FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Systém na bezpečné a efektivní zálohování dat
Student:	Michal Kunert
Vedoucí:	Ing. Zdeněk Muzikář, CSc.
Studijní program:	Informatika
Obor / specializace:	Bezpečnost a informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	do konce letního semestru 2021/2022

Pokyny pro vypracování

Navrhněte a implementujte systém na principu klient-server, který zabezpečí uživatelova data na klientském počítači synchronizací se serverem.

Na serveru využijte systém souborů ZFS a navrhněte synchronizační nástroj, který s využitím vlastností deduplikace a snapshotů bude průběžně archivovat data s minimálními nároky na úložný prostor a zároveň bude minimalizovat dopad případné nákazy ransomware.

Na serveru vytvořte grafické rozhraní, přes které bude k dispozici historie všech souborů.

Elektronicky schválil/a prof. Ing. Pavel Tvrdík, CSc. dne 28. prosince 2020 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Systém na bezpečné a efektivní zálohování dat

Michal Kunert

Katedra počítačových systémů
Vedoucí práce: Ing. Zdeněk Muzikář, CSc.

12. května 2021

Poděkování

Rád bych poděkoval Ing. Zdeňku Muzikářovi, CSc. za rady a pomoc při tvorbě této práce. Dále bych rád poděkoval rodině a přátelům za podporu během studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principu při 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 předpisu. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisu, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programu, 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 zpusobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelum). 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 zpusobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 12. května 2021 . . . . . .

Odkaz na tuto práci

Kunert, Michal. *Systém na bezpečné a efektivní zálohování dat.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2021.

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací systému, který dokáže zabezpečit uživatelská data pomocí synchronizace těchto dat mezi klientem a serverem. Na straně serveru je využit pro ukládání dat souborový systém ZFS. Součástí serverové části bude také webové rozhraní umožňující zobrazení historických i aktuálních verzí souborů. Práce začíná analýzou, ve které jsou specifikovány požadavky na řešení, jsou zde zhodnoceny konkurenční aplikace a také obsahuje popis použitelných technologií pro implementaci. Další kapitoly se zabývají návrhem a popisem implementace. K implementaci celého systému je využit programovací jazyk Python a pro vytvoření webové aplikace je použit webový framework Django. Pro evidování historických verzí souborů jsou použity snapshoty souborového systému ZFS. V poslední části práce jsou navržena možná vylepšení.

Klíčová slova zálohování, verzování souborů, webová aplikace, ransomware, WebDav, ZFS, Python

Abstract

The main aim of this bachelor thesis is to design and implement a system, which enables to secure user data by synchronizing this data between a client and a server. The server-side ZFS file system is used for data storage. The server part will also include a web interface that allows users to view files and historical revisions of those files. The thesis starts with the analysis, which contains specification of requirements for the solution, competing applications are evaluated here and this chapter also contains a description of suitable technologies for implementation. The next chapters deal with the design of the application and description of the implementation. A Programming language Python was used for implementation and Django framework was used for the creation of web application. ZFS filesystem snapshots are used for storing historical revisions of files. In the last part of the thesis, possible improvements are suggested.

Keywords backuping, file versioning, web application, ransomware, WebDav, ZFS, Python

Obsah

Seznam obrázků

Seznam výpisů kódu

Úvod

Moderní informační technologie se staly ve 21. století nedílnou součástí života každého člověka. Zařízení jako jsou mobilní telefony, počítače a různá embedded zařízení jsou hojně využívána pro práci i zábavu. Tyto technologie našly své uplatnění téměř ve všech odvětvích průmyslu a také ve státní správě.

S informačními technologiemi jsou úzce spojena data. S rychlým rozvojem internetu se začala tato data hojně vyměňovat mezi různými zařízeními napříč státy i kontinenty. Data často obsahují i citlivé údaje, které se týkají osob, firem a dalších subjektů. Toto je důvod, proč musí být chráněna před přístupem osob, kteří k nim nemají mít přístup. Ochrana dat je velice důležitá, ale také velice komplikovaná činnost.

Data musí být chráněna nejen před přístupem neoprávněné osoby, ale také před jejich úmyslným či neúmyslným smazáním. Velmi časté jsou také vyděračské útoky, kdy je cílem útočníka znepřístupnit data jejich majiteli. Opětovné zpřístupnění slibuje útočník až po zaplacení výkupného. Programu provádějící tyto útoky se říká ransomware. Ztráta dat je také velice často zapříčiněna poruchou zařízení, na kterém jsou uložena. Jedním z možných způsobů ochrany dat před jejich ztrátou je zálohování na externí datová média.

Tato práce se bude věnovat zabezpečení uživatelských dat a to hlavně z pohledu ochrany před jejich ztrátou, protože problém ztráty dat je stále aktuální a potýká se s ním mnoho lidí a firem. Text se bude skládat z analýzy požadavků na výsledné řešení, analýzy stávajících řešení a rešerše použitelných technologií. Na tyto kapitoly navážou sekce věnující se návrhu aplikace a její implementaci. Na konci se bude nacházet zhodnocení výsledného řešení včetně návrhu možných vylepšení.

Kapitola 1

Cíl práce

Hlavním cílem této práce je navrhnout a od základu implementovat systém, který pomůže uživatelům zabezpečit jejich data na klientských počítačích před úmyslným i neúmyslným smazáním. Data by měla být odesílána na úložiště mimo počítač (na server) se souborovým systémem ZFS. Dané řešení má také minimalizovat nároky na úložný prostor. Uživatel by také měl mít přístup ke svým datům odkudkoliv i mimo svůj počítač. Navržené řešení má být dostupné jak pro menší firmy, tak i pro domácnosti.

Tento cíl bude realizován pomocí jednotlivých dílčích cílů. Prvním cílem je sestavení požadavků na výsledné řešení. Zde budou popsány funkční i nefunkční požadavky a také případy užití. Dalším krokem je analýza již existujících nástrojů použitelných k zabezpečení uživatelských dat. U každého nástroje budou analyzovány jeho klady i zápory. Nakonec bude provedeno jejich celkové zhodnocení. Náplní třetího cíle je rešerše použitelných technologií vhodných pro použití v aplikaci. Předposledním krokem je navržení výsledného řešení v souladu s požadavky. A nakonec bude provedena samotná implementace aplikace dle návrhu.

Kapitola 2

Analýza

V analýze jsou definovány základní požadavky, které jsou kladeny na výsledné řešení spolu s rešerší použitelných technologií vhodných pro implementaci aplikace. Další část se věnuje současným řešením, která umožňují zabezpečení uživatelských dat.

2.1 Požadavky

Požadavky slouží k definování funkcionality, kterou bude výsledná aplikace disponovat. Dále umožňují definovat omezení kladená na výsledné řešení. Požadavky uvedené v této kapitole budou následně zohledněny při návrhu a implementaci aplikace.

2.1.1 Funkční

V této kapitole jsou popsány všechny funkční požadavky, které konkrétně popisují funkcionalitu výsledné aplikace.

F1 – Správa uživatelských účtů

Funkčnost systému bude dostupná pouze přihlášeným uživatelům prostřednictvím uživatelského účtu, který slouží k propojení osoby s jejími daty na serverové části. Uživatelé se budou moci přihlašovat a odhlašovat. Aplikace bude také umožňovat úpravu všech uživatelských účtů včetně jejich přidání a odebrání. Všichni uživatelé si také budou moci prohlédnout své osobní údaje. Aplikace musí zajistit, aby některé operace mohli vykonávat pouze účty se zvýšenými oprávněními.

F2 – Zabezpečení uživatelských dat

Systém bude zabezpečovat data uživatelů na jejich počítačích pomocí synchronizace s externím úložištěm v předem definovaném intervalu. Výsledné řešení tedy musí být typu klient-server. Systém musí být dále schopen minimalizovat dopady nákazy ransomware na klientské části. Naopak nebude umožněna oboustranná synchronizace se slučováním mezi klientem a serverem.

F3 – Evidence historie souborů

Systém bude evidovat historii všech souborů uložených na serverové části v uživatelem definovaných intervalech nebo také kdykoliv mimo tento stanovený čas. Aplikace musí také disponovat možností vytvoření pojmenované verze souboru. Historické verze úložiště, adresářů nebo souborů bude možné na serverové části obnovit. Tento obnovený obsah musí být možné stáhnout zpět do klientského počítače. Všechny historické verze budou moci být také úplně odstraněny.

F4 – Uživatelské kvóty

Aplikace bude umožňovat nastavení maximálního možného obsazení diskového prostoru serverové části pro každého uživatele. Pokud součet velikostí všech synchronizovaných souborů uživatele přesáhne nastavený limit, tak již nebude umožněno soubory synchronizovat. Do tohoto limitu se nezapočítávají historické verze úložiště a souborů. Uživatelé si svoje nastavení mohou prohlédnout v informacích o svém profilu.

F5 – Dostupnost souborů na serveru

Soubory na serverové části musí být přístupné přes grafické rozhraní, ve kterém si budou moci uživatelé nechat zobrazit seznam souborů a adresářů. Některé podporované typy souborů bude možné přímo zobrazit. U nepodporovaných souborů bude umožněno jejich stažení, které má být dostupné i u souborů s podporovaným zobrazení a u adresářů. Dostupnost včetně prohlížení a stahování musí být umožněna i u souborů a adresářů, které jsou součástí jakékoliv historické verze.

F6 – Manipulace se soubory

U aktuálních souborů uložených na serveru bude umožněna manipulace podobná té, která je nabízena obyčejnými průzkumníky souborů. Tyto operace nebudou dostupné u souborů, které jsou součástí historických verzí úložiště, adresářů nebo souborů.

2.1.2 Nefunkční

Jedná se o požadavky, které nesouvisí přímo s funkčností systému, ale definují omezení na provedení a design výsledného řešení.

N1 – Velikost dat

Aplikace bude ukládat uživatelská data s co nejnižšími nároky na úložný prostor. Stejné soubory by měly být uloženy pouze jednou. Historické verze souborů, adresářů a úložiště by měly být také uloženy s co nejnižšími nároky na úložný prostor.

N2 – Zabezpečení dat během přenosu

Data budou během přenosu mezi klientem a serverem zabezpečená tak, aby nedošlo k jejich odposlechu nebo změně během přenosu po neznámé síti.

N3 – Zabezpečení dat na serveru

Na serveru budou data uložena tak, aby k nim měl přístup pouze oprávněný uživatel, kterým může být správce systému nebo správce aplikace.

N4 – Platformy

Klientská aplikace by měla být multiplatformní, tudíž by uživatelé měli mít možnost instalace a spuštění na běžně používaných desktopových operačních systémech. Serverovou aplikaci by mělo být možno spustit na operačních systémech s podporou souborového systému ZFS.

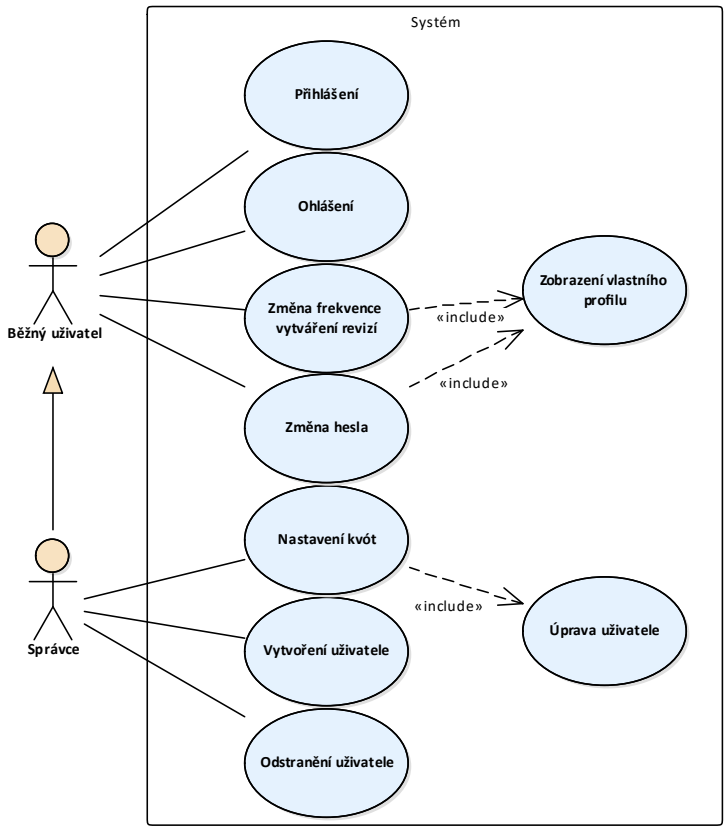
2.2 Případy užití

Případy užití vycházejí z požadavků, které jsou definovány v předchozí kapitole. Případy užití budou sloučeny do kapitol podle podobné funkcionality požadavků, které popisují. Ke kapitolám obsahujícím více případů užití jsou přidány obrázky, které slouží k lepší orientaci.

2.2.1 Správa účtů a úložiště

Správa uživatelů bude umožněna přes webové rozhraní serverové části. V aplikaci budou existovat dva typy účtů, jedná se o běžný účet a o účet správce. Účet správce aplikace je takový, který je vytvořen při instalaci aplikace. Mimo to může být jakýkoliv účet povýšen na účet administrátorský na stránce umožňující úpravu uživatele. Běžní uživatelé se budou moci přihlásit pomocí uživatelského jména a hesla, zobrazit si svůj profil a odhlásit se. V zobrazení profilu uvidí své osobní údaje a aktuálně využitý prostor úložiště včetně limitu

jeho využitelnosti. Dále si zde mohou uživatelé nastavit frekvenci automatického vytváření nových revizí celého úložiště. Bude možné nastavit měsíční, týdenní, denní a hodinovou frekvenci. Administrátoři musí mít navíc možnost nastavovat kvóty pro každého uživatele, upravovat jejich osobní údaje pomocí formuláře a mazat uživatelské účty. Vytváření uživatelů včetně jejich úložišť bude také umožněno pouze správcům. Všichni uživatelé si budou moci změnit své vlastní heslo po zadání původního hesla. Obnovování zapomenutého hesla bude opět provádět správce aplikace.

