Mendelova univerzita v Brně
Provozně ekonomická fakulta

Logové výstupy systému TEX v aplikaci TEXonWeb

Bakalářská práce

Vedoucí práce:
Ing. Jan Přichystal, Ph.D.

Tomáš Voráč

Brno 2017

Děkuji vedoucímu práce panu Ing. Janu Přichystalovi, Ph.D. za cenné rady a připomínky.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Logové výstupy systému TEX v aplikaci TEXonWeb**
vypracoval samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*
Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.
Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

místo a datum prohlášení .....

Abstract

Voráč, T. Log outputs of system TEXin application TEXonWeb. Bachelor thesis. Brno, 2017

This document deals with the extension of the web application TEXonWeb with the logging environment. The logging environment displays key information from the log file that contains the translation information. During the translation a lot of information is produced, but only a small part of this information is important for a regular user. The structure of the log file and the selection of key information are described in detail in this document. During the creation of the logging environment, emphasis was placed on simplicity. Web technologies (JavaScript, HTML, CSS) were used to implement it.

Abstrakt

Voráč, T. Logové výstupy systému TEX v aplikaci TEXonWeb. Bakalářská práce. Brno, 2017.

Práce se zabývá rozšířením webové aplikace TEXonWeb o logovací prostředí. Do logovacího prostředí se vypisují klíčové informace z logovacího souboru, který obsahuje informace o průběhu překladu. Během překladu vzniká velké množství informací, ovšem jen malá část těchto informací je důležitá pro běžného uživatele. V práci je podrobně popsána struktura logovacího souboru a výběr klíčových informací. Při tvorbě logovacího prostředí byl kladen důraz na jednoduchost. K implementace bylo použito webových technologií (JavaScript, HTML, CSS).

Obsah

1 Úvod a cíl práce **7**
1.1 Úvod . . . 7
1.2 Cíl práce . . . 7

2 Současný stav **9**
2.1 Typografický systém TEX . . . 9
2.2 LATEX . . . 10
2.3 Logovací soubor . . . 10
2.3.1 Obecné informace . . . 10
2.3.2 Rozbor logovacího souboru . . . 11
2.4 TEXonWeb . . . 15
2.4.1 Logovací výstup . . . 16
2.5 Přehled dostupných řešení . . . 16
2.5.1 TEXMaker . . . 16
2.5.2 ShareLaTEX . . . 18
2.5.3 Overleaf . . . 20

3 Metodika a postup řešení **22**
3.1 Metodika . . . 22
3.2 Výběr informací z logovacího souboru . . . 23
3.2.1 Způsob výběru informací . . . 24
3.3 Použité technologie . . . 24

4 Implementace **26**
4.1 Návrh řešení . . . 26
4.2 Tvorba logovacího prostředí a obslužné funkce . . . 26
4.2.1 Minimalizace a maximalizace logovacího prostředí . . . 29
4.3 Výběr dat z logovacího souboru . . . 30
4.4 Nutné změny před nasazením . . . 34

5 Diskuze **35**
5.1 Přínosy navrženého řešení . . . 35
5.2 Porovnání s konkurenčními editory . . . 35
5.3 Nevýhody navrženého řešení . . . 35
5.4 Rozšíření navrženého řešení . . . 36
5.5 Zhodnocení . . . 37

6 Závěr **38**

7 Literatura **39**

Přílohy **41**

A Seznam LaTEXových varování **42**

B Seznam LaTEXových chyb **43**

1 Úvod a cíl práce

1.1 Úvod

Webová aplikace TEXonWeb slouží pro editaci dokumentů v typografickém systému TEX a nástavbě LATEX. Tato aplikace je vyvíjena zejména studenty Mendelovy univerzity v Brně pod vedením Ing. Jana Přichystala, Ph.D. Studenti této univerzity aplikaci používají i v rámci výuky předmětu Zpracování textů na počítači a mnozí studenti ji využívají i k psaní ať už semestrálních, bakalářských nebo jakýchkoliv jiných prací, kde je kladen důraz na formální stránku dokumentu. Tuto aplikaci využívají i akademičtí pracovníci univerzity (Přichystal, 2014).

Systém TEX slouží k sazbě dokumentů, avšak nepracuje jako WYSIWIG[1] editor. TEX pracuje na principu značkovacího jazyka, a proto na první pohled může práce s ním vypadat složitě. Nicméně pokud se uživatel značky naučí, bude moci jednoduše a rychle vytvářet kvalitně vypadající dokumenty. Proto čas investovaný do nastudování problematik se uživateli bohatě vyplatí. Dále má TEX mnoho nástaveb, které značně ulehčují uživatelům práci. Jedou z těchto nástaveb, která je používána zejména pro práce na akademické půdě, je LATEX.

Při tvorbě dokumentů pomocí značkovacího jazyka (tedy i při práci s TEXem) je nutné vytvářet zdrojový text, ve kterém jsou obsaženy značky udávající vzhled dokumentu. Tento zdrojový text se následně překládá a v případě zdárného překladu se vytvoří výstupní souboru, který obsahuje naformátovaný text.

Při překladu zdrojového textu ovšem nevzniká pouze výstupní, ale i logovací soubor, který v sobě nese informace o průběhu překladu nezávisle na tom, zda je průběh zdárný či nikoliv. Během překladu se do logovacího souboru zapisuje velké množství informací. Pro běžného uživatele je složité a zdlouhavé se v tomto rozsáhlém souboru orientovat. Z tohoto důvodu má většina TEXových editorů, jako je například TEXMaker nebo ShareLATEX, v sobě implementované logovací prostředí, ve kterém se snaží uživateli zobrazovat jen pro něj potřebné informace. Tato práce se zabývá možností implementace právě takového logovacího prostředí do webové aplikace TEXonWeb, která momentálně nic takového nenabízí.

1.2 Cíl práce

Jak již bylo naznačeno v úvodu této bakalářské práce, tak cílem je rozšířit webovou aplikace TEXonWeb o prostředí, které bude provádět analýzu logovacích výstupů systému TEX. Důraz bude kladen zejména na jednoduchost a uživatelskou přívětivost. Díky tomuto prostředí nebudou muset uživatelé vyhledávat informace v rozsáhlých logovacích souborech, což přinese přehlednost pro začátečníky, kteří z touto aplikací často pracují. Budou zde srozumitelnou formou prezentovány klíčové informace z logovacího souboru, za něž lze považovat zejména chyby a varování.

[1] WYSIWIG je akronym z věty „What you see is what you get.“ tedy co vidíte je to, co dostanete.

Implementace bude provedena pomocí webových technologií na kterých je TEXonWeb postaven. Jedná se o HTML, CSS a programovací jazyka JavaScript včetně JavaScriptové knihovny jQuery. Pro výměnu dat mezi serverem a klientem bude použita AJAX[2] komunikace.

[2]Asynchronous JavaScript and XML

2 Současný stav

2.1 Typografický systém TEX

Systém TEX je navržen pro profesionální sazbu dokumentů. Z velké části byl navržen americkým informatikem Donaldem Knuthem. Jedná se o DTP[3] program. Výhoda oproti komerčně užívaným systémům je dle Olšáka (2001) jeho otevřenost, díky které si uživatelé mohou vytvářet vlastní nadstavby dle svých individuálních potřeb.

TEX nepatří mezi WYSIWYG editory, ve kterých je vzhled dokumentu v editoru zobrazen v podobě, v jaké by měl být i na výstupu. TEX funguje na principu značkovacího jazyka, kdy v editoru není jen vstupní text, ale i značky, které udávají vzhled dokumentu. Z tohoto důvodu vzniká nutnost text přeložit překladačem, který na základě námi nadefinovaných značek naformátuje text do výsledné podoby (Olšák, 2000).

Značky, které zapisujeme do vstupního souboru nám, jak již bylo výše zmíněno, udávají vzhled dokumentu. Mohou být ve formě primitivních řídících frekvencí (tzv. primitiv) nebo maker. Makra jsou řídící frekvence, jejichž podobu udávají primitiva. Uživatel si může nadefinovat svoje vlastní makra (Olšák, 2001). Je patrné, že pro uživatele je mnohem jednoduší používat makra než primitiva. Proto vznikla celá řada nadstaveb (formátů), které v sobě mají implementovány různá makra. Dle Olšáka (2000) je nejpoužívanější nástavbou LATEX. Mezi další nadstavby patří XeTEX, ConTEX, LuaTEX, PlainTEX. PlainTEX bývá často využíván i jako stavební kámen pro další nadstavby. Je to balík základních maker, který vyvinul samotný zakladatel TEXu.

Celý vstupní text lze vytvořit v jakémkoliv editoru. TEX následně čte tento soubor a vytváří DVI[4] soubor. DVI soubor není čitelný pro člověka, ale slouží pro program nazývaný device driver (ovladač). Tento program vytvoří výstupní soubor na základě značek, které jsou nadefinovány ve vstupním souboru (viz obrázek 1). Text ze vstupního souboru může být zpracován dvěma způsoby:

- Dávkovým, kdy se celý vstupní soubor přečte a na konci se zobrazí uživateli výsledek. TEX se sám snaží v případě některých chyb vytvořit výsledek tak, jak považuje za nejlepší. Proto se tento způsob zpracování vstupního souboru upřednostňuje.
- Interaktivní, kde jak již název napovídá, je uživatel v interakci s programem. Při tomto způsobu může uživatel zastavit běh programu a předat mu nové instrukce.

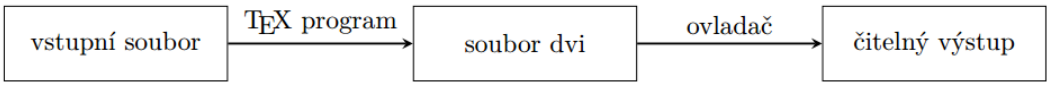
Většina implementací TEXu umožňuje výběr z obou variant (Doob, 1993).

[3] DTP je zkratkou slov „Desktop Publishing“, což je označení pro software sloužící k profesionální sazbě dokumentů.

[4] Značí zkratku „DeVice Independent“, jedná se tedy o soubor, který je nezávislý na zařízení.

Velkou výhodou při tvorbě dokumentu tímto způsobem je stejný vzhled dokumentu jak na monitoru, tak v jakémkoliv jiném výstupu. (Doob, 1993). Výstupní soubor bývá zpravidla ve formátu PDF[5].

Naopak mezi nevýhodu, která je zřejmá z principu tvorby dokumentů pomocí překladače, je nemožnost vidět vzhled dokumentu v době jeho psaní, ale vždy až po proběhlém překladu. Tato nevýhoda je však kompenzována výše popsanou výhodou. (Doob, 1993).



Obrázek 1: Schéma postupu zpracování vstupního souboru na výstupní výstupního souboru. (Doob, 1993)

2.2 LaTeX

Program LaTeX byl vyvinut Leslie Lamportem. LaTeX je velice oblíben zejména v akademické sféře. Mezi jeho největší výhody patří automatické číslování objektů v dokumentu, tvorba křížových odkazů či možnost vkládání grafických prvků (Olšák, 2000).

Tento formát je založen na důsledném dodržování logické struktury a používání souborů maker, kterým říkáme styly. V těchto stylech je upřesněno o jaký druh dokumentu se bude jednat (dopis, kniha, atp.) a dále jsou zde definovány další užitečné příkazy pro uživatele. Tento přístup nám přináší následující výhodu: odborník na danou oblast vytvoří styl, který si uživatelé pouze připojí do své práce a nemusí se tak starat o typografické zásady pro daný dokument (Doob, 1993).

2.3 Logovací soubor

2.3.1 Obecné informace

Logovací soubor má obvykle příponu .log. Jedná se také o běžný textový soubor, proto ho můžeme někdy vidět i s příponou pro textové soubory tedy .txt. Zápis logů do logovacího souboru bývá často spouštěn automaticky.

