro vytvoření textového pole použijeme funkci TextField. Nejdříve přidáme popis textového pole, za něj vykreslíme samotné textové pole a poté opět odřádkujeme.

```
$pdf->Cell(35, 5, 'Textove pole:');
$pdf->TextField('field01', 50, 5);
$pdf->Ln(6);
```

První parametr funkce TextField, je identifikační název. Ten je následován šířkou a výškou textového pole. Poté do dokumentu přidáme seznam, pomocí funkce ComboBox.

```
$pdf->Cell(35, 5, 'Vyberove pole:');
$pdf->ComboBox('field02', 30, 5, array(array('', '-'), array('M',
    'Muz'), array('Z', 'Zena')));
$pdf->Ln(6);
```

První parametry jsou totožné, čtvrtý parametr je pole, které obsahuje všechny možnosti, které může uživatel při vyplňování zvolit. Dále přidáme několik přepínačů.

```
$pdf->Cell(35, 5, 'Prepinace:');
$pdf->RadioButton('field03', 5, array(), array(), 'Zvolil 1');
$pdf->Cell(35, 5, 'Jedna');
$pdf->Ln(6);
$pdf->Cell(35, 5, '');
$pdf->RadioButton('field03', 5, array(), array(), 'Zvolil 2',
    true);
```

```
$pdf->Cell(35, 5, 'Dva');
$pdf->Ln(6);
$pdf->Cell(35, 5, '');
$pdf->RadioButton('field03', 5, array(), array(), 'Zvolil 3');
$pdf->Cell(35, 5, 'Tri');
$pdf->Ln(6);
```

Pokud chceme aby měl uživatel na výběr z několika možností, je důležité aby měli přepínače stejný identifikační název. Druhý parametr funkce RadioButton je velikost přepínače, ta je následována poli, ve kterých je možné upřesnit vlastnosti a parametry, které lze využít při práci s JavaScriptem. Pátý parametr je hodnota, která bude uložena při zvolení dané možnosti. Jako šestý parametr je možné zadat výchozí hodnotu. V našem případě bude po otevření souboru, předem zvolena druhá možnost. Další prvek, který můžeme do našeho formuláře přidat, je zaškrtávací políčko.

```
$pdf->Cell(35, 5, 'Zaskrtavaci policko:');
$pdf->CheckBox('field04', 5, true, array(), array(), 'Zaskrtnuto');
$pdf->Ln(6);
```

Zaškrtávací políčko se podobá přepínači, akorát je změněno pořadí a výchozí stav pole je definován jako třetí parametr místo šestého. Poté co vytvoříme, pomocí předchozích funkcí náš formulář, je ještě nutné definovaný obsah vykreslit. K tomu použijeme funkci Output.

```
$pdf->Output('formular.pdf', 'D');
```

Druhý parametr této funkce určuje, kam má být výsledný dokument odeslán. Pokud je zvoleno D, bude soubor odeslán webovému prohlížeči. Druhá možnost je například zvolit F, poté bude soubor uložen lokálně na serveru. Dokument, který je vytvořen tímto skriptem je poté možno opět umístit do složky s modulem, který ho začne využívat ke generování dokumentů s recenzí. Poté je opět nutno zkontrolovat obsah pole, které vznikne po parsování a přizpůsobit jemu funkci na nahrávání hodnot do databáze.

5 Testování

Vytvořený modul bylo nutné také otestovat.

