VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

APLIKACE PRO ŘÍZENÝ PŘÍSTUP KE VZDÁLENÝM DOKUMENTŮM PRO GNU/LINUX

APPLICATION FOR CONTROLLED ACCESS TO REMOTE DOCUMENTS FOR GNU/LINUX

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **JAN BERNARD**
AUTHOR

VEDOUCÍ PRÁCE — **RNDr. MAREK RYCHLÝ, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Bernard Jan**
Program: Informační technologie
Název: **Aplikace pro řízený přístup ke vzdáleným dokumentům pro GNU/Linux**
Application for Controlled Access to Remote Documents for GNU/Linux
Kategorie: Operační systémy

Zadání:

1. Seznamte se s požadavky týkající se zabezpečení přístupu k dokumentům v projektu Validované datové úložiště (VDU). Prozkoumejte možnosti virtuálních souborových systémů a integrace aplikací do desktopového prostředí v operačním systému GNU/Linux.
2. Navrhněte klientskou aplikaci pro VDU, která se připojí k úložišti a bude zpřístupňovat obsah úložiště pod zadaným přístupovým klíčem jako soubor virtuálního souborového systému s řízeným přístupem a správou verzí. Navrhněte automatické testy požadovaných vlastností takové aplikace.
3. Po konzultaci s vedoucím navrženou aplikaci implementujte včetně automatických testů.
4. Výsledek popište, vyhodnoťte a zveřejněte jako open-source.

Literatura:

- Interní dokumentace projektu Validované datové úložiště.
- VANGOOR, Bharath Kumar Reddy; TARASOV, Vasily; ZADOK, Erez. To FUSE or Not to FUSE: Performance of User-Space File Systems. In: 15th USENIX Conference on File and Storage Technologies (FAST 17). 2017, s. 59-72. Dostupné z: [https://www.usenix.org/conference/fast17/technical-sessions/presentation/vangoor]
- BURIHABWA, Dorian, et al. SGX-FS: Hardening a File System in User-Space with Intel SGX. In: 2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom). IEEE, 2018, s. 67-72. Dostupné z: [https://doi.org/10.1109/CloudCom2018.2018.00027]
- xdg-utils. *freedesktop.org* [online], 2019 [cit. 2020-10-26]. Dostupné z: [https://freedesktop.org/wiki/Software/xdg-utils/]

Pro udělení zápočtu za první semestr je požadováno:

- Body 1, 2 a rozpracovaný bod 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Rychlý Marek, RNDr., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 27. října 2020

Abstrakt

Práce se zaměřuje na synchronizaci a životní cyklus souborů stažených do uživatelského počítače ze vzdáleného validovaného datového uložiště. Z analýzy trhu vyplynulo, že na trhu je velký výběr aplikací, ale rozdíly mezi nimi jsou relativné velké. Na základě požadavků a stanovené komunikace s validovaným datovým uložištěm byla navržena a implementována aplikace pro systém GNU/Linux. Testování apliakce probíhalo na dvou vybrancých Linuxových distribucích s mock serverem zastupující vzdálené uložiště.

Abstract

The aim of this thesis is synchronization and life cycle of files downloaded to user's computer from validated data storage. Based on market analysis there is large selection of applications in market but there are relatively large differences along them. Application was designed and implemented for GNU/Linux according to requirements and given validated data storage communication. The application was tested on two selected Linux distributions with mock server represententing remote storage.

Klíčová slova

Validované datové uložiště, VDU, Linux, souborový systém, HTTP, FUSE, C++, synchronizace, soubor, dokument, oprávnění

Keywords

Validated data storage, VDS, Linux, filesystem, HTTP, FUSE, C++, synchronization, file, document, permission

Citace

BERNARD, Jan. *Aplikace pro řízený přístup ke vzdáleným dokumentům pro GNU/Linux.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce RNDr. Marek Rychlý, Ph.D.

Aplikace pro řízený přístup ke vzdáleným dokumentům pro GNU/Linux

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana RNDr. Marka Rychlého, Ph.D. a uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jan Bernard
6. května 2021

Poděkování

Rád bych poděkoval vedoucímu práce, doktorovi Markovi Rychlému, za vynikající vedení práce a jeho podnětné poznatky a rady. Také bych rád poděkoval všem svým kolegům, se kterými jsem procházel celé bakalářské studium. A v neposlední řadě děkuji všem svým přátelům a rodině za podporu a pochopení.

Obsah

Kapitola 1

Úvod

V době vysokorychlostního a stabilního internetu je využívání vzdálených uložišť běžnou záležitostí. Pro většinu uživatelů je to nástroj pro jednoduchou synchronizaci dat napříč několika zařízeními od mobilního telefonu po stolní počítač. Pokud vezmeme v úvahu pouze velká datová centra, jedná se pravděpodobně o nejlevnější a nejspolehlivější způsob uchování dat, protože tyto centra mají velkou kapacitu úložného prostoru a také zálohy na softwarové i hardwarové úrovni. Pod pojmem vzdálené uložiště si nemusíme představit jen velká datová centra se stovkami serverů, může se jednat o relativně malé uložiště ve firmě nebo dokonce o osobní/domácí řešení. Pro využívaní osobních/firemních uložišť se lidé uchylují v případech, je-li třeba uložit osobní nebo určitým způsobem citlivá data, která by nemohla být poskytnuta třetí straně. Nabízená řešení také nemusí poskytovat potřebnou funkcionalitu nebo jejich finanční model nesplňuje zákazníkova kritéria.

V průběhu let byly vyvinuty protokoly řešící sdílení souborů jako FTP, NFS a mnohé další. Sdílení souborů je komplexní záležitostí a obsahuje velké množství parametrů. Každý protokol se zaměřuje jen na určité parametry jako zabezpečení a spolehlivost přenosu, oprávnění pro jednotlivé uživatele a soubory, synchronizace modifikovaných souborů apod nebo pojme daný parametr odlišným způsobem. V dnešní době je většina komerčně využívaných aplikací vystavěna na proprietárních protokolech nebo na protokolech umožňující obecnější použití. Jedním z nejpoužívanějších aplikačních protokolů bude Hypertext Transfer Protokol neboli HTTP.