Obecně lze říct, že log je záznam o činnosti, která proběhla a zápisem více logů nám vzniká logovací soubor. Nový záznam vzniká vždy na prázdném řádku. Nemůže se tedy stát, aby na jednom řádků byly dvě události, ale jedna událost může být

[5]PDF je zkratkou „Portable Document Format“ jedná se tedy o přenosný formát dokumentů nezávislý na softwaru a hardwaru zařízení. Byl vyvinut firmou Adobe.

zapsána přes více řádků. Informace, které se z jednotlivých záznamů získávají, jsou následně využívány k analýze či ke zjištění, zda při proběhlé činnosti nedošlo k chybě. Pokud chyba vznikla, můžeme ve většině případů zjistit, co chybu způsobilo nebo kde chyba vznikla. Informace o tom, kde byla chyba vyvolána (číslo řádku s chybou) nám může velice urychlit opravení dané chyby. Mezi další časté záznamy patří datum, kdy logovací soubor vznikl či různá varování nebo informace o konfiguraci (TeXTerms, 2010). Mezi známé a často využívané logovací soubory (zejména u správců webů) patří například logovací soubor, který je vytvářen webovým serverem a obsahuje informace o činnostech prováděných na dané webové stránce.

2.3.2 Rozbor logovacího souboru

V době překladu zdrojového textu na výsledný tvar vzniká logovací soubor, kde jsou zaznamenány logy, které popisují průběh překladu a vzniku výsledného souboru. Tyto informace v logovacím souboru by se daly rozdělit na: základní informace, chyby, varování a bad box (speciální druh varování). Logovací soubor samozřejmě nabízí daleko větší množství informací než zde vypsané, ale tyto jsou pro uživatele nejpodstatnější.

Mezi *základní informace* patří informace o typu a verzi překladače. Tato informace je obsažena vždy na prvním řádku logovacího souboru v následujícím znění:

```
This is XeTeX, Version 3.1415926-2.5-0.9999.3 (TeX Live 2013)
(format=xelatex 2014.2.10)  17 APR 2017 23:23
```

Můžeme tedy vidět, že výstup byl zpracován za pomocí kompilátoru TEXLive 2013. Dále si můžeme všimnout, že na prvním řádku se dozvíme i přesný čas vytvoření logovacího souboru.

Naopak na úplném konci souboru (poslední řádek) je informace o tom, zda se překlad zdařil a případně kolik stran je na výstupu. Zde je ukázka zdařeného výstupu, kde je výstupem jedna strana, což nám říká informace uvedená v závorce na konci řádku.

```
Output written on tow.process.149246420118682.pdf (1 page).
```

V případě nezdařeného výstupu se na konci logovacího souboru objeví následující informace:

```
No pages of output.
```

V logovacím souboru se nachází i informace o připojených balíčcích. Tyto informace jsou zobrazovány v následující podobě:

```
Package: graphicx 1999/02/16 v1.0f Enhanced LaTeX Graphics (DPC,SPQR)
(/usr/local/texlive/2013/texmf-dist/tex/latex/graphics/keyval.sty
```

Můžeme tedy vidět, že se zde nachází název balíčku a cesta k danému souboru. Jedná se o soubor s koncovkou .sty což nám značí, že se v tomto souboru nachází soubor definující makra, která můžeme následně použít při tvorbě dokumentu.

V logovacím souboru se také nachází velké množství různých hlášek, které pouze podávají informace. Tyto hlášky mají pouze informativní charakter a nijak se dále neřeší. Zde jsou některé případy:

```
LaTeX Info: Redefining \- on input line 33.

LaTeX Font Info:   Try loading font information for EU1+lmr on input line 100.

.....................................................
. LaTeX info: "xparse/define-command"
.
. Defining document command \setsansfont with arg. spec. 'O{}m' on line 51.
.....................................................
```

Jak si můžete všimnout vždy se u těchto hlášek objevuje slovo info a následuje samotná informace.

Dále můžeme samozřejmě v logovacím souboru objevit informace o chybách. *Chyby* se zobrazují ve spodní části logovacího souboru. Jak si můžeme všimnout v následující ukázce. Chyby jsou vyvolány v době překladu z důvodu, že nastal závažný problém. Tato skutečnost vede ve většině případů k nezdařenému překladu. Z toho plyne, že nevznikne žádný výstup a v logovacím souboru se zobrazí výše zmiňovaná informace (No pages of output.). Formát chyby je vždy stejný. Na začátku řádku v logovacím souboru se nachází vykřičník, za vykřičníkem se nachází typ chyby a na následujících řádcích popis chyby, kde je ve většině případů uvedeno i číslo řádku, kde byla chyba zaznamenána (Wikibooks, 2015).

Chyby mohou vzniknou z různých příčin. Častou příčinou může být pouze překlep, kdy zapíšeme špatný příkaz, který neexistuje. V takovém případě se nám v logovacím souboru zobrazí následující log:

```
! Undefined control sequence.
l.6 \Tex

The control sequence at the end of the top line
of your error message was never \def'ed. If you have
misspelled it (e.g., `\hobx'), type `I' and the correct
spelling (e.g., `I\hbox'). Otherwise just continue,
and I'll forget about whatever was undefined.
```

Zde si můžeme všimnout, že na druhém řádku pod názvem chyby, je informace na kterém řádku byla chyba zjištěna (číslo za znaky l.) a také informace o příkazu, který chybu zapříčinil (Tex). Po prázdném řádku následuje informace k danému

typu chyby. Tato informace je vždy spojena s typem dané chyby bez ohledu na to, co přesně chybu způsobilo.

Pokud nám nastane chyba typu „Emergency stop“, tak se v logovacím souboru nemusí nacházet informace o tom, na kterém řádku chyba nastala. Číslo řádku se zde nemusí nacházet, protože LaTeX touto chybou označuje, že se dostal do potíží, ale chyba mohla vzniknout již dříve. Důvodem vzniku této chyby může být chybějící značka `\end` (Dommelen, 2017b). Ovšem tato situace by vyvolala i tuto chybu „! Undefined control sequence“, která by již obsahovala i informaci o čísle řádku. Chybu si můžete prohlédnout v následující ukázce:

```
! Emergency stop.
<*> tow.process.14934195101738.tex

*** (job aborted, no legal \end found)


Here is how much of TeX's~memory you used:
 13546 strings out of 493918
 280430 string characters out of 6148599
 318722 words of memory out of 5000000
 16771 multiletter control sequences out of 15000+600000
 3680 words of font info for 19 fonts, out of 8000000 for 9000
 1144 hyphenation exceptions out of 8191
 52i,0n,37p,10471b~stack positions out of 5000i,500n,10000p,200000b
No pages of output.
```

Na výše uvedené ukázce si můžeme všimnout, že výčet informací o chybách se nachází ve spodní části logovacího souboru. Pod poslední nalezenou chybou se již zobrazují pouze informace o množství využité paměti a poslední řádek, který informuje o množství stránek na výstupu (viz text výše věnovaný základním informacím logovacího souboru).

Existují ale i vyjímky, kdy se chyba nachází v prostřední části logovacího souboru. Jedná se například o následující chybu.

```
! LaTeX Error: \RequirePackage or \LoadClass in Options Section.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
 ...

l.13 \begin{document}

The package `' is defective.
It attempts to load `graphics' in the options section, i.e.,
between \DeclareOption and \ProcessOptions.
```

Tato chyba se zobrazuje v případě, kdy nemůže být současně použit daný balíček s vytvořeným nastavením (Dommelen, 2017a).

V případě, kdy je nalezeno *varování*, tak překlad nikdy nekončí nezdarem, ale úspěšně proběhne a vygeneruje výstup (Rolfe, 1999). Což je hlavní rozdíl mezi chybou a varováním. Při nalezeném varování je zapotřebí si projít dokument, zda je jeho podoba přijatelná, protože překladač text upraví, aby byl z jeho pohledu co nejlepší výstup. Tyto úpravy ovšem nemusí být vždy přijatelné.

První typ varování je ve formát LaTex Warning: a následuje popis varování, viz následující příklad:

```
LaTeX Warning: Citation `pokus' on page 1 undefined on input line 8.
```

Toto varování je zobrazeno v případě, kdy se odkazujeme na nedefinovanou citaci. Překladač se s tímto varováním vypořádá tak, že umístí na dané místo dva otazníky, čili například toto varování je jedno z těch, které je nutné opravit. Na tomto příkladě si můžeme všimnout, že i u varování se objevuje informace o čísle řádku, který toto varování způsobil. Na rozdíl od chyby u varování není informace o novém řádku ve formátu l. číslo řádku, ale je zde přímo napsáno „... undefined on input line 8“.

Druhý typ varování je ve formátu LaTex Font Warning: a opět následuje popis varování. Tento typ varování upozorňuje například na problémy s fontem písma, který nebyl nalezen, a proto byl nahrazen jiným (Dommelen, 2017a).

```
LaTeX Font Warning: Font shape `OT1/cmss/m/n' in size <4> not available
(Font)              size <5> substituted on input line 7.
```

Zde stojí za povšimnutí, že následující řádek začíná slovem Font v závorce a za tímto slovem následuje bližší popis.

Bad box je speciální typ varování. Platí pro něj tedy stejná charakteristika. Bad boxy můžeme rozdělit na overfull a underfull varování. V případě overfull nastal problém s příliš dlouhým (přetečeným) řádkem. Z toho vyplývá, že daný řádek vyčnívá z textu. Naopak underfull nastává v případě, kdy LaTeX musí natáhnout řádek a mezery mezi slovy či odstavci jsou větší, než jak je stanoveno. Často se tyto varování vůbec neřeší, zejména jedná-li se o underfull a o malé vychýlení (Dommelen, 2017a). Oba dva typy můžeme dále rozdělit na vbox a hbox. Kdy hbox značí, že je text příliš široký a vbox označuje příliš vysoký text.

V logovacím souboru je bad box zapsán v pořadí typ bad boxu (Overfull/Underfull) a následuje informace o tom zda, se jedná o hbox nebo vbox. Poté je popsána bližší informace k danému varování, jak si můžete všimnout v následující ukázce:

```
Overfull \hbox (119.57pt too wide) in paragraph at lines 7--8
[]\EU1/lmr/m/n/10 Strasnedlouhyradekbezjakeklivvytvorenemezeryzpusob
ujebadboxjaksimuzetevsimnoutpravevtetoukazce
 []

! Undefined control sequence.
l.9 \en
```

```
      {documt}
The control sequence at the end of the top line
of your error message was never \def'ed. If you have
misspelled it (e.g., `\hobx'), type `I' and the correct
spelling (e.g., `I\hbox'). Otherwise just continue,
and I'll forget about whatever was undefined.
```

V tomto příkladě se tedy jedná o Overfull a hbox. Jde tedy právě o příliš dlouhý řádek, který přesahuje přes stanovený okraj. Číslo v závorce (1.45999pt) značí, o kolik bodů bylo přesaženo přes okraj a následuje informace o tom, kde toto chování nastalo. Na této ukázce bych chtěl také ilustrovat skutečnost, že varování se nachází v logovacím souboru podobně jako chyby ve spodní části. Chyby a varování se zde vzájemně prolínají. Podle toho co bylo vyvoláno dříve.

V logovacím souboru můžeme objevit i následující varování:

```
*********************************************
* LaTeX warning: "xparse/redefine-command"
*
* Redefining document command \oldstylenums with arg. spec. 'm' on line 128.
*********************************************
```

Toto varování se nenachází v části logovacího souboru, které je popsané výše, ale nachází se spíše v první polovině logovacího souboru. Můžeme si všimnout, že tento typ varování začíná *, což je rozdíl oproti běžnému varování. Také se zde ve slově warning nachází malé w. Hlavní rozdíl oproti běžným varováním je, že nevzniká na základě překladu daného textu (nemá přímou spojitost s daným textem), ale souvisí s předdefinovanými makry. Jedná se tedy o neškodné varování.

2.4 TEXonWeb

TEXonWeb je webová aplikace dostupná na webové stránce https://tex.mendelu.cz/. Slouží pro sazbu dokumentů v systému TEX a nadstavby LATEX. Aplikace je využívána zejména studenty a právě kvůli studentům vznikla tato aplikace. Studenti měli často problém s instalací a konfigurací systému TEX na svém zařízení. Problémy nastávaly také při instalaci typografického systému TEX v učebnách univerzity (Přichystal, 2014).

