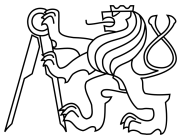
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Soukromé vzdálené úložiště se sdílením souborů
Student:	Jan Zípek
Vedoucí:	Ing. Miroslav Balík, Ph.D.
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Navrhněte a implementujte službu, která bude umožňovat provozovat soukromé vzdálené úložiště. Služba bude obsahovat API, pomocí kterého bude probíhat veškerá funkcionalita. Služba musí umožňovat:
- nahrávání souborů,
- stahování souborů,
- přesouvání vzdálených souborů ve vzdáleném úložišti,
- ukládání více verzí jednoho souboru,
- sdílení souborů,
- šifrování souborů na straně klienta,
- víceuživatelský přístup.

Serverovou část služby bude možno nainstalovat na webhostingovém serveru bez přístupu ke správě operačního systému.
Dále za použití tohoto API vytvořte nativní aplikaci, která bude využívat veškerou jmenovanou funkcionalitu, včetně šifrování souborů před odesláním na server. Aplikaci koncipujte jako multiplatformní.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 19. prosince 2015

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

Bakalářská práce

Soukromé vzdálené úložiště se sdílením souborů

Jan Zípek

Vedoucí práce: Ing. Miroslav Balík, Ph.D.

15. května 2016

Poděkování

Děkuji mým rodičům za nepřežitou podporu, děkuji mému vedoucímu za pomoc při vypracování práce a děkuji všem kantorům za poskytnuté vzdělání a podporu.

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 15. května 2016 .....

Odkaz na tuto práci

Zípek, Jan. *Soukromé vzdálené úložiště se sdílením souborů*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Bakalářská práce popisuje vývoj služby vzdáleného úložiště. Cílem bylo vytvořit bezpečnou službu vzdáleného úložiště s jednoduchou serverovou částí a klientem s podporou OS GNU/Linux a Microsoft Windows. Služba byla analyzována, následně byly jednotlivé komponenty navrženy a implementovány. Cíl byl splněn a služba byla publikována na platformě Github.

Klíčová slova Java, služba, vzdálené úložiště, cloud, šifrování souborů, PHP

Abstract

Thesis is focused on developing remote storage service. Goal was to create safe remote storage service with simple server side and client supporting OS GNU/Linux and Microsoft Windows. Service was analyzed and each component was then designed and implemented. The goal was met and service was presented at Github platform.

Keywords Java, service, remote storage, cloud, file encryption, PHP

Obsah

Úvod **1**

1 Cíl práce **3**

2 Analýza **5**
2.1 Cílová skupina 5
2.2 Současná řešení 5
2.3 Šifrování 9
2.4 Funkční požadavky 11
2.5 Nefunkční požadavky 11

3 Návrh **13**
3.1 Případy užití 13
3.2 Šifrování 15
3.3 Synchronizace 17
3.4 Starší verze souborů 17
3.5 Síťové API 18
3.6 Uživatelské rozhraní 19

4 Implementace **23**
4.1 Server 23
4.2 Klient 27
4.3 Instalace serveru 32

Závěr **35**
Budoucí rozvoj 35

Literatura **37**

A Seznam použitých zkratek **41**

B Obsah přiloženého CD **43**

C Uživatelská příručka **45**
C.1 Úvod 45
C.2 Požadavky 45
C.3 Instalace 45
C.4 První spuštění 45
C.5 Aplikace 46

D Správcovská příručka **51**
D.1 Instalace serveru 51
D.2 Správa uživatelů 51

Seznam obrázků

3.1 Návrh rozhraní hlavního okna. 22

4.1 Diagram tříd serveru. 26
4.2 Schéma databáze meta dat. 26
4.3 Ukázka mockup aplikace na OS Android. 31

C.1 Hlavní okno aplikace. 46
C.2 Dialog pro nahrávání souborů. 47
C.3 Dialog detailu souboru. 48
C.4 Výpis synchronizovaných složek. 49
C.5 Dialog nastavení synchronizované složky. 49

D.1 Instalace serveru. 52

Seznam tabulek

2.1 Porovnání existujících řešení 9

3.1 Pokrytí funkčních požadavků případy užití 15

Úvod

Ukládání dat na vzdálené úložiště je dnes naprosto běžnou činností, která není vyhrazena pouze pro vývojáře. Lidé si na vzdálené úložiště ukládají fotky z dovolené nebo různé důležité dokumenty, o které nechtějí přijít. Díky popularitě této služby vzniklo a existuje množství komerčních řešení.

V naprosté většině komerčních řešení vzdáleného úložiště jsou data ukládána na server v čisté formě a jste proto nuceni spolehnout se na zabezpečení poskytovatele služby. Nehledě na velikost a popularitu služby se ovšem nelze spolehnout na její bezpečnost.

I proto existuje speciální větev těchto služeb, takzvaná self-hosted úložiště. Od komerčních řešení se liší tím, že mají otevřený zdrojový kód a umožňují uživatelům nainstalovat si službu na svůj vlastní server a spoléhat se tak pouze na vlastní zabezpečení. Vzhledem k otevřeným zdrojovým kódům si uživatel navíc může bezpečnost služby sám ověřit. Ne každá z těchto existujících služeb ovšem umožňuje data dostatečně bezpečně šifrovat. Navíc mají existující služby specifické nároky na server, čímž značně omezují počet možných instalací a ne každá služba hledí na uživatelskou přívětivost samotného nastavení nové instalace serveru.

Tento projekt bude oproti existujícím řešením vynikat nejen bezpečností, ale i kompatibilitou s existujícími serverovými instalacemi, tedy i s webhostingy, a orientací na jednoduchost a kompaktnost instalace.

Práce se nejprve zaměří na analýzu existujících řešení a poté se bude zabývat vývojem samotné služby. Popíše uživateli metodiku, pomocí které se došlo k řešení jednotlivých funkčností a se kterou se dosáhne bezpečnosti ukládaných dat. V poslední řadě pak popíše, jak autor přistoupil k její implementaci a jak danou implementaci testoval.

Kapitola 1

Cíl práce

Cílem práce je vytvoření služby umožňující jednoduchou instalaci a provoz self-hosted služby pro nahrávání a sdílení souborů. Služba se bude skládat ze síťového API a klienta pro Microsoft Windows a GNU/Linux, který bude umožňovat nahrávání, stahování, prohlížení, přesouvání a mazání souborů. Navíc bude umožňovat synchronizaci souborů s lokální kopií, a to včetně uchovávání starších verzí. Služba bude jednoduchá pro instalaci a nenáročná, aby ji bylo možné instalovat i na webhostingovém serveru.

Kapitola 2

Analýza

Analýza je základní kámen vývoje každé aplikace. Je potřeba zhodnotit existující řešení a jejich výhody a nevýhody, pomocí čehož dojdeme k funkčnostem, které naší aplikaci učiní konkurenceschopnou a zajímavou pro naši cílovou skupinu. Dále je pak potřeba zjistit, jak nejlépe zajistit bezpečnost ukládaných dat. Následně můžeme určit požadavky na službu.

2.1 Cílová skupina

Tato služba se zaměřuje na pokročilejší uživatele, kteří mají znalosti práce na počítači a určité zkušenosti se správou serveru. Půjde o uživatele, kteří si jsou vědomi nebezpečnosti veřejných služeb a chtějí dosáhnout bezpečnosti svých dat, nebo si pouze chtějí jednoduše a rychle zřídit vzdálené úložiště.

Klientská aplikace pak bude zaměřena i na středně pokročilé uživatele se základní znalostí práce na počítači. Vzhledem k podpoře více uživatelů bude tudíž pokročilejší uživatel moci sdílet své úložiště s předem zvolenou skupinou lidí, kteří pak budou přistupovat pouze ke klientské aplikaci, jež už nebude vyžadovat pokročilé znalosti.

2.2 Současná řešení

V současné době existuje několik řešení, která mohou poskytovat stejné možnosti. Některá se orientují na týmovou kolaboraci, některá pouze na ukládání souborů, další na velké servery. V této sekci se budeme zabývat právě takovými službami a jejich rozdíly oproti službě popsané v této práci. V potaz bereme jen nepopulárnější OpenSource řešení. [1, 2, 3].

2.2.1 File transfer protocol a SSH file transfer protocol

File transfer protocol (dále jen FTP) a SSH file transfer protocol (dále jen SFTP) jsou nejjednodušší řešení problému. Díky rozšířenosti těchto protokolů

existuje velké množství aplikací, které s nimi pracují.

Pomocí existujících nativních aplikací je tedy možné provádět většinu vyžadovaných operací, a to včetně synchronizace lokálních souborů s těmi vzdálenými [2]. SFTP navíc zabezpečuje šifrování přenosu, není tedy možné odposlouchávat obsah souborů při přenášení.[4]

FTP ani SFTP nezajišťují sdílení. Je sice možné přes tyto protokoly sdílet vybrané složky, ale to vyžaduje, aby uživatel, s nímž soubory sdílíte, měl nainstalovanou aplikaci podporující dané protokoly. Navíc byste tak uživateli dávali přístup k celému úložišti a ne jednomu určitému souboru. Protokoly nabízejí určitou formu kontroly práv k přístupům, ale pro toto použití jsou značně nepraktické.

Ani jeden z těchto protokolů navíc nezajišťuje šifrování souborů na straně klienta, které je důležité pro zabezpečení souborů na serveru. Pokud nejsou soubory šifrované už na straně klienta, je možné je na straně serveru dešifrovat, protože se tam musí nacházet i klíč.

V poslední řadě se v těchto protokolech nepočítá s verzováním souborů. Umožňují vždy uložení pouze jedné verze souboru, jediné řešení by bylo verzování pomocí přejmenovávání starších verzí souboru.

2.2.2 Verzovací systémy

Pro účel synchronizace dat se vzdáleným serverem lze použít i verzovací systémy, například Git nebo Apache Subversion. Tyto systémy jsou orientované na textová data, ale dovedou pracovat i s binárními.[5]

Vzhledem k jejich orientaci na zdrojové kódy ovšem nejsou příliš uživatelsky příjemné, kromě toho umožňují opravdu jen synchronizaci souborů se vzdáleným úložištěm. Veřejné sdílení souborů je možné pouze rozšířením o další služby. Taková služba by ale navíc musela řešit problém autentizace a procházení souborů. Uživatelé musí odpovídat uživatelům na serveru, pro jejich správu je tudíž potřeba zkušenějšího správce.

Serverová část těchto systémů je realizována pomocí nativních aplikací, což v kombinaci s nutností používání systémových uživatelů znamená, že je možné instalovat je pouze na servery s plným přístupem.

U verzovacích systémů musíte synchronizovat vždy celé úložiště, navíc není synchronizace automatická a musí se spouštět ručně.

Oproti těmto řešením nabízí služba popsaná v této práci pohodlnou jednoduchou instalaci, podporu web hostingových serverů a nativní aplikaci pro synchronizaci a práci se soubory. Synchronizace je navíc jednoduší, stačí nastavit, jakou lokální složku synchronizovat s tou vzdálenou.

2.2.3 OwnCloud

Tato služba je přímo určená k zálohování a sdílení souborů. Je publikována pod licencí AGPL, která se od GPL liší dodatkem, jenž každé veřejně provozované

derivaci vynucuje publikaci zdrojového kódu.[6]

Pokud je o požadavky na služby, tato poskytuje většinu potřebných funkcionalit. Nahrávání jednotlivých souborů, procházení adresářové struktury bez její stažení, mazání a sdílení je možné pouze přes webové rozhraní.

Šifrování tato služba podporuje pouze jako rozšíření, které však funguje jen na straně serveru. Data jsou proto nešifrovaně přenášena od klienta na server a šifrována jsou až na straně serveru, takže šifrovací klíč je uložen pouze na straně serveru. To umožňuje správci serveru nebo potenciálnímu útočníkovi soubory dešifrovat.[7]

Služba je doplněna o nativního klienta podporujícího operační systémy Microsoft Windows, GNU/Linux a Mac OS. Klient se ale stará pouze o synchronizaci složek, neumožňuje procházení souborů bez jejich stažení ani jejich mazání a přejmenování a vůbec neumožňuje jejich sdílení.

