Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Použití cloudových úložišť v mobilních aplikacích

Plzeň 2014 Jaromír Staněk

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 26. června 2014

Jaromír Staněk

Abstract

Use of cloud storage in mobile applications

This thesis focuses on cloud storage services and their use in desktop and mobile applications. First part of the thesis is devoted to a brief introduction of cloud computing and its categories. This part is followed by an overview of selected cloud storage services with a particular focus on limitations in the use of the services and existing applications for accessing them. Next part studies the API provided for accessing the services. Based on the theoretical part a new library is proposed and implemented. As a part of the library design new functionality is proposed. With the use of the library two applications are then developed. First application is a desktop application pro accessing files in the cloud storages while the second one is a mobile application.

Obsah

1 Úvod

Mezi jeden z fenoménů dnešní doby, které se v oblasti informačních technologií často skloňují, patří tzv. *cloudy*. Zde se pod tímto pojem označují služby dostupné prostřednictvím sítě Internet a je možné je dále dělit do kategorií, které jsou popsány v následující kapitole. Jednou ze služeb poskytovaných v cloudu jsou i cloudová úložiště, ve kterých je možno uchovávat soubory uživatele. Jedná se o veřejné služby, které jsou často poskytovány bezplatně.

Tato diplomová práce se zaměřuje na cloudová úložiště a možnosti jejich využití na desktopové i mobilní platformě. Jedním z cílů práce je prozkoumat vybraná cloudová úložiště, jejich vlastnosti a analyzovat principy jejich fungování. Součástí analýzy je i prozkoumání již existujících knihoven a aplikací pracujících s cloudovými úložišti. Pro vybraná úložiště pak bude vytvořena aplikace, která bude schopna k těmto úložištím přistupovat a spravovat data v nich uložená.

Dalším cílem je rozšířit základní funkcionalitu o funkce usnadňující práci s uloženými daty. Jako usnadnění je zde uvažován stav, kdy uživatel je schopen docílit stejného výsledku s nižším počtem klientských aplikací, za kratší dobu nebo vykonáním nižšího počtu úkonů. Mezi takové funkce se řadí například synchronizace složek, nahrávání souborů na více úložišť zároveň nebo transfer souborů mezi úložišti. Výsledné aplikace by se tak měly svou funkcionalitou podobat již existujícím řešením a přinášet nad rámec existujících aplikací novou funkcionalitu.

2 Cloudové služby

Ve spojitosti s informačními technologiemi (IT) se stále častěji skloňuje slovo *cloud*. V první řadě je důležité tento pojem přesněji definovat a následně zavést pojmy s ním spojené. Poté je možno zavést rozdělení cloudu a určit umístění cloudových úložišť v rámci tohoto rozdělení.

Doslovný překlad slova *cloud* zní „oblak“. V počítačové terminologii se tímto pojmem běžně označuje velké seskupení objektů, které zdánlivě tvoří jednotný celek, přičemž jednotlivé prvky tohoto seskupení nejsou blíže zkoumány. Ve schématických nákresech jsou takováto seskupení znázorněna právě oblakem [1].

Základním pojmem z oblasti cloudů je *cloud computing*. Jedná se o model zpřístupnění a poskytování výpočetního výkonu uživatelům formou služeb. Základní charakteristiky tohoto přístupu jsou následující [1, 3]:

- **Služba na požádání** – uživatel platí pouze za to, co opravdu využívá a své požadavky může měnit bez nutnosti zásahu ze strany poskytovatele služeb.
- **Škálovatelnost a elasticita** – umožňuje dynamicky přidělovat a odebírat zdroje dle aktuálních potřeb uživatelů.
- **Sdílení zdrojů a nákladů** – rovněž označováno jako „multitenancy“, kde více uživatelů mezi sebou sdílí jak zdroje, tak i náklady s nimi spojené.
- **Monitorování výkonu** – automatická správa a optimalizace využívání zdrojů, transparentní monitorování služby.
- **Dostupnost** – ke službě je možno přistupovat z jakéhokoliv místa prostřednictvím Internetu.

Výhodami používání cloudových služeb je především odstínění uživatele od problémů spojených s provozem služby, přístup odkudkoliv, odolnost proti ztrátě dat a rychlé přizpůsobení potřebám uživatele. Mezi nevýhody pak patří, že uživatel neví, co se s jeho daty děje a kde jsou fyzicky uložena, dále také závislost na poskytovateli služby a připojení k síti Internet.

2.1 Rozdělení cloudových služeb

Služby v cloudu je možno rozdělit do několika kategorií v závislosti na úhlu pohledu. Při pohledu na model poskytovaných služeb, se uplatňuje následující dělení [1]:

- **Infrastruktura jako služba (IaaS)** – v tomto případě je poskytovatelem poskytován přímo hardware či jeho virtualizace. Poskytovatelů na této úrovni není z pravidla mnoho a služba je určena převážně pro společnosti, které potřebují rychle spustit nový hardware a nechtějí řešit problémy s ním spojené.
- **Platforma jako služba (PaaS)** – poskytována je předem připravená platforma, na které je možno vyvíjet a provozovat aplikace uživatele. Výhodou pro uživatele je odstínění od problémů spojených s provozem platformy.
- **Software jako služba (SaaS)** – poskytovatel poskytuje již hotový software. Služba na této úrovni je zaměřena především na koncové uživatele, kteří přistupují přímo k softwaru a jsou tak odstíněni od implementačních detailů.

Toto je pouze základní dělení, které bývá často rozšířeno o další kategorie. Jednou z rozšiřujících kategorií může být například *Storage as a service*, kde poskytovanou službou je právě cloudové úložiště [2]. Mezinárodní telekomunikační unií (ITU) jsou pak zavedeny ještě následující dvě kategorie: *Communications as a Service*, kde poskytovanou službou je komunikace v reálném čase, a *Network as a Service*, kde je poskytována konektivita [3].

Druhým pohledem na cloudové služby je jejich rozdělení podle způsobu nasazení:

- **Privátní cloud** – služba je provozována pouze v rámci organizace, ať už organizací samotnou, nebo externím dodavatelem.
- **Veřejný cloud** – jedná se o model, kdy je služba nabídnuta široké veřejnosti a tudíž k ní muže přistupovat kdokoliv.

- **Komunitní cloud** – model, kdy je služba sdílena mezi několika organizacemi či v omezené skupině uživatelů.
- **Hybridní cloud** – jedná se o kombinaci dvou a více cloudů (veřejných, komunitních a privátních), které jsou mezi sebou propojeny.

Tato práce je zaměřena na cloudová úložiště, která spadají do kategorie veřejných cloudů. Cloudová úložiště je rovněž možno klasifikovat jako služby, u kterých je poskytován již hotový software, do kterého nemá uživatel možnost zasahovat (SaaS).

3 Cloudová úložiště

V této kapitole jsou představena vybraná cloudová úložiště, se kterými se v dalších částech práce pracuje. Zaměření kapitoly je především na základní vlastnosti jednotlivých úložišť, na možnosti přístupu k úložišti pomocí aplikací třetích stran a na podporu tvorby takových aplikací ze strany poskytovatelů úložišť.

3.1 Vybraná úložiště

Pro účely této práce byla vybrána pouze některá z dostupných úložišť. Výběr úložišť byl proveden na základě popularity úložišť a s ohledem na dostupnost a podporu rozhraní pro programování aplikací (API), případně balíku nástrojů pro vývoj aplikací (SDK). Dalším faktorem, který ovlivnil výběr, bylo i to, že u vybraných úložišť je využíváno stejného způsobu pro autorizaci aplikace a výměnu informací. Ze široké množiny úložišť byla vybrána tato úložiště:

- **Dropbox** od společnosti Dropbox, Inc.,
- **Google Drive** od společnosti Google Inc.,
- **OneDrive** (původně SkyDrive) od společnosti Microsoft.

Pro úplnost je nutno ještě uvést další úložiště, která byla zkoumána, ale nebyla nakonec vybrána. Mezi hlavní důvody pro vyřazení úložiště patří například nedostupnost API či SDK, nedostupnost API či SDK pro zvolenou platformu a programovací jazyk, proprietární řešení autorizace, atp. Mezi zkoumaná a vyřazená úložiště patří například iCloud od společnosti Apple Inc., SugarSync od společnosti SugarSync, Inc., Box od společnosti Box, Inc., Amazon Cloud Drive od společnosti Amazon.com, Inc., Ubuntu One od společnosti Canonical Ltd., a další.

3.2 Srovnání úložišť

Každé z cloudových úložišť je implementováno jinak, liší se způsobem poskytování služeb, způsobem přístupu ke službě, velikostí poskytovaného prostoru a dalšími parametry. Následuje pohled na základní parametry úložišť, jejichž souhrn je následně zachycen v tabulce 3.2.

3.2.1 Licenční ujednání

Podmínkou pro užívání služeb cloudových úložišť je vždy vytvoření uživatelského účtu. Nedílnou součástí procesu vytváření uživatelského účtu je i odsouhlasení licenčních podmínek. Licenční podmínky všech úložišť si jsou velmi podobné. Mezi body, které je vhodné zmínit, patří například to, že podmínky užívání služby se mohou kdykoliv změnit. Dalším důležitým bodem je, že ačkoliv uživatel zůstává vlastníkem obsahu, může poskytovatel služby do jeho obsahu zasahovat. Významné je také, že poskytovatel nepřebírá zodpovědnost za obsah [4, 5, 6].

3.2.2 Kapacita úložišť

Důležitým parametrem, na který je třeba se zaměřit u jednotlivých úložišť, je velikost dostupného prostoru pro soubory a poplatky spojené s poskytováním tohoto prostoru. Všechna vybraná úložiště poskytují prostor zdarma, který je možno za určitých podmínek rozšířit, ať se jedná o bezplatné navýšení prostoru na základě nějaké akce, nebo rozšíření prostoru spojené s finančním poplatkem. Dalším důležitým aspektem poskytování prostoru je způsob, kterým je onen prostor využíván. Každé z vybraných úložišť využívá rezervovaný prostor odlišně.

Dropbox počítá do celkového obsazeného prostoru i obsah sdílených složek. Pokud tedy uživatelé mezi sebou sdílí nějaký obsah, započítává se jeho velikost do obsazeného prostoru všech těchto uživatelů [7].

Google Drive vyhrazený prostor sdílí i mezi další své služby. Obsazený prostor tak tvoří soubory z úložiště, emaily a uložené konverzace služby GMail a fotografie aplikace Google+, které jsou větší než 2048×2048 pixelů [8].

Pro OneDrive nebyla zjištěna žádná neobvyklost ve využívání dostupného prostoru či výpočtu jeho obsazení. V tomto směru funguje vše dle očekávání a využitý prostor odpovídá součtu velikostí všech souborů v úložišti.

3.2.3 Limit velikosti souborů

Každé z úložišť má jiné nastavení limitů pro soubory. Limity se týkají především maximální velikosti souboru v úložišti, maximální velikosti souboru při nahrávání přes klientskou aplikaci a webové rozhraní, nebo také omezení platná pro jednotlivé typy souborů.

Pro Dropbox platí, že maximální velikost souboru nahraného pomocí desktopové nebo mobilní aplikace není omezena. Stejně tak není omezena ani maximální velikost souboru v úložišti. Omezení platí pouze pro maximální velikost souboru nahraného přes webové rozhraní, které je nastaveno na 10 GB [9].

U Dropboxu nejsou zavedeny žádné limity pro jednotlivé typy souborů. Jediným limitem v tomto směru je možnost nastavení omezení přístupu aplikace pouze k určitým kategoriím souborů přes API. Mezi tyto kategorie patří textové soubory, dokumenty, obrázky, videa, hudba a elektronické knihy, a pro každou kategorii je definována množina koncovek souborů, podle kterých jsou jednotlivé soubory do kategorií přiřazovány [10].

Google Drive má maximální velikost souboru v úložišti omezenou na 1 TB. Větší soubory není možné v úložišti zobrazit. Pro soubory v Google Docs formátu platí zvláštní omezení. Dokumenty mohou obsahovat maximálně 1 024 000 znaků a nahrané dokumenty konvertované do formátu Google Docs nesmí být větší než 10 MB. Pro tabulky platí omezení na 400 000 buněk a 256 sloupců. Nahrané a konvertované tabulky do Google Docs formátu nesmí být větší než 20 MB. Pro tabulky platí ještě několik dalších omezení. Posledním typem souborů s omezením jsou prezentace, pro které platí, že nesmí být větší než 50 MB. Pro dokumenty, tabulky i prezentace, které nejsou převedeny do Google Docs formátu, platí maximální velikost souboru 1 TB [11].

Maximální velikost souboru pro úložiště OneDrive je 2 GB. Toto omezení platí i pro nahrávání souborů, ať už přes webové rozhraní, nebo přes klientskou aplikaci. Jednotlivé typy souborů nejsou na tomto úložišti zvlášť limitovány [12].

3.2.4 Autorizace přístupu

Pro vývoj aplikací je rovněž důležité znát způsob, jakým je aplikaci povoleno přistupovat k souborům uživatele. Pro přístup ke všem vybraným úložištím je využito OAuth 2.0 frameworku. Tento framework definuje metody, kterými je možno autorizovat aplikaci pro přístup k datům uživatele, a je popsán v dokumentu RFC 6749 vydaného pod IETF. Verze OAuth 2.0 rovněž nahrazuje předchozí verzi OAuth 1.0, se kterou není zpětně kompatibilní [13, 14, 15].

Autorizační framework je využíván hned z několika důvodů. Jedním z těchto důvodů je zjednodušení přístupu k uloženým datům, zvláště pak pokud se jedná o opakované přístupy, mezi kterými je delší časový odstup. Následuje několik případů, které jsou autorizačním frameworkem řešeny.

Přihlašování uživatele

Máme úložiště, ke kterému chceme přistupovat. Po přihlášení se pomocí uživatelského jména a hesla můžeme k úložišti přistupovat po dobu platnosti vytvořeného sezení. Po jeho vypršení se musíme znovu přihlásit. Pokud ale nechceme zadávat přihlašovací údaje opakovaně, musíme je někde uložit.

