Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

SAGELab – Správa multimediálních materiálů na cloud úložišti

Roman Svoboda

Vedoucí práce: Ing. Jiří Melnikov

11. května 2015

Poděkování

Tímto bych chtěl poděkovat ***Ing. Jiřímu Melnikovi*** za svědomité vedení mé bakalářské práce, odborný dohled a cenné rady, které mi v době vypracovávání poskytl. Stejně tak bych chtěl poděkovat svému oponentovi ***Ing. Jiřímu Chludilovi*** za ochotu a čas dělat mi oponenta.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 11. května 2015 .....

Odkaz na tuto práci

Svoboda, Roman. *SAGELab – Správa multimediálních materiálů na cloud úložišti*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Cílem této bakalářské práce je navrhnout, implementovat a otestovat backend do týmově vyvíjené webové aplikace, na jejíž implementaci nyní spolupracuje několik bakalantů. Aplikace je zaměřená na přípravu prezentace multimediálního obsahu pro zařízení SAGE (stěna dělených displejů s rozlišením 9600 × 4320 pixelů na FIT ČVUT). Backend implementovaný jako RESTful Web API bude poskytovat rozhraní pro upload a správu multimediálních datových souborů v rámci vybraného cloudového úložiště.

V první kapitole práce je čtenář seznámen s teorií Cloud computingu, ve druhé kapitole je popsána řešená problematika, a to včetně nastínění možností jejího řešení. Třetí kapitola obsahuje analytickou část práce, která zahrnuje zejména analýzu funkčních a nefunkčních požadavků a také rešerši cloudových softwarů a služeb. Ve čtvrté kapitole jsou vysvětleny implementační technologie a architektury, kterých bylo využito. Pátá kapitola popisuje celkový průběh realizace a šestá líčí způsob testování implementace, včetně zhodnocení výsledků a procesu testování. V závěru jsou souhrnně vyhodnoceny dosažené cíle a je zde nastíněna možnost dalšího pokračování v této práci.

Klíčová slova: SAGE^{TM}, SAGELab, Web API, REST, cloud.

Abstract

The aim of this work is to design, implement and test a backend for a team-developed web application which is being implemented by a few bachelors. The application is aimed to prepare presentation of multimedia content on SAGE devices (wall split screens with a resolution of 9600 × 4320 pixels at FIT CTU). Backend, implemented as a RESTful Web API, will provide an interface to upload and manage multimedia data files within the selected cloud storage.

The purpose of the first chapter is to familiarize the reader with the theory of Cloud computing, the second chapter describes the solution of the issue, including outlining the options for its solution. The third chapter contains the analytical part of the work, which includes analysis of functional and non-functional requirements, as well as research of cloud software and services. The fourth chapter explains the implementation of technology and architecture, which has been used. The fifth chapter describes the overall progress of the implementation and the sixth chapter describes method of testing implementation, including the results of the evaluation and testing process. In the end are evaluated the achievements and there is outlined the possibility of continuing this work.

Keywords: SAGE^{TM}, SAGELab, Web API, REST, cloud.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V dnešní digitální době je velmi podstatné rozlišení zobrazovacích zařízení, na kterých vizualizujeme a zpracováváme veškerý multimediální obsah. Standard velikosti rozlišení za posledních pár let prošel významným vývojem. Pro koncové uživatele totiž začalo hrát rozlišení důležitou roli při výběru/koupi nového multimediálního zařízení. Výrobci smartphonů, televizorů, monitorů, tabletů a počítačů se tomu museli samozřejmě přizpůsobit, a proto je Full HD[1] rozlišení naprostým standardem na dnešním trhu a lze rovněž předpokládat, že trend zvyšování velikosti rozlišení bude i v dalších letech pokračovat. Výrobci nyní na trh uvádějí první zobrazovací zařízení se 4K[1] rozlišením a v dohledné budoucnosti lze očekávat například 8K[1] rozlišení.

V některých specifických odvětvích dnešní standard rozlišení nemusí při zpracování digitálního obsahu bohužel dostačovat. Příkladem může být zdravotnictví, letectví, grafická a televizní studia, akademická půda atd. Z tohoto důvodu se realizují velkoplošná vizualizační zařízení s velmi vysokým celkovým rozlišením. Tato zařízení se technicky uskutečňují například spojením mnoha zobrazovacích zařízení v jeden celek, který se pak chová jako celistvý zobrazovací mechanismus. Nicméně atribut „chovat se při zobrazování celistvě“ je jeden z největších problémů, který je nutno řešit pomocí speciálního softwaru, jehož funkcí je všechna dílčí zobrazovací zařízení synchronizovat a ovládat.

Stoupající standard celkového rozlišení s sebou bohužel přináší i podstatné zvýšení velikosti multimediálního obsahu. Důsledkem toho se zvyšují nároky na souhrnnou kapacitu úložiště, na kterém jsou data o vysokém rozlišení uložena. Proto se často začíná využívat přístup, který je založen na principu, že data jsou fyzicky oddělena od systému, který je zpracovává, a to do nějakého virtuálního úložiště (cloudu), jež poskytuje snadné rozšíření kapacity nezávisle na původním systému.

[1]Full HD = rozlišení o velikost 1920 na 1080 pixelů; 4K = rozlišení o velikost 3840 na 2160 pixelů; 8K = rozlišení o velikost 7680 na 4320 pixelů.

Naším cílem společně s několika dalšími spolupracujícími bakalanty je v rámci většího projektu vyvinout webovou aplikaci, která by umožnila snadnou přípravu multimediálního obsahu a jeho následnou prezentaci pomocí velkoplošného vizualizačního zařízení SAGE (stěna 20 dělených displejů s rozlišením 9600 × 4320 pixelů umístěná na FIT ČVUT). Aplikace by zároveň řešila uložení datových souborů na cloudové úložiště. Souhrnně by se jednalo o softwarovou vrstvu/aplikaci fungující nad již zmíněným speciálním softwarem, který zajišťuje ovládání velkoplošných vizualizačních zařízení.

Tématem mé bakalářské práce je zabývat se problematikou možnosti oddělení datových souborů do vhodně zvoleného cloudového úložiště. Nedílnou součástí práce je také implementace backendu formou webové služby (Web API) pro výše uvedenou vyvíjenou webovou aplikaci na přípravu prezentace multimediálního obsahu pro zařízení SAGE. Backend bude nabízet rozhraní pro správu materiálů v rámci vytvořeného cloudového úložiště. Konkrétně se bude zaměřovat na upload a správu datových souborů na cloudovém serveru, jejich konverzi (tvorba náhledů z video souborů, tvorba náhledů z PDF dokumentů) a změnu kvality obrazových souborů (tvorbu zmenšenin).

Kapitola 1

Úvod do Cloud computingu

Kapitola se věnuje problematice Cloud computingu v rozsahu definování tohoto pojmu, historie a nakonec jeho základního rozdělení. Jelikož se klíčové slovo „***cloud***“ v práci objevuje poměrně často, je vhodné, aby čtenář byl s touto oblastí obeznámen.

1.1 Definice Cloud computingu

Otázka „*Co je to Cloud computing?*“ zní dle prvního dojmu velmi jednoduše, avšak po zevrubnějším prozkoumání problematiky lze zjistit, že zdání bohužel klame. Oblast Cloud computingu (dále také jako CC) je zatím velmi mladá a právě nyní prochází velmi bouřlivým vývojem, tudíž není jednoduché vymezit jedinou platnou definici CC. Situaci nepřispívá ani fakt, že mnoho IT odborníků, firem a institucí se pokouší vykládat pojem CC po svém.

Důsledkem těchto faktů je, že zřejmá a všemi akceptovaná definice zatím neexistuje. První českou definici formuloval technický ředitel firmy Abakowiki pan Jan Koděra a zní následovně: „*Cloud computing označuje souhrnně technologie a postupy používané v datových centrech a firmách pro zajištění snadné škálovatelnosti aplikací dodávaných přes Internet.*“.[1]

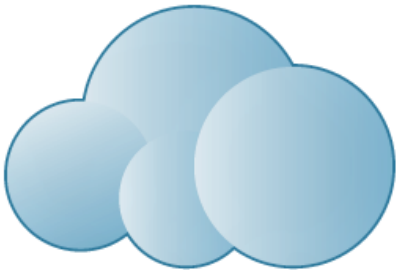
O prozatím nejkomplexnější definici se pokusila vědecká instituce NIST působící v USA dokumentem „**The NIST Definition of Cloud Computing**“. Tato definice se stala celosvětově nejlépe přijímanou a je v odborných kruzích hojně užívána. V českém překladu znamená: „*Cloud computing je model umožňující pohodlný, síťový přístup na požádání do sdílené paměti konfigurovatelných výpočetních zdrojů (např. sítě, servery, úložná zařízení, aplikací a služeb), které lze rychle zásobit a uvolnit s minimálním manažerským úsilím a řízením nebo interakcí s poskytovatelem služeb. Tento cloud (mrakový) model podporuje dostupnost a skládá se z pěti základních charakteristik, tří užitných modelů a čtyř modelů rozmístění.*“.[2]

1.2 Historie Cloud computingu

Historie CC sahá už do šedesátých let minulého století, kdy v roce 1961 pan John McCarthy, profesor z prestižní americké univerzity MIT, poprvé představil myšlenku sdílení ICT formou veřejné služby. Celou ideu prezentoval v téže logice, jako je například distribuce elektrické energie nebo vody. Elektrickou energii v dnešní době potřebuje téměř každá domácnost či firma pro napájení svých elektrických spotřebičů. Skoro nikdo si ale nebude kvůli potřebě elektrické energie pořizovat vlastní elektrárnu.[3]

Mnohem častěji je dnes využíván model, kdy jednu elektrárnu využívají desetitisíce odběratelů, kteří se k ní jednoduše připojují vzdáleně pomocí elektrorozvodné sítě. Ve světě současných počítačů přitom v hlavních přenesených rolích vystupují: elektrárna coby datové centrum poskytovatele CC, elektrorozvodná síť coby Internet a elektrický spotřebič coby počítač. Právě díky tomuto, něco málo přes padesát let starému, srovnání počítačů, elektřiny a podobných služeb souhrnně nazývaných v angličtině utility se Cloud computingu někdy říká také Utility computing.[3]

Logo Cloud computingu bylo převzato z oblasti telekomunikací. Oblak je využíván v telekomunikacích pro zobrazení telekomunikační sítě. Konkrétněji řečeno: V telekomunikacích se koncové stanice připojené do Internetu zobrazují jako krabičky zapojené do oblaku s nápisem Internet a Utility computing právě s Internetem hodně operuje. Právě proto IT oblast toto vyobrazení odsud přejmulo.[3]



Obrázek 1.1: Logo Cloud computingu[2]

1.3 Dělení Cloud computingu

CC je obecně dělen podle dvou základních hledisek, a to podle služby, kterou poskytuje a podle toho, jak je poskytován. Z těchto úrovní pohledu jsou odvozeny dva základní modely pro obecný popis, kterými jsou **model nasazení** a **distribuční model**.

[2]Zdroj obrázku číslo 1.1: `http://www.vector-magz.com/wp-content/uploads/2013/11/cloud-computing-logo.jpg`

1.3.1 Model nasazení

Model nasazení (anglicky deployment model) popisuje, jak je všeobecně cloud koncovému zákazníkovi poskytován a dělíme ho na čtyři části (submodely).