Mezi hlavní výhody aplikace patří možnost tvorby dokumentů odkudkoli, kde je připojení k internetu bez nutnosti instalace jakéhokoliv softwaru (Přichystal, 2016). Jako další výhodu vidím možnost používání této aplikace i bez nutnosti registrace. Neregistrovaný uživatel ovšem nemůže ukládat svou práci. Výhody přináší zejména začínajícím uživatelům, kdy jim nabízí spoustu vestavěných funkcí, které jim napomáhají při tvorbě dokumentů. Mezi tyto funkce patří například: možnost automatického vložení nezlomitelných mezer před jednopísmenné spojky a předložky, průvodce pro tvorbu tabulek či seznamů a v neposlední řadě kontrola pravopisu.

Zejména průvodce pro tvorbu tabulek je velice kladně hodnocen, protože tvorba tabulek nepatří mezi triviální úkony (Přichystal, 2014).

Dále má tato aplikace přednastaveny různé šablony, které mohou uživatelé využívat a napomáhají tak ke kvalitnější a rychlejší tvorbě. Jedná se o šablony pro Závěrečné práce, Vizitky, Žádosti, Životopisy, Dopisy, Kalendáře, Darovací smlouvy, atd.

Aplikace používá nejaktuálnější verzi kompilátoru TeXLive 2013 a implicitně je nastaveno kódování UTF-8 (Přichystal, 2014). Na serverové straně je použit jazyk Perl, který v době vzniku této aplikace patřil mezi populární skriptovací jazyky. Pomocí tohoto jazyku je například napsán skript, který, překládá zdrojový kód na výstup. Stará se také o vytváření logovacího souboru. Tento skript je spouště po kliknutí na tlačítko „PDF“. Na straně klienta je použit jazyk JavaScript, zejména pak jeho knihovna jQuery. Pomocí tohoto jazyka je naprogramováno uživatelské prostředí aplikace. Součástí knihovny jQuery jsou funkce pro AJAX komunikaci, která je v aplikaci také využívána. Bylo jí využito například při tvorbě průvodce pro tvorbu tabulek. Aplikace je provozována na Apache serveru. Dále aplikace využívá systém třetí strany Ace editor[6], což je open-sourcový[7] software, který je napsán v programovacím jazyce JavaSrcipt. Tento editor umožňuje automatické doplňování, nahrazování či vyhledávání slov.

2.4.1 Logovací výstup

V současné době aplikace dovoluje uživateli zobrazit pouze logovací soubor, který se vytvoří a uloží na serveru. Obsah tohoto celého souboru je vypsán do okna prohlížeče bez jakékoliv úpravy. Tento způsob je zejména pro začínající uživatele velice nepřehledný a také prodlužuje práci, kdy uživatel musí informace hledat v rozsáhlém souboru a ztrácí tím čas, který by mohl věnoval primární činnosti, kvůli které aplikaci využívá, tedy vytvoření kvalitního dokumentu.

2.5 Přehled dostupných řešení

Logovací prostředí patří do standardní výbavy webových i desktopových editorů, které slouží k tvorbě dokumentů prostřednictvím systému TEX. Při tvorbě této funkcionality do aplikace TEXonWeb se budu inspirovat populárními editory. Mezi něž patří TEXMaker, ShareLaTEX a Overleaf. Pro porovnání, zde nemám pouze webové aplikace, ale i desktopovou aplikaci.

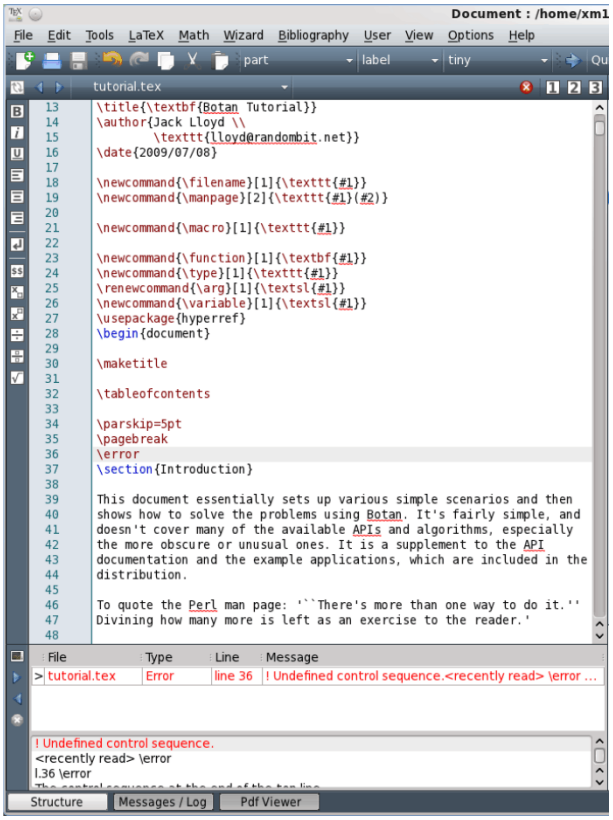
2.5.1 TEXMaker

TEXMaker je open source editor sloužící k tvorbě dokumentů pomocí jazyka TEX. Jedná se o desktopovou aplikaci. Je napsán v programovacím jazyku C++ a jeho

[6]Dostupný z http://ace.c9.io/

[7]Jedná se o software s otevřeným zdrojovým kódem, který se může využívat a upravovat.

autorem je Pascal Brachet. První verze vyšla v roce 2003. Nejnovější verze, která je vydávána pod číslem 4.5, vznikla v roce 2015 (TeXMaker, 2017a). Tato aplikace patří mezi tzv. multi-platformní software, který je dostupný na operačních systémech Linux, Windows a MacOS. Dle Šafaříka (2010) je pro svou jednoduchost doporučován zejména začínajícím uživatelům. Naopak nenabízí žádné speciální funkce pro pokročilé uživatele. Jako nevýhodu, zejména oproti aplikaci TEXonWebu, spatřuji ve složitosti instalace a konfigurace, zejména pro operační systém Windows, který je ovšem nejvíce rozšířen. Před instalací samotného TEXMakeru je nutné nainstalovat MikTEX[8]. Tato nevýhoda vzniká u všech desktopových editorů.

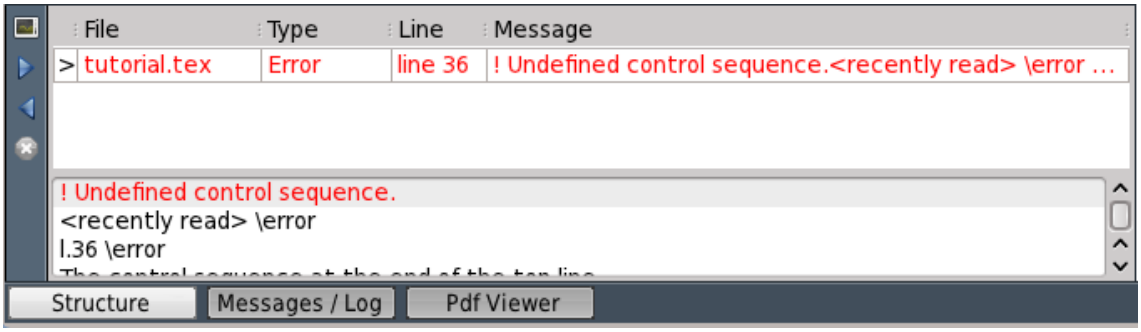


Obrázek 2: Část grafického rozhraní aplikace TEXMaker
Pozn.: Obrázek byl upraven. (TeXMaker, 2017b)

TEXMaker nabízí logovací prostředí, které se nachází ve spodní části editoru (viz obrázek 2). Toto prostředí si uživatel může vertikálně zvětšovat a zmenšovat podle potřeby nebo provést minimalizaci okna pomocí tlačítka Messages/Log, které se nachází v patičce editoru (viz obrázek 3). Informace se zde vypisují do tabulky a jsou barevně rozlišeny pro lepší přehlednost. Červeným písmem jsou zobrazovány chyby a modře ostatní informace. Na každý řádek je vypisována jedna informace

[8]Jedná se o kompilátor TEXu.

o dané činnosti. Celá akce pro překlad souboru se spouští tlačítkem ve tvaru šipky v horní části editoru.



Obrázek 3: Logovací prostředí v aplikace TEXMaker zobrazující jednu chybu. Pozn.: Obrázek byl upraven. (TeXMaker, 2017b)

Každý řádek tabulky se skládá z následujících sloupečků: název souboru, typ informace (chyba, varování), číslo řádku, kde nastala chyba a popis. V případě, že uživateli zobrazený popis nestačí, je možné kliknout na daný řádek tabulky a pod tabulkou se mu zobrazí veškeré informace, které jsou zapsány v logovacím souboru. Přičemž se zobrazí ta část logovacího souboru, která souvisí s označeným řádkem (viz obrázek 3). V levé části okna se nachází dvě modré šipky sloužící k překliknutí označeného řádku na řádek následující. Této funkcionality lze využít pouze pro procházení chyb.

2.5.2 ShareLaTEX

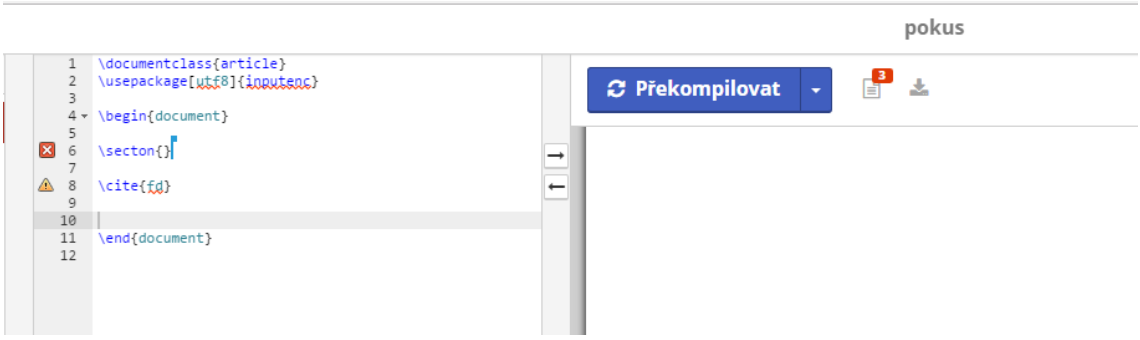
Jedná se o LaTEXový online editor. Pro práci s tímto editorem je nutné se registrovat. Při registraci je možné si vybrat ze tří typů tarifů „Personal“, „Collaborate“, „Proffesional“. Liší se zejména v ceně a množství lidí, kteří mohou spolupracovat na jednom projektu viz následující tabulka 1 (ShareLaTeX, 2017c).

Tarify webové aplikace ShareLaTEX

Název	Množství spolupracovníků	Cena v eurech/měsíc
Personal	1	zdarma
Collaborate	8	11
Professional	neomezeno	28

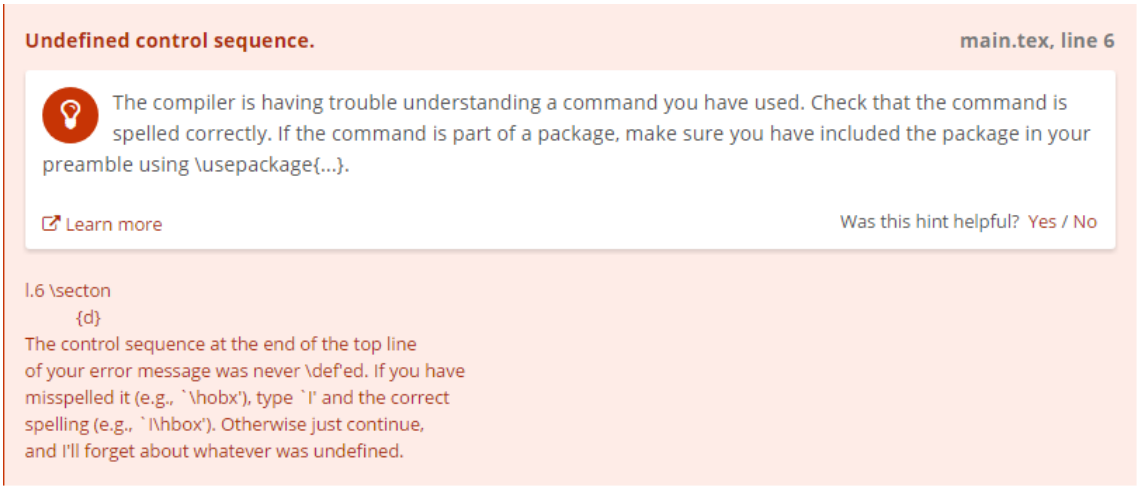
Tabulka 1: Pozn.: Jakýkoliv tarif si mohou uživatelé vyzkoušet na 7 dní zdarma. (ShareLaTeX, 2017c)

Díky své funkcionalitě, kdy nabízí pokročilé funkce pro správu dokumentů mezi více uživateli, které jsou ale mnohdy zpoplatněna se ShareLaTEX hodí zejména pro vícečlenné týmy pracující na jednom dokumentu (ShareLaTeX, 2017c).