V tomto bodě nastává problém, protože k uloženým údajům se potenciálně může dostat neoprávněná osoba. Navíc při změně přihlašovacích údajů bude opět nutné tyto údaje zadat do aplikace a uložit. Oba tyto problémy řeší OAuth 2.0. Přihlašovací údaje uživatele jsou potřeba jen během procesu autorizace, kterým je ve většině případů nutné projít pouze jednou. Po úspěšné autorizaci je aplikaci přidělen přístupový token, který je uložen a využíván pro opakovaný přístup. Spolu s přístupovým tokenem může být aplikaci přidělen i obnovovací token, který slouží pro získání nového přístupového tokenu po vypršení jeho platnosti.

Ukládání přihlašovacích údajů

Dalším bezpečnostním rizikem je samotná aplikace. Uživatel nemusí aplikaci důvěřovat a nechce tudíž skrze aplikaci předávat přihlašovací údaje vzdálenému serveru. Rizikem v tomto případě je, že aplikace může přihlašovací údaje zachytit a zneužít.

Při použití OAuth 2.0 frameworku aplikace v ideálním případě vůbec nepřijde do styku s citlivými údaji uživatele. Doporučeným krokem autorizač-

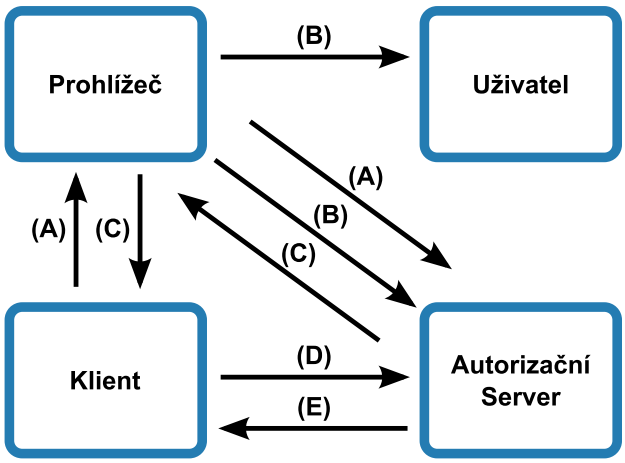
ního procesu je autentizace uživatele skrze aplikaci, které uživatel důvěřuje, nejčastěji výchozí webový prohlížeč uživatele.

Veškerá bezpečnostní rizika spojená s autorizací aplikace jsou popsána v sekci 10 dokumentu RFC 6749 [13].

OAuth 2.0 framework zavádí 4 základní způsoby získání přístupu k datům uživatele a možnost rozšířit tuto množinu o další způsoby. Z této množiny jsou pro další části práce důležité pouze dva způsoby získání přístupu, a to sice *Authorization Code Grant* a *Implicit Grant*. Tyto dva způsoby autorizace jsou totiž podporovány všemi vybranými úložišti.

I. Authorization code grant

Tento autorizační grant slouží pro získání jak přístupového, tak i obnovovacího tokenu. Tento způsob získání přístupu je vhodný pro klientské aplikace, které jsou provozovány na straně uživatele. Na obrázku 3.1 je zachycen průběh získávání tokenů.



Obrázek 3.1: Authorization code grant.

Průběh získávání tokenů je rozdělen do několika kroků. Kroky (A), (B) a (C) jsou rozděleny do dvou částí z důvodu jejich průchodu prohlížečem.

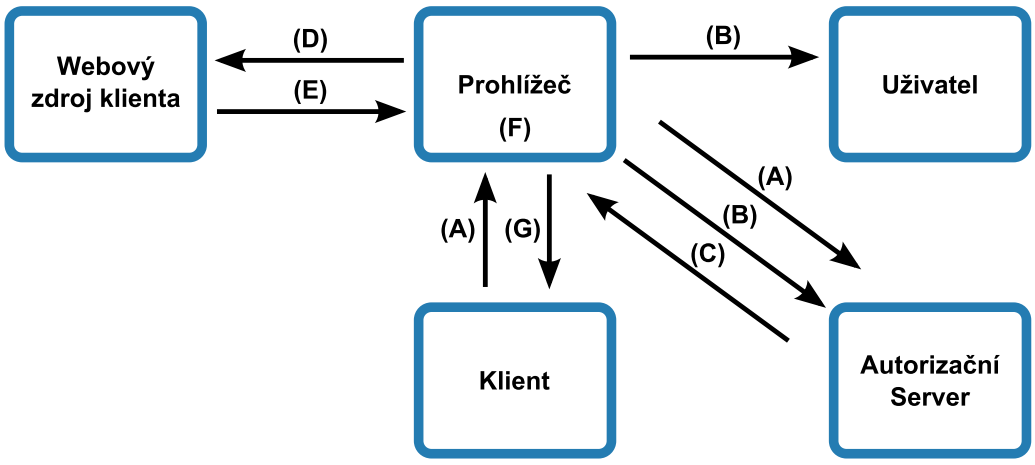
- **(A)** – Proces je započat tím, že klient sestaví požadavek na autorizaci, který vyšle skrz uživatelův prohlížeč na autorizační server. Součástí tohoto požadavku je identifikátor klienta, požadovaný rozsah přístupu,

stavová proměnná a jednotný identifikátor zdroje (URI), na který má být uživatel přesměrován po dokončení procesu autorizace.

- **(B)** – Autorizační server ověří identitu uživatele skrz prohlížeč a nabídne mu možnost požadavek klienta na autorizaci potvrdit nebo zamítnout.
- **(C)** – Autorizační server přesměruje prohlížeč zpět na klienta. Součástí URI, na kterou je prohlížeč přesměrován, je i autorizační kód a stavová proměnná, která byla zadána na začátku procesu.
- **(D)** – Klient pošle požadavek na získání tokenů autorizačnímu serveru. Součástí požadavku je i autorizační kód získaný z předchozího bodu a URI pro přesměrování, která byla použita pro získání autorizačního kódu.
- **(E)** – Autorizační server verifikuje požadavek a pokud je v pořádku, vrátí klientovi přístupový a případně i obnovovací token.

II. Implicit grant

Tento autorizační grant slouží pouze pro získávání přístupového tokenu a je optimalizován především pro JavaScriptové klienty spouštěné přímo v prohlížeči. Oproti předchozímu způsobu se liší především tím, že klient neposílá žádný oddělený požadavek na získání přístupového tokenu. Namísto toho je přístupový token součástí URI, na kterou je prohlížeč přesměrován.



Obrázek 3.2: Implicit grant.

Na obrázku 3.2 je zachycen průběh získávání přístupového tokenu. Kroky (A) a (B) jsou rozděleny do dvou částí z důvodu jejich průchodu prohlížečem.

- **(A)** – Proces je opět započat tím, že klient sestaví požadavek na autorizaci, který vyšle skrz uživatelův prohlížeč na autorizační server. Součástí tohoto požadavku je identifikátor klienta, požadovaný rozsah přístupu, stavová proměnná a URI pro přesměrování po dokončení procesu autorizace.
- **(B)** – Autorizační server ověří identitu uživatele skrz prohlížeč a nabídne mu možnost požadavek klienta na autorizaci potvrdit nebo zamítnout.
- **(C)** – Autorizační server přesměruje prohlížeč zpět na klienta. Součástí fragmentu URI je i přístupový token.
- **(D)** – Prohlížeč pokračuje v přesměrování tím, že vyšle požadavek na webový zdroj klienta, ovšem bez fragmentu obsahující přístupový token.
- **(E)** – Od webového zdroje klienta je navrácena webová stránka se skriptem, který je schopen přístupu k URI fragmentu.
- **(F)** – Prohlížeč vykoná získaný skript a extrahuje tak přístupový token z URI fragmentu.
- **(G)** – Prohlížeč pošle získaný přístupový token klientovi.

Všechna vybraná úložiště podporují oba dva popsané způsoby získání přístupového tokenu OAuth 2.0 frameworku. Získaný token je pak nutné při každém požadavku na úložiště předávat. To je možno provést třemi způsoby, buď vložením autorizační hlavičky do zasílaného požadavku, vložením přístupového tokenu jako parametru do těla požadavku, nebo vložením jako parametru přímo do URI požadavku. Způsoby použití přístupového tokenu jsou podrobněji popsány v RFC 6750 [16].

3.2.5 Způsob výměny informací

Architekturou pro výměnu dat mezi klientem a cloudovým úložištěm je representational state transfer (REST). Jedná se o datově orientovanou

architekturu určenou pro přístup ke zdrojům umístěným v distribuovaném prostředí. Mezi vlastnosti architektury patří následující [17, 18]:

- **Model klient – server** – Oddělení prezentační vrstvy od datové, kde klient má na starost uživatelské rozhraní, zatímco server pracuje s datovou vrstvou.
- **Bezestavovost** – Požadavky musí obsahovat kompletní sadu parametrů potřebných pro jejich vykonání. V případě nutnosti uchovávat stavové informace se tak děje na straně klienta.
- **Cacheable** – Možnost ukládat výsledky některých požadavků pro zvýšení výkonnosti systému.
- **Vrstvený systém** – Server nemusí být konečným serverem, ale může sloužit jako prostředník mezi klientem a dalším serverem.
- **Code on demand** – Možnost rozšíření funkcionality klienta pomocí kódu zaslaného serverem.
- **Jednotné rozhraní** – Jednotný přístup pro manipulaci s daty, který využívá URI pro identifikaci zdrojů. Navíc přenášené zprávy musí obsahovat dostatek informací pro zpracování zprávy.

Množina operací pro manipulaci s daty je *create*, *read*, *update* a *delete* (CRUD). Tato množina je konečná a všechny operace jsou mapovány na metody Hypertext Transfer Protokolu (HTTP), který je používán jako protokol pro výměnu zpráv. V tabulce 3.1 je mapování operací CRUD na metody HTTP [44].

Operace CRUD	**HTTP metody**	**Popis operace**
Create	POST	Vložení zdroje.
Read	GET	Získání zdroje.
Update	PUT	Modifikace zdroje.
Delete	DELETE	Smazání zdroje.

Tabulka 3.1: Mapování CRUD operací na HTTP metody.

Systém, který dodržuje všechny principy REST architektury je možné označit jako RESTful. U všech vybraných úložišť je možno využít přímého přístupu k datům pomocí CRUD operací REST architektury. Přímý přístup je podrobněji popsán v sekci 3.3.4.

Pro výměnu dat mezi klientem a cloudovým úložištěm je možno využít různých formátů dat. Formátem využívaným všemi vybranými úložišti je JavaScript Object Notation (JSON). Jedná se o textový formát dat nezávislý na programovacím jazyce. Tento formát je v současné době popsán dvěma standardy, RFC 7159 [20] a ECMA-404 [21]. Standardy ECMA jsou vydávány organizací Ecma International, která se zabývá tvorbou standardů pro informační a komunikační systémy. Zkratka ECMA značí European Computer Manufacturers Association.

Hlavními stavebními kameny formátu JSON jsou dvojice klíč – hodnota, které jsou používány k definici objektů, a pole hodnot. Formát JSON definuje 6 základních typů hodnot [19].

- **object** – Jedná se množinu dvojic klíč – hodnota, která je uzavřena do složených závorek (`{` a `}`). Jednotlivé páry jsou odděleny čárkou a oddělení klíče od hodnoty je provedeno pomocí dvojtečky. Klíče jsou tvořeny řetězcem.
- **array** – Pole hodnot je uzavřeno hranatými závorkami (`[` a `]`) a jednotlivé hodnoty jsou mezi sebou odděleny čárkou.
- **string** – Řetězec znaků je uzavřen do uvozovek (`"`) a nesmí obsahovat znak uvozovek a zpětné lomítko, které slouží jako uvozovací znak řídící sekvence uvnitř řetězce.
- **number** – Obecná číselná hodnota.
- **boolean** – Logická hodnota `true` nebo `false`.
- **null** – Prázdná hodnota.

Pomocí těchto základních hodnot je možno definovat prakticky jakoukoliv datovou strukturu. Výhodou formátu JSON oproti jiným formátům běžně používaným je především jednoduchost, čitelnost a nízká míra režie dat. Následuje ukázka zápisu dat ve formátu XML (kód 3.1) a stejných dat ve formátu JSON (kód 3.2). Extensible markup language (XML) je zde vybrán především jako ukázka alternativního formátu pro výměnu dat.

```xml
<person id="1">
    <firstName>Philip</firstName>
    <lastName>Black</lastName>
    <address>
        <street>Somestreet</street>
        <houseNumber>123</houseNumber>
        <city>Sometown</city>
    </address>
    <phone>
        <phoneNumber type="mobile">123456789</phoneNumber>
        <phoneNumber type="home">987654321</phoneNumber>
    </phone>
</person>
```

Kód 3.1: Ukázka dat ve formátu XML.

```json
{
    "id":1,
    "name":"Philip Black",
    "address":
    {
        "street":"Somestreet",
        "house_number":123,
        "city":"Sometown"
    },
    "phone":
    [
        {
            "type":"mobile",
            "number":123456789
        },
        {
            "type":"home",
            "number":987654321
        }
    ]
}
```

Kód 3.2: Ukázka dat ve formátu JSON.

	Dropbox	**Google Drive**	**OneDrive**
Velikost prostoru zdarma	2 GB	15 GB	7 GB
Maximální velikost souboru v úložišti	neomezená*	1 TB	2 GB
Maximální velikost souboru nahraného aplikací	neomezená*	1 TB	2 GB
Maximální velikost souboru nahraného přes webové rozhraní	10 GB	1 TB	2 GB
Autorizace	OAuth 1.0, 2.0	OAuth 2.0	OAuth 2.0
Komunikační architektura	REST	REST	REST
Komunikační protokol	HTTP	HTTP	HTTP
Formát zpráv	JSON	JSON	JSON

*) Soubor nemůže být větší než dostupná kapacita úložiště.

Tabulka 3.2: Srovnání cloudových úložišť.

3.3 Dostupná API a SDK

Každé z vybraných úložišť nabízí vývojářům vlastní API nebo SDK pro vývoj aplikací. API či SDK jsou ve všech případech dostupné pro různé programovací jazyky a různé platformy. Dále pak existují API a SDK třetích stran, které je pro vývoj rovněž možno použít. API a SDK třetích stran jsou ve většině případů dostupné pro kombinace programovacích jazyků a platforem, které oficiální distribuce nepokrývá.

3.3.1 Základní porovnání

Pro vývoj aplikací pro úložiště Dropbox jsou dostupné knihovny pro programovací a skriptovací jazyky Python, Ruby, PHP a Java. Dále jsou dostupné knihovny pro vývoj na mobilních platformách Android a iOS, a také pro platformu OS X. Mimo tyto oficiální knihovny jsou dostupné i knihovny třetích stran pro jazyky C++, C#, JavaScript, a další [22].

