

Bakalářská práce



**České
vysoké
učení technické
v Praze**

F3

**Fakulta elektrotechnická
Katedra počítačů**

REST služba 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 a implementaci webové služby pro konverzi \LaTeX 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 and implementation 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

1 Úvod	1
1.1 Cíle práce	1
2 Teorie a technologie	3
2.1 REST	3
2.2 Webová služba	4
2.3 PDF	4
2.4 LaTeX	5
2.5 Moodle a CourseWare	6
2.6 FURPS+	6
3 Analýza	8
3.1 Požadavky	8
3.2 Existující řešení	10
3.3 Kompilátory	11
4 Návrh vlastního řešení	13
4.1 Platforma.....	13
4.2 Architektura	13
4.3 Databáze a entitní model	14
4.4 Komunikace.....	15
4.5 Zabezpečení.....	16
4.6 Kompilace	17
4.7 PDF úpravy	17
4.8 Server	17
5 Implementace	19
5.1 Projekt	19
5.2 Aplikační server	22
5.3 Zpracovávání požadavků	22
5.4 Kompilace a úprava PDF	23
5.5 Ukládání dat	25
6 Testování	27
6.1 REST rozhraní	27
6.2 Jednotkové testy.....	27
7 Závěr	28
Literatura	29
A Seznam zdrojů	30
B Seznam zkratk	31

Obrázky

2.1 Struktura objektů	5
4.1 Klient server diagram	14
4.2 Diagram vrstev	14
4.3 Entitní diagram	15
4.4 Sekvenční diagram	16
4.5 Diagram nassazení	18
5.1 Diagram balíčků	20
5.2 Diagram tříd balíčku Responders	21
5.3 Diagram tříd týkající se balíčku Processes	22
5.4 Adresářový strom	25

Tabulky

3.1 Funkční požadavky	8
3.2 Nefunkční požadavky	9

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 několik částí, které je potřeba splnit k úspěšnému dokončení projektu. 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 na základě něhož bude naimplementována. Nakonec bude služba otestována a zhodnocena.

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 existující 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:

- Naimplementovat funkční aplikaci
- Otestovat aplikaci

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

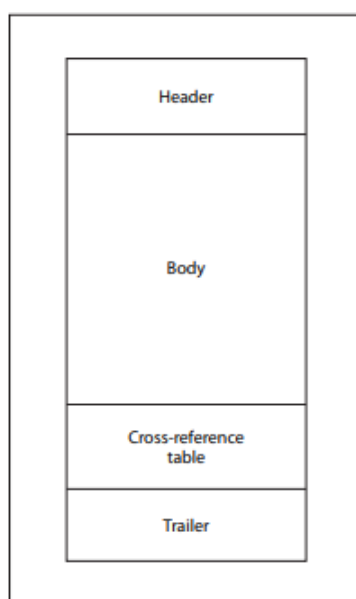
Representational state transfer (dále jen REST) je architektura pro komunikaci mezi distribuovanými systémy. Termín zavedl R. T. Fielding ve své disertační práci[4], kde toto rozhraní bylo taktéž popsáno a vymezeno. Mimo jiné stanovil i těchto 5 základních pravidel, která by měla být dodržena, aby se aplikace nazývala RESTful[6]:

- Klient-Server (Client-Server) - Toto omezení staví na principu oddělení zodpovědností (Separation of Concerns), nebo-li existuje klientská část starající se o uživatelské rozhraní a ta je oddělena od serverové části přistupující k databázi. Zlepšuje škálovatelnost systému a zjednodušuje použitelnost na různých platformách.
- Bezstavovost (Stateless) - Každý požadavek musí přenášet všechna související data, server totiž neuchovává žádné informace o nynějším spojení a každý požadavek bere jako nový.
- Keš (Cache) - Data přenášená v odpovědi mohou být označena 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) a 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

Vzhledem k povaze této práce je potřeba také zmínit systém práv a povolení. Existují dva typy práv, která jsou rozlišeny na základě poskytnutého hesla. Buď je zadáno uživatelské heslo, které umožní uživateli jenom otevřít daný soubor. Nebo heslo vlastníka, díky kterému je možno upravovat uživatelské heslo a jeho práva k manipulaci se souborem. Jde povolit tyto možnosti:

- Upravování obsahu dokumentu
- Kopírování textu a obrázků
- Vyplňování formulářů a polí
- Tisknutí dokumentu



Obrázek 2.1: Struktura objektů

■ 2.4 LaTeX

Vychází z typografického sázecího systému \TeX , který popisuje Pavel Satrapa ve své knize [7]: „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 \TeX u 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 jinými nástroji ke psaní náročnější.

