Bakalářská práce

České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Vytvoření REST služby pro konverzi LaTeX souborů do PDF

TADEÁŠ KYRAL

Vedoucí: Ing. LUKÁŠ ZOUBEK
Obor: Softwarové inženýrství a technologie
Zaměření: žádné
Leden 2019

Poděkování

Především bych chtěl poděkovat svému vedoucímu Ing. Lukáš Zoubek za příjemnou spolupráci a cenné rady při našich konzultacích. Také bych chtěl poděkovat Bc. Jiří Fryčovi za poskytnutí informací a rad a v neposlední řadě mé rodině a kamarádům, kteří mě při psaní této práce podporovali.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškerou použitou literaturu.

V Praze, 11. ledna 2019

Abstrakt

Práce pojednává o návrhu webové služby pro konverzi LaTeXu do PDF pro uživatele v Moodle a CourseWare. Služba má uživatelům usnadnit práci s LaTeX ovými soubory přímo skrze portál, bez potřeby stahování a kompilace na svém stroji.

Klíčová slova: LaTeX, PDF, Moodle, CourseWare, Web service, REST, Java EE

Vedoucí: Ing. LUKÁŠ ZOUBEK
Katedra ekonomiky, manažerství a humanitních věd

Abstract

Thesis consists of design of web service providing conversion of LaTeX to PDF. This service will be available for users of Moodle and CourseWare. It helps them to create PDF inside of web browser instead of compiling it on their computer.

Keywords: LaTeX, PDF, Moodle, CourseWare, Web service, REST, Java EE

Title translation: Creation of REST service for conversion from LaTeX to PDF

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

LaTeX patří v akademickém prostředí mezi velmi oblíbené typografické systémy a je hojně využíván ke psaní skript a odborných prací. Mnoho akademiků využívá LaTeX i k vytváření materiálů pro studenty, primárně v oblasti matematiky, jelikož nabízí jednoduché nástroje ke psaní vzorců. Hlavním zdrojem materiálů pro studenty jsou portály Moodle a CourseWare, kam vyučující dokumenty nahrávají a můžou je tam i upravovat. Nynější editor podporuje jenom řádkové příkazy a není schopen zpracovat celý LaTeX dokument natož s více zdrojovými soubory. Tento stav mnoha učitelům nevyhovuje, protože by chtěli svoje dokumenty upravovat a přímo kompilovat v prohlížeči bez potřeby je stahovat a následně zase nahrávat.

Práce se dělí na čtyři hlavní části. Nejdříve proběhne seznámení s potřebnou teorií a technologiemi, dále budou specifikovány požadavky na službu, poté budou představena již podobná existující řešení a porovnají se s požadavky. Následně se přistoupí k návrhu samotné aplikace.

1.1 Cíle práce

Hlavním cílem práce je poskytnout konverzi LaTeX souborů do PDF pro uživatele Moodle a CourseWare. V semestrálním projektu budou rozpracovány tyto dva dílčí cíle: analýza a návrh.

1.1.1 Analýza

V analýze jsou stanoveny tyto cíle:

- Získat a zformulovat požadavky na službu.
- Naleznout již eistující řešení, zabývající se touto problematikou a porovnat je vůči požadavkům.
- Porovnat LaTeX kompilátory.

1.1.2 Návrh

V návrhu budou naplněny tyto cíle:

- Navrhnout řešení daného problému.
- Na základě požadavků stanovit technologie.

1.1.3 Implementace

V rámci implementace budou dosaženy tyto cíle:

Kapitola 2

Teorie a technologie

V této části si přiblížíme technologie potřebné k návrhu a vývoji webové služby specifikované v zadání práce. Postupně budou vysvětleny všechny zásadní pojmy, které pomůžou čtenáři doplnit znalosti v dané problematice.

2.1 REST

Representational state transfer (dále jen REST) je architektura pro komunikaci mezi distribuovanými systémy. Termín zavedl R. T. Fielding ve své disertační práci[3], kde toto rozhraní bylo taktéž popsáno a vymezeno. Mimo jiné stanovil i těchto 5 základních pravidel, která by měla být dodržena, aby se aplikace nazývala RESTful[4]:

- Klient-Server (Client-Server) - Toto omezení staví na principu oddělení zodpovědností (Separation of Concerns), nebo-li je klientská část starající se o uživatelské rozhraní je oddělena od serverové části přistupující k databázi. Zlepšuje škálovatelnost sytému a zjednodušuje použitelnost na různých platformách.
- Bezestavovost (Stateless) - Každý požadavek musí přenášet všechna související data, server totiž neuchovává žádné informace o nynějším spojením a každý požadavek bere jako nový.
- Keš (Cache) - Data přenášená v odpovědi mohou být označená jako kešovatelná, tudíž si je klient může uložit a kdykoliv použít znova.
- Jednotné rozhraní (Unified interface) - Základem tohoto omezení je princip HATEOAS (Hypermedia As The Engine Of Application State), který říká, že klient nepotřebuje znát pravidla komunikace dopředu a data musí obsahovat odkazy na další data v aplikaci. Měla by být jasně definovaná adresa zdroje (např. URI), reprezentace přenášených dat (např. HTML), typ média (např. JSON)
- Vrstvený systém (Layered system) - Přidáním vrstev se aplikace, kde každá vrstva je izolovaná a může komunikovat jenom se sousedícími vrstvami, zpřehlední a zlepší se její škálovatelnost.