Pro Google Drive jsou dostupné knihovny pro rodinu jazyků .NET, jazyky Java, JavaScript, Objective-C, PHP, Python, Go, Ruby, a další [23].

Pro OneDrive jsou dostupná SDK pro vývoj na mobilních platformách iOS, Android a Windows Phone. Dále je též podporován vývoj pro Windows a .NET [24].

3.3.2 Dostupná dokumentace

Pro každou oficiální API knihovnu nebo SDK balík je dostupná online dokumentace. Navíc jsou z oficiálních zdrojů dostupné i příklady kódu pro vybrané jazyky a platformy. Google Drive a OneDrive nad rámec běžné dokumentace ještě poskytují interaktivní webové rozhraní, pomocí kterého je možno vyzkoušet si klíčové koncepty v praxi. Google Drive toto rozhraní zobrazuje pro každou část dokumentace API, zatímco OneDrive toto interaktivní rozšíření z dokumentace vyčlenil[25].

Dropbox nabízí příklady a dokumentaci pro jednotlivé knihovny a zvlášť pak ucelenou dokumentaci popisující všechny základní koncepty implementované úložištěm [22, 27].

U úložiště Google Drive je dokumentace rozdělena do několika částí. Principy autorizace jsou popsány v části *Authorize Requests*, zatímco základní koncepty úložiště jsou v části *Developer Guide*. Část *API Reference* pak popisuje jednotlivé části API, které je možno si prostřednictvím interaktivního rozhraní i vyzkoušet [23, 28].

Úložiště OneDrive má rovněž rozdělenou dokumentaci do více částí. Základní principy pro přístup k úložišti [26], dokumentace API, interaktivní SDK a ukázkové příklady jsou tak odděleny. Některé části dokumentace jsou duplikovány na více místech. Toto platí například pro dokumentaci přímého přístupu k úložišti pomocí RESTu, která je popsána jednak jako součást dokumentace API [29], a pak také zvlášť na několika místech [30, 31].

3.3.3 Náhled do obsahu knihoven

Knihovny pro přístup k úložištím je možno dekomponovat do několika elementárních částí. Pro tuto práci mají velký význam především části pro

autorizaci a přenos zpráv a souborů.

Při bližším průzkumu knihoven pro Dropbox byly zkoumány verze 1.5.3 a 1.7.6 pro Javu a verze 1.5.3 a 1.6.1 pro Android. Knihovna verze 1.5.3 pro Javu využívá autorizace pomocí OAuth 1.0, HTTP klienta od Apache [33] pro komunikaci a knihovny JSON.simple [34] pro zpracování předávaných zpráv. Oproti tomu verze 1.7.6 již podporuje autorizaci pomocí OAuth 2.0, využívá standardního zabezpečeného HTTP spojení, které je součástí Javy, pro komunikaci a knihovny Jackson [35] pro zpracování zpráv.

U verzí knihoven pro Android došlo rovněž k přechodu od autorizace pomocí OAuth 1.0 k OAuth 2.0, kde OAuth 1.0 zůstává nadále podporován. Pro komunikaci slouží HTTP klient od Apache [33] a pro zpracování zpráv knihovna JSON.simple [34].

Pro Google Drive je dostupná jednotná Java knihovna pro desktopové aplikace i platformu Android. Zkoumána byla verze knihovny 1.18.0-rc. Obsahem této knihovny je mnoho knihoven, které zajišťují kompatibilitu knihovny s různými platformami. Knihovnu je tak možno využít nejen pro Javu na desktopu a Android, ale i pro Google App Engine a pro servletové aplikace. Pro zpracování zpráv je možno využít buď knihovny GSON, která je implementací JSON formátu od společnosti Google, knihovnu Jackson nebo Java Data Objects (JDO), které jsou součástí Javy. Pro komunikaci je možno využít různých HTTP klient v závislosti na používané platformě. K dispozici je například HTTP klient od Apache nebo HTTP klient, který je součástí Javy.

Pro OneDrive je pro vývoj desktopových aplikací dostupná pouze knihovna v jazyce C#. Pro platformu Android byla zkoumána knihovna ve verzi 5.5, která využívá pro komunikaci HTTP klienta od Apache, který je součástí platformy Android. Pro zpracování zpráv je rovněž využito dostupných prostředků platformy, kterou je výchozí implementace org.json [19]. Pohled na knihovnu pro C# je zajímavý z hlediska autorizačního procesu, během kterého je oproti ostatním řešením využíváno webového prohlížeče, který je komponentou .NET frameworku.

Při podrobnějším pohledu na obsah knihovny je také důležité podívat se na licenční ujednání, které je z pravidla její součástí. Dropbox i OneDrive distribuují knihovnu pod vlastní licencí, zatímco Google Drive používá licenci Apache verze 2.0 [32].

3.3.4 Přímý přístup

Ke každému z vybraných úložišť je možné přistupovat i přímo bez využití některé z dostupných knihoven. Každé z úložišť má tuto možnost dobře zdokumentovanou. Pro každou operaci jsou poskytovány informace o HTTP požadavku, který má být úložišti zaslán, parametrech a těle tohoto požadavku, odpovědi serveru na tento požadavek a chybových kódech. Každé úložiště má toto rozhraní pro přístup odlišné. Pro Dropbox a Google Drive je toto rozhraní popsané v rámci dostupné dokumentace [27, 28]. OneDrive má rozhraní pro přímý přístup popsáno třemi dokumenty, z toho se každý zabývá odlišnými aspekty přístupu [29, 30, 31].

Hlavními rozdíly mezi implementacemi jsou použité metody protokolu HTTP, dále pak zasílané parametry a přijímané odpovědi od serverů. Všechna úložiště využívají standardních HTTP metod specifikovaných v RFC 2616 – `GET`, `POST`, `PUT` a `DELETE` [44]. Pro některé operace ovšem využívají různá úložiště různých HTTP metod. Google Drive navíc využívá ještě metody `PATCH`, která je definována v RFC 2068, které je již zastaralé [28, 45]. Pro OneDrive jsou definovány a využívány dvě nestandardní metody – `COPY` a `MOVE`. Jak již jejich název napovídá, slouží pro kopírování a přesun souborů v rámci úložiště [29].

3.4 Existující aplikace

Pro každé z vybraných úložišť existuje celá řada aplikací, které poskytují přístup k souborům v úložištích. Aplikace je možno rozdělit podle několika kritérií. Jedním z kritérií je vydavatel aplikace. Aplikace je tak možno rozdělit na ty, které dodává přímo poskytovatel úložiště, a ty, které jsou vydávány někým jiným (klienti třetích stran). Dalším kritériem dělení může být například platforma, pro kterou jsou aplikace určeny, nebo množina úložišť, ke kterým je možno pomocí zvolené aplikace přistupovat.

3.4.1 Nativní klienti

Jako nativní klienti jsou zde označovány klientské aplikace poskytované přímo k úložišti. Tyto aplikace pracují na principu synchronizace lokální složky. Po instalaci klienta se vytvoří v lokálním souborovém systému složka

určená pro synchronizaci. Obsah této složky je automaticky synchronizován s obsahem úložiště. Platí zde pravidlo, že obsah cloudového úložiště má přednost před obsahem lokální složky. Toto chování je patrné například v situaci, kdy vznikne konflikt mezi lokálním souborem a souborem v úložišti. Ve výsledku je zachován vzdálený soubor beze změny, zatímco lokální soubor je přejmenován.

Jednou z dalších vlastností, která je podporována u klientských aplikací poskytovaných k úložišti, je možnost výběru složek, které se synchronizují. Na lokální úložiště se tak nestahuje kompletní obsah úložiště, ale jen vybrané složky. Je nutno rovněž zmínit, že u těchto aplikací je možné se připojit pouze k jednomu účtu daného úložiště. Rovněž je možno získat klientské aplikace i pro mobilní platformy.

3.4.2 Klienti třetích stran

Existuje celá řada klientských aplikací třetích stran, které pracují s cloudovými úložišti. Některé aplikace je možno stáhnout a nainstalovat na počítač či mobilní platformu, jiné jsou koncipovány pouze jako webová služba. Výhodami těchto aplikací oproti aplikacím poskytovaných spolu s úložným prostorem je především rozšířená funkcionalita. Často se jedná právě o sdružení více úložišť pod jednu aplikaci.

Desktopové aplikace

Mezi příklady desktopových aplikací patři SME Cloud Explorer nebo CarotDAV. Druhá zmíněná aplikace je jednoduchým nástrojem pro správu souborů ve více úložištích. Z rozšiřujících funkcionalit disponuje například možností přístupu k souborům pomocí protokolu pro přenos souborů (FTP), šifrováním souborů nebo možností komprese a automatického rozdělení souboru [36].

Aplikace SME Cloud Explorer podporuje přenos souborů mezi úložišti, šifrování nebo zamykání souborů. Nad rámec tohoto je aplikace propojena s webovou službou a je možno získat i klienty pro mobilní platformy. Transfer souborů mezi úložišti je tak realizován prostřednictvím webové služby [37].

Webové služby

Dostupných webových služeb je velké množství. Zde jsou vyjmenovány pouze některé. Tyto služby mají společné hlavně to, že dokáží přistupovat k různým cloudovým úložištím. Webové služby sdružující cloudová úložiště je možné dále rozdělit podle principu jejich fungování. Existují tak služby zaměřené na správu souborů, služby orientované na dokumenty, synchronizační služby, a další. Častým prvkem je pak i propojení služby se sociálními sítěmi a webovými službami pro správu fotografií.

Mezi správce souborů patří například služby CloudKafé, Jolidrive, primadesk a MultCloud. Všechny tyto služby vyžadují registraci pro užívání. Mezi sebou se pak liší především mírou zpoplatnění a funkcemi, které nabízí. Všechny tyto služby podporují transfer souborů mezi úložišti [38, 39, 40, 41].

Služba cloudHQ je pak online synchronizačním klientem. Stejně jako u ostatních služeb, i zde je potřeba registrace před možností užívání služby. Ve službě je pak možné nastavit směr, kterým bude synchronizace probíhat [42].

Poslední zajímavou službou je IFTTT. Tato služba se od ostatních zde vyjmenovaných služeb podstatně liší. Není zde totiž nabídnuta klasická správa souborů, ale možnost vytvoření spouštěčů a přiřazení k nim nějaké akce. Touto službou je například podporováno sledování obsahu úložiště Dropbox, kde při nahrání nového souboru může být tento soubor nakopírován i na další úložiště [43].

4 Návrh knihovny

Pro implementaci jednotného přístupu k více úložištím zároveň je možno využít existující knihovny poskytované úložišti. Stejně tak je ale možné zvolit vlastní řešení a vytvořit jednotnou knihovnu pro přístup k vybraným cloudovým úložištím. V této kapitole je odůvodněna volba řešení, po které následuje analýza klíčových aspektů a návrh implementace zvoleného řešení.

4.1 Volba vlastního řešení

Po analýze dostupných knihoven pro přístup ke cloudovým úložištím bylo rozhodnuto o vytvoření vlastní knihovny sjednocující přístup. Motivací pro tuto volbu byla možnost implementace rozšiřujících funkcí na nižší úrovni, potenciální snížení velikosti knihovny, lepší možnost znovupoužitelnosti komponent knihovny, nezávislost na knihovnách dodávaných provozovateli úložišť a implementace knihovny umožňující snadné rozšíření na další cloudová úložiště. Zvolenou mobilní platformou pro implementaci je Android. Pro desktopovou aplikaci byla zvolena implementace v programovacím jazyce Java. Výstupem implementačního procesu by měla být knihovna použitelná jak pro desktopovou aplikaci, tak i pro mobilní platformu.

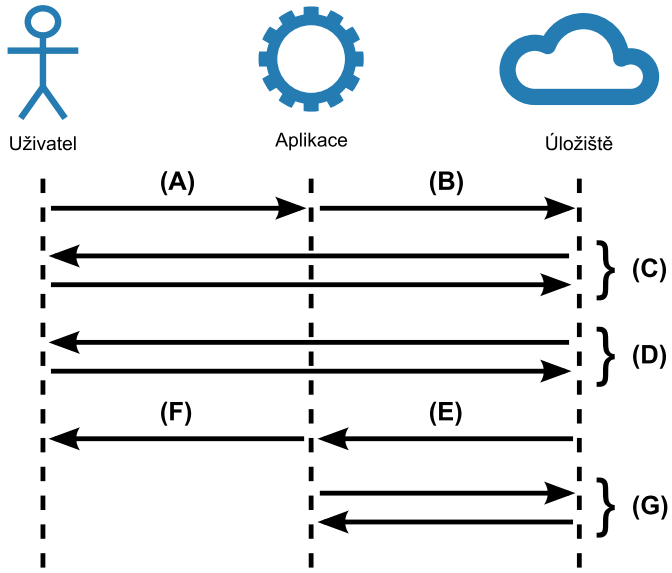
4.2 Požadovaná funkcionalita

V první řadě je potřeba aby knihovna mohla zahájit proces autorizace a získat přístupové tokeny po jeho skončení. Dalším důležitým aspektem je podpora základních operací se soubory a složkami. Cílem je tak vytvoření jednotného rozhraní pro všechna úložiště. Pod jednotné rozhraní spadá nejen jednotné volání operací nad úložišti, ale také sjednocení formátů dat získávaných z úložišť, které zahrnují metadata souborů a složek, informace o uživateli a úložišti, a chybová hlášení. Následuje návrh a implementace rozšířené funkcionality.

4.3 Proces autorizace

Pro implementaci byly zvoleny autorizační procesy *authorization code grant* a *implicit grant*. Tyto dva autorizační procesy jsou podporovány všemi vybranými úložišti, přičemž první zmíněný se využívá hlavně pro instalované klientské aplikace. Druhý zmíněný proces je využíván převážně pro webové služby. Pro vyšší míru znovupoužitelnosti výsledné knihovny byla zvolena implementace i tohoto procesu autorizace.

Proces autorizace je popsán v sekci 3.2.4. Pro maximální komfort uživatele se zachováním vysoké míry bezpečnosti jsem při návrhu vycházel ze sekce 9 dokumentu RFC 6749. Pro důvěryhodnost aplikace je tak zvoleno otevření okna výchozího prohlížeče během autorizačního procesu [13]. Pro vyšší komfort je zvoleno použití lokálního webového serveru pro odchytávání zpětného přesměrování po autorizaci aplikace. Výsledný proces pro *authorization code grant* je znázorněn na obrázku 4.1.