■ 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 využíváno 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	<ul style="list-style-type: none">• 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:<ul style="list-style-type: none">• 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 má jediný a specifický účel, z toho plyne menší množství funkcionalit.

3.1.2 Nefunkční požadavky

Typ	Požadavky
Použitelnost	<ul style="list-style-type: none"> Dokumentace k rozhraní na swagger ¹ Přístup skrz REST Api, pouze s tokenem²
Spolehlivost	<ul style="list-style-type: none"> Služba není kritická Uptime 95% času Ochrana proti špatnému vstupu Autorizace pomocí tokenů
Výkon	<ul style="list-style-type: none"> Zvládnutí obslužení desítek požadavků najednou Ukládání výsledných dokumentů po dobu jednoho měsíce Maximální doba tvorby dokumentu 5 minut
Podporovatelnost	<ul style="list-style-type: none"> Služba může být rozšířena o kompilování samotného TeXu a může podporovat vytváření i jiných formátů z L^AT_EXu
Implementace	<ul style="list-style-type: none"> Platforma - Java EE Komunikace - REST Api Operační systém - Debian nebo CentOS
Rozhraní	<ul style="list-style-type: none"> Komunikuje s Moodle a CourseWare, tyto portály posílají data
Fyzické	<ul style="list-style-type: none"> Musí být provozováno na serverech ČVUT

Tabulka 3.2: Nefunkční požadavky

Důvody 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 služba³ 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 Commons⁴. 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 služba⁵ 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 služba⁶, 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-source⁷ 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 Pro⁸, 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 L^AT_EX 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 L^AT_EX 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 L^AT_EX 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

MikTeX⁹ 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 nejnútnejší, 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/cls-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 TeXLive¹⁰ 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 nejn nutnější. 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

Návrh vlastního řešení

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

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. Proto data budou ukládány do souborového systému.

4.1 Platforma

Na základě požadavků bude aplikace postavena na Java EE[2], 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.

4.2 Architektura

Architektura[5] popisuje z jakých částí se aplikace skládá, jak mezi sebou tyto části komunikují a jakým způsobem procházejí informace a požadavky aplikací. Při navrhování architektury je potřeba zvážit několik věcí, například rozšiřitelnost, modularizaci, složitost a pro jaký případ ma sloužit.

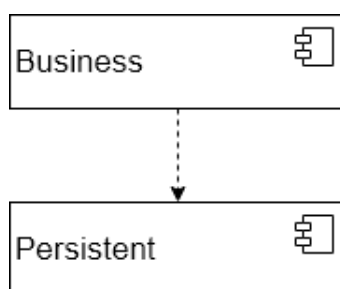
Nejpoužívanější architekturou u webových aplikací je klient-server architektura s n vrstvami, která aplikaci rozděluje na n fyzických médií a n vrstvá architektura, která

označuje n logických celků.¹ Nejčastěji jsou obě architektury 3 vrstvé, tedy rozdělení je následující: prezentační, aplikační a databázová vrstva. Ovšem služba, kterou se zabýváme v této práci není plnohodnotná webová aplikace, tudíž nemá uživatelské rozhraní a vystavuje jenom určité API, odpadá tedy prezentační vrstva. Ani neobsahuje databázový server, ve výsledku zůstává jenom aplikační fyzická vrstva, což z naší služby dělá jedno-vrstvou klient-server aplikaci 4.1. Zbývá ještě vyřešit logickou architekturu, kde se budeme držet také zavedých postupů, ale zbavíme se prezentační vrstvy a zůstane nám 2-vrstvá architektura obsahující byznys logiku a přístup k databázi 4.2.



Obrázek 4.1: Klient server diagram

To znamená, že program přijímá požadavek od klienta, podle typu ho zpracuje v byznys vrstvě a případně předá datábazové pro persistentní uložení. Byznys vrstva bude obsahovat celou logiku aplikace, mimo jiné kompilaci a úpravu PDF.