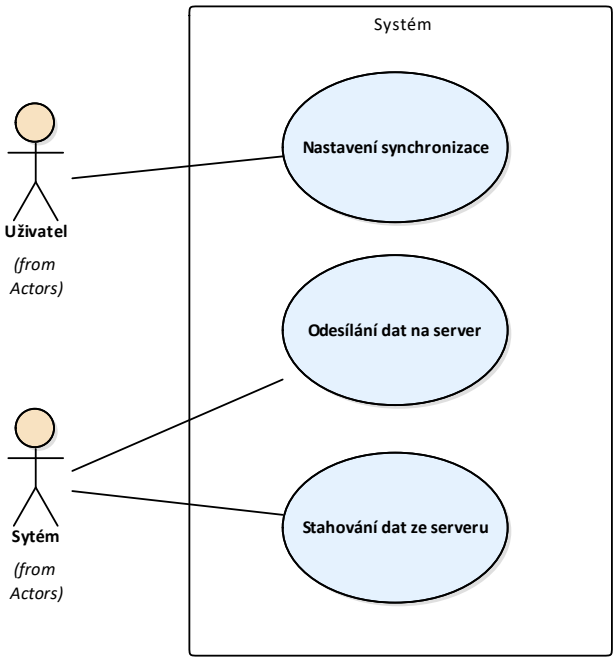


Obrázek 2.1: Diagram případu užití pro administraci ve webové aplikaci

2.2.2 Synchronizace dat

O synchronizaci dat se má starat klientská aplikace, která bude nainstalována na počítači obsahujícím zabezpečovaná data. Tato aplikace bude mít za cíl odesílání dat na server a také jejich stahování z tohoto serveru. Synchronizace se má řídit nastaveními, která bude možné změnit přímo v aplikaci. V nastavení bude možné měnit frekvenci synchronizace, synchronizované adresáře

a také údaje o serverové části. Uživatel si vždy musí vybrat, zdali si přeje data ze serveru stahovat, nebo je na vzdálené úložiště odesílat. Uživatelská data musí být synchronizována v zadaných intervalech se serverem, proto je nezbytné, aby klientská aplikace běžela neustále v systému jako služba.



Obrázek 2.2: Diagram případu užití pro klientskou aplikaci

2.2.3 Práce se soubory na serveru

Uživatelská data budou na serveru dostupná přes webové rozhraní, ke kterému mohou uživatelé přistupovat pomocí svých webových prohlížečů. Toto webové rozhraní bude po přihlášení nabízet možnost zobrazení obsahu adresářů a některých typů souborů. Navigace mezi soubory a adresáři musí být podobná jako u většiny správců souborů. Podporovanými typy souborů pro otevření jsou následující formáty – PDF, JPG, JPEG, PNG, GIF, BMP, MP4 MP3 a prostý text. Všechny typy souborů bude možné stáhnout, odstranit, přejmenovat, kopírovat a nebo přesunout. Tyto operace budou dostupné pouze u aktuálních souborů přes kontextové menu a některé z těchto operací budou podporovat vícenásobný výběr. V kontextovém menu aktuálního souboru nebude chybět možnost vytvoření nové pojmenované verze souboru, která se vytvoří po zadaní jména této nové verze.

2.2.4 Práce s historickými verzemi

Historické verze dat bude moci uživatel spravovat přes webové rozhraní serverové části. Aplikace musí umožňovat vytvoření historické verze celého úložiště (všech souborů) a nebo pojmenované verze jednotlivých souborů.

Historické verze úložiště budou vznikat automaticky v závislosti na nastavení profilu uživatele, ale také i mimo tento stanovený čas po kliknutí na tlačítko. Již existující verze si budou moci uživatelé prohlédnout v seznamu verzí. U každé historické verze bude dostupná operace pro její odstranění, obnovení a otevření. Obnovení verze způsobí nahrazení aktuálního obsahu úložiště obsahem dat v historické verzi a také odstranění novějších pojmenovaných i nepojmenovaných historických verzí úložiště. Aplikace dále umožní pracovat s adresáři a soubory v historických verzích, tak jak je to popsáno 2.2.3 s výjimkou operací měnící obsah souborů či adresářů. Tyto operace nebudou podporovány. Jakýkoliv soubor nebo adresář bude také možné obnovit, ale na rozdíl od obnovení celého úložiště nedojde k ovlivnění jiných historických verzí.

Vznik pojmenované verze je popsán v kapitole 2.2.3. Zobrazení všech pojmenovaných verzí souboru bude umožněno v kontextovém menu souboru aktuální verze. U každé pojmenované verze musí existovat možnost jejího odstranění, otevření, respektive stažení. Historickou verzi bude také možno obnovit bez ovlivnění jiných verzí.

2.3 Existující řešení

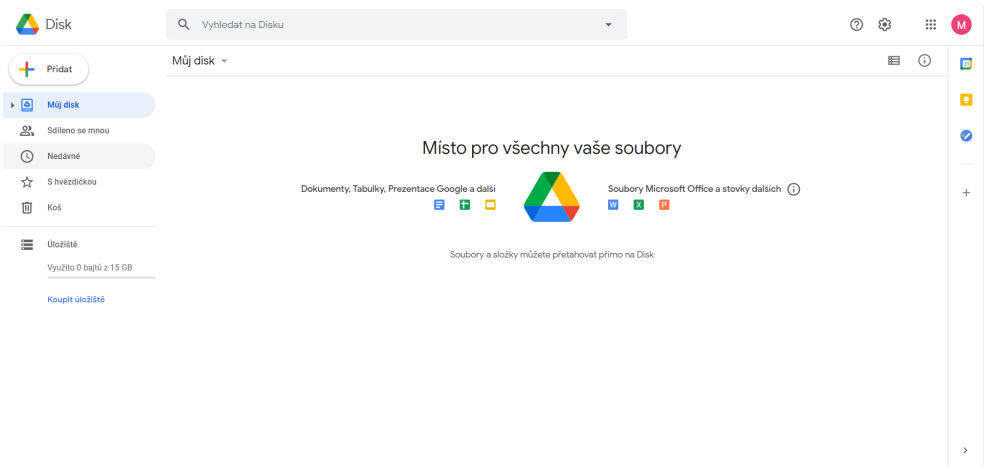
Tato kapitola se bude věnovat již existujícím řešením, která dokáží zabezpečit uživatelská data s cílem ochrany před nechtěnou úpravou, smazáním a také ransomware útoky. U všech řešení budou rozebrány jejich výhody a nevýhody. Nakonec bude uvedeno celkové zhodnocení těchto nástrojů s ohledem na požadavky, které jsou definovány v předchozí kapitole.

2.3.1 Disk Google

Jedná se o cloudové úložiště vyvinuté a provozované společností Google. Tato služba podporuje oboustrannou synchronizaci uživatelských dat se vzdálenými servery, což umožňuje mít stále aktuální data na více uživatelských zařízeních. Toto řešení je vhodné pro osobní i firemní použití. Serverová část aplikace je provozována na serverech provozovatele, tudíž pro synchronizaci souborů je nutné nainstalovat pouze klientskou aplikaci. Tuto službu je také možné využívat i bez klientské aplikace, protože podporuje nahrávání souborů pomocí prohlížeče. Díky tomu, že je Disk Google jedna z mnoha služeb poskytovaných společností Google, je toto úložiště velice dobře integrované s kancelářským balíkem a dalšími službami od Googlu. [1]

Uživatelé této služby nemají nad svými daty úplnou kontrolu, protože se tato data nacházejí na serverech provozovatele. Google nicméně uvádí, že uložená data jsou zašifrovaná a že zaměstnanci společnosti nemají k samotným datům přístup. [2]

Na druhou stranu je toto řešení velmi jednoduché a pro používání není nutná složitá instalace serverové části. Další výhodou této služby je velmi jednoduché sdílení souborů a spolupráce na nich. Nevýhodou aplikace je to, že neuchovává historii souborů po neomezenou dobu, což nemusí stačit v případě nákazy ransomware. Aplikace dále neumožňuje obnovení celého úložiště do předchozí verze. Uživatelé mohou využít zdarma úložný prostor o velikosti 15 GB. Úložiště s větším úložným prostorem jsou nabízena za poplatek. Klientská aplikace je poskytována pro zařízení s operačními systémy Microsoft Windows, Android a iOS. Naopak pro linuxové operační systémy neexistuje oficiální klientská aplikace. [1]

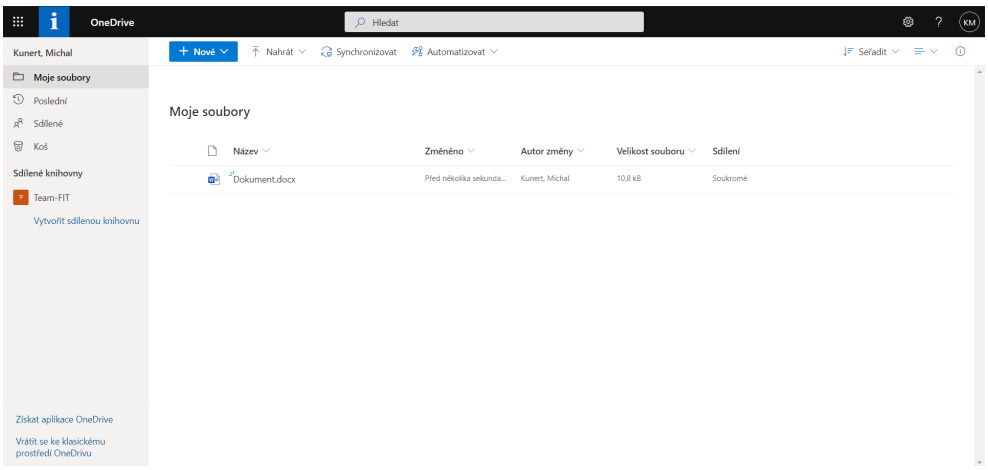


Obrázek 2.3: Webové rozhraní aplikace Google Disk [1]

2.3.2 Microsoft OneDrive

Dalším z cloudových úložišť je OneDrive od společnosti Microsoft. Toto řešení je velmi podobné službě Google Disk. Jedna z výhod tohoto řešení je velmi dobrá provázanost klientské aplikace s operačním systémem Microsoft Windows. Díky této provázanosti jsou do klientského zařízení stahovány pouze používané soubory, čímž se šetří místo na tomto zařízení. Klientská aplikace také existuje pro mobilní operační systémy Android a iOS. Tato služba také nabízí obnovení celého úložiště, sdílení souborů a také kancelářský balík Microsoft Office. Toto úložiště je dostupné pro osobní i firemní použití. Opět se jedná o proprietární řešení s malým přehledem o datech, protože uživatelská data

jsou opět uložena na serverech společnosti Microsoft. Cenová politika je také velmi podobná politice společnosti Google. [3]



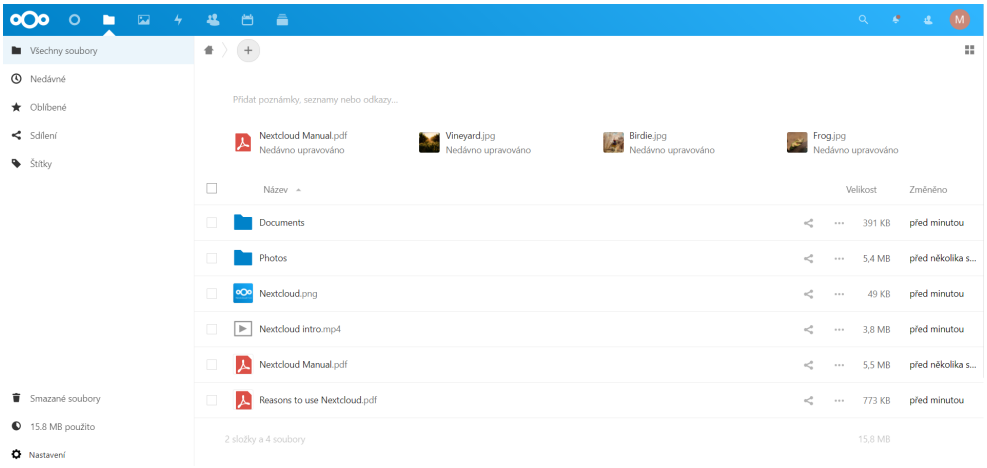
Obrázek 2.4: Webové rozhraní aplikace OneDrive [3]

2.3.3 Nextcloud

Dalším řešením pro hostování souborů je NextCloud, který vznikl oddělením od aplikace ownCloud. Obě aplikace si jsou velmi podobné, proto zde bude popsán pouze NextCloud, který je v poslední době populárnější. [4, 5]

NextCloud se skládá z klientské a serverové části. Serverová část uchovává soubory a také nabízí webové rozhraní. Jedná se o otevřenou aplikaci, kterou je možné nainstalovat na vlastní server. Díky této možnosti mají uživatelé plnou kontrolu nad svými daty, která jsou na vzdálené úložiště ukládána. Webové rozhraní aplikace NextCloud nabízí nástroje pro zobrazení a úpravu souborů. Klientská aplikace je dostupná pro všechny běžně používané operační systémy. Dále nechybí možnost nastavení kvót pro jednotlivé uživatele. Toto řešení nenabízí možnost obnovení celého úložiště do předchozí verze, ale nabízí možnost detekce útoku ransomware a obnovení zašifrovaných souborů. Detekce a obnovení poškozených souborů je možná díky aplikaci Ransomware Recovery, která byla vyvinuta na Kostnická univerzitě v Německu. [4, 6]

NextCloud dále umožňuje sdílení souborů mezi uživateli, obchod s aplikacemi a také chat. Použití tohoto systému je zdarma, ale je možné využít i placenou Enterprise verzi, která poskytuje rychlejší podporu a konzultace [7]. Toto řešení není primárně určeno pro běžné uživatele jako předchozí dvě řešení, protože pro provoz je potřeba serverovou část nainstalovat na vlastní server nebo na webhosting. [4]



Obrázek 2.5: Webové rozhraní aplikace NextCloud [4]

2.3.4 Pydio

Další open-source nástrojem je Pydio. Tato aplikace je svými vlastnostmi velmi podobná nástroji NextCloud. Pydio nicméně umí lépe pracovat s velkými soubory. Aplikace je dostupná ve dvou verzích. Uživatelé si mohou vybrat mezi verzí pro domácí užití a pro firemní účely. [8]

