PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO
KATEDRA INFORMATIKY

BAKALÁŘSKÁ PRÁCE

Editor PDF souborů

2012 Lukáš Gaál

Anotace

Práce se zabývá vytvořením počítačového programu pro manipulaci s PDF soubory. Program je určený pro osobní počítače a jeho hlavním rysem je moderní uživatelské rozhraní ovládané hlavně pomocí myši a techniky „Táhni a pusť“. Program je založen na knihovně iText, která se stará o práci s PDF soubory. Výsledkem je program PdfEditor*, který splňuje veškeré požadavky na grafické uživatelské rozhraní. Součástí práce je popis PDF souboru a jeho využití. Dále se práce věnuje použitým technologiím, návrhu a implementaci programu pro operační systém Microsoft Windows a platformu .NET Framework 4.0.*

Děkuji Mgr. Petru Krajčovi, Ph.D. za odborné vedení a rady při zpracování této bakalářské práce. Také děkuji všem, kteří mě po dobu studia podporovali a měli se mnou trpělivost.

Obsah

Seznam obrázků

Seznam tabulek

1. Úvod

Cílem práce je vytvořit grafické uživatelské rozhraní nad knihovnou pro tvorbu PDF dokumentů iText [1]. Aplikace umožňuje základní úpravy existujících PDF dokumentů. Konkrétně se jedná o rozdělení dokumentu na jednotlivé stránky, manipulace s jednotlivými stránkami (přeskládání, odstranění, rotace a kopírování stran mezi dokumenty), přidání vodoznaku do dokumentu. Do dokumentu je možné přidat heslo nebo heslo odstranit, pokud uživatel zná heslo původní. Dále je aplikace schopná vkládat do jednotlivých stran komentáře a jednoduchou grafiku, například šipky. Výsledná aplikace klade důraz zejména na uživatelskou přívětivost. Dalšími funkcemi aplikace jsou vkládání prázdných stránek do dokumentu, změna fontu písma, změna barvy a šířky u vložené grafiky a vytváření nových, prázdných dokumentů.

Pro řešení práce jsem měl na výběr mezi jazykem Java a C#. Knihovna iText je k dispozici pro oba programovací jazyky. Pro využití v jazyce C# je určena knihovna iTextSharp. Vzhledem k předchozím zkušenostem jsem si vybral C# a jako cílovou platformu si zvolil .NET Framework ve verzi 4.0. V této verzi jsou k dispozici knihovny Windows Presentation Foundation (WPF), které oproti starším Windows Forms poskytují snazší tvorbu bohatého grafického prostředí a jeho oddělení od logické části aplikace. Díky využití vrstvy DirectX pro vykreslování lze tvořit plynulé aplikace i při využití mnoha animací. Cílovým operačním systémem se stal pouze Microsoft Windows, i když jsem zamýšlel využití projektu MONO [14] ke spuštění i na OS Linux a dalších systémech. .NET 4.0 není momentálně tímto projektem podporován, proto není aplikaci možné provozovat na jiných platformách.

Mým záměrem je vytvořit jednoduchou aplikaci určenou pro osobní počítač, kde je hlavní okno rozděleno na dvě pracovní plochy. Na levé ploše se zobrazují jednotlivé stránky ve formě ikon nebo miniatur. Tyto stránky je možné jednoduše přetahovat po pracovní ploše pomocí „táhni a pusť“ techniky. Samozřejmostí programu je použití klávesových zkratek pro manipulaci stránek a textu jako vyjmout a vložit a další. Na pravé pracovní ploše se zobrazuje aktuálně vybraná stránka. Stránka jde přiblížit respektive oddálit. Na stránku se vkládá grafika ve formě čáry, šipky, komentáře a vodoznaku. S grafikou jde pohybovat pomocí ovládacích prvků, které jsou umístěny kolem ní. Součástí aplikace je i vestavěná nápověda namísto vytváření standardních balíčků MS Windows **chm* s nápovědou. Jednotlivé kapitoly nápovědy se zobrazují na levé pracovní ploše a vlastní text na pravé.

Knihovna iText je určena pro vytváření PDF, ale neumí PDF stránky vykreslit nebo uložit ve formě obrázku. Pro vykreslování PDF dokumentů existuje řada nástrojů, které si liší způsobem použití. Jedním z často používaných nástrojů je knihovna PdfRasterizer [15]. Knihovna je vytvořená pro .NET aplikace a snadno se používá, protože generuje stránku přímo jako objekt *FixedDocument*. Toto je výchozí objekt při práci s dokumenty ve WPF. V neplacené verzi knihovna bo-

hužel generuje do stránky výrazný vodoznak. Z tohoto důvodu jsem se rozhodl použít projekt PDFUtil [9] a knihovnu GhostScript [12], která neomezuje vzhled vykreslované stránky. PDFUtil i GhostScript jsou k dispozici pod licencí GPL. Použití těchto nástrojů si vyžádalo, aby program PDFEditor byl také distribuovaný jako svobodný software pod licencí GPL [13].

2. Popis problematiky

V této kapitole se budeme zabývat popisem PDF souboru, jeho historický vývoj a přehled verzí. Dále je zde umístěn stručný popis již existujících, volně šířených programů a hlavní rysy, kterými se program *PdfEditor* odlišuje.

2.1. Jak vzniklo PDF

Portable Document Format (PDF)[4] byl navrhnut společností Adobe Systems Incorporated v roce 1993 jako náhrada za jazyk PostScrip (PS) . PS je interpretovaný programovací jazyk pro tvorbu tiskových sestav. Jeho hlavním úkolem je popisovat vzhled jednotlivých částí jako jsou text, obrázky a různé grafické tvary. Jeho součástí je také rozhraní k ovládání tiskáren a zobrazovacích zařízení jako obrazovka monitoru. Se vzrůstající potřebou výměny informací v elektronické podobě a následné potřebě informace zobrazit na všech zařízeních a operačních systémech stejně firma Adobe vyvinula nový formát Interchange PostScript. Formát byl později přejmenován na PDF. První program pro práci se jmenoval Acrobat a nebyl dodáván zadarmo. Součástí programu byl i Adobe Exchange (umožňuje výměnu dat s ostatními uživateli programu Acrobat, tvorbu dokumentů, tisk a přidávání poznámek) a Adobe Distiller, který sloužil pro konverzi z formátu PS do PDF. V další verzi se program rozdělil na Adobe Reader (k dispozici zdarma) a Adobe Acrobat Program.

2.2. Výhody PDF

PDF není na rozdíl od PS programovací jazyk, ale definice formátu binárního souboru, který se skládá z množství objektů. Jimm King [5] z firmy Adobe PDF označuje jako objektově orientovaný Post Script. Mezi výhody PDF lze zařadit nezávislost stránek. V PS musíte před interpretací jakékoli stránky interpretovat všechny předcházející stránky. Stránka v PDF obsahuje všechny potřebné informace ve formě odkazů na objekty, jako jsou samotný text, font písma, obrázky a další.

Z důvodů rozšíření formátu mezi různé, specifické obory, Adobe v roce 2007 dovolila tvorbu ovladačů a aplikací, které produkují výstup ve formě PDF. S tímto krokem souviselo otevření formátu pro veřejnost a vytvoření standardu ISO-32000-1 [4]. Každá firma nebo instituce nyní může psát rozšíření některé základní

verze PDF. Rozšíření nemusí čekat na začlenění do ISO standardu a výsledná verze PDF se značí pomocí čtyřpísmenného prefixu a základní verze PDF. Např. firma Adobe má prefix ADBE.

2.3. Přehled verzí

Přehled verzí a hlavních rozšíření najdete v tab. 2. na str. 36. Tabulka byla převzata z [1]. Od verze 1.7 se číslování uvádí i s názvem rozšíření. Tyto rozšíření upřesňují prvky, které se můžou v PDF souboru vyskytovat.

2.4. Typy PDF

Existuje několik souborů pravidel v závislosti na oblasti použití PDF (stavitelství, vodohospodářství, zdravotnictví). Pomocí tohoto dělení můžeme při kontrole PDF říct, zda je validní. Tyto pravidla vychází z ISO normy a zapisují se jako *PDF/x*, kde x nabývá následujících hodnot.

- A. Určeno pro archivaci. Je to podmnožina pravidel definovaných ve verzi 1.4 a má za úkol zaručit správné zobrazovaní dokumentů nezávisle na programu, který PDF vytvořil. Výsledkem je, že PDF soubor se bude zobrazovat stejně i za mnoho let od jeho vytvoření.
- A level B. Zde není povolené šifrování. Všechny fonty vyskytující se v textu musí být součástí souboru. Nesmí se zde objevit žádný multimediální obsah. JavaScript a spouštění externích souborů je rovněž zakázáno.
- A level A. Obsahuje všechno z levelu B a navíc ve struktuře PDF musí být použity tagy. Pomocí tagů lze snáze extrahovat text a další obsah.
- E. Soubor pravidel pro inženýrství (Engineering). Založeno na PDF 1.6. Obsahuje podporu pro poznámky a komentáře. Podpora kreslení složitých objektů ve 3D.
- UA. Podpora pro nevidomé. Obrázky obsahují doprovodný text. Struktura textu musí odpovídat logickému pořadí výskytu v dokumentu.
- H. Využití ve zdravotnictví. Předpisuje určité zabezpečení pro obsah jako karta pacienta, rentgen atd.
- X. Využití v nakladatelstvích a tiskárnách. Předpisuje v jakém formátu jsou uloženy barvy. Například ve verzi 1 se jedná o CMYK formát. PDF musí mít přibalené všechny použité fonty a obsahovat informaci o čísle verze. Nejsou zde povoleny žádné aktivní prvky jako formulářové objekty nebo multimédia.