Obrázek 4: Část prostředí aplikace ShareLaTEX při zapnutém PDF prohlížeči.

Logovací prostředí se nachází na pravé straně editoru a je možné jej zvětšovat/zmenšovat. Uživatel si musí vybrat, zda chce vidět výsledný či logovací výstup. Mezi těmito možnostmi může uživatel vybírat pomocí tlačítka v pravé horní části editoru, které vypadá jako list papíru. U tohoto tlačítka se po překladu objevuje čtvereček s číslem (obrázek 4). Číslo indikuje součet počtů chyb a varování, které byly během překladu nalezeny. Barva čtverečku nám může napovědět, zda se jedná o chybu či varování. Pokud byla nalezena jedna či více chyb, tak má čtvereček červenou barvu. V případě, kdy se vyskytly pouze varování má čtvereček oranžovou barvu. U případů, kdy nenastal žádný problém, není u tlačítka zobrazen žádná čtvereček. Na obrázku 4 si lze všimnou, že po překladu, ve kterém byla zjištěna chyba, se zobrazí křížek vedle čísla řádku, jež chybu vyvolal a v případě varování se vedle čísla řádku zobrazí vykřičník.



Obrázek 5: Ukázka popis chyby nacházející se v logovacím prostředí aplikace ShareLaTEX

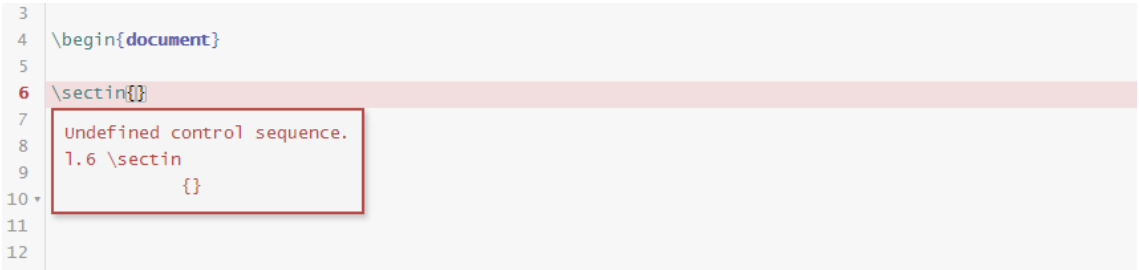
Když si zobrazíme samotný logovací výstup, tak uvidíme podrobný popis chyb a varování(obrázek 5), které jsou i zde pro přehlednost barevně rozlišeny. V případě,

že na danou chybu či varování klikneme levým tlačítkem myši, tak se nám v editoru jako aktivní řádek zobrazí ten, který chybu/varování způsobil. Dále se v tomto prostředí nachází tlačítko „Zobrazit logy“, které slouží k zobrazení celého logovacího souboru bez jakékoliv úpravy.

2.5.3 Overleaf

Jedná se o online LaTEXeditor. Je zde možné pracovat i bez registrace, ale nejsou pak k dispozici vešekeré funkce. Při registraci si uživatel může vybrat ze tří verzí. V bezplatné verzi je velikost úložiště omezena na 1 GB. Kromě této bezplatné verze nabízí editor dvě placené verze, které nabízí prémiové služby jako je například možnost rychlého ukládání do Dropboxu, sledování celé historie dokumentu a větší úložiště. Konkrétně 10GB u verze „Pro“[9] a 20GB u verze „Pro+Teach“[10]. Kromě velikosti kapacity se verze „Pro+Teach“ liší od „Pro“ ve funkci „Teaching Toolkit“. Tato funkce slouží pro učitele, kteří mohou jendoduše zadávat úkoly studentům. Ti nasledně tyto úkoly vypracují a předají je zpět učiteli ke kontrole (Overleaf, 2017b).

Na rozdíl od předchozích editorů zde není vytvořeno běžné logovací prostředí. Také se v tomto editoru neobjevuje po překladu seznam všech chyb či varování. Upozorňování na chyby nebo varování, zde přichází již v průběhu psaní, kdy se neustále kontroluje, zda na některém řádku nenastala chyba či varování. Pokud v době, než se stihne napsaný text zkontrolovat, je udělána více než jedna chyba, tak editor bude vždy zobrazovat jen tu první nalezenou. Protože jakmile dojde k jediné chybě, je překlad považován za nezdařený, a proto se již v překladu dále nepokračuje. U varování, je princip jiný a to z toho důvodu, že když nastane varování neznačí to nezdařený překlad. Z této skutečnosti vyplývá, že editor nás upozorní na všechny varováni, nikoliv jen na první nalezené, jako je tomu u chyb.



Obrázek 6: Ukázka zobrazení chyby v editoru Overleaf

V případě, kdy byla nalezena chyba, tak se nám daný řádek zvýrazní červeným pozadím, jak je vidět na obrázku 6. U varování jsou řádky zvýrazněny žlutou barvou viz obrázek 7. Při kliknutí na daný řádek se nám zobrazí důvod vzniku chyby/varování.

[9] Cena je 10,5 euro/měsíc.
[10] Cena je 17,5 euro/měsíc.

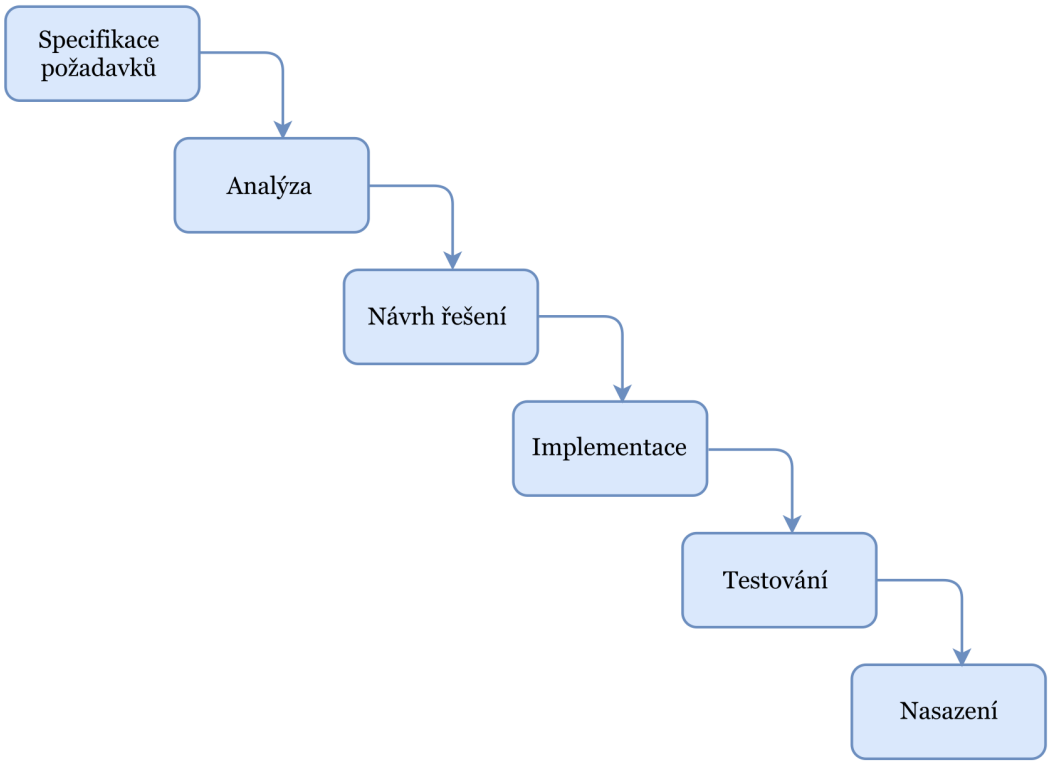
Obrázek 7: Ukázka zobrazení varování v editoru Overleaf

Samozřejmostí je i možnost nahlédnutí do neupraveného logovacího souboru. Tento náhled je zde řešen formou downloadu celého neupraveného logovacího souboru.

3 Metodika a postup řešení

3.1 Metodika

Jako metodiku práce jsem si vybral Vodopádový model. Tento model rozděluje projekt na fáze, podle kterých se lineárně postupuje. Tedy až po splnění jedné fáze se přistupuje k další. Mohou nastat výjimky, kdy se daná fáze nedokončí a postupuje se k následující. Obvykle se tak stává mezi fázemi „implementace“ a „testování“, kdy se testují naimplementované části systému, přičemž nemusí být udělána celá implementace.



Obrázek 8: Vodopádový model

Fáze vodopádového modelu:

1. *Specifikace požadavků:* Požadavky na logovací prostředí byly uvedeny v rámci zadání bakalářské práce. Zde jsem si pouze s vedoucím práce panem Ing. Janem Přichystalem, Ph.D upřesnil detaily.
2. *Analýza:* V této fázi jsem provedl rozbor logovacího souboru, abych zjistil informace, které se v souboru vyskytují a také jejich strukturu. Dále jsem provedl průzkum již hotových řešení v konkurenčních editorech.

3. *Návrh řešení:* Zde jsem si určil postupy a pořadí kroků implementace. Do této fáze by se dala zahrnout i volba programovacího jazyka, zde jsem měl ovšem zjednodušenou situaci, protože jsem logicky využil technologie na kterých je TEXonWeb postaven.
4. *Implementace:* V této fázi jsem provedl samotnou implementaci navrženého řešení. Nejdříve jsem naimplementoval grafickou část logovacího prostředí, kterou jsem následně otestoval (zde jsem udělal výjimku a naimplementovanou grafickou část jsem otestoval, i když nebyla dokončena celá implementace). V další části implementace jsem vytvořil skript, který vybírá informace z logovacího souboru.
5. *Testování:* V této fázi zkoumám použitelnost vytvořené implementace a případně zapracovávám vylepšení.
6. *Nasazení:* Tato fáze přijde na řadu po dokončení testování. Potřebné úpravy před nasazením jsou uvedeny v další části práce (kapitola „Nutné změny před nasazením“).

Tento model jsem si vybral, protože je ideální pro případy, kdy známe od začátku požadavky na projekt. Díky fázím, které na sebe logicky navazují, se v projektu postupuje koordinovaně, což vede k rychlejší a efektivnější práci. Výhoda tohoto modelu tedy spočívá v možnosti rozplánovat si harmonogram práce na základě fází.

3.2 Výběr informací z logovacího souboru

Na základě prozkoumání logovacích prostředí různých TEXových editorů (webových i desktopových) a rozboru logovacího souboru jsem se rozhodl, které informace budou vypisovány do logovacího prostředí aplikace TEXonWeb.

Budou zde vypisovány veškeré varování a chyby včetně bližšího popisu. Zejména chyby jsou klíčovým prvkem pro vznik logovacího prostředí. Tyto informace o chybách musí být v logovacím prostředí vypisovány. Jedná se o ty nejdůležitější informace, které uživatel hledá. Varování jsem zde zařadil z toho důvodu, že někdy mohou upozornit na změnu v souboru, která není přípustná z hlediska koncového vzhledu dokumentu. Samozřejmostí je vypisování čísel řádků, kde chyby či varování vznikly (pokud je tato informace v logovacím souboru obsažena). Chyby a varování umožňují zobrazovat veškeré, v předchozí kapitole zkoumané, TEXové editory.