1. **Veřejný (Public Cloud computing)**

   Jedná se o model, kdy je koncovému zákazníkovi nabídnuta pouze výpočetní služba. Veřejným cloudem je například Skype hlavně z důvodu, že je určen pro velký počet klientů, kterým většinou poskytuje stejnou nebo velmi podobnou funkcionalitu.[3]

2. **Soukromý (Private Cloud computing)**

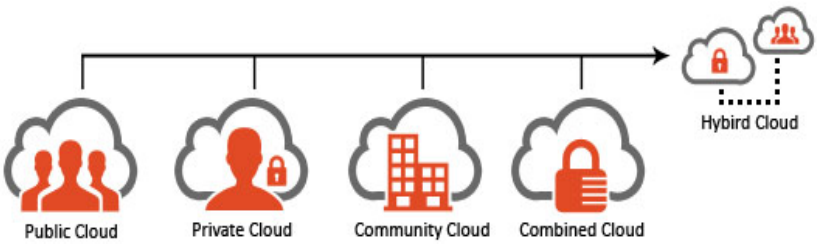
   Tento typ vznikl jako nutná alternativa k veřejnému cloudu, která se snaží eliminovat rizika a omezení plynoucí pro koncového zákazníka z nasazení veřejného cloudu. Soukromý cloud je provozován pouze pro účely konkrétní organizace, a to buď organizací samotnou, nebo třetí stranou (formou tzv. outsourcingu).[1]

3. **Hybridní (Hybrid Cloud computing)**

   Jak je již z názvu zřejmé, jedná se o kombinaci výše uvedených druhů. Hybridní cloudy navenek vystupují jako jeden celek, ale uvnitř jsou propojeny pomocí standardizačních technologií. Tohoto modelu zpravidla využívají velcí korporátní zákazníci, kterým varianta hybridního cloudu nabízí mnohem vyšší flexibilitu než samotný veřejný a soukromý cloud.[1], [3]

4. **Komunitní (Comunity Cloud computing)**

   Jedná se o model, kdy je infrastruktura cloudu sdílena mezi několika organizacemi, které ji využívají a kladou na ni stejné, či velmi podobné požadavky.[4]



Obrázek 1.2: Model nasazení[3]

[3]Zdroj obrázku číslo 1.2: `http://www.flexsin.com/images/img-cloud.jpg`

1.3.2 Distribuční model

Distribuční model (anglicky service model) nahlíží na poskytování cloudu z odlišného pohledu než model nasazení, který se zabýval otázkou *„Jak je cloud obecně u daného zákazníka nasazen?“*, zatímco distribuční model spíše odpovídá na otázku *„Jaký specifický typ služby je zákazníkovi v rámci cloudu poskytován?“*.

Distribuční model se dělí na tyto tři základní části (submodely):

1. **Infrastruktura jako služba – IaaS (Infrastructure as a service)**

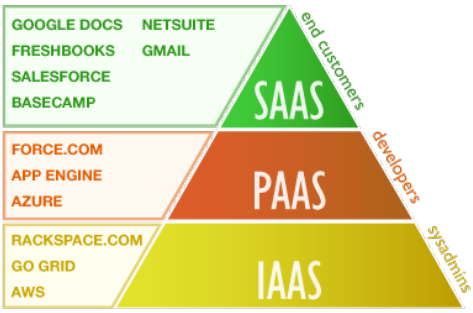
   Samotný název již napovídá, že v tomto modelu je formou služby zákazníkovi poskytována infrastruktura. Zákazník tedy nemusí investovat do nákupu hardwaru, jednoduše si ho jen pronajímá v rámci služby. Na takto pronajatý hardware si může nainstalovat jakýkoliv svůj software a provozovat ho na něm.[2]

2. **Platforma jako služba – PaaS (Platform as a service)**

   PaaS nabízí ucelenou platformu (prostředí), nad kterou může zákazník začít vyvíjet, testovat a provozovat své aplikace (produkty). Hlavním účelem je, aby se koncový zákazník nemusel zabývat konfigurací a správou cloudové infrastruktury, a to včetně serverů, operačních systémů, konfigurace prostředí jednotlivých aplikací, hostingu atd.[5]

3. **Software jako služba – SaaS (Software as a service)**

   Cílem SaaS je nabídnout zákazníkovi software (formou služby), který je provozován na cloudové infrastruktuře. Poskytovaný SW je typicky přístupný přes nějakého tenkého klienta a většinou jedná o přístup z webového prohlížeče.[1]



Obrázek 1.3: Distribuční model[4]

[4]Zdroj obrázku číslo 1.3: `http://www.globaldots.com/wordpress/wp-content/uploads/2013/03/pas_ias_sas.png`

Kapitola 2

Popis problematiky

V úvodu bylo už zmíněno, že problém zobrazování obsahu na velkoplošných vizualizačních zařízeních řeší speciální software. V dnešní době je nabízeno hned několik takovýchto softwarů. Bohužel drtivá většina z nich je specializována pouze na jednu konkrétní oblast zpracování obrazu. Existuje však SW, který se snaží pokrýt oblast vizualizace na velkoplošných zařízeních o něco komplexněji. Jedná se o software s názvem SAGE2™ (z anglického Scalable Adaptive Graphics Environment) v českém překladu „*škálovatelné adaptivní grafické prostředí*“ a vyvíjí ho zaměstnanci univerzity Illinois v Chicagu.

Právě z důvodu universálnosti a snadné rozšiřitelnosti je SW SAGE2™ používám v laboratoři **SAGELab** na FIT ČVUT[5].

Obrázek 2.1: Stěna 20 dělených displejů na FIT ČVUT (SAGELab) se spuštěným softwarem SAGE2[6]

[5]Více informací o laboratoři SAGELab se lze dozvědět na: `http://sagelab.cesnet.cz/o-laboratori/`

[6]Zdroj obrázku číslo 2.1: `http://sage.fit.cvut.cz/wp-content/uploads/2014/11/cesnet-sage2-alternative.jpg`

Bohužel SW SAGE2^{TM} má z pohledu užití běžným uživatelem mnoho podstatných nevýhod. Mezi nejzávažnější patří: v některých situacích užití velmi neintuitivní ovládání, komplikované a nedoladěné GUI, některé očekávané funkce chybí zcela, a to například: autentizace, správa uživatelů, správa souborů aj. Pro běžného uživatele tyto nedostatky mohou naneštěstí představovat mnoho překážek v použití.

Souhrnným cílem této bakalářské práce a spolupracujících bakalantů je v rámci společného projektu navrhnout moderní webovou aplikaci nad SW SAGE2^{TM}, která by umožnovala využít stěnu umístěnou v laboratoři SAGE-Lab pro prezentování multimediálního obsahu téměř komukoliv a zároveň vyřešila hlavní nevýhody SAGE2^{TM}.

Cílem této BP je věnovat se problematice možnosti uložení datových souborů na vytvořený vzdálený cloud server. Konkrétně se jedná o navržení backednu do výše uvedené webové aplikace formou RESTful Web API služby, která bude poskytovat rozhraní pro upload, správu a konverzi souborů na vzdáleném cloudovém úložišti.

2.1 Možnosti řešení

V podkapitole jsou popsány dva možné přístupy, jakými lze problematiku této bakalářské práce řešit.

1. **Využití self-hosted cloudového softwaru**

   První varianta potencionálního řešení počítá s tím, že by se pro cloudový server využila již dostupná varianta open-source softwaru na trhu tzv. „***self-hosted cloud software***“, který po nainstalování na server nabízí vybudování vlastního privátního cloudového úložiště velmi elegantním způsobem. Soubory jsou uloženy na centrálním vzdáleném úložišti a lze je synchronizovat s počítači a mobilními zařízeními. Nabízené funkce jsou pak velmi podobné jiným dnes populárním SaaS cloudům, jako je například Dropbox a Google Disk, avšak s tím rozdílem, že uživatel může mít zřízený vlastní cloud server bez uměle stanovených limitů na úložný prostor nebo připojení klientů. RESTful Web API by zajišťovalo nahrávání a správu souborů nad vytvořeným self-hosted cloud serverem.

2. **Využití dostupného veřejného SaaS cloudu**

   Druhá varianta možného řešení počítá s tím, že by se pro uložení dat, namísto vlastního privátního cloudového serveru, využila dostupná cloudová služba, jako je například Dropbox nebo Google Disk. Vytvořené RESTful Web API by pak komunikovalo s vybranou cloudovou službou a zajišťovalo nahrávání a správu souborů.

Kapitola 3

Analýza a návrh

3.1 Analýza požadavků

Pro snadnější a jednodušší vývoj každého nově vznikajícího systému je nejdříve nutné podrobněji specifikovat jeho funkčnosti. V oboru softwarového inženýrství k tomu slouží tzv. „*Analýza funkčních a nefunkčních požadavků*“, která zevrubněji popisuje to, co má systém umět, jak se má chovat a specifikuje případné vymezení systému (hardwarové, spolehlivostní a mnoho dalších).

V této podkapitole jsou uvedeny funkční a nefunkční požadavky, jež byly stanoveny v rámci podrobné analýzy, na které se podíleli spolupracující bakalanti a další uživatelé systému SAGE2™ v SAGELab na FIT ČVUT. Zapojeni do analýzy byli z toho důvodu, aby se zužitkovaly jejich cenné zkušenosti získané v SAGELab, aby analýza byla přesnější a také reflektovala skutečné potřeby na systém od relevantních uživatelů.

Na konci je uvedena podkapitola, kde jsou popsány případy užití (anglicky use case), které více zpřesňují funkční požadavky a definují způsob interakce mezi uživatelem a systémem.

3.1.1 Funkční požadavky

Podkapitola pojednává podrobněji o funkčních požadavcích, které jsou na nově vznikající systém kladeny.

F1 Manipulace se soubory

Systém bude uživateli poskytovat rozhraní pro základní operace se soubory uložených na vytvořeném vzdáleném cloud serveru. Jedná se především o tyto operace: nahrát, smazat, přejmenovat, přesunout a stáhnout daný soubor.

F2 Manipulace se složkami
Systém bude také uživateli nabízet rozhraní pro základní operace se složkami umístěných na vzdáleném cloud serveru. Jedná se hlavně o tyto operace: vytvořit, smazat, přejmenovat, přesunout a stáhnout složku.

F3 Generování náhledů a zmenšenin
Systém bude poskytovat rozhraní pro generování náhledů a zmenšenin z datových souborů, a to v těchto třech konkrétních podobách:

1. vygenerovat zmenšeninu z obrazového souboru,
2. vygenerovat obrazový náhled z PDF souboru (tj. provést konverzi),
3. vygenerovat obrazový náhled z video souboru (tj. provést konverzi).

3.1.2 Nefunkční požadavky

Tato podkapitola popisuje nefunkční (doplňkové) požadavky, které více zpřesňují design a celkové provedení vyvíjeného systému.

N1 Dostupnost systému
Systém bude dostupný na síti Internet, a to pomocí webové služby, konkrétně jako RESTful Web API. Pro využití této služby nebude uživatel muset instalovat žádné další dodatečné programy a doplňky.

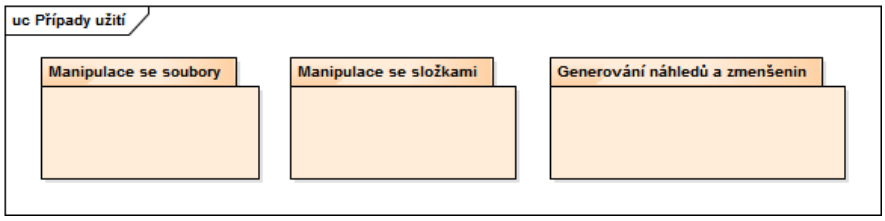
N2 Uložení dat
Systém bude ukládat data na vytvořený vzdálený cloud server. Konkrétní technologie pro tento server bude vybrána na základě analýzy (rešerše) dostupných cloudových technologií.

N3 Formát výměny dat
Webová služba implementovaná pomocí RESTful Web API bude pro komunikaci s vytvořeným cloud serverem a ke komunikaci s uživatelem využívat formát pro výměnu dat zvaný JSON, a to z důvodů jeho jednoduchosti, aktuální rozšířenosti a podporovatelnosti ve většině dnešních programovacích jazycích. Formát JSON rovněž využívají ostatní bakalanti, a proto je důležité používat jednotný formát v rámci zachování kompatibility a jednoduchosti.

N4 Operační systém
Systém bude implementován, nasazen a otestován na vybraném operačním systému rodiny Linux, a to v rámci zachování kompatibility se softwarem SAGE2^{TM}, který se nyní na FIT ČVUT v SAGELab využívá.

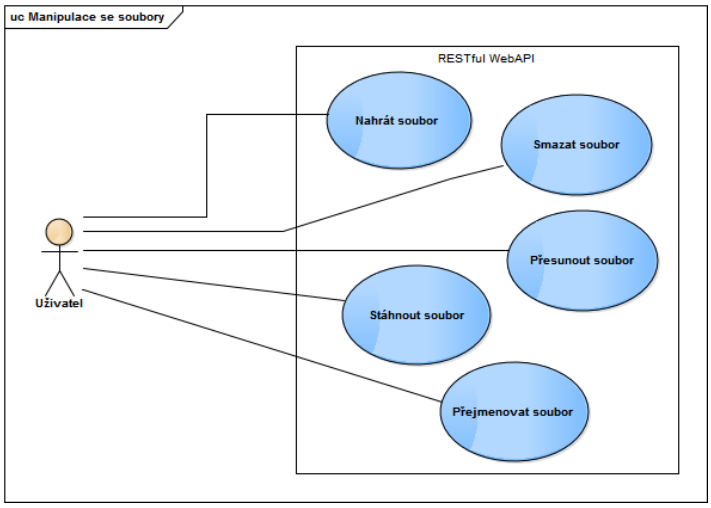
3.1.3 Případy užití (Use Cases)

Podkapitola popisuje případy užití, které podrobně specifikují interakci mezi uživatelem a vznikajícím systémem. Nutno dodat, že uživatel je zde pouze abstraktní role. Konkrétním uživatelem může být jak lidský uživatel, tak i nějaký automatizovaný systém či služba.