2.5. Vnitřní struktura PDF

Existuje 8 základních datových typů používaných v PDF. Jejich výčet je uveden v tab. 1. na str. 35 a slouží k definování informací použitých objektů v PDF. Mimo tyto základní datové typy se ve struktuře často opakují i typy *Indirect Object* a *Indirect Reference*. První z nich slouží k definici objektu v PDF a druhý k odkazování se na něj. Zápis každého *Indirect Object* začíná jedinečným číslem, následuje číslo revize/generace objektu (používá se u inkrementálního vytváření PDF) a končí značkou „obj“. K ukončení zápisu objektu se používá značka „endobj“. V příkladu níže vidíme definice objektu
č. 3. (3 0 obj - první stránka PDF) a v něm odkaz na objekt č. 5 (5 0 R - nese informaci o řetězci Ahoj Světe), kde písmeno R znamená odkaz (reference) na objekt.

Struktura PDF souboru se dá rozdělit na 4 části.

- Hlavička - Každý platný soubor začíná znaky %PDF- verze a na dalším řádku je několik bajtů. Druhý řádek se používá, aby ostatní editory nepovažovaly PDF za textový soubor.
- Tělo - Obsahuje všechny nepřímé objekty, ze kterých se skládá dokument. Najdeme zde stránky, komentáře, poznámky, fonty a další.
- Tabulka odkazů - Uchovává ukazatele na nepřímé objekty definované v těle dokumentu jako offset (vzdálenost od počátku) v souboru PDF. Pro získání libovolného objektu nemusíme procházet celý soubor sekvenčně. Místo toho využíváme náhodný přístup, což vede ke zrychlení při vykreslování stránek.
- Zápatí - V zápatí najdeme slovo *startxref*, které následuje adresa tabulky odkazů. Dokument končí znaky %%EOF.

Zde je vidět zjednodušený výpis PDF s rozdělením na jednotlivé sekce. PDF obsahuje jednu stránku s textem „Ahoj Světe!“. Není zde použitá komprese FlatDecode. Příklad je převzatý z článku [6] a mírně upravený.

```
=======HLAVIČKA============================
%PDF-1.4
=======TĚLO================================
1 0 obj
<<
  /Type /Catalog
  /Pages 2 0 R
>>
endobj

2 0 obj
<<
```

```
  /Type /Pages
  /MediaBox [ 0 0 200 200 ]
  /Count 1
  /Kids [ 3 0 R ]
>>
endobj

3 0 obj
<<
  /Type /Page
  /Parent 2 0 R
  /Resources <<
    /Font <<
      /F1 4 0 R
    >>
  >>
  /Contents 5 0 R
>>
endobj

4 0 obj
<<
  /Type /Font
  /Subtype /Type1
  /BaseFont /Times-Roman
>>
endobj

5 0 obj  % page content
<<
  /Length 44
>>
stream
BT
70 50 TD
/F1 12 Tf
(Ahoj Světe!) Tj
ET
endstream
endobj
=======TABULKA KŘÍŽOVÝCH ODKAZŮ=========
xref
0 6
0000000000 65535 f
0000000010 00000 n
```

```
0000000079 00000 n
0000000173 00000 n
0000000301 00000 n
0000000380 00000 n
trailer
<<
  /Size 6
  /Root 1 0 R
>>
======ZÁPATÍ============================
startxref
492
%%EOF
```

Ze struktury souboru je již zřejmé, proč nemůžeme změnit například řetězec. Po jakékoli změně je potřeba zanést změny do tabulky odkazů a přepočítat adresy ostatních objektů. PDF ve velké míře také používají kompresi aplikované na objekty. Tato komprese se dá při vytváření pomocí iTextu omezit nebo úplně potlačit.

Pro samotné vykreslování je definovaná sada příkazů umístěných v objektu obsahujících „stream“. Jedná se o jakýsi mini jazyk, který používá postfixový zápis a množství grafických a textových operátorů. Zájemce může najít kompletní popis operátorů a jejich použití ve standardu ISO [4]. Zde jsou popsány pouze příkazy použité výše v příkladu u objektu č. 5.

- BT (begin text) signalizuje začátek textu.
- 70 50 TD znamená posunutí začátku textu na souřadnice [70,50].
- /F1 12 Tf mění font na font F1 o velikosti 12 bodů. Font F1 je svázaný s objektem č. 4 ve zdrojovém slovníku na první PDF stránce v objektu č. 3.
- (Ahoj Světe!) Tj zobrazuje sekvenci znaků Ahoj Světe!.
- ET (end text) signalizuje konec textu.

2.6. Existující nástroje

PDF vznikl jako formát určený pouze pro čtení, který má zaručovat identické zobrazení na všech zařízeních. I přesto je k dispozici mnoho nástrojů, které dokáží určité části souboru změnit. Tyto nástroje se dají rozdělit na on-line editory pro základní úpravy a aplikace určené pro osobní počítač. Aplikace určené pro osobní počítače se velmi liší ve způsobu ovládání a možnostech úprav. Následuje přehled některých zástupců podle způsobů ovládání.

Příkazová řádka

- Pdftk - jedná se o nejrozšířenější nástroj na úpravu PDF souborů pomocí příkazového řádku a skriptů. Poskytuje funkce jako slučování dokumentů, rotace, přesun a odstranění stránek. PDF lze zaheslovat nebo naopak heslo odstranit, vyplňovat formuláře, vkládat vodoznak a další. Pro jeho snadnější ovládání existuje grafická nadstavba *PDFTK Builder*.

Okenní (desktopové) aplikace

- Foxit Reader - umožňuje vkládat různou grafiku a text nebo existující text měnit. Přidává komentáře. Exportuje data z formulářů nebo dokáže data do formulářů importovat. Neumí vytvořit nový dokument ani přidávat formulářové prvky. Funkce záměna textu se mi bohužel neprojevila.

- Pdf-XChangeViewer - do existujících dokumentů přidává jednoduchou grafiku, poznámky a komentáře. Ve verzi Pro dokáže přidávat nové, prázdné stránky, nebo stránky z jiných dokumentů. V této verzi také dokáže stránky mazat nebo dělat výřezy části stránky.

- OpenOffice - s doinstalovaným rozšířením od firmy Oracle umí PDF soubor zobrazit díky převodu do formátu *odf*. Do takto zobrazeného dokumentu lze přidávat další prvky z repertoáru OpenOffice Draw. Výsledný soubor lze exportovat do PDF. Ne všechny elementy PDF se ale konvertují správně. Například jsou zde problémy se správným formátováním textu, tabulek a zobrazení obrázků.

- Inkscape - dokáže otevřít některou stránku z PDF souboru. S textem a obrázky zachází jako s vlastními objekty. Lze je otáčet, posouvat a mazat. Dále je na stránku možné vkládat vektorovou grafiku. Výslednou stránku lze uložit ve formátu PDF.

- BeCyPDFMetaEdit - umožňuje měnit uložená metadata jako jméno autora, čas vytvoření a jiné. Dokáže změnit výchozí zobrazení stránek (jedna nebo dvě na stránku), číslování stránek. Jako nejužitečnější zde shledávám možnost změnit záložky na stránky. Jedná se o kompletní změnu struktury, názvu záložek, odkazu na stránky, včetně přiblížení na odkazované stránce.

Online editory

- TouchPDF - umí zobrazit obsah PDF, přidávat textové poznámky, tvary a obrázky. Dále umožňuje sloučit nebo rozdělit PDF a s menšími nepřesnostmi dokáže nahrazovat i text.

- PDFescape - přidává různou grafiku a formulářové prvky. Stránky dokáže otočit, přesunout a oříznout.

Po standardizaci formátu se vytvořila i řada projektů k poskytování nejrůznější funkcionality ve formě knihoven. Některé slouží i pro samotné vykreslování PDF. Mezi nejznámější patří Adobe PDF Library, GNU PDF, iText a Poppler. Knihovny jsou napsány v mnoha programovacích jazycích.

2.7. Výhody proti konkurenci

Program *PdfEditor* se oproti existujícím nástrojům snaží nabídnout snadnější práci při změně stránek v dokumentu. Manipulace se provádí pohodlně pomocí myši, kdy se vybrané stránky přetáhnou na místo určení v rámci aktuálního dokumentu nebo dokumentu jiného. Program dokáže vytvářet nové dokumenty a vkládat do dokumentů prázdné stránky velikosti A4. Dále umí otáčet a posunovat vloženou grafiku jako komentář nebo vodoznak.

U ostatních aplikací jsem se s manipulací stránek nesetkal. Pokud ano, je uživatel nucen vyplnit několik dialogů, kde pomocí přepínačů a textových polí nastaví, co se se stránkou má dělat nebo odkud se má stránka zkopírovat.