Obrázek 4.1: Implementace procesu pro *authorization code grant*.

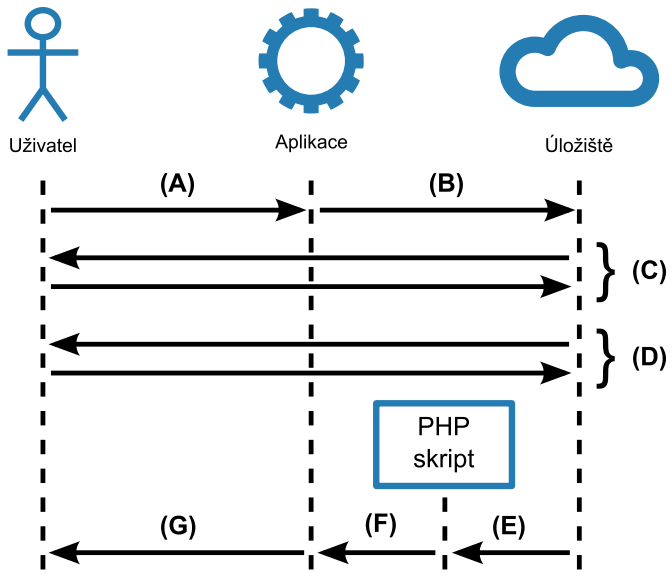
- **(A)** – Uživatel spustí proces autorizace aplikace.
- **(B)** – Aplikace přesměruje uživatele na autorizační server úložiště tím, že vyvolá nové okno uživatelova výchozího prohlížeče.
- **(C)** – Uživatel provede přihlášení k úložišti skrze okno prohlížeče, tudíž aplikace nemá přístup k přihlašovacím údajům uživatele.

- **(D)** – Uživatel provede autorizaci aplikace.
- **(E)** – Uživatel je přesměrován z webového rozhraní úložiště na adresu webového serveru aplikace.
- **(F)** – Aplikace zpracuje přijatá data od úložiště a zobrazí uživateli webovou stránku s výsledkem autorizace.
- **(G)** – V případě úspěšné autorizace provede aplikace výměnu autorizačního kódu za přístupový token.

Aby nebylo nutné rezervovat lokální port pro webový server, bylo zvoleno naslouchání na náhodném portu přiděleného systémem. Problémem v tomto bodě je především to, že podle sekce 10.6. RFC 6749 je autorizační server povinen mít zaregistrovány URI pro přesměrování [13]. Každé z vybraných úložišť se k tomuto bodu staví trochu odlišně. Dropbox povoluje lokální server, ale je nutno registrovat každý port zvlášť, Google Drive rovněž povoluje lokální server, ovšem čísla portů neřeší, a OneDrive vůbec nerozpozná adresu lokálního serveru (`localhost` ani `127.0.0.1`) a vyžaduje tak přesměrování na doménu, která je v rámci registrovaných aplikací pro OneDrive unikátní.

Pro vyvarování se problémům s autorizačním procesem bylo zvoleno přesměrování pomocí skriptu hypertextového preprocesoru (PHP) na doméně 3. řádu `https://home.zcu.cz`. Aby bylo možné požadavek úspěšně přesměrovat, je potřeba znát číslo lokálního portu, na který se má přesměrování provést. Pro vyřešení této obtíže je využito proměnné `state`, která slouží v autorizačním procesu jako prevence cross-site request forgery (CSRF) útoku popsaného v sekci 10.12. RFC 6749 [13]. Aby tato proměnná co nejlépe plnila svůj účel, je první část proměnné tvořena číslem lokálního portu, po kterém následuje pomlčka a série náhodných znaků. Při použití tohoto formátu je možné číslo portu jednoduše získat zpět.

Implementace procesu pro *implicit grant* je podobná jako pro *authorization code grant* a je zachycena na obrázku 4.2. Hlavním problémem je v tomto případě, kromě již vyřešených, extrahování přístupového tokenu z URI fragmentu. Pro tento účel poslouží opět již zmíněný PHP skript, který vrátí uživateli jednoduchou webovou stránku s JavaScriptem pro získání přístupového tokenu. Úkolem JavaScriptu je tedy získání přístupového tokenu z URI fragmentu a přesměrování uživatele na lokální webový server klienta, kterému bude přístupový token předán jako parametr URI.



Obrázek 4.2: Implementace procesu pro *implicit grant*.

- **(A)** až **(D)** – shodné s popisem procesu pro *authorization code grant*.
- **(E)** – Uživatel je přesměrován na PHP skript, který dodá uživateli JavaScript pro extrahování přístupového tokenu z URI fragmentu.
- **(F)** – Uživatel je přesměrován na adresu webového serveru aplikace, v tuto chvíli již s extrahovaným přístupovým tokenem.
- **(G)** – Aplikace zpracuje přijatá data a zobrazí uživateli webovou stránku s výsledkem autorizace.

Ostatní granty není nutné pro správné fungování aplikace implementovat. Vybraná úložiště využívají *authorization code grant* a *implicit grant* pro autorizaci. Na co je ovšem nutné se zaměřit je ukládání přístupových a obnovovacích tokenů. Podle sekcí 10.3. a 10.4. dokumentu RFC 6749, vydané tokeny musí být bezpečně přenášeny a uchovávány [13]. Přenos tokenů řeší zabezpečené spojení, ale pro uchovávání tokenů je potřeba navrhnout způsob pro jejich uložení.

Pro ukládání tokenů je potřeba zvolit nějaký způsob šifrování. Na výběr je zde šifrování se symetrickým nebo s asymetrickým klíčem. Výhodou symetrického klíče je, že stejným klíčem se provede zašifrování tokenů i jejich následné dešifrování. Pro asymetrický klíč jsou potřeba dva klíče, veřejný klíč

pro zašifrování tokenů a privátní klíč pro dešifrování. Vzhledem k tomu, že bude přistupováno k tokenům výhradně skrze aplikaci, bude lepší volbou šifrování se symetrickým klíčem. Implementace pak může buď využít předem nastavený výchozí klíč, nebo požadovat po uživateli zadání klíče (hesla) [47].

Pro šifrování se symetrickým klíčem existuje celá řada algoritmů, ze kterých je možno vybírat. Mezi nejznámější patří [47]:

- **AES** (Advanced Encryption Standard),
- **Blowfish**,
- **DES** (Data Encryption Standard) a jeho varianty,
- **Serpent**,
- **Twofish**,
- a mnoho dalších.

Hlavními kritérii volby algoritmu byla dostupnost implementace algoritmu pro Javu, nejlépe již zakomponovaná přímo v Javě samotné, a bezpečnost algoritmu. Z výše zmíněných byl nakonec zvolen algoritmus AES. Jedná se o blokovou šifru s velikostí bloku 128 bitů a délkou klíče 128, 192 nebo 256 bitů [48, 49]. Jako další bylo potřeba zvolit mód, ve kterém bude algoritmus pracovat. Na výběr byly módy *Electronic Codebook* (ECB), *Cipher block chaining* (CBC), *Output feedback mode* (OFB) a *Counter mode* (CTR), z čehož poslední dva módy vyžadují převod blokové šifry na proudovou [50, 51]. Po prostudování dostupných módů a jejich vlastností byl nakonec zvolen mód CBC. Mód CBC využívá zřetězení bloků při šifrování a odstraňuje tím nevýhody módu ECB. V obrázku 4.3 je pak znázorněno šifrování tokenů pomocí symetrického klíče.

Obrázek 4.3: Symetrické šifrování tokenů.

4.4 Metadata

Základem jednotného přístupu k úložištím je sjednocení metadat souborů a složek. Každé z úložišť popisuje soubory a složky jinými metadaty, která vrací uživateli ve formátu JSON. Prvním úkolem tedy bylo vyhledání shodných rysů metadat ze všech úložišť. K tomuto účelu slouží tabulka 4.1, která zachycuje některá vybraná metadata. Ve sloupcích pro jednotlivá úložiště je vždy zachycen název atributu a jeho datový typ (název: `typ`). Použité datové typy v tabulce 4.1 jsou mapovány na datové typy formátu JSON převodní tabulkou 4.2.

Z tabulky 4.1 vyplývá, že existuje pouze několik málo atributů, které je možno získat ze všech úložišť. Mezi tyto atributy patří název souboru či složky, velikost, obsah (pokud se jedná o složku), rodičovskou složku a datum modifikace. Ostatní atributy se vyskytují pouze u některých úložišť, což je komplikací, protože některé z těchto atributů mohou mít klíčovou roli pro prováděné operace. Navíc ještě každý z atributů může být v různých úložištích označen jiným názvem a reprezentován jiným datovým typem, což celou situaci ještě více komplikuje.

Rozdílné názvy atributů je možno vyřešit mapováním. V praxi to znamená hned dvojí mapování. První mapování musí proběhnout při získání metadat, aby bylo možné metadata uložit v jednotné podobě pro pozdější užití. Během prvního mapování je rovněž nutné mapovat do hloubky. To znamená, že pokud přijatá data ve formátu JSON obsahují pole a objekty, je nutné mapovat i atributy obsažené v těchto datových strukturách. Druhé mapování je pak zpětné mapování z uložené podoby na podobu vyžadovanou konkrétním úložištěm. Při zpětném mapování nemusí docházet pouze k mapování atributů do formátu JSON, ale rovněž na parametry URI předávané HTTP požadavkem.

Atributy, které hrají klíčovou roli při volání operací, jsou pro každé úložiště odlišné. Pro Dropbox je hlavní cesta k souboru, pomocí které je v úložišti soubor jednoznačně identifikován [27]. Pro Google Drive a OneDrive tuto roli zastává identifikátor souboru [28, 29, 30].

Jelikož je pro soubory i složky využíváno stejného formátu dat, je klíčové určit, zda se jedná právě o soubor či složku. U Dropboxu je přítomný atribut *is_dir*, který toto přímo určuje hodnotou typu `boolean` [27]. U ostatních úložišť je toto určení obtížnější. Pro OneDrive je přítomen atribut *type*, který může obsahovat jeden z řetězců `file`, `notebook`, `album`, `audio`, `folder`,

Parametr	**Dropbox**	**Google Drive**	**OneDrive**
Název	path*: `str`	title: `str`	name: `str`
Velikost	bytes: `num`	quotaBytesUsed: `num`	size: `num`
Identifikátor	———	id: `str`	id: `str`
Cesta	path: `str`	———	———

Obsah	contents:
	`[`
	`{file_obj},`
	`...`
	`]`
		items:
		`[`
		`{file_obj},`
		`...`
		`]`
			data:
			`[`
			`{file_obj},`
			`...`
			`]`

Typ	is_dir: `bool`	———	type: `str`

Smazaný	is_deleted: `bool`	labels:
		`{`
		trashed: `bool`
		`}`
			———

Sdílený	———	shared: `bool`	shared_with:
			`{`
			access: `str`
			`}`

Rodič	path**: `str`	parents:
		`[`
		`{file_obj},`
		`...`
		`]`
			parent_id: `str`

MimeType	mime_type: `str`	mimeType: `str`	———
Checksum	———	md5Checksum: `str`	———
Vytvoření	———	createdDate: `date`	created_time: `date`
Modifikace	modified: `date`	modifiedDate: `date`	updated_time: `date`

*) Název souboru či složky je možno získat parsováním cesty.

**) Rodiče souboru či složky je možno získat parsováním cesty.

Tabulka 4.1: Porovnání vybraných atributů dostupných metadat.

Typ z tabulky 4.1	Typ formátu JSON	Popis
`str`	`string`	řetězec znaků
`num`	`number`	číselná hodnota
`bool`	`boolean`	hodnota `true` nebo `false`
`date`	`string`	řetězec reprezentující datum
`[ ... ]`	`array`	pole hodnot
`{ ... }`	`object`	reprezentace objektu
`{file_obj}`	`object`	objekt metadat souboru

Tabulka 4.2: Mapování datových typů.

`photo` a `video`. Řetězce `album` a `folder` identifikují složky, hodnoty `file`, `notebook`, `audio`, `photo` a `video` pak určují, že se jedná o soubor [29].

Pro Google Drive je rozlišení složek a souborů ještě o něco složitější. Není zde totiž přítomen žádný speciální atribut a toto rozlišování se provádí na základě typů Multipurpose Internet Mail Extensions (MIME), který je přenášen atributem *mimeType*. Typ MIME, který určuje složku v úložišti Google Drive, je *application/vnd.google-apps.folder* [28].

Mezi vlastnostmi úložišť Dropbox a Google Drive je možnost přístupu ke smazaným souborům. U každého smazaného souboru nebo složky je nastaven příslušný atribut, který označuje, zda se jedná o smazaný obsah. OneDrive tuto možnost nepodporuje.

U souborů a složek úložišť Google Drive a OneDrive lze zjistit, zda jsou sdíleny s jinými uživateli. Ačkoliv i Dropbox podporuje sdílení souborů a složek mezi uživateli, není možné o sdílení nic zjistit z dostupných metadat.

4.5 Chybová hlášení

Nedílnou součástí dat získávaných od cloudových úložišť jsou i chybová hlášení. V tabulce 4.3 jsou zachyceny formáty chybových hlášení pro jednotlivá úložiště. Z této tabulky je vidět, že formáty chybových hlášení se mezi úložišti podstatně liší. Pro Dropbox je vracen pouze řetězec s popisem chyby, zatímco pro Google Drive a OneDrive je návratovou hodnotou chybový objekt (`{error_obj}`). Součástí chybového objektu je vždy kód chyby (code) a chybové hlášení (message).

Dropbox	**Google Drive**	**OneDrive**

error: `str`	error:
	`{`
	errors:
	`[`
	`{error_obj},`
	`...`
	`],`
	code: `num,`
	message: `str`
	`}`
		error:
		`{`
		code: `str,`
		message: `str`
		`}`

Tabulka 4.3: Porovnání formátů chybových hlášení.

Problémem u kódu chyby je, že Google Drive vrací číselnou hodnotu, zatímco OneDrive řetězec. Vzhledem k tomu, že kód chyby je předáván i stavovým kódem HTTP odpovědi a odpovídá hodnotě předávané v chybovém objektu, je možné jej získávat přímo z HTTP odpovědi [28, 29, 44]. Tímto způsobem se vyřeší i absence kódu chyby v odpovědi pro Dropbox. Pro vybraná úložiště je tedy vždy dostupný kód chyby a chybové hlášení.