5.1 Kompatibilita

5.1.1 Webové prohlížeče

Nejprve jsem se rozhodl otestovat kompatibilitu s rozdílnými internetovými prohlížeči. Prvním prohlížečem, který jsem testoval byla Mozilla Firefox verze 47.0.1. Na tomto prohlížeči proběhlo vše správně, po stažení souboru se objeví dialogové okno, které umožňuje uživateli otevřít soubor pdf v doporučeném programu Acrobat Reader. A zpětné nahrání opět proběhlo bez problému. Dalším prohlížečem byl Microsoft Edge 40, který je v dnešní době nainstalován jako výchozí prohlížeč na počítačích používajících operační systém Microsoft Windows 10. I prohlížeč Edge neměl se stažením a zobrazením nejmenší problém. Poté jsem stažení otestoval i na prohlížeči Internet Explorer 11. Ten, podobně jako Edge, zobrazil dialogové okno, které otevřelo dokument automaticky v programu Acrobat Reader. Jako poslední jsem otestoval nejpoužívanější prohlížeč Google Chrome verze 57.0. U něj jsem narazil na problém. Google Chrome totiž umožňuje zobrazení obsahu pdf dokumentů ve vestavěném prohlížeči. Poté co jsem se pokusil otevřít stažený dokument, se otevřela nová záložka, ve které se dokument s formulářem zobrazil. Tento vestavěný prohlížeč dokonce umožňuje i vyplnění obsahu formuláře. Vyplněná data by se měla ukládat do souboru, jelikož však Google Chrome otevírá jen náhled, není možné otevřený soubor měnit. Pokud by uživatel chtěl vyplnit recenzi online, je k tomu na stránkách TSD určen webový formulář, proto nedává smysl vyplňování offline pdf furmuláře pomocí webového prohlížeče. Aby nedocházelo k nedorozumění, byl do záhlaví pdf formuláře raději přidán informační text, který upozorňuje uživatele aby formuláře raději vyplnil pomocí programu Acrobat Reader.

5.1.2 Editory

Vyplnění formuláře bylo otestováno jak v bezplatné verzi programu Acrobat Reader DC 17.9 tak v placené verzi Adobe Acrobat 11. Obě verze umožnily bezproblémové vyplnění a uložení formuláře. Otestoval jsem i speciální znakové sady. Pokud chce uživatel vyplnit formulář například v angličtině nebo

češtině, nedochází k žádnému problému. Uživatel může dokonce do formuláře uložit i speciální znaky, které se nacházejí v UTF-8. Při psaní těchto znaků si program Acrobat Reader sám dohledá a nainstaluje potřebné rozšíření. To co je možné do formuláře vyplnit není tedy omezeno formátem souboru, ale spíše verzí a rozšířením programu Acrobat Reader, kterým uživatel dokument vyplňuje. Co se týče velikosti, je tento formát skoro bez omezení. Uživatel může do textových polí vložit klidně několik stránek textu a nedojde k žádnému drastickému snížení rychlosti.

5.2 Generování a nahrávání

Při testování generování a nahrání mě zajímala především rychlost. Obával jsem se že pokud bude článek, který je připojen na konci formuláře, moc velký, dojde k velkému zpomalení. To se však nestalo. Zpracování bylo otestováno s pdf souborem o velikosti 10 MB. Soubor byl zpomalen spíše internetovým připojením, než samotným zpracováním.

5.3 Odhalené chyby

Během testování bylo odhaleno několik chyb. Ty nejzajímavější jsem se rozhodl popsat v této sekci.

5.3.1 Diakritika v nadpisu

Při vytváření dynamických textů jsem se soustředil na to, aby bylo možné vygenerovat dokumenty s českou diakritikou. Formuláře, které jsem generoval měli v nadpisu mé jméno, které obsahovalo písmeno "á". Až pozdější testy ukázali, že skript nedokázal vykreslit všechny diakritické znaky české abecedy. Naštěstí to nebyl problém v kódování, ale v použitém fontu. Základní fonty, které knihovna TCPDF používá, totiž všechny české znaky neumí.

- courier : Courier
- courierB : Courier Bold
- courierBI : Courier Bold Italic
- courierI : Courier Italic
- helvetica : Helvetica
- helveticaB : Helvetica Bold

- helveticaBI : Helvetica Bold Italic
- helveticaI : Helvetica Italic
- symbol : Symbol
- times : Times New Roman
- timesB : Times New Roman Bold
- timesBI : Times New Roman Bold Italic
- timesI : Times New Roman Italic
- zapfdingbats : Zapf Dingbats

Naštěstí je součástí knihovny TCPDF i více fontů, o kterých v dokumentaci není zmínka. Fonty freesans a freeserif dokáží zobrazit i českou diakritiku a tak jsou ideální pro náš modul. Pokud je nutné generovat české texty doporučuji využití freesans pro bezpatkové písmo a freeserif pro patkové.