2.3.5 Nástroj rsync

Jedná se o aplikaci, která pomáhá synchronizovat data mezi dvěma umístěními. Tato umístění mohou být jak lokální, tak i vzdálená. Snahou nástroje je minimalizace datových přenosů tím, že přenáší pouze změny, které v souborech nastaly. Při přenosu souborů může být použita komprese a komunikace může probíhat také zabezpečeně. Tento nástroj umožňuje kopírování dat se zachováním přístupových práv a také práv vlastnických. Dále umožňuje kopírování symbolických linků. Tato aplikace může být využita k zálohování souborů, kdy se soubory budou na vzdálený počítač posílat pomocí protokolu SSH. [9]

Tento nástroj je možné použít pro zálohování dat, ale sám o sobě nenabízí možnost webového rozhraní s podporou zobrazení souborů na serverové části. Rsync také nedokáže zmírnit následky nákazy ransomware bez úprav úložiště obsahující zálohu souborů. Tato aplikace také neumožňuje evidování historií souborů a jednoduché obnovení celého úložiště do předchozího stavu. Pro automatické odesílání dat na server by při použití této aplikace byl zapotřebí další nástroj.

```
# Synchronizace souborů a adresářů
# se zachováním metadat
rsync --archive --delete --acls . /server
```

Výpis kódu 2.1: Použití nástroje rsync

2.3.6 Duplicati

Duplicati je open-source nástroj, který slouží k zálohování uživatelských souborů. Jedná se o aplikaci, která v zadaných časových intervalech ukládá šifrované, přírůstkové a komprimované zálohy na záložní úložiště. Toto úložiště může být některé z podporovaných cloudových úložišť (OneDrive, Google Disk, Box.com, Amazon Cloud Drive, ...), dále je možné ukládat data na servery podporující FTP, SSH a WebDav. Data mohou být na tato úložiště ukládána zašifrována například pomocí šifry AES s délkou klíče 256 bitů. Klientská aplikace se nastavuje přes webové rozhraní nebo přes konzolovou aplikaci. Uživatelé si mohou nastavit zálohované adresáře, umístění záloh, četnost zálohování a obnovují se zde také data ze zálohy. Na druhou stranu tento nástroj nedisponuje webovým rozhraním na serverové části a neumožňuje vytvoření pojmenovaných revizí souborů. [10]

Při použití tohoto nástroje mají uživatelé nad svými daty plnou kontrolu, protože se jedná o open-source aplikaci s podporou ukládání na vlastní úložiště, například NAS. Ukládání dat do cloudových úložišť, kterými může být například již zmíněný OneDrive nebo Google Disk, je také díky šifrování bezpečné. Tento nástroj také dokáže zmírnit dopady nákazy ransomware.

2.3.7 Zhodnocení

Bylo porovnáno celkem šest nástrojů, které lze použít k zabezpečení uživatelských dat. Microsoft OneDrive a Google Disk jsou cloudová úložiště, která slouží primárně k synchronizaci souborů mezi zařízeními, ale umožňují také velmi dobře chránit uživatelská data, včetně ochrany souborů před poškozením způsobeným nákazou ransomware. Tato data jsou chráněna, protože tyto aplikace uchovávají i historie souborů. Oba tyto nástroje také disponují možností tvorby pojmenovaných verzí souborů. Jedná se o řešení, která jsou velmi jednoduchá na použití v domácnostech i ve firmách. Nevýhodou těchto systémů je jejich uzavřenost a téměř nemožná kontrola nad daty uloženými na vzdálených serverech.

Další dvě aplikace, kterými jsou Pydio a NextCloud, si mohou uživatelé nainstalovat na své vlastní servery, což je určeno hlavně pro uživatele, kteří chtějí mít přehled o svých datech a nechtějí je nahrávat na servery cizích společností. Opět se ale jedná spíše o cloudová úložiště, než o systémy pro zálohování.

Dále byly porovnány nástroje, které umožní synchronizaci souborů a také aplikace, které slouží přímo k zálohování souborů. Těmto aplikacím chybí především možnost zobrazení souborů přes webové rozhraní na serverové části.

Při porovnání byly zohledněny požadavky popsané v kapitole 2.1, ale žádný z uvedených systémů pro zabezpečení uživatelských dat plně nepodporuje požadovanou funkcionalitu, nebo jsou pro zadané požadavky příliš komplexní. Z těchto důvodů se autor této práce rozhodl navrhnout a implementaci vlastní systému pro zabezpečení uživatelských dat.

2.4 Ransomware

V požadavcích na výsledné řešení se nachází, že by celý systém měl minimalizovat dopady nákazy ransomware na klientském počítači, proto v této kapitole bude popsán problém nákazy ransomware.

Jedná se o druh škodlivého kódu, který infikuje počítač oběti za účelem zašifrování uživatelských dat a nebo zablokování celého počítače. Ransomware se může dále rychle šířit na další počítače prostřednictví již infikovaných počítačů. Zašifrovaná data jsou pro oběti nepoužitelná, protože neznají klíč k jejich dešifrování. Útočník požaduje za zpřístupnění těchto dat výkupné většinou ve formě kryptoměny Bitcoin nebo jiné měny, u které je nemožné vystopovat příjemce platby. Některé druhy tohoto škodlivého kódu hrozí zveřejněním citlivých informací majitele počítače, pokud nedojde k zaplacení výkupného. V poslední době začaly být cílem útoků větší organizace před jednotlivci, protože tyto organizace disponují větším rozpočtem a mohou zaplatit vyšší výkupné. Dále tyto organizace vlastní soubory, které jsou důležité pro provoz těchto společností. Ransomware se nejčastěji šíří pomocí phishingového e-mailu, ale také pomocí bezpečnostních chyb v aplikacích používanými obětmi. Mezi známé vyděračské aplikace se řadí CryptoLocker, TorrentLocker, WannaCry a další. V této době se většina útoků zaměřuje na operační systém Microsoft Windows, ale existují již i programy, které se zaměřují na operační systém Android. Popularita nákazy ransomware je u počítačových zločinců velmi vysoká, protože dokáže těmto zločincům vydělat až miliony liber měsíčně. [11, 12] V roce 2019 bylo provedeno kolem 188 milionů ransomware útoků [13].

2.5 Rešerše použitelných technologií

V této kapitole budou uvedeny jednotlivé technologie, které by mohly být použity pro výsledné řešení na základě požadavků definovaných v předchozích kapitolách. U každé technologie bude uveden pouze její popis. Detailní umístění včetně použití bude uvedeno v kapitole zabývající se návrhem řešení.

2.5.1 HTTP

Jedná se o protokol, který umožňuje načítání zdrojů ze vzdálených serverů, které se nazývají webové. Klientem je nejčastěji webový prohlížeč, který se také stará o zobrazování výsledné stránky. Tato webová stránka může být složena ze zdrojů jako jsou dokumenty (nejčastěji HTML), obrázky a také videa. Jedná se o protokol aplikační vrstvy, který se používá TCP vrstvu. HTTP lze také použít k načtení částí dokumentů a k aktualizaci webových stránek na vyžádání. [14]

Komunikace mezi oběma stranami probíhá pomocí zpráv, přičemž klient je iniciátorem komunikace. Zprávy odesílané klientem se nazývají požadavky a zprávy posílané servery se nazývají odpovědi. Klientský požadavek se skládá z metody definující operaci, kterou chce klient provést (GET, POST, OPTIONS, HEAD, atd.). V požadavku se dále vyskytuje cesta ke zdroji, verze protokolu, volitelné hlavičky a případná data (například u metody POST). V odpovědi serveru se nachází verze protokolu, návratový kód (200 – úspěch, 404 – dokument nenalezen, 500 – vnitřní chyba serveru a mnohé další), textový popis návratového kódu a také data. [14]

HTTP protokol sám o sobě neumožňuje šifrování komunikace. Pro zabezpečení přenášených dat se tento protokol používá ve spojení s TLS nebo SSL. Toto spojení se pak nazývá HTTPS. Samotný HTTP protokol obvykle používá TCP port 80, ale HTTPS spojení běžně využívá TCP port 443. [14]

Jedná se o jednoduchý a pomocí hlaviček rozšiřitelný protokol. Další jeho vlastností je bezstavovost. Tato vlastnost neumožňuje uchovávat stav komunikace a jednotlivé dotazy tak mezi sebou nesouvisí. Tato vlastnost je negativní v případě složitějších webů, a proto byl tento protokol rozšířen o HTTP cookies, které umožňují udržet stav relace. Existuje několik verzí tohoto protokolu, přičemž nejstarší je označena jako 0.9 a vyvinul ho Tim Berners-Lee a jeho tým v letech 1989–1991. [14, 15]

2.5.2 WebDav

World Wide Web Distributed Authoring and Versioning je rozšíření protokolu HTTP verze 1.1, které umožňuje provádět vzdálenou správu souborů, které jsou uloženy na webovém serveru. Tento standard, který je definovaný RFC 4918, používá HTTP protokol pro zajištění komunikace mezi klientem a serverem, na kterém jsou uloženy soubory. Klientské aplikace mohou využít standardní HTTP metody, ale také některé další definované tímto rozšířením (PROPFIND, LOCK, MKCOL, atd.). Součástí každého požadavku a odpovědi může být XML dokument, který obsahuje informace upřesňující komunikaci. V tomto strukturovaném dokumentu se nacházejí například informace o metadatech souborů. [16]

WebDav protokol umožňuje vzdálenou práci se soubory a adresáři na serveru. Soubory je možno mazat, kopírovat, přesouvat, stahovat a také nahrávat

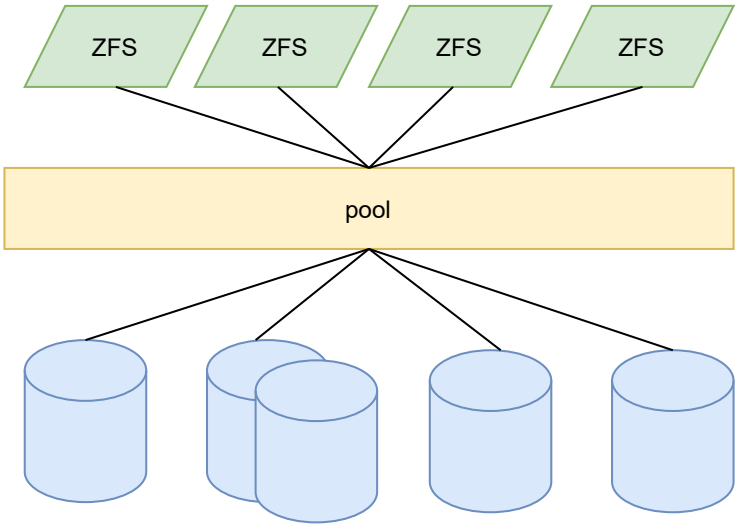
na server. Ke každému souboru je možno přidávat a upravovat jejich vlastnosti. WebDav dále podporuje uzamykání souborů, které je má chránit před nechtěnou úpravou. Existují dva typy zámků Exclusive Lock a Shared Lock. Další možností je práce s adresáři, které jsou ze strany tohoto rozšíření chápány jako kolekce. WebDav dále zavádí nové stavové kódy pro HTTP odpověď serveru. Jedná se například o kód 423, který značí zámek na cílovém nebo zdrojovém prostředku. [16]

Díky tomu, že je WebDav pouhé rozšíření HTTP protokolu, podporuje jej velké množství webových serverů. Jedná se například o Apache, Nginx, lighttpd a další. Dále pro tento protokol existuje velké množství klientů na různých platformách. K dalším výhodám plynoucím z HTTP protokolu se řadí umožnění komunikace se server, který má striktně nastaveným firewallem. [17]

2.5.3 ZFS

ZFS je souborový systém vyvinutý společností Sun Microsystems pro operační systém OpenSolaris v roce 2005. Následně byl tento souborový systém uvolněn pod open-source licencí CDDL. ZFS odstraňuje některé nevýhodné vlastnosti jiných souborových systémů. Mezi jeho hlavní vlastnosti patří princip copy on write, kdy jakákoliv změna dat probíhá transakčně, čímž je změna jako celek přijata, anebo zamítnuta. Zařízení pro uchovávání dat jsou sdružována do poolů. Pool je sám o sobě souborový systém, v němž je možné hierarchicky vytvářet další ZFS souborové systémy. Z poolu se alokují bloky dat různých velikostí, které mají stromovou strukturu. ZFS je zaměřen na integritu dat, proto je pro každý datový blok vypočten kontrolní součet, který je následně uložen v nadřazeném bloku. Pokud se při čtení datových bloků neshoduje uložený a při čtení vypočtený kontrolní součet, dokáže ZFS poškozený datový blok opravit, pokud datové úložiště obsahuje redundantní data. Obsahy datových bloků mohou být komprimované nebo šifrované. [18, 19]

ZFS dále podporuje tvorbu snapshotů. Jedná se o zachycení stavu uložených dat v určitém časovém okamžiku. Snapshoty jsou ukládány tak, aby docházelo k co nejmenším nárokům na úložiště a jsou přístupné pouze pro čtení. ZFS umožňuje obnovení těchto dat a také jejich prohlížení. Pro úsporu úložného prostoru je podporována deduplikace, která umožňuje uložení více stejných bloků dat pouze jednou. RAID-Z je další z vlastností, která umožňuje zabezpečení dat v případě poruchy disku. Kapacita tohoto souborového systému je téměř nevyčerpatelná, protože umožňuje uložení 16 miliard GB dat a v jednom adresáři může být až 2^{48} souborů. ZFS dále nabízí jednoduchou administraci, která umožňuje snadné vytváření a správu souborových systémů bez nutnosti použití více příkazů. Podpora kvót na maximální využitelné místo pro uživatelské soubory je v tomto systému nativní. [20]



Obrázek 2.6: Architektura ZFS

2.5.4 Webová aplikace

Jedná se o aplikaci, která je poskytována uživatelům vzdáleně pomocí sítě. Nejčastěji ji poskytují webové servery ve spolupráci s aplikačními a databázovými servery. Mezi výhody těchto aplikací patří vysoká kompatibilita, protože uživatelé přistupují k těmto aplikacím pomocí webových prohlížečů, které jsou dostupné pro všechny běžně používané platformy. Další předností tohoto typu aplikace je snadné používání bez nutnosti instalace. Při použití webových aplikací také nemusejí uživatelé provádět jejich aktualizace, protože se o vše stará provozovatel dané služby. V neposlední řadě se dají provozovat i na méně výkonném zařízení, protože se o výpočetní výkon starají vzdálené servery. K nevýhodám těchto aplikací se řadí nutnost internetového připojení a také to, že uživatelé mají menší kontrolu nad svými daty. [21, 22]