Synchronizace je základní, umožňuje nastavit globální filtr názvů souborů, které se nemají synchronizovat a omezit velikost synchronizovaných souborů. Tyto možnosti jsou pouze globální a uživatel si je nemůže nastavit pro jednotlivé složky.

Pro komunikaci s klientem služba používá protokol WebDAV, úpravu běžně používaného HTTP [8]. Od HTTP se liší dostatečně na to, aby bylo potřeba používat vlastní implementaci komunikace, což znemožňuje využití tohoto protokolu v čistě webově orientovaných aplikacích.

Služba má vysoké nároky na hardware a značně omezující nároky na serverové prostředí [9]. Instalace této služby je náročná a vyžaduje pokročilé znalosti problematiky. Kromě jmenovaných možností navíc obsahuje i další, pro tento problém nedůležité služby (kalendář, kontakty a přehrávání hudby), které dělají řešení zbytečně složitější.

Služba popsaná v této práci nabízí nativního klienta, který kromě synchronizace umožňuje soubory i procházet, přemisťovat, mazat a sdílet bez nutnosti stažení všech souborů na disk. Navíc podporuje šifrování na straně klienta, které zajišťuje, že soubory na server chodí již zašifrované a na serveru se nenachází žádné indicie umožňující soubory dešifrovat. V poslední řadě nabízí jednoduchou instalaci a má příznivější požadavky na server, umožňující instalaci na naprosté většině serverů podporujících PHP.

2.2.4 SparkleShare

Tato služba má podobně jako OwnCloud naprostou většinu požadovaných funkcionalit. Umožňuje synchronizaci souborů, ukládá různé verze souborů a poskytuje jednoduše ovladatelné webové rozhraní pro správu úložiště.

Služba má i nativního klienta, který umožňuje provádět synchronizaci nejen se SparkleShare úložišti, ale s jakýmkoli existujícím Git repozitářem.

Pro ukládání a přenos souborů používá verzovací systém Git, z čehož pro ni plyne omezení. Nezvládá synchronizaci adresářových struktur, které obsahují

Git repozitáře, a v případě binárních souborů větších velikostí může dojít k omezení výkonu.

Instalace a používání Git repozitáře na straně serveru vyžaduje přístup k operačnímu systému serveru, takže není možné službu instalovat na webhostingové servery.

Projekt vůbec neumožňuje sdílení souborů s třetí stranou bez přístupu k Git repozitáři. Takové sdílení by bylo potřeba realizovat přes další službu, tedy stejně jako u verzovacích systémů.

2.2.5 Seafile

Toto řešení je velice blízké řešení popsanému v této práci. Obsahuje všechny funkce, které jsou vyžadované pro řešení problému.

Na straně serveru je služba založená na jazyce C, díky čemuž dosahuje služba dobré optimalizace výkonu. Díky tomu ji ale není možné instalovat na webhostingové servery a není příliš vhodná pro menší řešení.

V případě použití šifrování jsou data šifrována před odesláním na server, není tedy možné dešifrovat data na serveru, a to ani jeho správcem.

V případě sdílení s lidmi bez přístupu ke službě lze soubory sdílet pomocí odkazu nebo emailu. Případně lze soubory sdílet přímo s uživateli služby na stejném serveru.

Tato služba je také částečně zaměřená na týmovou kolaboraci, obsahuje funkce umožňující sdílet jednu složku s více uživateli či předem definovanými skupinami s nastavitelnými právy.

Součástí služby je i nativní klient pro operační systémy Microsoft Windows, GUN/Linux, MAC OS, Android a iOS. Nativní klient podporuje pouze synchronizaci určených složek.

2.2.6 Shrnutí

Jak lze vidět v tabulce 2.1, služba popsaná v této práci nabízí oproti ostatním řešením nativního klienta, jenž umožňuje správu souborů bez jejich stažení a synchronizace na lokálním úložišti.

Další výhodou je šifrování na straně klienta, které i v případě kompromitovaného serveru umožňuje maximální zabezpečení obsahu souborů. Navíc může v tomto případě klient používat libovolný šifrovací algoritmus bez jakékoli závislosti na serveru. Toto šifrování má jediný problém, a to nutnost soubor dešifrovat, pokud by jej chtěl dotyčný sdílet s třetími stranami.

Instalace na webhostingovém serveru je také jedním z důležitých požadavků, které splňují jen některé služby a služba popsaná v této práci. Ta se navíc orientuje i na nejjednodušší možnou instalaci, aby byla použitelná i méně zkušenými uživateli.

Pevně definované Rest API, které obsahuje všechny dostupné funkce služby, je také důležitý požadavek, který splňuje jen minimem služeb. Takto defino-

Tabulka 2.1: Porovnání existujících řešení

Řešení	Šifrování	Sdílení	Rest API	Klient web	Klient nativní[a]	Webhosting
FTP/SFTP	Ne	Ne	Ne	Ano	Ano	Ano[b]
Verzovací systémy	Ne	Ne	Ne	Ano	Ne	Ne
OwnCloud	Ne	Ano	Ano[c]	Ano	Ne	Ano
SparkleShare	Ano	Ne	Ne	Ne	Ne	Ano
SeaFile	Ano	Ano	Ano	Ano	Ne	Ne
Toto řešení	Ano	Ano	Ano	Ano	Ano	Ano

[a] Nativní klient, který umožňuje správu souborů bez jejich stažení a synchronizace
[b] Není možné služby na webhosting instalovat, ale každý webhosting aspoň jednu z nich podporuje pro nahrávání souborů webu
[c] WebDAV se dá považovat za implementaci Rest API

vané API umožňuje vznik jak nativního, tak i webového klienta, které budou obsahovat veškerou dostupnou funkcionalitu.

2.3 Šifrování

Pokud uživatel ukládá na vzdálené úložiště citlivá data, je v jeho zájmu, aby následně k datům neměl přístup nikdo jiný. Pomocí šifrování přenosu dat se dá zabránit odposlechu obsahu třetí stranou, ovšem uživatel už nemá jak zabezpečit, aby nikdo nemohl neoprávněně přistupovat k uloženým souborům na vzdáleném úložišti. K souborům může mít přístup správce serveru nebo potenciální útočník.

Tomu se dá zabránit šifrováním obsahu souboru. Klíč šifrování bude vlastnit pouze uživatel a data budou uložena v šifrované podobě, takže i když získá útočník přístup k samotným souborům, není schopen získat jejich skutečný obsah. Tímto je tedy zajištěno, že soubory nebudou kompromitovány bez ohledu na implementaci a stav vzdáleného úložiště.

2.3.1 Na straně serveru

Šifrování souborů lze zajistit na straně serveru. Ušetří se tím výkon klienta, ale toto řešení nese bezpečností riziko. Když obsah souboru šifruje server, má také k dispozici veškeré údaje potřebné k vytvoření šifrovacího a dešifrovacího klíče. Potenciální útočník by tudíž v případě, že má plný přístup na server, mohl získat obsah, i když je uložen v zašifrované podobě.

Útočník by mohl dále získat přístup k souborům úpravou serverové části, která by způsobila, že by byl soubor uložen nešifrovaně nehledě na požadavek klienta na šifrování souboru.

Šifrování na straně serveru je tedy použitelné pouze v případě, že uživatel chce soubor sdílet i s třetí stranou, což by vyžadovalo možnost soubor dešifrovat na straně serveru.

2.3.2 Na straně klienta

V případě šifrování souborů na straně klienta se vyhneme problémům s možným útokem na server. Server dostane obsah souboru již zašifrovaný a útočník tedy v případě použití dostatečně bezpečného šifrovacího algoritmu nemůže soubor dešifrovat.

Navíc si v tomto případě může klient vybrat, jakým algoritmem bude obsah šifrovat, takže může každý uživatel použít algoritmus, který mu bude nejvíce vyhovovat.

Toto řešení je tak oproti šifrování na straně serveru bezpečnější, a proto bude také použito ve službě popsané v této práci.

2.3.3 Algoritmus pro šifrování dat

Pro šifrování dat existuje několik algoritmů. Zde je jejich základní popis a vlastnosti. Pro šifrování dat se v tomto případě nejvíce hodí blokové šifry, tedy šifry, které šifrují vstupní data po blocích, kdy každý blok je šifrován stejnou transformací.

V dnešní době existuje velké množství šifrovacích algoritmů, ovšem v naprosté většině případů se novější algoritmy neliší lepším zabezpečením, spíše se zaměřují na výkon. Pro klienta se zvažovaly následující algoritmy:

AES Symetrická bloková šifra využívající klíče délky 128/192/256 bitů, využívaná vládou USA pro šifrování ve státních a federálních úřadech [10]. V současné době existují útoky, které jsou rychlejší než útoky hrubou silou, ovšem pro 256 bitový klíč potřebuje nejrychlejší možný útok $2^{254.4}$ operací. Tento algoritmus je v současnosti využíván i k šifrování souborů a i přes jeho velkou publicitu a popularitu nebyla dodnes nalezena efektivní kryptoanalýza [11].

Blowfish Symetrická bloková šifra umožňující používání klíče délky 32 až 448 bitů. Díky komplexní inicializaci klíče, jejíž náročnost přibližně odpovídá dešifrování 4 kiB dat, je vhodná i k ukládání menších objemů dat, například hesla [12]. Pro kratší délky klíče je tento algoritmus náchylný na útoky[13], proto je doporučeno používat delší klíče, tedy aspoň 128 bitů, pro které stejně jako v případě AES momentálně neexistuje efektivní kryptoanalýza.[14]

DES/TripleDES DES je předchůdce AES, takže se také jedná o symetrickou blokovou šifru. Kvůli malé velikosti klíče (pouhých 56 bitů) má malou odolnost proti útokům hrubou silou. TripleDES je vylepšení DES, v němž se DES použije třikrát v řadě, díky čemuž se dosáhne délky klíče až 168 bitů [15]. Oproti ostatním jmenovaným je tento algoritmus pomalý.

Vzhledem k výše uvedenému byl pro klienta zvolen AES jakožto základní algoritmus pro šifrování souborů. Pro šifrování menších dat, například uživatelského klíče, byl vybrán Blowfish.

2.4 Funkční požadavky

Vzhledem k počátečním cílům služby a existujícím řešením byly stanoveny následující požadavky:

F1 Nahrávání souborů Klient bude uživateli umožňovat nahrávat na vzdálený server soubory.

F2 Stahování souborů Klientská část služby bude uživateli umožňovat stahovat jím nahrané soubory ze vzdáleného serveru.

F3 Sdílení souborů Uživatel bude moci sdílet soubory s třetí stranou. Soubory půjde stáhnout pomocí webového klienta. Sdílet půjdou pouze speciálně označené soubory.

F4 Synchronizace Klient bude umožňovat synchronizaci lokálního úložiště se vzdáleným úložištěm.

F5 Šifrování souborů Klient umožní uživateli zašifrovat soubory před nahráním na server.

F6 Komunikace pomocí API Klient bude komunikovat se serverovou částí pomocí zdokumentovaného API.

F7 Historie souborů Služba bude ukládat starší verze přepsaných souborů a umožní uživateli jejich stažení.

F8 Instalace serveru Serverová část bude dodávána s instalačním skriptem, který bude umožňovat uživateli provést základní nastavení serveru.

2.5 Nefunkční požadavky

NF1 Podpora Windows a GNU/Linux Klient bude podporovat operační systémy Windows a GNU/Linux.

NF2 Jednoduchý a kompaktní server Serverová část služby bude jednoduchá na instalaci a vzhledem k tomu, že bude poskytovat pouze API, bude i maximálně kompaktní.

NF3 Podpora existujících serverů Server bude vytvořen tak, aby podporoval co největší množství již existujících serverových instalací.

NF4 Bezpečnost uložených souborů Klient a server bude používat takovou kombinaci šifrovacích algoritmů, která bude zajišťovat maximální bezpečnost souborů uložených na vzdáleném úložišti

Kapitola 3

Návrh

V návrhu služby se již zaměříme na návrh řešení požadavků určených v analýze.