Dále jsem se rozhodl, že je vhodné uživateli zajistit možnost zobrazení informace o verzi překladače a datu vytvoření logovacího souboru. Tuto informaci jsem zde zařadil i z toho důvodu, kdyby došlo k chybě na serveru, při které by se nevytvářel nový logovací soubor. Uživatel si tak může zkontrolovat, zda se mu zobrazuje výstup z aktuálního logovacího souboru. V neposlední řadě jsem se rozhodl pro výpis posledního řádku logovacího souboru, který informuje o množství stránek na výstupu. Myslím si, že tato informace je pro uživatele zajímavá.

3.2.1 Způsob výběru informací

Výše popsané informace o chybách a varováních budu získávat z logovacího souboru za pomocí regulárních výrazů. Informace o verzi překladače, datu a počtu stránek na výstup bude snadné získat, protože se nachází vždy na prvním a posledním řádku logovacího soubor.

U varování nejdříve naleznu řádek, který začíná jedním z následujících slov či slovních spojení (Overfull, Underfull, LaTeX Warning, LaTeX Font Warning) viz podkapitola Rozbor logovacího souboru. Dále, abych získal i následující podstatné informace, nejen první řádek, tak budu vybírat všechny řádky následující po prvním řádku až po první prázdný řádek.

Co se týče výběru chyb, tak v první řádě naleznu řádek začínající znakem vykřičník (!). Dále ovšem nemohu pokračovat stejně jako v případě varování, protože u chyby je potřeba zaručit získání všech informací, které jsou v logovacím souboru k dané chybě obsaženy. A existují i takové zápisy logu chyby, které obsahují informaci o čísle řádku, kde chyba vznikla, až na konci pasáže, která souvisí s danou chybou. Bohužel mezi tímto řádkem (informujícím o chybě) a prvním řádkem chyby existuje několik prázdných řádků. Z tohoto důvodu jsem se rozhodl, že budu vypisovat informace k dané chybě, až do doby, než bude následující řádek začínat opět znakem vykřičník (indikace nové chyby) nebo nastane-li řádek, který indikuje nějaký typ varování a nebo do doby než následující řádek bude začínat slovy „Here is how much of TeX's memory you used:“, protože většina chyb se řadí nakonec logovacího souboru právě před pasáž informující o množství využité paměti viz podkapitola Rozbor logovacího souboru.

Ovšem existují výjimky, kdy se chyba zobrazí v jiné části logovacího souboru, proto jsem se rozhodl udělat následující omezení. Cyklus pro zjištění detailních informací o chybě ukončím i na základě nalezení řádku, který bude informovat o připojeném balíčku (Package) nebo o nějaké LaTeXové informaci (LaTeX Info, LaTeX Font Info) podrobněji v podkapitole Rozbor logovacího souboru. Dále bude omezeno množství řádků, které budou informovat o chybě, na 20 (dostatečně velký počet řádků na zobrazení informací o chybě). Toto omezení jsem zařadil z důvodu, kdyby se za chybou objevila nějaká jiná nespecifikovaná hláška, tak aby se zbytečně nevypisoval celý její obsah. Díky tomuto postupu je zaručeno, že se uživatel vždy dozví veškeré informace k dané chybě bez ohledu na to, kde se chyba nachází. Výjimečně může nastat situace, kdy budou zobrazeny i některé řádky navíc, které s chybou nesouvisí.

3.3 Použité technologie

Jelikož přidávám funkcionalitu do již existující aplikace TeXonWeb, tak by bylo nesmyslné používat jiné technologie než na kterých je tato aplikace postavena.

Tato aplikace je postavena zejména na programovacím jazyku JavaScript, pomocí něhož je vytvořeno celé GUI[11] aplikace. Jedná se o programovací jazyk, který vytváří funkcionalitu na straně klienta. U této aplikace není ovšem využíváno čistého JavaScriptu, ale knihovny jQuery, která usnadňuje a výrazně zkracuje programování oproti čistému JavaScriptu. Mezi výhodu této knihovny lze zařadit možnost manipulace s obsahem webu za pomocí tzv. CSS selektorů (Čápka, 2017) a také tato knihovna nabízí řadu funkcí pro podporu AJAX technologie, která je v aplikaci často využívána. Tato technologie slouží ke získávání a posílání dat mezi klientem a serverem bez nutnosti opakovaného načítání webové stránky. Tuto technologii budu využívat i já k získání dat z logovacího souboru, který je uložen na serveru.

Webové aplikace TEXonWeb používá open-sourcový produkt Ace editor. Tento editor slouží zejména pro weby obsahující editory, které slouží k programování. Má v sobě implementované mnohé služby, například zvýrazňování a našeptávání syntaxe pro více než 110 jazyků a další jazyky se dají libovolně a jednoduše do tohoto editoru přidat. Mimo jiné tento Ace nabízí i možnost značení vybraného řádku ikonami, které indikují různé situace, například křížek (indikace chyby), vykřičník (indikace varování) nebo písmeno i (indikace pro informaci) (Ace, 2017). Ace editor používá výše zmiňovaný online editor ShareLaTEX, který využívá i veškeré zde popsané služby a funkce.

Nutností pro tvorbu webových aplikací je HTML a kaskádové styly (CSS), proto samozřejmě i tyto webové technologie byly využívány při tvorbě TEXonWebu. Aplikace je dále postavena na jazyku Perl. Tento jazyk se používá na straně serveru. V současnosti se pro tvorbu webových aplikací již příliš nepoužívá.

V dnešní době je trend vytvářet tzv. client side aplikace, které používají pouze programovací jazyky pracující na straně klienta. Proto i já budu ve své práci používat zejména jazyk JavaScript.

[11] GUI je akronym slov „Graphical User Interface“, tedy grafické uživatelské prostředí.

4 Implementace

4.1 Návrh řešení

Fázi implementace jsem začal vytvořením grafické podoby logovacího prostředí. Využil jsem k tomu technologie HTML a CSS. Následně jsem pomocí jQuery přidal funkcionalitu starající se o minimalizaci a maximalizaci prostředí.

Následně jsem se začal zabývat samotným skriptem pro výběr klíčových informací z logovacího souboru právě do připraveného logovacího prostředí. Tento skript jsem psal v jazyce JavaScript za pomocí knihovny jQuery a regulárních výrazů. Pro stažení souboru ze serveru na klienta používám AJAX komunikaci.

Na závěr jsem se rozhodl, že by bylo vhodné zobrazovat informace nejen v logovacím prostředí, ale také přímo v editoru. Konkrétně jsem se rozhodl pro zobrazení ikony vykřičníku při nalezeném varování a křížku při nalezené chybě. Tyto ikony se budou objevovat vedle čísla řádku, který s touto chybou souvisí. V případě, kdy uživatel najede kurzorem myši na tuto ikonku, tak se zobrazí podrobný popis k dané chybě či varování. Tato funkcionalita bude fungovat jen v případě, kdy bude v logovacím souboru informace o čísle řádku dané chyby či varování. Tuto funkcionalitu naipmlementuji do webové aplikace pomocí funkce, kterou nabízí Ace editor. Tato funkce nabízí možnost k vybranému řádku přidat ikonu a také při najetí myší na tuto ikonu dokáže zobrazit požadovaný text.

4.2 Tvorba logovacího prostředí a obslužné funkce

Do již existující struktury aplikace TEXonWeb jsem přidal HTML znaky vytvářející základní část logovacího prostředí. Rozhodl jsem se, po vzoru aplikace TEXMaker ,umístit logovací prostředí do spodní části aplikace (viz obrázek 9). Veškeré informace se budou vypisovat do tabulky, která má tři sloupce. První sloupec označuje typ logy (chyba, varování, informace), ve druhém sloupci je obsah samotné chyby a v posledním sloupci se nachází informace o čísle řádku, který chybu způsobil.

Jak si můžete všimnout na obrázku 9, pro lepší přehlednost při vyhledávání informací jsem do logovacího prostředí implementoval záložky. Mezi záložkami lze přepínat a vybírat si, zda chceme zobrazit výpis pouze chyb, varování či ostatních informací nebo výpis všech těchto věcí dohromady. Poslední záložka "výpis logů" slouží pro zobrazení výpisu všech informací (bez úpravy), které jsou zapsány v logovacím souboru. Pro lepší přehlednost jsem se rozhodl i pro různou barvu písma při vypisování těchto informací. Pro chyby jsem zvolil barvu červenou, pro varování oranžovou a pro informace barvu modrou.

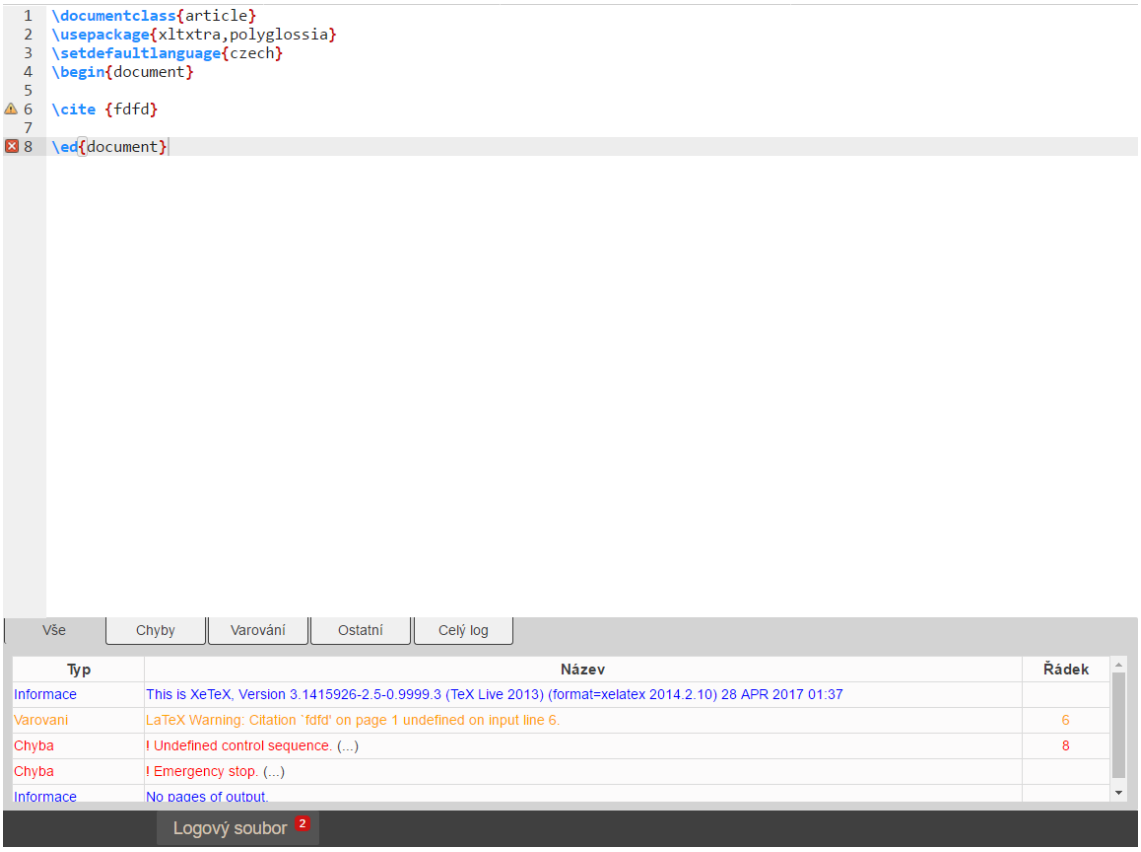
Pří kliknutí na řádek tabulky se uživateli jako aktivní řádek v editoru objeví ten, který chybu či varování vyvolal (funguje jen u výpisu chyb či varování, které mají číslo ve sloupci řádek). Pro tuto funkcionalitu jsem využil funkci Ace editoru (Editor.gotoLine()). Jako parametr této funkce stačí předat číslo a tato funkce se již postará o to, aby se jako aktivní řádek editoru zobrazil právě řádek s tímto číslem.