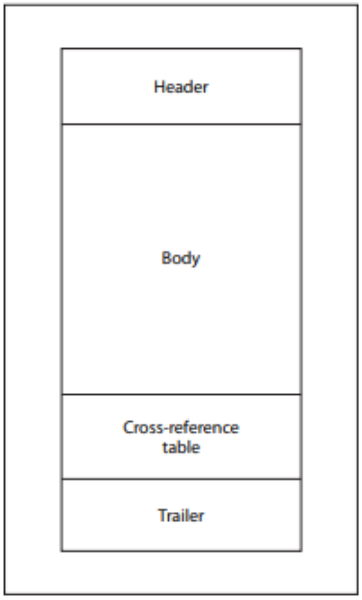
Nejčastějším typem protokolu využívající tuto architekturu je Hypertext Transfer Protocol(HTTP). Pomocí čtyř hlavních metod GET, PUT, POST, DELETE v požadavku poslaném z klienta server buď vrátí požadovaná data, přijme data poslaná v těle a uloží do databáze, nebo data vymaže.

2.2 PDF

Portable Document Format (dále jen PDF), jak už název napovídá, představuje o formát dokumentů, jejichž hlavním cílem je poskytování nezávislosti na platformě. Je velmi podobný programovacímu jazyku PostScript, ze kterého vzešel. Syntaxe souboru je nejlépe pochopitelná jako čtyři komponenty[1]:

- Objekty (Objects) - Základní jednotka celého souboru např. Pole, Čísla, Řetězce znaků. Jednotlivé objekty se popisují množinou znaků, která je definována lexikálními konvencemi.
- Struktura souboru (File Structure) - Samotný soubor se skládá ze čtyř částí, jak je i vidět na obrázku 2.1, ty si stručně popíšeme:
  - hlavička - identifikuje verzi PDF
  - tělo - obsahuje použité objekty, které reprezentují obsah dokumentu
  - tabulka odkazů - obsahuje odkazy na objekty v podobě počtu bytů od začátku souboru, kvůli náhodnému přístupu bez potřeby číst celý soubor
  - závěrečná sekce - udává pozici tabulky odkazů a speciálních objektů

  Tato struktura napomáhá náhodnému přístupu k jednotlivým částem a usnadňuje jejich aktualizaci.
- Struktura dokumentu (Document Structure) - Popisuje hierarchickou strukturu objektů v těle dokumentu.
- Proud obsahu (Content stream) - Objekt, ve kterém se nacházejí instrukce k vykreslování grafických elementů. Každá stránka má minimálně jeden a na rozdíl od ostatních objektu je procházen sekvenčně.



Obrázek 2.1: Struktura objektů

2.3 LaTeX

Vychází z typografického sázecího systému TEX, který popisuje Pavel Satrapa ve své knize [5]: „*Patří do rodiny tak zvaných značkovacích jazyků (markup languages) a dal by se zjednodušeně charakterizovat jako programovací jazyk pro sazbu textů. Jeho základním vstupem je textový soubor, který obsahuje jak sázený dokument, tak příkazy ovlivňující sazbu. Určité znaky mají přiřazen speciální význam a jejich prostřednictvím jsou v textu odlišeny řídicí konstrukce. Typickým příkladem je zpětné lomítko, jímž začínají příkazy.*“

LATEX je rozšíření TEXu o balíček přednastavených řídicích konstrukcí. Hlavní cílem těchto systémů je jednoduchost při psaní matematických a jiných vzorců. Ovšem je také velmi oblíben kvůli možnosti jednoduše upravovat dokumenty podle potřeby, přestože prvotní seznámení je ve srovnání s jínymi nástroji ke psaní náročnější.

Základní soubory mají příponu tex kromě souboru s nastavením, ten je ve formátu cls, neboli class file. Tyto soubory mohou být uspořádány do stromové struktury, kde v kořenu stromu je jeden hlavní soubor, do kterého jsou vnořovány další. To napomáhá přehlednosti velkých dokumentů a také používání již vytvořených.

Soubor se skládá z preambule, která obsahuje nastavení pro celý dokument a případně používané balíčky. Ve druhé části dokumentu se mimo jiné nacházejí kapitoly, sekce a podobně. Základní syntaxe může vypadat takto.

```
\documentclass{thesis}
\usepackage{csquotes}

\begin{document}
Hello world!
\end{document}
```

Výše zmíněné balíčky slouží k přidávání dalších předvytvořených řídících konstrukcí. Tyto balíčky jsou buď nainstalovány zároveň s instalací kompilátoru, nebo je možné je ručně doinstalovávat podle potřeby.