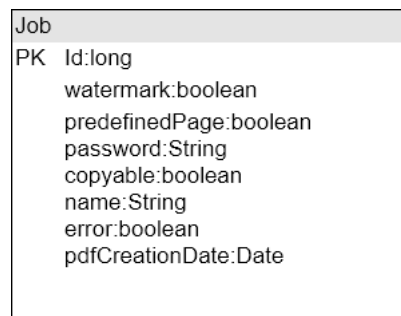
Obrázek 4.2: Diagram vrstev

4.3 Databáze a entitní model

Jak bylo řečeno jedná se o jednoduchou aplikaci, tudíž není zapotřebí složité databáze s vlastním serverem. Vystačíme si s SQLite², což je velmi odlehčená verze klasických SQL databází, jejíž databáze je jenom jednoduchý soubor na disku, do kterého zapisuje napřímo. Na základě dostupných informací z požadavků a cíle služby víme, že bude stačit jediná tabulka, která bude držet hlavně informace o požadavcích klienta. Dále také jméno hlavního tex souboru, které je nutné pro kompilaci a primární klíč id, který slouží pro identifikaci. Mimo jiné také obsahuje vnitřní stavy průběhu zpracování L^AT_EX souborů. Všechny atributy jsou vypsány v diagramu 4.3, který je vyobrazen níže.

¹V anglických zdrojích se používají výrazy N-Tier Client-Server architecture a N-Layered architecture, kde tier znamená to samé jako layer, tudíž vrstva

²<https://www.sqlite.org/>



Obrázek 4.3: Entitní diagram

4.4 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 4.4 je zobrazena komunikace mezi serverem a klientem.

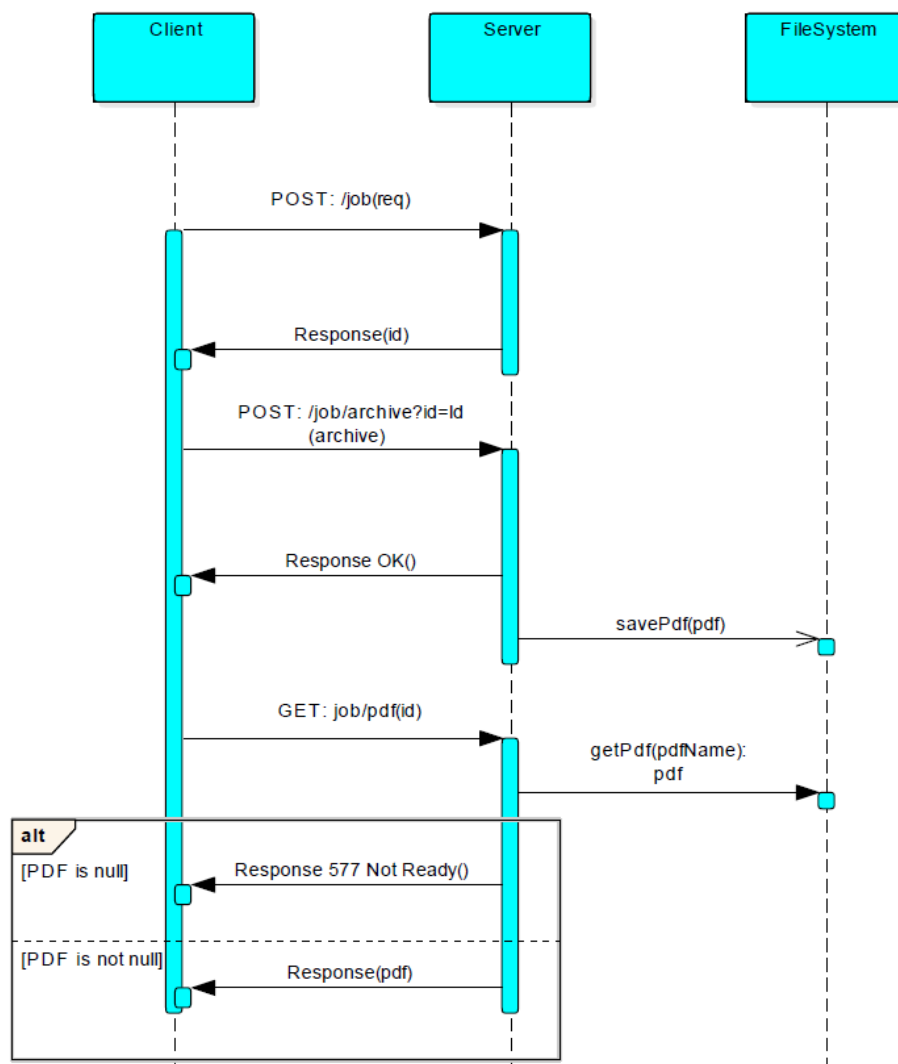
Kompletní dokumentace k rozhraní se nachází na [swagger³](https://app.swaggerhub.com/apis/Tadky/Thesis/1). Na ukázkou vypíšeme alespoň používané zdroje:

- **POST** /job - Slouží k předání požadavků na výsledné PDF.
- **POST** /job/archive?id=<jobId> - Musí obsahovat zip archiv s potřebnými soubory pro kompilaci. Jméno archivu musí být stejné jako hlavního textu.
- **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
- **GET** /info - Slouží k získání informací ohledně serveru, tedy verze kompilátoru, operačního systému, zbývajících volného místa na disku

Všechny tyto zdroje navíc obsahují parametr⁴ token, který zajišťuje autentizaci. Ta je rozebrána v další sekci.

³<https://app.swaggerhub.com/apis/Tadky/Thesis/1>

⁴Typ parametru, který se posílá v samotné URL daného dotazu ve formě klíč=hodnota



Obrázek 4.4: Sekvenční diagram

4.5 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 OAuth⁵ 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 uloženému v properties⁶ souboru. Token bude posílán v parametru dotazu v otevřené formě.

⁵Protokol pro autentizaci a autorizaci aplikací.

⁶Soubor v němž jsou data uspořádány klíč = hodnota

4.6 Kompilace

Pro vytvoření PDF je nutné příslušně zkompileovat 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 tudíž je může odrazovat od používání aplikace. 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.

4.7 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 podle požadavků. Nejvíce vhodný se zdá PDFtk⁷, který je pod GPL⁸ licencí. Nabízí mnoho možností úprav, pro naše použití jsou nejdůležitější tyto: spojení dvou PDF souborů do jednoho, přidání vodoznaku, zaheslování souboru a omezení práv na něm. Tedy po zkompileování do PDF se za pomoci výše zmíněné aplikace aplikují požadavky, které byly obdrženy od klienta.

4.8 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 Ubuntu, 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

⁷<https://www.pdfabs.com/tools/pdftk-server/>

⁸Poskytuje svobodu šíření, provozování a upravování daného software.

Kapitola 5

Implementace

5.1 Projekt

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.

Pro sestavení projektu je využit Maven, který nabízí jednoduché inkludování externích balíčků a také pluginy pro sestavování a nasazování aplikace.

5.1.1 Java EE

Celý projekt je naimplementován v Javě EE verze 8. Ta obsahuje nové DateTime API, Streams API aj., dále například podporuje použití lambda výrazů, což zkracuje zápis a umožňuje například předávání funkcí. Tato funkcionality byla využita při přijímání požadavků 5.1 a kontrolování tokenů 5.2. Dalším důležitým aspektem Javy EE jsou beans, které už byly představeny v sekci 4.1 na tyto informace navážeme jejich anotacemi, které byly použity v rámci projektu. Většina bean je typu stateless, což znamená že si nepamatují stav napříč požadavky jednoho klienta, jenom po dobu jednoho daného požadavku, poté je objekt vrácen do poolu a může ho využít jiný klient. Dále je použita request scope beana, která je vytvořena s každým novým http požadavkem a není zničena dokud je požadavek aktivní. Objekt *Starter*, který je použit pro prvotní procesy po naběhnutí aplikace a objekt *PeriodicalCleaning*, který periodicky promazává nepotřebné soubory jsou typu Singleton. Což je beana, která je vytvořena jenom jednou a je globálně sdílená pro celou aplikaci. Byly využity i jiné anotace a funkce, ale pro představu tyto stačí.

5.1.2 Struktura projektu

Nyní si představíme strukturu projektu, tedy diagram balíčků, který zobrazuje tři hlavní balíčky. První se jmenuje Api, který slouží pro zpracování požadavků (detailněji popsáno v sekci 5.3), ten obsahuje tři další složky jmenovitě:

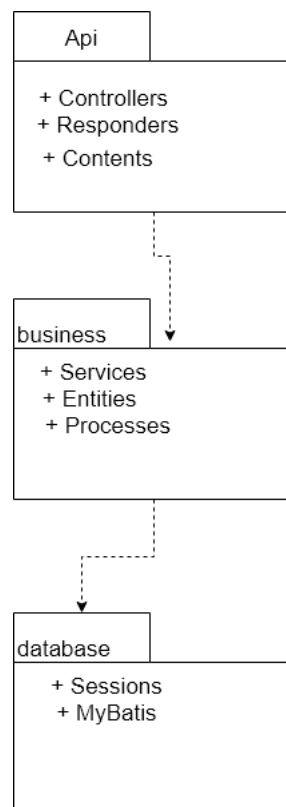
- Controllers - Třídy zdrojů obsahující metody pro přijímání požadavků
- Responders - Třídy, které se starají o sestavování odpovědí klientům
- Contents - Třídy představující objekty pro serializaci a deserializaci JSON formátu

Další balíček v pořadí je Business, ten představuje všechnu logiku, která je aplikována na data. Nejdůležitějším balíčkem je Processes, ale zase si popíšeme všechny.