Webové aplikace fungují díky kombinaci kódu na straně serveru a klienta. Kód na straně serveru se většinou zabývá ukládáním a načítáním dat z databáze. Dále jsou tato data serverovou částí zpracovávána. K tvorbě serverové části jsou vhodné jazyky jako Python nebo Java. Pro tvorbu části zpracovávané na klientském zařízení se nejčastěji používají jazyky jako je JavaScript, kaskádové styly (CSS) a HTML. Kód na straně klienta se zabývá prezentací informací uživateli. [22]

2.5.5 HTML

Jedná se o značkovací jazyk sloužící pro tvorbu dokumentů, které tvoří webové stránky. HTML jazyk určuje strukturu a obsah těchto stránek zobrazovaných pomocí webových prohlížečů. HTML také umožňuje propojení jednotlivých

stránek mezi sebou hypertextovými odkazy. Každý HTML dokument je tvořen elementy, které určují strukturu a význam jejich obsahu. Element je většinou tvořen otevírací značkou, samotným obsahem a uzavírací značkou. K jednotlivým elementům mohou být přidány také atributy. [23]

2.5.6 CSS

Jedná se je jazyk sloužící pro nadefinování způsobu grafického zobrazení elementů z HTML dokumentu. Tento jazyk umožňuje úplné oddělení vzhledu dokumentu od jeho obsahu a struktury. Kaskádové styly mohou být uloženy v externích souborech, kterým se říká šablony. Toto řešení dokáže šetřit čas, protože umožňuje nadefinování stylu pro více HTML dokumentů. [24]

2.5.7 SSH Protokol

SSH je protokol sloužící pro připojení ke vzdálenému počítači ve formě terminálu. Tento protokol je na rozdíl od *telnetu* šifrovaný. SSH také umožňuje síťové tunelování, což je přenos jakýchkoliv dat přes nezabezpečený síťový kanál. SSH pracuje na principu klient-server, kde server většinou naslouchá na TCP portu 22. Pro SSH protokol existuje velké množství klientů, jedná se například o PuTTY nebo WinSCP. OpenSSH a Tectia SSH jsou naopak servery podporující protokol SSH. [25]

2.5.8 Python

Jedná se o vysokoúrovňový, interpretovaný, dynamicky typovaný programovací jazyk, který je vyvíjen jako open-source. Python podporuje objektové, procedurální, ale i funkcionální programování. Podle [26] je Python jedním z nejoblíbenějších programovacích jazyků a jeho oblíbenost stále roste. Python najde uplatnění v různých aplikacích a to díky velkému množství balíčků, kterými je rozšířen. Uplatnění tohoto jazyku můžeme najít například u webových aplikací, u aplikací zabývajícími se vědeckými výpočty a zpracováním dat. Dále nechybí možnost vytvářet grafické uživatelské rozhraní aplikace pomocí balíčku *Tk*. Python je také označován za jazyk, který je jednoduchý na naučení. [27]

2.5.9 Django

Django je open-source webový framework, který je napsán v programovacím jazyce Python. Tento framework byl vyvíjen od roku 2003 jako součást redakčního systému deníku Lawrence Journal-World. O několik let později byl uvolněn jako open-source projekt pod BSD licencí. Postupem času se Django stalo oblíbeným webovým frameworkem. Mezi výhody Djanga patří automatické generování administračního rozhraní, podpora MVC modelu, automatické generování formulářů z databázových modelů, objektový přístup k databázi,

odlehčený webový server určený pro vývoj a také snaha o co nejmenší možné opakování kódu. Tento framework také umožňuje jednoduchou lokalizaci pomocí unixového nástroje gettext. [28]

2.5.10 JavaScript

JavaScript je interpretovaný objektový programovací jazyk. Tento jazyk se primárně používá při tvorbě webů. Původně se kód napsaný v JavaScriptu vykonával pouze na klientské straně pomocí webových prohlížečů, kterými byl také stažen ze serveru spolu s HTML dokumentem a kaskádovými styly. JavaScript se v poslední době rozšířil také na stranu serveru, kde umožňuje generovat HTML stránky. K tomuto účelu existuje mnoho webových frameworků pro JavaScript. Jedná se například o Vue.js nebo React. [29]

2.5.11 AJAX

Asynchronous JavaScript and XML je technologie postavená na programovacím jazyce JavaScript. Tato technologie umožňuje provádět změny webové stránky bez nutnosti jejího nového načtení. Data ze serveru jsou načtena pomocí asynchronní komunikace na pozadí. Díky této technologii je práce s webovými aplikacemi rychlejší a uživatelsky přívětivější. AJAX umožňuje vytvoření různých našeptávačů, single-page aplikací a má také mnoho dalších využití. [30]

2.5.12 jQuery

Jedná se o knihovnu jazyku JavaScript, která umožňuje jednodušší výběr a úpravu elementů v HTML dokumentu pomocí JavaScriptu. Tato knihovna také ulehčuji práci s technologií AJAX. Knihovna jQuery je podporována mnoha webovými prohlížeči. [31]

Kapitola 3

Návrh

Tato kapitola se věnuje návrhu výsledného systému, který je navrhován na základě funkčních a nefunkčních požadavků uvedených v kapitole číslo 2.

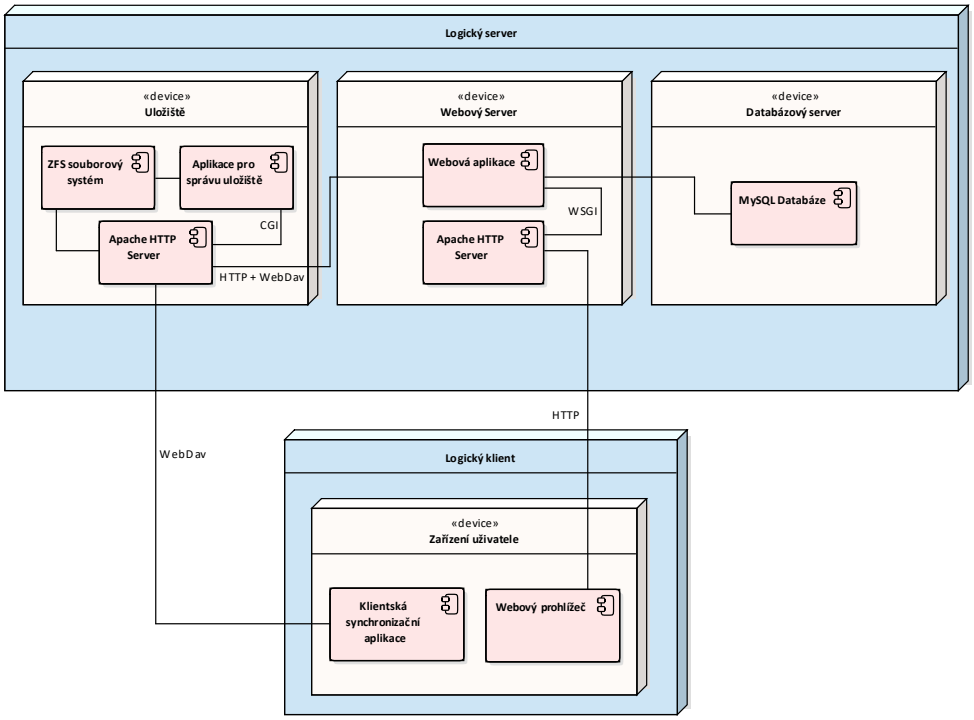
3.1 Komponenty

Výsledné řešení umožňující zabezpečení uživatelských dat bude obsahovat několik komponent. Každá komponenta bude plnit určitou předem definovanou funkcionalitu. Jednotlivé komponenty budou mezi sebou komunikovat určeným způsobem.

Komponenty řešení budou spadat do dvou hlavních části. Bude se jednat o serverovou a klientskou část. Hlavním cílem serverové části je uchovávání uživatelských souborů spolu s jejich historií. Serverová část bude také poskytovat webové rozhraní pro práci s těmito soubory. Ve webovém rozhraní bude také umožněna administrace uživatelských účtů, jejichž cílem je přiřazení dat jejich vlastníkům. Hlavním úkolem klientské části aplikace je odesílání dat na server a také jejich stahování z tohoto serveru. V návrhu řešení budou použity technologie popisované v kapitole číslo 2.5.

Seznam komponent (1., 2. a 3. serverová část, 4. klientská část):

1. Úložiště dat s aplikací pro jeho správu
2. Databáze
3. Webová aplikace
4. Synchronizační aplikace



Obrázek 3.1: Model nasazení

3.2 Úložiště dat

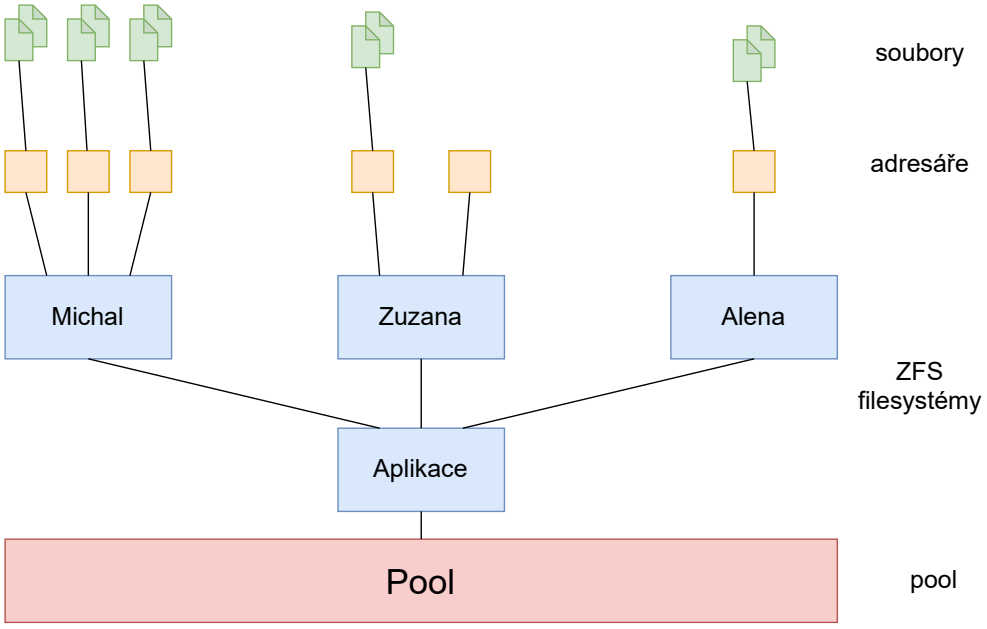
Hlavním účelem této serverové komponenty je uchovávání samotných souborů spolu s jejich historickými verzemi a poskytování nástroje pro správu tohoto úložiště. Při návrhu je počítáno s tím, že nástroj pro správu úložiště bude využíván pouze webovou aplikací, proto bude podporovat pouze funkcionalitu využívanou touto aplikací. O webové aplikaci pojednává kapitola číslo 3.3. Pokročilejší administrátorské operace, například přiřazení diskových zařízení do nového poolu, bude muset provést správce systému přímo v operačním systému pomocí nástrojů pro jeho správu.

Pro uložení souborů bude využit souborový systém ZFS, který má několik vlastností vhodných pro použití v tomto řešení. Jedná se o hierarchickou strukturu nebo deduplikaci. Tento souborový systém také disponuje nativní podporou snapshotů. ZFS dále umožňuje velice jednoduše zvýšit prostor pro ukládání dat a je také odolný proti výpadům disků v případě použití RAID-Z.

Historické verze souborů a celého úložiště budou uchovávány pomocí snapshotů. Uživatel bude mít možnost obnovení celého úložiště, adresářů i jednotlivých souborů. Pokud bude uživatel požadovat pouze obnovení některých souborů nebo adresářů bez smazání historických verzí, dojde k nahrazení zvo-

lených aktuálních dat těmi historickými ze snapshotu pomocí nástroje rsync. Pro přístup k souborům snapshotu bude využita skrytá složka *.zfs*. Další podporovanou funkcionalitou bude obnovení celého úložiště do původního stavu včetně odstranění všech historických verzí (snapshotů) až obnovovaného snapshotu. Tato operace bude provedena pomocí příkazu *rollback* na souborovém systému. Takto obnovená data budou muset být následně stažena zpět do klientského počítače pomocí aplikace popisované v kapitole číslo 3.5.

Každému uživateli bude po jeho registraci vytvořen vlastní ZFS souborový systém, do kterého budou ukládány jeho synchronizované adresáře a soubory. Uživatelské ZFS souborové systémy budou sdruženy do ZFS souborového systému určeného pro celou aplikaci. Tato datová struktura bude vytvořena nad poolem datových zařízení. Diagram zachycující strukturu úložiště se nachází na obrázku číslo 3.2. Mezi doporučená nastavení souborového systému se řadí zapnutí deduplikace, která snižuje nároky na datová úložiště. Deduplikace pracuje s datovými bloky a měla by být zapnuta pro ZFS souborový systém určený pro celou aplikaci, tudíž tato vlastnost bude fungovat napříč všemi uživateli. Komprese je další funkcionalitou ZFS, která bude použita pro snížení nároků na úložiště dat.



Obrázek 3.2: Zachycení struktury úložiště

Součástí této komponenty je aplikace sloužící pro administraci úložiště. Bude se jednat o nástroj, který bude hlavně volat příkazy nad souborovým systémem ZFS. Toto řešení umožní vytvoření univerzální webové aplikace, která bude nezávislá na způsobu uložení dat. Webová aplikace bude muset

pouze dodržet předepsaný způsob komunikace s nástrojem pro administraci úložiště, který může být napsán pro libovolný souborový systém. Podporovanými operacemi v administrátorské aplikaci je založení prostoru pro nového uživatele, odstranění uživatele spolu s jeho úložným prostorem, změna jeho hesla (tokenu), nastavení kvót pro uživatele, obnovení celého úložiště do verze z historie, obnovení adresáře a souboru do verze z historie, vytvoření nové verze úložiště, plánování úloh pro evidování historických verzí, zobrazení výpisu historických verzí úložiště a další. Aplikace bude také nabízet možnost nastavení souborového systému ZFS do navrženého doporučeného nastavení. Některé z těchto operací budou moci být provedeny pouze uživateli s oprávněními správce aplikace. Avšak tento nástroj nebude vyhodnocovat práva uživatelů sám, ale bude vycházet z hodnot zadaných v parametrech. Bude se jednat o uživatelské jméno a také o příznak určující, zdali je uživatel správcem aplikace. Tyto uživatelské údaje nebudou sloužit k autentizaci, protože se o ni bude starat nadřazená webová aplikace. Zadané uživatelské údaje budou sloužit čistě k potřebám zaznamenávání chyb.