Samozřejmostí je i případné posunutí se v editoru, pokud chce uživatel přejít na řádek, který se nenachází v aktuální části obrazovky. Jedinou nevýhodou je, že čísla řádku se v této funkci počítají od 0. Čili pokud bychom chtěli, aby se jako aktivní řádek označil například řádek číslo 2, musí se do funkce jako parametr zadat číslo 3.

```
1 editor.gotoLine(value,0, true);
```

Ukázka kódu 1: Funkce sloužící pro změnu aktivního řádku editoru

Můžete si všimnout, že tato funkce má více než jeden parametr. První parametr je výše zmiňované číslo nového aktivního řádku. Druhý parametr je číslo, které označuje před kolikátým znakem na daném řádku se objeví kurzor (pokud je toto číslo větší než kolik má řádek znaků, bude kurzor za posledním znakem). Poslední parametr značí, zda chceme animaci.



Obrázek 9: Část grafického rozhraní aplikace TEXonWeb při maximalizovaném logovacím prostředí.

Pokud na řádek tabulky kliknete dvakrát, tak se zobrazí celý popis k dané chybě včetně bližšího popisu (viz obrázek 10). U řádků, kde je tato možnost (k informaci se nachází i bližší popis) jsou uvedeny následující znkay "(...)". Podrobný popis se zobrazí tak, že se rozšíří řádek tabulky. V případě opětovného dvojitého kliknutí se

Obrázek 10: Logovací prostředí aplikace při zobrazeném detailním popisu chyby.

bližší popis skryje a řádek tabulky se zúží na původní velikost. K tomuto efektu jsem použil funkce knihovny jQuery toggle(), která slouží ke skrývání a odkrývání prvků. Obsah bližší informace jsem musel obalit do tagu <pre>, který slouží k zachování bílých znaků (mezera, odsazení, odřádkování), aby byla informace lépe čitelná.

Jelikož je na stejný HTML element navázána funkce, jak na jedno kliknutí, tak na dvojklik, je zapotřebí zamezit spouštění funkce, která slouží pro jedno kliknutí v případě dvojkliku. Tento problém jsem vyřešil pomocí následující funkce:

```
1  var doba_cekani = 400;
2  var pocet_kliku = 0;
3  var casovac = null;
4
5  $(function(){
6      $(".table_log").on("click","tr", function(e){
7          pocet_kliku++;
8          if(pocet_kliku === 1) {
9       var value = $(this).attr('value');
10             casovac = setTimeout(function() {
11          if(!(typeof(value) === 'undefined')){
12    editor.resize(true);
13    editor.gotoLine(value,0, true);
14      }
15                 pocet_kliku = 0;
16              }, doba_cekani);
17          } else {
18              clearTimeout(casovac);
19               $(".podrobnosti",this).toggle();
20              pocet_kliku = 0;
21          }
22      })
23      .on("dblclick", function(e){
24          e.preventDefault();
25      });
26 });
```

Ukázka kódu 2: Funkce pro obsluhu jednokliku i dvojkliku

Tato funkce je založena na tom, že počítá počet kliknutí. V případě jednoho kliknutí se spustí funkce , která způsobí změnu aktivního řádku. V případě více kliknutí se spustí funkce, která zobrazuje podrobné informace k danému logu. Za dvojklik se považují dva kliky, které byly provedeny v rozmezí 400 milisekund.

4.2.1 Minimalizace a maximalizace logovacího prostředí

Při implementaci logovacího prostředí docházelo k problému, kdy po provedení minimalizace nebo maximalizace logovacího prostředí nedocházelo k přizpůsobování editoru aplikace. Například když bylo logovací okno minimalizované a uživatel napsal dostatečně dlouhý text, který zasahoval minimálně do začátku okraje logovacího okna, a následně by chtěl logovací okno maximalizovat, tak by se nezměnila výška editoru a uživatel by se nemohl k textu který by byl zakryt logovacím prostředím dostat. Abych tuto chybu odstranil bylo nutné během každé maximalizace/minimalizace logovacího prostředí volat funkce Ace editoru resize() a renderer.updateFull(). Na kód starající se o maximalizaci/minimalizaci prostředí se můžete podívat v následující ukázce a povšimnout si právě těchto funkcí starajících se o správné nastavení velikosti editoru na základě ostatních prvků.

```
1     $("#btn_log_maxi").click(function () {
2         var height_log = $(".log_file").height();
3         $(".log_file").toggle();
4     if ($('.log_file').is(":visible")) {
5               $('#codeEdit').css('bottom', height_log);
6         } else {
7             height_log = 0;
8         }
9      $("#btn_log_mini").show();
10     $('#codeEdit').css('bottom', height_log);
11     editor.resize()
12   editor.renderer.updateFull()
13    });
```

Ukázka kódu 3: Maximalizace a minimalizace logovacího prostředí

Jak si můžete všimnout, tak pro zjištění, zda je logovací okno maximalizované nebo minimalizované slouží funkce`$('.log_file').is(":visible")`, která vrací hodnotu true nebo false na základě css elementu visible.

V části kódů který se stará o maximalizaci se musí nejdříve zjistit velikost logovacího prostředí, která se následně předá jako parametr do funkce css(). [12] Tato funkce nastavuje hodnotu konce editoru. Zjišťování velikosti logovacího prostředí při každé maximalizaci je z toho důvodu, že výška logovacího okna je udávána v procentech (je tedy různá při změně velikosti prohlížeče), a proto nestačí aby byla hodnota zjištěna jen jednou při načtení stránky.

V druhé části kódu, který se stará o nastavení parametrů v případě minimalizace logovacího prostředí, je situace obdobná, jen zde není potřeba hodnota výšky logovacího prostředí, protože při minimalizaci je nulová. Čili se do proměnné `"height_log"` přiřazuje hodnota 0.

Jak jsem se již zmínil, tak velikost logovacího prostředí je udávána v procentech. Proto je zapotřebí funkce, která bude zachytávat změnu velikosti okna prohlížeče a upravovat tak pozici konce editoru. Kdyby tato funkce nebyla implementována

[12] Funkce css mění hodnotu kaskádového stylu. Jako parametr se předává pole JSON

a uživatel by změnil velikost okna prohlížeče (výšku okna), tak by se proporcionálně zmenšila velikost logovacího prostředí. Nicméně konec editoru by byl stále na stejném místě, což je samozřejmě nežádoucí. Zde se na tuto funkci můžete podívat.

```
1   $(window).resize(function () {
2           var   height_log = 0;
3          if ($('.log_file').is(":visible")) {
4             height_log = $(".log_file").height();
5          }
6          $('#codeEdit').css('bottom', height_log);
7        editor.resize();
8        editor.renderer.updateFull();
9      }).resize();
```

Ukázka kódu 4: Reakce na změnu velikosti okna prohlížeče

U této funkce bylo nutné pomocí podmínky rozlišit, kdy je logovací prostředí maximalizované a kdy je naopak minimalizované. V případě, kdy je minimalizované je zapotřebí, aby byl konec editoru až na konci stránky, tedy podobně jak v případě minimalizace logovacího prostředí je do proměnné `"height_log"` přiřazena hodnota 0.

U tlačítka sloužícího pro maximalizaci/minimalizaci se nachází červený čtvereček, ve kterém se zobrazuje číslo indikující počet nalezených chyb (tohoto čtverečku si můžete všimnout na obrázku 9). V případě, kdy není nalezena žádná chyba, tak se u tlačítka nenachází žádný čtvereček. Této funkcionality jsem docílil zejména za pomocí kaskádových stylů a JavaScriptu (jQuery), kdy nejdříve zjistím počet řádků v tabulce (pouze v části tbody), která zobrazuje seznam chyb. Následně pokud je zde alespoň jeden řádek, tak hodnotu počtu řádků vypíšu pomocí funkce `\$("#id_tagu").text(pocet_radku)` do vytvořeného HTML tagu a zobrazím pomocí funkce `\$("#number_errors").css("visibility", "visible"))`.

4.3 Výběr dat z logovacího souboru

Nejdříve pomocí AJAX komunikace získám obsah logovacího souboru ze serveru. V obsahu tohoto souboru změním znaky <> za jejich ekvivalent v HTML entitách pomocí následujícího kódu.

```
1 data=data.replace(/</g,"&lt");
2 data=data.replace(/>/g,"&gt");
```

Ukázka kódu 5: Změna znaků

Tato záměna je nutná, aby se část textu mezi symboly <> nebrala jako HTML tag, ale jako prostý text.

Následně tento obsah logovacího souboru, který je uložen v proměnné ”data” rozdělím do pole, přičemž jeden prvek pole je jeden řádek logovacího souboru. Tohoto jsem dosáhl za pomocí funkce knihovny jQuery ”split(”/n”)”, ke které jsem přidal parametr /n (znak označující konec řádku). Následně jsem si vytvořil regu-

lární výrazy na základě informací popsaných v kapitole „Způsob výběru informací“. Zde jsou jednotlivé regulární výrazy včetně komentáře co mají za úkol nalézt:

```
1  //zjistí zacatek chyby
2  var chyba= /^!.*/;
3  //zjisti radek obsahující 49. nebo 49 * znaci zacatek latexove
       informace (ukonceni chyby)
4  var ukonceni_chyby_znaky=/^(\.{49}|\*{49}).*/
5  //zacatek logu o~latexovych informacich nebo balickach (ukonceni chyby)
6  var ukonceni_chyby_slovy=/^(Package|LaTeX|LaTeX Font):\s.*/
7  //zjisti zda je v~textu cisle radku odkazující na varovani
8  var cislo_varovani=/on input line\s\d+/;
9  //zjisti zda je v~textu číslo rádku odkazjici na bad boxu
10 var cislo_badBox=/at lines\s\d+/;
11 //zjistí zda je v~textu číslo rádku odkazujici na chyby
12 var cislo_chyby = /^l\.[1-9].*/;
13 //nalezení zacatku varování
14 var varovani=/^LaTeX Warning:\s.*/;
15 //nalezeni zacatku dalsiho typu latexoveho varovani (ukonceni chyby)
16 var varovani_latex=/^LaTeX Font Warning:\s.*/;
17 //nalezeni zacatku bad boxu (Overfull)
18 var overfull = /^Overfull\s\\(h|v)box.*/;
19 //nalezeni čzaátku bad boxu (Underfull)
20 var underfull =/^Underfull\s\\(h|v)box.*/;
21 //zjisti zacatek logu informujicim o~vyuzite pameti (ukonceni chyby)
22 var konec_chyby = /^Here is how much of TeX´s~memory you used:/;
```

Ukázka kódu 6: Regulární výrazy

Následuje hlavní část skriptu, kdy pomocí cyklu procházím prvky pole. Když se hodnota v poli shoduje s vytvořenými regulárními výrazy (nalezena chyba nebo varování), tak si ukládám tuto hodnotu do nové proměnné. Následně jsem vytvořil nový cyklus, ve kterém se do proměnné ukládají i následující prvky pole (podrobnosti k chybám/varováním) do doby, než opět dojde ke splnění podmínky na základě nalezení vzoru podle regulárního výrazu, který odkazuje na začátek nového logu. Během průchodu tohoto cyklu zjišťuji, zda se v proměnné neobjevilo číslo řádku (opět za pomocí výše vypsaných regulárních výrazů), kde chyba/varování vznikla/o. V případě, kdy bylo takové číslo nalezeno, je zavolána následující funkce:

```
1 function cisloRadkuChyby(retezec){
2   var hodnota = /l.\d+/;
3   var pouze_cisla = /\d+/;
4   var pom=hodnota.exec(retezec);
5   return pouze_cisla.exec(pom);
6 };
```