Obrázek 3.1: Model jednotlivých případů užití

3.1.3.1 Manipulace se soubory

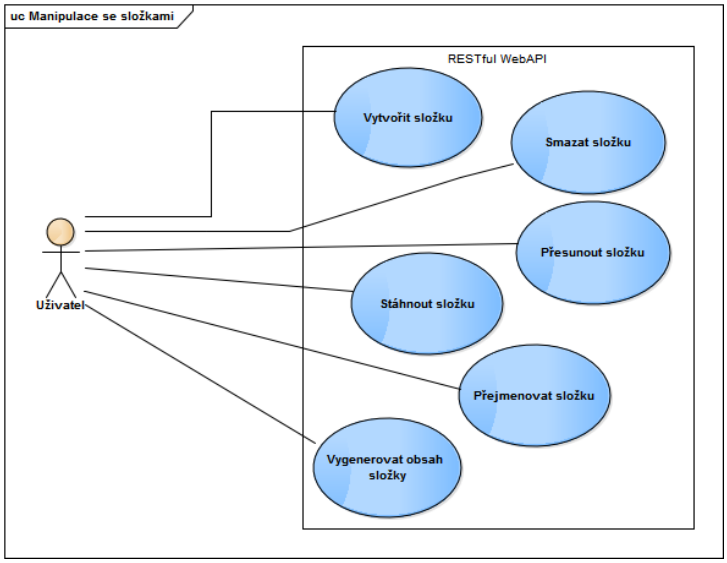


Obrázek 3.2: Model případu užití pro F1 Manipulace se soubory

- **Nahrát soubor**
  - (a) Případ užití začíná, pokud bude chtít uživatel nahrát soubor na vzdálený cloud server.
  - (b) Uživatel zavolá příslušný skript RESTful Web API, kterému předá absolutní cestu k souboru, jež má být na cloud server nahrán.
  - (c) Systém nahraje daný soubor na vzdálený cloud server.
  - (d) O výsledku celé akce systém informuje uživatele JSON výstupem.

- **Stáhnout soubor**
  (a) Případ užití začíná, pokud bude chtít uživatel stáhnout soubor ze vzdáleného cloud serveru.
  (b) Uživatel zavolá příslušný skript RESTful Web API, kterému předá absolutní cestu k souboru, jež je uložen na cloud serveru.
  (c) Systém vygeneruje konkrétní URL, ze kterého bude možné soubor stáhnout a zobrazí ho uživateli ve formátu JSON.
- **Smazat soubor**
  (a) Případ užití začíná, pokud bude chtít uživatel smazat nějaký soubor ze vzdáleného cloud serveru.
  (b) Uživatel zavolá příslušný skript RESTful Web API, kterému předá absolutní cestu k souboru, jež je uložen na cloud serveru.
  (c) Systém vymaže soubor ze serveru.
  (d) O výsledku celé akce systém informuje uživatele JSON výstupem.
- **Přesunout soubor**
  (a) Případ užití začíná, pokud bude chtít uživatel přesunout nějaký existující soubor na vzdáleném cloud serveru.
  (b) Uživatel zavolá daný skript RESTful Web API, kterému předá absolutní cestu k souboru, jež má být přesunut a novou cestu, na kterou má být přemístěn.
  (c) Systém přesune soubor z původního na nově definované umístění.
  (d) O výsledku celé akce systém informuje uživatele JSON výstupem.
- **Přejmenovat soubor**
  (a) Případ užití začíná, pokud bude chtít uživatel přejmenovat existující soubor na vzdáleném cloud serveru.
  (b) Uživatel zavolá daný skript RESTful Web API, kterému předá absolutní cestu k souboru, jež má být přejmenován a jeho nový název.
  (c) Systém přejmenuje soubor na nově definovaný název.
  (d) O výsledku celé akce systém informuje uživatele JSON výstupem.

3.1.3.2 Manipulace se složkami



Obrázek 3.3: Model případu užití pro F2 Manipulace se složkami

- **Vytvořit složku**
  (a) Případ užití začíná, pokud bude chtít uživatel vytvořit novou složku na vzdáleném cloud serveru.
  (b) Uživatel zavolá příslušný skript RESTful Web API, kterému předá absolutní cestu nově vytvářené složky (včetně jejího názvu).
  (c) Systém vytvoří novou složku na serveru, která bude umístěna na zadanou cestu.
  (d) O výsledku celé akce systém informuje uživatele JSON výstupem.
- **Smazat složku**
  (a) Případ užití začíná, pokud bude chtít uživatel smazat existující složku ze vzdáleného cloud serveru, a to včetně jejího obsahu.
  (b) Uživatel zavolá příslušný skript RESTful Web API, kterému předá absolutní cestu k právě mazané složce (včetně jejího názvu).
  (c) Systém smaže zadanou složku ze serveru.
  (d) O výsledku celé akce systém informuje uživatele JSON výstupem.

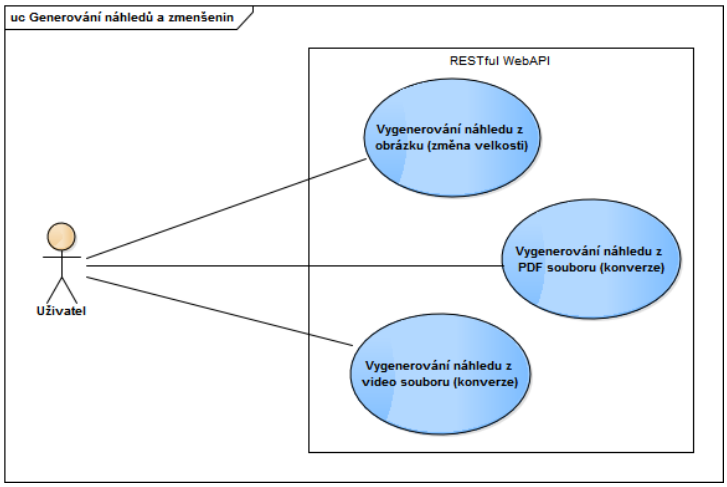
- **Stáhnout složku**
  (a) Případ užití začíná, pokud bude chtít uživatel stáhnout obsah celé složky ze vzdáleného cloud serveru.
  (b) Uživatel zavolá daný skript RESTful Web API, kterému předá absolutní cestu ke složce, jež má být stažena.
  (c) Systém vygeneruje konkrétní URL, ze kterého bude možné složku ve formě archivního souboru (ZIP) stáhnout a zobrazí ho ve formátu JSON uživateli.

- **Přejmenovat složku**
  (a) Případ užití začíná, pokud bude chtít uživatel přejmenovat nějakou existující složku na vzdáleném cloud serveru.
  (b) Uživatel zavolá daný skript RESTful Web API, kterému předá absolutní cestu ke složce, jež má být přejmenována a její nový název.
  (c) Systém přejmenuje složku na nově definovaný název.
  (d) O výsledku celé akce systém informuje uživatele JSON výstupem.

- **Přesunout složku**
  (a) Případ užití začíná, pokud bude chtít uživatel přesunout exitující složku na vzdáleném cloud serveru.
  (b) Uživatel zavolá daný skript RESTful Web API, kterému předá absolutní cestu ke složce, jež má být přesunuta a její novou absolutní cestu, na kterou má být složka přemístěna.
  (c) Systém přesune složku z původního na nově definované umístění.
  (d) O výsledku celé akce systém informuje uživatele JSON výstupem.

- **Vygenerování obsahu složky**
  (a) Případ užití začíná, pokud bude chtít uživatel vygenerovat obsah dané složky umístěné na vzdáleném cloud serveru.
  (b) Uživatel zavolá příslušný skript RESTful Web API, kterému předá absolutní cestu ke zvolené složce.
  (c) Systém uživateli zobrazí obsah zadané složky ve formátu JSON.

3.1.3.3 Generování náhledů a zmenšenin



Obrázek 3.4: Model případu užití pro F3 Generování náhledů a zmenšenin

- **Vygenerování náhledu z obrázku (změna velkosti)**
  - (a) Případ užití začíná, pokud bude chtít uživatel vygenerovat zmenšeninu z obrazového souboru, který je umístěn na PC, kde je spuštěno RESTful Web API.
  - (b) Uživatel zavolá příslušný skript RESTful Web API, kterému předá absolutní cestu k obrazovému souboru plus výšku a šířku, na kterou má být výsledný soubor zmenšen.
  - (c) Systém vygeneruje zmenšeninu z určeného souboru o definované výšce a šířce.
  - (d) O výsledku celé akce systém informuje uživatele JSON výstupem.
- **Vygenerování náhledu z PDF souboru (konverze)**
  - (a) Případ užití začíná, pokud bude chtít uživatel vygenerovat obrazový náhled z PDF souboru, který je umístěn na PC, kde je spuštěno RESTful Web API.
  - (b) Uživatel zavolá daný skript RESTful Web API, kterému předá absolutní cestu k PDF souboru plus výšku, šířku, číslo stránky, ze které má být výsledný náhled vygenerován a nakonec ještě výstupní formát (např.: JPG, PNG aj.).
  - (c) Systém vytvoří v určeném výstupním formátu obrazový náhled z konkrétního čísla stránky PDF souboru o definované výšce a šířce.
  - (d) O výsledku celé akce systém informuje uživatele JSON výstupem.

- **Vygenerování náhledu z video souboru (konverze)**
  (a) Případ užití začíná, pokud bude chtít uživatel vygenerovat obrazový náhled z video souboru, který je umístěn na PC, kde je spuštěno RESTful Web API.
  (b) Uživatel zavolá příslušný skript RESTful Web API, kterému předá absolutní cestu k video souboru plus výšku, šířku, číslo snímku videa, ze kterého má být výsledný náhled vygenerován a nakonec ještě výstupní formát (např.: JPG, PNG aj.).
  (c) Systém vytvoří v určeném výstupním formátu obrazový náhled z konkrétního snímku videa o definované výšce a šířce.
  (d) O výsledku celé akce systém informuje uživatele JSON výstupem.

3.2 Rešerše self-hosted cloudových softwarů

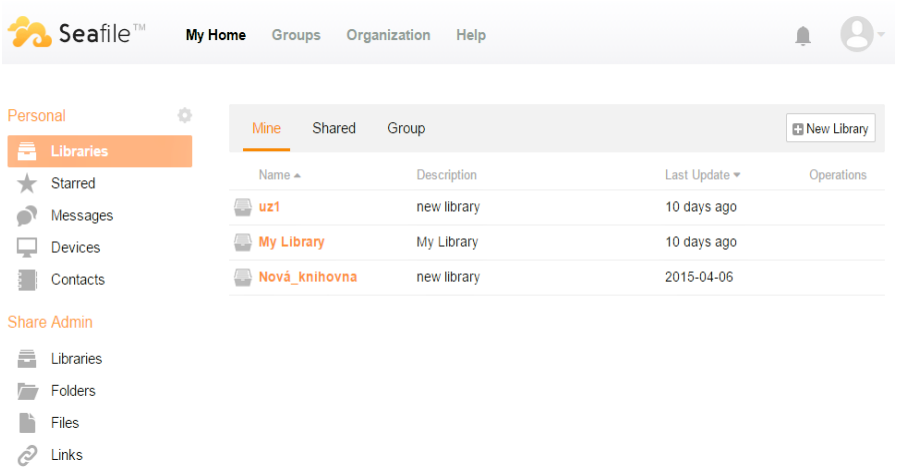
V této podkapitole jsou souhrnně uvedeny výsledky rešerše dostupných softwarů, které umožnují vytvořit si vlastní privátní self-hosted cloud server.

3.2.1 Seafile™

Je to open-source SW, který nabízí koncovému uživateli možnost vybudování vlastního privátního cloudového úložiště (neboli self-hosted cloud serveru). Soubory jsou uloženy na vytvořeném centrálním serveru a lze je snadno synchronizovat s osobními počítači a mobilními zařízeními přes klienta Seafile. Nabízené funkce jsou velmi podobné jiným dnešním populárním cloudovým službám, jako jsou například Dropbox a Google Disk, ale s tím rozdílem, že uživatel může mít zřízený vlastní cloud server bez uměle stanovených limitů na úložný prostor, nebo počet připojených klientů.