3. Uživatelský manuál

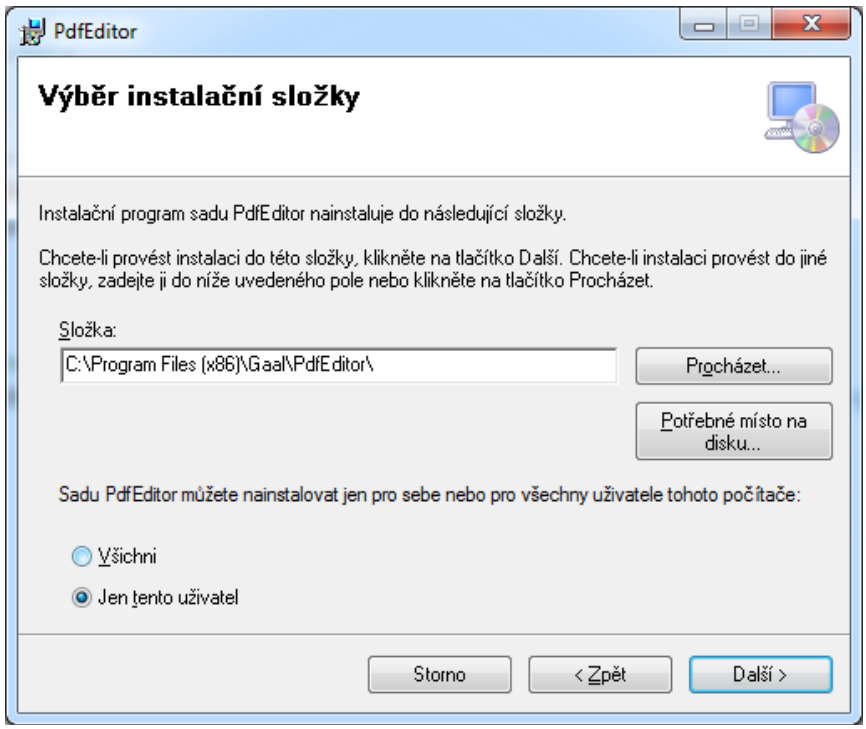
3.1. Požadavky na systém

Pro bezproblémovou funkčnost programu je vyžadovaná platforma Microsoft .NET Framework 4 Client Profile (x86 a x64) běžící na operačním systému Microsoft Windows XP Service Pack 3 a vyšší s 512MB operační paměti. Doporučené rozlišení monitoru pro start aplikace je 800x600. Program bez problému běží i na procesorech Intel Atom a integrovaných grafických čipech. Zde je ovšem potřeba počítat s pomalejším načítáním jednotlivých stránek a velkých dokumentů.

3.2. Instalace

Instalace se provádí otevřením *Setup.msi* souboru v adresáři *Install* na přiloženém disku. Po spuštění se zobrazí instalační dialogové okno 1., kde je uživatel vyzván k zadání základních informací. Uživatel zde může změnit cestu k adresáři, kam bude program nainstalován. Pro instalaci je třeba mít správcovská práva. V případě, že na cílovém počítači není nainstalován nezbytný .NET Framowork 4.0, bude uživateli zobrazena stránka na internetu, kde si vše může stáhnout. Další možností je chybějící Framework ručně nainstalovat z adresáře *Install*. V tomto případě dojde k instalaci z DVD. Na přiloženém DVD je i instalátor pro Windows Installer 3.1 pro případ, že na cílovém počítači jsou zakázané aktualizace a Windows Installer není doinstalován automaticky.

Během instalace se ve výchozím nastavení ve složce *Program Files* vytvoří adresář *Gaal\PdfEditor* s potřebnými soubory. Dále vzniknou odkazy na program *PdfEditor*. Jeden na ploše a další v nabídce Start.



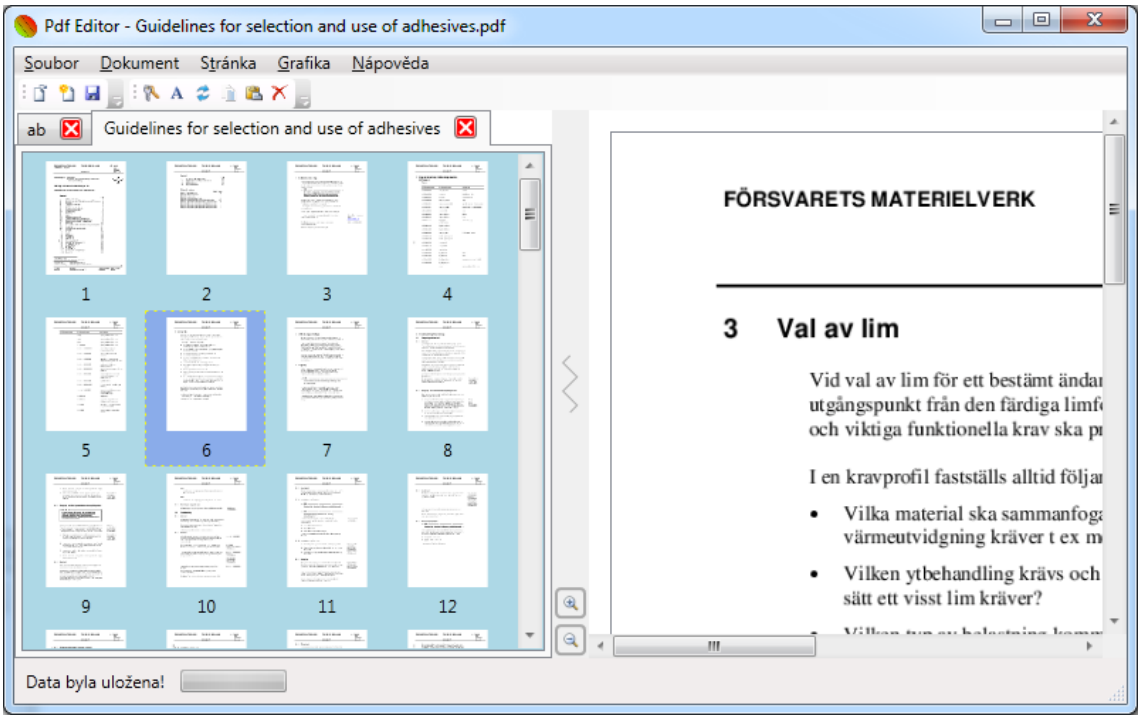
Obrázek 1. Instalační dialogové okno

3.3. Jak s programem pracovat

3.3.1. Hlavní okno

Po startu aplikace se uživateli zobrazí hlavní okno (obr. 2.). Okno je přehledně rozděleno na 4 hlavní části, které poskytují prostor pro přímou manipulaci PDF. Záhlaví okna nese název aplikace, který následuje název aktuálně otevřeného souboru. Pod ní se nalézá panel menu. V menu uživatel najde všechny funkce programu. Pro zpřístupnění rychlých voleb je k dispozici panel nástrojů. Hlavní pracovní plocha je rozdělena na dvě menší. Jedna slouží pro zobrazení stránek a druhá pro vlastní editaci grafiky na stránce. Ve spodní části se nalézá informační panel. Uživatel zde nalezne informaci o zdárném uložení souboru a indikátor průběhu, který informuje o načítání dokumentu nebo stránky.

Oknu můžeme podle potřeby měnit velikost tažením za libovolný okraj nebo jej minimalizovat na lištu Start.



Obrázek 2. Program PdfEditor

3.3.2. Panel menu

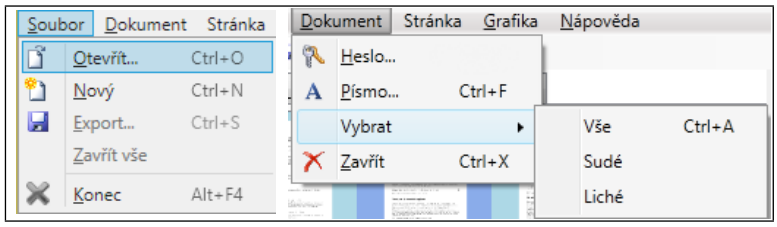
V tomto panelu má uživatel pohromadě všechny funkce programu. Pokud není některou funkci možné provést, daná položka se jeví jako neaktivní a nejde na ni kliknout. U některých položek jsou také zobrazeny klávesové zkratky.

Pod záložkou *Soubor* (obr. 3.) se nalézají funkce jako otevření a uložení dokumentu, vytvoření nového dokumentu a ukončení programu. Pokud jsou v dokumentu neuložené změny, uživatel je dotázán, jestli chce aplikaci skutečně zavřít, nebo jestli se mají změny uložit.

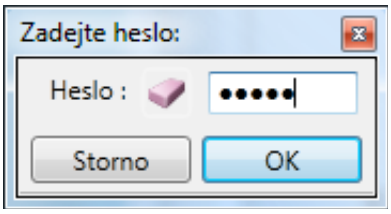
Pod záložkou *Dokument* (obr. 3.) jsou umístěny funkce pro provedení změn v rámci otevřeného dokumentu. *Heslo* zobrazí uživateli dialog (obr. 4.), kde je možno nastavit nové heslo nebo staré heslo odstranit použitím ikonky gumy. *Písmo* zobrazí dialog změny fontu a velikosti písma (obr. 5.). Nastavení se promítne pro výchozí písmo všech vkládaných komentářů a současně se změní písmo aktuálně zvýrazněného komentáře.

Pod záložkou *Stránka* (obr. 6.) jsou umístěny funkce spojené s používáním stránek jako kopírování, odstranění, rotace, vložení do dokumentu. *Nová* vloží do dokumentu prázdnou stránku formátu A4 a *Vrátit* vloží poslední odstraněné nebo vyjmuté stránky aktuálního dokumentu na místo, odkud byly odstraněny.