4.6 Informace o uživateli

Posledními daty, která je potřeba sjednotit, jsou informace o uživateli a jeho účtu v rámci úložiště. Každé úložiště používá odlišný formát dat. Rozdíly jsou pak zachyceny v tabulce 4.4. Pro každé úložiště je možné získat jméno uživatele, jeho identifikátor a celkovou kapacitu prostoru dostupnou pro uživatele. Rozdílné je získávání velikosti obsazeného, resp. volného prostoru. Dropbox a Google Drive navrací velikost obsazeného prostoru, zatímco pro OneDrive je dostupná informace o volném prostoru. Z dostupných dat pak lze dopočítat velikost volného, resp. obsazeného prostoru pro všechna úložiště.

Parametr	**Dropbox**	**Google Drive**	**OneDrive**
Jméno	display_name: `str`	name: `str`	name: `str`
Identifikátor	id: `str`	permissionId: `str`	id: `str`

Celkový prostor	quota_info:
	`{`
	quota: `num`
	`}`
		quotaBytesTotal:
		`num`
			quota: `num`

Využitý prostor*	quota_info
	`{`
	normal: `num`
	shared: `num`
	`}`
		quotaBytes-
		UsedAggregate:
		`num`
			———

Volný prostor	———	———	available: `num`

*) Pro Dropbox je využitý prostor součtem *normal* (prostor využitý pro soubory uživatele) a *shared* (prostor využitý soubory s uživatelem sdílenými).

Tabulka 4.4: Formát informací o uživateli a účtu.

4.7 Množina operací

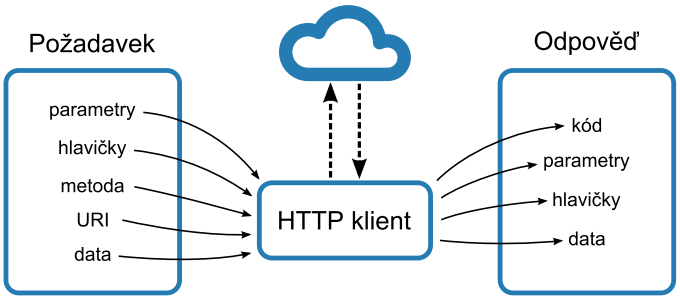
Množina operací určených pro implementaci vychází přímo z množin operací podporovaných jednotlivými úložišti. Pro spolehlivou funkcionalitu je možno implementovat pouze ty operace, které jsou podporovány všemi úložišti zároveň. Výslednou množinu operací je možno rozdělit na operace pro manipulaci se soubory a složkami, a na operace pro získání informací o uživateli a úložišti.

4.7.1 Generická operace

Pro zjednodušení implementace jednotlivých operací bylo zvoleno vytvoření abstraktní operace, která je základem pro všechny ostatní operace. Hlavní úlohou této operace je obstarat komunikaci s úložištěm. Pro tento účel musí operace umět sestavit HTTP požadavek, odeslat ho, přijmout odpověď a zpracovat ji. Předpokladem pro sestavení požadavku a zpracování odpovědi je již zmíněné mapování atributů (viz sekce 4.4).

V obrázku 4.4 je znázorněn návrh generické operace. Ostatní operace dodají generické operaci pouze potřebná data pro sestavení HTTP požadavku

a získají zpět data z odpovědi úložiště ve sjednoceném formátu. Každá operace musí dodat URI, na které má být požadavek odeslán a HTTP metodu požadavku. Pro vykonání operace není nutné specifikovat parametry, hlavičky a/nebo data požadavku. Jelikož je naprostá většina operací prováděna nad chráněnými daty, je pro přístup k nim potřeba dodat rovněž přístupový token, který je vložen do HTTP požadavku jako parametr nebo jako hlavička.



Obrázek 4.4: Návrh generické operace.

4.7.2 Manipulace se soubory

Pro každé úložiště existuje množina operací, která je určena výhradně pro manipulaci se soubory a složkami. V této sekci jsou popsány základní operace dostupné na všech úložištích a návrh jejich jednotného volání, včetně vstupních a výstupních parametrů. V tabulce 4.5 je přehled používaných HTTP metod [27, 28, 29, 30].

Stažení souboru – Pro stažení souboru je na všech úložištích používána HTTP metoda `GET`. Vstupem jsou metadata stahovaného souboru, cílový lokální soubor a zda má být lokální soubor přepsán v případě jeho existence. Výstupem je pak nutně lokální soubor, který byl stažen.

Nahrání souboru – Nahrávání souborů má každé z úložišť implementováno odlišně. Zde je hlavním zaměřením nahrávání souborů po částech. Dropbox pro nahrávání souboru využívá opakovaného volání HTTP metody `PUT`, pomocí kterého se na úložiště nahrává obsah souboru. Nahrávání souboru je zakončeno voláním metody `POST`. Google Drive očekává jako první krok metadata nového souboru předávaná pomocí `POST`. Následuje opakované volání `PUT`, kterým se předává obsah souboru. OneDrive nepodporuje nahrávání souboru po částech. Celý soubor je na úložiště nahráván jedním požadavkem typu `PUT`.

Vstupem operace pro nahrávání souboru je lokální soubor, cílová složka a název souboru, a zda má být případný existující soubor v úložišti přepsán. Výstup operace pak tvoří metadata nově nahraného souboru.

Úprava obsahu souboru – Tato operace je téměř shodná s operací nahrávání souboru. Rozlišení mezi úpravou a nahráváním je nutné pro úložiště Google Drive, které vyžaduje, aby první volání proběhlo HTTP metodou `PUT` namísto `POST`. Pokud by bylo volání provedeno metodou `POST`, byl by v úložišti vytvořen nový soubor, který by měl stejný název jako již existující soubor v úložišti. Pro uživatele by to znamenalo snížení přehlednosti obsahu úložiště. Vstupy a výstupy operace jsou shodné se vstupy a výstupy operace nahrávání. Na vstupu operace jsou navíc vyžadována metadata souboru, který má být změněn.

Získání obsahu složky – Operace pro získání obsahu složky je vykonávána na všech úložištích HTTP metodou `GET`. Vstupem pro operaci jsou metadata složky, jejíž obsah má být získán, a zda mají být ve výsledku zahrnuty smazané a sdílené soubory. Na výstupu jsou očekávána metadata složky doplněna o pole s metadaty souborů a složek, které složka obsahuje. Komplikace nastávají u úložiště Google Drive, které standardně vrací jako výsledek kompletní seznam souborů a složek v celém úložišti. Pro odstranění tohoto problému a získání pouze požadovaného obsahu je nutné odeslat ještě jeden `GET` požadavek, kterým jsou získány identifikátory souborů a složek, které složka obsahuje.

Vytvoření složky – Nová složka je na všech úložištích vytvořena pomocí HTTP metody `POST`. Vstupem operace je název nové složky a také metadata rodičovské složky. Výstupem pak metadata nově vytvořené složky.

Získání metadat – Pro získání metadat souboru či složky je všemi úložišti využito metody `GET`. Vstupem operace musí být jednoznačný identifikátor zdroje. V případě Dropboxu je to cesta ke zdroji, v případě Google Drivu a OneDrivu pak identifikátor. Na výstupu operace jsou očekávána kompletní metadata zdroje.

Vyhledávání – Vyhledávat je možné na všech úložištích. Použitou HTTP metodou je ve všech případech `GET`. Vstupem je pak vyhledávaný řetězec a zda má být vyhledáváno i mezi smazanými soubory. Výstupem je pak pole s metadaty souborů a složek. Problémem při vyhledávání je, že metadata úložišť Google Drive a OneDrive neobsahují cestu k souboru či složce. Pro získání umístění souboru či složky je tak potřeba vzít z metadat identifi-

kátor rodičovské složky a získat její metadata. Pokud není rodičovská složka kořenem adresářové struktury úložiště, je potřeba získat metadata rodičovské složky aktuální složky. Takto se pokračuje až do nalezení kořenové složky. Ze získaných metadat složek je možno sestavit cestu k souboru či složce z vyhledávání.

Kopírování – Pro kopírování platí u Google Drivu omezení pouze na soubory. Aby bylo možné vytvořit jednotnou operaci kopírování, je nutné omezit kopírování u ostatních úložišť rovněž pouze na soubory. Dropbox a Google Drive využívají pro kopírování HTTP metodu `POST`. OneDrive definuje vlastní HTTP metodu `COPY`. Vstupem operace jsou metadata kopírovaného souboru, metadata cílové složky a cílový název souboru v případě, že je podporováno přejmenování souboru během operace. Výstupem jsou metadata kopírovaného souboru.

Přesun – Přesunovat je možno soubory i složky. Dropbox využívá pro přesun HTTP metodu `POST`, zatímco Google Drive využívá `PUT`. Zvláštním případem je zde OneDrive, který má pro přesun definovanou nestandardní HTTP metodu `MOVE`. Na vstupu operace se očekávají metadata souboru (či složky) určeného pro přesun, metadata cílové složky a případně nový název souboru, pokud je v rámci přesunu povolena i změna názvu. Výstupem pak jsou nová metadata souboru či složky.

Přejmenování – Operace přejmenování není implicitně dostupná pro všechna vybraná úložiště. Dropbox tuto operaci neimplementuje, nicméně je možno ji nahradit operací přesunu souboru, pro kterou je využíváno HTTP metody `POST`. Ostatní úložiště podporují přejmenování metodou `PUT`. Operace je aplikovatelná na soubory i složky. Vstupem metody jsou metadata souboru či složky a nový název. Výstupem pak jsou nová metadata souboru či složky.

Smazání – Tato operace je aplikovatelná pro soubory i pro složky. Následkem vykonání této operace je smazání vybrané složky (včetně obsahu) nebo souboru. Dropbox používá pro mazání HTTP metodu `POST`, zatímco ostatní úložiště využívají metody `DELETE`. Vstupním parametrem této operace jsou metadata souboru či složky. Výstup této operace by měl indikovat, zda byla operace úspěšně provedena.

Operace	Dropbox	Google Drive	OneDrive
Stažení	`GET`	`GET`	`GET`
Nahrání	`PUT` + `POST`	`POST` + `PUT`	`PUT`
Úprava obsahu	`PUT` + `POST`	`PUT` + `PUT`	`PUT`
Obsah složky	`GET`	`GET` + `GET`	`GET`
Vytvoření složky	`POST`	`POST`	`POST`
Metadata	`GET`	`GET`	`GET`
Vyhledávání	`GET`	`GET`	`GET`
Kopírování	`POST`	`POST`	`COPY`
Přesun	`POST`	`PUT`	`MOVE`
Přejmenování*	`POST`	`PUT`	`PUT`
Smazání	`POST`	`DELETE`	`DELETE`

*) U Dropboxu je pro přejmenování využito operace přesunu.

Tabulka 4.5: HTTP metody operací.

4.7.3 Informační operace

Do této kategorie patří operace pro získávání informací o uživateli a jeho účtu v rámci úložiště. Všechny zde vyjmenované operace využivají HTTP metodu `GET`. Jelikož OneDrive rozděluje získávání informací o uživateli a získávání informací o účtu do dvou odlišných volání, je nutné rozdělit tato volání i u ostatních úložišť.

Informace o uživateli – Cílem této operace je získat základní informace o uživateli, mezi které patří uživatelské jméno a identifikátor uživatele. Na vstupu operace nejsou potřeba žádná data.

Informace o účtu – Výstupem této operace je zjištění celkového prostoru úložiště a jeho obsazené, resp. volné části. Ani tato operace nevyžaduje žádný vstup.

5 Návrh rozšířené funkcionality

Cílem této kapitoly je návrh funkcionality pro inteligentní práci se soubory v cloudových úložištích. Pod pojmem inteligentní je zde chápáno takové chování aplikace, které přináší uživateli zjednodušení práce. Za zjednodušení je možné považovat například snížení počtu různých aplikací, které by bylo pro dosažení stanoveného cíle potřeba instalovat, zkrácení času potřebného pro dosažení cíle, nebo snížení počtu úkonů, které by bylo nutné vykonat pro dosažení cíle.

Zde navržená funkcionalita se opírá o dostupnou dokumentaci pro přístup k úložištím a základní sjednocené operace navržené v předchozí kapitole. Cílem navržených rozšíření je nabídnout uživateli funkce, na které je zvyklý z existujících programů, a vylepšit je. Navržená funkcionalita se bude lišit od stávajících programů především tím, že bude aplikovatelná přes všechna vybraná úložiště.

5.1 Paralelní nahrávání

Vzhledem k tomu, že skrze aplikaci bude přistupováno k více cloudovým úložištím zároveň, nabízí se zde možnost implementace paralelního nahrávání souboru na více úložišť. Pro uživatele by tato možnost znamenala úsporu času, kterou by strávil nahráváním souboru na každé úložiště zvlášť. Navrhovaná funkcionalita totiž počítá se souběžným spuštěním nahrávání na všechna úložiště zároveň. Celkový čas potřebný pro nahrání souboru by se tak rovnal času potřebnému pro nahrání souboru na úložiště s nejnižší rychlostí nahrávání.

Komplikací by mohlo být přerušení nahrávání souboru. V takovém případě by mohlo dojít k situaci, že by přerušení nastalo v momentě, kdy už by byl soubor na některá úložiště nahrán, ale na ostatní ne. Nejjednodušším řešením by bylo situaci ignorovat, což je možné, protože na žádném z úložišť se nekompletní soubory nijak neodráží na využité kapacitě úložiště a nejsou přidány do cílové složky. Další možností by bylo již nahrané soubory smazat a vrátit tak úložiště do stavu před započetím operace.

5.2 Zrychlení stahování

Teoreticky je možné získat pouze vybranou část souboru z úložiště. Předpokladem pro tento způsob přístupu k datům je jeho podpora na vybraném úložišti. Pokud tedy úložiště takovýto způsob přístupu k datům podporuje, bývá zpravidla někde zdokumentován. Pro zjištění podpory bylo tedy nutné důkladně prostudovat dokumentaci a zdrojové kódy API vybraných úložišť. Úložiště přímo specifikují v dokumentacích svých API tuto možnost, a jakým způsobem je možno ji využít [27, 28, 30]. Jedná se o podporu hlavičky `Range` protokolu HTTP/1.1, pomocí které je možno specifikovat rozsah bajtů souboru, který má být stažen [44, 46]. Jiná úložiště mohou tuto možnost rovněž podporovat, avšak může být teoreticky zavedena i jiným způsobem.