3.2.1.1 Shrnutí nejdůležitějších funkcí

- klient dostupný pro mnoho OS: Windows, Linux, OS X, Android, iOS,
- webový klient,
- šifrování souborů na straně klienta,
- správa verzí souborů,
- správa uživatelů,
- selektivní synchronizace souborů,
- veřejné sdílení souborů,
- podpora týmové spolupráce.[6]



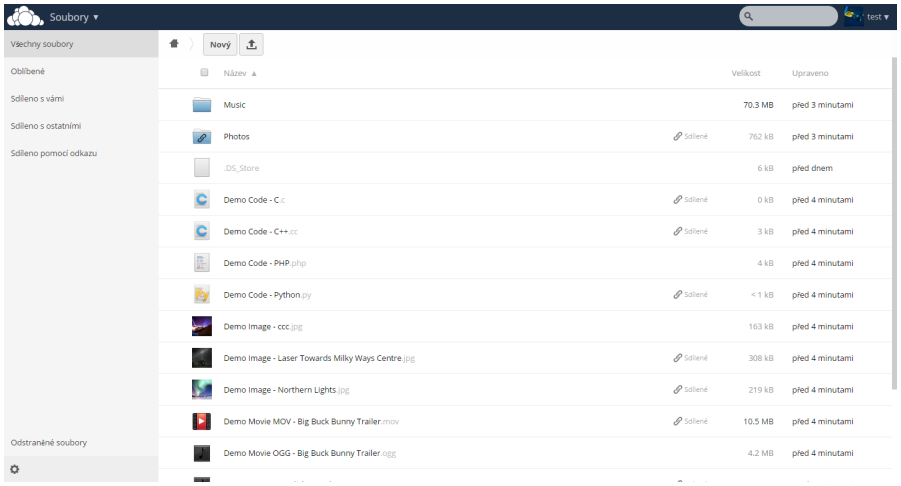
Obrázek 3.5: Webový klient Seafile™

3.2.2 ownCloud

Druhým open-source softwarem, který nabízí možnost vytvoření selft-hosted cloud serveru je ownCloud. OwnCloud klade velký důraz na své webové rozhraní, které má oproti jiným alternativám zatím funkcionálně nejlepší. V rozhraní je totiž implementováno mnoho užitečných pluginů, například na: prohlížení fotek, základní editaci fotek, přehrávání videí, zobrazování PDF souborů, integraci se sociálními sítěmi aj. V tomto ohledu je ownCloud opravdu „mílové“ kroky před konkurencí.

3.2.2.1 Shrnutí nejdůležitějších funkcí

- klient dostupný pro mnoho OS: Windows, Linux, OS X, Android, iOS,
- webový klient s mnoha funkcemi,
- šifrování uživatelských souborů,
- správa verzí souborů,
- správa uživatelů,
- veřejné sdílení souborů,
- možnost připojit externí služby (Dropbox, Google Disk aj.).[7]



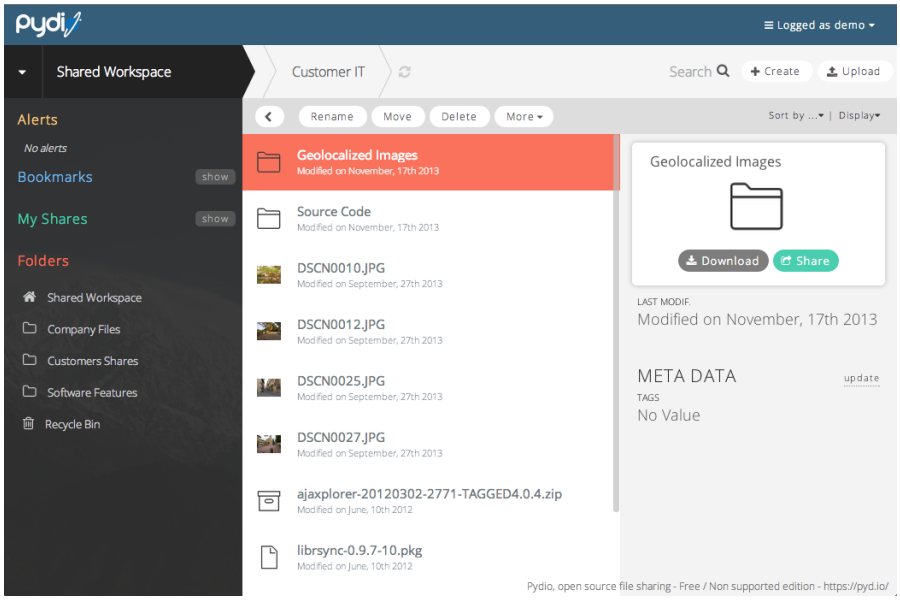
Obrázek 3.6: Webový klient ownCloudu

3.2.3 Pydio

Třetím z řady uváděných open-source SW se nazývá Pydio. Nabízí velmi sofistikovaného webového klienta s mnoha integrovanými doplňky (velmi podobný konkurenčnímu ownCloudu). Funkčně je velmi srovnatelný s již uvedenými alternativami.

3.2.3.1 Shrnutí nejdůležitějších funkcí

- klient dostupný pro mnoho OS: Windows, Linux, OS X, Android, iOS,
- webový klient s mnoha funkcemi,
- správa verzí souborů,
- správa uživatelů,
- šifrování uživatelských souborů,
- veřejné sdílení souborů.[8]



Obrázek 3.7: Webový klient Pydio[7]

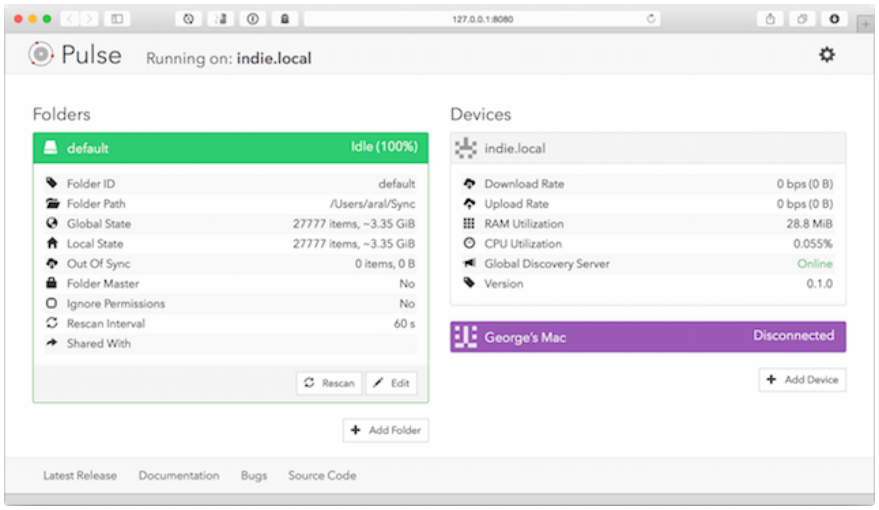
[7]Zdroj obrázku číslo 3.7: `https://pyd.io/wp-content/uploads/2014/09/03-WORKSPACE-MAIN-SCREEN.png`

3.2.4 Pulse

Posledním uvedeným SW je Pulse, který svým uživatelům nabízí opět podobné funkce jako ostatní. Bohužel se zatím nachází ve stádiu beta vývoje, a tudíž zkušený uživatel určitě vybere nějaké stabilní konkurenční řešení, protože Pulse bohužel nenabízí žádnou funkci, kterou by nenabízeli ostatní.

3.2.4.1 Shrnutí nejdůležitějších funkcí

- webový klient,
- správa verzí souborů,
- správa uživatelů.[9]



Obrázek 3.8: Webový klient Pulse[8]

[8]Zdroj obrázku číslo 3.8: `http://cdn.makeuseof.com/wp-content/uploads/2014/12/indie-pulse.png?6055c3`

3.3 Rešerše dostupných cloudových služeb (SaaS)

V této podkapitole jsou souhrnně uvedeny výsledky rešerše veřejně dostupných cloudových služeb (SaaS) na trhu. Rešerše se zaměřila na nejznámější a nejpoužívanější cloudové služby současnými uživateli. Jedná se konkrétně o: iCloud, Egnyte, Google Apps, OpenDrive, Amazon Cloud Drive a DropBox.

Služby byly porovnány na nejdůležitější kritéria, jakými jsou například: cena služby, nabízené funkce, platformní dostupnost ad. Výsledky srovnání jsou shrnuty v závislosti na porovnávaném kritériu do několika tabulek níže.

Tabulka 3.1: Cena služby (nejmenší možná v dolarech)[10]

iCloud	Egnyte	Google Apps	Open-Drive	Dropbox	Amazon Cloud
1,66 $	24,99 $	5 $	5 $	9,99 $	1,66 $

Tabulka 3.2: Získaný volný prostor za výše uvedené ceny[10]

iCloud	Egnyte	Google Apps	Open-Drive	Dropbox	Amazon Cloud
15 GB	150 GB	1 GB, 25 GB pro e-mail	100 GB	100 GB	20 GB

Tabulka 3.3: Možný obsah na úložišti[10]

	iCloud	Egnyte	Google Apps	Open-Drive	Dropbox	Amazon Cloud
E-maily	ano	ano	ano	ne	ne	ne
Kontakty	ano	ano	ano	ne	ne	ne
Kalendář	ano	ano	ano	ne	ne	ne
Dokumenty	ano	ano	ano	ano	ano	ano
Tabulky	ano	ano	ano	ano	ano	ano
Hudba	ano	ano	ano	ano	ano	ano
Fotografie	ano	ano	ano	ano	ano	ano
Videa	ano	ano	ano	ano	ano	ano
Prezentace	ano	ano	ano	ne	ano	ano

Tabulka 3.4: Dostupné funkce[10]

	iCloud	Egnyte	Google Apps	Open-Drive	Dropbox	Amazon Cloud
Zobrazení souborů	ano	ano	ano	ano	ano	ano
Editování souborů	ano	ano	ano	ano	ano	ne
Automatická synch. souborů	ano	ano	ne	ano	ano	ne
Autentizace	ano	ano	ano	ano	ano	ano
Šifrování při přenosu	ano	ano	ano	ano	ano	ano
Streaming médií	ano	ne	ne	ano	ne	ano
Veřejné sdílení souborů	ne	ano	ano	ano	ano	ne
Přístup přes lokální klienty	ano	ano	ano	ano	ano	ne

Tabulka 3.5: Možnost přístupu přes mobilního klienta[10]

	iCloud	Egnyte	Google Apps	Open-Drive	Dropbox	Amazon Cloud
iPhone	ano	ano	ano	ano	ano	ne
iPad	ano	ano	ano	ano	ano	ne
Windows phone	ne	ano	ano	ne	ne	ne
Android	ne	ano	ano	ne	ano	ano
BlackBerry	ne	ne	ano	ne	ano	ne

Tabulka 3.6: Podpora operačních systémů klienty[10]

	iCloud	Egnyte	Google Apps	Open-Drive	Dropbox	Amazon Cloud
Windows (XP až 8.1)	ano	ano	ano	ano	ano	ano
Mac OS X	ano	ano	ano	ano	ano	ano
Linux	ne	ano	ne	ne	ano	ne

3.4 Zvolené řešení

V podkapitole **2.1 Možnosti řešení** bylo popsáno, jak je možné problematiku, kterou se zabývá tato bakalářská práce, řešit. Byly nastíněny dva možné přístupy, z nichž byl pro realizaci zvolen **první** popsaný, který počítá s vybudováním vlastního cloudového serveru pomocí specializovaného softwaru.

První varianta byla upřednostněna před druhou hlavně z těchto důvodů:

- Na vlastním (privátním) self-hosted cloud serveru je mnohem jednodušší a pohodlnější rozšířit využívané prostředky (např.: velikost RAM paměti, velikost dostupné paměti na HDD, výpočetní výkon aj.) než na veřejně dostupné cloudové službě (například DropBox aj.), kde by se musel za každé navýšení prostředků zvýšit placený měsíční paušál.
- Veřejně dostupné cloudové služby nabízejí zákazníkům své produkty velmi specificky, a to ve formě unifikovaných balíčků, což je značně, vzhledem k budoucí rozšiřitelnosti funkcí systému, omezující. Privátní cloudvé úložiště provozované na vlastním serveru v tomto ohledu nabízí o mnoho větší variabilitu.
- Z hlediska bezpečnosti a autorských práv je samozřejmě vlastní cloud server taktéž lepší volbou, protože nabízí možnost určit si vlastní bezpečnostní politiku ochrany dat a přístupu k serveru obecně. Není tedy problém uložená data na vlastním serveru šifrovat, kdežto veřejně dostupné cloudouvé služby tuto možnost nenabízejí, ať je to z jakýchkoliv důvodů.