Cílem této práce je prozkoumat aktuální nabídku a možnosti na poli vzdálených uložišť a navrhnout aplikaci pro řízený přístup ke vzdáleným dokumentům pro platformu GNU/Linux. Aplikace bude zaměřena na řízený životní cyklus souborů s vynucenými oprávněními pro jednotlivé soubory daná vzdáleným uložištěm. Komunikace skrze HTTP protokol byla definovaná externím zadavatelem. Zdrojové kódy jsou dostupné na veřejném Github repozitáři. [1]

[1] https://github.com/PlayerBerny12/VUT-IBT-Code

Kapitola 2

Motivace a existující řešení

Popularita a využití vzdálených uložišť roste, protože lidé využívají více zařízení a potřebují mezi nimi jednoduchou synchronizaci nebo na jednom souboru potřebuje spolupracovat více lidí. Dalším podnětem využívání vzdálených uložišť je většího množství dat a potřeba tyto data efektivně sdílet a bezpečně uložit. Postupně se digitalizují další systémy například ve státní správě nebo dalších podnikatelských sektorech.

Na trhu je velké množství služeb zaměřujících se na různé klientské potřeby. Diverzita nabídky je až překvapivě velká. Většina se zaměřuje na ukládání a sdílení dat ve svém ekosystému, které jsou určené pro široké použití. Pokud jsou požadavky specifické, tak možnost vlastní konfigurace není možná. Pro částečnou úpravu nebo automatizaci procesů lze využít poskytované API, ve většině případu se jedná o variantu REST API. Poslední možností je vystavění obdobné služby z několika existujících aplikací nebo vytvoření nové.

v kontextu této práce jsou požadavky následující:

- Dodržování oprávnění i na klientském systému
  - read-only
  - write-only
  - read/write
- Životní cyklus souboru (stažení)
  - stažení
  - případná modifikace
  - smazání souboru po vypršení platnosti
- Autentifikace
  - uživatelské jméno a klientský certifikát
  - uživatelské jméno a heslo

2.1 Současná řešení

Pro bližší analýzu byl vybrán vzorek služeb a programů obsahující velké ekosystémy služeb po open source aplikace. Zkoumáno bylo několik parametrů, jako poskytovaná funkcionalita pro jednotlivce a firmy, cena, integrace s ostatními systémy a aplikacemi apod. Obecně

nelze jednoznačně určit nejlepší uložiště, ale je možné poukázat na rozdílné vlastnosti a vyzdvihnout kladné. Koncový uživatel se následně může rozhodnout dle svých potřeb.

2.2 Google Drive

Pro synchronizaci obsahu z Google Drive na osobní počítač se využívá aplikace pojmenovaná Google File Stream. Je možné ji provozovat pouze na systémech Windows a Mac OS.[6] Pro Linuxové distribuce lze využít například neoficiální open source aplikaci Google Drive OCamlFUSE[1] využívající technologii FUSE, která bude popsána v následující kapitole 3.4.6. Google Disk poskytuje plně funkční webové rozhraní, ve kterém je možné dokumenty přímo upravovat bez nutnosti stahování. Problém nenastal ani v případě konkurenčních Microsoft Office dokumentů.

Google Workspace (dříve Google Suite) je možné pořídit v několika balíčcích. Například balíček Bussiness Standard nabízí 2 TB uložiště za 10,40 EUR měsíčně. Balíček obsahuje další služby jako firemní email a schůzky až o 150 účastnících na Google Meet.[10]

Každý uživatel má vlastní disk s omezenou kapacitou podle balíčku. Na disku lze vytvářet standartní složkovou hierarchii a je možné sdílet jednotlivé soubory nebo obsah složek s ostatními Google uživateli. Verze Enterprise umožňuje vytváření dalších disků, na které je možné přiřadit seznam uživatelů. Každý uživatel má nastavenou jednu z 6 rolí, které vymezují jeho možnosti. k dispozici je API, která pokrývá veškeré možnosti webového rozhraní jako vytvoření souboru, sdílení atd.[9]

2.3 Dropbox

Dropbox má oficiální balíčky své aplikace pro Ubuntu a Fedoru, případně je možné si aplikaci zkompilovat ze zdrojových souborů. Podpora Windows a Mac OS je samozřejmostí. Z práce „Personal Cloud Storage Benchmarks and Comparison“ lze vyčíst, že Dropbox se zaměřuje na efektivitu přenosu a minimální zatížení sítě. Využívá menší množství TCP spojení oproti jeho konkurentům a také stejně jako Google Drive komprimuje data před odesláním.[1] Toto řešení nemusí dosahovat nejvyšší výkonosti, ale nezatěžuje tolik infrastrukturu.

Webové rozhraní je více orientováno na správu souborů a jejich historii. Nabízí jednoduché obnovení smazaných souborů nebo návrat k předchozí verzi. Nenabízí tolik možností jako Google Drive, ale upravovat dokumenty ve webovém prohlížeči lze také. Na výběr je mezi Microsoft Office Online nebo Google Worksapce. Sdílení souborů a složek je možné i s uživateli, kteří nemají Dropbox účet.

Balíček Plus za 9,99 EUR měsíčně dává uživateli přístup ke 2 TB uložišti. Firemní balíčky obsahují rozšíření pro lepší správu uživatelů, vytváření skupin a admin panel/konzole. Dropbox má také API nabízející dostatečnou funkcionalitu pro případnou integraci s jinými systémy apod.[7]

2.4 pCloud

Poskytovatel pCloud podporuje nejpoužívanější platformy, a to včetně mobilních. Za 9,99 EUR měsíčně dostane uživatel 2 TB uložiště. Jako jediný z poskytovatelů v analýze nabízí

[1] https://github.com/astrada/google-drive-ocamlfuse

balíček s jednorázovou platbou za cenu 350 EUR. Jedná se o identický balíček, pouze forma platby se liší. Všechna data jsou přenášena přes šifrovaný kanál a všechny kopie souborů na pěti různých serverech jsou šifrovaná 256bitovým AES klíčem.[17]

Webové rozhraní je jednoduché, avšak oproti konkurentům má méně funkcí. Lze zobrazit náhled dokumentů, ale upravovat je nelze. Každý soubor má tzv. revize a je možné obnovit obsah souboru na vybranou revizi. Revize jsou uchovány po dobu jednoho roku. Soubory je možné sdílet i s uživateli, kteří nemají účet u pCloudu.

Pro firemní zákazníky je nabízený rozšířený management uživatelů a monitoring s logy aktivity jednotlivých uživatelů.