Tato možnost je klíčová pro využití redundantního uložení souborů ve více úložištích ve prospěch koncového uživatele. Kromě možnosti navázání stahování souboru po výpadku klienta se tím otevírá i možnost pro navázání stahování souboru z jiného úložiště, než ze kterého byl soubor stahován. Ještě zajímavější je pak možnost paralelního stahování souboru z více úložišť zároveň.

Paralelní stahování souboru funguje v principu tak, že soubor je rozdělen do více částí, které jsou stahovány nezávisle na sobě a sestaveny až na straně uživatele, který je stahuje. Hlavní výhodou tohoto přístupu je z hlediska uživatele teoretické navýšení rychlosti stahování souboru, ale pouze v případech, že již uživatel nevyužívá plnou kapacitu linky, kterou je připojen k úložišti. Z pohledu cloudových úložišť je pak možné jako pozitivum považovat snížení zátěže serverů a konektivity v důsledku rozložení stahování mezi různá úložiště a tím i přenášení menších datových objemů z každého úložiště.

Pod pojmem paralelního stahování si lze představit nejen stahování z více úložišť najednou, ale též stahování z jednoho úložiště ve více vláknech najednou. Z tohoto pohledu je taktéž teoreticky možné zvýšit rychlost přenosu souboru. Vše opět závisí na poskytovateli úložného prostoru, zda tuto možnost povoluje a jakým způsobem upravuje rychlost přenášení dat. Nevýhodou tohoto přístupu je dočasné zvýšení zátěže úložiště oproti běžnému stahování v jednom vlákně.

Možnosti stahování souboru jsou názorně ilustrovány v několika následujících obrázcích. V obrázku 5.1 je znázorněno sekvenční stahování souboru z jednoho úložiště. V obrázku je zachyceno rozdělení souboru do několika částí a jejich postupné stahování. V tomto případě jsou jednotlivé

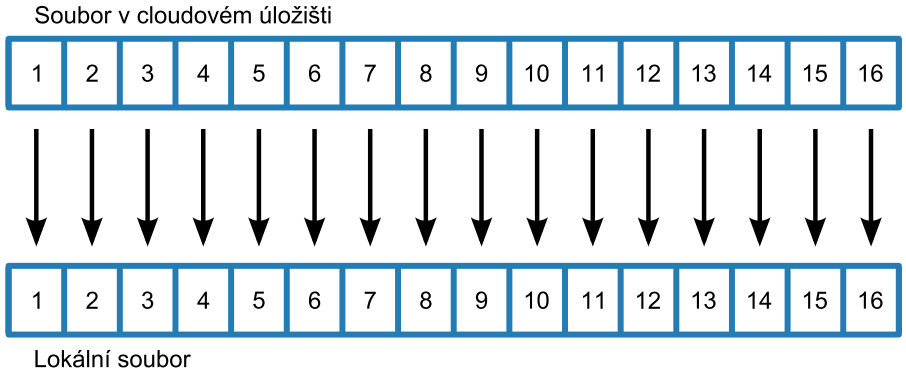
části souboru stahovány jedním vláknem v pořadí podle čísel, kterými jsou označeny.

Paralelní stahování z jednoho úložiště je pak znázorněno obrázkem 5.2. Od předchozího obrázku se paralelní stahování liší především tím, že vláken pro stahování je více. Soubor je opět rozdělen do několika částí a vláknům jsou jednotlivé části přidělovány opět v sekvenčním pořadí dle jejich číselného označení. Vlákna mezi sebou o jednotlivé části soupeří. Každá jednotlivá část je tak přidělena tomu vláknu, které o ni jako první zažádá, přičemž žádost může vyslat pouze vlákno, které dokončilo stahování předchozí části.

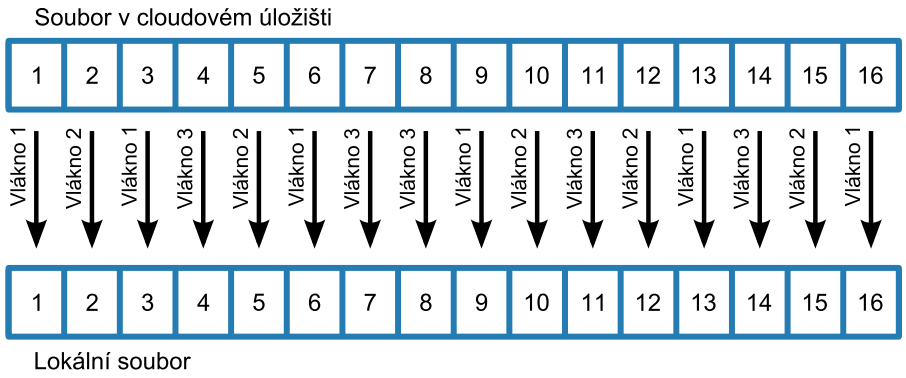
Dále následuje sekvenční stahování z více úložišť v obrázku 5.3. V tomto případě je opět použito jen jedno vlákno, které stahuje části souborů dle jejich číselného označení. Od stahování z jednoho cloudového úložiště se tento přístup liší především tím, že každá jednotlivá část souboru je stahována z jiného úložiště.

Posledním obrázkem 5.4 je kombinovaný přístup pro stahování pomocí více vláken z více úložišť. Soubor je zde rozdělen do částí, o které jednotlivá vlákna soupeří. Každému z vláken je přiděleno jedno cloudové úložiště, ze které pak části souborů stahuje.

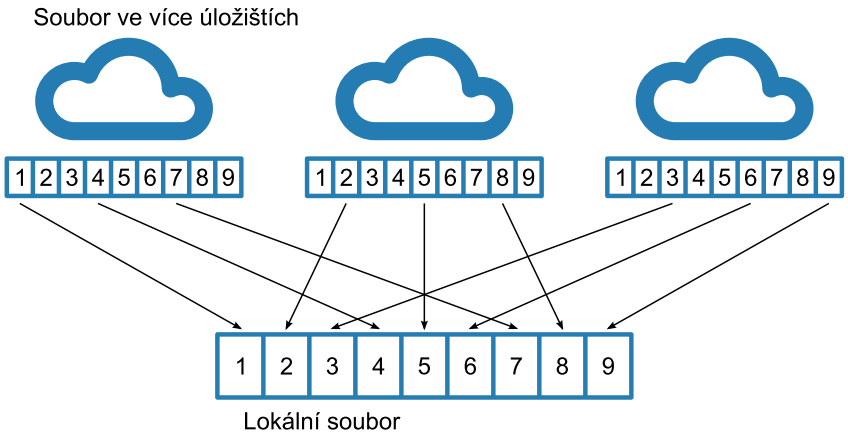
Pro implementaci byl zvolen postup stahování znázorněný obrázkem 5.4. Jedná se o univerzální postup, který je možné volbou počtu vláken a cloudových úložišť převést na zbylé tři zmíněné postupy. Také je možné u tohoto přístupu předpokládat nejvyšší míru urychlení stahování souboru. Dosažené výsledky tohoto přístupu k datům v úložištích jsou blíže popsány a vyhodnoceny v sekci 6.4.



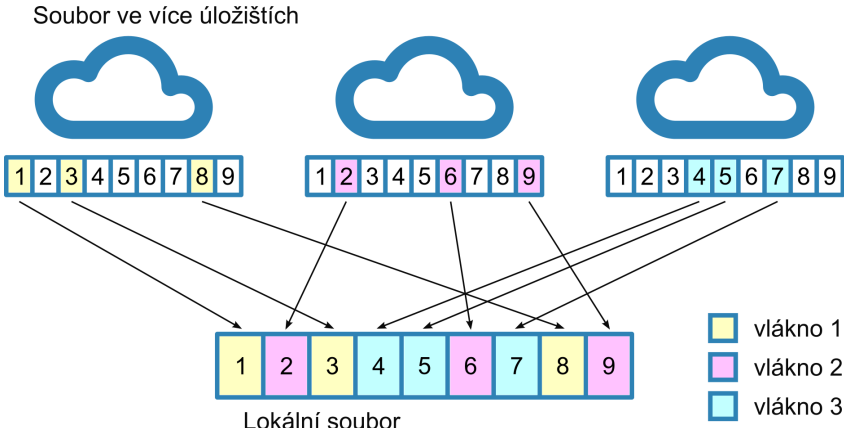
Obrázek 5.1: Sekvenční stahování souboru jedním vláknem z jednoho úložiště.



Obrázek 5.2: Stahování souboru ve více vláknech z jednoho úložiště.



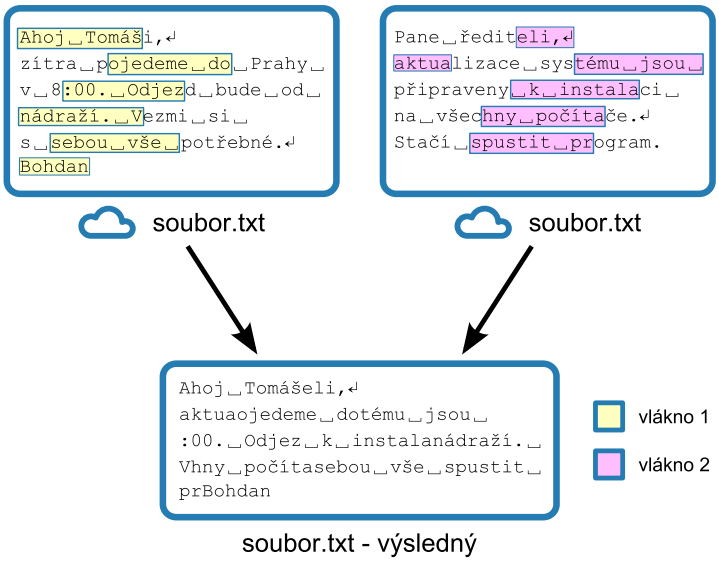
Obrázek 5.3: Sekvenční stahování souboru jedním vláknem z více úložišť.



Obrázek 5.4: Stahování souboru z více úložišť ve více vláknech.

5.3 Kontrola konzistence dat

Pro stahování souborů z více úložišť zároveň je potřeba zaručit konzistenci obsahu souborů napříč úložišti. Pokud by byť jediný ze souborů neobsahoval stejná data jako ostatní, mohlo by během stahování dojít k poškození výsledného souboru. Výsledný stažený soubor by totiž obsahoval části dat ze všech stahovaných souborů a neodpovídal žádnému z nich. Tento případ je názorně ilustrován obrázkem 5.5.



Obrázek 5.5: Problém nekonzistence souborů napříč úložišti.

Pro ošetření tohoto problému existuje několik možných řešení. Jedním řešením by bylo stažení souboru ze všech úložišť zároveň a porovnání jejich obsahu. Následně by pak bylo možno vybrat jednu verzi souboru, která by odpovídala představám uživatele. Tento přístup ovšem není příliš vhodný, protože by soubor bylo nutné stahovat N-krát, kde N je počet úložišť, ze kterých by byl soubor stahován.

Lepším přístupem je získání pouze kontrolních součtů souborů a jejich porovnání před stažením. U kontrolních součtů se předpokládá, že dva soubory mají shodný kontrolní součet pouze v případě, že mají i shodný obsah. Toto ovšem nemusí platit vždy a mohou existovat dva obsahově odlišné soubory se stejným kontrolním součtem. Před tímto problémem je možno se bránit kontrolou velikosti souborů. Pravděpodobnost, že by dva stejně

velké soubory s rozdílným obsahem měly stejné kontrolní součty, je prakticky nulová. Kontrolní součty je navíc možno ukládat a přistupovat k nim později.

Jelikož jediným úložištěm, které poskytuje kontrolní součty jako součást metadat, je Google Drive, je nutné pro soubory v ostatních úložištích kontrolní součty počítat manuálně [28]. Pro vypočtení kontrolního součtu souboru je potřeba soubor nejprve stáhnout.

Důležité je rovněž zpětné přiřazení kontrolního součtu k souboru. To je možné pouze v případě, že soubor nebyl změněn. Spolu s kontrolním součtem je tedy nutné uchovávat i metadata souboru, ke kterému kontrolní součet patří. Pro jednoznačné přiřazení kontrolního součtu k souboru je potřeba znát jednoznačný identifikátor souboru, ať už se jedná o cestu k souboru v případě úložiště Dropbox, nebo identifikátor u úložišť Google Drive a OneDrive. Dále by měl odpovídat i název souboru. Co odpovídat musí, je velikost souboru, a datum a čas poslední úpravy. Pokud je dostupný údaj o datu a čase vytvoření souboru, musí odpovídat i ten. Poslední důležitý atribut, který musí odpovídat, je identifikátor uživatelského účtu v rámci úložiště.

5.4 Synchronizace složek

Na základě kontrolních součtů je možno synchronizovat lokální a vzdálené složky. Zde je nejdůležitější detekce změny souboru. Soubor, který byl pozměněn, je pak potřeba nahrát na úložiště nebo z úložiště stáhnout, v závislosti na tom, kde změna nastala.

Synchronizaci je možno buď provádět kontinuálně nebo jednorázově. Kontinuální přístup vyžaduje sledování změn v lokálním souborovém systému a ve vzdálených úložištích zároveň. Pro sledování lokálních změn je potřeba program, který by pomocí funkcí operačního systému změny sledoval. Sledování vzdálených změn pak vyžaduje periodické dotazy na metadata sledovaných souborů. Některá úložiště dokonce podporují zasílání notifikací v případě změn souborů.

Jednorázová synchronizace je jednodušší, protože nevyžaduje žádné sledování souborů, ať už lokálních nebo vzdálených. Pro jednorázovou synchronizaci tak stačí získání kontrolních součtů souborů v synchronizovaných složkách, jejich porovnání a vyhodnocení změn. Po identifikaci změněných souborů je potřeba určit směr, kterým se změny budou propagovat.

5.4.1 Selektivní synchronizace

Selektivní synchronizace je vlastnost aplikace, která umožňuje uživatelům zvolit si v rámci synchronizační složky obsah, který bude synchronizován. Ve výsledku se tak synchronizují pouze uživatelem označené soubory.