3.4.1 Vybrané technologie

Na základě provedených analýz byly pro realizaci vybrány tyto technologie:

- Privátní self-hosted cloud server, na kterém budou uložena data bude realizován pomocí open-source softwaru s názvem Seafile^{TM}, který vzešel z analýzy dostupných softwarů.
- Cloud Web API, které bude zajišťovat komunikaci s vytvořeným cloud serverem bude využívat architekturu REST. Tento požadavek vzešel na základě analýzy funkčních a nefunkčních požadavků.
- RESTful Web API bude implementováno technologií PHP a pro výměnu dat se bude využívat formát JSON, který byl určen v analýze funkčních a nefunkčních požadavků.

Kapitola 4

Použité technologie

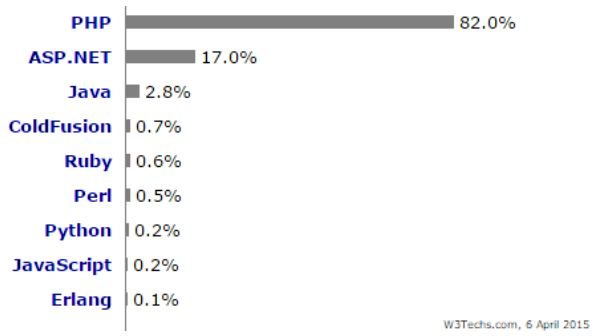
Kapitola popisuje technologie a architektury, kterých bylo během implementování řešení využito.

4.1 PHP 5

PHP (rekurzivní zkratka PHP: Hypertext Preprocessor, česky „PHP: Hypertextový preprocesor“) je open-source skriptovací jazyk, který v dnešní době slouží pro tvorbu dynamických webových stránek.[11]

Počátky tohoto jazyka spadají do roku 1994, kdy se pan Rasmus Lerdorf rozhodl vytvořit jednoduchý systém pro počítání uživatelských přístupů ke svým webovým stránkám. Sada těchto skriptů byla ještě téhož roku vydána pod názvem „Personal Home Page Tools“, zkráceně tedy PHP.[12] Systém si postupně získal celosvětovou proslulost a pro svou oblíbenost byl nadále vyvíjen, až vývoj dospěl do dnešní aktuální verze 5.6.8 (vydána 16. 4. 2015).[13]

V současnosti je PHP nejrozšířenější skriptovací jazyk pro web s tržním podílem 82 % (k 6. 4. 2015).



Obrázek 4.1: Přehled užívanosti prog. jazyků při vývoji webových aplikací[9]

[9]Zdroj obrázku číslo 4.1: `http://w3techs.com`

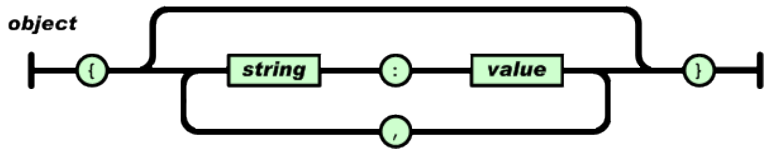
4.2 JSON

JSON (anglicky JavaScript Object Notation) je jednoduchý textový formát založený na jazyce JavaScript, který slouží pro výměnu dat. V dnešní době se jedná o nejvyužívanější formát pro výměnu dat mezi webovými aplikacemi a také webovými službami (API), kde JSON vyhrál konkurenční boj s formátem XML (anglicky Extensible Markup Language), a to vzhledem ke své celkové jednoduchosti. Zápis tohoto formátu je velmi dobře čitelný pro člověka a jednoduše zpracovatelný strojově, viz ukázka níže.[14]

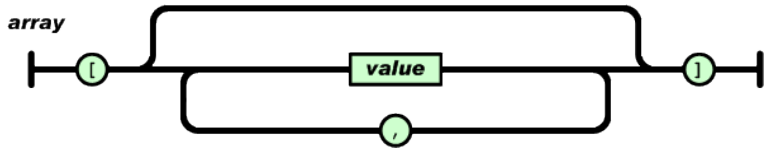
```
[ {"id": "1", "name": "Finite_automata"},
  {"id": "2", "name": "Pushdown_automata"},
  {"id": "3", "name": "Turing_machine"} ]
```

I přes fakt, že je JSON odvozen z JavaScriptu, je zcela na jazyce nezávislý, avšak užívá dobře známé konvence z jazyků C, C++, Perl, Python atd. Vnitřní struktura formátu JSON využívá pouze dvou následujících datových struktur:

1. **Kolekce párů název/hodnota**, která je v JSON implementována jako objekt – viz obrázek číslo 4.2.[15]
2. **Seřazený seznam hodnot** v JSON implementovaný jako pole – viz obrázek číslo 4.3.[15]



Obrázek 4.2: Znázornění implementace kolekce párů název/hodnota[10]



Obrázek 4.3: Znázornění implementace seřazeného seznamu hodnot[11]

[10]Zdroj obrázku číslo 4.2: `http://www.json.org/object.gif`
[11]Zdroj obrázku číslo 4.3: `http://www.json.org/array.gif`

4.3 Architektura REST

REST (z anglického Representational State Transfer) je dnes velmi oblíbená architektura využívaná při návrhu moderních webových aplikací a webových služeb. Autorem je pan Roy Thomas Fielding, který v roce 2000 napsal disertační práci[12] s názvem „*Architectural Styles and the Design of Network-based Software Architectures*“, což v českém překladu znamená „*Architektonické styly a návrh síťově založených softwarových architektur*“, ve které poprvé představil a definoval myšlenku REST. Vzhledem k tomu, že pan Fielding je jedním z hlavních spoluautorů HTTP protokolu (z anglického Hypertext Transfer Protocol), nikoho tedy nezaskočí, že REST využívá právě HTTP.[16]

REST je datově orientovaná architektura, která slouží pro přístup k datovým zdrojům. Každý zdroj je na webu jednoznačně identifikován svým URI (z anglického Uniform Resource Identifier). REST k tomuto URI přistupuje pomocí HTTP protokolu a pro práci s datovými zdroji používá čtyři operace CRUD (Create, Read, Update a Delete), které jsou v HTTP implementované jako známé metody tohoto protokolu: GET, PUT, POST, DELETE... [17]

Na konec je důležité poznamenat, že REST je pouze architektonický styl. Nejedná se zatím o žádný oficiální standard nebo implementační specifikaci, která by byla nějakou respektovanou autoritou v oblasti webu oficiálně uznána (například populární W3C).[18]

Tabulka 4.1: Využití metod HTTP pro „RESTful“ webové služby[13]

Datový zdroj	GET	PUT	POST	DELETE
URI kolekce, např.: http://example.com/resources/	Vrátit (Retrieve) seznam členů kolekce	Vyměnit (Replace) celou kolekci za jinou	Vytvořit (Create) nový záznam do kolekce	Smazat (Delete) celou kolekci
URI prvku, např.: http://example.com/resources?id=2323	Vrátit (Retrieve) člena kolekce s id = 2323	Nahradit (Replace) člena kolekce	Vytvořit (Create) v členu kolekce nový záznam	Smazat (Delete) člena z kolekce

[12] Disertační práci pana Fieldinga lze nalézt na: `https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf`

[13] Zdroj tabulky číslo 4.1: `http://cs.wikipedia.org/wiki/Representational_State_Transfer`

Kapitola 5

Realizace

Kapitola souhrnně popisuje průběh realizace této práce. Vzhledem k detailně provedeným analýzám a rešerším bylo možno při realizace postupovat velmi systematicky a přímočaře. Díky tomu během implementace nenastaly žádné neočekávané situace. Realizace byla rozdělena celkem do dvou hlavních etap. V první byl vytvořen a zprovozněn cloud server ve druhé etapě bylo implementováno RESTful Web API.

5.1 Vytvoření cloud serveru

V kapitole **3.4 Zvolené řešení** bylo již uvedeno, že pro vytvoření privátního self-hosted cloud serveru byl na základě analýzy vybrán open-source software s názvem **Seafile**^{TM}, jež spadá do rodiny softwarů, které uživateli po nainstalování na server nabízí vybudovat si vlastní (self-hosted) cloudové úložiště.

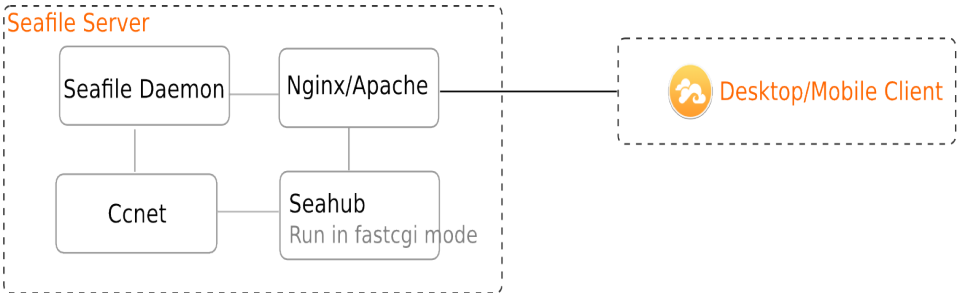
Před vlastním nasazením SW Seafile^{TM} bylo potřeba připravit server tak, aby splňoval minimální požadavky na něho kladené. Nejdříve bylo nutné nainstalovat operační systém. Z analýzy nefunkčních požadavků vyplynulo, že se má jednat o nějakou distribuci OS Linux. Nakonec byl vybrán OS Fedora 21 Server, a to vzhledem k autorovým zkušenostem s administrací serveru pod touto linuxovou distribucí.

Následně bylo potřeba nainstalovat balíčky třetích stran, které SW Seafile^{TM}, pro zajištění svých deklarovaných funkčností, vyžaduje. Jednalo se konkrétně o tyto balíčky:

- databáze MySQL,
- PHP 5.x,
- HTTP server Apache,
- python 2.7,
- python-setuptools, python-imaging a python-mysqldb.

Úspěšným nainstalováním výše uvedených balíčků již nebránilo nic tomu nainstalovat a nakonfigurovat SW Seafile™ na server, konkrétně v aktuální dostupné verzi 4.0.6.[14]

Po nainstalování SW Seafile™ lze zjistit, že se skládá z několika komponent (viz obrázek číslo 5.1), které společně zajišťují chod všech nabízených funkcionalit, jako je například: synchronizace přes HTTP/HTTPS protokol, přístup ke cloud serveru skrz webového klienta a další.[15]



Obrázek 5.1: Komponenty SW Seafile™[16]

Posledním krokem bylo nakonfigurování webového serveru[17] Apache, který zaručuje, aby některé služby softwaru Seafile™ byly dostupné přes síť Internet skrz protokol HTTP nebo HTTPS.[18]

[14] Podrobný návod na instalaci a konfiguraci SW Seafile™ lze nalézt na: `http://manual.seafile.com/deploy/using_mysql.html`

[15] Více informací o celkové architektuře SW Seafile™ a porobný popis jedotlivých komponent systému lze nalézt na: `http://manual.seafile.com/overview/components.html`

[16] Zdroj obrázku číslo 5.1: `http://manual.seafile.com/images/seafile-arch-new-http.png`

[17] Více informací o úloze webového serveru lze najít například na: `http://cs.wikipedia.org/wiki/Webový_server`

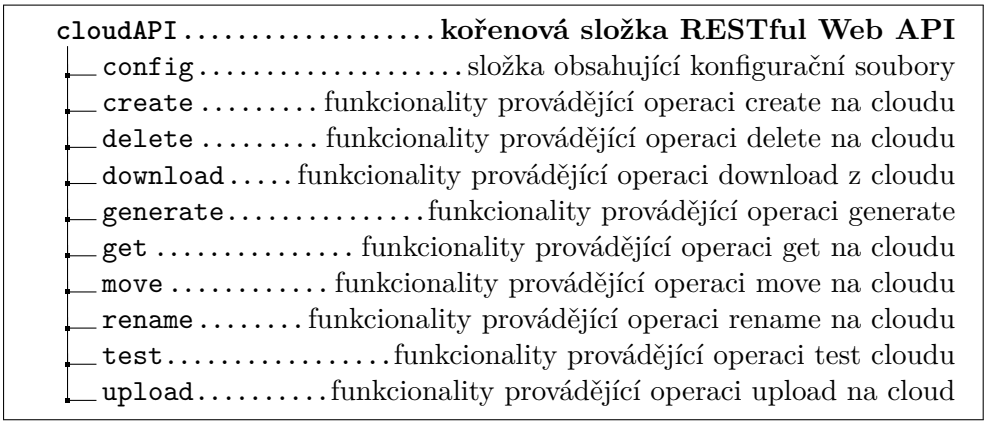
[18] Podrobný návod ke konfiguraci Apache pro SW Seafile™ lze najít na: `http://manual.seafile.com/deploy/deploy_with_apache.html`

5.2 Implementace RESTful Web API

Vytvořením fungujícího privátního self-hosted cloud serveru mohla začít implementace prvních funkcionality Web API, které bude zajišťovat správu souborů a složek nad cloudovým úložištěm.