3.1 Případy užití

Případy užití ilustrují funkcionalitu naší služby z pohledu uživatele [16]. V tomto případě definují, jak bude vypadat uživatelská interakce s klientskou aplikací naší služby. Také tím pokrývají všechny funkční požadavky, jak lze vidět v tabulce pokrytí 3.1.

UC1 Nahrání souborů Uživatel může na vzdálené úložiště nahrát soubory

1. Uživatel klikne na tlačítko pro nahrání souboru ve výpisu úložiště.
2. Aplikace umožní uživateli vybrat soubory z lokálního úložiště.
3. Uživatel vybere soubory, které chce nahrát.
4. Uživatel zahájí nahrávání kliknutím na tlačítko start.
5. Aplikace zobrazí průběh nahrávání souborů.
6. Aplikace informuje uživatele o výsledku stahování, v případě chyby zobrazí příslušnou chybovou hlášku.

UC2 Stahování souborů Uživatel může soubory uložené na vzdáleném úložišti stáhnout na lokální úložiště

1. Uživatel ve výpisu vzdáleného úložiště najde soubory, které chce stáhnout.
2. Uživatel použije tlačítko akce download.
3. Aplikace zobrazí uživateli seznam vybraných souborů.
4. Uživatel zahájí stahování kliknutím na tlačítko start.

5. Aplikace zobrazí průběh stahování souborů.
6. Aplikace informuje uživatele o výsledku stahování, v případě chyby zobrazí chybovou hlášku.

UC3 Sdílení souboru Uživatel může sdílet jednotlivé soubory s třetí stranou.

1. Uživatel ve výpisu vzdáleného úložiště najde soubor, který chce sdílet.
2. Aplikace zobrazí podrobnosti o souboru a možné akce.
3. V případě, že je soubor soukromý, uživatel vybere akci zveřejnění souboru.
4. Uživatel použije zobrazený odkaz pro sdílení souboru.

UC4 Synchronizace složky Uživatel může synchronizovat lokální složku se složkou na vzdáleném úložišti.

1. Uživatel si v hlavním oknu aplikace vybere seznam synchronizovaných složek.
2. Uživatel klikne na tlačítko přidat složku.
3. Aplikace zobrazí uživateli možnosti synchronizace, a to včetně výběru lokální a vzdálené složky.
4. Uživatel klikne na tlačítko uložit a uloží tím nastavení složky.
5. Uživatel vybere nově vytvořenou složku ze seznamu synchronizovaných složek.
6. Uživatel klikne na akci *synchronizovat vybrané složky*.
7. Aplikace zobrazí výpis vybraných složek a možné akce.
8. Uživatel klikne na tlačítko *zahájení synchronizace* a zahájí synchronizaci.
9. Aplikace bude průběžně informovat o stavu synchronizace, a to včetně stavu stahovaných a nahrávaných souborů.
10. Aplikace informuje uživatele o výsledku synchronizace, v případě chyby zobrazí chybovou hlášku.

UC5 Stažení starší verze souboru Uživatel bude mít možnost stáhnout si starší verzi souboru uloženého na vzdáleném úložišti.

1. Uživatel si v hlavním výpisu vybere soubor, který má starší verze.
2. Aplikace zobrazí uživateli informace souboru a počet verzí.
3. Uživatel vybere akci *zobrazení starších verzí*.
4. Aplikace zobrazí uživateli seznam starších verzí souborů.

Tabulka 3.1: Pokrytí funkčních požadavků případy užití

	F1	F2	F3	F4	F5	F6	F7	F8
UC1	X				X	X		
UC2		X			X	X		
UC3			X			X		
UC4				X	X	X		
UC5					X	X	X	
UC6								X

5. Uživatel vybere verzi a klikne na akci *stažení*.
6. Aplikace zobrazí průběh stahování souboru.
7. Aplikace informuje uživatele o výsledku stahování, v případě chyby zobrazí chybovou hlášku.

UC6 Instalace serverové části Uživatel bude mít možnost nainstalovat na server serverovou část služby.

1. Uživatel otevře instalační aplikaci.
2. Uživatel vyplní potřebné údaje.
3. Uživatel vybere akci *validovat*.
4. Aplikace zkontroluje vyplněné údaje a případně zobrazí chyby.
5. Uživatel vybere akci *instalovat*.
6. Aplikace zobrazí průběh instalace.
7. Aplikace zobrazí výsledek instalace a v případě, že došlo k chybě, zobrazí chybovou hlášku.

3.2 Šifrování

Na správném návrhu šifrování závisí celá bezpečnost služby. Na základě analýzy bylo stanoveno, že šifrování bude probíhat na straně klienta a pomocí klíče, ke kterému má přístup pouze uživatel.

3.2.1 Klíč

K čisté formě klíče může mít přístup pouze uživatel, který ho vlastní. Navíc musí být k dispozici paralelně z více zařízení. Vzhledem k těmto požadavkům byla stanovena následující možná řešení:

Uživatelské heslo Nejpřímějším řešením je použití uživatelského hesla. Uživatel toto heslo musí zadat při přihlášení, a proto se na něj nekladou žádné další požadavky. Navíc bude takový klíč po přihlášení k dispozici na jakémkoliv zařízení. Hlavní nevýhodou je závislost bezpečnosti šifrovacího algoritmu na entropii[17] uživatelského hesla. Jednoduchá hesla o malé délce by v tomto případě znamenala jednoduchou prolomitelnost šifry. Navíc by bylo potřeba při změně uživatelského hesla přešifrovat celé úložiště.

Derivace uživatelského hesla Tímto řešením se vyhneme hlavní nevýhodě použití uživatelského hesla. Pomocí derivační funkce, například hashovací funkce, dosáhneme u libovolného hesla i se slabou entropií silného klíče s velkou délkou. Při použití správně derivační funkce navíc prodloužíme dobu generování klíče, čímž se zpomalí útok hrubou silou. Ovšem požadavek přešifrování celého úložiště v případě změny hesla zůstává.

Vygenerovaný klíč Vygenerovaný klíč zajišťuje dostatečnou bezpečnost šifrovacího algoritmu a zároveň nebude potřeba při změně hesla přešifrovat celé úložiště. Nevýhodou je, že by takový klíč musel být uložen u klienta a klient by musel klíč při přihlášení nějakým způsobem poskytnout. Navíc by v případě použití dalších zařízení musel klíč přenášet.

Vygenerovaný klíč zašifrovaný derivací uživatelského hesla Jedná se o spojení všech předchozích řešení. Klíč se vygeneruje u klienta a následně zašifruje pomocí derivace uživatelského hesla. Zašifrovaný klíč se pak uloží na server, což bezpečnost zašifrovaných dat přímo neohrozí. Díky tomu bude poté klíč k dispozici i na více zařízení a po uživateli se stále bude vyžadovat pouze jeho heslo. V případě změny hesla se pak přešifruje pouze klíč a úložiště zůstane nezměněné.

Jako řešení bylo vybráno použití vygenerovaného klíče zašifrovaného pomocí derivace hesla. Kromě velké entropie a možnosti paralelního využití klíče toto řešení umožňuje klientovi vybrat si délku klíče a algoritmus použitý při jeho generování.

3.2.2 Sdílení

Z bezpečnostních důvodů není možné sdílet soubory zašifrované uživatelským klíčem s třetí stranou. V takovém případě by buď třetí strana, nebo server musel znát klíč k šifrování. Pro sdílení souborů je proto potřeba jejich obsah přešifrovat pomocí klíče uloženého na serveru nebo nechat obsah nezašifrovaný.

Vzhledem k povaze sdílených souborů a náročnosti dešifrování obsahu na serveru bylo rozhodnuto, že sdílené soubory budou ukládány v nezašifrované formě.

3.3 Synchronizace

Synchronizace bude řešena mezi vzdálenou složkou a lokální složkou. Aplikace bude muset při každé synchronizaci porovnat celou lokální a vzdálenou strukturu a vyhodnotit, zda je soubor na lokálním úložišti nový, aktualizovaný, nezměněný, starý či neexistující. To, jestli je soubor nový či neexistující, aplikace zjistí jednoduše porovnáním existence ve vzdáleném úložišti.

Pro zjištění dalších stavů souboru už musí aplikace porovnat obsah souborů a vyhodnotit, zda se soubor změnil na lokálním či vzdáleném úložišti, a pomocí toho zjistit, která verze je novější. Řešení porovnávání souborů pomocí jejich obsahu není příliš efektivní. V případě velkých souborů by bylo příliš pomalé, navíc by se vzhledem k tomu, že na serveru jsou soubory uloženy šifrovaně, musel každý soubor buď dešifrovat, nebo šifrovat.

Abychom se takovému porovnávání vyhnuli, použijeme kontrolní součet. Pomocí hashovací funkce převedeme obsah souboru na blok dat o předem dané velikosti, který poté budeme moci porovnávat. Hashovací funkce je taková funkce, která umožňuje převést libovolně velký blok dat na blok dat o pevně dané velikosti. Kromě toho zajišťuje, že stejný blok dat vždy převede na stejný výsledný blok dat. Proto je tato funkce v našem případě vhodná pro porovnávání obsahu souborů.

Vzhledem k tomuto řešení porovnávání vzniká požadavek, aby bylo v případě nahrávání souboru potřeba zároveň odeslat i kontrolní součet jeho nezašifrovaného obsahu, aby aplikace při výpočtu kontrolního součtu lokálního souboru nemusela obsah šifrovat.

Ve chvíli, kdy aplikace zjistí, že se soubory liší, musí zjistit, na jakém úložišti se nachází aktualizovaná verze. To udělá pomocí času poslední úpravy. Zde je navíc potřeba před porovnáním časů synchronizovat lokální čas se vzdáleným, aby nedošlo k chybným výsledkům v případě špatně nastaveného času.

Poté, co se rozhodne, jestli je soubor aktualizovaný či starý, soubor nahraje nebo stáhne a synchronizace je dokončena.

3.4 Starší verze souborů

Služba bude umožňovat ukládání starších verzí souborů. Starší verze souboru vznikne při synchronizaci v případě, že se doje k aktualizaci obsahu souboru. Pro ukládání nových verzí by bylo možné ukládat pouze změněnou část souboru, nebo jeho kompletní obsah. V případě ukládání změněné části souboru by bylo potřeba uchovávat všechny předchozí verze souboru a bylo by potřeba implementovat funkci porovnání obsahu. Aby bylo možné v budoucnu omezit počet uložených verzí, bude služba ukládat kompletní obsah všech verzí.

3.5 Síťové API

Veškerá komunikace se službou musí probíhat pomocí síťového API. Je tedy potřeba navrhnout dostatečně sofistikované a univerzální řešení, které by umožňovalo veškerou vyžadovanou funkcionalitu. Nejdříve přesně určíme, jaké operace bude muset API umožňovat, a poté se na základě těchto operací vybere vyhovující protokol.

3.5.1 Funkce

Na základě funkčních požadavků bylo stanoveno, že síťové API bude umožňovat následující funkcionalitu:

Zjištění informací o uživateli a serveru Klient bude potřebovat zjistit parametry serveru a informace o uživateli.

Nahrávání souborů Zpracovat nahrávané soubory a správně je umístit do úložiště.

Stahování souborů Na základě poskytnutých parametrů poskytnout uživateli obsah souboru.

Úprava souborů Uživatel bude mít možnost soubory přejmenovávat, přesouvat, mazat a přešifrovávat.

Procházení úložiště Procházet všechny uživatelské složky a soubory

Správa uživatelů Správce bude mít přístup ke správě uživatelů, tedy jejich k přidávání, úpravě a mazání.

3.5.2 Protokol

Protokol bude určovat, jak bude vypadat komunikace mezi serverem a klientem. Může přímo definovat, jak bude vypadat celá komunikace včetně podporovaných funkcí, jako je tomu u WebDAV, nebo pouze obecně definovat, jak se budou přenášet data komunikace. Bylo zvažováno použití následujících možností:

TCP Možné řešení je navržení vlastního protokolu na bázi TCP. Výhodou tohoto řešení by byla maximální efektivita protokolu. Nevýhodou je složitost návrhu i implementace, která vyžaduje práci se síťovou komunikací. TCP protokol také při použití v prohlížeči nelze využít v jazyku Javascript a tím by se znemožnilo možné budoucí rozšíření o webového klienta.