2.5 Syncthing

Open source aplikace Syncthing synchronizuje soubory peer-to-peer. Nejedná se o čistou peer-to-peer architekturu, protože jeden z uzlů může být nastaven jako server a veškeré data budou synchronizována vůči tomuto uzlu. Syncthing je možné provozovat na většině dnešních systémů jako Windows, Linux, BSD a Mac OS.[19]

Aplikace poskytuje webové rozhraní, které je určené pouze pro nastavení parametrů jako discovery protokol pro objevování ostatních uzlů, jaké složky mají být synchronizovány nebo kolik verzí jednotlivých souborů má být uchováváno a mnoho dalšího. Velká přizpůsobivost umožňuje upravit fungování aplikace vlastním potřebám, na druhou stranu bude náročné udržovat systém s větším množstvím uživatelů vystavěný na této aplikaci. Jedná se tedy spíše o domácí řešení.

2.6 Shrnutí

Trendy na poli vzdálených uložišť určují velké firmy jako Google nebo Micorsoft. Důkazem tohoto jevu je například integrace aplikací obou těchto firem do webového rozhraní Dropboxu. Aplikace One Drive od Microsoftu nebyla blíže analyzovaná, protože se jedná o službu pouze pro platformu Windows.

Všechny služby mají webové rozhraní, ke kterému je aplikace synchronizující obsah uložiště občas brána jako doplněk. Sdílení nebo obnova předchozí verze souboru je vždy možná ve webové aplikaci, ale ne vždy jsou tyto akce dostupné i na desktopové verzi.

Aplikace na jejíž základě je vytvořená tato práce nebude synchronizovat celé uložiště, ale bude pracovat jen s jednotlivými soubory. Uživateli na vyžádání stáhne právě jeden soubor, se kterým bude chtít uživatel v danou dobu pracovat (zobrazit nebo upravit obsah). Synchronizace celého uložiště nebo obsahu jedné složky nebude možná. Jedná se spíše o náhradu možnosti zobrazovat a upravovat soubor přímo ve webovém prohlížeči jako to umožňuje například Google Drive.

Kapitola 3

Návrh řešení a použité technologie

Prvním krokem navrhovací fáze bylo seznámení se zadáním a požadavky. Externí zadavatel dodal podrobnou dokumentaci REST API pro validované datové uložiště, která byla stěžejním dokumentem při návrhu. Jednalo se formální textový popis, který nebyl v žádném standardizovaném formát. API obsahuje funkce pro ověření spojení, autentifikaci a práci se soubory. VDU nebylo při vzniku této práce k dispozic, proto byl pro potřeby vývoje vytvořen Mock Server na základě přepisu poskytnuté dokumentace do standardizovaného formátu OpenAPI.

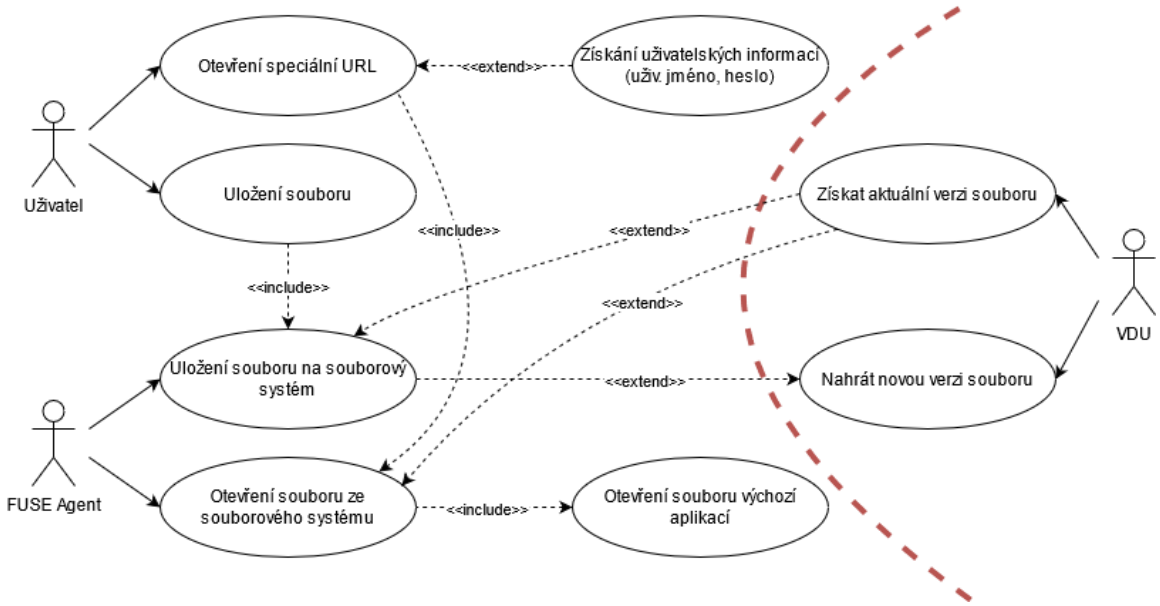
Pro vynucení práv jednotlivých souborů je standartní souborový systém nevyhovující, protože po stažení souboru do počítače uživatele ztrácí VDU kontrolu nad tímto souborem. Po konzultaci s vedoucím práce, doktorem Markem Rychlým, byla zvolena technologie FUSE neboli Filesystem in userspace. FUSE umožňuje vytvořit virtuální souborový systém a implementovat jednotlivé operace jako čtení, zápis atd. Tím je možné dosáhnout dostatečnou kontrolu nad oprávněními a obecně životním cyklem souboru v uživatelově počítači. Znalý uživatel by byl schopen tento systém obejít a přistoupit k souborům z hostovaného souborového systému. k takové operaci jsou však potřeba oprávnění superuživatele `root`.

3.1 Případy užití

Na základě požadavků bylo možné přímo vytvořit diagram případu užití. Pouze jedna věc nebyla konkrétně specifikovaná. Webová aplikace musí předat té desktopové přístupový token, pomocí kterého lze volat potřebné API metody VDU. v úvahu přišlo stažení „fake“ souboru obsahující pouze přístupový token. Druhou možností bylo přesměrování na speciální URL z VDU. Prohlížeč by na základě přesměrování rozpoznal, že má otevřít určenou aplikaci a předat ji token jako jeden z parametrů volání. Varianta se speciální URL působí na uživatele mnohem ucelenějším dojmem a nevytváří na souborovém systém dále nepotřebné soubory.

Diagram případů užití zachycuje celý systém jako celek a zobrazuje jednotlivé funkce a jejich návaznosti, které je třeba dále analyzovat a specifikovat v dalších fázích návrhu. v diagramu 3.1 vystupují tři aktéři a to uživatel, FUSE agent/démon a VDU. Implementace VDU není obsahem této práce, proto je oddělena červenou přerušovanou čarou. v diagramu není zanesená například autentifikace, na které závisí veškerá komunikace s VDU. Pro přehlednost byla tato část vypuštěna.