Pro získání výsledného dokumentu, například ve formátu PDF, je potřeba vše zkompilovat. To provádí kompilátor.

2.4 Moodle a CourseWare

Ačkoliv tyto portály se v mnoha aspektech liší, v rámci této práce je můžeme považovat za ekvivalentní. Oba slouží pro podporu studia, a to hlavně v podobě poskytování informací o předmětech studentům, kteří je mají zapsané. Vyučující takto může předat informace o harmonogramu kurzu, přednášky a jiné materiály. Moodle je komplexnější open-source systém s možností upravovat ho k potřebě vzdělávacího zařízení, který navíc poskytuje prostředí pro vytváření a psaní testů, zapisování docházky, komunikaci se studenty a jiné.

2.5 FURPS+

Vychází z klasifikace požadavků FURPS(**F**unctionality, **U**sability, **R**eliability, **P**erformance, **S**upportability) se kterou přišel Robert Grady v roce 1992. Roku 1999 Jacobson at el rozšířili specifikaci o znaménko „+“, které přidává požadavky a omezení na návrh, implementaci, rozhraní a hardware[2].

- Funkční požadavky
  - Fukčnost(Functionality) - Požadavky popisující všechny hlavní prvky produktu i důležité aspekty z pohledu architektury např. lokalizovaný systém pro více jazyků.
- Nefunkční požadavky
  - Použitelnost (Usability) - Zaměřuje se na uživatelskou přívětivost nejenom samotné aplikace, ale i dokumentace týkající se estetiky a konzistence.
  - Spolehlivost (Reliability) - Spolehlivost systému v podobě doby běhu, správnosti fungování a četnosti výpadků.
  - Výkon (Performance) - Vypovídá o výkonnosti systému, jak rychle dokáže zpracovávat požadavky, spustit se atd.
  - Podporovatelnost (Supportability) - Popisuje testovatelnost, škálovatelnost, konfigurovatelnost...

- Návrh (Design) - Omezení na návrh systému např. požadavek na relační databázi
- Implementace (Implementation) - Specifikuje typ programovacího jazyku, platformu apod.
- Rozhraní (Interface) - Komunikace s externími systémy
- Fyzické (Physical) - Definuje požadavky na hardware, na kterém daný software poběží, i co se týče fyzické velikosti

Toto rozdělení nám pomáhá identifikovat požadavky. Přispívá k vyšší kvalitě systému a snižuje pravděpodobnost přehlédnutí funkcionality. Právě díky těmto vlastnostem je velmi oblíbené a využívané k vývoji jakéhokoliv software.

Kapitola 3

Analýza

3.1 Požadavky

Nyní představíme požadavky kladené na aplikaci. Ty jsou buď požadované zadávajícím nebo odvozené z potřeb, ke kterým bude služba používána, ale i z omezení plynoucích z prostředí, v jakém bude provozována, v tomto případě z prostředí fakulty státní vysoké školy, jmenovitě Fakulty elektrotechnické, ČVUT.
Za pomoci metody FURPS+ rozdělíme požadavky a omezení na funkční a nefunkční.

3.1.1 Funkční požadavky

Typ	Požadavky

Funkčnost	• Webová služba s REST rozhraním pro komunikaci
	• Umět přijmout požadavky a soubory
	• Zkompilování LaTeX souborů s požadovaným nastavením do PDF
	• Uložení PDF a jeho poskytnutí
	• Uživatel může zvolit tyto nastavení:
	  • Přidat vodoznak na každou stranu PDF
	  • Nakonec souboru přidat stránku s předem stanoveným obsahem
	  • Výsledný soubor PDF bude chráněný
	  • Výsledný soubor PDF nebude kopírovatelný nebo tisknutelný

Tabulka 3.1: Funkční požadavky

Služba ma jediný a specifický účel, z toho plyne menší množství funkcionalit.

3.1.2 Nefunkční požadavky

Typ	Požadavky
Použitelnost	• Dokumentace k rozhraní na swagger [1] • Přístup skrz REST Api, pouze s tokenem[2]
Spolehlivost	• Služba není kritická • Uptime 95% času • Ochrana proti špatnému vstupu • Autorizace pomocí tokenů
Výkon	• Zvládnutí obsloužení desítek požadavků najednou • Ukládání výsledných dokumentů po dobu jednoho měsíce • Maximální doba tvorby dokumentu 5 minut
Podporovatelnost	• Služba může být rozšířena o kompilování samotného TeXu a může podporovat vytváření i jiných formátů z LaTeXu
Implementace	• Platforma - Java EE • Komunikace - REST Api • Operační systém - Ubuntu nebo CentOS
Rozhraní	• Komunikuje s Moodle a CourseWare, tyto portály posílají data
Fyzické	• Musí být provozováno na serverech ČVUT

Tabulka 3.2: Nefunkční požadavky

Dúvody některých požadavků nemusejí být úplně jasné, proto je zmíníme.