V této kapitole popisovaná komponenta bude komunikovat s ostatními komponentami pomocí HTTP protokolu spolu s jeho nástavbou WebDav. Čisté HTTP spojení bude využito ke komunikaci mezi webovou aplikací a nástrojem pro správu úložiště, který nebude přijímat přímo HTTP požadavky, ale bude ovládán HTTP serverem pomocí protokolu CGI, který umožňuje propojení webového serveru s externí aplikací. Webová aplikace se bude na HTTP serveru autentizovat, aby nedocházelo k neoprávněnému přístupu. Tato autentizace bude probíhat pomocí HTTP Basic autentizace, která sice posílá hesla v otevřené podobě, ale při použití šifrovaného protokolu HTTPS budou již hesla chráněna. Použití HTTPS protokolu je nutné, protože se po neznámé síti budou přenášet také soubory a v dnešní době se již jedná o standard. Pomocí WebDav protokolu se budou přenášet soubory mezi úložištěm a klientskou aplikací i mezi úložištěm a webovou aplikací. Tento protokol je navržen pro použití, protože umožňuje mimo jiné uzamykání souborů. WebDav protokol je nástavba nad HTTP protokolem, proto bude fungovat i se striktně nastaveným firewallem. Pro autentizaci v protokolu WebDav je navrženo také použití HTTP Basic autentizace. Souborový systém uživatele bude vybrán webovým serverem na základě údajů zadaných při této autentizaci. Údaji potřebnými pro autentizaci jsou jméno a token, který je možné vygenerovat ve webové aplikaci. Pro autentizaci do úložiště uživatele tak není využito heslo pro přístup do webové aplikace.

V operačním systému úložiště nebude mít každý uživatel svůj účet, ale bude existovat pouze jeden uživatel, který bude využíván webovým serverem. Pro práci se souborovým systémem jsou požadována zvýšená oprávnění, ale není vhodné, aby nástroj pro správu úložiště běžel neustále se zvýšenými oprávněními. Zvýšená oprávnění pro práci se souborovým systémem budou poskytnuta pomocí příkazu sudo, který bude využit u příkazů vyžadující zvýšená oprávnění. Sudo je příkaz, který umožňuje spuštění jakéhokoliv příkazu

s oprávněními jiného uživatele, kterým je nejčastěji uživatel s administrátorskými oprávněními k systému. Tento příkaz bude nakonfigurován tak, aby nepožadoval heslo a bude umožňovat zvýšení práv pouze pro příkazy týkající se souborového systému ZFS a příkazu *chown*. Pro zajištění minimálních možných oprávnění nebude využíván příkaz sudo pro všechny operace pracující se souborovým systémem. ZFS umožňuje povolení některých operací pouze pro konkrétní ZFS souborový systém i jiným uživatelům, než je správce systému. Povolení těchto operací se provede pomocí příkazu *allow*. Tato funkcionalita bude použita při vytváření nových snapshotů, mazání těchto snapshotů a také pro nastavování kvót.

Nástroj pro správu úložiště bude muset zajistit, aby se některé operace v souborovém systému vykonávaly opakovaně. Jedná se konkrétně o operaci vytvoření nové verze úložiště (vytvoření nového snapshotu). Toto opakované spouštění bude vyřešeno pomocí nástroje Cron. O nastavení souboru crontab, který slouží ke konfiguraci nástroje Cron se bude opět starat aplikace pro správu úložiště na základě požadavku od webové aplikace.

3.3 Webová aplikace

Jedná se o aplikaci, která bude nabízet možnost administrace uživatelských účtů, práci se soubory a také s historickými verzemi úložišť a souborů. U některých typů souborů bude také umožněno jejich zobrazení. V této kapitole bude popsán návrh této aplikace včetně popisu zvolené architektury, která bude využita při implementaci aplikace.

3.3.1 Architektura

Při návrhu webové aplikace bude zvolena vícevrstvá architektura. Konkrétně se bude jednat o třívrstvou architekturu s využitím návrhového vzoru MVC.

3.3.1.1 Třívrstvá architektura

Jedná se o architekturu, při které je aplikace rozdělena do tří vrstev, které spolu komunikují pomocí definovaného rozhraní. Pro závislost vrstev by mělo platit, že vyšší vrstva je závislá na nižší vrstvě, ale nižší vrstva by neměla být nikdy závislá na vrstvě vyšší. Pokud je každá vrstva závislá pouze na jedné nižší vrstvě pak se jedná o striktní třívrstvou architekturu. V případě, že je závislá na dvou nižších vrstvách, tak se jedná o architekturu relaxovanou. Architektura definuje následující vrstvy (od nejvyšší po nejnižší):

- prezentační vrstva,
- doménová (business) vrstva,
- datová vrstva.

Prezentační vrstva se stará o zobrazování výstupu uživateli a o sběr jeho požadavků. Dále také umožňuje navigaci mezi stránkami. Prezentační vrstva obsahuje HTML stránky a třídy, které provádějí zpracovávání uživatelského požadavku. Doménová vrstva obsahuje veškerou logiku aplikace. Tato vrstva také provádí validaci uživatelského vstupu a také další výpočetní operace nad daty. Datová vrstva zajišťuje uložení dat na perzistentní médium a také načtení těchto dat z něj. [32]

Použití této architektury má řadu výhod oproti jiným architekturám. Díky nezávislosti komponent můžou vývojáři kteroukoliv komponentu vyměnit za jinou bez zásahu do celé aplikace. Dále tato architektura umožňuje použití některé komponenty v jiné aplikaci a také usnadňuje testování. Dalším přínosem třívrstvé architektury je možnost použití více prezentačních vrstev pro jednu aplikaci. [32]

3.3.1.2 MVC Architektura

Jedná se o architektonický vzor, který řeší problémy na prezentační vrstvě. Hlavním cílem této architektury je oddělení doménové logiky od prezentační vrstvy. Aplikace by měla být v rámci těchto dvou vrstev rozdělena ještě do dalších částí, kterými jsou Model, View a Controller. Komponenta Controller má nejdříve za cíl zpracování vstupu od uživatele. Vstup může být například ve formě HTTP požadavku. Takto zpracovaná data jsou následně poslána Modelu, který zodpovídá za doménovou logiku. Dále Controller požádá View, které má za cíl prezentaci dat uživateli, o vytvoření výstupu. Součástí požadavku ze strany Controlleru na View je také poskytnutí potřebných dat od Modelu. Model může být aktivní nebo pasivní. Aktivní Model dokáže upozorňovat View na změny, které nastaly. Pasivní Model pouze reaguje na požadavky. [32, 33]

MVC je většinou součástí používaných webových frameworků, kterými jsou například Nette pro PHP, Ruby On Rail pro Ruby. [32].

3.3.2 Framework

Framework je softwarová platforma, která slouží programátorovi jako podpora při vývoji aplikací. Cílem frameworků je vyřešení typických problémů a umožnění programátorovi řešit pouze problémy týkající se jeho konkrétní aplikace. Webové frameworky umožňují rychlý vývoj, řeší nízkoúrovňovou bezpečnost webů, nutí dodržovat model MVC a také prosazují moderní postupy při vývoji webů. Existuje velké množství frameworků pro mnoho programovacích jazyků. Jedná se například o Spring pro Javu, Django pro Python, AngularJS pro JavaScript nebo Nette Framework pro PHP. [34, 35]

3.3.3 Prezentační vrstva

Při návrhu této vrstvy je počítáno s tím, že budou využity HTML dokumenty, které budou ve formě šablony v závislosti na zvoleném frameworku. Je nutné využít HTML šablony, protože se některé části těchto dokumentů budou generovat automaticky. K HTML dokumentům bude přidán styl pomocí kaskádových stylů (CSS). Při procházení úložiště uživatelem bude nutné neustále překreslovat výpis složek a adresářů na webové stránce. Načítání celé stránky je při procházení úložiště nechtěné, proto je nutné zvolit technologii takovou, aby docházelo pouze k překreslení obsahu, který se mění. V tomto případě se bude jednat pouze o okno s výpisem souborů a adresářů. Pro vyřešení tohoto problému je navrženo zvolit technologii AJAX, která je postavená na JavaScriptu. Pro jednodušší práci s elementy na HTML stránce bude využita knihovna jQuery, která je rovněž postavena na JavaScriptu.

3.3.4 Doménová a datová vrstva

Webová aplikace bude muset přistupovat k uživatelským souborům uloženým v úložišti popsaném v kapitole číslo 3.2. Z tohoto důvodu bude nutné, aby datová vrstva uměla pracovat s protokolem WebDav. Mimo to bude muset být přistupováno k záznamům uloženým v relační databázi, ale tato funkcionalita je podporována běžně používanými webovými frameworky, které k těmto datům umožňují přístup pomocí objektově relačního mapování. Doménová vrstva se bude starat o kontrolu parametrů a také o různé výpočty.

Pro implementaci webové aplikace bude využit programovací jazyk Python, protože tento jazyk je autorovi této práce dobře znám. Jako webový framework bude využito Django, které je nad tímto programovacím jazykem postaveno.

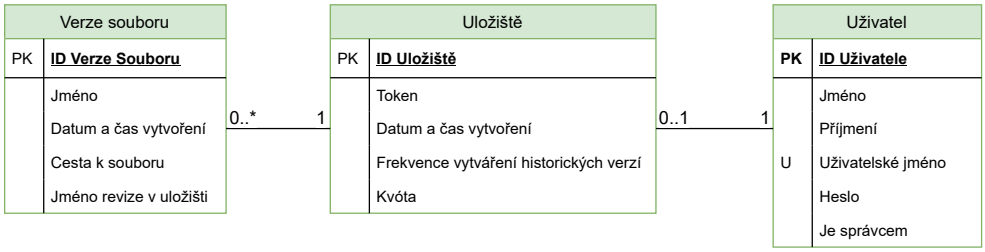
3.4 Databáze

Pro provoz webové aplikace musí být k dispozici databáze, která bude sloužit pro ukládání informací potřebných pro provoz této aplikace. Jedním z údajů nutných k evidování jsou informace o registrovaných uživatelích. Tyto údaje budou sloužit především pro autentizaci uživatelů. V této databázi bude také potřeba ukládat údaje související s vytvořenou pojmenovanou verzí souboru. S daty uloženými v této databázi bude pracovat pouze webová aplikace.

Při návrhu je počítáno s použitím relační databáze, protože se jedná o velmi rozšířený a osvědčený způsob ukládání dat. Konkrétně je navrženo použití databázového systém SQLite, který je velmi jednoduchý na použití. Správci aplikace si nicméně budou moci vybrat i jiný systém pro ukládání dat podporovaný webovou aplikací.

Databáze bude obsahovat tři tabulky. První bude evidovat informace o registrovaných uživatelích, další bude evidovat úložiště patřící uživatelům a po-

slední bude obsahovat informace o pojmenovaných verzích souborů. Detailní informace o těchto tabulkách včetně jednotlivých atributů a vazeb mezi nimi jsou uveden na obrázku číslo 3.3.



Obrázek 3.3: Model databáze

3.5 Synchronizační aplikace

Hlavním účelem této aplikace je synchronizace dat uložených na klientském počítači uživatele se serverem. Server představuje komponentu popsanou v kapitole číslo 3.2. Tato synchronizace má probíhat v předem určených intervalech, proto bude nutné nechat tento nástroj spuštěný neustále na pozadí. Mimo synchronizaci souborů musí tato aplikace dále nabízet možnost zobrazení stavu a změnu nastavení synchronizace.

Tato synchronizační aplikace bude rozdělena do dvou částí, které mohou být spuštěny v různých procesech. Cílem první části je samotná synchronizace souborů a adresářů. Cílem druhé části je poskytování grafického uživatelského rozhraní pro nastavení. Synchronizační část aplikace bude moci být spuštěna jako služba na pozadí a bude provádět nahrávání souborů na server a také jejich stahování z tohoto serveru. V rámci optimalizace budou nahrávány pouze soubory, u nichž došlo od minulé synchronizace ke změně. Při opačném procesu se budou stahovat všechny soubory. V rámci stahování a nahrávání souborů bude také docházet na straně cíle k odstraňování souborů, které se již nenachází na zdrojovém úložišti.

Klientská aplikace bude pracovat se dvěma soubory, které budou uloženy v domovském adresáři uživatele. Samotná synchronizace se bude řídit nastaveními, která se nacházejí v konfiguračním souboru. Tento soubor bude upravován částí poskytující grafické uživatelské rozhraní. Druhý soubor bude obsahovat informace o poslední synchronizaci a také o stavu synchronizace (probíhá synchronizace, čeká se na začátek synchronizace). Synchronizační část aplikace musí být informována o změnách v konfiguračním souboru. Toto je nutné, protože je důležité upravit čas spuštění další synchronizace v případě, že si uživatel zvolil kratší interval mezi synchronizacemi, než je aktuálně nastavený. Mimo to bude také nutné informovat synchronizační službu

o tom, že má být provedena synchronizace mimo obvyklý čas. Původně bylo navrhováno zajištění této komunikace pomocí zasílání signálů. Konkrétně by část aplikace umožňující nastavení synchronizace zasílala signály SIGUSR1, SIGUSR2 synchronizační službě. Toto řešení nebylo nakonec zvoleno, protože některé operační systémy nepodporují potřebné signály.

Nakonec bylo zvoleno takové řešení, kdy si synchronizační služba sama monitoruje změny v konfiguračním souboru a v případě potřeby si upraví začátek další synchronizace. Samotné monitorování změn musí být zajištěno tak, aby nedocházelo k neustálým kontrolám a tím nebyl zatěžován systém. Tento problém bude vyřešen pomocí podpory ze strany operačního systému. O extra synchronizaci mimo stanovený čas bude synchronizační služba informována stejným způsobem. S monitorováním souborového systému souvisí i další funkcionalita, která by umožnila provádění synchronizace po změně jakéhokoliv souboru. Autor této práce se nakonec rozhodl neimplementovat tuto funkcionalitu, ale určitě se jedná o námět k dalšímu vylepšení.

3.6 Bezpečnost a efektivita aplikace

Tato kapitola se věnuje bezpečnosti aplikace a její efektivitě. Bezpečnost je zde rozebrána, jak z pohledu uložení dat na serveru, tak z pohledu jejich přenosu mezi klientem a serverem.