WebDAV Řešení používané například službou OwnCloud. Jedná se o nadstavbu HTTP protokolu, která k doručení obsahu využívá formát XML. Všechny funkce jsou předem pevně definované pomocí standardu a není tedy prostor pro vlastní rozšíření. Tento protokol by bez výraznějších úprav neumožňoval správu uživatelů.

SOAP Jedná se o protokol přímo navržený pro tvorbu obecných síťových API. [18] Předávání informací probíhá ve formátu XML za využití protokolu HTTP. Oproti Rest API lze přímo pomocí protokolu definovat, jak má vypadat komunikace a jaké funkce server podporuje. Nevýhodou tohoto řešení je tímto vzniklá složitost implementace. Server musí předem klientovi odeslat všechny podporované funkce a jejich parametry v protokolu v definovaném formátu a klient pak musí přesně podle přijaté definice odesílat požadavky.

Rest s použitím JSON Obecná definice API využívajícího pouze HTTP protokol. V našem případě by se jednalo o využití HTTP pro předávání požadavků a odpověď by server poskytoval uloženou jako JSON [19]. Toto řešení nám umožňuje jednoduchou realizaci veškeré funkcionality a vzhledem k tomu, že HTTP protokol a jeho parametry mají v případě běžně používaných serverových aplikací, by toto řešení umožňovalo zjednodušit samotnou implementaci API na straně serveru.

Použití protokolu WebDAV by v tomto případě nebylo ideální, hlavně kvůli jeho neuniverzálnosti. Vlastní TCP protokol by byl zase nepraktický. Bylo by potřeba spouštět vlastní server na vyhrazeném portu, což by značně omezilo počet kompatibilních serverových řešení.

Jako nejvíce praktická řešení se jeví SOAP a Rest. V našem případě bylo rozhodnuto použít Rest, implementace SOAP by byla složitější a nepřinesla by žádné důležité výhody.

3.6 Uživatelské rozhraní

Uživatelské rozhraní se týká pouze klienta, interakce se serverem bude probíhat výhradně přes síťové API. Vzhledem k cílové platformě klientské aplikace, konkrétně MS Windows a GNU/Linux, bylo rozhodnuto, že rozhraní bude rozděleno na několik různých dialogů, kdy každý dialog bude vyhrazen pro určitou funkci.

Následuje popis jednotlivých dialogů, jejich funkcionalit a potřebných komponent.

3.6.1 Přihlášení

Tento dialog umožní uživateli vyplnit adresu serveru a přihlašovací údaje. Měl by navíc informovat uživatele, pokud server odmítl autentizaci nebo se server

na zadané adrese nenachází.

Dále bude s výjimkou hesla vyplněné údaje ukládat, aby byly při příštím spuštění předvyplněné.

3.6.2 Vytvoření klíče

V případě, že se uživatel přihlásí na účet, který ještě nemá nastavený šifrovací klíč, ho aplikace vyzve k nastavení nového hesla, a to vzhledem k tomu, že klíč musí být zašifrovaný heslem.

Kromě možnosti zadání hesla zobrazí uživateli i parametry generovaného klíče. Po vytvoření klíče bude uživatel přesměrován na hlavní okno stejně jako po úspěšném přihlášení.

3.6.3 Hlavní okno

Hlavní okno bude zobrazovat výpis vzdáleného úložiště, výpis synchronizovaných složek, umožní uživateli zobrazit nastavení a v poslední řadě umožní nahrání souborů na úložiště.

Výpis vzdáleného úložiště bude zobrazovat soubory a složky na vzdáleném úložišti. U každého souboru pak bude zobrazovat jeho velikost a datum poslední úpravy. Konečně bude umožňovat uživateli soubory smazat, přesunout nebo stáhnout. V případě výběru individuálního souboru by pak měl zobrazit detail souboru.

Výpis synchronizovaných složek bude zobrazovat seznam lokálních a synchronizovaných složek se vzdálenými a datum poslední synchronizace každé složky. Dále bude umožňovat spuštění synchronizace jedné nebo více složek. V případě výběru individuální složky by pak měl zobrazit detail synchronizované složky.

3.6.4 Detail souboru

Detail souboru bude zobrazovat všechny dostupné informace o souboru, jeho název, velikost, datum nahrání, datum poslední synchronizace, starší uložené verze, informace o šifrování, zda je soubor veřejný a případně odkaz pro sdílení souboru s třetí stranou.

Dále bude umožňovat akce zveřejnění, přejmenování, smazání, přesunutí a stažení.

3.6.5 Detail synchronizované složky

Detail synchronizované složky bude zobrazovat základní nastavení složky, tedy cestu k lokální složce, cestu ke vzdálené složce a omezení synchronizovaných souborů.

Dále bude umožňovat spuštění synchronizace dané složky.

3.6.6 Stahování

Kdykoliv uživatel vybere akci stahování, měl by se mu zobrazit tento dialog. Bude obsahovat výpis stahovaných souborů včetně jejich velikosti, možnost určení složky, do které se mají soubory stáhnout, a akce zrušení a započetí stahování.

V průběhu stahování bude uživateli zobrazovat stav jednotlivých souborů a celkový stav stahování.

Po stažení všech souborů informuje uživatele o výsledku a skryje se.

3.6.7 Nahrávání

Tento dialog umožní uživateli nahrát jeden nebo více souborů. Bude obsahovat možnost výběru cílové složky, přidání souborů, výpis vybraných souborů včetně velikosti a akce zrušení a započetí nahrávání.

V průběhu nahrávání bude zobrazovat stav jednotlivých souborů i celkového nahrávání.

Po dokončení nahrávání informuje uživatele o výsledku a skryje se.

3.6.8 Synchronizace

Dialog synchronizace se zobrazí v případě synchronizace složky. Bude obsahovat stav synchronizace, výpis stahovaných souborů, výpis nahrávaných souborů a možnost synchronizaci zastavit.

Na konci synchronizace informuje uživatele o výsledku a skryje se.

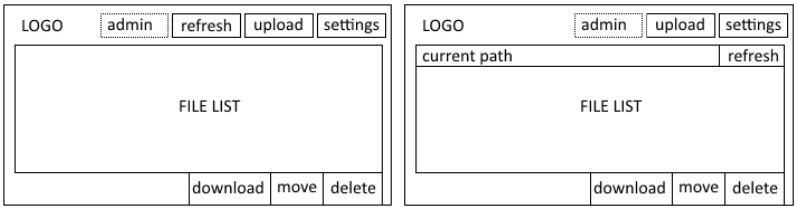
3.6.9 Nastavení

Nastavení aplikace umožní uživateli změnit heslo a používaný druh šifrování. Pro synchronizaci by mělo zobrazovat potřebné údaje pro spuštění přes příkazovou řádku.

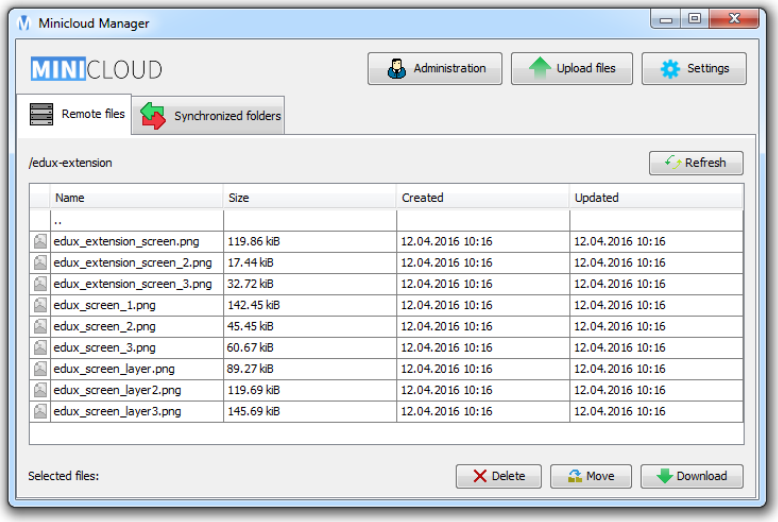
3.6.10 Rozložení prvků a zpětná vazba

Ve chvíli, kdy bylo určeno, jaké komponenty jsou u každého dialogu potřebné, byl v grafickém programu vytvořen návrh rozložení těchto komponent v každém dialogu. Tento konečný návrh rozhraní byl následně použit při implementaci.

Grafický návrh prošel několika verzemi, jak je vidět na obrázku 3.1. Návrh byl vždy představen předem vybraným potenciálním uživatelům a pomocí zpětné vazby upravován až do konečné podoby. Poslední úpravy budou probíhat až po implementaci a testování, kdy se očekávají možné problémy související se zvolenou implementační metodou.



(a) První verze. (b) Druhá verze.



(c) Implementace pod OS Windows 7.

Obrázek 3.1: Návrh rozhraní hlavního okna.

Kapitola 4

Implementace

Implementace byla rozdělená do tří částí. Nejdříve bylo potřeba implementovat server, proti kterému byl poté implementován klient. Poslední část implementace pak byla instalace serveru.

4.1 Server

Serverová část bude spravovat vzdálené úložiště. Prakticky to znamená, že bude zpracovávat požadavky od klientů, které budou přicházet přes API. Serverová část nebude mít žádný frontend a bude se ovládat pouze přes API.

4.1.1 Implementační jazyk

Při výběru implementačního jazyka jsme se řídili hlavně požadavky na širokou podporu existujících serverových instalací a faktem, že server bude pouze zajišťovat komunikaci přes síťové API.

4.1.1.1 Python

Jedná se o skriptovací jazyk, který v určitých případech lze využít pro implementaci síťového serveru. Lze v něm využít TCP komunikaci a obsahuje i knihovnu přímo určenou pro komunikaci přes HTTP. Výkon pro tento druh aplikace by byl naprosto dostačující. Jediným problémem při využití tohoto jazyku by byla malá podpora mezi webhostingovými servery.

4.1.1.2 PHP

Další skriptovací jazyk, primárně určený pro tvorbu webových aplikací. Implementovat v něm HTTP API tedy bude možné. V případě webové aplikace je PHP pouze nadstavbou existujícího webového serveru, nestará se tedy přímo o HTTP provoz, ale pouze o jeho výsledek. To značně usnadňuje vývoj.

Tento jazyk je podporován naprostou většinou webhostingových serverů a je možné ho nainstalovat na většinu existujících serverů.

4.1.1.3 Java

Objektově orientovaný jazyk určený pro tvorbu obecných aplikací. V současné době jsou dostupné frameworky, které by ulehčily tvorbu HTTP API. Vzhledem k jeho popularitě je dostupné velké množství nástrojů, které práci v tomto jazyce značně ulehčují.

V porovnání se skriptovacími jazyky je hlavní nevýhodou tohoto jazyka náročnost na zdroje, hlavně na operační paměť. Implementovat webovou aplikaci v tomto jazyce se vyplatí, pokud se u aplikace očekávají tisíce požadavků za sekundu. Pak by byla tato implementace výhodnější než použití skriptovacích jazyků. V projektu popsaném v této práci se ale vzhledem k tomu, že je určen pro soukromé použití, počítá s menším vytížením.

Další nevýhodou je menší podpora serverů. Instalace webové aplikace implementované v tomto jazyce vyžaduje plný přístup k serveru, čímž se vylučuje podpora běžných webhostingových serverů.

4.1.1.4 Použitý jazyk

Zejména díky velké rozšířenosti a menší náročnosti v případě menšího použití byl pro implementaci serverové části vybrán jazyk PHP. Tento jazyk nám umožnil rychlý a flexibilní vývoj serverové části a velkou podporu existujících serverů a webových hostingů.

4.1.2 Třídy

Aby bylo možné serverovou část v budoucnu rozšířit, byly některé komponenty nejdříve implementovány čistě abstraktně, byly tudíž nadefinovány pouze názvy používaných metod a v dokumentaci bylo popsáno, jak by se měla implementace chovat. Abstraktní třídy tedy byly:

MetaStorage Abstraktní třída, která reprezentuje databázi meta dat. Meta data obsahují informace o uživatelích, složkách a souborech.