- Platforma - Je požadováno prostředím vývoje, kde Java EE je hlavní platforma. Tudíž je zajištěna podpora.
- Operační systém - Také vyžadováno z pohledu podpory, zmíněné systémy jsou na serverech, kde bude služba nasazena nejčastěji používány a tudíž správci tyto systémy znají.
- Služba není kritická - Poskytuje funkčnost, která nijak neovlivňuje základní funkcionality určených portálů.
- Musí být provozováno na serverech ČVUT - Jelikož aplikace bude pracovat s dokumenty a popřípadě i informacemi, spojenými s pracovníky školy, je potřeba tato data chránit a uchovávat na vlastních serverech z důvodu GDPR.

[2] https:/swagger.io/

Z tabulky je vidět, že služba není nijak kritická a působí jenom jako doplněk výše zmíněných portálů. Musí ovšem splňovat vyšší bezpečnostní nároky souvisejícími prostředím, v jakém se bude používat.

3.2 Existující řešení

Na základě stanovených požadavků v minulé kapitole přejdeme k analýze již existujících řešení. Momentálně uživatelé Moodle mohou vkládat do svých souborů řádkové příkazy, což není úplně dostačující a nesplňuje to požadavky. Samozřejmě se dají používat pro vytváření PDF z LaTeX souborů kompilátory, které lze stáhnout a používat lokálně. To ovšem není předmětem této práce, a proto se podíváme na webové služby, které více odpovídají potřebám a požadavkům. Pár vybraných si představíme.

3.2.1 OverLeaf

Placená služba[3] pro tvorbu LaTeX dokumentů, kterou lze s některými omezeními používat i zdarma. Je velmi oblíbená hlavně kvůli přívětivému prostředí pro tvorbu dokumentů a jejich správu. Také nabízí výhody pro určité zájmové skupiny, nejzajímavější vzhledem k tématu této práce je předplatitelská služba OverLeaf Commons [4]. Ta poskytuje sdílené prostředí se všemi výhodami pro zaměstnance a studenty univerzity. Ale jako všechny ostatní služby je i tato placená, ovšem není jisté, jestli je poskytována všem univerzitám a ani za jakých podmínek.

3.2.2 BlueLaTeX

Open-source služba[5] pro kompilaci LateX souborů. Kompilovat lze na jejich serveru nebo nabízejí kód pro spuštění na vlastním. Kromě samotné serverové implementace je k dispozici i webový klient. Vše je zatím pouze v Beta verzi a samotní tvůrci varují před možnými nedostatky a problémy. Na jejich oficiálních stránkách je poslední aktivita z roku 2015 a zdá se, že tento projekt už není aktivní. Hlavním lákadlem je souběžná spolupráce více lidí na jednom dokumentu a také možnost provozovat server lokálně.

3.2.3 ScienceSoft

Starší služba[6], která mimo kompilace LaTeX souborů vložených přes webový prohlížeč poskytuje i jiné rozhraní pro poskytnutí souborů, a to jmenovitě REST Api a SOAP. Jak bylo řečeno, tak tato služba je starší, tudíž pro kompilaci používá zastaralý TexLive 2008 a její vývoj není aktivní.

[3] https://www.overleaf.com
[4] https://www.overleaf.com/for/universities
[5] http://www.bluelatex.org/
[6] http://sciencesoft.at/latex/index?lang=en

3.2.4 ShareLaTeX

Velmi podobný BlueLaTeXu, tedy open-source[7] nabízející jimi hostovanou verzi nebo lokální verzi pro vlastní potřebu, obě verze obsahují mnoho funkcí včetně grafického prostředí pro uživatele. Pod jménem ShareLaTeX se vyskytuje jenom výše zmíněné, ovšem společně s OverLeaf také spravují službu Pro[8], která je určená pro firmy, ale i univerzity. Ta se pyšní možností nasazení na vlastních serverech, správou uživatelů, podporou a zabezpečením.

3.2.5 Porovnání

Každé z těchto řešení má své problémy, ať už že je zastaralé a neaktualizované nebo nesplňující zanalyzované požadavky(sekce 3.1). Do první skupiny patří ScienceSoft a BlueLaTex, kde první z jmenovaných ani neposkytuje kód pro implementaci na lokálním serveru a druhý je v nedodělaném stavu, což může vést k nefunkčnosti a bezpečnostním rizikům. Služba Overleaf Commons nesplňuje stejný požadavek jako řešení od ScienceSoft, ale nabízí zajímavé prostředí pro vytváření a správu LaTeX dokumentů pro studenty i zaměstnance, o čemž by univerzita mohla popřemýšlet. Nejnadějnější možností se zdá být ShareLaTeX, který poskytuje k použití i jenom backendovou část pro kompilaci LaTeX a komunikaci, ale celý je implementovaný v CoffeeScriptu, což je v rozporu s požadavkem na implementaci, kde je Java EE.