Pro selektivní synchronizaci je nejprve důležité určit, zda má prioritu lokální nebo vzdálená složka. Většina existujících klientů používá vzdálenou složku jako prioritní. V praxi to znamená, že uživatel si vybírá, které ze vzdálených podsložek se budou synchronizovat na lokální disk.

Přístup s prioritní vzdálenou složkou není ovšem vhodný v případě, že je pro synchronizaci zvoleno více úložišť najednou. Nastává zde totiž problém s určením prioritní složky. Pokud by bylo jedno z úložišť určeno jako hlavní, musela by synchronizace probíhat i mezi úložišti. Lepším řešením tak je volba lokální složky jako prioritní. Uživateli se tak nabízí možnost vybrat si na která úložiště budou lokální podsložky synchronizovány.

5.5 Transfer souborů

Další navrženou rozšiřující funkcí je přesun souborů mezi úložišti. Možností pro implementaci této funkcionality je opět více. Ideálním řešením by byl přesun souborů přímo mezi úložišti. To ovšem není bohužel možné, protože by tato možnost musela být podporována přímo úložišti. Soubory je tedy nutné přesunovat prostřednictvím klientské aplikace.

První variantou řešení je stažení celého souboru na lokální disk a jeho následné nahrání na cílové úložiště. Hlavní výhodou tohoto přístupu je možnost využití rychlejšího stahování popsaného v sekci 5.2. Nevýhodou pak je potřeba dostatečně velkého prostoru na lokálním disku pro dočasné uložení souboru. Další nevýhodou je, že soubor může být nahrán na vzdálené úložiště až po jeho kompletním stažení na lokální disk.

Druhou variantou řešení by mohlo být sekvenční stahování souboru do mezipaměti a současné nahrávání stažených dat na cílové úložiště. Mezipaměť by tak mohla mít velikost zlomku velikosti celého souboru a nemusela by vůbec využít lokální diskový prostor. Nevýhodou však je pomalejší stahování dat.

Pokud se vezme v potaz i kontrola konzistence dat a synchronizace složek, je více než vhodné vypočítávat kontrolní součet souboru při jeho průchodu skrze klienta. Toto přináší další nevýhodu pro druhou variantu řešení, jelikož by bylo nutné duplikovat přijatá data. Jedna kopie dat by byla nahrávána na cílové úložiště, zatímco z druhé kopie by byl vypočten kontrolní součet.

Z navržených variant byla zvolena metoda stažení celého souboru a jeho následné nahrání na cílové úložiště. Důvodem této volby byla možnost využití stahování ve více vláknech. Dalším důvodem pro tuto volbu byla možnost využití již navržených operací pro implementaci transferu souborů mezi úložišti.

5.6 Rozvržení funkcionality

V následujícím obrázku 5.6 je zachyceno rozdělení navržené základní i rozšířené funkcionality mezi implementované programové části. Jednotlivá funkcionalita je symbolicky označena. `USER INFO` a `USER QUOTA` reprezentují základní operace pro získání informací o uživateli a jeho kapacity prostoru v rámci úložiště. Následují operace pro manipulaci se soubory.

Operace `LIST FOLDER` představuje operaci získání obsahu vzdálené složky. `CREATE FOLDER` je pak vytvořením složky v úložišti. Operace pro nahrávání a stahování jsou označeny jako `UPLOAD` a `DOWNLOAD`. Operace přejmenování souborů a složek je označena jako `RENAME`. Přesun a kopírování souborů je pak `MOVE` a `COPY`. Operace mazání souborů a složek je nazvána `DELETE`. Operace `SEARCH` pak představuje operaci vyhledávání souborů v úložišti. Poslední základní funkcionalitou je získávání metadat souborů a složek označené jako `METADATA`.

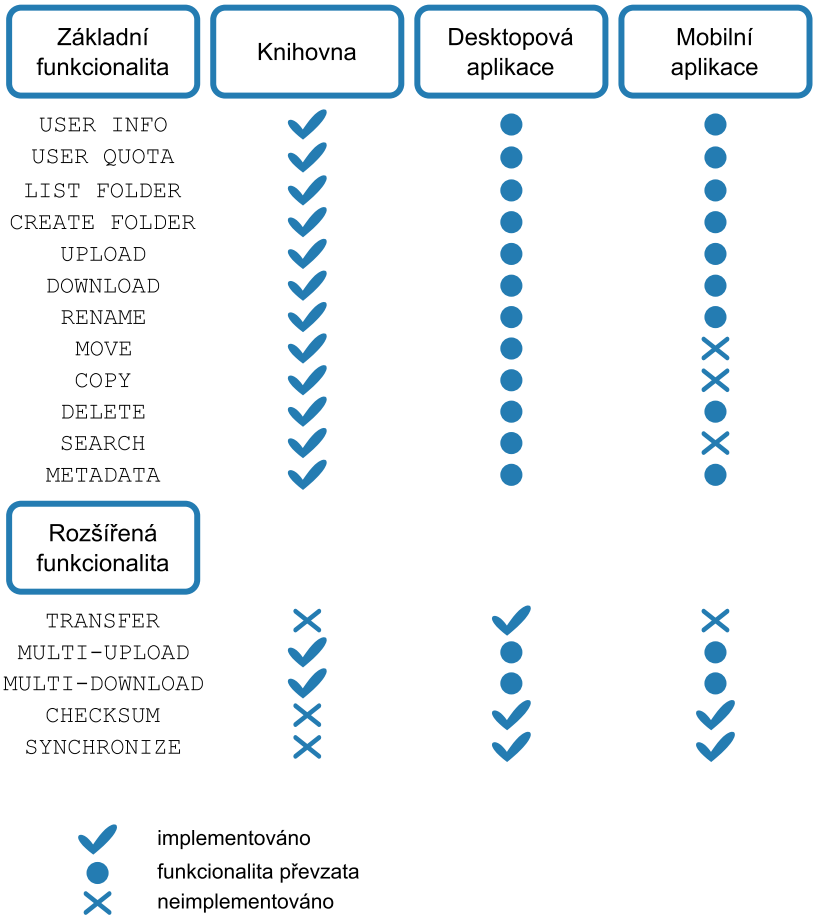
Rozšířená funkcionalita obsahuje operace pro transfer souborů mezi úložišti `TRANSFER`. Tato operace zahrnuje kopírování i přesun souborů. `MULTI-UPLOAD` představuje operaci pro nahrávání souboru na více úložišť zároveň a `MULTI-DOWNLOAD` operaci pro stahování souboru ve více vláknech. `CHECKSUM` pak označuje rozšířenou funkcionalitu pro vytváření kontrolních součtů a jejich porovnávání v rámci některých operací. Poslední je operace synchronizace `SYNCHRONIZE`, která v sobě zahrnuje i selektivní synchronizaci.

Veškerá základní funkcionalita je navržena jako součást knihovny. Z rozšířené funkcionality obsahuje knihovna ještě operace pro nahrávání souboru na

více úložišť a stahování souboru z více úložišť. Zbylá rozšiřující funkcionalita byla navržena jako součást aplikace využívající implementovanou knihovnu.

Desktopová aplikace obsahuje veškerou funkcionalitu, která je obsažena v knihovně. Nad rámec již poskytnuté funkcionality implementuje aplikace i zbylou rozšiřující funkcionalitu.

Součástí mobilní aplikace je pouze omezená funkcionalita. Ze základních operací byly vynechány operace pro přesun a kopírování souborů a vyhledávání. Z rozšiřující funkcionality je pak vynechána operace pro transfer souborů mezi úložišti. Operace pro nahrávání souborů na více úložišť a stahování z více úložišť jsou využívány pouze v rámci synchronizace, nikoliv samostatně.



Obrázek 5.6: Přehled implementované funkcionality.

6 Implementace knihovny

Pro tvorbu knihovny bylo nutné zvolit nejprve komponenty. Z návrhu knihovny vyplývá, že je potřeba určit několik základních stavebních kamenů, ze kterých bude knihovna složena. Komponenty knihovny jsou tvořeny HTTP klientem, komponentou pro práci s daty ve formátu JSON, komponentou pro autorizaci pomocí OAuth 2.0 a webovým serverem. Dále pak bylo rozhodnuto o implementaci sekce 5.2 jako součásti knihovny.

6.1 Volba komponent

První komponentou, kterou bylo potřeba zvolit, byl HTTP klient. V úvahu připadal HTTP klient od Apache, HTTP klient od Googlu a implementace HTTP spojení dostupná jako součást Javy. Z těchto variant byl nakonec zvolen HTTP klient od Apache, protože se jedná o robustní řešení, které je navíc dostupné jako výchozí HTTP klient na platformě Android [33, 52].

Další komponentou je parser pro data ve formátu JSON. Zde se vycházelo ze seznamu dostupných knihoven na *json.org* [19]. Hlavními kandidáty byly knihovny JSON.simple, Jackson a GSON [34, 35, 53]. Při výběru bylo přihlíženo na dokumentaci knihoven a na výsledky srovnávacích testů rychlosti serializace a deserializace. Na základě srovnávacích testů byla nakonec zvolena knihovna Jackson [54, 55, 56].

U autorizační komponenty byl výběr prováděn na základě seznamu klientských knihoven pro OAuth 2.0 [14]. Po analýze dostupných knihoven byla nakonec zvolena vlastní implementace. Výhodou této volby je především možnost zakomponování webového serveru pro získávání přístupových tokenů z autorizačního procesu.

Pro volbu webového serveru bylo vycházeno ze seznamu existujících open source webových serverů [57]. Cílem bylo vybrat co nejjednodušší implementaci, která bude zároveň podporovat parsování HTTP požadavků a naslouchání na náhodném portu. Po prozkoumání dostupných řešení byl zvolen HTTP server Simple, který splňuje všechna kritéria výběru [58].

6.2 Odlišnosti cloudových úložišť

Při implementaci je potřeba se soustředit na drobné odlišnosti mezi jednotlivými úložišti. Hlavní dosud nepopsané rozdíly jsou v konfiguraci aplikace na straně úložiště a v některých operacích, které se skládají z více různých požadavků na server.

6.2.1 Konfigurace na straně úložišť

Aby bylo možné používat aplikaci pro přístup k úložišti, je nutné ji na úložišti zaregistrovat. Registrace aplikace vyžaduje registrovaného uživatele. Registrací aplikace je získán identifikátor aplikace a tajný klíč, které jsou potřeba pro autorizaci přístupu.

U Dropboxu je důležité povolit aplikaci přístup ke všem souborům uživatele. V opačném případě by byl přístup aplikace omezen pouze na jednu složku speciálně vytvořenou pro aplikaci. Po vytvoření aplikace má aplikace přidělen status vývoje a pro získání produkčního statusu je potřeba odeslat žádost z administračního rozhraní Dropboxu. Aplikaci je jinak možno nastavit ikonu, popis a domovskou stránku [59].

Při registraci aplikace na Google Drive je potřeba při vytváření OAuth přístupových informací zvolit typ aplikace jako *Installed application* a *Other*. Kromě registrace aplikace je nutné ještě povolit přístup ke konkrétnímu API, v tomto případě *Drive API*. Dále je možné nastavit ikonu aplikace, podmínky užití a zásady ochrany osobních údajů, které budou zobrazeny při autorizaci aplikace [60].

Pro OneDrive je rovněž možné nastavení ikony aplikace a přiložení podmínek užití a zásad ochrany osobních údajů. Oproti ostatním úložištím je zde možné nastavit lokalizovaný název aplikace. Zvláštně formulovanou možností nastavení je zde volba mezi mobilní a desktopovou aplikací. Nastavení by mělo upravovat, které autorizační granty bude možné v aplikaci využít. Při testování však nebyl zjištěn žádný vliv tohoto nastavení na knihovnu. Pro knihovnu byla nakonec zvolena volba desktopové aplikace. S touto volbou byla později knihovna využita i na mobilné platformě bez problémů [61].

6.2.2 Rozfázování operací

Některé operace vyžadují vykonání více požadavků než pouze jednoho. Přitom je nutné ještě rozlišovat, zda všechny vykonávané požadavky v rámci jedné operace mají stejný základ nebo ne. V případě stejného základu je možné zpracovávat požadavek uvnitř cyklu. V opačném případě je potřeba operaci rozdělit do více fází. Důvodem pro rozfázování operací jsou operace *získání obsahu složky* a *nahrávání souboru*.

Knihovna implementuje rozdělení do tří fází – přípravné, exekuční (hlavní) a ukončovací. První fáze slouží jako zahájení operace. V druhé fázi je vykonán hlavní požadavek celé operace a v poslední fázi je provedeno ukončení operace. Jednotlivé operace nemusí využít všech fází.

Získání obsahu složky – Pro tuto operaci je potřeba rozdělení do více fází z důvodu způsobu, jakým je získáván obsah složky na úložišti Google Drive. V tomto konkrétním případě je získáván zvlášť seznam všech souborů v úložišti a seznam identifikátorů souborů, které jsou obsahem konkrétní složky. Do preparační fáze byl tedy vyčleněn požadavek na získání seznamu identifikátorů. V hlavní fázi pak zůstal požadavek na získání obsahu složky.

Nahrávání souboru – Tato operace má pro každé úložiště odlišný průběh. Pro Dropbox platí, že v prvním požadavku je posílána první část dat. V odpovědi serveru je pak obsažen identifikátor relace pro nahrávání souboru. Následuje opakované nahrávání částí souboru dokud není soubor nahrán celý. Operace je zakončena speciálním požadavkem, kterým je potvrzeno nahrání souboru na úložiště a nastavení jeho umístění v úložišti [27].

Google Drive má pro nahrávání souboru opačný postup. Nejprve jsou poslána metadata a teprv poté je nahráván soubor po částech [28]. Pro OneDrive není možné nahrávání souboru po částech a stačí tak pouze jeden požadavek [31].

V průběhu implementačního procesu bylo rozhodnuto o implementaci souběžného nahrávání souboru na více úložišť zároveň v rámci knihovny. Následkem tohoto rozhodnutí bylo přesunutí procesu nahrávání a jeho fází do samostatného vlákna. V důsledku tohoto přesunu je pro nahrávání souboru využita pouze hlavní fáze, která spouští vlákna pro nahrávání. Poslední fáze tak není využívána žádnou operací a zůstává tak připravena pro možná rozšíření knihovny.

6.3 Rozvržení knihovny

Knihovna je rozvržena do několika základních částí, které navzájem spolupracují. Základní částí knihovny je správa podporovaných cloudových úložišť, uživatelských účtů a přístupových tokenů.