- Services - Třídy obsahující hlavní logiku, tedy rozhodují co dělat s příchozími daty
- Processes - Třídy pro kompilaci a úpravu PDF
- Entities - Třídy představující objekty používání v aplikaci

Poslední se jmenuje Database a obsahuje tyto složky:

- Sessions - Třídy pro komunikaci s databází
- MyBatis - Třídy pro mapování SQL dotazů a nastavování spojení



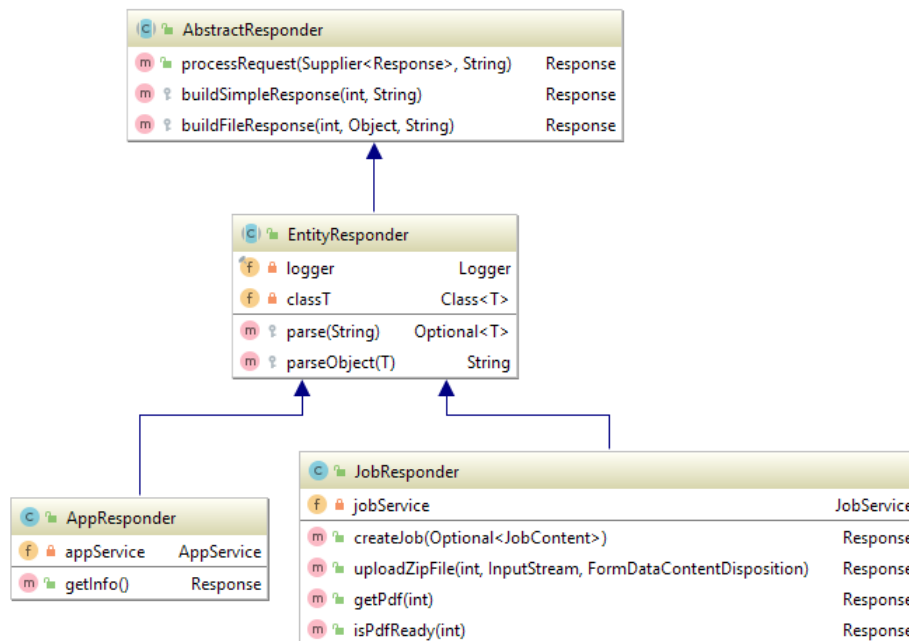
Obrázek 5.1: Diagram balíčků

Projekt obsahuje i další nevyobrazené balíčky, ale ty jsou spíše pomocné a tudíž nejsou tak důležité z pohledu funkčnosti a průchodu aplikací např. Utils, Exceptions, Enums, atd.

■ 5.1.3 Diagram tříd

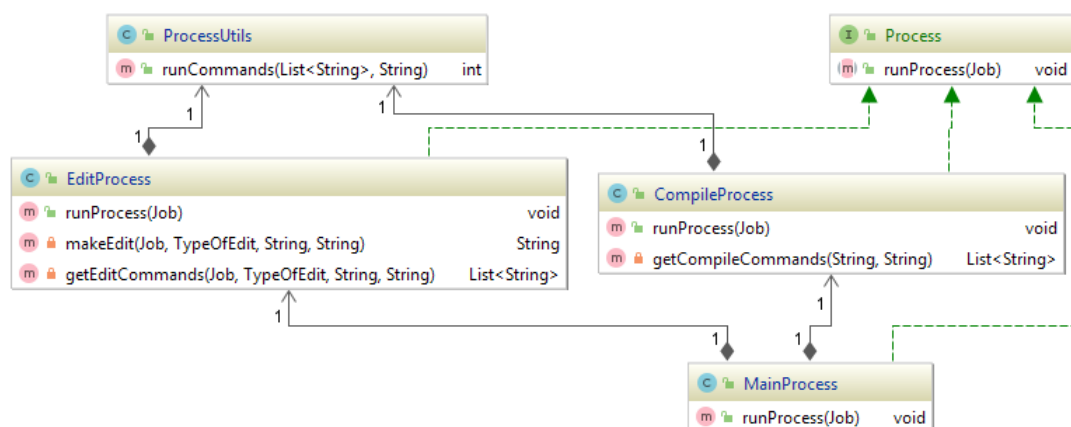
Na ukázkou byly vytvořeny dva diagramy tříd, ty popisují jejich provázanost. První diagram 5.2 ukazuje hierarchii, kde úplně nahoře je obecná abstraktní třída, která má