3.3 Kompilátory

Nejdůležitější částí celé aplikace bude kompilátor, který bude provádět kompilaci LaTeX souborů, ale může poskytnout i podporu pro operace s výsledným PDF. Tyto operace vycházejí ze stanovených funkčních požadavků (sekce 3.1.1). Jelikož kompilace bude probíhat na serveru s Linuxovým operačním systémem, nebudeme zahrnovat do srovnání kompilátory pro Windows.

3.3.1 MikTeX

MikTeX [9] je velmi oblíbený hlavně na operačním systému Windows, a to díky příjemné instalaci a kvůli možnosti doinstalovávat balíčky „on-the-fly“, neboli za běhu. Ale je poskytován mimo jiné i pro Linux, a to například pro potřeby této práce v zajímavé verzi „Just enough TeX“. Tato instalace obsahuje jen to nejnutnější, tedy bez zbytečných balíčků navíc. Je vyvíjen jediným programátorem, který se stará o celou distribuci, což je do budoucna poněkud rizikové z pohledu udržitelnosti. Podporované Linuxové operační systémy jsou např. nejnovější Ubuntu a Debian.

[7] https://github.com/sharelatex/clsi-sharelatex
[8] https://www.overleaf.com/for/enterprises
[9] https://miktex.org

3.3.2 TeXLive

TeXLive je kompilátor spravovaný skupinou přispěvatelů, kteří ho udržují. Existují dvě různé cesty jak TeXLive[10] nainstalovat a od toho se odvíjí správa balíčků. Verze Native TeX Live a distribuce přímo pro daný operační systém. Liší se počtem balíčků po instalaci a způsobem jejich aktualizací. V Native verzi probíhají aktualizace pomocí TeX Live Manager neboli tlmgr, jinak se o to stará samotný operační systém. V případě verze Native je možno si i zvolit schéma, které určuje množství balíčků např. scheme-basic obsahuje jenom to nejnutnější. Podporuje skoro všechny Linuxové distribuce.

3.3.3 Porovnání

Oba kompilátory jsou si velmi podobné a liší se jenom v drobnostech, některé z nich už byly nastíněny v jejich popisech. Jedním z důležitých aspektů je velikost instalace, tedy i s balíčky, v této kategorii nabízejí oba to samé. K tomuto se váže práce s balíčky, kde už je jiná situace, a to kvůli jasné výhodě MikTeXu spočívající ve stahování balíčků za běhu. Něco podobného lze dosáhnout i v TexLive, ale je nutno použít externí skripty, což může mnohem více ovlivňovat rychlost kompilace. Rozdíl je také ve správě samotných kompilátorů, kde MikTeX je náchylnější k výpadkům aktualizací nebo celkovému zastavení vývoje, a to kvůli jedinému člověku, který se o něj stará. To může vést i k dalším problémům, co se týče bezpečnosti a podpory.

[10] https://www.tug.org/texlive/pkginstall.html

Kapitola 4

Vlastní řešení

V této části navrhneme podobu vlastního řešení problému specifikovaného v předchozích kapitolách. Návrh by měl nabízet jednoduché a vyhovující řešení vycházející ze stanovených požadavků a cílů.

4.1 Návrh

Cílem práce je poskytnout konverzi LaTeX souborů do PDF, čehož bude dosaženo pomocí webové služby. Tato služba bude poskytovat svoje rozhraní a funkcionalitu portálům Moodle a CourseWare. Jedná se o jednoduchou aplikaci s jediným účelem, tedy kompilace LaTeX souborů do PDF. Tudíž není potřeba složité databáze, ale stačí pouze SQLite[1], což je velmi odlečená verze klasických SQL databází, která ukládá data přímo do souboru v souborovém systému, kam bude ukládán i výsledný soubor PDF.

4.1.1 Platforma

Na základě požadavků bude aplikace postavena na Java EE, což je platforma pro vývoj webových aplikací rozšiřující standardní Javu SE. Oproti ní poskytuje některé zásadní techniky navíc, jednu si tedy představme.

Vkládání závislostí (dependency injection) umožňuje objektu používat jiné objekty bez potřeby ho zatěžovat jejich vytvářením. Objekty, které můžeme takto vkládat, se nazývají beany a právě o jejich vytváření a zánik se stará Contexts and Dependency Injection (dále jen CDI) kontejner.

Dále je potřeba specifikovat aplikační server. Ten poskytuje pro webové aplikace běhové prostředí, tedy zajišťuje správu databázových spojení apod. Na základě zkušeností je vybrán open-source Payara, který staví na GlassFish, oproti němu poskytuje častější aktualizace a opravy chyb.