Všechny akce jsou spuštěny uživatelem a většina z nich vyvolá reakci FUSE démona. Jedná se například o uložení, otevření nebo přejmenování souboru ve virtuálním souborovém



Obrázek 3.1: Diagram případu užití

systému. Speciální případ nastává při otevření URL s vlastním protokolem, který má zavolat aplikaci, jež je k danému protokolu v systému přiřazena. Příkladem takového protokolu může být `mailto`, využívaný často u webových stránek. Přesměrování na adresu s tímto protokolem otevře správce elektronické pošty.

K námi definovanému protokolu pojmenovaném `vdu` je třeba mít v systému přiřazenou aplikaci, která pomocí přístupového tokenu uloženého v URL (`vdu://<přístupový token>`) stáhne žádaný soubor z VDU na virtuální souborový systém. Na základě této znalosti byly uskutečněny rozhodnutí při návrhu architektury.

3.2 Definice REST API

API obsahuje sedm metod pro ověření dostupnosti, autentifikaci a práci se soubory. Každá funkce má sadu možných návratových hodnot, která je podmnožinou návratových hodnot definovaných protokolem HTTP. Podle tohoto kódu lze rozpoznat, za bylo dané volaní úspěšné, případně jaká chyba při volání nastala. Kódy z rozsahu 200 až 299 označují úspěšná volání a kódy 400 až 599 indikují chybu na straně klienta nebo serveru.

Standardizace definice REST API byla vytvořena v OpenAPI standardu, pro který je dostupných mnoho nástrojů. Jedním z nich je open source nástroj Swagger, pomocí kterého můžete generovat html dokumentaci. Vygenerované rozhraní umožňuje i jednoduchou formu testování API. Pomocí standardizované definice bylo také možné rychle vytvořit Mock Server ve webové aplikaci Postman.

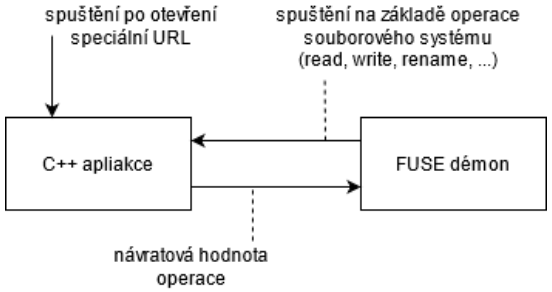
- /ping
  - GET /ping — pro ověření dostupnosti VDU
- /auth
  - GET /auth/key — obnova autentifikačního tokenu

  - POST /auth/key — získání autentifikačního tokenu
  - DELETE /auth/key — zneplatnění autentifikačního tokenu
- /file
  - GET /file/<file-access-token> — stažení obsahu souboru
  - POST /file/<file-access-token> — nahrání obsahu souboru
  - DELETE /file/<file-access-token> — zneplatnění přístupového tokenu

3.3 Architektura

Zvažovány byly dvě implementační varianty, jedna verze pracoval pouze s FUSE démonem a druhá rozdělila funkčnost do samostatné aplikace a FUSE démona. První variantu by bylo možné zpracovat dvěma obdobnými způsoby, jež by vedli ke obdobnému řešení. FUSE démon je napsaný v jazyce C, takže jedno z nabízejících se řešení by byla implementace celé aplikace jednom jazyce. Druhým řešením by bylo vytvoření C++ knihovny, která by měla interface pro jazyk C. Takto vytvořená knihovna by mohla být konzumována FUSE démonem. Nejedná se však o nijak kriticky náročnou aplikaci na výkon, takže by varianta s C++ knihovnou byla upřednostňovaná, protože standartní knihovna jazyka C++ a jeho konstrukce umožňují jednodušší vývoj.

Rozdělení na dvě samostatné aplikace, z nichž jedna je FUSE démon by mohlo přinést problémy v udržitelnosti takového systému. Pokud by jedna z aplikací nefungovala, celý systém by nemohl pracovat korektně. Hlavní aplikace by obsluhovala komunikaci s VDU a byla by vždy spuštěna voláním z FUSE démona nebo nepřímo uživatelem. Se znalostí získanou při návrhu diagramu případů užití víme, že je třeba mít registrovanou aplikaci obsluhující otevření speciální URL. i kdybychom vzali místo URL soubor se speciální koncovkou (např. *.vdu) obsahující přístupový token je stále třeba mít přiřazenou aplikaci pro obsluhu tentokrát otvírání souboru. Démon běžící na pozadí po celou dobu běhu systému nemůže být přiřazen jako aplikace obsluhující takové události. Na základě těchto okolností byla vybrána varianta se dvěma oddělenými aplikacemi k dalšímu zpracování.



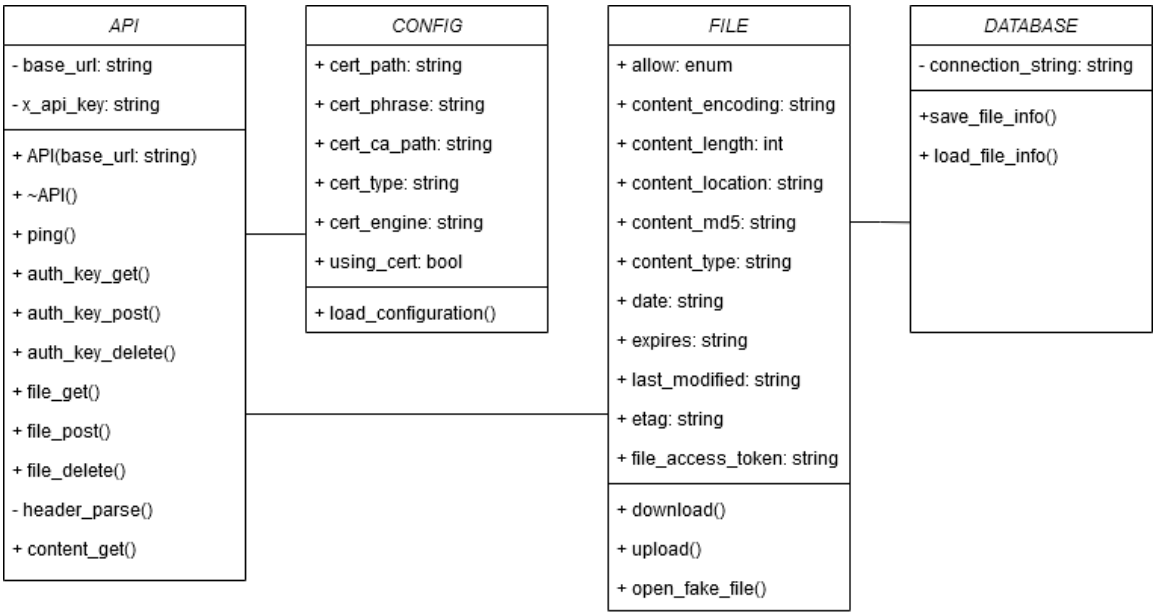
Obrázek 3.2: Architektura

Pokud FUSE démon má vykonat akci, pro kterou nemá dostatečné informace, spustí jako další subproces druhou hlavní aplikaci a počká na návratovou hodnotu, na základě které se rozhodne, jakým způsobem onu akci dokončí. Může se jednat o situace jako: je soubor určený jen pro četní nebo i zápis, je daný soubor stále validní validní (nevypršel datum expirace) apod.