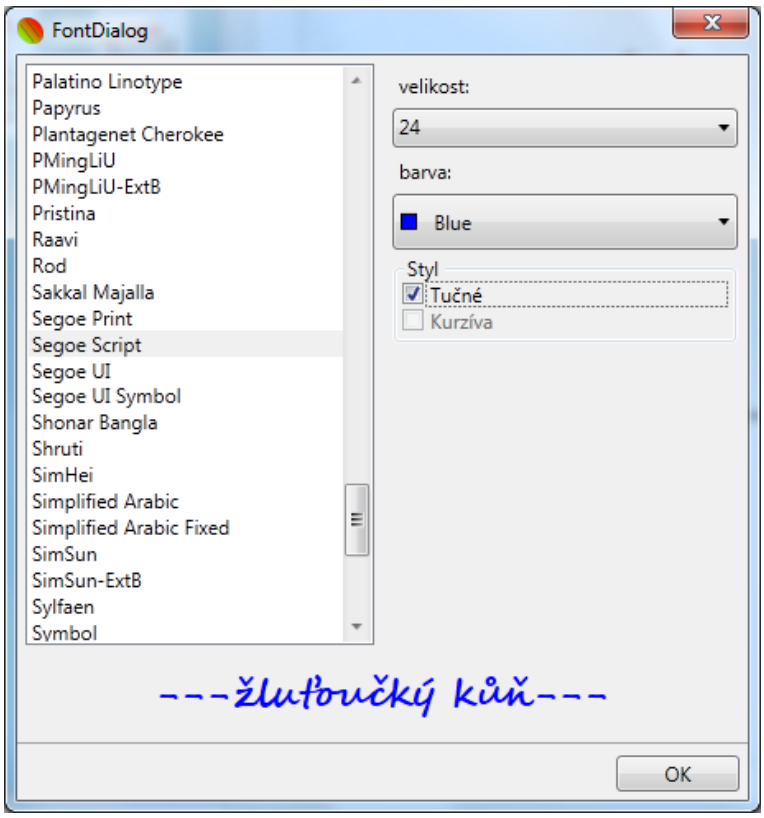
Záložka *Grafika* (obr. 6.) obsahuje všechny grafické prvky, které je možno na stránku vkládat. Obsahuje možnost vložení šipky, čáry, komentáře a vodo-



Obrázek 3. Menu Soubor a Dokument

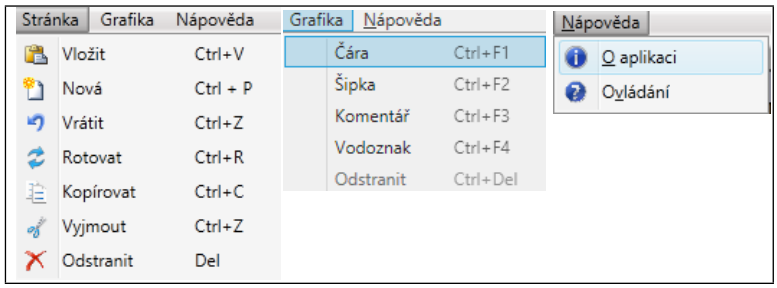


Obrázek 4. Dialog změny hesla

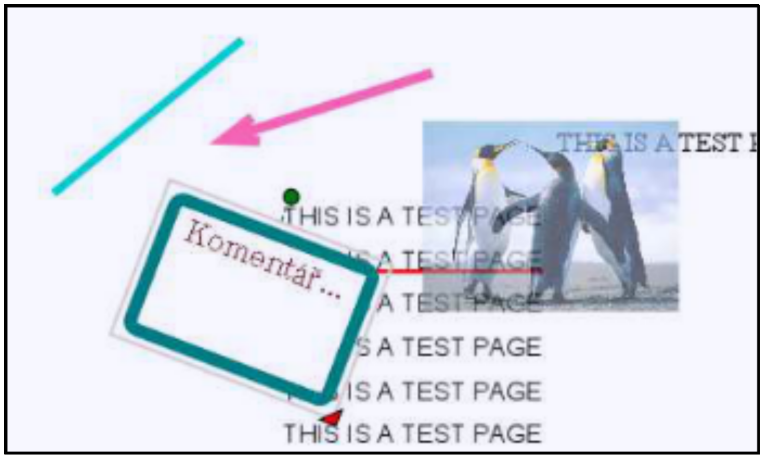


Obrázek 5. Dialog Fontů

znaku. Příklady vložené grafiky najdeme na obr. 7. Vybranou grafiku je také možno odstranit pomocí poslední položky *Odstranit*.



Obrázek 6. Menu Stránka, Grafika a Nápověda



Obrázek 7. Vložená čára, šipka, komentář a vodoznak

Pod poslední záložkou (obr. 6.) se skrývá vestavěná nápověda (obr. 8.) a dialog, který nese informace o programu a autorovi.

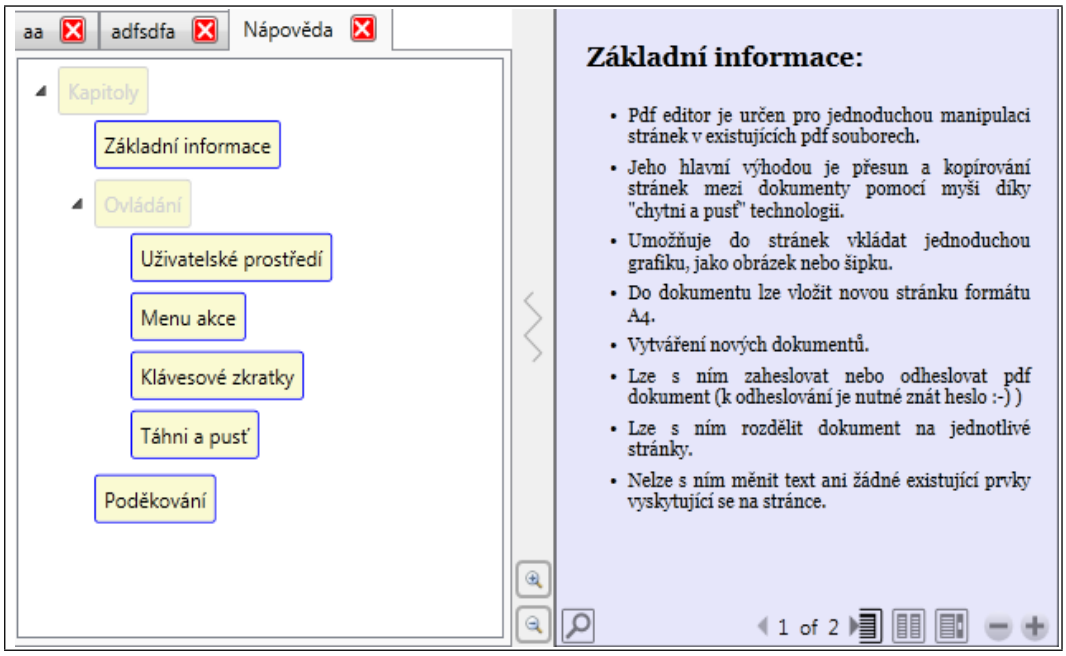
3.3.3. Klávesové zkratky

Klávesové zkratky slouží zejména zkušenějším uživatelům k rychlému vyvolání funkce bez vybírání z panelu menu. Přehled zkratek najdete v tab. 3. na str. 37. Pokud není možné zvolenou funkci v dané chvíli provést, nebude provedena (např. vložit novou stránku do vestavěné nápovědy).

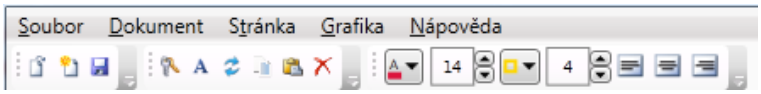
3.3.4. Nástrojová lišta

Nástrojová lišta (obr. 9.) obsahuje nejčastěji používané funkce ve formě tlačítek s obrázky. Na tyto obrázky se dá kliknout myší, pokud je taková funkce aktuálně k dispozici. Při ponechání kurzoru myši nad obrázkem se zobrazí jeho textový popis.

Následuje popis jednotlivých funkcí nástrojové lišty zleva. Otevřít, Nový dokument, Uložit, Změna hesla, Změna fontu, Rotovat doprava, Kopírovat stránku,



Obrázek 8. Vestavěná nápověda



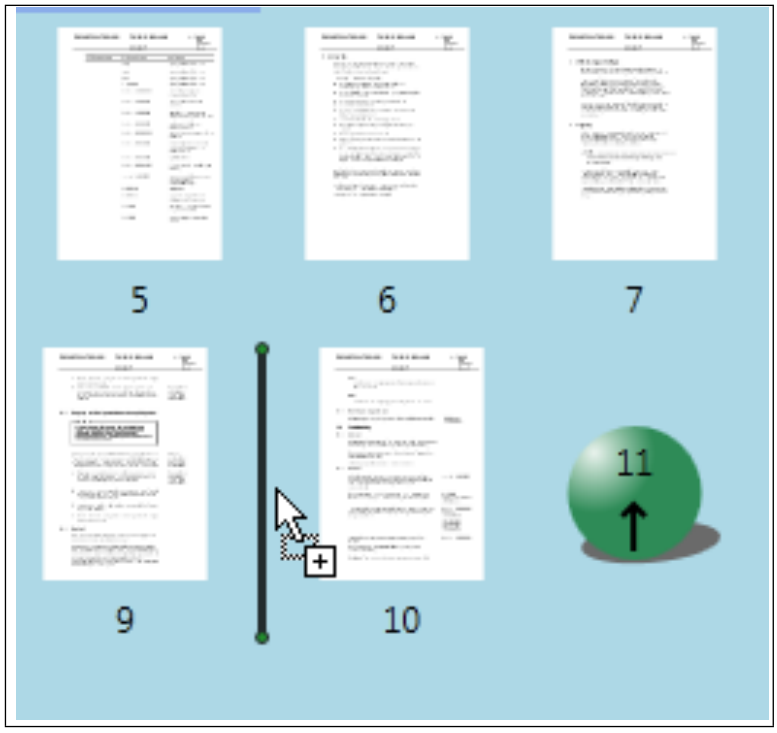
Obrázek 9. Nástrojová lišta

Vložit stránku, Smazat stránku, Barva písma nebo čáry, Velikost písma nebo Tloušťka čáry, Barva rámečku, Šířka rámečku, Zarovnání textu vlevo, na střed a vpravo.