Uživatelské soubory jsou chráněny před smazáním nebo před poškozením tím, že jsou v pravidelných intervalech odesílány na server a zde jsou opět v pravidelných intervalech vytvářeny jejich historické verze pomocí snapshotů. Snapshoty jsou pouze ke čtení a tudíž nemohou být změněny. Změněny nebo odstraněny mohou být pouze poslední verze dat na serveru po provedené synchronizaci, čímž nevzniká velká škoda při časté frekvenci vytváření nových snapshotů, protože celé úložiště může být pomocí těchto snapshotů obnoveno. Riziko ztráty dat může představovat odstranění snapshotů. Tento problém je vyřešen tak, že snapshoty mohou být odstraňovány pouze nástrojem pro správu úložiště, který je popsán v kapitole číslo 3.2. S tímto nástrojem může pracovat pouze webová aplikace, proto není možné odstraňovat snapshoty přes přístup pomocí WebDav protokolu. Exkluzivní přístup webové aplikace, k nástroji pro správu úložiště, je zajištěn pomocí autentizace heslem a pomocí kontroly IP adresy, čímž může být zamezen přístup k tomuto nástroji z vnější sítě. Webová aplikace obsahuje řešení problému bezpečnosti a každý uživatel si může po úspěšné autentizaci smazat pouze své soubory. Pomocí WebDav protokolu mohou uživatelé přistupovat také pouze ke svým datům. Uživatelé se autentizují pro použití WebDav protokolu pomocí HTTP Basic autentizace a stanovení jejich úložiště je provedeno HTTP serverem pomocí přepisovacích pravidel adresy URL. Toto přepisování je inspirováno ukázkou nastavení webové serveru Apache v článku [36]. Komunikace mezi všemi komponentami je také zabezpečená díky použití protokolu HTTPS.

Minimální nároky na úložný prostor jsou zajištěny použitím souborového systému ZFS se zapnutou kompresí a deduplikací. Snapshoty uchovávající historické verze uživatelských dat jsou také šetrné k úložnému prostoru. Během odesílání dat na server jsou odesílány pouze chybějící nebo pozměněné soubory, čímž se šetří objem přenesených dat.

Kapitola 4

Implementace

Tato kapitola se věnuje realizaci samotného systému a jsou zde také popsány vzniklé problémy a jejich řešení. Implementace vychází z návrhu, který je uveden v kapitole číslo 3. Text popisující implementaci je rozdělen do kapitol podle komponent řešení.

4.1 Nástroj pro správu úložiště

Pro implementaci této části řešení je využít programovací jazyk Python, který disponuje velkým množstvím externích knihoven. Tento jazyk také umožňuje vytvářet multiplatformní aplikace. Navzdory této vlastnosti je nástroj pro správu úložiště plně funkční pouze v linuxových operačních systémech kvůli některým specifikám popsaným níže. Jako vývojové prostředí je využit nástroj PyCharm vyvíjený společností JetBrains.

Nástroj pro správu úložiště bude využíván pouze webovou aplikací, proto musí tento nástroj podporovat pouze funkcionalitu, kterou webová aplikace využije. Daná funkcionalita je aplikaci poskytována skrze rozhraní protokol HTTP. Argumenty jsou tomuto nástroji předávány v textovém formátu JSON. Příklad uživatelského vstupu ve formě slovníku je zobrazen ve výpisu 4.1. Ovládání pomocí CLI tato aplikace nepodporuje.

Konfigurace tohoto nástroje je možná pomocí konfiguračního souboru *configuration.conf*. Tento soubor obsahuje umístění úložiště souborů, nastavení tohoto úložiště, ale také cesty k nástrojům nainstalovaných v operačním systému, které jsou pro provoz nástroje pro správu úložiště nezbytné. Jedná se o aplikaci *zfs*, *file*, *rsync* a *chown*. Konfigurace nástroje je reprezentována třídou **Configuration**. Pokud aplikace nemůže tento konfigurační soubor nalézt, jsou použity doporučené výchozí hodnoty.

Funkcionalita nástroje pro správu uložiště je rozdělena do několika modulů, z nichž každý plní svou funkci. Některé významné třídy z těchto modulů jsou popsány v následujících podkapitolách.

4.1.1 OperationSelector

Účelem této třídy je zpracování uživatelského vstupu, jehož příklad je uveden na 4.1. Na základě tohoto vstupu je zavolána metoda rozhraní *FilesystemManagementAPI*, která slouží k provedení příslušné operace na souborovém systému. Dále zde probíhá kontrola výsledků operací a případné zaznamenávání chyb. Mezi povinné argumenty se řadí identifikační údaje uživatele,

```
arguments = {
    "executor": "michal",
    "admin": "True",
    "operation": "remove_user",
    "username": "user11"
}
```

Výpis kódu 4.1: Příklad uživatelského vstupu

který danou operaci spouští. Jedná se o jeho uživatelské jméno a příznak určující, zdali se jedná o správce. Oba tyto údaje jsou nastavovány webovou aplikací a nejsou již ověřovány heslem. Uživatelské jméno slouží hlavně k účelům zaznamenávání chyb a příznak je kontrolován před vykonáním operace, která vyžaduje zvýšená oprávnění. Kontrola oprávnění probíhá tak, že metoda vybraná na základě uživatelského vstupu sloužící pro volání operací na souborovém systému prostřednictvím rozhraní *FilesystemManagementAPI* je dekorována pomocí dekorátoru *admin_required*. Samotný dekorátor se nachází na 4.2.

```
def admin_required(func):
    def checked(*args, **kwargs):
        if not args[0].admin:
            args[0].logger.error("permission denied")
            return {"result": ErrorCode.PERMISSION_ERROR}
        return func(*args, **kwargs)
    return checked
```

Výpis kódu 4.2: Dekorátor sloužící pro kontrolu oprávnění

4.1.2 FilesystemManagementApi

Jedná se o abstraktní třídu, která představuje rozhraní určující způsob komunikace se souborovým systémem. Díky tomuto rozhraní je možné jednoduše vyměnit souborový systém, aniž by se musel složitě přepisovat kód celého nástroje pro správu úložiště.

4.1.3 ZFSManagement

Tato třída rozšiřuje abstraktní třídu *Filesystem_management_api* a jedná se tak o implementaci umožňující práci s konkrétním souborovým systémem. Konkrétně reprezentuje souborový systém ZFS.

Metody pracují se souborovým systémem tak, že v operačním systému spouští externí příkazy pro spravování souborového systému a sledují jejich návratové kódy. Spouštění externích příkazů v systému musí být bezpečné, aby uživatel nemohl provést Command Injection. Jedná se o útok, kdy uživatel zadá místo validního řetězce (například jména) příkaz, kterému rozumí shell. Takovýto příkaz může vypadat následovně: `michal; rm -rf *`. V tomto případě by mohlo dojít ke smazávaní všech souborů v aktuálním adresáři aplikace. Tento problém je vyřešen pomocí Python objektu Popen z modulu subprocess, který nepřijímá jednotlivé parametry příkazu oddělené mezerou, ale jako položky v listu. U tohoto objektu je dále nastaveno, aby nebyly interpretovány znaky, kterým rozumí shell. Samotný příkaz se provede pomocí metody *communicate*. Některé z příkazů konfigurující ZFS jsou na výpisu kódu 4.3 uvedeny tak, jak by byly zadávány do shellu.

```
# vytvoření nového úložiště
sudo zfs create pool/uloziste/michal
# aktuálně využité místo na úložišti
sudo zfs list -Hp -o used pool/uloziste/michal
# obnovení celého úložiště
sudo zfs rollback -r pool/uloziste/michal@today
```

Výpis kódu 4.3: Příkazy ukazující konfiguraci souborového systému ZFS

Dále je nutné pracovat se souborem obsahující tokeny pro HTTP Basic autentizaci. Tento soubor je využíván WebDav serverem při kontrole přístupů uživatelů ke svým datům. S tímto souborem pracuje nástroje pro správu uložiště pomocí generátoru hesel htpasswd.

Úložiště musí být schopno automaticky vytvářet historické verze uživatelských úložišť. Pro tyto účely je využit nástroj operačního systému Cron, který je nakonfigurován v souboru crontab. Jedná se o příkaz, který je schopen v zadaných časech spouštět různé procesy. Pro plánování a rušení naplánovaných operací v souboru crontab je využit externí modul python-crontab [37].

4.2 Webová aplikace

Pro vývoj webové aplikace je v souladu s návrhem použit programovací jazyk Python spolu s webovým frameworkem Django. Samotná aplikace respektuje model MVC. Popis implementace webové aplikace je rozdělen do dvou hlavních kapitol, které budou dále obsahovat další podkapitoly.

4.2.1 Prezentační vrstva

Tato vrstva webové aplikace se skládá ze tří hlavních částí, kterými jsou Views, HTML soubory a skripty jazyka JavaScript.

4.2.1.1 Views

Webový framework Django plně implementuje MVC architekturu, ale názvy jednotlivých částí jsou oproti standardu odlišné. Komponenta View je v Djangu nazvána jako Template a Controller je v Djangu nazván jako View. Název pro modely zůstává nezměněn. Protože pro implementaci práce je použit framework Django, bude i zde v textu použita konvence právě tohoto frameworku.

Pro každou stránku webové aplikace existuje jedno View, které je reprezentováno Python třídou. Tyto třídy jsou potomci Django tříd *View* a *ListView*. Všechny potřebné View komponenty se nacházejí v balíčku views_pkg. V tomto balíčku se dále nachází třída *FileManagementView*, která se stará o obsluhování asynchronní komunikace Ajax. Tato komunikace probíhá pomocí textového formátu JSON. Pokud dojde během zpracování uživatelského požadavku v některém View k chybě a nebo je potřeba sdělil uživateli nějakou zprávu, tak je využit Django messages framework.

4.2.1.2 Uživatelské rozhraní

Tato část aplikace se stará o generování výsledné stránky pro uživatele. Realizace je provedena pomocí HTML šablon, které přijímají data od pohledů (Views). Pro vytvoření interaktivní webové aplikace jsou přidány skripty v jazyce JavaScript. V těchto skriptech jsou dále využity externí knihovny. Jedná se například o již zmíněné jQuery pro snadné vybírání HTML elementů a také o Ajax pro asynchronní komunikaci se serverem využitou při listování v uživatelských adresářích a pro práci s nimi.

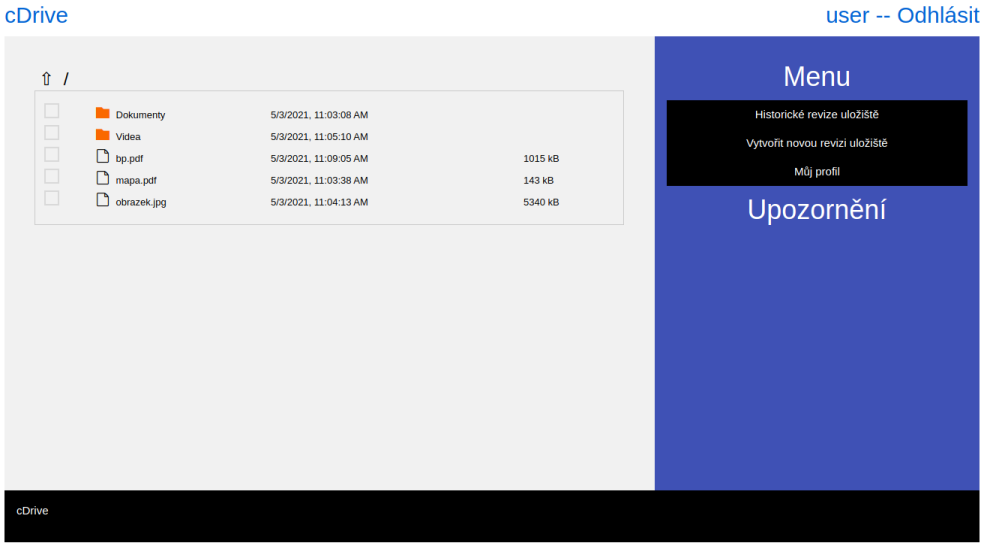
Soubor *file_list.js* obsahuje funkce, které slouží pro práci se seznamem uživatelských souborů. Tento seznam je realizován pomocí frameworku Metro 4 [38] postaveném na jazyce JavaScript, konkrétně se jedná o komponentu ListView. Tato komponenta umožňuje vytvoření naformátovaného seznamu souborů a také zjednodušuje detekci základních událostí vyvolaných uživatelem. Funkce v souboru *file_list.js* volají metody objektu ListView frameworku Metro 4 s vhodně nastavenými parametry pro použití v aplikaci. Komponenta ListView reprezentující seznam souborů je zachycena na části obrázku 4.1.

List View frameworku Metro 4 nepodporuje kontextové menu, které je potřeba pro rozšířenou práci se soubory (odstranění souboru, vytvoření pojmenované verze, přejmenování, ...). Z tohoto důvodu je použita knihovna jQuery contextMenu [39], která umožňuje vytvoření tohoto menu. Kontextové menu je definováno v souboru *file_manager.js*. V tomto souboru se také nacházejí funkce, které jsou zavolány po kliknutí na položku kontextového

menu, ale také i po dvojitém kliknutí na soubor. Tyto funkce realizují následující operace – otevření souboru, jeho přejmenování, odstranění, zkopírování, vyjmutí a vložení. Mimo to také umožňují vytvoření nové pojmenované verze souboru a také zobrazení pojmenovaných verzí daného souboru. Soubor *revisions_manager.js* je velmi podobný souboru *file_manager.js*, ale obsahuje funkce, které pracují s historickými revizemi.

Na některých stránkách je potřeba zobrazit list položek (seznam uživatelů, seznam historických verzí), který může být v některých případech dlouhý. Z tohoto důvodu je nutné použít stránkování. Stránkování je realizováno pomocí vestavěného Django stránkovače ve třídě ListView. ListView je rodičovskou třídou view použitého pro realizaci stánky obsahující daný list položek. Kromě podpory ze strany view je také nutné přidat odkazy do HTML šablon, které zajistí navigování mezi stránkami s obsahem. Tyto odkazy byly do řešení přejaty z dokumentace k frameworku Django [40].