V této variantě je FUSE démon velmi minimalistický a zbytek funkcionality je implementován v druhé aplikaci, která běží jen pokud je k tomu vyzvaná, narozdíl od neustále běžícího démona. Démon tedy po celou dobu běhu zabírá méně místa, protože nemusí mít načtené v paměti všechny potřebné knihovny. Tyto knihovny mohou být v paměti i tak namapované, protože je mohou využívat jiné aplikace. Démon musí být spuštěný s oprávněními superuživatele `root` a menší množství kódu běžícího s vysokými oprávněními implikuje menší náchylnost na bezpečnostní chyby. Hlavní aplikace může běžet jen s právy přihlášeného uživatele, což je jedna z dalších výhod oproti druhé zvažované variantě.

3.3.1 Diagram tříd

Aby implementace probíhala s možná nejmenším množstvím komplikací a nebylo třeba již uvažovat nad dalšími rozhodnutími, byl vytvořen diagram tříd 3.3, který se pokouší podrobně vizualizovat třídy s atributy a metodami, jež je třeba implementovat. Jedná se pouze o návrh, který se v průběhu vývoje bude měnit, ale pokud je návrh kvalitní a vývojáři se jej snaží dodržet, neměl by být výsledek zásadně rozdílný.



Obrázek 3.3: Diagram tříd - návrh

Na obrázku 3.4 je možné vidět diagram tříd naimplementovaného programu. Na první pohled vypadá velmi rozdílně, po bližším prozkoumáním lze vidět jednu novou třídu oproti návrhu. Jedná se o třídu zobrazující notifikace pro uživatele, která nemá vliv na hlavní funkcionalitu. Dále přibylo několik podpůrných funkcí a také přibily některé atributy, případně se přejmenovali. Při implementaci nenastaly žádné zásadní potíže způsobené chybným nebo nepřesným návrhem, tudíž lze návrh označit za úspěšný.

Obrázek 3.4: Diagram tříd - implementace

3.4 Technologie

V této sekci budou popsány jednotlivé technologie, které ovlivnili tuto práci. Pro samotný návrh stačí elementární znalosti daných technologií, ale pro implementaci a jejich správné využití je třeba jejich bližší pochopení.

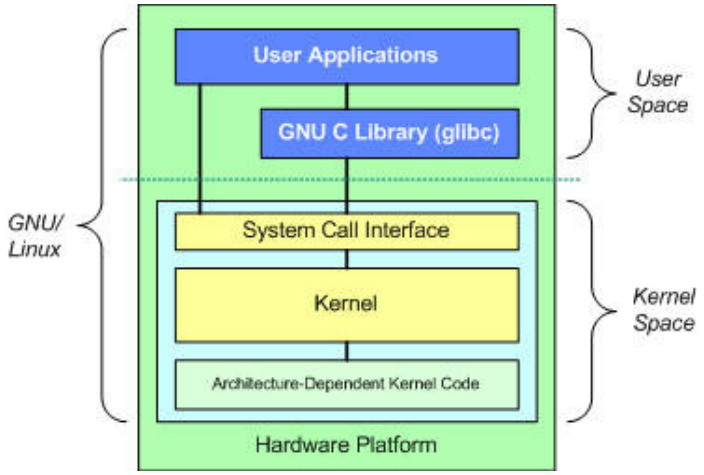
3.4.1 GNU/Linux

GNU je rekurzivní zkratka pro „GNU's Not Unix!“. Jedná se o projekt, který měl za cíl vystavět nový operační systém kompatibilní s Unixem. Všechen kód vytvořený pod tímto projektem je tzv. free software (svobodný software). Jedná se o filosofii vývoje software jejíž

parafrázovaná definice je následující. Svobodný (free) software se vztahuje ke svobodě, ne k ceně. Slovo svobodný (free) by mělo být bráno ve významu svobodný projev (free speech) nikoliv pivo zdarma (free beer) [5].

V projektu GNU postupně vznikali jednotlivé softwarové balíky, které měli ve výsledku poskládat nový operační systém. Pod projektem GNU vnikli aplikace jako překladač GCC, textový formátovač TeX, terminálový shell Bash a mnohé další. Na počátku 90. let byl systém téměř hotový, scházelo jen jádro operačního systému. Ve stejném období dokončoval Linus Torvalds své jádro operačního systému nazývané Linux.[22][18]

Vznikla tedy myšlenka tyto dva projekty spojit a po vyřešení problémů s kompatibilitou jednotlivých částí projektů vnikl GNU/Linux. Balíčky projektu GNU umožnovali práci se systémem postaveným nad Linuxovým jádrem. Všechny dnešní distribuce jako jsou Debian, Fedora a další jsou distribuce systému GNU/Linux. Linux je název pouze jádra pro operační systém, který zevšeobecněl a je tímto názvem chybně označován celý systém. Projekt GNU dodnes vyvíjí své jádro pojmenované GNU Hurd, se kterým by vznikl operační systém GNU, což byl počáteční cíl projektu.[18]



Obrázek 3.5: Architektura GNU/Linux. Převzato z [13]

Architekturu GNU/Linux vyobrazená na obrázku 3.5 vyznačuje hranici mezi kernelem a uživatelským prostorem. Jádro/kernel běží v tzv. priviligovaném režimu, což je režim s nejvyššími oprávněním a může na CPU provádět jakékoliv operace. Všechny ostatní aplikace beží v uživatelském prostoru a jsou omezeny jen na provádění podmnožiny operací oproti priviligovanému režimu. Pokud aplikace potřebuje vykonat operaci, která vyžaduje vyšší oprávnění, zažádá jádro operačního systému, aby tuto operaci provedlo za místo ní. Tím je docíleno vyššího zabezpečení operačního systému.