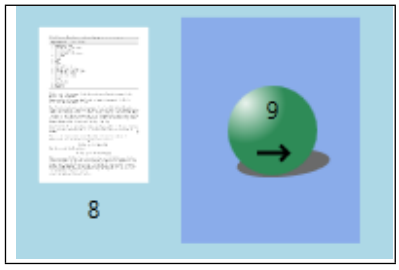
3.3.5. Pracovní plocha

Pracovní plocha se dělí na pravou a levou. Na levé se zobrazují stránky ve formě miniatur s číslem stránky nebo zelené ikony se šipkou, která symbolizuje natočení stránky (obr. 11.). Jedná se o posunovací seznam, ve kterém se dá pohybovat pomocí šipek na klávesnici nebo pomocí kolečka myši. Hlavní účel plochy je pohodlné přesouvání stránek v rámci dokumentu nebo mezi dokumenty. Posunování se provádí zvýrazněním stránky pomocí myši nebo klávesnice a tažením myši po pracovní ploše s levým tlačítkem myši stále stlačeným. Během posunování myši je pomocí animace ostatních stránek a změnou kurzoru myši signalizováno, kam budou stránky vloženy (obr. 10.). Stránky je možné také kopírovat při současném podržení tlačítka *Control* a puštěním levého tlačítka myši. Při přesouvání stránky mezi dokumenty je potřeba táhnutím myši najet nad záložku se jménem cílového dokumentu a po zobrazení dokumentu pokračovat

ve výběru místa vložení jako v předešlém případě.



Obrázek 10. Vložení stránky pomocí „Táhni a pusť“



Obrázek 11. Miniatury stránek

V levé části se po otevření vestavěné nápovědy zobrazují témata nápovědy (obr. 8.). Zavření dokumentu se standardně provádí kliknutím na kříž v záložce s názvem dokumentu. Obsahuje-li dokument neuložené změny je uživateli zobrazen dialog s potvrzením zavření.

Pravá část plochy slouží ke zobrazení detailu vybrané stránky nebo vybraného tématu nápovědy. Mezi plochami se nachází posuvník, kterým se může měnit poměr velikosti ploch. Na tomto posuvníku jsou situovány i tlačítka pro přiblížení a oddálení detailu stránky nebo textu nápovědy. Na plochu se vkládají grafické objekty šipka, komentář apod. Po výběru zvolené grafiky z menu musí uživatel

zadat dva body, které určují výchozí velikost a umístění grafiky. Výběr bodu se provádí myší. Grafickým objektem lze manipulovat, pokud je označen myší. Po označení se kolem něj zobrazí vodící body. V jeden okamžik může být vybrán pouze jeden grafický objekt. Po kliknutí na prázdné místo v dokumentu grafický objekt ztratí zvýraznění. Vložená grafika se dá posouvat a rotovat pomocí myši, která mění kurzor v závislosti na možnostech manipulace.

3.4. Odinstalace

Odinstalování programu je možné dvěma způsoby. Otevřením okna v Ovládacích Panelech - Přidat nebo odebrat programy. Toto okno můžeme rychle spustit klávesovou zkratkou *Win+R* a napsáním *appwiz.cpl*. Druhou možností je využít odkazu na odinstalátor v nabídce Start\PdfEditor.

4. Programátorská část

Program jsem vytvořil v programovacím jazyce C# pro platformu .NET a Windows Presentation Foundation (WPF) s využitím několika externích knihoven. Místo vybrané knihovny pro vykreslení PDF *PdfRasterizer* jsem přistoupil ke knihovně *GhostScript* [12], která je napsána v nativním kódu, ale poskytuje zhruba dvojnásobné zrychlení při vytváření obrazu stránek. Do vytvořených stránek nevkládá velice výrazný vodoznak, a tím umožňuje uživateli přehlednější práci se stránkou.

4.1. Návrh programu

Při návrhu programu jsem počítal s oddělením grafické a logické části programu. Řešením se stalo rozdělení do několika vrstev (obr. 12.). V programu je využit návrhový vzor „Model-View-ViewModel“, který doporučuje firma Microsoft při psaní složitějších programů pomocí technologie WPF. Problematika WPF je detailně popsána v knize pana Petzolda [2].

Následuje stručný popis jednotlivých vrstev.

- *View* - v této vrstvě se nachází samotná okna a ostatní prvky, které uživatel může vidět na obrazovce. Komunikace s nižší vrstvou je zabezpečena pomocí *Datových* a *Příkazových* vazeb popsaných níže.

- *ViewModel* vrstva se stará o správný chod aplikace. Je zodpovědná za svůj vnitřní stav a spolupráci svých součástí. O veškerých změnách informuje nadřazenou vrstvu pomocí událostí a datových vazeb.

  Datové vazby slouží k vzájemnému propojení objektů na různých vrstvách, ale i mezi objekty, které implementují *INotifyPropertyChanged* nebo jsou

potomkem *DependencyObject* z knihovny WPF. Například prvek *TextBox* a jeho vlastnost *Text* je svázaný s objektem aplikace *Comment.Text*. Mechanismus v pozadí hlídá veškeré změny provedené v libovolném objektu a tuto změnu okamžitě promítá na druhý objekt. Chování datové vazby se dá ve velké míře ovlivnit. Například můžeme nastavit pouze jednocestnou vazbu. Taktéž se dá v XAML definovat konvertor, který převádí objekty jednoho typu na druhý nebo svázanou hodnotu vhodně upravuje. WPF k některým konverzím používá své vlastní konvertory. Například *String to Int*. Informace o tom, který konvertor se má používat, je součástí metadat u jednotlivých tříd.

Na obr. 13. na str. 24 je znázorněna hierarchie základních tříd určených pro vrstvu *ViewModel*. Objekty těchto tříd se nejčastěji umísťují do vlastnosti *DataContext* vizuálních prvků WPF. Po definování vazby na objekt ho mechanismus datových vazeb hledá právě zde.

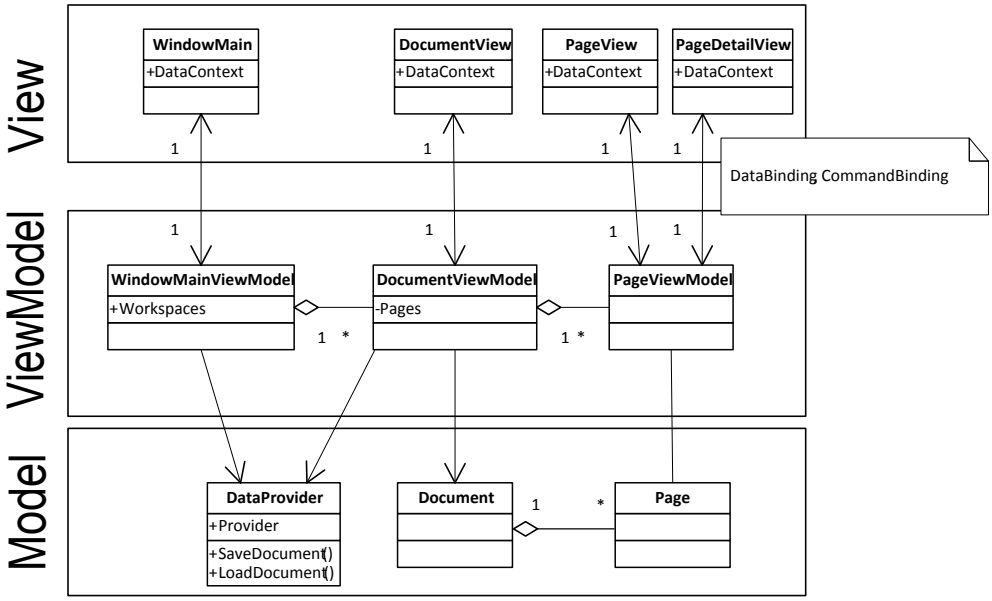
Změny v objektech na *ViewModel* vrstvě se provádí pomocí *CommandBinding* (příkazových vazeb). Objekty na této vrstvě implementují rozhraní *ICommand*. Rozhraní definuje dvě metody. Jsou to *Execute* a *CanExecute*. Ovládací prvky, které používají *CommandBindig* pomoci nich poznají, zda je možné daný příkaz provést. Když *CanExecute* vrátí *true*, je zavolaná funkce *Execute*.

- *Model* vrstva obsahuje pomocné třídy pro práci s daty na disku nebo objekty z externích knihoven. Objekty této vrstvy vytváří nadřazená vrstva a u pomocných objektů se během života programu neprovádí žádné změny.

Většina programového kódu pro tvorbu grafické části je psaná v jazyce XAML (Extensible Application Markup Language). Jedná se o značkovací jazyk podobný XHTML. V XAML se definuje vizuální vzhled oken a ostatních částí. Pomocí stylů a šablon se dá rychle napsat, jak mají vypadat nevizuální prvky. Soubor XAML můžou dále upravovat grafici v programu Microsoft Expression Blend, aniž by znali samotný kód pracující s ovládacími prvky. Programátor je schopen psát i dynamickou změnu ovládacích prvků na základě změn jejich stavů. Například u tlačítka máme možnost změnit barvu zvýraznění při kliknutí díky definování vazby na změnu vlastnosti *IsClicked*. V XAML se taky pohodlně definují takzvané datové vazby popsané v textu výše. Některé pokročilé funkce jsou ovšem napsány přímo v kódu C#. Jedná se zejména o implementaci „Táhni a pusť“ v souboru *DocumentView.cs*.

Dále je uveden příklad definování šablony pro objekt *Comment*. Lze si všimnout definování datové vazby například pro vlastnost *Text* nebo barvu písma. Také je zde definovaná vazba mezi položkou kontextového menu a příkazem pro uzavření komentáře.