například metodu *processRequest*, která je popsána v sekci 5.3. Pod ní je třída *EntityResponder*, která je také abstraktní a navíc generická, ta představuje už konkrétnější skupinu s metodami pro parsování JSON do objektů daných typem a naopak. Od ní dědí v našem případě dvě třídy, *JobResponder* vytváří odpovědi pro žádosti na adresu /job a *AppResponder* na adrese /app.



Obrázek 5.2: Diagram tříd balíčku Responders

Na druhém diagramu 5.3 je vidět jaké třídy se hlavně starají o kompilaci a editaci. Všechny kromě *ProcessUtils*, která jenom poskytuje metodu pro samotnou exekuci příkazů, implementují rozhraní *Process* s metodou *runProcess*. Ta je jediná která by měla být v jednotlivých třídách volána, představuje proběhnutí požadovaného procesu se vším všudy. V našem případě je nejdříve volána metoda z třídy *MainProcess* a ta následně zajistí kompilaci a úpravu za pomoci vyobrazených tříd *CompileProcess* a *EditProcess*.



Obrázek 5.3: Diagram tříd týkající se balíčku Processes

5.2 Aplikační server

V této sekci hlavně rozebereme nastavení serveru. Je použita Payara ve verzi 191 Full, ale dále zmiňované nastavení není nijak závislé na verzi. Bohužel jsem nebyl schopen nastavit JDBC pool v Payaře, což ale bylo vyřešeno jiným způsobem popsáním v sekci 5.5.1. Byly přidány čtyři JNDI zdroje pro nastavení cesty k hlavnímu adresáři, cesty k souboru s databází a cest ke spustitelným souborům programů MikTeX a PDFtk. Nic jiného k úspěšnému spuštění aplikace není potřeba.

5.3 Zpracovávání požadavků

Požadavky klienta jsou přijímány skrze REST Api, jak můžeme vidět na ukázce 5.1, kde je vyobrazena funkce, která reaguje na http požadavek typu POST na základní adrese /job. Přijímá data ve formátu JSON, které obsahují požadavky na výsledné PDF, ale také token, používaný k autentizaci.

```

@POST
@Consumes(MediaType.APPLICATION_JSON)
public Response postJob(@QueryParam("token") String token, String data) {
    return jobResponder.processRequest(()
        -> jobResponder.createJob(data), token);
}

```

Ukázka kódu 5.1: Funkce pro přijímání HTTP požadavků

Token je zkontrolován ve funkci *processRequest* 5.2, ta kromě tokenu přijímá i parametr *Supplier<Response>*, který představuje obecnou funkci bez parametru s návratovou hodnotou typu *Response*. Tudíž se jako první zkontroluje daný token, pokud je nesprávný je rovnou vrácena odpověď *UNAUTHORIZED* v opačném případě je zavolána předaná funkce pomocí metody *get()*, kterou definuje interface *Supplier*.

```
public Response processRequest(Supplier<Response> request, String token) {
    Response response = null;
    if (TokenUtils.checkToken(token)) {
        response = request.get();
    } else {
        response = buildSimpleResponse(401, "Bad_token");
    }
    return response;
}
```

Ukázka kódu 5.2: Funkce na ověření tokenu v požadavku

Následuje objekt typu Responder, která se stará o sestavování odpovědí klientovi na základě výjimek či výstupů ze servisní vrstvy. Ta obsahuje logiku aplikace, na základě typu požadavku, buď data persistentně uloží nebo například započne hlavní proces služby, tedy kompilaci. O ukládání dat se starají Session objekty, které si získají spojení s databází a vykonají daný SQL dotaz.