Ukázka kódu 7: Zjištění čísla řádku v případě chyby

Zde se nejdříve vybere část řetězce, která v sobě nese informaci o čísle řádku (konkrétně se jedná o znak l.,za kterým následuje číslo řádku). Dále se z tohoto výrazu vybere pouze číslo (vynechám znaky l.). Obdobně se postupuje i u funkce pro zjiš-

tění čísla řádku poukazující na vznik varování. Jen jsou zde jiné regulární výrazy pro zjištění části textu, která je bezprostředně před číslem řádku.

```
1 //bezne varovani
2 var line=/on input line\s\d+/
3 //bad box
4 var line=/at lines\s\d+/
```

Ukázka kódu 8: Regulární výrazy pro zjištění čísla řádku varování

Po zjištění čísla řádků a textu chyby/varování, jsou tyto informace poslány do následující funkce, která slouží k vytvoření řádku tabulky v logovacím prostředí.

```
1  function vytvorRadekTabulky (cislo_radku,popis, typ, selektor,trida){
2  var specificka_tabulka = $(selektor);
3  var tabulka = $("#table_log_all");
4  var radek=$("<tr>").addClass(trida);
5  var text=/.*\n/.exec(popis)
6  popis=popis.replace(/.*\n/,"");
7  if (!popis){
8  popis="neexistuje dalsi informace";
9  }
10 cislo_radku="";
11   }else{
12   radek.attr("value",cislo_radku);
13   };
14   var bunky ="<td>" + typ + "</td><td>" + text + "<div class='
        podrobnosti'><pre><code>" + popis + "</code></pre></div>" + "</td
        ><td></pre>" + cislo_radku + "</td>";
15   radek.append(bunky);
16   radek.appendTo(tabulka);
17   radek.clone().appendTo(specificka_tabulka);
18   $(".podrobnosti").hide();
19 }
```

Ukázka kódu 9: Funkce sloužící k vytvoření řádku tabulky

Vstupními parametry do této funkce jsou *číslo řádku*, kde nastala chyba/varování, *popis* chyby/varování/informace, *typ* zda se jedná o chybu/varování/informaci, *selektor*, který určí, do které tabulky se má řádek vytvořit (zda do tabulky informující o chybách, varováních nebo informacích) a poslední parametr *třída* se přiřazuje k samotnému řádku a slouží pro barevné rozlišení zpráv.

V této funkci si vytvořím tři proměnné. Nejdříve vytvořím proměnnou, která odkazuje na jednu ze tří tabulek (chyba, varování, informace). Druhá proměnná odkazuje na tabulku, kde jsou vypsány veškeré zprávy a třetí proměnná nám bude odkazovat na buňku pro číslo řádku. Následně zjistím text prvního řádku, který se nachází v proměnné popis (hlavní část, která v sobě nese informaci o jaký typ chyby/varování se jedná, bude se zobrazovat v tabulce logovacího prostředí). Dále tento první řádek odstraním a zjistím, zda je zobrazen k dané zprávě i nějaký bližší popis ,pokud ano, tak tento popis také přidám do tabulky. Nic méně tuto informaci skryji. Uživatel si ji bude moci zobrazit dvojklikem na daný řádek tabulky.

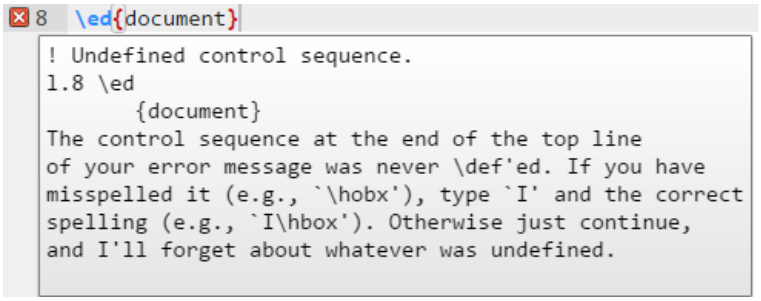
Následně zjišťuji, zda je k dané zprávě připojeno i číslo řádku a případně je vypsáno do dané buňky. V neposlední řadě si vytvořím zbylé buňky řádku a uložím je do proměnné ”buňky” (popis musí být uzavřen v tagu pre, aby bylo zachováno formátování a v tagu code, abych mohl změnit font písma) a tento řádek připojím k tabulkám.

Kromě vytváření tabulky, se volá i funkce sloužící k vytvoření ikonek u čísla řádku, kde vznikla chyba nebo varování. Těchto ikonek jste si mohli všimnout na obrázku 9. K této funkcionalitě jsem využil přednastavené funkce Ace editoru, do které je nutné zaslat pole JSON, a tato funkce již vytvoří ikonky, včetně popisu (viz obrázek 11). V případě, kdy by se na jednom řádku objevilo více chyby nebo varování, tak by se řadily pod sebe. Vyšší prioritu má chyba, proto v případě chyby a varování na jednom řádku bude zobrazena ikonka křížku (značí chybu).

```
1  function vytvorIndikaci (cislo_radku,text,typ) {
2     pole_json.push({
3           "row" : cislo_radku,
4           "text"  : text,
5           "type" : typ
6       });
7  };
8  ...
9  ...
10  editor.getSession().setAnnotations(pole_json);
```

Ukázka kódu 10: Funkce Ace editoru sloužící k zobrazení ikon

Během cyklu tedy volám funkci s názvem vytvorIndikaci, která dostává jako parametr číslo řádku chyby/varování, text chyby/varování včetně podrobností a typ. Typ určí, zda se jedná o chybu nebo varování, což slouží k určení ikony (křížek/vykřičník). Po ukončení cyklu už jen zavolám funkci `editor.getSession().setAnnotations(pole_json)`.



Obrázek 11: Popis chyby při najetí na ikonku křížku

Mimo cyklus se zjišťují informace o verzi, času překladu a počtu stránek na výstupu, protože informace o verzi a času překladu je vždy na prvním řádku logovacího souboru (první prvek pole). Informace o množství stránek na výstupu je naopak poslední řádek logovacího souboru (poslední prvek pole). Tyto hodnoty se pak také

předávají do výše zmíněné funkce (ukázka kódu 9), která vytváří nové řádky do tabulky.

4.4 Nutné změny před nasazením

Vývoj samotné aplikace jsem neprováděl do funkční aplikace TEXonWeb, abych nehrozil její funkcionalitu, a také aby uživatelé mohli tuto aplikaci i nadále plnohodnotně používat. Jelikož jsem nepracoval s backendovou částí aplikace, mohl jsem si dovolit vyvíjet novou funkcionalitu v nefunkční maketě webu, která pouze simulovala její grafické rozhraní.

Pro nasazení na server je potřeba učinit následující akce:

- Přidat do hlavičky v souboru TeXonWeb.html odkaz na script log.js a log.css, případně obsah souboru log.css překopírovat do souboru main.css, protože je zbytečné mít tento soubor samostatně.
- Přidat obsah souboru log.html do souboru TexonWeb.html. První část přidat mezi tagy </section> za tag </pre>, který ukončuje sekci starající se o zobrazení příkazů do editoru při jeho spuštění. Tento tag <pre> má id ”codeEdit”. Druhou část přidat do patičky editoru, tedy mezi tagy <footer id=”statusbar” class=”shadow-controls”></footer>. Konkrétně tento kód dát před tag </footer>.
- V souboru log.js, u funkce starající se o stažení obsahu logovacího souboru ze serveru, je potřeba změnit adresu následujícím způsobem:

```
1 var uzivate = $('#user').val();
2 jQuery.get('/tmp/' + uzivatel + '.log', function(data) {
3 ...
4 ...
5 });
```

- U souboru main.js, který je momentálně používán, je nezbytné odstranit funkci pro zobrazení logovacího souboru, protože je navázána na kliknutí na tlačítko, které nyní slouží pro spuštění akcí, které vedou k výpisu informací do logovacího prostředí. Také je již tato funkce pro otevření logovacího souboru v novém okně zbytečná, protože celý obsah si může uživatel zobrazit přímo v logovacím prostředí.
- Pro správné zobrazování všech znaků (např. česká diakritika) je potřeba, aby byly logovací soubory ukládány v kódování UTF-8

5 Diskuze

5.1 Přínosy navrženého řešení

Hlavní výhodou logovacího prostředí pro uživatele je, že mohou mnohem rychleji a efektivněji vyhledávat chybové stavy, které vznikly během překladu. Největší přínos je to zejména pro začínající uživatele, kteří s touto aplikací často pracují. Domnívám se, že jim tato nová funkcionalita ulehčí práci, kdy už nebudou muset vyhledávat klíčové informace v rozsáhlém a neupravené logovacím souboru, ale budou jim vypisovány do nově vytvořeného logovacího prostředí. Výhodou pro pokročilé uživatele je to, že mohou díky této funkcionalitě urychlit svoji práci při hledání chyb a varování.

5.2 Porovnání s konkurenčními editory

Nespornou výhodou oproti konkurenčním řešením (uvedených v kapitole "Přehled dostupných řešení") je možnost jednoduše si vyfiltrovat například pouze chyby či varování. Dále jsem zde skloubil možnost zobrazení chyby v tabulce, podobně jako to má TEXMaker, s informacemi o chybě/varováním přímo u řádku v editoru, jak je tomu u webových editorů ShareLaTEX či Overleaf. Moje řešení má tedy jak výhody tabulky (přehlednost), tak i možnost zhlédnout informace přímo v editoru a to i v případě minimalizovaného logovacího prostředí.

Výpis samotného obsahu informací je v podstatě totožný. Všechny editory vypisují chyby a varování. Také se všechny editory snaží o barevné rozlišení těchto informací. V mém řešení se navíc vypisují informace o verzi překladače, času vyhotovení logovacího souboru a počtu stránek na výstupu. Podobně jako všechny konkurenční editory nabízím možnost zobrazení celého logovacího souboru. Zde je řešení uděláno podobně jako v TEXMakeru nebo ShareLaTEXu, kdy je možné soubor zobrazit přímo v logovacím prostředí. V případě Overleaf editoru je možnost stažení si logovacího souboru. Já se domnívám, že lepším řešením je možnost zobrazit si logovací soubor přímo v logovacím prsotředí.

5.3 Nevýhody navrženého řešení

Hlavní nevýhody vidím v následující body:

- Skript pro výpis klíčových informací se nespouští automaticky, ale až po stisknutí tlačítka „log“.
- Horší ovladatelnost pomocí mobilu či tabletu.
- I přes relativně rychle fungující průběh získávání klíčových informací z logovacího souboru, kdy jsem nezaznamenal žádné promlky, tak by jistě šel udělat hlubší refactoring kódu a urychlit tak jeho běh, což by vedlo i ke snížení zátěže na klientově zařízení.

- V případě výskytu chyby v logovacím souboru na neočekávaném místě, se ve výpisu v logovacím prostředí s velkou pravděpodobností objeví i nějaké řádky, které s chybou nesouvisí (podrobnosti v podkapitole „Způsob výběru informací“).

Jako menší nevýhody vidím následující:

- Nemožnost přizpůsobit si velikost logovacího prostředí (je zde možnost pouze minimalizace/maximalizace)
- Oproti jiným webovým editorům může vzhled logovacího prostředí působit zastarale.

5.4 Rozšíření navrženého řešení

Jako možnosti dalšího rozšíření se jeví implementovat funkce, které by odstranily poznatky z kapitoly „Nevýhody navrženého řešení“. Zejména by pak bylo vhodné vytvořit funkci automatického spouštění skriptu po kliknutí na tlačítko „PDF“ (slouží k vytvoření výstupního souboru). Tento problém je komplikovaný zejména v tom, že se nevytváří nové logovací soubory, ale pouze se provede update toho starého. Takže se nemůže vytvořit funkce, která by pouze kontrolovala vznik existence souboru, jelikož soubor s daným názvem je na serveru vždy (vyjma prvního překladu uživatele). Proto by mohl vzniknout problém, který by způsoboval získávání informací z neaktuálního logovacího souboru. Podobnou funkcí, kterou by bylo vhodné implementovat, je možnost pouhého sestavení dokumentu. Při této akci by se nevytvářel žádný výstupní soubor, jen by se vytvořil logovací soubor a proběhl výběr klíčových informací. Tato funkce by zamezila vzniku zbytečných výstupních souborů u případů, kdy uživatel pouze kontroluje, zda má správný syntaktický zápis.