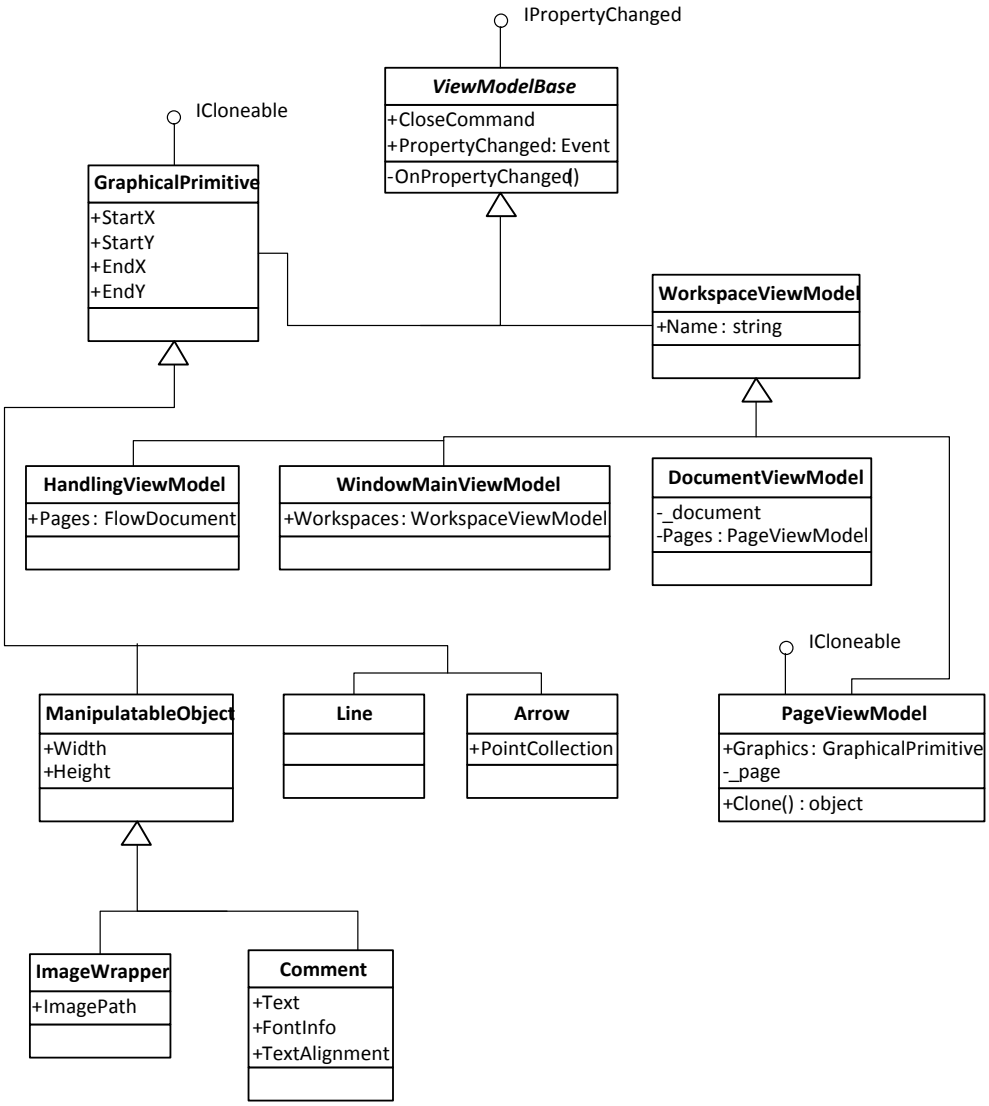
```
<DataTemplate DataType="{x:Type clr:Comment}">
```



Obrázek 12. Rozdělení do vrstev

```
 <Grid
   Panel.ZIndex="20"
   Width="{Binding Width, Mode=TwoWay}"
   Height="{Binding Height, Mode=TwoWay}" >
   <Border CornerRadius="{Binding CornerRadius}"
        BorderThickness="{Binding BorderThickness}"
        Background="White"
        SnapsToDevicePixels="True"
        PreviewMouseDown="Border_PreviewMouseDown">
...
     <TextBox.ContextMenu>
...
         <MenuItem Header="Odstranit" Command="{Binding CloseCommand}" />
     </TextBox.ContextMenu>
    </TextBox>
   </Border>
   <ContentControl Style="{StaticResource ResourceKey=ResizeRotateStyle}"
                clr:ContentDesignerBase.ApplicableElement="{Binding}"
                Focusable="False" >
   </ContentControl>
 </Grid>
</DataTemplate>
```

Následuje ukázka změny zvýraznění u objektu *ListBoxItem* pro signalizaci *KeyboardFocus*. Uplatnění najdeme například při potlačení ozdobného rámečku u vkládaných grafických elementů v detailu stránky. Celý detail stránky je ve sku-



Obrázek 13. Diagram tříd kostry programu

tečnosti implementován jako *ListBox* se změněným stylem.

```
<Style x:Key="keyboardFocus">
   <Style.Setters>
    <Setter Property="Control.Template" >
     <Setter.Value >
      <ControlTemplate>
          <Rectangle Stroke="Yellow" StrokeThickness="1"
     StrokeDashArray="2 4" />
      </ControlTemplate>
     </Setter.Value>
    </Setter>
```

```
   </Style.Setters>
</Style>

<Style TargetType="ListBoxItem">
 <Setter Property="FocusVisualStyle"
 Value="{DynamicResource keyboardFocus}" />
</Style/
```

Z důvodů ukládání velkého počtu obrázků na disk a jejich načítání grafickým subsystémem WPF jsem nebyl schopen spolehlivě všechny obrázky z disku při ukončení programu odstranit. Soubory byly uzamčeny. Pro vyřešení tohoto problému jsem vytvořil podprogram *TempDeleter.exe*. Podprogram jako svůj argument při spouštění přijímá cestu k textovému souboru. Tento soubor obsahuje seznam všech dočasně vytvořených souborů programem *PdfEditor*. Spouští se po ukončení programu *PdfEditor* a postupně vymaže všechny dočasné soubory.

4.2. Hlavní třídy programu

***DialogHelper*:** Statická třída, která nabízí velikou škálu statických metod pro tvorbu dialogových oken. Návrhový vzor MVVM se nehodí pro přímé využívání více oken. Touto třídou jsem zajistil oddělení logické a grafické části programu při interakci s uživatelem.

***GSWrapper*:** K vykreslování stránek je použita knihovna *GhostScript*, která je napsána v jazyce C. Pro volání knihovních funkcí a správu paměti mimo automatickou správu paměti platformy .NET bylo potřeba vytvořit pomocnou třídu. Tato třída je převzata z projektu PDFUtil. Třída byla mírně upravena pro použití v programu. Nalézají se zde dvě hlavní statické metody:

- CreateImage(. . .) - ve formě vstupních parametrů definujeme umístění PDF souboru, jméno souboru pro uložení obrázku PDF stránky, číslo stránky a případně heslo. Výsledný obrázek je uložen v rozlišení 100 bodů na palec.
- CreateThumbnailFile(. . .) - vstupní parametry má stejné jako předchozí funkce. Výstupní soubor s obrázkem má rozlišení 11 bodů na palec.

***DataProvider*:** Je navržená jako jedináček a slouží k načtení PDF dokumentu a vytvoření objektu *Document*. Stejně tak je určena k ukládání změněných dokumentů a vytváření nových. Jeho hlavní metody jsou *SaveDocument* a *LoadDocument*. Ukládání souboru probíhá ve třech fázích. V první se jedná o zkopírování všech stránek z jednotlivých dokumentů do jednoho

dočasného souboru. Pokud se jedná o novou stránku, je místo kopírování vytvořena nová stránka formátu A4.

Dočasný soubor je potom znovu otevřen a ke každé stránce je přidána informace o rotaci a dodatečná grafika spolu s transformační maticí. Matice má za úkol grafický prvek správně posunout a natočit. Dočasný soubor je potom přejmenován na původní jméno. Ve třetí fázi, která se spouští v závislosti na parametru předaného v *ExportDialogViewModel.SaveSeparately* dochází k rozložení dokumentu na jednotlivé stránky a jejich uložení do souborů s číselnou příponou podle umístění stránky v rámci dokumentu.

Tato třída je jediné místo pro práci s objekty externí knihovny iText a po celou dobu života si hlídá seznam objektů použitých k načtení PDF stejně, jako seznam objektů (např. *PdfReader*), které byly použity k zápisu změn do jednotlivých stránek. Po zápisu změn se totiž tyto změny uchovávají ve vnitřním bufferu a jsou znovu zapsány i do ostatních upravovaných stránek. Z tohoto důvodu musí být některé objekty knihovny znovu vytvořeny.

Při vkládání komentáře se současně s textem ukládá i informace o použitém fontu. Většina fontů operačního systému, které uživatel bude při práci s programem používat, je chráněna autorským zákonem, a proto jsem se rozhodl nepřibalovat je do výsledného PDF, ale použít jenom informaci o jejich jméně.

***PageDetailTemplateSelector*:** Třída vytvořená z důvodů použití objektu *PageViewModel* na dvou místech v programu. Poprvé jako miniatura na levé pracovní ploše a dále jako detail stránky na pravé pracovní ploše. V XAML jsou proto definovány dvě šablony, které určují vzhled objektu. Metoda *SelectTemplateobjecto* vrací šablonu, která bude použita.

***EditorCommand*:** Implementuje rozhraní definované ve WPF *ICommand* a metody *CanExecute a Execute*. Slouží k definování tzv. „CommandBinding“ v jazyce XAML u tlačítek a položek menu.

***WindowViewModelBase*:** Jedná se o základní třídu, od které dědí všechny třídy na *ViewModel* vrstvě. Je zde definována událost *RequestClose* a příkaz *CloseCommand*. Pomocí navázání na událost *RequestClose* je nadřazený objekt informován o zámyslu uzavření. V programu to nejčastěji znamená vyřazení z *Pages* (jedná se o kolekci stránek, kterou spravuje *DocumentViewModel*) nebo *Workspaces* (kolekce otevřených dokumentů, kterou spravuje *WindowMainViewModel*).

***WorkspaceViewModel*:** Je následníkem *ViewModelBase* a obsahuje jediný veřejný atribut *Name*. Od této třídy dědí všechny ostatní, které jsou určeny ke zobrazení na levé pracovní ploše. Atribut *Name* se použije pro název záložky a jako část textu v záhlaví programu.