Web API bylo implementováno ve skriptovacím jazyce PHP 5 s využitím architektury REST používající JSON, jako formát pro reprezentaci výstupních dat uživateli. Při realizace bylo ještě užito doplňků ImageMagick[19] a FFmpeg[20]. Tyto dva komplementy jsou open-source softwarové kolekce, které umožnují zpracování a konverzi, jak obrazových souborů (ImageMagick), tak i video souborů (FFmpeg).

Celé API bylo naprogramováno s ohledem na možnost jednoduchého rozšíření v budoucnu a logicky rozčleněno vzhledem k vykonávaným funkcím – viz obrázek číslo 5.2. K celkovému zvýšení universálnosti bylo nastavení centralizováno do jednoho konfiguračního souboru, ve kterém je nutné specifikovat všechny důležité parametry pro správnou činnost celého API. Jedná se například o: IP adresu Seafile cloud serveru, autorizační token pro komunikaci se Seafile cloud serverem a další. Konfigurační soubor je umístěn v implementaci na cestě „*./cloudAPI/config/config.php*“.



Obrázek 5.2: Logické rozčlenění funkcionalit vytvořeného API

[19] Více informací o doplňku ImageMagick lze nalézt na: `http://www.imagemagick.org`
[20] Více informací o doplňku FFmpeg lze nalézt na: `https://www.ffmpeg.org`

5.2.1 Popis dílčích funkcionalit vytvořeného Web API

V této podkapitole jsou detailně vysvětleny jednotlivé funkcionality API. Tento popis nebyl situován pouze do příloh, a to z toho důvodu, že vytvoření API je jednou z hlavních částí této práce, proto je jeho dokumentace nezbytná k pochopení práce jako celku, a tudíž je umístěna do hlavní části.

1. **Otestování, zda je Web API dostupné**
   - ***Popis chování:*** Zavoláním bude zjištěno, zda je aktuálně dostupné Web API Seafile cloudového serveru, jehož IP adresa je uložena v konfiguračním souboru „./cloudAPI/config/config.php“.
   - ***Příklad volání API:*** `http://www.servername.com/cloudAPI/test/test_avl.php`
   - ***Formát odpovědi:*** JSON
   - ***Odpověď:*** Bude vrácen řetězec „Cloud API is available.“, pokud je API na zadaném serveru dostupné. Naopak, pokud není dostupné, bude vráceno „Cloud API is not available.“.
2. **Vygenerování autorizačního tokenu**
   - ***Popis chování:*** Pro zadané uživatelské jméno a heslo vygeneruje jedinečný autorizační token, pomocí kterého je autorizována komunikace se Seafile cloud serverem.
   - ***Příklad volání API:*** `http://www.servername.com/cloudAPI/get/get_auth_token.php?username=root@sagelab.cz&password=pass`
   - ***HTTP metoda vůči cloud serveru:*** POST
   - ***Parametry:***
     - ◇ username – jedná se o uživatelské jméno uživatele Seafile cloud serveru
     - ◇ password – jedná se o heslo uživatele Seafile cloud serveru
   - ***Formát odpovědi:*** JSON
   - ***Odpověď:*** Bude vrácen vygenerovaný autorizační token. Například: {"token": "a527c6e26ac5c434d64c347427312e181b57ac84"}.
3. **Zjištění, zda je autorizační token funkční**
   - ***Popis chování:*** Zavoláním bude zjištěno, zda je zadaný autorizační token platný.
   - ***Příklad volání API:*** `http://www.servername.com/cloudAPI/test/test_avl_tok.php?token=a527c6e26ac5c434d64c347427312e184`
   - ***HTTP metoda vůči cloud serveru:*** GET

- ***Formát odpovědi:*** JSON
- ***Odpověď:*** Bude vrácen řetězec „pong“, pokud je zadaný autorizační token funkční. Pokud není funkční, pak je vráceno: {"detail": "You do not have permission to perform this action."}.

4. **Vygenerování tokenu k nahrávání souborů**

   - ***Popis chování:*** Pro zadanou knihovnu (repositář) vygeneruje časově omezený token, který je potřeba pro nahrávání souborů do knihovny umístěné na cloud serveru.
   - ***Příklad volání API:*** `http://www.servername.com/cloudAPI/get/get_upload_file_link.php?repo=2f7e5017-f07f-40cb`
   - ***HTTP metoda vůči cloud serveru:*** POST
   - ***Parametry:***
     - ◇ repo – jedná se o jednoznačný identifikátor knihovny (repositáře) umístěné na cloud serveru
   - ***Formát odpovědi:*** JSON
   - ***Odpověď:*** Bude vrácen vygenerovaný token určený pro nahrávání souborů. Například: „76d114c9-4126-4497-9ace-1cfded97ca6e“.

5. **Získání informací o konkrétním souboru**

   - ***Popis chování:*** Pro zadaný soubor, který je uložen na cloud serveru, vrátí podrobné informace.
   - ***Příklad volání API:*** `http://www.servername.com/cloudAPI/get/get_info_file.php?repo=75f8ebbe-e326-4021-a75d-ac&dpath=/&filename=kompilace.txt`
   - ***HTTP metoda vůči cloud serveru:*** POST
   - ***Parametry:***
     - ◇ repo – jedná se o jednoznačný identifikátor knihovny
     - ◇ dpath – absolutní cesta ke složce, ve které je soubor v dané knihovně uložen
     - ◇ filename – název souboru včetně jeho přípony
   - ***Formát odpovědi:*** JSON
   - ***Odpověď:*** Budou vráceny podrobné informace o souboru. Například: {"id": "26b7bde7573a0bad7f631cfb411e13fc2f14fb38", "mtime": 1428223314, "type": "file", "name": "kompilace.txt", "size": 156}.

6. **Smazání souboru**
   - ***Popis chování:*** Smaže zadaný soubor, který je uložen na vzdáleném cloud serveru.
   - ***Příklad volání API:*** `http://www.servername.com/cloudAPI/delete/delete_file.php?repo=75f8ebbe-e326-4021-a75d-ac84&dpath=/&filename=soubor.txt`
   - ***HTTP metoda vůči cloud serveru:*** DELETE
   - ***Parametry:***
     - ◇ repo – jedná se o jednoznačný identifikátor knihovny
     - ◇ dpath – absolutní cesta ke složce, ve které je soubor v dané knihovně uložen
     - ◇ filename – název souboru včetně jeho přípony
   - ***Formát odpovědi:*** JSON
   - ***Odpověď:*** V případě úspěchu vrácen řetězec „success“.
7. **Smazání složky**
   - ***Popis chování:*** Smaže zadanou složku (včetně obsahu), která je uložena na vzdáleném cloud serveru.
   - ***Příklad volání API:*** `http://www.servername.com/cloudAPI/delete/delete_dir.php?repo=75f8ebbe-e326-4021-a75d&name=/ddd`
   - ***HTTP metoda vůči cloud serveru:*** DELETE
   - ***Parametry:***
     - ◇ repo – jedná se o jednoznačný identifikátor knihovny
     - ◇ name – absolutní cesta ke složce, která je v dané knihovně uložena, a která má být smazána
   - ***Formát odpovědi:*** JSON
   - ***Odpověď:*** V případě úspěchu vrácen řetězec „success“.
8. **Smazání knihovny**
   - ***Popis chování:*** Smaže zadanou knihovnu (včetně obsahu), která je uložena na vzdáleném cloud serveru.
   - ***Příklad volání API:*** `http://www.servername.com/cloudAPI/delete/delete_lib.php?repo=6b79eb96-e711-4449-9a8e-790cc4c272ca`
   - ***HTTP metoda vůči cloud serveru:*** DELETE
   - ***Parametry:***
     - ◇ repo – jedná se o jednoznačný identifikátor knihovny (repositáře), která má být smazána
   - ***Formát odpovědi:*** JSON
   - ***Odpověď:*** V případě úspěchu vrácen řetězec „success“.

9. **Vytvoření nové knihovny (repositáře)**
   - ***Popis chování:*** Vytvoří novou knihovnu na cloud serveru.
   - ***Příklad volání API:*** `http://www.servername.com/cloudAPI/create/crea_new_lib.php?name=Nova_knihovna`
   - ***HTTP metoda vůči cloud serveru:*** POST
   - ***Parametry:***
     - ◇ name – název knihovny, která má být vytvořena
   - ***Formát odpovědi:*** JSON
   - ***Odpověď:*** V případě úspěchu jsou vráceny detailní informace o nově vytvořené knihovně.
10. **Vytvoření nové složky**
    - ***Popis chování:*** Vytvoří novou složku na vzdáleném cloud serveru.
    - ***Příklad volání API:*** `http://www.servername.com/cloudAPI/create/crea_new_dir.php?repo=75f8ebbe-e326&name=/nova_slozka`
    - ***HTTP metoda vůči cloud serveru:*** POST
    - ***Parametry:***
      - ◇ repo – jedná se o jednoznačný identifikátor knihovny
      - ◇ name – absolutní cesta ke složce (včetně jejího názvu), která má být v dané knihovně vytvořena
    - ***Formát odpovědi:*** JSON
    - ***Odpověď:*** V případě úspěchu vrácen řetězec „success“.
11. **Přejmenování složky**
    - ***Popis chování:*** Přejmenuje zadanou složku, která je uložena na vzdáleném cloud serveru.
    - ***Příklad volání API:*** `http://www.servername.com/cloudAPI/rename/rename_dir.php?repo=2f7e5017-f07f-40c5b&old_name=/video/1&new_name=2`
    - ***HTTP metoda vůči cloud serveru:*** POST
    - ***Parametry:***
      - ◇ repo – jedná se o jednoznačný identifikátor knihovny
      - ◇ old_name – absolutní cesta ke složce uložené v dané knihovně
      - ◇ new_name – nový název pro složku
    - ***Formát odpovědi:*** JSON
    - ***Odpověď:*** V případě úspěchu vrácen řetězec „success“.

12. **Přejmenování souboru**
    - ***Popis chování:*** Přejmenuje zadaný soubor, který je uložen na vzdáleném cloud serveru.
    - ***Příklad volání API:*** `http://www.servername.com/cloudAPI/rename/rename_file.php?repo=2f7e5017-f07f-40cb-81f9558&old_name=od.txt&new_name=new.txt&dpath=/g/gg`
    - ***HTTP metoda vůči cloud serveru:*** POST
    - ***Parametry:***
        - ◇ repo – jedná se o jednoznačný identifikátor knihovny
        - ◇ old_name – název souboru, který je v dané knihovně uložen
        - ◇ new_name – nový název souboru
        - ◇ dpath – absolutní cesta k souboru old_name
    - ***Formát odpovědi:*** JSON
    - ***Odpověď:*** V případě úspěchu vrácen řetězec „success“.
13. **Přesunutí souboru**
    - ***Popis chování:*** Přesune zadaný soubor, který je uložen na vzdáleném cloud serveru.
    - ***Příklad volání API:*** `http://www.servername.com/cloudAPI/move/move_file.php?repo=2f7e5017-f07f-40cb-81f9-dsd&filename=file.html&odpath=/scripts&ndpath=/`
    - ***HTTP metoda vůči cloud serveru:*** POST
    - ***Parametry:***
        - ◇ repo – jedná se o jednoznačný identifikátor knihovny
        - ◇ filename – název souboru, který je v dané knihovně uložen
        - ◇ odpath – absolutní cesta ke složce, ve které je soubor uložen
        - ◇ ndpath – absolutní cesta, do které má být soubor přesunut
    - ***Formát odpovědi:*** JSON
    - ***Odpověď:*** V případě úspěchu vrácen řetězec „success“.
14. **Získání výpisu složky**
    - ***Popis chování:*** Vrátí výpis obsahu zadané složky, jež je uložena na vzdáleném cloud serveru.
    - ***Příklad volání API:*** `http://www.servername.com/cloudAPI/get/get_list_dir.php?repo=2f7e5017-f07f-40cb&dpath=/scripts`
    - ***HTTP metoda vůči cloud serveru:*** POST

- ***Parametry:***
  - ◇ repo – jedná se o jednoznačný identifikátor knihovny
  - ◇ dpath – absolutní cesta ke složce (včetně jejího názvu), jejíž obsah má být vypsán
- ***Formát odpovědi:*** JSON
- ***Odpověď:*** V případě úspěchu je vrácen detailní výpis obsahu složky.