Samotné HTML soubory jsou nastylovány pomocí kaskádových stylů. Pro tyto účely je využita šablona uvedená na [41]. Obsah takto nastylovaných HTML souborů je inspirován příkladem uvedeným na [41]. Dále je nutné přidat styl ke zprávám, které jsou generovány pomocí Django messages framework. Tento styl je zajištěn pomocí nástroje Bootstrap [42].



Obrázek 4.1: Zachycení obrazovky webové aplikace

4.2.2 Doménová a Datová vrstva

Tato část aplikace se skládá z Python modulů, které se starají o práci s databází a také provádějí různé výpočty. Tyto moduly také poskytují funkcionalitu umožňující ovládání vzdáleného úložiště a získávání uživatelských dat z tohoto úložiště.

4.2.2.1 Databázové modely

S databází pracuje webová aplikace pomocí Objektově-Relačního Mapování (ORM), které je poskytováno frameworkem Django. Tento způsob zjednodušuje práci s databází, umožňuje vytvoření aplikace nezávislé na způsobu uložení dat a také zabraňuje napadení databáze pomocí SQL injection. Pro každou tabulku v databázi existuje třída, která ji reprezentuje a umožňuje práci s ní.

4.2.2.2 Moduly umožňující práci s úložištěm

Webová aplikace potřebuje pracovat s daty uloženými na vzdáleném serveru, který umožňuje přístup k datům pomocí protokolu WebDav. Samotné datové úložiště reprezentuje třída *DataStorage*, která poskytuje metody umožňující nahrávání, respektive stahování souborů a adresářů. Metody v této třídě využívají externí knihovnu webdavclient3 [43], která umožňuje práci se soubory na serveru s podporou protokolu WebDav.

Mezi významné třídy se také řadí třída *StorageManager*, která reprezentuje nástroj pro správu úložiště. Jedná se o nástroj popisovaný v kapitole číslo 4.1. Tato třída obsahuje metody pokrývající celou funkčnost tohoto nástroje. Komunikace je realizována pomocí HTTP protokolu, který přenáší data v textovém formátu JSON.

Pro stažení aktuální nebo historické verze souboru pomocí webové aplikace je potřeba, aby měl HTTP server webové aplikace tento soubor k dispozici. Tento soubor se ale nachází na WebDav úložišti a je nepřípustné stahované soubory před jejich stažením ukládat na webový server. Tento problém je vyřešen použitím Django třídy StreamingHttpResponse. Jedná se o třídu reprezentující HTTP odpověď, která nepotřebuje mít všechna data potřebná k této odpovědi uložená, ale stačí jí Python generátor dat. S generátorem souvisí další problém a to ten, že používaný modul webdavclient3 neobsahuje metodu, která vrátí generátor dat s obsahem souboru. Použitý WebDav klient obsahuje pouze metodu, která stáhne soubor do úložiště. Tento problém je vyřešen vytvořením nové metody pro stahování obsahu, která tento problém řeší. Tato metoda využívá jiné metody modulu webdavclient3 než je její hlavní metoda pro stahování. Pokud si uživatel přeje stažení adresáře nebo dvou a více souborů, je vytvořen archiv pomocí externího modulu zipstreamnew [44], který umí vytvořit proud dat obsahující zazipovaný obsah.

Pro otevírání souborů je nutné znát jejich typ. Typ souboru je aplikací určen nejdříve pomocí jeho koncovky. Pokud název souboru koncovku neobsahuje nebo je pro webovou aplikaci koncovka neznámá, tak aplikace zkusí určit typ souboru heuristicky pomocí nástroje *file*, který je volán nástrojem pro správu úložiště. Identifikace typu zajišťují metody objektu *FileIdentifier*.

4.3 Klientská synchronizační aplikace

Pro vývoj této části systému je opět využit programovací jazyk Python, protože jeho použitím je vytvořen celý systém postavený na jednom jazyku, což je autorem této práce považováno jako výhoda pro uživatele. Mimo to mohou být i v této komponentě využity některé pro autora již známé knihovny, které byly použity již v předchozích komponentách.

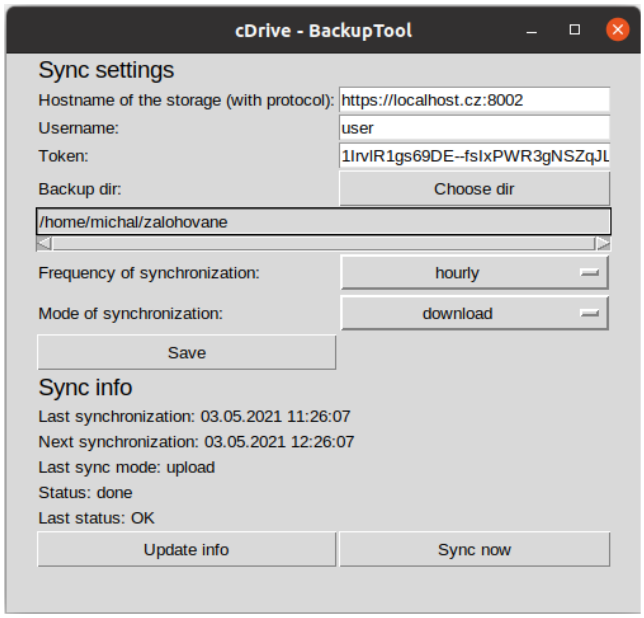
4.3.1 Grafické uživatelské rozhraní

Pro vytvoření grafického uživatelského prostředí je využit modul Tkinter. Definice jednotlivých elementů grafického rozhraní je umístěna ve třídě *GUI*. V této třídě jsou také umístěny metody obsluhující události vykonané uživatelem. Tato třída dále nepřímo pracuje s daty uloženými v konfiguračním soubor, který definuje nastavení synchronizace. Tento konfigurační soubor je reprezentován třídou *Configuration*. V uživatelském rozhraní jsou také zobrazovány informace o poslední a následující synchronizaci. Tyto informace jsou získávány ze souboru, který je upravován synchronizační službou. Konfigurační soubor a soubor s informacemi o synchronizaci je ukládán do adresáře uživatele. V každém operačním systému se tento adresář nachází v jiném umístění. Z tohoto důvodu je využit externí modul config-path [45], který řeší právě tento problém. Obrazovka zachycující nástroj uživatelské rozhraní pro nastavení synchronizace se nachází na obrázku 4.2.

4.3.2 Služba zajišťující synchronizaci dat

Tato synchronizační služba bude standardně od grafického prostředí oddělena, protože se bude jednat o samostatný proces. V případě, že nebude tato synchronizační služba z jakéhokoliv důvodu spuštěna, bude mít uživatel k dispozici možnost provedení jednorázové synchronizace přímo procesem poskytující nastavení aplikace. Dostupnost synchronizační služby bude ověřována pomocí jména a čísla běžícího procesu. Číslo procesu (PID) bude ukládáno do souboru obsahující informace o synchronizaci.

Pro synchronizaci dat se serverem (stahování a nahrávání) je využit stejný modul [43], který je využit při implementaci webové aplikace. Tento modul umožňuje nahrávat data na WebDav server a také je z něj stahovat. Dále tento modul nabízí funkcionalitu umožňující chytrou synchronizaci souborů. Daná funkcionalita nemůže být v této práci využita, protože sice umožňuje



Obrázek 4.2: Zachycení obrazovky nástroje pro nastavení synchronizace

odesílání pouze nových nebo změněných souborů, ale nepodporuje odstranění souborů z cílového zařízení. Z tohoto důvodu musela být provedena implementace nové metody pro synchronizaci využívající funkcionalitu modulu webdavclient3 [43]. Tyto nové metody jsou umístěny ve třídě *DataStorage*, která reprezentuje vzdálené úložiště. Při psaní této nové funkcionality se autor práce inspiroval původními metodami modulu webdavclient3. Konkrétně se jedná o *push* a *pull*.

Synchronizace bude probíhat v uživatelem definovaných časových intervalech, proto bude muset synchronizační aplikace běžet neustále na pozadí. Činnost synchronizace bude vykonávat jedno vlákno reprezentované třídou *Syncer*. Pokud nenastane čas synchronizace, bude toto vlákno uspané až do času další synchronizace. V případě potřeby bude moci být toto vlákno předčasně probuzeno. Jedná se o situaci, kdy se změní konfigurační soubor nebo je vyžadována okamžitá synchronizace. Pro potřebu uspávání a probouzení vláken je využita třída Event, která je součástí standardní knihovny jazyku Python. Jak je již popsáno v návrhu, synchronizační služba bude muset monitorovat souborový systém. Tento monitoring je zajištěn pomocí externího modulu Watchdog [46]. Na události, které nastaly v souborovém systému bude reagovat obsluha událostí, která je reprezentovaná třídou *FileChangeHandler*.

Kapitola 5

Možná vylepšení

Výsledné řešení splňuje všechny definované funkční i nefunkční požadavky. Mezi eventuální vylepšení se řadí přidání nových funkcionalit a také zlepšení stability aplikace při práci více uživatelů na serveru. Možná vylepšení funkcionality jsou následují:

- Synchronizace souborů po každé jejich změně.
- Oboustranná synchronizace dat s podporou slučování.
- Rozšíření počtu podporovaných typů souborů pro zobrazení.
- Umožnění úpravy souborů přímo na serverové části.
- Podpora sdílení souborů na serveru mezi uživateli a umožnění společné práce na nich.
- Rozšíření funkcionality klientské aplikace – umožnění vytvoření nové verze souboru bez použití webové aplikace, detailnější informace o synchronizování.

Kapitola 6

Návod k instalaci a použití

Tato kapitola obsahuje instrukce k instalaci celého systému a také návod na použití. Pro jednodušší instalaci je možné využít instalační skripty, které nainstalují nezbytný software a provedou konfiguraci. Tyto instalační skripty jsou funkční pro operační systémy založené na Debian GNU/Linux. Skripty jsou otestovány na operačním systému Ubuntu 20.04 (Focal Fossa). Celý systém pro zálohování dat je rozdělen na serverovou a klientskou část. Serverová část může být dále rozdělena do dvou dalších komponent. Pro plnou funkčnost je zapotřebí nainstalovat všechny části. Návod na jejich instalaci bude uveden v následujících kapitolách. Samotná instalace jednotlivých komponent by měla probíhat ve stejném pořadí, v jakém je zde uvedena.

6.1 Úložiště - serverová část

V této kapitole bude popsána potřebná konfigurace serverového úložiště a bude zde také uveden postup na instalaci nástroje pro jeho správu.

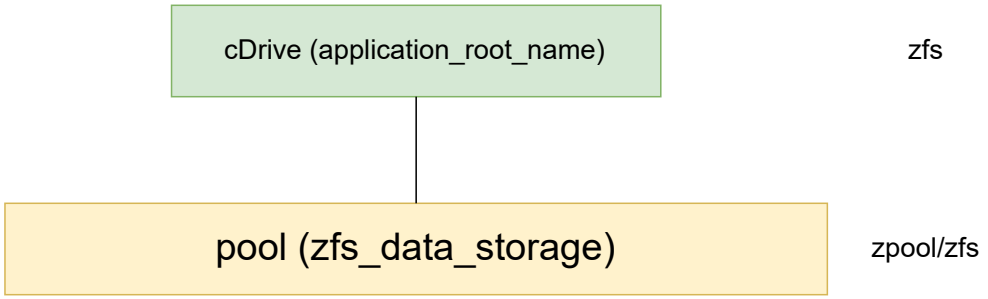
1. Požadavky

Pro úložiště je nutné využít zařízení, na kterém je nainstalován operační systém s podporou souborového systému ZFS. K tomuto systému musí být připojeno zařízení pro ukládání dat, kde musí být využit souborový systém ZFS. V tomto operačním systému je dále zapotřebí mít nainstalovaný interpret programovacího jazyku Python ve verzi 3.8 nebo vyšší.

2. Instalace

K úložišti je přistupováno pomocí protokolu WebDav, proto je nutné nainstalovat webový server a nakonfigurovat ho pro použití tohoto protokolu. Dále je nutné nastavit HTTP Basic autentizaci pro přístup k datům uživatele na

základě zadaných přihlašovacích údajů. Tento webový server musí přistupovat k úložišti se souborovým systémem ZFS. Souborový systém musí v sobě obsahovat další ZFS souborový systém pro aplikaci. Doporučená struktura je uvedena na obrázku 6.1. Údaje v závorkách na tomto obrázku značí jméno atributu v konfiguračním souboru nástroje pro správu úložiště, který je popsán níže v této kapitole.



Obrázek 6.1: Vstupní struktura ZFS

Po nastavení úložiště je nutné nainstalovat nástroj pro jeho správu a nakonfigurovat ho pomocí konfiguračního souboru, který v základu obsahuje doporučená nastavení. Webová aplikace komunikuje s tímto nástrojem pomocí HTTP serveru. Propojení mezi HTTP serverem a samotnou aplikací musí být provedeno pomoc CGI rozhraní. HTTP server umožňující přístup k tomuto nástroji by také měl být nakonfigurován tak, aby umožnil přístup pouze z jedné IP adresy (adresa serveru webové aplikace). Přístup by také měl být podmiňován heslem. Pro provoz tohoto nástroj je nutné vytvořit nové virtuální prostředí *venv* a nainstalovat potřebné moduly uvedené v souboru *requirements.txt*. Interpret tohoto prostředí je následně nutné přidat na začátek skriptu *___main___.py*, který je spouštěn webovým serverem.

Je důrazně doporučeno používat v obou případech pouze zabezpečenou komunikaci HTTPS. Pro instalaci je doporučené využít připravený instalační skript, který provede všechny potřebné výše popsané kroky. Tento instalační skript musí být spouštěn s oprávněními správce systému.

6.2 Webové aplikace - serverová část

1. Požadavky

Požadavkem před instalací webové aplikace je nutnost mít nainstalovaný interpret jazyka Python ve verzi 3.8 nebo vyšší.

2. Instalace

Pro provozování webové aplikace je potřeba nainstalovat webový server s podporou WSGI skriptů. K tomu je nutné vytvořit nové virtuální prostředí *venv* a nainstalovat do něj potřebné moduly obsažené v souboru *requirements.txt*. Webová aplikace se konfiguruje pomocí souboru *settings.py*, kde je mimo jiné nutné nastavit i údaje o relační databázi. Potřebné tabulky do této databáze jsou vytvořeny pomocí *manage.py* skriptu s argumentem `migrate`. Tento skript musí být spouštěn Python interpretem v nově vytvořeném prostředí. Tento samý skript s argumentem `createsuperuser` musí být použit pro vytvoření prvního super-uživatele, který nevlastní úložiště. Pro kompletní instalaci této aplikace je ovšem doporučeno využít instalační skript, který opět provede všechny výše popsané kroky. Tento instalační skript je opět nutné spouštět s oprávněními správce systému.