3.4.2 C++/C

Jazyk C patří do skupiny dlouho používaných programovacích jazyků. Se svoji bohatou historií sahající až do 70. let 20. století velmi ovlivnil další nástupnické procedurální jazyky jako C++, C#, Java atd. v roce 1989 byl jazyk standardizován jako ANSI C a následně přišli ISO standardy označované jako C99, C11, C17. Jazyk C je řazen mezi vysokoúrovňové programovací jazyky, přestože relativně jednoduchý a obsahuje malé množství konstrukcí v porovnání s ostatními jazyky.[11]

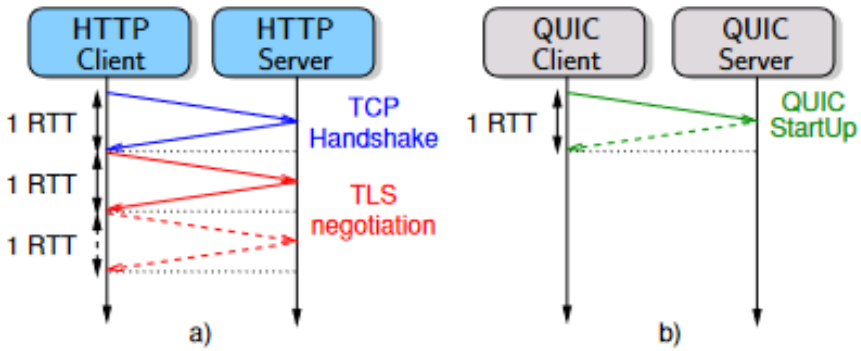
Vývoj C++ je oproti C dynamičtější, což můžeme vidět na jednotlivých ISO standardech C++98, C++03, C++11, C++14, C++17 a C++20. Jazyk C++ se v poslední dekádě velmi proměnil a období od standardu C++11 je označováno za dobu „moderního C++“. v posledním standardu přibyla například podpora konceptů nebo rozsahů.[12]

Při využití překladačů GCC nebo G++ lze využít tzv. GNU rozšiřující standardy. Poté je možné využít některé konstrukce, které nejsou specifikované v oficiálním ISO standardu. V případě jazyka C se jedná o standardy pojmenované GNU99, GNU11 a GNU17, v případě C++ se jedná o GNU++11 a jemu podobné.

3.4.3 HTTP

Hypertext Transfer Protokol je protokol aplikační vrstvy TCP/IP stacku. Byl vytvořen pro přenášení hypertextových souborů v 90. letech 20. století a dnes se jedná o nejrozšířenější aplikační protokol. Poslední stabilní verze protokolu označená jako HTTP/2 byla vydaná v roce 2015. Aktuálně je ve fázi specifikace nástupce pojmenovaný HTTP/3.

HTTP v čisté formě dnes již téměř není možné potkat, ale využívá se ve spojení s šifrovacím protokolem TLS. Při komunikaci se nejprve vytvoří TCP spojení a poté šifrovaný kanál, skrze který je HTTP bezpečně přenášeno. Toto řešení není nejefektivnější, protože vznikalo postupně a spojovali se technologie, které nebyly od počátku navržené pro tento účel. Ve většině případů uživatel na stabilním připojení nezaregistruje žádné potíže, kterými tyto technologie trpí. Jedním z problémů je dlouhá časová prodleva, než se kanál ustálí pro přenášení dat. Tento problém má vyřešit právě nová verze HTTP.[3]



Obrázek 3.6: Porovnání HTTP verze 2 (a) a 3 (b). Převzato z [3]

Nová verze využívá nový protokol QUIC, který je přenášen protokolem UDP namísto TCP jako tomu bylo u předchozích verzích HTTP. Na obrázku 3.6 je možné vidět, že navázání komunikace je ze tří výměn informací zredukováno jen na jednu. Formát hlavičky HTTP by měl zůstat nezměněný, a tudíž je zpětně kompatibilní.

Pro tento projekt je důležitá práce s datumy, přesněji formáty umožněné přenášet ve validní HTTP hlavičce. RFC7231 specifikuje tři formáty datumů, z nichž dva jsou označené jako zastaralé, ale jsou stále validní. Udržované jsou kvůli zpětné kompatibilitě s protokolem HTTP/1.1.[8]

- Upřednostňovaný formát
  - Sun, 06 Nov 1994 08:49:37 GMT (IMF-fixdate formát)
- Zastaralé formáty
  - Sunday, 06-Nov-94 08:49:37 GMT (RFC 850 formát)
  - Sun Nov 6 08:49:37 1994 (ANSI C asctime() formát)

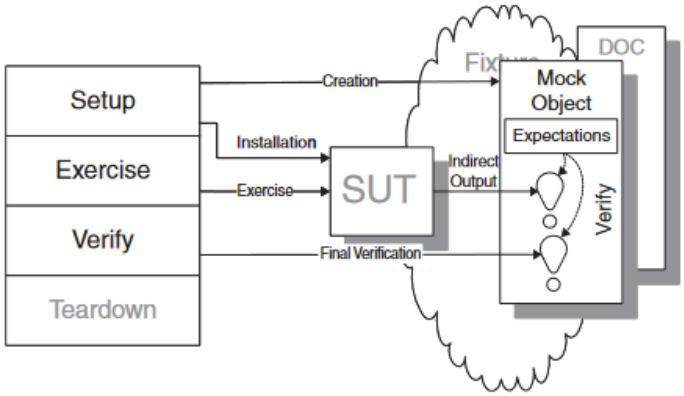
3.4.4 OpenAPI

Jedná se o standard pro definování metod REST API, který má v čase psaní této práce poslední verzi 3.0.3. Zápis je možný ve dvou jazycích: JSON a YAML. Oba je možné strojově zpracovat a přitom jsou pro člověka stále čitelné.