15. **Stažení obsahu složky**

- ***Popis chování:*** Vytvoří webový odkaz ke stažení celého obsahu složky, která je umístěna na vzdáleném cloud serveru.
- ***Příklad volání API:*** `http://www.servername.com/cloudAPI/download/down_dir.php?repo=2f7e5017-f07f-40cb-81f9&dpath=/scripts`
- ***HTTP metoda vůči cloud serveru:*** POST
- ***Parametry:***
  - ◇ repo – jedná se o jednoznačný identifikátor knihovny
  - ◇ dpath – absolutní cesta ke složce (včetně jejího názvu), která má být stažena
- ***Formát odpovědi:*** JSON
- ***Odpověď:*** V případě úspěchu vrátí URL link, kde je po omezený čas obsah dané složky dostupný ke stažení ve formátu ZIP. Například: „`http://195.113.232.52:8082/files/84bb7b14-c0ad-44c4-b1d/scripts`“.

16. **Stažení souboru**

- ***Popis chování:*** Vytvoří webový odkaz ke stažení souboru, který je umístěn na vzdáleném cloud serveru.
- ***Příklad volání API:*** `http://www.servername.com/cloudAPI/download/down_file.php?repo=2f7e5017-f07f-40cb-81f9-d4f0c18d8&dpath=/scripts&filename=file.php`
- ***HTTP metoda vůči cloud serveru:*** POST
- ***Parametry:***
  - ◇ repo – jedná se o jednoznačný identifikátor knihovny
  - ◇ dpath – absolutní cesta k souboru uloženého v dané knihovně
  - ◇ filename – název souboru, který má být stažen, a to včetně jeho přípony
- ***Formát odpovědi:*** JSON
- ***Odpověď:*** V případě úspěchu vrátí URL link, kde je po omezený čas daný soubor dostupný ke stažení. Například: „`http://195.113.232.52:8082/files/1f43290-c51c-485-b5/file.php`“.

17. **Nahrání souboru**

- ***Popis chování:*** Nahraje nový soubor na vzdálený cloud server.
- ***Příklad volání API:*** `http://www.servername.com/cloudAPI/upload/upload_file.php?repo_token=76d114c9-4126-44975&pathfile=/var/www/html/index.html&dpath=/aa`
- ***HTTP metoda vůči cloud serveru:*** POST
- ***Parametry:***
  - ◇ repo_token – jedná se o token, který slouží k nahrávání souborů do dané knihovny
  - ◇ pathfile – jedná se o absolutní cestu k nahrávanému souboru
  - ◇ dpath – absolutní cesta ke složce, do níž má být soubor na cloud serveru nahrán
- ***Formát odpovědi:*** JSON
- ***Odpověď:*** V případě úspěchu vrátí jednoznačný identifikátor (file_id) nahraného souboru. Například: „4ef3069d22967d298534fe98“.

18. **Vygenerování náhledu z obrázku (změna velkosti)**

- ***Popis chování:*** Vygeneruje náhled z obrazového souboru o zadané výšce a šířce.
- ***Příklad volání API:*** `http://www.servername.com/cloudAPI/generate/gen_thumb_image.php?pathfile=/var/www/html/to_do.jpg&width=1024&height=768`
- ***Parametry:***
  - ◇ pathfile – jedná se o absolutní cestu k obrazovému souboru, ze kterého má být vytvořena jeho zmenšenina
  - ◇ width – šířka finální zmenšeniny
  - ◇ height – výška finální zmenšeniny
- ***Výstup:*** Bude vytvořena zmenšenina o zadané výšce a šířce. Zmenšenina bude uložena ve složce, ve které byl umístěn původní soubor. Její pojmenování se řídí pravidlem: „původní název_thumbnail_widthxheight.přípona“.
- ***Formát odpovědi:*** JSON
- ***Odpověď:*** V případě úspěchu vrátí řetězec „Image file: název zmenšeniny successfully created.“. Například: „Image file: car_thumbnail_1024x768.jpg successfully created.“.

19. **Vygenerování náhledu z PDF souboru**

    - ***Popis chování:*** Vygeneruje náhled ve formě obrazového souboru o zadané výšce a šířce z PDF dokumentu, tj. provede se konverze.
    - ***Příklad volání API:*** `http://www.servername.com/cloudAPI/generate/gen_thumb_pdf.php?pathfile=/var/www/html/soubor.pdf&width=1024&height=768&oformat=png&page=2`
    - ***Parametry:***
        - ◇ pathfile – jedná se o absolutní cestu k PDF souboru, ze kterého má být vytvořen náhled
        - ◇ width – šířka finálního obrazového náhledu
        - ◇ height – výška finálního náhledu
        - ◇ oformat – formát obrazového náhledu
        - ◇ page – stránka PDF dokumentu, ze které se vytvoří náhled
    - ***Výstup:*** Bude vytvořen náhled o zadané výšce a šířce. Náhled bude uložen ve složce, ve které byl umístěn původní PDF soubor. Jeho pojmenování se řídí pravidlem: „původní název_thumbnailpg_page_ widthxheight.oformat“.
    - ***Formát odpovědi:*** JSON
    - ***Odpověď:*** V případě úspěchu vrátí řetězec „Image file: název náhledu successfully created.“. Například: „Image file: info_thumbnailpg_ 2_1024x768.png successfully created.“.

20. **Vygenerování náhledu z video souboru**

    - ***Popis chování:*** Vygeneruje náhled o zadané výšce a šířce z daného video souboru, tj. provede se konverze.
    - ***Příklad volání API:*** `http://www.servername.com/cloudAPI/generate/gen_thumb_video.php?pathfile=/var/www/html/lec1.mp4&width=1024&height=768&oformat=jpg&frame=22`
    - ***Parametry:***
        - ◇ pathfile – jedná se o absolutní cestu k video souboru, ze kterého má být vytvořen náhled
        - ◇ width – šířka finálního obrazového náhledu
        - ◇ height – výška finálního náhledu
        - ◇ oformat – formát obrazového náhledu
        - ◇ frame – snímek video souboru, ze kterého se vytvoří náhled
    - ***Výstup:*** Bude vytvořen náhled o zadané výšce a šířce. Náhled bude uložen ve složce, ve které byl umístěn původní video soubor. Jeho pojmenování se řídí pravidlem: „původní název_thumbnailfr_frame_ widthxheight.oformat“.

- ***Formát odpovědi:*** JSON
- ***Odpověď:*** V případě úspěchu vrátí řetězec „Image file: název náhledu successfully created.“. Například: „Image file: lec1_thumbfr_22_1024x768.jpg successfully created.“.

5.2.1.1 Stavové kódy

Na uživatelův požadavek ještě v relevantních případech Web API odpovídá příslušnými HTTP kódy, které specifikují, zda byl požadavek proveden úspěšně, nebo skončil nějakou chybou.[21]

Tabulka 5.1: Seznam HTTP stavových kódů, které jsou API vraceny

Číslo HTTP kódu	Význam
200	OK
201	CREATED
202	ACCEPTED
301	MOVED PERMANENTLY
400	BAD REQUEST
403	FORBIDDEN
404	NOT FOUND
409	CONFLICT
429	TOO MANY REQUESTS
440	REPO PASSWD REQUIRED
441	REPO PASSWD MAGIC REQUIRED
500	INTERNAL SERVER ERROR
520	OPERATION FAILED

[21] Více informací o stavových kódech HTTP lze nalézt například na: `http://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml`

Kapitola 6

Testování

V této kapitole je popsán způsob testování implementovaného řešení. Nejprve je vysvětleno, jak testování probíhalo a na jaké konkrétní testy bylo zaměřeno. Poté následuje zhodnocení výsledků celého procesu testování, které se zaměřuje především na fakt, zda byly během testování nalezeny chyby. Pokud ano, jak byly závažné a zároveň zda byly odstraněny.

6.1 Průběh testování

Testování se soustředilo především na následující dva typy testů:

1. **Testy základní funkčnosti**

   V rámci těchto testů byly otestovány jednotlivé funkce implementovaného RESTful Web API. Cílem testů základní funkčnosti bylo zjistit, zda jednotlivé funkce správně vykonávají svou deklarovanou úlohu. Zaměřovalo se tedy na dílčí funkce API, které byly podrobně specifikovány ve funkčních požadavcích, a proto byly testy pokryty všechny jednotlivé funkce. Typickým zástupcem z této kategorie je například test se scénářem: „*Smažte složku s absolutní cestou /scripts/new_scripts/, která je umístěná v knihovně s ID 75f8ebbe-e326-4021-a75d-ac846b2a6db8.*“. Scénáře ke všem provedeným testům, a to včetně jejich výsledků, lze najít v příloze **A Testování základní funkčnosti**.

2. **Testy zaměřené na spolehlivost a stabilitu**

   Testy spadající do této kategorie se zaměřovaly na otestování spolehlivosti dílčích funkcí implementace vůči nečekaným vstupům a nečekané (limitní) zátěži. Cílem těchto testů bylo zjistit, zda API správným způsobem reaguje na záměrně špatné vstupy, události a zda je všeobecně stabilní. Typickými příklady testů spadajících do této třídy jsou: vytváření mnoha nových složek na cloud serveru s rozsáhlými názvy, mazání mnoha neexistujících složek, provádění akcí na serveru pomocí špatného

autentizačního tokenu, přesunutí neexistujícího souboru, generování náhledů z neexistujícího souboru a tak dále. Při tomto testování byly využity linuxové shell příkazy pro zjednodušení pracnosti testování.

6.2 Shrnutí výsledků testování

Celý proces testování byl nezávisle opakován v době vytváření implementace tak, aby byla zvýšena jeho efektivita. Během provádění testů základní funkčnosti nebyly nalezeny žádné chyby a testy dopadly dle očekávání.

Testy zaměřené na spolehlivost a stabilitu vůči limitním a nečekaným vstupům či událostem dopadly téměř pozitivně, avšak během provádění byly nalezeny chyby v následujících funkčnostech:

1. Vygenerování náhledu z obrázku (změna velkosti) – Pokud byl na vstup zadán neexistující soubor, ze kterého měla být vygenerována zmenšenina, pak implementace tento druh chyby nedetekovala a provedla neplatný požadavek. O provedení neplatného požadavku nebyl rovněž informován ani uživatel ve výstupní JSON zprávě.
2. Vygenerování náhledu z PDF souboru – Analogicky stejný druh chyby jako u prvního bodu.
3. Vygenerování náhledu z video souboru – Analogicky stejný druh chyby jako u prvního a druhého bodu.

Nalezené chyby nebyly zásadního charakteru a neovlivňovaly funkčnost implementace jako celku. V případě chybného vstupu byla způsobena nefunkčnost pouze třech výše uvedených funkcí implementace. Všechny nalezené nedostatky byly samozřejmě opraveny, následně byly znovu provedeny testy, aby bylo zjištěno, zda chyby byly opraveny správným způsobem, a také zda se jejich odstraněním nezanesly do systému chyby další. Díky tomu lze systém nyní již plnohodnotně využívat a je zcela připraven k nasazení.

Závěr

Cílem této bakalářské práce bylo nejprve vybudovat vhodné cloudové úložiště, a poté implementovat backend formou RESTful Web API, které bude poskytovat rozhraní pro nahrávání, správu a konverzi multimediálních datových souborů nad vytvořeným cloudovým úložištěm.