5.4 Kompilace a úprava PDF

Jak bylo uvedeno v návrhu (sekce 4.6), pro kompilování L^AT_EX souborů je vybrán **MikTeX**, ten je potřeba nejdříve nainstalovat v příslušné konfiguraci, která byla popsána v téže sekci, vybraná verze je 2.9.7. Samotná kompilace je prováděna pomocí PdfLatex, ten je spouštěn s těmito parametry:

- interaction = nonstopmode - program nečeká na vstup
- include-directory - slouží k určení adresáře, kde se nachází tex soubory
- output-directory - adresář pro výstupní PDF soubor
- aux-directory - adresář pro ostatní výstupní soubory, jako jsou logy
- jméno hlavního tex souboru

Výstupem je tedy soubor PDF, na který je potřeba ještě aplikovat požadavky od klienta. K tomu je použit program **PDFtk** verze 2.02. Pro požadované úpravy jsou potřeba tyto argumenty:

- output - určuje výstupní adresář
- cat - spojuje dva PDF soubory dohromady
- background - přidává vodoznak
- owner_pw a user_pw - první nastavuje heslo vlastníka a druhý heslo uživatele¹
- allow copycontents - umožňuje kopírovat text
- allow printing - povoluje tisk

¹Rozdíl je popsán v kapitole 2.3

Jelikož se argumenty kromě output, allow copycontents a printing nedají řetězit, je každá úprava vykonávána samostatně. Pro lepší představu následuje ukázka právě zaheslování a přidání práv pro kopírování a tisk.

```
commands.addAll(Arrays.asList("output", directoryPath + outputFileDir);
commands.addAll(Arrays.asList("owner_pw", job.getPassword()));
commands.addAll(Arrays.asList("user_pw", String.valueOf(job.hashCode())));
if (job.isCopyable() || job.isPrintable()) {
    commands.add("allow");
}
if (job.isCopyable()) {
    commands.add("copycontents");
}
if (job.isPrintable()) {
    commands.add("printing");
}
```

Ukázka kódu 5.3: Zaheslování a úprava práv souboru PDF

Můžeme vidět, že nejdříve se nastaví argument output, dále se nastavuje heslo vlastníka, to byl přijmuto od klienta, naopak jako heslo uživatele se použije vygenerovaný hash, který zajišťuje dostatečnou bezpečnost a náhodnost. Jestliže si uživatel přál kopírovatelnost a tisknutelnost je přidán argument allow a za ním dané povolení, buď jedno nebo obě.

■ 5.4.1 Vykonávání příkazů

Java poskytuje rozhraní pro spouštění procesů a zacházení s nimi. Nejdříve je potřeba ProcessBuilderu předat pole stringů, kde první je název nebo cesta ke spustitelnému souboru a ostatní jsou argumenty daného příkazu. Zavoláním metody start na právě vytvořeném objektu, získáme objekt Process, který už představuje běžící proces definovaný zadanými argumenty. Nyní je možné číst výstupy (normální i chybový) programu, ale je i umožněno psaní na vstup. Metoda *waitFor* čeká na dokončení procesu a její návratová hodnota udává zda-li byl program ukončen chybou nebo normálním způsobem.

```
public int runCommands(List<String> commands, String logName) {
    Process process = new ProcessBuilder(commands).start();
    InputStream is = process.getInputStream();
    BufferedReader reader = new BufferedReader(new InputStreamReader(is));

    String logFile = directoryPath + DirUtils.LOG_DIR + logName + ".txt";
    BufferedWriter log = new BufferedWriter(new FileWriter(logFile));

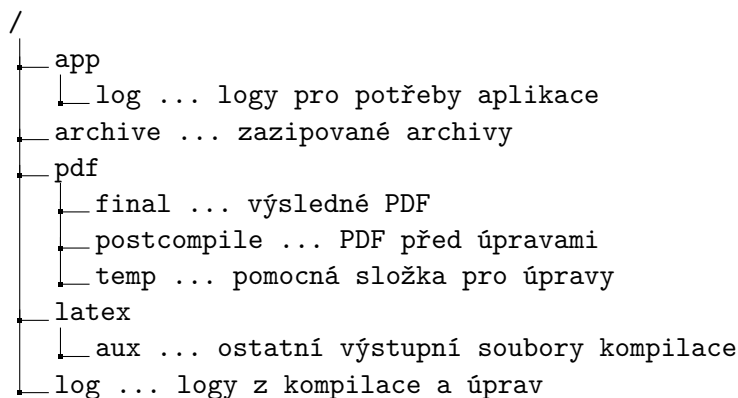
    String line;
    while ((line = reader.readLine()) != null) {
        log.write(line + System.lineSeparator());
    }
    log.flush();
    log.close();
    return process.waitFor();
}
```

Ukázka kódu 5.4: Metoda pro spuštění procesu

Tato ukázka kódu představuje zmiňované zacházení s procesy používané pro kompilaci a úpravu do PDF, kde výstup programu je vypisován do textového souboru, pro budoucí použití.