Co se týká problému s chybou na neočekávaném místě, tak zde by bylo vhodné v případě nalezení takové chyby (jsou vypisovány i řádky z logovacího souboru, které s chybou nesouvisí) zjistit, jaká informace se za touto chybou vyskytuje a následně tuto informaci zobecnit formou regulárního výrazu a tento výraz přidat do skriptu jako zarážku cyklu, který získává informace o chybě. Věřím, že tento problém by neměl příliš nastávat, protože jsem se snažil objevit veškeré výskyty chyb a případné opravení není nikterak složité.

Zbylé nedostatky by šly řešit relativně snadno. Zejména pak vzhled samotného logovacího prostředí je možné jednoduše upravit za pomocí kaskádových stylů. Stejně tak případné rozšíření o vypisování dalších informací z logovacího souboru znamená jednoduchou úpravu skriptu. Stačilo by pouze tuto novou informaci zobecnit (opět) formou regulárního výrazu. Následně tento regulární výraz přidat do podmínky a za tuto podmínku přidat již existující funkce.

5.5 Zhodnocení

I přes veškeré výše zmíněné nevýhody přináší toto řešení do aplikace TEXonWeb novou funkcionalitu, která ulehčí práci s aplikací a přinese větší pohodlí během jejího používání. Všem uživatelům tato funkcionalita ušetří čas, který by museli věnovat hledání klíčových informací v logovacím souboru. Navíc si myslím, že informace zobrazované v logovacím prostředí jsou přehledně prezentovány a uživatel by neměl mít problém s používáním této nové funkcionality.

6 Závěr

Cílem této práce bylo navrhnout řešení, které by rozšířilo webovou aplikaci TEXonWeb o logovací prostředí. Toto logovací prostředí by mělo být zejména snadno použitelné a přehledné.

Před zahájením návrhu samotného řešení jsem se musel nejdříve seznámit s aplikací TEXonWeb, zejména pak s její frontendovou částí. Také jsem se musel seznámit s typografickým systémem TEX a nadstavbou LaTEX, kde jsem se následně zaměřoval zejména na strukturu a obsah informací v logovacím souboru. Po seznámení se s těmito technologiemi jsem nemusel řešit problém, v jakém programovacím jazyku budu provádět implementaci, neboť bylo více než vhodné pokračovat v technologiích, ve kterých je TEXonWeb napsán (JavaScript, HTML, CSS). Během návrhu a implementace samotného řešení jsem se snažil maximálně vyhovět požadavkům na jednoduchost a snadnou použitelnost logovacího prostředí. Během tvorby jsem navrhované řešení konzultoval s vedoucím práce panem Ing. Janem Přichystalem, Ph.D. a snažil se jeho věcné připomínky do implementace zahrnout.

Domnívám se, že navržené řešení splňuje požadované cíle. V současné době probíhá testování. Jelikož jsem pracoval pouze s frontendovou částí aplikace, tak je možné provádět testování i bez nasazení na server.

Prováděl jsem manuální testy pro zjištění funkčnosti i ovladatelnosti. Zatím nebyly nalezeny žádné kritické chyby. Protože veškeré operace probíhají na straně klienta, nelze v podstatě otestovat prostředí na všech počítačových konfiguracích. Rychlost se tedy odvíjí od daného zařízení, ale v tomto ohledu by neměl nastat žádný problém. V případě velkého množství chyb a varování nalezených na jednom řádku, by mohla nastat následující situace. Při najetí kurzorem myši na ikonku informující o chybě/varování, by mohl být zobrazený text na tolik rozsáhlý, že by se "nevešel" uživateli na monitor. Respektive by si nemohl zobrazit touto formou poslední nalezené informace o chybě/varování. O tom, kolik přesně řádků je možné zobrazit rozhoduje zejména rozlišení monitoru nebo případně nastavení parametrů v prohlížeči jako je "zoomování" stránky. Při běžném nastavení velikosti stránky na 100 procent by ani k této situaci nemělo docházet (u běžného Full HD [13] monitoru se zobrazí kolem 55 řádků). Případně si může uživatel všechny chyby či varování podrobně prohlédnout v tabulce logovacího prostředí. Logovací prostředí jsem testoval v prohlížečích Google Chrome a Mozilla Firefox.

Po úpravách, které jsou uvedeny v kapitole „Nutné změny před nasazením“, je tedy možné logovací prostředí plně využívat v aplikaci TEXonWeb. Řešení je možné si prohlédnout na adrese: `https://akela.mendelu.cz/~xvorac1/TeXonWeb.html` nebo na testovacím serveru TEXonWebu `https://tex.mendelu.cz/devel`. Zdrojové kódy jsou uloženy jako příloha v Univerzitním informačním systému Mendelovy univerzity v Brně.

[13] rozlišení 1920×1080px

7 Literatura

Ace *The high performance code editor for the web* [online]. [cit. 2017-28-04]. https://ace.c9.io/.

Čápka, D. *Úvod do jQuery* [online]. 2017 [cit. 2017-03-29]. Dostupné z: http://www.itnetwork.cz/javascript/jquery-zaklady/javascript-tutorial-funkcionalni-programovani-a-jquery-webova-kalkulacka.

Doob, M. *Jemný úvod do TEXu: Manuál pro samostatné studium* [online]. 1993 2. rozš. a dopl. vyd. Přeložil Daneš J., přeložil Veselý J. Praha: Československé sdružení uživatelů TEXu [cit. 2017-03-19]. Dostupné z: https://ksp.mff.cuni.cz/encyklopedie/jemny-uvod-do-TeXu.pdf.

Dommelen, L. *Warnings while processing LaTeX files* [online]. [cit. 2016-12-04]. Dostupné z: https://www.eng.fsu.edu/ dommelen/l2h/warnings.html.

Dommelen, L. *Warnings while processing LaTeX files* [online]. [cit. 2016-12-04]. Dostupné z: https://www.eng.fsu.edu/ dommelen/l2h/errors.html.

Olšák, P. *Typografický systém TEX.* 2. vyd. Brno: Konvoj, 2000 ISBN 80-7302-007-6.

Olšák, P. *TEXbook naruby.* 2. vyd. Brno: Konvoj, 2001 ISBN 80-85615-91-6.

Overleaf *Plans and Pricing* [online]. 2017 [cit. 2017-03-19]. Dostupné z: https://www.overleaf.com/plans/annually.

Overleaf *Overleaf* [online]. 2017 [cit. 2017-03-19]. Dostupné z: https://www.overleaf.com/.

Přichystal, J., 2014. Možnosti tvorby dokumentů v TEXu pomocí webového prohlížeče. In:*Zbornik príspevkov medzinárodnej konferencie OSSConf 2014* [online]. Bratislava: SOIT [cit. 2017-03-19]. Dostupné z: http://ossconf.soit.sk/images/zborniky/zbornik2014.pdf.

Přichystal, J. *Dokumentace k aplikaci TEXonWeb* [online]. [cit. 2016-12-04]. Dostupné z: https://tex.mendelu.cz/navod.pl.

Rolfe, A. *Warnings* [online]. 1999 [cit. 2016-12-04]. Dostupné z: https://www.stuff.mit.edu/afs/sipb/project/www/latex/guide/node28.html.

ShareLaTeX *Dropbox and LaTeX* [online]. 2017 [cit. 2017-03-19]. Dostupné z: https://cs.sharelatex.com/dropbox.

ShareLaTeX *ShareLaTeX* [online]. 2017 [cit. 2017-03-19]. Dostupné z: https://cs.sharelatex.com.

ShareLaTeX *Tarify a ceny* [online]. 2017 [cit. 2017-03-19]. Dostupné z: https://cs.sharelatex.com/user/subscription/plans.

ŠAFAŘÍK, P. *Editory pro TeX: Aplikace s grafickým rozhraním* [online]. 2010 [cit. 2017-03-19]. Dostupné z: https://www.linuxexpres.cz/software/textove-editory/editory-pro-tex-aplikace-s-grafickym-rozhranim.

TEXTERMS *Log File* [online]. 2010 [cit. 2016-12-04]. Dostupné z: https://techterms.com/definition/logfile.

TEXMAKER *The universal LaTeX editor* [online]. 2017 [cit. 2017-03-19]. http://www.xm1math.net/texmaker/index.html.

TEXMAKER *Screenshots* [online]. 2017 [cit. 2017-03-19]. http://www.xm1math.net/texmaker/shots.html.

WIKIBOOKS *LaTeX/Errors and Warnings* [online]. 2015 [cit. 2016-12-04]. Dostupné z: https://www.en.wikibooks.org/wiki/LaTeX/Errors_and_Warnings.

Přílohy

A Seznam LaTeXových varování

- LaTeX Warning: Citation '...' on page nnn undefined on input line nnn.
- LaTeX Warning: Command ... invalid in math mode on input line nnn.
- LaTeX Warning: Float too large for page by nn.nnnpt on input line nnn.
- LaTeX Warning: 'h' float specifier changed to 'ht'.
- LaTeX Warning: Label '...' multiply defined.
- LaTeX Warning: Label(s) may have changed. Rerun to get cross-references right.
- LaTeX Warning: Marginpar on page nnn moved.
- LaTeX Warning: No ... typeface in this size, using ...
- LaTeX Warning: Oval too small.
- LaTeX Warning: `\oval, \circle, or \line` size unavailable on input line nnn.
- LaTeX Warning: Reference '...' on page nnn undefined on input line nnn.
- LaTeX Warning: There were multiply-defined labels.
- LaTeX Warning: There were undefined references.
- LaTeX Warning: You have requested release '...' of LaTeX,
- LaTeX Warning: You have requested, on input line ..., version
- LaTeX Warning: ... in math mode.

(Dommelen, 2017a)

B Seznam LaTeXových chyb

- ! LaTeX Error: Bad `\line or \vector` argument.
- ! LaTeX Error: Bad math environment delimiter.
- ! LaTeX Error: Bad use of `\\`
- ! LaTeX Error: `\begin{...} ended by \end{...}`
- ! LaTeX Error: Can be used only in preamble.
- ! LaTeX Error: `\caption` outside float
- ! LaTeX Error: Command ... already defined
- ! LaTeX Error: Command `\circle` invalid in math mode.
- ! LaTeX Error: Command `\end{itemize}` invalid in math mode
- ! LaTeX Error: Counter too large.
- ! LaTeX Error: Environment ... undefined.
- ! LaTeX Error: File '...' not found
- ! LaTeX Error: Float(s) lost.
- ! LaTeX Error: Illegal character in array arg.
- ! LaTeX Error: Lonely `\item--perhaps` a missing list environment.
- ! LaTeX Error: Missing `\begin{document}`
- ! LaTeX Error: Missing p-arg in array arg
- ! LaTeX Error: Missing @-exp in array arg
- ! LaTeX Error: No such counter
- ! LaTeX Error: Not in outer par mode.
- ! LaTeX Error: Option clash for package csvtools.
- ! LaTeX Error: `\pushtabs and \poptabs` don't match.
- ! LaTeX Error: `\RequirePackage or \LoadClass` in Options Section.
- ! LaTeX Error: Something's wrong–perhaps a missing `\item`
- ! LaTeX Error: Suggested extra height (...) dangerously large.
- ! LaTeX Error: Tab overflow.
- ! LaTeX Error: There's no line here to end

- ! LaTeX Error: This may be a LaTeX bug
- ! LaTeX Error: This NFSS system isn't set up properly.
- ! LaTeX Error: Too deeply nested.
- ! LaTeX Error: Too many columns in eqnarray environment.
- ! LaTeX Error: Too many unprocessed floats
- ! LaTeX Error: Two `\LoadClass` commands.
- ! LaTeX Error: Two `\documentclass or \documentstyle` commands.
- ! LaTeX Error: Undefined tab position.
- ! LaTeX Error: Unknown option ... for package ...
- ! LaTeX Error: `\usepackage` before `\documentclass`.
- ! LaTeX Error: `\<` in mid line
- ! LaTeX Error: ... undefined

(Dommelen, 2017b)