Specifikace je velmi obsáhlá a umožňuje popsat API velmi detailně a přitom efektivně. Například je možné vytvářet šablony datových strukturu, které jsou používány jako vstupní nebo naopak návratové hodnoty jednotlivých funkcí. v definici je pak možné na tuto šablonu odkázat. Struktura souboru je hierarchická a skládá se z jednotlivých objektů. Každý objekt má povinné a volitelné atributy. Definovat tedy lze hlavičky požadavku, URL query parametry, návratové kódy a k nim přiřazené hlavičky odpovědí a odpovědi samotné, případně příklady možných odpovědí. Toto je jen stručný výpis možností, které standard OpenAPI definuje.[16]

3.4.5 Mock Server

Mock Server je speciální případ testovacího vzoru Mock Object, který spadá do kategorie Test Double. Test Double je kategorie testovacích vzorů, do které patří vzory jako právě Mock Object, Fake Object, Test Stub apod. Jednotlivé vzory mohou být mezi sebou zaměnitelné, protože jejich definice a použití je velmi podobné. Tato kategorie vrozů je využívaná pro systémy, které pracují s určitou externí komponentou, která není dostupná nebo by testování ovlivnila svými nežádoucími vedlejšími účinky.[15, 522–524] k využití Mock Serveru bylo přistoupeno z důvodu nedostupnosti VDU, při vzniku této práce. Mock Server nebyl využíván jen pro účeli testování, ale i během vývojové fáze. Aby nedošlo k zanesení chyby do software během vývoje, byla napsána sada testů, která ověřovala správnou implementaci Mock Serveru vůči předem specifikované definici.

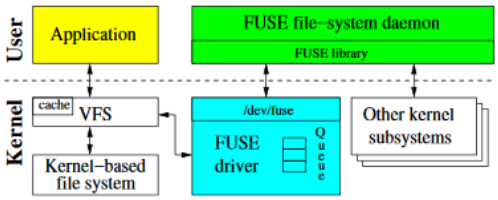


Obrázek 3.7: Test Double schéma. Převzato z [15, 544]

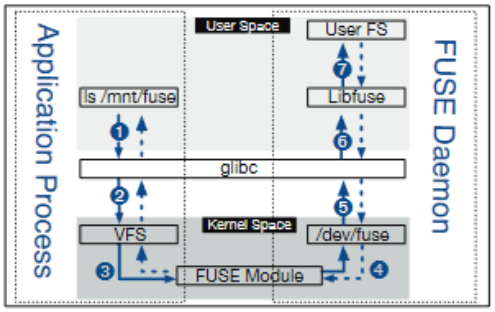
Mock Object pokrývá funkcionalitu Test Stub vzoru a na vývojáři záleží, jakým způsobem využije tyto možnosti. v kontextu této práce by se používaný Mock Server dal klasifikovat jako pouhý Test Stub, protože není využívaná verifikující část, kterou definuje Mock Object vzor.[15, 524–525] Verifikovaný je pouze výstup funkcí v testovaném systému.

3.4.6 Souborový systém v uživatelském prostoru (FUSE)

FUSE je označení pro File system in userspace a jedná se o framework umožňující vytvořit vlastní virtuální souborový systém v uživatelském prostoru. FUSE se skládá ze dvou čistí, démona a modulu pro jádro operačního systému. Je-li modul načtený registruje ovladač v Linuxovém VFS (Virtual filesystem). Tento ovladač poté představuje proxy pro všechny běžící démony. FUSE driver obsahuje několik front pro obsloužení všech potřebných akcí. Nejprioritnější fronta je pro obsloužení přerušení, ostatní fronty jsou pro obsluhu synchronních, asynchronních a a tvz. forget požadavků. Priorita zpracování těchto front je daná poměrem 8 ku 16 pro synchronní a asynchronní ku forget požadavkům.[20]



(a) Architektura. Převzato z [20]



(b) Datový tok. Převzato z [2]

Obrázek 3.8: Vysokoúrovňové diagramy architketury a datového toku pro FUSE

Démon zpracovává postupně požadavky, které se přidávají do front ve FUSE ovladači. Na obrázku 3.8b je vidět jak požadavek prochází jednotlivými částmi systému. Aplikace přistoupí na připojený FUSE souborový systém a požadavek projde skrze VFS a FUSE ovladač až k démonovi. Démon daný požadavek zpracuje, ve většině případů přistoupí na hostitelský souborový systém a vykoná na něm požadovanou operaci. Odpověď je poté navrácena stejným způsobem, jakým byla přenesena k FUSE démonovi.

Démon nemusí implementovat všechny operace, záleží jen na požadavcích. Je-li potřeba vytvořit souborový systém jen pro čtení, implementují se například operace INIT, OPEN, GETATTR, READ a RELEASE. Mohou být implementované další kvůli optimalizacím nebo řešící specifické případy. Více informací k jednotlivým operacím je možné dohledat v dokumentaci libfuse[1].

[1] http://libfuse.github.io/doxygen/structfuse__operations.html

Skupina	Typ požadavku
Speciální	INIT, DESTROY, INTERRUPT
Metadata	LOOKUP, FORGET, BATCHFORGET, CREATE, UNLINK, LINK, RENAME, RE-NAME2, OPEN, RELEASE, STATFS, FSYNC, FLUSH, ACCESS
Data	READ, WRITE
Atributy	GETATTR, SETATTR
Rozšířené atributy	SETXATTR, GETXATTR, LISTXATTR, REMOVEXATTR
Symbolické linky	SYMLINK, READLINK
Složky	MKDIR, RMDIR, OPENDIR, RE-LEASEDIR, READDIR, READDIRPLUS, FSYNCDIR
Zamikání	GETLK, SETLK, SETLKW
Misc	BMAP, FALLOCATE, MKNOD, IOCTL, POLL, NOTIFY-REPLY

Tabulka 3.1: Tabulka typů požadavku pro zpracování FUSE démonem. Převzato z [20]

3.4.7 D-Bus

D-Bus je systém pro posílání zpráv mezi jednotlivými aplikacemi. Přesněji se jedná o Inter-Process Communication (IPC) a Remote Procedur Calling (RPC) mechanizmus vytvořený pro komunikaci mezi jednotlivými procesy na stejném počítači. D-Bus funguje na Unix-like systémech a port pro operační systém Windows je také dostupný. Objekty jsou identifikovány kombinací tzv. „bus name“ a „object path“.[4]

Existují dva typy „bus name“, jeden je unikátní a je přiřazen každé připojené aplikaci na sběrnici. Pro tyto unikátní jména lze vytvořit alias nazývaný jako „well-known bus name“. Například pro přístup ke keyringu je používaný alias „org.freedesktop.secret“. „Object path“ poté identifikuje danou funkci zpřístupněnou pro D-Bus. Komunikace klienta s danou službou poté funguje velmi podobně jako zjišťování IP adresy z doménového jména pomocí DNS. Klient započne komunikaci dotazem na přeložení „well-known bus name“. v odpovědi dostane unikátní „bus name“, který může využít pro komunikaci s cílovým procesem.[4]