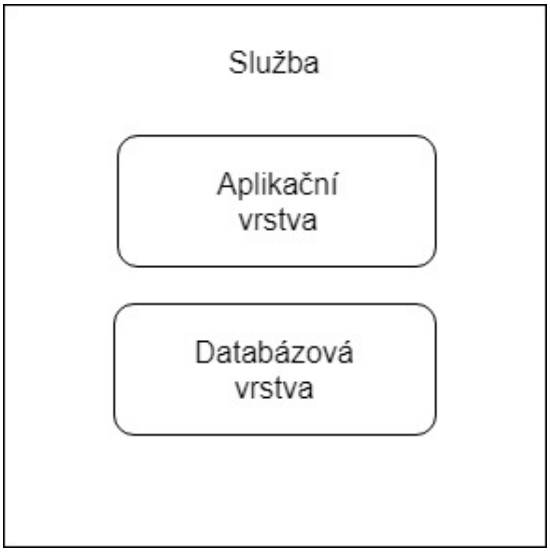
4.1.2 Architektura

Architektura popisuje z jakých částí se aplikace skládá, jak mezi sebou tyto části komunikují a jakým způsobem procházejí informace a požadavky aplikací. Při navrho-

[1] https://www.sqlite.org/

vání architektury je potřeba zvážit několik věcí, například rozšířitelnost, modularizaci, odezva a složitost.

Nejpoužívanšjší architekturou u webových aplikací je klient-server architektura s n vrstvami, která aplikaci rozděluje na n fyzických médií a n vrstvá architektura, která označuje n logických celků.[2] Nejčastěji jsou obě architektury 3 vrstvé, tedy rozdělení je následující: prezentační, aplikační a databázová vrstva. Ovšem služba, kterou se zabýváme v této práci nemá uživatelské rozhraní a vystavuje jenom určité API, tudíž odpadá prezentační vrstva. Ani neobsahuje databázový server, zůstává tedy jenom aplikační fyzická vrstva, což z naší služby dělá jedno-vrstvou klient-server aplikaci. Zbývá ještě vyřešit logickou architekturu, kde se budeme držet také zavedeých postupů, ale zbavíme se prezentační vrstvy a zůstane nám 2-vrstvá architektura obsahující byznys logiku a přístup k databázi.

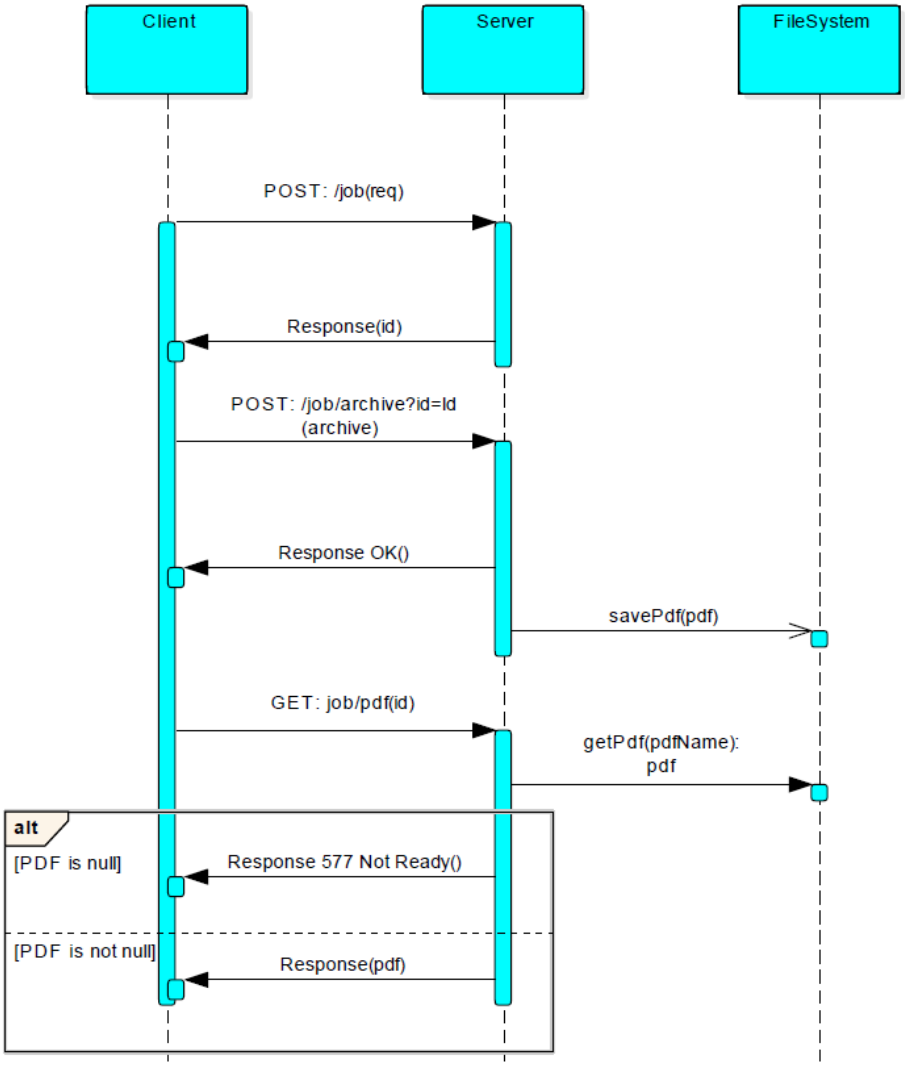


Obrázek 4.1: Architektura

4.1.3 Komunikace

Pro komunikaci mezi klientem a serverem je vybráno REST Api. Poskytuje jednoduché rozhraní, se kterým je schopen komunikovat jakýkoliv systém pouze na základě znalosti struktury požadavků a odpovědí. Posílání zpráv bude postaveno nad protokolem HTTP pomocí GET a POST metod. Na obrázku 4.2 je zobrazena komunikace mezi serverem a klientem.