Na základě detailně provedených analýz a rešerší se podařilo navrhnout takové řešení, které 100% splňuje nároky na něho kladené. Úspěšně byly implementovány všechny stanovené funkční a nefunkční požadavky, a to dokonce i s mírným přesahem. Celé API bylo navrhnuto a implementováno takovým způsobem, aby bylo možné ho v budoucnu co nejsnadněji rozšiřovat o nové funkcionality. Během vývoje byl kladen důraz i na zevrubné a intenzivní testování, díky tomu bylo také odhaleno a opraveno několik chyb v API. Následně po úspěšném otestování bylo API nasazené na server.

V budoucnu je možné na práci určitě smysluplně navázat. Nabízí se možnost rozšířit jednak samotný self-hosted cloudový server o nové funkce, jakými jsou například: podpora protokolu HTTPS, autentizace uživatelů přes SSO protokol či LDAP a další. Rovněž je možné rozšířit funkcionality samotného RESTful Web API například v těchto oblastech: správa a sdílení souborů, práce s historií a verzemi souborů, šifrování a také v oblasti řízení přístupů uživatelů.

Zpracování a realizace této bakalářské práce pro mě bylo velkým přínosem. Dozvěděl jsem se mnoho nových informací zejména z oblastí: vizualizace pomocí zařízení s velmi vysokým rozlišením, softwaru SAGE2$^{\text{TM}}$, Cloud computingu, tvorby cloudových úložišť, architektury REST a návrhu (Web) API. Zároveň jsem si prohloubil své dosavadní vědomosti o skriptovacím jazyku PHP 5, formátu pro uložení dat JSON, HTTP(S) protokolu a administraci webového serveru pod OS Linux. Všechny nabyté znalosti jistě zúročím i v budoucích projektech.

Literatura

[1] Budín, E.: *Cloud computing se zaměřením na dostupnost a bezpečnost dat.* Bakalářská práce, Masarykova univerzita v Brně. Fakulta filozofická. Kabinet informačních studií a knihovnictví, 2012 [cit. 2015-04-10]. Dostupné z: `http://is.muni.cz/th/212378/ff_b/bakalarska_prace.pdf`

[2] Souček, A.: *Zajištění dodávek aplikací a dat přes Internet.* Diplomová práce, Univerzita Palackého v Olomouci. Právnická fakulta, 2013 [cit. 2015-04-10]. Dostupné z: `https://theses.cz/id/3ljymq`

[3] Mácha, P.: Cloud computing – historie a budoucnost. Dimension data magazín. [online], [cit. 2015-04-18]. Dostupné z: `http://www.ddconnect.cz/brezen-2012/datova-centra.html`

[4] Setíkovská, B.: *Cloud Computing.* Diplomová práce, České vysoké učení technické v Praze. Fakulta elektrotechnická. Katedra řídicí techniky, 2010 [cit. 2015-04-10]. Dostupné z: `https://support.dce.felk.cvut.cz/mediawiki/images/c/c8/Dp_2010_setikovska_blanka.pdf`

[5] Meloun, J.: *Přínosy a omezení Cloud Computingu.* Bakalářská práce, Vysoká škola ekonomická v Praze. Fakulta informatiky a statistiky. Katedra informačních technologií, 2012 [cit. 2015-04-10]. Dostupné z: `http://www.vse.cz/vskp/32739_prinosy_aomezeni_cloud_computingu`

[6] Seafile: Seafile. Seafile official website. [online], [cit. 2015-04-26]. Dostupné z: `http://seafile.com/en/product/private_server`

[7] ownCloud: ownCloud features. ownCloud official website. [online], [cit. 2015-04-26]. Dostupné z: `https://owncloud.org/features`

[8] Pydio: What is Pydio? Pydio official website. [online], [cit. 2015-04-26]. Dostupné z: `https://pyd.io/about`

[9] Ind.ie: Ind.ie Labs. Ind.ie official website. [online], [cit. 2015-04-26]. Dostupné z: http://labs.ind.ie

[10] Lee, B.: 2015 Best Cloud Computing Services Comparisons and Review. Top ten reviews website. [online], [cit. 2015-04-26]. Dostupné z: http://cloud-services-review.toptenreviews.com

[11] ThePHPGroup: What is PHP? The PHP Group official website. [online], [cit. 2015-04-26]. Dostupné z: http://php.net/manual/en/intro-whatis.php

[12] Zajíc, P.: Historie jazyka PHP. Linuxsoft.cz. [online], [cit. 2015-04-26]. Dostupné z: http://www.linuxsoft.cz/article.php?id_article=171

[13] ThePHPGroup: Download PHP. The PHP Group official website. [online], [cit. 2015-04-26]. Dostupné z: http://php.net/downloads.php

[14] Hassman, M.: JSON: jednotný formát pro výměnu dat. Zdrojak.cz. [online], 29.9.2008 [cit. 2015-04-26]. Dostupné z: http://www.zdrojak.cz/clanky/json-jednotny-format-pro-vymenu-dat

[15] JSON.org: Úvod do JSON. JSON.org. [online], [cit. 2015-04-26]. Dostupné z: http://www.json.org/json-cz.html

[16] Kay, R.: Representational State Transfer (REST). Computerworld.com. [online], 6.8.2007 [cit. 2015-04-27]. Dostupné z: http://www.computerworld.com/article/2552929/networking/representational-state-transfer--rest-.htmll

[17] Malý, M.: REST: architektura pro webové API. Zdrojak.cz. [online], 3.8.2009 [cit. 2015-04-27]. Dostupné z: http://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api

[18] Elkstein, M.: Learn REST: A Tutorial. rest.elkstein.org. [online], [cit. 2015-04-27]. Dostupné z: http://rest.elkstein.org

Příloha **A**

Testování základní funkčnosti

Tato příloha obsahuje všechny scénáře **testů základní funkčnosti**, a to včetně jejich výsledků. Postup celého procesu testování je podrobně popsán v kapitole číslo 6.

A.1 Testovací scénáře

1. Zjistěte, zda je autentizační token, který je uložen v konfiguračním souboru implementace, platný pro komunikaci s RESTful Web API cloud serveru.
2. Vygenerujte autentizační token pro komunikaci s Web API cloud serveru uživateli root@sagelab.cz s heslem 12345.
3. Vygenerujte autentizační token pro nahrávání souborů do knihovny s ID 75f8ebbe-e326-4021-a75d-ac846b2a6db8.
4. Získejte informace o souboru seafile-tutorial.doc, který je umístěn v kořenovém adresáři knihovny s ID 75f8ebbe-e326-4021-a75d-ac846b2a6db8.
5. Smažte soubor s názvem cti.txt, který je umístěn v adresáři /scripts/ knihovny s ID 75f8ebbe-e326-4021-a75d-ac846b2a6db8.
6. Vytvořte novou knihovnu s názvem Nova_knihovna.
7. Smažte knihovnu s ID 2f7e5017-f07f-40cb-81f9-d4f0c18d865f (knihovna výše vytvořená).
8. Smažte složku s absolutní cestou /scripts/new_scripts/, která je umístěna v knihovně s ID 75f8ebbe-e326-4021-a75d-ac846b2a6db8.
9. Vytvořte novou složku s názvem new_scripts ve složce s absolutní cestou /scripts/ v knihovně s ID 75f8ebbe-e326-4021-a75d-ac846b2a6db8.
10. Přejmenujte složku s absolutní cestou /scripts/new_scripts/, která je umístěna v knihovně s ID 75f8ebbe-e326-4021-a75d-ac846b2a6db8 na složku s novým názvem new_script.

11. Nahrajte do kořenového adresáře knihovny s ID 75f8ebbe-e326-4021-a75d-ac846b2a6db8 soubor s názvem index.html, který je na vašem PC uložen na cestě /var/www/html/index.html.
12. Přejmenujte soubor index.html, který je nahraný v kořenovém adresáři knihovny s ID 75f8ebbe-e326-4021-a75d-ac846b2a6db8 na soubor s novým názvem index.php.
13. Přesuňte soubor index.php, který je nahraný v kořenovém adresáři knihovny s ID 75f8ebbe-e326-4021-a75d-ac846b2a6db8 do složky s názvem new_script, jejíž absolutní cesta v knihovně je /scripts/new_script/.
14. Získejte výpis obsahu složky pojmenované scripts s absolutní cestou /scripts/, která je umístěna v knihovně s ID 75f8ebbe-e326-4021-a75d-ac846b2a6db8.
15. Vygenerujte URL odkaz ke stažení celého obsahu složky pojmenované scripts s absolutní cestou /scripts/, která je umístěna v knihovně s ID 75f8ebbe-e326-4021-a75d-ac846b2a6db8.
16. Vygenerujte URL odkaz ke stažení souboru pojmenovaného index.php s absolutní cestou /scripts/new_script/, který je umístěn v knihovně s ID 75f8ebbe-e326-4021-a75d-ac846b2a6db8.
17. Vygenerujte zmenšeninu (náhled) z obrázku pojmenovaného minions.jpg s absolutní cestou /var/www/html/, který je uložen na serveru, na kterém běží vytvořené RESTful cloud Web API. Výsledná zmenšenina bude mít šířku 1024 pixelů a výšku 768 pixelů.
18. Vygenerujte náhled z druhé stránky PDF souboru pojmenovaného do.pdf s absolutní cestou /var/www/html/, který je umístěn na serveru, na kterém běží vytvořené RESTful cloud Web API. Výsledný náhled bude mít šířku 1024 pixelů, výšku 768 pixelů a jeho výstupní formát bude PNG.
19. Vytvořte náhled z 222. snímku video souboru pojmenovaného lec01.mp4 s absolutní cestou /var/www/html/, který je uložen na serveru, na kterém běží vytvořené RESTful cloud Web API. Výsledný náhled bude mít šířku 1024 pixelů, výšku 768 pixelů a jeho výstupní formát bude JPG.

A.2 Výsledky testovacích scénářů

1. pong
2. {"token": "a527c6e26ac5c434d64c347427312e181b57ac84"}
3. 3a983c26-4bd3-45f5-a24a-bada024f3b5b
4. {"id": "b88ab96740ef53249b9d21fb3fa28050842266ba", "mtime": 142682, "type": "file", "name": "seafile-tutorial.doc", "size": 300544}
5. success
6. {"encrypted": "",
   "enc_version": 0,
   "repo_id": "ab7901e6-c0fb-4599-8a18-48784d315865",
   "magic": "",
   "relay_id": "e961e17589acbdf1af1031011dddcdeb5f0c8541",
   "repo_version": 1,
   "relay_addr": "195.113.232.52",
   "token": "e5b6e22ae002221492cdabf23f6978b2bcc0ba39",
   "relay_port": "10001",
   "random_key": "",
   "email": "root@sagelab.cz",
   "repo_name": "Nova_knihovna"}
7. success
8. success
9. success
10. success
11. 4ef3069d22967d298534fe98b8b956c5cdd98927
12. success
13. success
14. [{"mtime": 1428322330, "type": "dir", "name": "new_script", "id": "ba52b53d4e781711ceeb451c51e65d651ce69311"}]
15. http://195.113.232.52:8082/files/822a-0ad-461-9d3-846320a/scripts
16. http://195.113.232.52:8082/files/b677-65d-44a-a2c-407c561/index.php
17. Image file: minions_thumbnail_1024x768.jpg successfully created.
18. Image file: do_thumbnailpg_2_1024x768.png successfully created.
19. Image file: lec01_thumbfr_222_1024x768.jpg successfully created.

Příloha B

Seznam použitých zkratek

BP	Bakalářská práce
FIT	Fakulta informačních technologií
ČVUT	České vysoké učení technické
SAGE	Scalable adaptive graphics environment
REST	Representational state transfer
API	Application programming interface
PDF	Portable document format
CC	Cloud computing
IT	Informační technologie
NIST	National institute of standards and technology
MIT	Massachusetts institute of technology
ICT	Information and communication technologies
SW	Software
IaaS	Infrastructure as a service
PaaS	Platform as a service
SaaS	Software as a service
GUI	Graphical user interface
JSON	JavaScript object notation
URL	Uniform resource locator

URI	Uniform resource identifier
PC	Personal computer
PNG	Portable network graphics
OS	Operační systém
RAM	Random-access memory
HDD	Hard disk drive
PHP	Hypertext preprocessor
XML	Extensible markup language
HTTP	Hypertext transfer protocol
HTTPS	Hypertext transfer protocol secure
CRUD	Create, read, update, delete
W3C	World wide web consortium
SSO	Single sign-on
LDAP	Lightweight directory access protocol
SQL	Structured Query Language

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace RESTful Web API
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `BP_Svoboda_Roman_2015.pdf` ..... text práce ve formátu PDF