ContentStorage Abstraktní třída reprezentující úložiště dat souborů. Tato třída na základě poskytnutých meta dat, které jsou reprezentované třídou `MetaFile`, poskytne implementaci abstraktní třídy `ContentStorageFile`, pomocí které se pak pracuje s daty souboru. Kromě toho obstarává mazání dat odstraňovaných souborů.

ContentStorageFile Abstraktní třída, která reprezentuje data jednoho určitého souboru v úložišti. Umožňuje proudový přístup k datům souborů.

Serverová část dále obsahuje plně implementované třídy:

Api Třída zpracovává požadavky na základě rest specifikace a následně je předává třídě `ApiHandler`, která pak implementuje logiku provedení požadavků a přípravu odpovědi.

ApiHandler Implementuje logiku funkčnosti, přijme zpracovaný požadavek a pomocí tohoto požadavku provede vyžadovanou funkci. Na závěr vrátí odpověď obsahující požadovaná data.

ApiResponse Rozšiřuje základní odpověď `Response` o specifikaci Rest protokolu služby popsané v této práci. Kromě dat poskytuje datový typ odpovědi pro usnadnění zpracování na straně klienta.

MetaFile Třída reprezentující meta informace o souboru. Tato třída se používá při komunikaci jak s `MetaStorage`, tak s `ContentStorage`.

MetaPath Třída, která reprezentuje meta informace o složce. Tato třída se používá při komunikaci jak s `MetaStorage`, tak s `ContentStorage`.

MetaUser Reprezentuje meta informace o uživateli. Tato třída se používá při komunikaci jak s `MetaStorage`, tak s `ContentStorage`.

Request Zpracovává hodnoty požadavku, které jsou následně použity v třídách Api a ApiHandler, včetně informací potřebných pro zpracování přijatých souborů.

Response Základní třída reprezentující odpověď serveru včetně serializace dat odpovědi.

DBOMetaStorage Základní implementace `MetaStorage`, která k ukládání meta dat využívá obecnou databázi s podporou jazyka SQL. Schéma databáze můžete vidět na obrázku 4.2. DBO je PHP knihovna umožňující obecné připojení k SQL databázi.

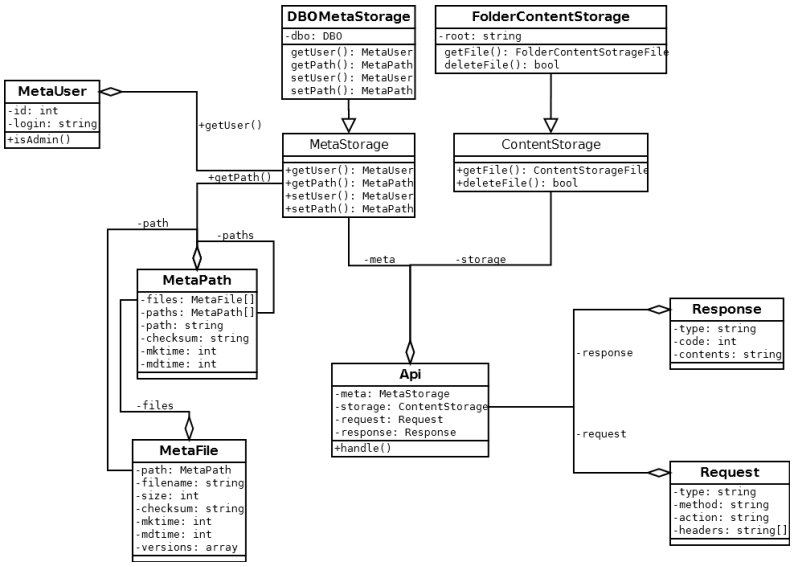
FolderContentStorage Základní implementace `ContentStorage`, jež ukládá obsah souborů do specifikované složky.

FolderContentStorageFile Implementace `ContentStorageFile` poskytovaná třídou `FolderContentStorage`. Umožňuje proudový přístup k souborům uloženým ve složce.

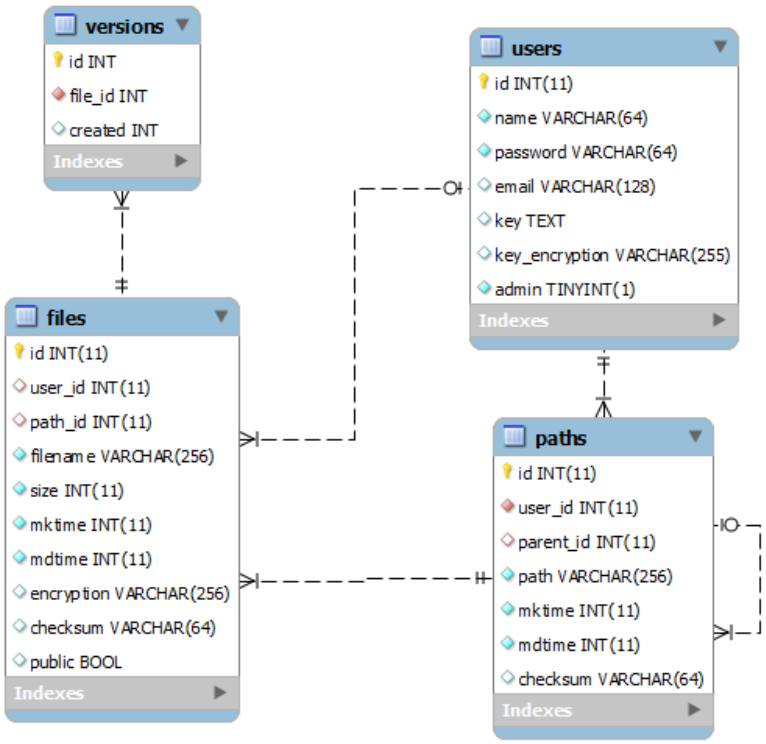
Komunikace jednotlivých tříd je znázorněna na diagramu 4.1

4.1.3 Síťové API

Vzhledem k nadstavbě webového serveru nemusí serverová aplikace popsaná v této práci pracovat na síťové vrstvě, tudíž zpracovává pouze požadavky a vrací náležitou odpověď.



Obrázek 4.1: Diagram tříd serveru.



Obrázek 4.2: Schéma databáze meta dat.

V případě použití PHP se o síťovou komunikaci stará webový server, který serverové části pouze předává požadavky. Implementace API se tedy skládala jen ze zpracování požadavku a předání dat komponentě, jež se stará o vykonávání požadavků.

Výsledek operace následně opět stačilo předat webovému serveru a ten již zajistil předání klientovi.

Jak bylo určeno v návrhu, pro komunikaci se používá protokol Rest/JSON. PHP od verze 5.2.0 plně podporuje serializaci a deserializaci dat ve formátu PHP, proto v implementaci stačilo pouze využít oficiální implementaci.

4.1.4 Testování

Serverová část byla testována průběžně v průběhu implementace. K jejímu testování byly použity dvě metody testování.

Hlavní metodou testování v prvních fázích implementace bylo testování programátorem, tedy manuální testování právě implementovaných funkcí. V pozdějších fázích nebyla tato metoda dostatečně účinná, proto bylo potřeba připravit automatické testy.

Ze skupiny automatických testů byly implementovány unit a integrační testy.

4.1.4.1 Unit testy

Unit testy jsou automatické testy, které testují jednotlivé třídy nebo komponenty aplikace v izolovaném prostředí. Pokud je třída závislá na jiných komponentách, tyto testy jsou pouze simulované, ve skutečnosti tedy nekomunikují se skutečnými komponenty. Tím je zajištěno, že se testuje skutečně pouze daná komponenta a nezasahují do ní chyby ostatních komponent. Nemusí tedy odhalit problémy při komunikaci jednotlivých komponent. V tomto případě Unit testy kontrolovaly funkčnost tříd pro přijímání požadavků, práci s meta daty a práci s daty.

4.1.4.2 Integrační testy

Tyto testy kontrolují komunikaci mezi jednotlivými komponenty. Jsou tedy spouštěny po unit testech, kdy už víme, že jednotlivé třídy fungují tak, jak mají. V našem případě se pomocí integračních testů kontrolovala komunikace komponenty přijímající požadavky s meta databází a úložným prostorem.

4.2 Klient

Klientská část je rozdělena na dva oddělené projekty, knihovnu pro komunikaci se serverem a na aplikaci. Toto rozdělení umožňuje možnou budoucí implementaci klientů pro jiné platformy.

4.2.1 Implementační jazyk

4.2.1.1 C++

Tento jazyk nabízí naprosté většiny platforem, ovšem za cenu složitější kompilace, která se musí řešit pro každou platformu zvlášť.

Pomocí existujících nástrojů a knihoven by tvorba uživatelského rozhraní nebyla problém.

Hlavní nevýhodou tohoto jazyka je složitější vývoj, kdy se musí počítat s kompatibilitou s více platformami. V tomto jazyce navíc mohou vznikat chyby ve správě paměti, které je velice složité dohledat.

Výhodou řešení je vyšší teoretický výkon.

4.2.1.2 Java

Tento jazyk také nabízí podporu většiny platforem, ovšem vyžaduje instalaci implementace JVM.

Za pomoci existujících nástrojů a knihoven je tvorba uživatelského rozhraní jednoduchá.

Vývoj je mnohem jednodušší, mimo jiné díky tomu, že není potřeba ručně spravovat paměť.

4.2.1.3 Použitý jazyk

Vzhledem k lepší podpoře více platforem a mnohem jednoduššímu vývoji byla vybrána Java.

4.2.2 Knihovna

Knihovna zajišťuje veškerou komunikaci se serverem. Kromě komunikace pomocí síťového API také zajišťuje šifrování, stahování souborů, nahrávání souborů a synchronizaci složek. Jednotlivé funkce API jsou pro přehlednost rozdělené do příslušných balíčků knihovny.

Volání všech funkcí je vzhledem k povaze komunikace asynchronní, pro každé volání se tedy vytváří nové vlákno. Ke komunikaci vlákna s volající funkcí se používají události. Třída, které volá danou metodu, se nejdřív zaregistruje jako naslouchač. Poté přijímá události, pomocí kterých může zjistit průběh volání.

4.2.2.1 Balíčky

Knihovna je pro zpřehlednění kódu rozdělena do několika logických struktur. Toto rozdělení nemá žádný vliv na funkčnost a značně zpřehledňuje dokumentaci. Knihovna obsahuje následující balíčky:

api Obsahuje třídu zajišťující komunikaci se serverem a třídy reprezentující meta data.

api.events Události, které odesílá třída zajišťující komunikaci. Každá možná odpověď serveru má vlastní třídu, která předpřipraví přijatá data.

api.encryption Vnitřní logika knihovny zajišťující šifrování.

api.upload Obsahuje třídu obstarávající nahrávání souborů.

api.upload.events Události odesílané při nahrávání souborů.

api.download Obsahuje třídu obstarávající stahování souborů.

api.download.events Události odesílané při stahování souborů.

sync Obsahuje třídy zajišťující synchronizaci lokálních složek se vzdálenými.

sync.events Události odesílané při synchronizaci složek.

4.2.2.2 Synchronizace

Knihovna poskytuje implementaci synchronizace lokální složky se vzdálenou složkou pomocí třídy `SyncFolder`. Tato třída umožňuje nastavení cesty k lokální složce, cesty ke vzdálené složce, limit velikosti a regulární výraz pro filtrování souborů.

Třída si nejprve připraví fronty pro stahování a nahrávání změněných souborů. Poté vyžádá kompletní výpis stromu vzdálené složky. Jakmile ho dostane, začne ho porovnávat se stromem lokální složky a začne plnit frontu potřebnými operacemi. Na závěr synchronizace se postupně spouští stahovací a nahrávací fronta.

Operace synchronizace probíhá asynchronně a pro hlášení stavu synchronizace se stejně jako u třídy pro komunikaci používají události.

Jak bylo zmíněno v návrhu, pro porovnávání souborů se používá kontrolní součet. V tomto případě byl použit algoritmus MD5.