5.5 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.



Obrázek 5.4: Adresářový strom

Na obrázku 5.4 můžeme vidět adresářovou strukturu s popisem k čemu dané složky slouží.

5.5.1 SQLite a MyBatis

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 na disku, ke kterým poskytuje rozhraní knihovna SQLite JDBC. Použitá verze JDBC v této práci je 3.27.2.1. Tato databáze má velmi omezený počet typů například neobsahuje boolean, což bylo vyřešeno pomocí mapování frameworkem MyBatis², který je popsán o odstavec níže. Pomohl také k vyřešení problému ohledně JDBC poolu, který nešel nastavit v aplikačním serveru.

MyBatis. Je knihovna pomocí, která umožňuje mapování databáze na entity, předdefinovat si SQL dotazy a komunikovat se samotnou databází. Je použita ve verzi 3.5.0. Hlavní výhodou je velká customizace, která je potřeba při zacházení s SQLite, jak už bylo napsáno výše, bylo vytvořeno mapování z typu boolean na typ integer a naopak.

²<http://www.mybatis.org/mybatis-3/>

Dále dokáže spravovat spojení s databází, je možné nastavit pooling atd. Nastavení mapování entit a dotazů a samotná konfigurace se píše do XML souborů. Kde byl například nastaven již zmíněný JDBC pool, to vše je vidět v ukázce níže 5.5.

```
<environment id="development">
  <transactionManager type="JDBC"/>
  <dataSource type="POOLED">
    <property name="driver" value="org.sqlite.JDBC"/>
    <property name="url" value="${url}"/>
  </dataSource>
</environment>
```

Ukázka kódu 5.5: Konfigurace JDBC poolu

Kapitola 6

Testování

Testování této aplikace je rozděleno na dvě části. Jelikož se jedná o službu, která poskytuje akorát restové rozhraní, tak první část bude obsahovat ruční otestování pomocí aplikace Postman ¹. Druhá část

6.1 REST rozhraní

K tomuto testování byla použita knihovna Rest Assured, která nabízí jednoduché prostředí pro vytváření HTTP požadavků a kontrolování správnosti odpovědí. Testy byly vytvořeny pro všechny zdroje.

```
@Test
void invalidToken(){
    String badToken = "asdfasdf";
    RestAssured.given().queryParam("token",badToken)
        .contentType(ContentType.JSON).body("test")
        .then().expect().statusCode(401).when().post(context);
}
```

Ukázka kódu 6.1: Požadavek se špatným tokenem

6.2

¹<https://www.getpostman.com/>

Kapitola 7

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 důvodů 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 návrh bylo navázáno samotnou implementací



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] The java ee 6 tutorial. <https://docs.oracle.com/javaee/6/tutorial/doc/javaeetutorial6.pdf>, 2013.
- [3] DUTOIT, B. B. . A. H. *Object-Oriented Software Engineering Using UML, Patterns, and Java*. 2009.
- [4] 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.
- [5] FOWLER, M. *Patterns of Enterprise Application Architecture*. 2011.
- [6] HESAM, MASNE, S., HARRY, AND RAJAN. REST API Tutorial. <https://restfulapi.net/>. Online, Přístup 17.11.2018.
- [7] SATRAPA, P. *LaTeX pro pragmatiky*. 2011.
- [8] W3C. Web service architecture. <https://www.w3.org/TR/ws-arch/wsa.pdf>, 2004.



Příloha A

Seznam zdrojů

- Obrázek 2.1 – Převzato z dokumentace PDF[1] 25. 11. 2018.
- Obrázek 4.4 – Vytvořeno autorem 30. 12. 2018.
-



Příloha B

Seznam zkratek

FURPS – Functionality, Usability, Reliability, Performance, Supportability

HTTP – Hypertext Transfer Protocol

API – Aplikační rozhraní

CW – CourseWare

T_EX – Typografický sázecí systém

L^AT_EX – Lamport T_EX - balík maker pro T_EX

PDF – Portable Document Format

REST – Representational State Transfer

GB – Gigabyte

MB – Megabyte

GPL – GNU Genereal Pulbic License

OS – Operační systém

URL – Uniforme Resource Locator

SQL – Structured Query Language

JDBC – Java Database Connectivity