***WindowMainViewModel*:** Hlavní třída celého programu. Zde jsou příkazy pro načítaní a ukládání dokumentů. Třída poskytuje kolekci všech otevřených dokumentů pod vlastností *Workspaces*. Zabezpečuje uzavírání těchto podřízených instancí pomocí správného navázání na jejich události.

***DocumentViewModel*:** Udržuje si v kolekci *Pages* seznam všech instancí třídy *PageViewModel*. Při přidání nebo odebrání má za úkol správně navázat nebo zrušit delegáty metod na jejich události. Dále je zde logika pro změnu hesla.

***Page*, *Document*:** Objekty reprezentující PDF soubor na nejnižší vrstvě. Obsahují informace o počtu stran, původním umístění, hesle, velikosti atd. Tyto objekty jsou zabaleny v *PageViewModel* a *DocumentViewModel*. Jejich hlavní náplní je zpřístupnit vlastnosti svých podřízených objektů. Provedené změny v dokumentu nebo stránce se projeví pouze v třídách na *ViewModel* vrstvě.

***PageViewModel*:** Třída obsahuje kromě zpřístupňování vlastností objektu na nižší vrstvě také důležitý mechanismus pro tvorbu miniatur a náhledů stránek. Třída je zodpovědná za synchronizaci vláken během získávání těchto miniatur, protože knihovna *GhostScript* nepodporuje volání z více vláken. Z tohoto důvodu existují dvě statické fronty, do kterých se postupně ukládají požadavky na vykreslení. Jedna fronta je pro požadavky na miniatury stránek a druhá pro celou stránku. Jako synchronizační primitivum je použito „Signal and Wait“ a k operacím s frontami standardní „Lock“ jazyka C#.

Hlavní metody:

- *AddToThumbnailQueuePageViewModel* - přidá do interní fronty pro miniatury objekt *PageViewModel*. Tento objekt obsahuje informaci o zdrojovém souboru a čísle stránky. Tato stránka bude po zpracování uložena jako soubor s obrázkem.
- *AddToRenderPageQueuePageViewModel* - slouží ke stejnému účelu jako předchozí. Výsledný obrázek bude uložen s vyšším rozlišením.
- *GSPumpMethod* - metoda, která je spuštěna na novém vlákně ve statickém konstruktoru třídy. Prostředí .NET volá statický konstruktor pouze jednou, a to když vytváří první instanci třídy nebo se přistupuje ke statickému členu třídy. Pokud ve frontách existují nevyřízené požadavky, spouští se podle priorit. Přednostně je odbavována fronta s požadavky na vykreslení celé stránky. V momentě, kdy je prázdná, obsluhuje se fronta s požadavky vykreslení miniatur stránek.

***ContentMover, ContentRotater*:** Třídy odvozené od *ContentDesignerBase*. Jedná se o třídy na vrstvě *View* sloužící pro změnu polohy vložené grafiky

na stránce. Definují hlavní vlastnost *ApplicableElemet*, která musí být typu *ManipulatableObject*. Při událostech „DragDelta“ jsou potom těmto elementům měněny vlastnosti jako rotace nebo souřadnice umístění na stránce. Základem pro vytvoření tříd byl internetový projekt „WPF Diagram Designer“ [10], ve kterém autor popisuje manipulace vizuálních prvků WPF na panelu *Canvas*.

4.3. Překlad programu

Pro vývoj programu jsem použil Microsoft Visual Studio 2010 spolu s operačním systémem Microsoft Windows 7 Professional. V možnosti sestavení je jako cílová platforma vybrána X86 s parametrem BUILD. Projekt je dělen na několik menších částí z důvodu znovupoužitelnosti. Tyto části jsou nejčastěji zkompilovány do podoby knihoven. Následuje výčet těchto částí.

TempDeleter - jedná se o spustitelný podprogram. Tento podprogram se spouští s jedním argumentem. Jako argument se předává jméno souboru, který obsahuje cesty k dočasným souborům. Dočasné soubory se po spuštění pokouší odstranit.

GSWrapper - slouží jako rozhraní pro práci s knihovnou gsdll32.dll [12], která je napsána v nativním kódu.

help_lib - obsahuje obrázky pro vestavěnou nápovědu aplikace. Knihovna byla vytvořena z důvodu volných XAML souborů, které se na ni odkazují.

Projekt - zde se nachází celá aplikace *PdfEditor*.

Installer - jeho účel je vytvořit instalační soubor *Setup.msi* a do něj přibalit všechny komponenty, na kterých je aplikace závislá.

Pro překlad dokumentace byla použita sada programů z balíku TexLive [11]. Z důvodu častého opakování příkazů byl vytvořen pomocný skript pro překlad. Tento skript najdete v přiloženém DVD pod jménem *translate.bat* v adresáři */doc*.

4.4. Komplikace při vývoji

Během programování jsem narazil na celou řadu problémů. Jako hlavní bych zmínil problém při kopírování objektů v jazyce C#. Dokumentace doporučuje pro vytvoření kopie objektu použít metody třídy *MemberWiseClone* a na kopírovanou třídu implementovat rozhraní *ICloneable*, kde v metodě *Clone* provádíte kopírování všech proměnných uložených na haldě. Problém ovšem nastává, když třída poskytuje události. V případě navázání delegátů na události se tyto delegáty

někdy zkopírují a někdy ne. Také podpůrné technologie v prostředí .NET udržují v paměti ukazatel na objekt, který provádí vazbu. Když potom tento objekt zrušíme, nedojde k jeho odstranění automatickou správou paměti. Zde dochází k úniku paměti. Dále, když na takto zkopírovaný objekt navážeme vlastní delegáty, běhové prostředí .NET je volá pouze někdy. Někdy se splete a zavolá je pro objekt, který tyto vazby definoval dříve. Pomocí reflexe jsem se pokoušel tyto nechtěné delegáty odstranit, ale výsledek nebyl stoprocentní. Rozhodl jsem se proto k vytvoření privátního kopírovacího konstruktoru, kde dochází ke zkopírování potřebných proměnných. Máme tímto plnou kontrolu nad kopírovanými proměnnými.

Externí knihovna *PdfRasterizer* někdy z neznámého důvodu padala a zamykala zdrojové PDF soubory. Toto byl další důvod, proč jsem se rozhodl změnit knihovnu pro vykreslování PDF na *Ghostscript*.

Při vytváření instalátoru *Setup.msi* ve Visual Studiu jsem chtěl přibalit instalátory ke všem závislostem programu. Jedná se o instalátory Windows Installer 3.1 a .NET Framework 4.0 installer. Visual Studio je automaticky zkopíruje ze svých knihoven do cílového adresáře k *Setup.msi* podle nastaveného jazyka. Při hledání si ovšem kontroluje, jestli jsou tyto instalátory taky v cílovém jazyce a když ne, odmítne projekt přeložit. Českou lokalizaci jsem nenašel, a tak je instalátor v anglickém jazyce.

Dlouho byl hlavní nedostatek programu ve zobrazování dokumentu s velkým počtem stránek. K zobrazení miniatur stránek jsem použil *ListBox* systému WPF. Tento ovládací prvek ve výchozí šabloně umožňuje zobrazování velkého počtu položek pomocí panelu *VirtualizedStackPanel*. Panel vytváří vizuální kontejnery pro položky, které se na něm nacházejí v části, která je zrovna viditelná. Procesu postupného vytváření položek se říká virtualizace.

VirtualizedStackPanel je určený pro zobrazování jednotlivých položek pouze pod sebou. Z tohoto důvodu jsem přistoupil k napsání vlastního ovládacího prvku *VirtualizedWrapPanel*, který dokáže zobrazovat více položek vedle sebe v závislosti na dostupné šířce okna. Minimální počet položek na řádek je 1.

V dokumentaci k WPF [3] není problematika vytvoření virtuálního panelu dostatečně popsána, a proto jsem využil internetových článků [7] a [8], které se zabývají implementací rozhraní *IScrollInfo* a abstraktní třídou *VirtualizingPanel*. Druhá jmenovaná třída slouží jako bázová třída ke všem panelům, které potřebují jakýmkoliv způsobem zobrazovat velký počet prvků.

Pro urychlení výpočtu velikosti panelu a zobrazovaných položek jsem přistoupil k zadání pevné velikosti vizuálních kontejnerů pomocí vlastností *ItemHeight* a *ItemWidth*. Další možností je procházet všechny prvky a měřit jejich potřebnou velikost. Toto se však ukázalo časově a výpočetně náročnější. Pro nasazení v programu stačí první, jednodušší metoda.

Panel jednou vytvořené kontejnery při jejich opuštění viditelné oblasti neruší. Zde je ještě prostor pro zvýšení výkonu pomocí recyklace a vytvoření vnitřního bufferu zobrazených a zrušených kontejnerů. Pro srovnání ještě dodám, že

vytvoření všech vizuálních kontejnerů pro testovací soubor *test400* trvalo 15s. Po implementaci virtualizace se čas dostal pod 1s.

4.5. Známé nedostatky programu

Zde bych chtěl upozornit na jeden nedostatek programu, který se zatím nepodařilo odstranit.

- Při současném otevření PDF a vestavěné nápovědy nastává problém při kopírování stránek pomocí myši. Jakmile se objeví kurzor nad panelem s nápovědou a následně se vrátí nad původní dokument, ztratí se informace o kopírovaných stránkách.

Závěr

Výsledkem bakalářské práce je počítačový program pro snadnou manipulaci stránek v PDF. Program je určený pro operační systém Microsoft Windows a platformu .NET Framework. Na program jsou kladeny dva hlavní požadavky. Pohodlné ovládání pomocí myši, zejména přesouvání stránek metodou *Táhni a pusť* a využití externí knihovny iText pro manipulaci PDF.