4.2.2.3 Šifrování

Z analýzy vyplývá, že pro šifrování se bude přednostně používat algoritmus AES. Knihovna bude implementovat pouze šifrování, generaci klíče tedy bude dodávat aplikace využívající knihovnu a poskytne pouze výsledný klíč.

Šifrování bylo realizováno pomocí Java Cryptographic Extension (JCE) frameworku. Jedná se o framework, jenž umožňuje obecnou práci se šiframi a kromě základních šifer podporuje i rozšíření o vlastní implementaci jakékoli šifry. V našem případě postačí základní implementace dodávaná s Javou, která garantuje podporu šifrování AES [20].

Tato implementace plně podporuje využití proudů, proto bylo možné implementovat šifrování a dešifrování tak, aby probíhalo přímo při stahování, potažmo nahrávání. Tím se ušetřil výkon a paměť.

Jak bylo určeno při návrhu architektury, aplikace získá klíč z meta informací uživatele v šifrované podobě a poté ho pomocí derivace uživatelského hesla dešifruje. Vzhledem k podpoře ve frameworku JCE byl jako derivační funkce vybrán algoritmus PBKDF2 [21]. Po načtení informací o uživateli ze serveru je tedy potřeba, aby aplikace, která využívá knihovnu, nastavila uživateli použité heslo. Knihovna následně pomocí tohoto hesla dešifruje klíč.

V průběhu vývoje se ukázalo, že na základě legislativních problémů je v JCE v případě algoritmu AES uměle omezena délka klíče na 128 bitů. Naše aplikace vzhledem k větší bezpečnosti vyžaduje podporu klíče dlouhého 256 bitů. V zemích, kde takto silná šifra není zakázána, lze toto omezení vypnout oficiálně podporovanou instalací definičního souboru, který ujistí implementaci šifrování, že uživatel používáním šifry neporušuje legislativu.

Omezení lze obejít také modifikací interních tříd JCE. Toto řešení je ovšem závislé na vnitřní struktuře JCE, která není nijak garantovaná, proto je možné, že se v příští verzi Javy změní. Proto není vhodné pro produkční prostředí.

Poslední možné řešení problému by bylo využití jiné implementace šifrování. Ovšem vzhledem k tomu, že v tuto chvíli je již JCE plně integrován do knihovny, bylo rozhodnuto použít oficiální řešení instalace bezpečnostního souboru. Tento soubor a instrukce na jeho instalaci jsou tedy součástí distribučního balíčku.

4.2.2.4 Budoucí využití

Knihovna byla navržena tak, aby ji bylo možné využít v dalších možných implementacích klienta. V rámci testování využitelnosti knihovny byla vytvořena takzvaná mockup aplikace pro operační systém Android.

Tato aplikace využívala zmíněnou knihovnu a kromě přihlášení a výpisu souborů umožňovala i stažení souborů, a to včetně těch šifrovaných.

Díky připravené knihovně její vývoj zabral pouhých 5 hodin programátorského času a většina tohoto času byla využita pro správnou implementaci uživatelské interakce.

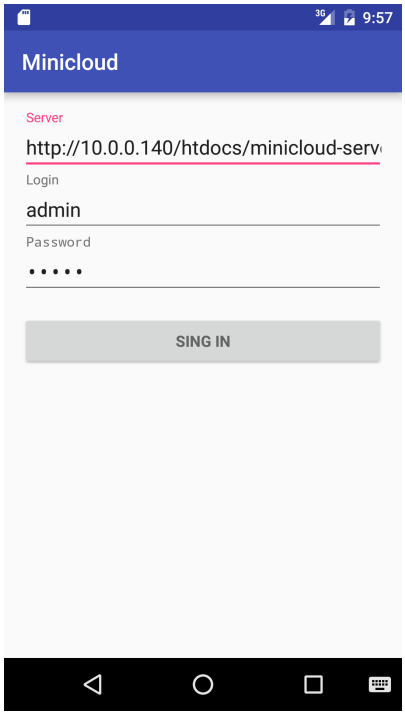
Ukázky z této aplikace můžete vidět na obrázku 4.3.

4.2.3 Aplikace

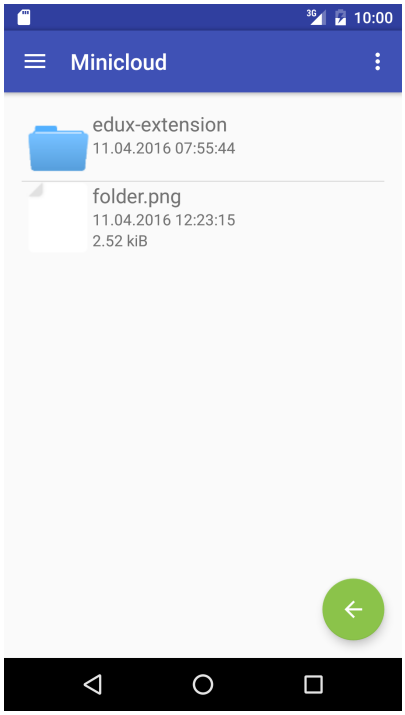
Vzhledem k tomu, že veškerou funkcionalitu již pokrývá knihovna, samotná aplikace je pouze implementaci uživatelského rozhraní.

4.2.3.1 Uživatelské rozhraní

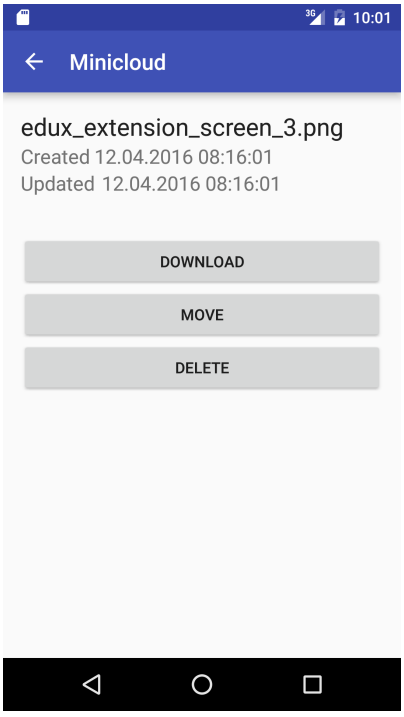
Uživatelské rozhraní bylo plně navrženo před započetím implementace. Při jeho implementaci již bylo potřeba se soustředit pouze na správnou funkčnost



(a) Přihlášení.



(b) Výpis souborů.



(c) Akce souboru.

Obrázek 4.3: Ukázka mockup aplikace na OS Android.

jednotlivých prvků.

vzhledem k podpoře používaného prostřední NetBeans byla pro implementaci vybrána knihovna Swing. Tato knihovna umožňuje implementovat uživatelské rozhraní nezávisle na platformě, na které bude aplikace běžet.

4.2.3.2 Textový režim

Textový režim aplikace umožňuje spustit nastavenou synchronizaci bez využití grafického prostředí a potřeby přihlašovat se na server heslem. Toto prostředí vzniklo hlavně proto, aby bylo možné naplánovat automatickou synchronizaci a zároveň se nemusel v uživatelském rozhraní implementovat běh na pozadí, který by byl vzhledem k podpoře více platforem složitý. Použití textového režimu umožňuje uživateli pro plánovanou synchronizaci použít nástroje poskytované operačním systémem.

Před použitím textového režimu se musí uživatel přihlásit do grafického rozhraní, v němž v sekci nastavení nalezne synchronizační klíč. Ten obsahuje veškeré potřebné informace pro synchronizaci, a to včetně uživatelského klíče. Jde sice o bezpečnostní riziko, ale jedná se menší riziko než případ, v němž by byl uživatel nucen zadat své heslo v čisté formě.

Textový režim v průběhu synchronizace informuje uživatele o průběhu pomocí předem definovaného textového formátu zpráv. Tento formát je speciálně navržen tak, aby byl jednoduše strojově zpracovatelný v případě, že by uživatel chtěl s logy synchronizace dále nakládat.

4.2.4 Testování

Na závěr vývoje byla, současně s aplikací otestována celá služba předem vybranou skupinou uživatelů. Testování probíhalo pomocí scénářů, vytvořených na základě případů užití. Pro distribuci a hlášení chyb byla využita platforma Github, pomocí které probíhala i komunikace s testery.

Na základě těchto testů pak byla vydána finální verze služby, která splňuje všechny předem stanovené cíle.

4.3 Instalace serveru

Instalace serveru bude usnadňovat nasazení a počáteční konfiguraci. Umožní uživateli nastavit správcovský účet, nastavit přístupové údaje k databázi meta dat a určit úložiště dat souborů. Kromě nastavení potřebných údajů instalace zkontroluje také jejich správnost.

Vzhledem k tomu, že serverová část je již implementovaná v jazyce PHP, i tento jazyk používá instalace ve spojení s HTML, CSS a Javascriptem.

Instalace nejprve vyzve uživatele k zadání potřebných údajů a poté zadané údaje zkontroluje. Tato funkcionalita je realizována pomocí AJAX požadavků

se zadanými údaji. Pomocí Javascriptu jsou následně uživateli zobrazeny výsledky validace a případně je umožněno spustit instalaci. Instalace se také spustí AJAX požadavkem, jehož výsledek se opět zobrazí uživateli, tentokrát jako konečný.

AJAX požadavky zpracovává PHP backend, jenž vykonává všechny potřebné funkce. Validace dat a instalace je implementovaná tak, aby co nejvíce reflektovala skutečnou funkčnost služby a tím se dosáhlo důvěryhodnosti výsledku.

Závěr

Cílem této práce bylo vytvoření kompletní a kompaktní služby vzdáleného úložiště. Nejdříve byla zanalyzována existující řešení a v čem se od nich tato služba liší. Poté se analýza zaměřila na zabezpečení obsahu úložiště. Na závěr se pak analýza věnovala vymezením požadavků na službu, určených v předchozích krocích.

V části návrhu se tato práce zabývala jak dosáhnout vytyčených úkolů. Nejdříve bylo určeno jak dojít k dostatečnému zabezpečení při použití algoritmu určeném v analýze. Dále bylo potřeba navrhnout, jak bude fungovat synchronizace, mimo jiné aby následný návrh komunikace se synchronizací počítal. Návrh pokračoval vymezením funkcionality síťového API a použití správného protokolu, který nejvíce vyhovuje této službě. Na závěr bylo navrženo uživatelské rozhraní klientské aplikace, čím se dosáhlo uživatelsky příjemného uživatelského rozhraní.

V praktické části se pak zabývá práce implementací navržených komponent. Jako první byla implementována serverová část, to proto, aby proti ní šlo implementovat část klientskou. U každé části byl nejdříve vybrán vhodný implementační jazyk. Při implementaci byla každá část průběžně testována programátorem, serverová část byla podrobena automatickými testy a výsledná část byla podrobena testováním předem vybrané skupiny uživatelů, na jehož základě byly provedeny finální opravy.

Výsledek práce je funkční služba vzdáleného úložiště, která umožňuje všechny předem určené funkce. Kromě služby obsahuje výsledek i dokumentaci síťového API a Java knihovnu použitelnou při implementaci libovolné klientské aplikace.

Budoucí rozvoj

Výsledná služba by šla rozšířit o klienta určeného pro další platformy, například Android. Klientská Java knihovna, jak bylo popsáno v části implementace, je přizpůsobena používání i na tomto operačním systému, což bylo

ověřeno vytvořením funkčního prototypu. Dalším možným vylepšením by byla vylepšená správa uživatelů, například možnost přidat omezení využitého prostoru pro uživatele, nebo možnost sdílet úložiště mezi dalšími uživateli. Z pohledu klientské aplikace je možné rozšířit podporu šifrovacích algoritmů.

Literatura

[1] Quirk, K.: 3 Self-Hosted Dropbox Alternatives, Tested. http://www.makeuseof.com/tag/3-self-hosted-dropbox-alternatives-tested/, cit. 2015-11-01.

[2] Brinkmann, M.: Four self-hosted Dropbox-like services businesses can use. http://betanews.com/2012/06/30/four-self-hosted-dropbox-like-services-businesses-can-use/, cit. 2015-11-01.

[3] Regan, P.: Self-Hosted Cloud-Storage Comparison - 2014 Edition. http://blog.patshead.com/2014/09/self-hosted-cloud-storage-comparison-2014-edition.html, cit. 2015-11-01.