5.3.2 Speciální znaky ve formuláři

Program Acrobat Reader umožňuje uživateli vyplnit do formuláře jakékoliv znaky. Při testování jsem zjistil, že tyto znaky nemusí být kompatibilní s nastavením tabulky v databázi a proto jsem přidal kontrolu. Při neúspěchu se uživateli zobrazí chybová hláška. Tato chyba však nastane jen ve velmi extrémních případech, když se uživatel snaží vyplnit do formuláře například emodži a jiné neobvyklé znaky.

5.3.3 Nekorektně vyplněný formulář

V první verzi mého modulu mohl uživatel vyplnit svůj formulář jakkoliv. Při nahrávání poté došlo k přemazání původní verze recenze tou novou. To že mohl uživatel nahrát i formulář jen s jedním vyplněným polem, by mohlo být potencionálně matoucí. Uživatel by mohl předpokládat, že nahráním dalšího formuláře, který obsahuje ostatní pole, dojde ke sloučení. Ve skutečnosti však dojde k přemazání původní odpovědi, odpovědí novou. Z toho důvodu jsem raději přidal kontrolu, která zjišťuje zda byl formulář vyplněn kompletně a žádná povinná pole nabyla vynechána. Současně jsem také přidal kontrolu rozmezí odpovědí. Uvědomil jsem si totiž, že pokud má uživatel na výběr z rozmezí 1-10, je možné upravit formát souboru a nahrát tak číslo mimo toto rozmezí, i pokud mu to neumožňuje vzhled formuláře.

5.3.4 Špatné pdf

Další chyba, která byla odhalena až testováním bylo zvolení špatného pdf. Když jsem vytvářel první verzi, tak jsem si neuvědomil, že by se uživatel mohl pokusit nahrát soubor, který neobsahuje námi vygenerovaný formulář pdf. Samozřejmě by tím ničeho nedocílil, protože by soubor nešel zpracovat, modul však selhal na jiné chybové hlášce a tak mohl uživatele zmást. Přidal jsem tedy další chybovou hlášku, která uživatele informuje o nahrání nekorektního souboru.

5.3.5 Dočasné soubory

Při běhu programu je vytvořeno velké množství dočasných souborů, které jsou poté smazány. Jeden test však odhalil, že při jedné speciální chybové hlášce nedojde k tomuto úklidu a soubory zůstanou ve složce s modulem. Tuto chybu nebylo těžké odstranit a k odstranění dočasných souborů nedojde jen v situaci, kdy je běh skriptu násilně ukončen.

5.3.6 Špatné parsování

Chyba, která se hledala nejhůře byla chyba v parsování textu. Byla to současně chyba, na kterou jsem přišel až poměrně pozdě. Parsování textu totiž na první pohled vypadalo funkčně a bezchybně. Velké množství testů prošlo bez problémů, pouze jediný měl zvláštní výsledek. V tomto testu byla vyplněna všechna pole až na pole předposlední. Při nahrání tohoto formuláře do databáze došlo k duplikaci poslední hodnoty i na místo předposlední. Díky tomuto testu jsem si uvědomil, že můj cyklus může v tomto speciálním případě načíst hodnotu nesprávného pole. Modul jsem opět upravil tak, aby ukládal i nevyplněná pole z formuláře.

5.3.7 Injekce obsahu buněk

Častý problém s parsováním textu, je ten, že pokud umožníte uživateli zadat libovolný řetězec, není možné zajistit, aby došlo ke správnému parsování ve sto procentech případů. Parsování se totiž řídí speciálními znaky a nebo řetězci, které mohou být obsaženy i v úsecích textu, které se snažíte získat. V praxi je tento problém řešet takzvaným escapováním znaků. Pokud chce uživatel zadat daný speciální znak, aniž by byl použit, je možné před něj umístit escapovací symbol. Při jednom z testů jsem se tedy pokusil vyplnit do textových polí vstup, který generuje nástroj PDFToolkit jako výstup. Ukázka tohoto výstupu je v kapitole 3.3.2. Tento test mi ověřil, že pokud by

uživatel zadal tři pomlčky, ukončil by tím dané pole a zbytek vstupu by se již nedal načíst. Naštěstí se mi povedlo tuto chybu opravit. PDFToolkit totiž umí reprezentovat konce řádek jako xml speciální znaky. Při vygenerování vstupu tedy dokáži rozlišit konce řádek, které vytvořil PDFToolkit a které vytvořil uživatel. Tento rozdíl mi umožnil přizpůsobit parser tak, aby nebylo možné vyplnit do polí takový text, který by nebylo možné zpracovat.