Strukturu práce lze rozdělit do tří částí. Nejdříve v textu popisuji formát PDF a možnosti využití knihovny iText. V další části je uživatelský manuál k programu. V poslední části najdeme rozdělení programu do vrstev a popis hlavních tříd z programátorského hlediska.

Zadané cíle práce se mi podařilo splnit. Program umí otevřít PDF dokument, přesouvat a mazat stránky. Na jednotlivé stránky dokáže vkládat jednoduchou grafiku jako šipku a komentář. K dispozici je možnost dokument chránit heslem nebo heslo odstranit. K odstranění hesla je však zapotřebí znát heslo původní.

Jako nejdůležitější rozšíření programu bych viděl podporu funkcí *Zpět* a *Vpřed*, se kterými jsem při návrhu struktury programu nepočítal. Program je nyní komplikovaný a nebylo by jednoduché tyto funkce doprogramovat. Jako částečné řešení pro aktuální verzi programu jsem implementoval funkci *Zpět* pouze pro vyjmuté a smazané stránky. V další verzi bych rád umožnil úpravu existujících objektů na stránce (např. tabulky, text, obrázky). Aktuální verze knihovny iText takovéto úpravy neumožňuje nebo je tato funkčnost v experimentální fázi a není ji doporučeno používat.

Conclusions

The result of this bachelor thesis is a computer program *PdfEditor* for easy PDF manipulation. The Program is aimed at Microsoft Windows operating system and .NET Framework platform. The Program is subject to two main requirements. Easy handling using mouse, especially moving pages by *Drag and Drop* feature and use of external library iText for PDF manipulation.

The structure of the thesis can be divided into three parts. First of all there is PDF format described in the text and then possibilities of use of iText library. In the next section there is user's program manual. In the last part of thesis, one can find program separation into layers and description of the main classes from a programmer point of view.

I managed to fulfill assigned objectives. The program is capable of opening PDF document, moving and deleting pages. One can place a simple graphics objects, for instance, an arrow or comment, into a page detail. It is possible to put a password into a document or remove it. It is necessary to know the old password to perform this operation.

As the most important thing to improve I see *Undo* and *Redo* function support which I did not calculate with during the structure designing of the program. The Program is quite complicated now and it would be not easy to implement this functions. As a partial solution for actual program version I implemented function *Undo* only to return removed or deleted pages. In next version I would like to allow modification of existing objects on the page (e.g. tables, text, pictures). Actual version of iText library does not support such modifications or this functionality is in an experimental phase and it is not recommended to use it.

Reference

[1] Lowagie, Bruno. *iText in Action Second Edition.* Manning, Greenwich, 2011.

[2] Petzold, Charles. *Mistrovství ve Windows Presentation Foundation.* Computer Press, Brno, 2008.

[3] Elektronická dokumentace WPF, [1.1.2013], `http://msdn.microsoft.com/en-us/library/ms754130(v=vs.100).aspx`.

[4] Norma ISO pro PDF, elektronická dokumentace, [1.12.2012], `http://www.adobe.com/devnet/pdf.html`.

[5] Přednáška Jimma Kinga, prezentace, [16.12.2012], `http://home.comcast.net/~jk05/presentations/jcking01.pdf`.

[6] Úvod do PDF, www článek, [1.12.2012], `http://www.gnupdf.org/Introduction\_to\_PDF`.

[7] Implementace IScrollInfo, www články, [1.12.2012], `http://www.switchonthecode.com/tutorials/wpf-tutorial-implementing-iscrollinfo`, `http://blogs.msdn.com/b/bencon/archive/2006/12/09/iscrollinfo-tutorial-part-iv.aspx`.

[8] Virtualizace ve WPF, www článek, [1.12.2012], `http://blogs.msdn.com/b/dancre/archive/2006/02/06/implementing-a-virtualized-panel-in-wpf-avalon.aspx`.

[9] Projekt Renauda Bompoise PdfUtil, www stránka, [4.7.2012], `http://community.devexpress.com/forums/p/55582/188207.aspx\#188207` .

[10] Příklad pro manipulaci grafických prvků ve WPF, www článek, [22.6.2012], `http://www.codeproject.com/Articles/22952/WPF-Diagram-Designer-Part-1` .

[11] Sdružení uživatelů texu, www stránka, [12.12.2012], `http://www.cstug.cz` .

[12] Domovská stránka projektu GhostScript, www stránka, [1.12.2102], `http://www.artifex.com` .

[13] GNU GPL licence, www stránka, [1.12.2012], http://www.gnu.org/licenses/gpl.html .

[14] Projekt MONO, www stránka, [1.12.2012], http://www.mono-project.com/WPF .

[15] Projekt PdfRasterizer, www stránka, [3.5.2012], http://www.tallcomponents.com .

A. Tabulky

Typ Pdf	**Popis**
Boolean	stejný účel jako v ostatních jazycích
Numeric Object	používá se k popisu souřadnic, rozměrů. Jedná se o číslo
String	sekvence znaků uzavřená v závorkách
Name	používají se jako identifikátory objektů, ve struktuře PDF začínají lomítkem
Array	jedno dimenzionální pole, například umístění obrázku [10 10 100 150]
Dictionary	tabulka klíčů a hodnot. Jedná se o hašovací tabulku
Stream	kolekce bajtů, nejčastěji komprimovaná vložená data. v PDF umístěn mezi *stream* a *endstream*
Null Object	hodnota null jako v ostatních jazycích

Tabulka 1. Základní datové typy PDF

Verze Pdf	Rok	Verze	Rozšíření
1.0	1993	Acrobat 1	Zobrazení textu a grafiky na monitoru, tisk na tiskárně.
1.1	1994	Acrobat 2	Chránění heslem, externí odkazy, barva nezávislá na zařízení
1.2	1996	Acrobat 3	Komprese obsahu, interaktivní formuláře, podpora asijských znaků
1.3	1999	Acrobat 4	souborové přílohy, digitální podpis, číslování stránek
1.4	2001	Acrobat 5	128 bitové šifrování, průhlednost, podpora tagů pro čtečky obrazovek
1.5	2003	Acrobat 6	rozšíření komprese a šifrování, vestavěné multimédia
1.6	2004	Acrobat 7	podpora pro šifrovací standard AES, škálování při tisku
1.7	2006	Acrobat 8	více možností pro tisk, zlepšení 3D
1.7 ex3	2008	Acrobat 9	ISO-32000-1
2.0	2011	Acrobat 10	ISO-32000-2

Tabulka 2. Vývoj verzí PDF

Zkratka	Funkce
Ctrl+O	Vyvolá standardní dialog otevření PDF souboru.
Ctrl+N	Vyvolá dialog vytvoření nového, prázdného dokumentu.
Ctrl+E	Vyvolá dialog uložení aktuálně zobrazeného dokumentu.
Ctrl+F4	Ukonči aplikaci.
Ctrl+P	Do aktuálního dokumentu vloží novou stránku formátu A4.
Ctrl+F	Zobrazí dialog výběru stylu písma jako na obr. 5..
Ctrl+C	Zkopíruje do paměti aktuálně vybrané stránky nebo zvýrazněný text v komentáři.
Ctrl+X	Vyjme do paměti aktuálně vybrané stránky nebo zvýrazněný text v komentáři.
Ctrl+V	Je-li v paměti zkopírovaná stránka, vloží ji do aktuálního dokumentu nebo do komentáře, pokud je aktivní.
Ctrl+W	Zavře otevřený dokument.
Ctrl+Z	Vrátí poslední vymazané stránky na jejich původní místa.
Ctrl+R	Otočí vybrané stránky o 90 stupňů.
Ctrl+X	Vyjme vybrané stránky do paměti.
Ctrl+1	Vloží do aktuální stránky čáru.
Ctrl+2	Vloží do aktuální stránky šipku.
Ctrl+3	Vloží do aktuální stránky komentář.
Ctrl+4	Vloží do aktuální stránky vodoznak.
Del	Odstraní vybraný grafické prvek na stránce. Pokud takový není, vymaže vybrané stránky.

Tabulka 3. Klávesové zkratky

B. Obsah přiloženého DVD

`bin/`
: Instalátor *Setup.msi* programu a další program *setup.exe* spustitelný přímo z DVD k instalaci podpůrných technologií z internetu. Adresář obsahuje i instalátor pro .NET Framework 4.0 a Windows Installer 3.1.

`doc/`
: Dokumentace práce ve formátu PDF, vytvořená dle závazného stylu KI PřF pro diplomové práce, včetně všech příloh, a všechny soubory nutné pro bezproblémové vygenerování PDF souboru dokumentace, tj. zdrojový text dokumentace, vložené obrázky a překládací skript *translate.bat*.

`src/`
: Kompletní zdrojové texty programu *PdfEditor* se všemi potřebnými (převzatými) zdrojovými texty, knihovnami a dalšími soubory pro bezproblémové vytvoření spustitelných verzí programu.

`readme.txt`
: Instrukce pro instalaci a spuštění programu *PdfEditor*, včetně požadavků pro jeho provoz.

Navíc DVD obsahuje:

`data/`
: Ukázková a testovací data použitá v práci a pro potřeby obhajoby práce.

`gpl_licence.txt`
: Kompletní znění GNU GPL licence. Její zahrnutí do programu vyžadují licenční podmínky externích knihoven a části zdrojových textů.

U všech materiálů převzatých z internetu je tato informace součástí zdrojového kódu programu. Jejich zahrnutí dovolují podmínky pro jejich šíření a kopírování. Nejčastěji jde o GPL Copyright. Celé znění najdete na přiloženém CD v souboru *gpl_licence.txt*. Z důvodů použití těchto materiálu je i celá práce pod licencí GNU GPL verze 3.

Všechny ikony použité v programu jsou součástí grafické knihovny Visual Studio Image Library. K jejich použití jsem se rozhodl z důvodů lepší integrace do operačního systému MS Windows. Jejich použití není omezeno žádným licenčním ujednáním.