[4] The Internet Society: *SSH File Transfer Protocol*. Cit. 2015-11-01. Dostupné z: https://tools.ietf.org/html/draft-ietf-secsh-filexfer-13

[5] Spinellis, D.: Git. *IEEE Software*, ročník vol. 29, č. issue 3, 2012: s. 100–101, ISSN 0740-7459, doi:10.1109/MS.2012.61. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6188603

[6] Free Software Foundation, Inc.: *GNU AFFERO GENERAL PUBLIC LICENSE*. Cit. 2015-11-01. Dostupné z: http://www.gnu.org/licenses/agpl-3.0.html

[7] OwnCloud: How ownCloud uses encryption to protect your data. https://owncloud.org/blog/how-owncloud-uses-encryption-to-protect-your-data/, cit. 2015-11-01.

[8] L. Dusseault, E.: HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV). RFC 4918, Network Working Group, July 2007. Dostupné z: http://www.ietf.org/rfc/rfc4918.txt

[9] ownCloud developers, T.: OwnCloud 8.0 Server System Requirements. https://doc.owncloud.org/server/8.0/admin_manual/installation/system_requirements.html, cit. 2016-24-04.

[10] Burr, W.: Selecting the Advanced Encryption Standard. *IEEE Security & Privacy Magazine*, ročník 99, č. 2, 2003: s. 43–52.

[11] Biclique Cryptanalysis of the Full AES. In *Advances in Cryptology – ASIACRYPT 2011: 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, ISBN 978-3-642-25385-0, s. 344–371.

[12] Priyadarshini, P.; Prashant, N.; Narayan, D.; aj.: A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish. *Procedia Computer Science*, ročník 78: s. 617–624.

[13] Rijmen, V.: *Cryptanalysis and Design of Iterated Block Ciphers*. Dizertační práce, 1997.

[14] Alabaichi, A.; Ahmad, F.; Mahmod, R.: Security analysis of blowfish algorithm. IEEE, 2013, s. 12–18.

[15] Barker, W. C.; Barker, E.: *Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher: NIST Special Publication 800-67*. Gaithersburg, Maryland: CreateSpace Independent Publishing Platform, revision 2 vydání, 2012, ISBN 1478178175.

[16] Gomaa, H.: *Software modeling and design*. New York: Cambridge University Press, 2011, ISBN 05-217-6414-9.

[17] Password Entropy and Password Quality. In *Network and System Security (NSS), 2010 4th International Conference on*, Piscataway: I E E E, první vydání, 2010, ISBN 9781424484843, s. 583–587.

[18] Moreau, J.-J.; Hadley, M.; Nielsen, H. F.; aj.: SOAP Version 1.2 Part 1: Messaging Framework. W3C recommendation, W3C, Červen 2003, http://www.w3.org/TR/2003/REC-soap12-part1-20030624/.

[19] Bray, T.: The JavaScript Object Notation (JSON) Data Interchange Format. RFC 7159, RFC Editor, March 2014, http://www.rfc-editor.org/rfc/rfc7159.txt. Dostupné z: http://www.rfc-editor.org/rfc/rfc7159.txt

[20] Oracle: *Java Cryptography Architecture (JCA) Reference Guide*. Cit. 2016-04-17. Dostupné z: http://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html

[21] Kaliski, B.: PKCS #5: Password-Based Cryptography Specification Version 2.0. RFC 2898, RFC Editor, September 2000, http://www.rfc-editor.org/rfc/rfc2898.txt. Dostupné z: http://www.rfc-editor.org/rfc/rfc2898.txt

Příloha **A**

Seznam použitých zkratek

AES Advanced Encryption Standard

AJAX Asynchronous JavaScript and XML

API Application Programming Interface

DES Date Encryption Standard

FTP File Transfer Protocol

HTTP Hypertext Transfer Protocol

JSON JavaScript Object Notation

PHP PHP Hypertext Preprocessor

PBKDF2 Password-Based Key Derivation Function 2

SFTP SSH File Transfer Protocol

SOAP Simple Object Access Protocol

UI User Interface

WebDAV Web-based Distributed Authoring and Versioning

XML Extensible Markup Language

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `dist` ..... adresář s distribučními balíčky
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS

Příloha C

Uživatelská příručka

C.1 Úvod

Tato příručka vás seznámí s použitím klientské aplikace služby.

C.2 Požadavky

Aplikace vyžaduje JRE 6 a vyšší, nebo OpenJDK. Podporované operační systémy jsou Microsoft Windows XP, nebo novější a GNU/Linux.

C.3 Instalace

Aplikace je dodávána pomocí distribučního balíku ve formátu ZIP. Aplikace se nainstaluje rozbalením jeho obsahu do libovolné složky. Balík obsahuje spustitelné soubory pro všechny podporované operační systémy.

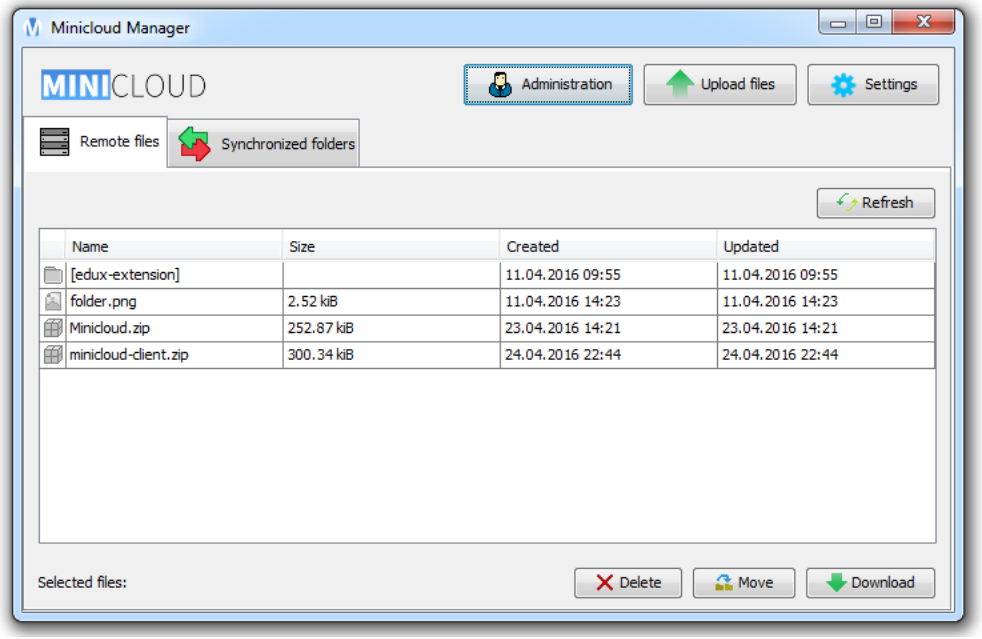
minicloud.bat Spustitelný soubor pro Microsoft Windows. Lze použít pro spuštění textového režimu aplikace.

minicloud.exe Spustitelný soubor pro Microsoft Windows. Nelze použít pro spuštění textového režimu aplikace.

minicloud.sh Spustitelný soubor pro GNU/Linux. Lze použít pro spuštění textového režimu aplikace.

C.4 První spuštění

Při prvním spuštění aplikace na operačním systému Microsoft Windows je možné, že vás aplikace vyzve k instalaci bezpečnostního souboru, který umožní



Obrázek C.1: Hlavní okno aplikace.

používání silnějšího šifrování. Tuto instalaci můžete provést ručně podle instrukcí přiložených v instalačním balíčku nebo pomocí spustitelného souboru `install_key.bat`, který instalaci automatizuje.

Dále, po přihlášení, pokud se jedná o nově vytvořený účet, budete vyzváni k zadání nového hesla. Jedná se důležité bezpečnostní opatření a je nutné zadat nové bezpečné heslo, které se nijak nepodobá původnímu. Po potvrzení nového hesla aplikace vytvoří nový šifrovací klíč, který se bude do budoucna používat pro šifrování souborů.

C.5 Aplikace

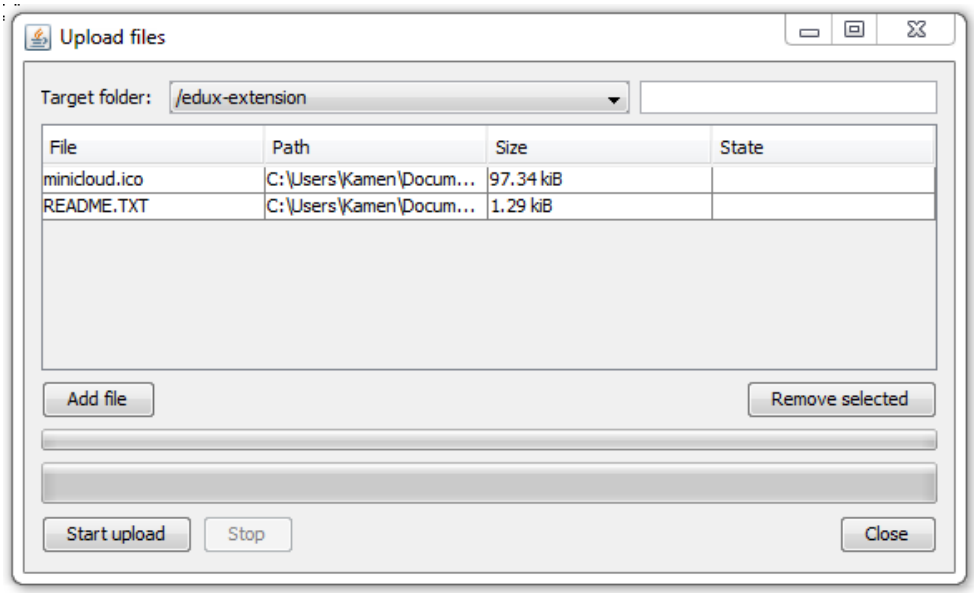
C.5.1 Hlavní obrazovka

Na hlavní obrazovce můžete vidět několik možností C.1.

V pravém horním rohu jsou tlačítka `Settings` a `Upload files`. Pomocí tlačítka `Settings` lze změnit heslo a nastavení šifrování. Pomocí `Upload files` poté lze nahrávat soubory na vzdálené úložiště.

Uprostřed aplikace můžete vidět výpis souborů uložených na vzdáleném úložišti a akce, které se dají se soubory provést.

Dále hlavní obrazovka obsahuje přepínání mezi výpisem vzdáleného úložiště a výpisem synchronizovaných složek.



Obrázek C.2: Dialog pro nahrávání souborů.

C.5.2 Nahrávání souborů

Nahrávání souborů spustíte kliknutím na tlačítko `Upload files` v hlavní obrazovce. Dialog vás poté vyzve k zadání cílové složky na vzdáleném úložišti a k výběru souborů k nahrání C.2. Soubory k nahrání přidáte kliknutím na tlačítko `Add file`. Přidané lokální soubory se zobrazují ve výpisu uprostřed obrazovky. Pomocí tlačítka `Remove selected` je možné vybrané soubory odebrat.