3.4.8 Keyring

Pro ukládání hesel nebo tokenů pro aktuálně přihlášeného uživatele v systému lze využít službu pojmenovanou keyring. Podle implementace se heslo uloží na disk nebo je jen načtené v operační paměti. Pokud je uživatel přihlášený, je mu heslo k dispozici, při vypnutí systému nebo odhlášení uživatele je heslo z operační paměti odstraněno. Ukládá-li se heslo na disk, je zašifrováno a odšifrovat ho může jen uživatel, který heslo do keyringu uložil. Pokud není heslo k dispozici musí jej aplikace od uživatele vyžádat a do keyringu případně uložit.[14]

GNU/Linux obsahuje keyring aplikaci pojmenovanou linux-keyring. Některé desktopová prostředí obsahují své keyring aplikace, které rozšiřují možnosti základního linux-keyringu. Například desktopové prostředí GNOME obsahuje aplikaci gnome-keyring a desktopové prostředí KDE zase aplikaci kde-wallet.[14]

Kapitola 4

Popis implementace a nasazení aplikace

- Seznam použitých knihoven

4.1 Definice API

4.2 Mock server

4.3 Vyžívané cesty/soubory v Linuxové adresářové struktuře

4.3.1 Konfigurační soubor

4.3.2 SQLite databáze

4.4 Autentifikace

4.5 Práce se soubory

4.5.1 Otevření souboru v příslušné aplikaci

[21]

4.6 Nasazení aplikace

4.6.1 DPKG

4.6.2 RPM

4.6.3 Sestavení ze zdrojových souborů

Kapitola 5

Testovací sada

- Testy pro funkce třídy API

5.1 Google Test

Kapitola 6

Závěr

Literatura

[1] Bocchi, E., Drago, I. a Mellia, M. Personal Cloud Storage Benchmarks and Comparison. *IEEE Transactions on Cloud Computing*. 2017, sv. 5, č. 4, s. 751–764. DOI: 10.1109/TCC.2015.2427191. Dostupné z: https://ieeexplore.ieee.org/abstract/document/7096995.

[2] Burihabwa, D., Felber, P., Mercier, H. a Schiavoni, V. SGX-FS: Hardening a File System in User-Space with Intel SGX. In: *2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. 2018, s. 67–72. Dostupné z: https://ieeexplore.ieee.org/document/8590996.

[3] Carlucci, G., De Cicco, L. a Mascolo, S. HTTP over UDP: An Experimental Investigation of QUIC. In: *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. Association for Computing Machinery, 2015, s. 609–614. DOI: 10.1145/2695664.2695706. ISBN 9781450331968. Dostupné z: https://doi.org/10.1145/2695664.2695706.

[4] Cocagne, T. *DBus Overview* [online]. pythonhosted.org, červenec 2012 [cit. 2021-05-05]. Dostupné z: https://pythonhosted.org/txdbus/dbus_overview.html.

[5] *Co je to svobodný software?* [online]. gnu.org, říjen 2019 [cit. 2021-05-03]. Dostupné z: https://www.gnu.org/philosophy/free-sw.html.

[6] *Deploy Google Drive for desktop* [online]. Google, 2021 [cit. 2021-04-21]. Dostupné z: https://support.google.com/a/answer/7491144?hl=en.

[7] *Dropbox* [online]. Dropbox, 2021 [cit. 2021-04-21]. Dostupné z: https://www.dropbox.com/?landing=dbv2.

[8] Fielding, R. a Reschke, J. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content* [Requests for Comments]. RFC Editor, červen 2014 [cit. 2021-04-25]. DOI: 10.17487/RFC7231. Dostupné z: https://www.rfc-editor.org/info/rfc7231.

[9] *Google Drive for developers - API Reference* [online]. Google, 2021 [cit. 2021-04-21]. Dostupné z: https://developers.google.com/drive/api/v3/reference.

[10] *Google Workspace* [online]. Google, 2021 [cit. 2021-04-21]. Dostupné z: https://workspace.google.com/.

[11] *History of C* [online]. cppreference.com, březen 2021 [cit. 2021-05-04]. Dostupné z: https://en.cppreference.com/w/c/language/history.

[12] *History of C++* [online]. cppreference.com, březen 2021 [cit. 2021-05-04]. Dostupné z: https://en.cppreference.com/w/cpp/language/history.

[13] Jones, M. *Anatomy of the Linux kernel* [online]. IBM, červen 2007 [cit. 2021-05-06]. Dostupné z: https://developer.ibm.com/technologies/linux/articles/l-linux-kernel/.

[14] *What is: Linux keyring, gnome-keyring, Secret Service, and D-Bus* [online]. RTFM: Linux, DevOps, and system administration, 12. července 2019 [cit. 2021-05-02]. Dostupné z: https://rtfm.co.ua/en/what-is-linux-keyring-gnome-keyring-secret-service-and-d-bus/.

[15] Meszaros, G. *XUnit Test Patterns*. Addison-Wesley, 2007. ISBN 978-0-13-149505-0.

[16] *OpenAPI Specification* [online]. SmartBear Software, 2021 [cit. 2021-05-05]. Dostupné z: https://swagger.io/specification/.

[17] *PCloud* [online]. pCloud, 2021 [cit. 2021-04-21]. Dostupné z: https://www.pcloud.com/eu.

[18] Stallman, R. *Linux a systém GNU* [online]. gnu.org, červenec 2020 [cit. 2021-05-03]. Dostupné z: https://www.gnu.org/gnu/linux-and-gnu.cs.html.

[19] *Syncthing* [online]. Syncthing, 2021 [cit. 2021-04-21]. Dostupné z: https://syncthing.net/.

[20] Vangoor, B. K. R., Tarasov, V. a Zadok, E. To FUSE or Not to FUSE: Performance of User-Space File Systems. In: *15th USENIX Conference on File and Storage Technologies (FAST 17)*. 2017, s. 59–72. Dostupné z: https://www.usenix.org/conference/fast17/technical-sessions/presentation/vangoor.

[21] *Xdg-utils* [online]. freedesktop.org, červen 2019 [cit. 2021-04-20]. Dostupné z: https://freedesktop.org/wiki/Software/xdg-utils/.

[22] Yalta, A. T. a Lucchetti, R. The GNU/Linux platform and freedom respecting software for economists. *Journal of Applied Econometrics*. 2008, s. 279–286. DOI: https://doi.org/10.1002/jae.990. Dostupné z: https://onlinelibrary.wiley.com/doi/abs/10.1002/jae.990.

Příloha A

Obsah přiloženého média

Složka 1 - XXX

Složka 2 - XXX

Složka 3 - XXX