[2] V anglických zdrojích se používají výrazy N-Tier Client-Server architecture a N-Layered architecture, kde tier znamená to samé jako layer, tudíž vrstva



Obrázek 4.2: Sekvenční diagram

Kompletní dokumentace k rozhraní se nachází na swagger[3]. Na ukázku vypíšeme alespoň používané zdroje:

- **POST** `/job` - Slouží k předání požadavků na výsledné PDF.
- **POST** `/job/archive?id=<jobId>` - Musí obsahovat zip archiv s potřebnými soubory pro kompilaci.
- **GET** `/job/pdf?id=<jobId>` - Vrací výsledné PDF, pokud už je hotové.
- **HEAD** `/job/pdf?id=<jobId>` - Odpověď obsahuje pouze hlavičku, která informuje o stavu PDF

[3] https://app.swaggerhub.com/apis/Tadky/Thesis/1

- **GET** `/info` - Slouží k získání informací ohledně serveru, tedy verze kompilátoru, operačního systému, zbývajícího volného místa na disku

Všechny tyto zdroje navíc obsahují parametr[4] token, který zajišťuje autentizaci. Ta je rozebrána v další sekci.

4.1.4 Zabezpečení

Zabezpečení bude zavedeno jenom v minimální míře pomocí statických tokenů, které budou mít dané portály pro sebe k dispozici. Vzhledem k povaze dat není nutné používat OAuth[5] server, jehož implementace by byla značně nad rámec mé práce. Tokeny budou sloužit k autentizaci portálů, jímž budou tokeny vygenerovány a předány jejich správcům. Každá zpráva poslaná na server bude obsahovat token, který se bude ověřovat oproti tokenu uloženém v properties[6] souboru. Token bude posílán v parametru dotazu v otevřené formě.

4.1.5 Kompilace

Pro vytvoření PDF je nutné příslušně zkompilovat celý LaTeX projekt. Pro úspěšnou kompilaci je potřeba, aby kompilátor měl k dispozici všechny balíčky, které jsou použity v projektu. Jsou tři způsoby, jak toho docílit.

1. Stáhnout všechny balíčky již při instalaci kompilátoru
2. Nainstalovat čistý kompilátor
   a. Stáhnout několik balíčků a jenom ty se budou používat
   b. Získávat balíčky podle potřeby za běhu

První možnost je zbytečná z důvodu mnoha nepoužívaných balíčků a velikosti na disku. Druhá možnost nabízí dvě podmožnosti, kde volba s přednastavenými balíčky vytváří omezení pro uživatele a tím pádem je může odrazovat od používání. Nejvíce vhodná se zdá poslední možnost, která eliminuje všechny neduhy předchozích, tím pádem je i zvolena.

Na základě porovnání kompilátorů (sekce 3.3.3) z předchozí kapitoly je zvolen MikTeX, který poskytuje vše potřebné a nabízí vhodné funkcionality pro téma této práce např. doinstalovávání balíčku za běhu. Bude nainstalován ve verzi „Just enough TeX“, tudíž nebude obsahovat žádné balíčky. Ty se právě budou doinstalovávat až na požadavek při kompilaci.

4.1.6 PDF úpravy

Jelikož jsou kladeny požadavky na výsledný soubor PDF a bohužel kompilátory tyto úpravy nepodporují, je potřeba použít jiný nástroj, který bude umožňovat měnit soubor

[4] Typ parametru, který se posílá v samotné URL daného dotazu ve formě klíč=hodnota
[5] Protokol pro autentizaci a autorizaci aplikací.
[6] Soubor v němž jsou data uspořádány klíč = hodnota

podle požadavků. Nejvíce vhodný se zdá PDFtk[7], který je pod GPL[8] licencí a nabízí vše potřebné. Tedy po zkompilování do PDF se za pomoci výše zmíněné aplikace aplikují požadavky, které byly obdrženy od klienta.

4.1.7 Server

V této fázi návrhu máme už všechny potřebné informace, aby mohl být zvolen operační systém a stanoveny požadavky na server. Jsou známy dvě omezující podmínky kladené na systém. První vychází z požadavků: operační systém musí být Debian nebo CentOS. Druhou určuje kompilátor. Jelikož byl zvolen MikTeX, který je podporován na Debian a Ubutnu, volba je jednoznačná. Na základě průniku je vybrán nejnovější Debian, tedy verze 9 s označením „stretch“.