6.3 Zálohovací nástroj - klientská část

1. Požadavky

Požadavkem před instalací webové aplikace je nutnost mít nainstalovaný interpret jazyka Python ve verzi 3.8 nebo vyšší.

2. Instalace

Pro instalaci je nutné vytvořit nové virtuální prostředí *venv* a nainstalovat do něj potřebné moduly uvedené v souboru *requirements.txt*. Po instalaci prostředí je možné spustit skript *backup_tool_gui.py*, který poskytuje grafické prostředí k nastavení synchronizace. Pokud uživatel požaduje, aby byla synchronizace prováděna i bez interakce uživatele, je nutné nastavit skript *backup_tool_service.py* jako službu operačního systému.

Všechny výše popsané kroky je opět možné provést pomocí skriptu, který je určen pro distribuce založené na Debian GNU/Linux s podporou systemd. Tento instalační skript **je nutné** spouštět pod identitou uživatele, který ho spouští. **Není** tedy možné spouštět tento skript pomocí příkazu sudo.

6.4 Návod

Jako návod k použití je možné využít případy užití uvedené v kapitole 2.2.

Závěr

Cílem této práce bylo navrhnout a implementovat systém, který pomůže uživatelům se zabezpečením jejich dat. Řešení mělo být typu klient-server, kdy by se uživatelská data přenášela z klientské části na serverovou část a v případě potřeby i opačným směrem. Serverová část měla také poskytovat uživatelské rozhraní, které by mělo umožňovat zobrazení některých typů souborů. Celý systém také měl minimalizovat dopady nákazy ransomware.

Byl proveden návrh systému zabezpečující uživatelská data synchronizací se serverem. Na základě tohoto návrhu byla následně provedena implementace. Výsledná aplikace disponuje všemi vytyčenými funkčními i nefunkčními požadavky. Před návrhem a implementací byla provedena analýza a zhodnocení některých dalších nástrojů, které by tyto požadavky mohly splňovat. Mezi těmito nástroji se nacházely OneDrive, Google Disk, Duplicati, Nextcloud a další. Na konci této kapitoly bylo konstatováno, že žádné z uvedených řešení plně neodpovídá definovaným požadavkům. Z tohoto důvodu se autor této práce rozhodl pro vytvoření vlastního řešení. V kapitole analýza je dále uveden popis některých technologií vhodných pro implementaci výsledného řešení.

V další kapitole je uveden návrh systému pro zabezpečení uživatelských dat. Celý systém byl rozdělen do dvou hlavních částí, kterými jsou serverová a klientská část. Klientská část má za cíl synchronizaci dat se serverovou částí, jejímž cílem je uchování zálohovaných souborů spolu s jejich historií. Mimo to bude tato část poskytovat webové rozhraní pro práci se soubory a historickými verzemi těchto souborů.

Serverová část se skládá ze dvou komponent, kterými jsou webová aplikace a samotné úložiště dat. Pro ukládání dat byl použit souborový systém ZFS, který disponuje vlastnostmi vhodnými pro použití v této aplikaci. Jedná se o deduplikaci a nativní podporu snapshotů. Toto úložiště bude dostupné ostatním službám (klientská aplikaci, webová aplikace) prostřednictvím protokol HTTP, konkrétně se jedná o jeho rozšíření WebDav. Při návrhu webové aplikace je využita třívrstvá architektura s využitím návrhového vzoru MVC.

Pro implementaci všech částí systému se autor rozhodl využít programovací jazyk Python. Pro tvorbu webové aplikace byl využit webový framework Django, který je postaven na jazyku Python. Nakonec je uveden návod k použití a instalaci. Celý systém pro zálohování uživatelských dat je možno vylepšit, jak z pohledu funkcionality, tak z pohledu optimalizace.

Bibliografie

1. In: *Google Disk* [online]. Google, 2021 [cit. 2021-04-03]. Dostupné z: https://drive.google.com/.
2. ZÍTKO, Jan. GOOGLE APPS A BEZPEČNOST. In: *blog GAPPs* [online]. Sievert Consulting, 2014 [cit. 2020-12-28]. Dostupné z: https://google-apps.cz/google_apps_bezpecnost/.
3. In: *OneDrive* [online]. Microsoft Corporation, 2021 [cit. 2021-04-03]. Dostupné z: https://onedrive.live.com/.
4. In: *NextCloud* [online]. NextCloud GmbH, 2021 [cit. 2021-04-03]. Dostupné z: https://nextcloud.com/.
5. FELDMAN, David. Battle of the Clouds. In: *Nextcloud vs ownCloud – The Whole Story* [online]. CiviHosting, 2020 [cit. 2020-12-27]. Dostupné z: https://civihosting.com/blog/nextcloud-vs-owncloud/.
6. How Nextcloud helps protect against ransomware. In: *NextCloud* [online]. NextCloud GmbH, 2020 [cit. 2020-12-28]. Dostupné z: https://nextcloud.com/blog/how-nextcloud-helps-protect-against-ransomware/.
7. JANÍK, David. Proč a jak nainstalovat Nextcloud? In: *Váš Hosting* [online]. Váš Hosting, 2018 [cit. 2020-12-28]. Dostupné z: https://www.vas-hosting.cz/blog-proc-a-jak-nainstalovat-nextcloud#koliknextcloudstoji.
8. Best Alternatives to OwnCloud in 2020. In: *FileCloud Blog* [online]. CodeLathe Technologies Inc, 2020 [cit. 2020-12-27]. Dostupné z: https://www.getfilecloud.com/blog/2020/01/best-alternatives-to-owncloud-2020/.
9. HOLVE, Michael. Pokročilé zálohování s Rsync. In: *ROOT.CZ* [online]. Internet Info, 2007 [cit. 2021-01-01]. Dostupné z: https://www.root.cz/clanky/pokrocile-zalohovani-s-rsync/.

10. In: *Duplicati 2 User's Manual* [online]. The Duplicati Team, 2020 [cit. 2021-01-01]. Dostupné z: https://duplicati.readthedocs.io/en/latest/01-introduction/.

11. ZAVARSKY, Pavol; LINDSKOG, Dale et al. Experimental analysis of ransomware on windows and android platforms: Evolution and characterization. *Procedia Computer Science*. 2016, roč. 94, s. 465–472. Dostupné z DOI: 10.1016/j.procs.2016.08.072.

12. BREWER, Ross. Ransomware attacks: detection, prevention and cure. *Network Security*. 2016, roč. 2016, č. 9, s. 5–9. Dostupné z DOI: 10.1016/S1353-4858(16)30086-1.

13. CLEMENT, J. Number of ransomware attacks per year 2019. In: *Statista* [online]. 2021 [cit. 2021-04-04]. Dostupné z: https://statista.com/statistics/494947/ransomware-attacks-per-year-worldwide/.

14. An overview of HTTP. In: *MDN Web Docs* [online]. MDN contributors, 2021 [cit. 2021-01-22]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview.

15. Evolution of HTTP. In: *MDN Web Docs* [online]. MDN contributors, 2021 [cit. 2021-01-22]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP.

16. DUSSEAULT, Lisa et al. *HTTP extensions for web distributed authoring and versioning (WebDAV)*. 2007. Tech. zpr. RFC 4918, June.

17. KIMBALL, John. WebDAV: What it is, where it turns up, and its alternatives. In: *Comparitech* [online]. Comparitech Limited, 2020 [cit. 2021-03-04]. Dostupné z: https://www.comparitech.com/net-admin/webdav/#WebDAV_servers_and_clients_still_going_strong.

18. ZFS. In: *ArchWiki* [online]. 2021 [cit. 2021-01-23]. Dostupné z: https://wiki.archlinux.org/index.php/ZFS.

19. MUZIKÁŘ, Zdeněk; ŽĎÁREK, Jan. Filesystém ZFS. In: *Administrace OS Unix* [online]. 2019 [cit. 2021-01-23]. Dostupné z: https://courses.fit.cvut.cz/BI-ADU/media/lectures/06/06-zfs.pdf.

20. Oracle Solaris ZFS Administration Guide. In: *Oracle Help Center* [online]. Oracle Corporation, 2010 [cit. 2021-03-04]. Dostupné z: https://docs.oracle.com/cd/E19120-01/open.solaris/817-2271/index.html.

21. Web application (Web app). In: *TechTarget* [online]. TECHTARGET, 2019 [cit. 2021-04-05]. Dostupné z: https://searchsoftwarequality.techtarget.com/definition/Web-application-Web-app.

22. What Is a Web Application? How It Works, Benefits and Examples. In: *Indeed - Career Guide* [online]. Indeed, 2021 [cit. 2021-04-05]. Dostupné z: https://www.indeed.com/career-advice/career-development/what-is-web-application.

23. HTML: HyperText Markup Language. In: *MDN Web Docs* [online]. 2021 [cit. 2021-02-02]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTML.

24. CSS Introduction. In: *W3Schools Online Web Tutorials* [online]. Refsnes Data, 2021 [cit. 2021-02-03]. Dostupné z: https://www.w3schools.com/css/css_intro.asp.

25. In: *SSH (Secure Shell)* [online]. SSH Communications Security, 2020 [cit. 2021-02-03]. Dostupné z: https://www.ssh.com/ssh/.

26. SVOBODA, Radek. Nejoblíbenější programovací jazyky současnosti a jejich budoucí trendy. In: *CoolClub* [online]. CoolPeople, 2020 [cit. 2021-02-03]. Dostupné z: https://club.coolpeople.cz/nejoblibenejsi-programovaci-jazyky-soucasnosti-a-jejich-budouci-trendy/1372.html.

27. VAN ROSSUM, Guido; DRAKE JR, Fred L. *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

28. DVOŘÁK, Pavel. Django: Úvod a instalace. In: *Zdroják* [online]. 2009 [cit. 2021-02-03]. Dostupné z: https://www.zdrojak.cz/clanky/django-uvod-a-instalace/.

29. ŠTRÁFELDA, Jan. JavaScript. In: *Jan Štráfelda: průvodce online projektem* [online]. [N.d.] [cit. 2021-02-03]. Dostupné z: https://www.strafelda.cz/javascript.

30. ŠTRÁFELDA, Jan. Co je AJAX. In: *Jan Štráfelda: průvodce online projektem* [online]. [N.d.] [cit. 2021-02-03]. Dostupné z: https://www.strafelda.cz/ajax.

31. What is jQuery? In: *jQuery* [online]. OpenJS Foundation, 2021 [cit. 2021-02-23]. Dostupné z: https://jquery.com/.

32. MLEJNEK, Jiří. Před. 6 - Architektonické vzory. In: *BI-SI1* [online]. 2019 [cit. 2021-02-02]. Dostupné z: https://moodle-vyuka.cvut.cz/pluginfile.php/312270/mod_resource/content/1/06.prednaska.pdf.

33. ČÁPKA, David. MVC architektura. In: *ITnetwork.cz* [online]. 2021 [cit. 2021-04-06]. Dostupné z: https://www.itnetwork.cz/navrh/mvc-architektura-navrhovy-vzor.

34. MONUS, Anna. 10 PHP Frameworks For Developers – Best of. In: *HONGKIAT* [online]. 2018 [cit. 2021-02-02]. Dostupné z: https://www.hongkiat.com/blog/best-php-frameworks/.

35. SVOBODA, Radek. Nejpoužívanější softwarové frameworky pro vývoj webových aplikací. In: *CoolClub* [online]. CoolPeople, 2020 [cit. 2021-04-06]. Dostupné z: https://club.coolpeople.cz/nejpouzivanejsi-softwarove-frameworky-pro-vyvoj-webovych-aplikaci/1379.html.

36. KAPICA, Aleš. WebDAV. In: *AbcLinuxu.cz* [online]. 2014 [cit. 2021-05-04]. Dostupné z: https://www.abclinuxu.cz/blog/kenyho_stesky/2014/12/webdav.

37. OWENS, Martin. Python Crontab. In: [online]. 2020 [cit. 2021-04-06]. Dostupné z: https://gitlab.com/doctormo/python-crontab/.

38. In: *Metro 4* [online]. 2020 [cit. 2021-04-06]. Dostupné z: https://www.metroui.org.ua/.

39. SWISNL. In: *jQuery contextMenu plugin & polyfill* [online]. 2020 [cit. 2021-04-06]. Dostupné z: https://www.github.com/swisnl/jQuery-contextMenu.

40. Pagination. In: *Django documentation* [online]. 2021 [cit. 2021-04-06]. Dostupné z: https://docs.djangoproject.com/en/3.1/topics/pagination/.

41. Screen 50/50 Template. In: *W3.CSS Templates* [online]. Refsnes Data, 2021 [cit. 2021-05-05]. Dostupné z: https://www.w3schools.com/w3css/tryit.asp?filename=tryw3css_templates_fifty&stacked=h.

42. Alerts. In: *Bootstrap* [online]. Bootstrap team, [n.d.] [cit. 2021-04-06]. Dostupné z: https://getbootstrap.com/docs/4.0/components/alerts/.

43. EZHOV-EVGENY. Python WebDAV Client 3. In: [online]. 2020 [cit. 2021-04-06]. Dostupné z: https://github.com/ezhov-evgeny/webdav-client-python-3.

44. ARJAN-S. python-zipstream. In: [online]. 2020 [cit. 2021-04-06]. Dostupné z: https://github.com/arjan-s/python-zipstream.

45. BARRY-SCOTT. config-path. In: [online]. 2021 [cit. 2021-04-21]. Dostupné z: https://github.com/barry-scott/config-path.

46. GORAKHARGOSH. Watchdog. In: [online]. 2021 [cit. 2021-04-21]. Dostupné z: https://github.com/gorakhargosh/watchdog.

Příloha A

Seznam použitých zkratek

AJAX Asynchronous JavaScript and XML

CGI Common Gateway Interface

CSS Cascading Style Sheet

GUI Graphical user interface

HTML Hypertext Markup Language

MVC Model-view-controller

CLI Command Line Interface

SSH Secure Shell

WebDav World Wide Web Distributed Authoring and Versioning

XML Extensible markup language

ZFS Zettabyte File System

Příloha B

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
- `videos`
  - `showcase.mp4` ..... ukázka použití aplikace ve formě videa