5.3.8 Obecné testování

Před zprovozněním celého modulu mi vedoucí práce umožnil vyzkoušet generování formulářů na již naplněné databázi. Při těchto testech bylo odhaleno, že u jedné konkrétní verze nadpisu, byl limit počtu písmen moc velký a došlo k zalomení. Dokument je i přesto čitelný, ale z důvodů jednotnosti, jsem limit snížil tak, aby všechny nadpisy zůstaly na jedné řádce. Také jsme zjistili, že někteří uživatelé při psaní článku nenahráli soubor ve formátu pdf. Proto byl modul upraven, aby v takovém případě umožnil alespoň stažení formuláře, bez přiloženého článku. Otestoval jsem desítky článků a tyto dvě chyby byly jediné. Generování a následné nahrání ostatních článků bylo bezproblémové.

Literatura

[1] Adobe. *End of support for FormsCentral* [online]. Adobe, 2016. [cit. 2017/03/29]. Dostupné z: `https://helpx.adobe.com/acrobat/kb/end-of-support-formscentral.html`.

[2] Adobe. *Adobe Designer* [online]. Adobe, 2003. [cit. 2017/03/29]. Dostupné z: `http://www.adobe.com/products/server/adobedesigner/`.

[3] *About fillable PDF forms and determining their capabilities* [online]. Adobe Systems Incorporated., 2016. [cit. 2017/03/29]. About fillable forms. Dostupné z: `https://helpx.adobe.com/acrobat/kb/create-fillable-pdf-forms-acrobat.html`.

[4] *A BRIEF HISTORY OF ADOBE SYSTEMS INC.* [online]. Investintech.com Inc., 2012. [cit. 2017/03/29]. History of the company. Dostupné z: `http://www.investintech.com/resources/articles/adobehistory/`.

[5] *Where has LiveCycle gone?* [online]. 2010. [cit. 2017/03/29]. User experience with LiveCycle. Dostupné z: `https://forums.adobe.com/thread/1187815`.

[6] *Library FPDF* [online]. FPDF, 2017. [cit. 2017/03/29]. FPDF Documentation. Dostupné z: `https://www.fpdf.org`.

[7] *Free PDF Document Importer* [online]. Setasign, 2007. [cit. 2017/03/29]. Dostupné z: `https://www.setasign.com/products/fpdi/about/`.

[8] Mathew, A. *Adobe Systems Incorporated Success Story* [online]. Successstory.com, 2017. [cit. 2017/03/29]. Webpage about successful stories. Dostupné z: `https://successstory.com/companies/adobe-systems-incorporated`.

[9] *mPDF Manual* [online]. Ian N Back, 2015. [cit. 2017/03/29]. Dostupné z: `https://mpdf.github.io/`.

[10] *Library PDFlib* [online]. PDFlib, 2017. [cit. 2017/03/29]. PDFlib Documentation. Dostupné z: `https://www.pdflib.com/`.

[11] PDFlib GmbH München, G. Reference Manual. *A library for generating PDF on the fly*. 1997, s. 226. Dostupné z: `https://www.uni-regensburg.de/rechenzentrum/medien/cms-support/pdflib-manual-5_02.pdf`.

[12] Rochkind, M. *Generating PDFs with PHP and FPDF (and TCPDF).* Amazon, 2013.

[13] *Snappy opensource github page* [online]. KnpLabs, 2016. [cit. 2017/03/29]. Dostupné z: `https://github.com/KnpLabs/snappy`.

[14] *Open Source PHP class for generating PDF documents* [online]. Nicola Asuni, 2004. [cit. 2017/03/29]. Dostupné z: `https://tcpdf.org/`.

[15] *wkhtmltopdf site* [online]. Ashish Kulkarni, 2013. [cit. 2017/03/29]. Dostupné z: `https://wkhtmltopdf.org/`.