Je také potřeba stanovit velikost diskového pole potřebného pro fungování. Jelikož nejvíce místa budou zabírat balíčky a potažmo výsledné pdf, budeme vycházet hlavně z těchto údajů. Přibližná velikost všech balíčků obsažených v plné verzi kompilátoru je zhruba 4GB, což bylo zjištěno na základě testovací instalace. Dále je potřeba odhadnout velikost všech PDF, které musí server uchovávat po dobu jednoho měsíce, v jeden moment. Horní odhad pro počet vytvořených PDF za den je 30 a pro velikost souboru je 10MB. Tedy na konci měsíce se může očekávat velikost cca 10GB. Tedy celkové požadované místo je 15GB.

[7] https://www.pdflabs.com/tools/pdftk-server/

[8] Poskytuje svobodu šíření, provozování a upravování daného software.

Kapitola 5

Implementace

Jak už bylo zmíněno v sekci architektura, aplikace nemá prezentační vrstvu a sestává se jenom z aplikačního serveru, který má na starosti komunikaci s klienty, byznys logiku a persistenci dat.

5.1 Ukládání dat

Data jsou ukládána několika způsoby za prvé se využívá databáze do které se ukládají požadavky klienta na výsledné PDF. Dále se používají již zmíněné properties soubory v kterých se nachází tokeny. A v neposlední řadě se všechny vygenerované PDF ukládají do klasického souborového systému.

VLOZIT OBR SLOZEK

5.1.1 SQLite

Tato databáze staví na klasickém SQL jazyku a stejně jako všechny ostatní nejznámější databáze i tato je relačním. Ovšem narozdíl od ostatních nepotřebuje ke svému běhu server, pouze vytváří databáze v podobě souborů, ke kterým poskytuje rozhraní.

5.2 Kompilace a úprava PDF

5.3 Zpracovávání požadavků

Kapitola 6

Závěr

Cíle, které byly stanoveny v úvodu, byly dosaženy. Po seznámení s technologiemi a teorií relevantní pro téma této práce proběhla analýza.

V analýze byly pomocí metody FURPS+ určeny požadavky vyplývající z potřeb a omezení stanovených zadávajícím či prostředím. Dále byly zanalyzovány vybrané existující řešení téhož problému, ovšem žádné z nich nesplnilo všechny požadavky. Také došlo na analýzu a porovnání vybraných kompilátorů, ze které vzešlo, že výběr je malý a rozdíly mezi nimi minimální.

Na analýzu je navázáno návrhem vlastního řešení, v této části je popsáno, jaké komponenty bude aplikace obsahovat a jak bude fungovat.

Platformou pro implementaci bude Java EE s aplikačním serverem Payara. Byl navrhnut způsob komunikace a komunikační protokol mezi službou a klienty pomocí REST Api, včetně autentizace klientů, která bude prováděna na základě tokenů. Nejdůležitější částí aplikace bude LaTeX kompilátor, přičemž ním byl zvolen MikTeX. Jeden z hlavních důvodu je možnost stahovat balíčky za běhu, které se bude využívat. Výsledné PDF soubory se budou uchovávat na serveru po dobu jednoho měsíce v chráněném souborovém systému.

Na základě zvoleného kompilátoru a požadavků byl vybrán operační systém Debian 9, který musí být přítomen na serveru pro správné fungování aplikace. Aby služba mohla vyhovět všem požadavkům a provozu je požadováno 15GB volného místa na disku pro instalaci, balíčky a uchovávání výsledných PDF.

Na tento semestrální projekt bude navázáno implementací, která bude vycházet ze zde dosažených závěru a výstupů.

Literatura

[1] Document management - portable document format - part 1: Pdf 1.7. `https://www.adobe.com/content/dam/acom/en/devnet/pdf/PDF32000_2008.pdf`, 2008.

[2] Dutoit, B. B. . A. H. *Object-Oriented Software Engineering Using UML, Patterns, and Java*. 2009.

[3] Fielding, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, PhD Dissertation, University of California, Irvine, 2000. Online, Přístup 17.11.2018.

[4] Hesam, Masne, S., Harry, and Rajan. REST API Tutorial. `https://restfulapi.net/`. Online, Přístup 17.11.2018.

[5] Satrapa, P. *LaTeX pro pragmatiky*. 2011.

Příloha A

Seznam zdrojů

- Obrázek 2.1 – Převzato z dokumentace PDF[1] 25. 11. 2018.
- Obrázek 4.2 – Vytvořeno autorem 30. 12. 2018.

Příloha B

Seznam zkratek

FURPS – Functionality, Usability, Reliability, Performance, Supportability

HTTP – Hypertext Transfer Protocol

API – Aplikační rozhraní

CW – CourseWare

TEX – Typografický sázecí systém

LATEX – Lamport TEX - balík maker pro TEX

PDF – Portable Document Format

REST – Representional State Transfer

GB – Gigabyte

MB – Megabyte

GPL – GNU Genereal Pulbic License

OS – Operační systém

URL – Uniforme Resource Locator

SQL – Structured Query Language