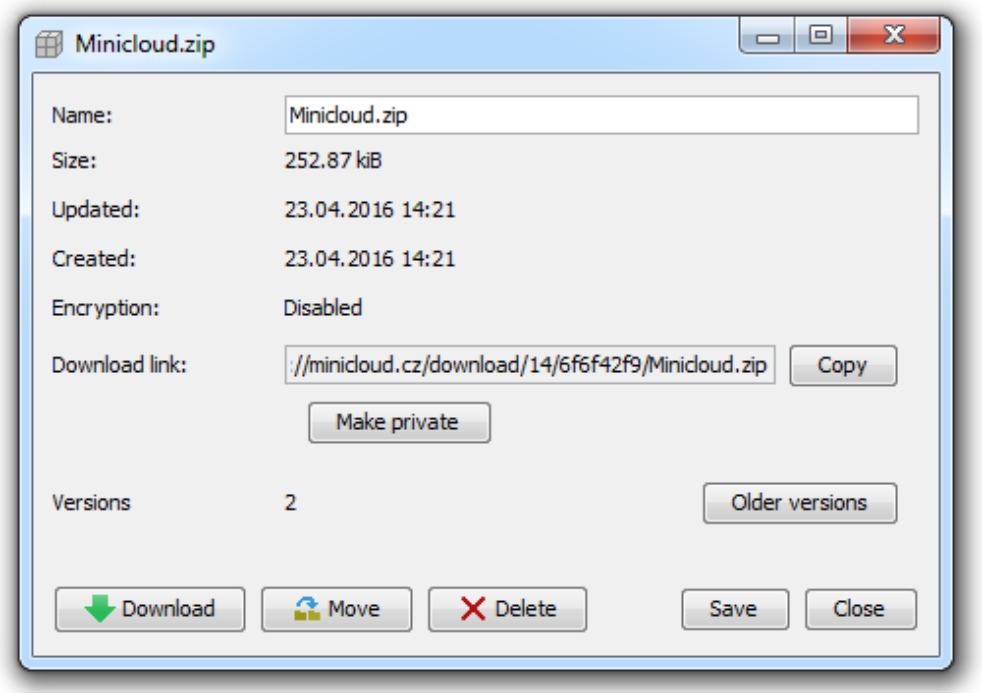
Nahrávání zahájíte tlačítkem `Start upload`. V průběhu nahrávání vás bude aplikace informovat o průběhu zobrazením stavu nahrávání u každého souboru. Nahrávání lze přerušit pomocí tlačítka `Stop`.

C.5.3 Změna hesla

Heslo lze změnit v dialogu nastavení. Ten zobrazíte tlačítkem `Settings` na hlavní obrazovce. Nastavení hesla je v sekci Profile. Heslo změníte zadáním nového hesla do polí `Password` a `Password again` pro ověření. Po zadání nového hesla změnu potvrdíte kliknutím na tlačítko `Save`.

C.5.4 Sdílení souboru

Pro sdílení souboru, musíte nejdříve soubor najít ve výpisu vzdáleného úložiště. Poté ho vybrat dvojitým kliknutím, čímž se zobrazí dialog detailu souboru C.3.



Obrázek C.3: Dialog detailu souboru.

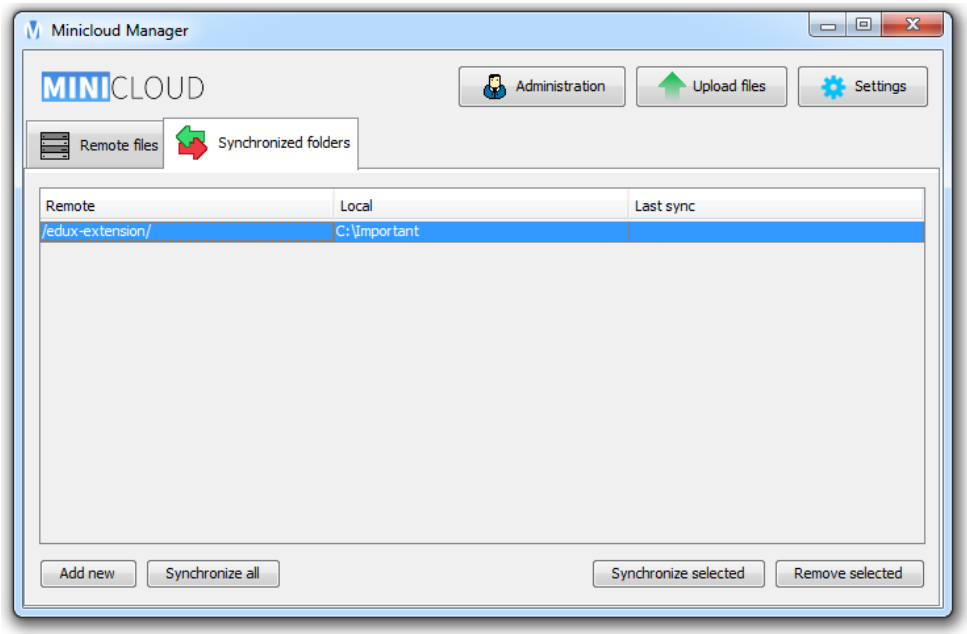
Pokud již je soubor veřejný, zobrazí se v kolonce `Download link` odkaz, který lze použít pro sdílení souboru.

Pokud je soubor soukromý, musí se nejdříve zveřejnit. To lze učinit kliknutím na tlačítko `Make public` pod kolonkou `Download link`. V případě že soubor je šifrovaný, bude potřeba nejdříve soubor nahrát v čisté formě.

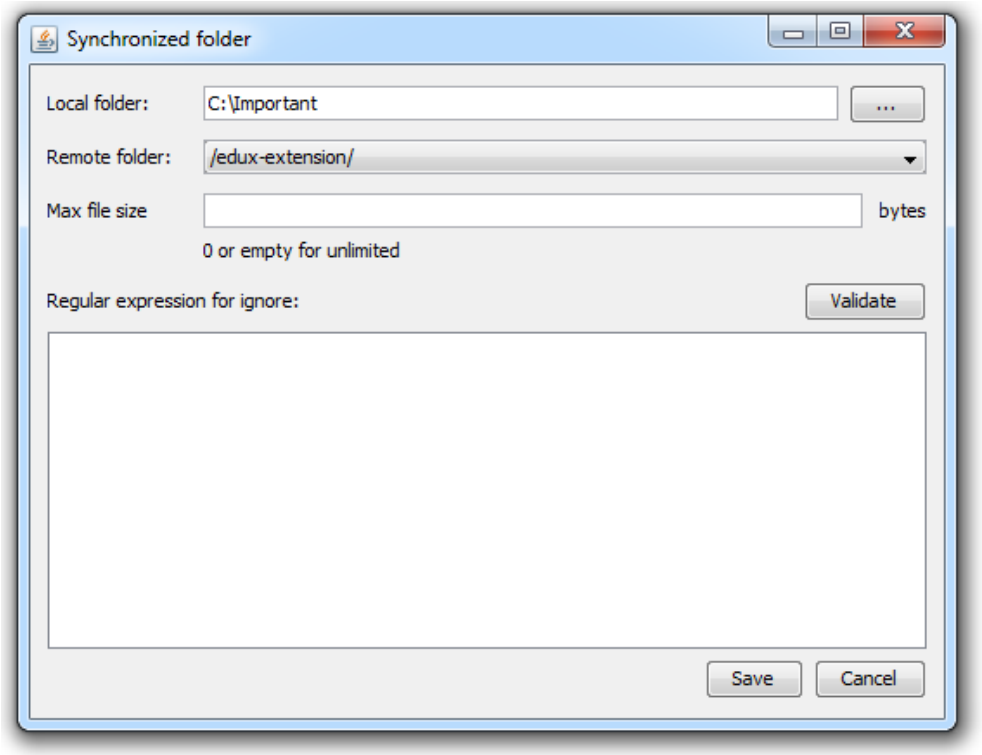
C.5.5 Synchronizace lokální složky

Seznam synchronizovaných složek najdete v hlavní obrazovce, v sekci `Synchronized folders` C.4. Novou synchronizovanou složku vytvoříte pomocí tlačítka `Add new`. Zobrazí se dialog, pomocí kterého se nastavuje synchronizovaná složka C.5. `Local folder` je cesta k složce na lokálním úložišti a `Remote folder` je cesta ke vzdálenému úložišti. Kromě těchto údajů lze navíc omezit maximální velikost synchronizovaného souboru, to znamená že soubory větší než zadaná velikost aplikace ignoruje. Nastavení potvrdíte tlačítkem `Save`.

Po uložení nastavení se synchronizovaná složka zobrazí ve výpisu. Synchronizaci poté zahájíte vybráním složky a kliknutím na tlačítko `Synchronize selected`, nebo `Synchronize all`.



Obrázek C.4: Výpis synchronizovaných složek.



Obrázek C.5: Dialog nastavení synchronizované složky.

Příloha D

Správcovská příručka

D.1 Instalace serveru

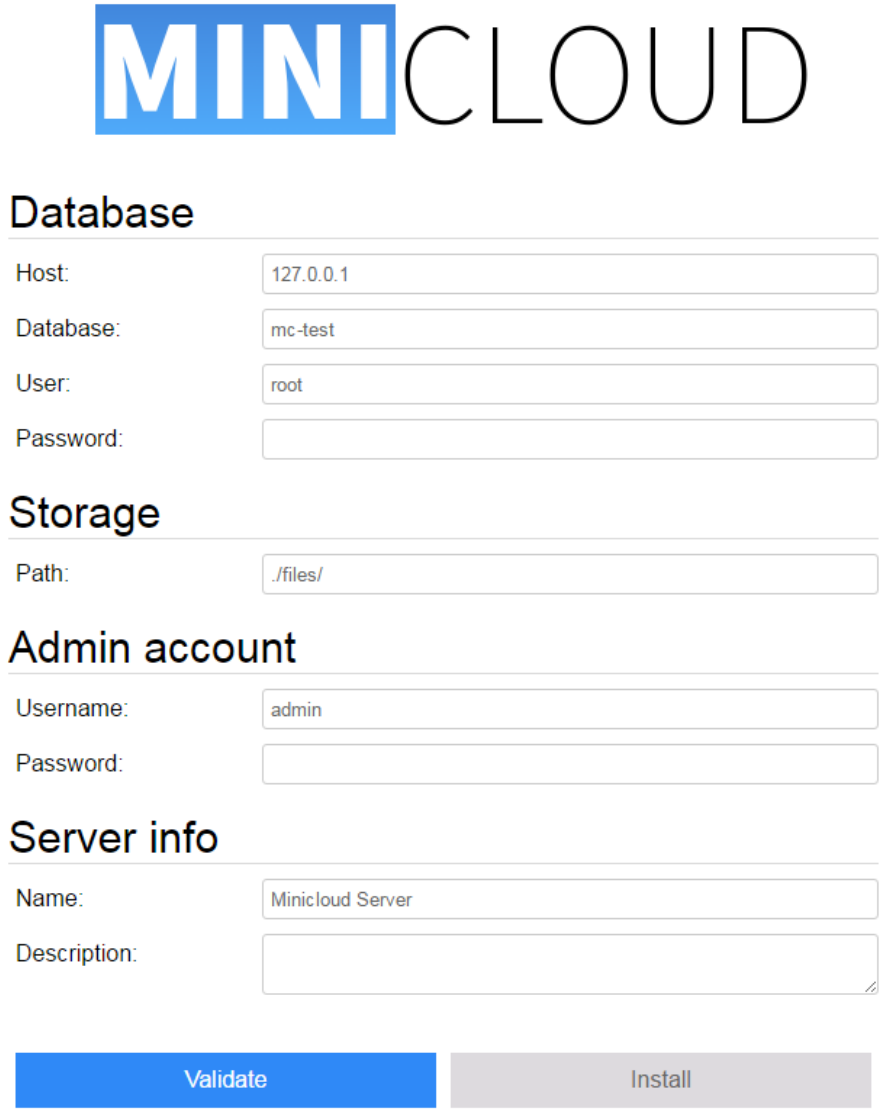
Server se instaluje rozbalením obsahu distribučního balíčku do lokace, kde bude server provozován. Po rozbalení je potřeba v prohlížeči otevřít /install.php a poté pokračovat podle instrukcí na instalační stránce D.1.

Po dokončení instalace bude, pomocí údajů zadaných na instalační stránce, vytvořen správcovský účet. Tento účet se používá pro správu server pomocí klientské aplikace.

D.2 Správa uživatelů

Pro správu uživatelů je potřeba přihlásit se pomocí klientské aplikace na správcovský účet. Po přihlášení se v pravém horním rohu aplikace objeví možnost `Administration`. Nově otevřený dialog zobrazí výpis uživatelů. Pomocí tlačítka `Delete` je možné vybrané uživatele a veškerý jejich obsah smazat.

Pro přidání uživatele je zde tlačítko `Add new`. V dialogu stačí vyplnit základní údaje a vybrat jestli bude uživatel také správce, nebo pouze běžný uživatel. Zadané heslo bude jednorázové a uživatel bude po prvním přihlášení vyzván k jeho změně.



Obrázek D.1: Instalace serveru.