Cloudová úložiště – Rozhraním pro správu úložišť je `CloudManager`. V knihovně je vytvořena i jeho jednoduchá implementace, která načítá definice cloudových úložišť ze souborů ve formátu JSON.

Každá definice cloudového úložiště obsahuje základní identifikaci úložiště, přístupové informace pro autorizaci aplikace a definice požadavků pro autorizaci a pro jednotlivé operace. Každý požadavek pak obsahuje URI požadavku, HTTP metodu požadavku a může obsahovat definici mapování parametrů, seznam parametrů volání, hlavičky požadavku a tělo požadavku.

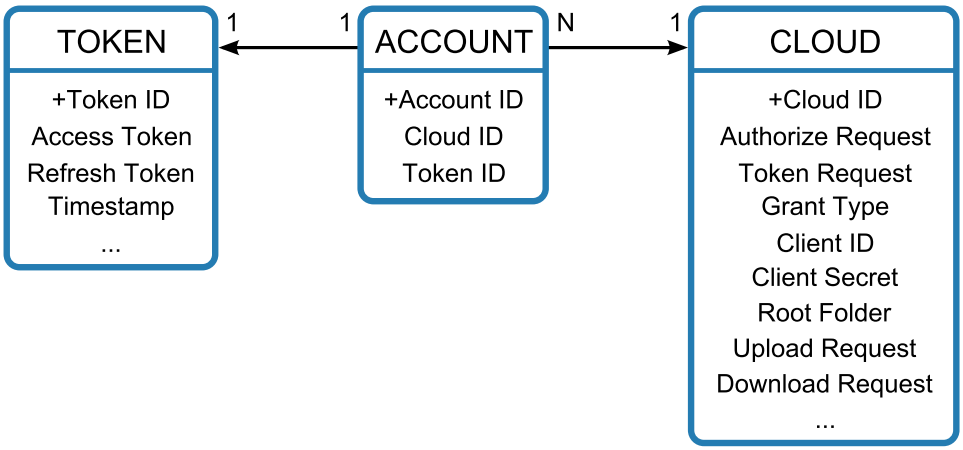
Uživatelské účty – Pro správu uživatelských účtů je vytvořeno rozhraní `AccountManager`. Dále je dostupná jeho implementace pracující se soubory na lokálním disku. Informace uchovávané pro každý uživatelský účet jsou tvořeny názvem účtu, identifikátorem cloudového úložiště a identifikátorem tokenu.

Přístupové tokeny – Pro každý uživatelský účet je potřeba ukládat přístupové a obnovovací tokeny. Jelikož jsou přístupový i obnovovací token vázány pouze na jeden účet, je možné je uchovávat společně v páru. Pro správu tokenů je připraveno rozhraní `CredentialStore` a dvě základní implementace tohoto rozhraní, které pracují se soubory.

První implementací je základní ukládání tokenů do souboru. Na výběr je zde typ serializace. Je možno volit mezi základní serializací objektů dostupné v Javě a převodem objektů do formátu JSON. Pro získání tokenu je nutné znát jeho identifikátor, pod kterým je uložen. Identifikátory jsou tvořeny řetězci a je možné je zvolit ručně nebo generovat náhodně. Výchozí délka generovaného identifikátoru je 8 znaků.

Druhou implementací je rozšíření první implementace o šifrování souboru. Pro šifrování je dle návrhu využito symetrické šifrování AES [48]. V implementaci je nastaven výchozí klíč pro šifrování, který je možno změnit. Pro serializaci dat je zde využíváno pouze objektové serializace Javy a není ji možno změnit. Tato implementace splňuje požadavky na ukládání tokenů specifikované v dokumentu RFC 6749 [13].

Pro správu úložišť, uživatelů a tokenů platí následující logika. Každá definice cloudového úložiště se vztahuje pouze k jednomu úložišti. Mělo by se předejít duplicitním definicím úložišť. Více uživatelských účtů může přistupovat k jednomu cloudovému úložišti. Pro každý uživatelský účet existuje vlastní sada tokenů. Tyto vazby jsou pak znázorněny v obrázku 6.1.



Obrázek 6.1: Vazby mezi cloudovými úložišti, tokeny a uživatelskými účty.

Další částí knihovny je vlastní implementace autorizačního procesu. Do této části patří především rozhraní `OAuth2Grant`, které tvoří základ pro implementaci jednotlivých autorizačních procesů. Implementací tohoto rozhraní lze také zavést rozšiřující granty. Jednou z metod rozhraní je metoda `authorize`, která spouští autorizační proces. Pokud se jedná o proces, při kterém je vyžadována interakce s uživatelem, je touto metodou navrácen požadavek na interakci s uživatelem. Knihovna rovněž poskytuje rozhraní `AuthorizationCallback`, jehož implementaci je právě předáván požadavek na interakci s uživatelem. Implementace tohoto rozhraní tak může například otevřít okno prohlížeče a zobrazit uživateli stránku s požadavkem na autorizaci uživatele.

Součástí autorizace je i naslouchání na lokálním portu. Jelikož se jedná o náhodný port, je starostí naslouchajícího serveru i předání informací o přiděleném portu. Implementace předávání čísla lokálního portu odpovídá návrhu a je předáváno parametrem `state`. Nedílnou úlohou naslouchajícího serveru je přijetí požadavku od autorizačních serveru a předání přijatých dat pomocí rozhraní `RedirectCallback`. Od třídy implementující toto rozhraní je pak navrácen obsah webové stránky, která má být zobrazena uživateli.

Poslední velkou částí knihovny je implementace jednotlivých operací. Pro snazší implementaci je zde vytvořena abstraktní třída `Operation<T>`, která slouží jako základ pro všechny ostatní operace. Součástí této třídy je i typový parametr `T`, který určuje typ návratových dat operace. Každá operace rozšiřující tuto abstraktní operaci musí obsahovat implementaci metod `abort`, `operationBegin`, `operationExecute` a `operationFinish`. Poslední tři jmenované metody tvoří dohromady celý proces operace a jsou volány z metody `execute`. Rozdělením metody `execute` do tří dalších metod je dosaženo rozfázování operace (více v sekci 6.2.2).

Metoda `abort` – Tato metoda slouží pro přerušení celé operace. Pokud tedy operace využívá například více vláken, musí při zavolání této metody být všechna vlákna přerušena a běh operace ukončen. Pro operaci běžící v jednom vlákně to znamená přerušení právě vykonávaného požadavku HTTP klientem.

Metoda `operationBegin` – Metoda je první fází operace, ve které by mělo docházet k získání dat potřebných pro vykonání další fáze operace. Většina operací má tuto metodu prázdnou, protože nevyžadují žádné předzpracování. Operacemi, které vyžadují předzpracování, jsou operace *získání obsahu složky* a *nahrávání souboru*.

Metoda `operationExecute` – Jedná se o hlavní fázi operace, která je implementována všemi operacemi. V této fázi je zpravidla provedeno odeslání požadavku na cloudové úložiště a zpracování přijaté odpovědi. Operace *stahování souboru* a *nahrávání souboru* si v této fázi vytvářejí vlákna.

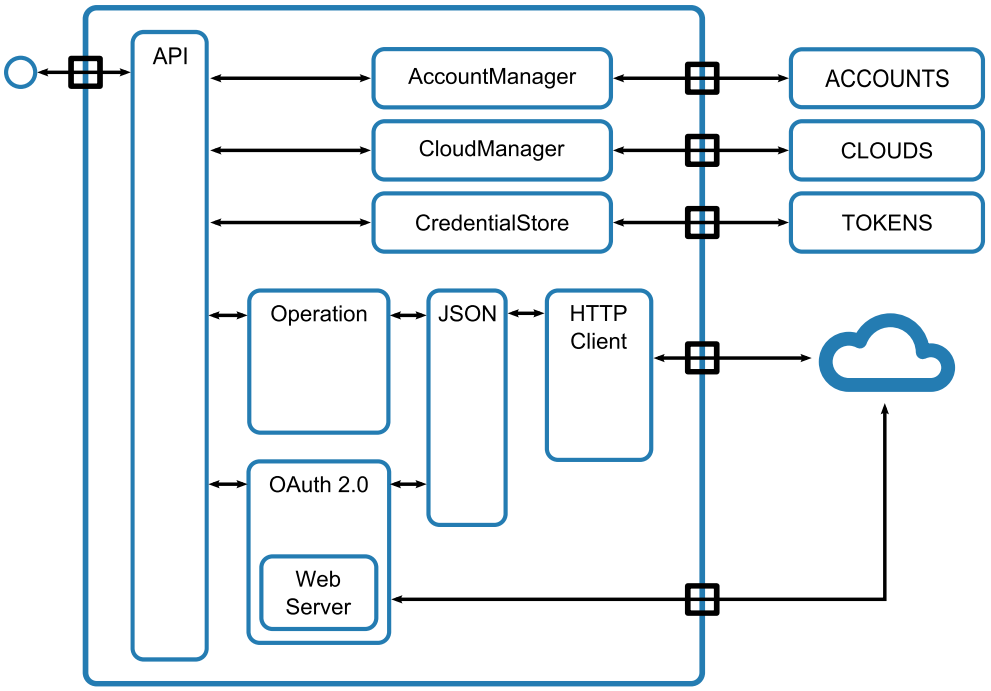
Metoda `operationFinish` – Tato metoda slouží k odeslání požadavku sloužícího pro ukončení celé operace. Ve většině případů je i tato metoda ponechána prázdná. Výjimku tvoří operace *nahrávání souboru*, která zde v případě nahrávání souboru po částech posílá požadavek na ukončení nahrávání souboru.

Jak již bylo zmíněno, některé operace si vytvářejí vlákna. Konkrétně se jedná o operace *stahování souboru* a *nahrávání souboru*. Pro stahování souboru je dělení do vláken prováděno z důvodu paralelního stahování, ať už se jedná o stahování z jednoho nebo z více úložišť. U nahrávání souboru je prostřednictvím více vláken umožněno nahrávat soubor na více úložišť zároveň (viz sekce 5.1).

Pro operace stahování a nahrávání souboru je dostupná abstraktní třída `ProgressListener`, která umožňuje sledování průběhu těchto operací. Jedi-

nou abstraktní metodou této třídy je metoda `onProgress()`, která je vyvolávána po uplynutí časového intervalu. Časový interval je možné předat třídě pomocí konstruktoru. Příslušné operace tak předávají třídě počty přenesených bajtů a ta se stará o jejich předání uživateli. Pro stahování i nahrávání souborů musí být vždy dodána knihovně implementace této třídy.

Obrázek 6.2 zachycuje zjednodušený schematický model knihovny, ve kterém je kladen především důraz na zobrazení vnitřní struktury knihovny a zachycení oblastí, do kterých náleží jednotlivé komponenty knihovny. Na levé straně je vstupní bod knihovny. Knihovna poskytuje své metody aplikacím formou API. Na druhé straně pak knihovna získává a ukládá uživatelské účty, definice cloudových úložišť a tokeny, a přistupuje ke cloudovým úložištím. Uvnitř knihovny jsou pak vidět jednotlivé komponenty, ze kterých se knihovna skládá, a způsob, kterým mezi sebou jednotlivé komponenty spolupracují.



Obrázek 6.2: Vnitřní struktura knihovny.

Samotná knihovna pak obsahuje 69 souborů, které obsahují celkem 52 tříd, 7 rozhraní, 7 enumerací a 4 výjimky. Soubory jsou rozděleny do 5 balíků se specializovaným zaměřením a jednoho obecného balíku. Specializované balíky tvoří následující balíky:

- `filesystem` – Obsahem tohoto balíku jsou jednotlivé operace prováděné nad úložišti. Součástí balíku jsou i třídy umožňující sledování průběhu některých operací.
- `http` – Tento balík obsahuje implementaci HTTP metod `COPY` a `MOVE`, které využívá OneDrive.
- `json` – V tomto balíku jsou převážně třídy typu Plain Old Java Object (POJO), které jsou mapovány JSON parserem.
- `oauth2` – Obsahem tohoto balíku je implementace autorizace pomocí OAuth 2.0. Součástí balíku je i webový server pro zachycení přesměrovaní po dokončení procesu autorizace.
- `utils` – V tomto balíku jsou obsažena rozhraní pro správu uživatelských účtů, definic cloudových úložišť a tokenů. Součástí balíku jsou i základní implementace těchto rozhraní a třída `Utils`, která zprostředkovává statické metody pro úpravu dat.

Rozsah knihovny byl měřen pomocí metriky Lines of Code (LOC). Celkový rozsah knihovny přesáhl 11 tisíc řádek kódu, z toho přibližně 31% je obsaženo v balíku `filesystem`, 26% v balíku `oauth2`, 16% je ve výchozím balíku, 15% v balíku `json`, 11% v balíku `utils` a 1% v balíku `http` [66].

6.4 Dosažené výsledky

Implementace knihovny podporuje všechny základní operace navržené pro implementaci. Rovněž je v knihovně implementován navržený autorizační proces. Nad rámec těchto vlastností je ještě implementováno rychlejší stahování souborů navržené v sekci 5.2 a paralelní nahrávání ze sekce 5.1. Rychlejší stahování pak bylo testováno a jeho výsledky následně porovnány s tradičním sekvenčním stahováním. Výsledky pro rychlejší stahování jsou zaznamenány v tabulce 6.1. Graficky jsou pak výsledky znázorněny v obrázku 6.3.

Pro testování rychlejšího stahování byl připraven soubor o velikosti 256 MB. Při stahování byl tento soubor rozdělen na několik částí, které byly stahovány nezávisle na sobě. Dělení souboru bylo provedeno automaticky knihovnou. Měřeny byly časy stahování souboru pro různé konfigurace vláken a úložišť. Testování proběhlo na symetrické lince s kapacitou přenosu dat 100 Mb/s. Každé měření bylo provedeno 10krát.

Vzhledem k tomu, že výsledky měření jsou ovlivňovány různými nepředvídatelnými vlivy, nepřineslo by v tomto případě více měření nutně přesnější výsledky. Jelikož se jedná o cloudovou službu, ke které se přistupuje prostřednictvím sítě Internet, měnily by se výsledky v závislosti na denní době a vytížení jednotlivých síťových prvků mezi knihovnou a úložištěm. Účelem zde prezentovaných výsledku tedy není určit přesné hodnoty, ale pouze podložit funkčnost navrženého konceptu řešení.

Pro zpracování výsledků měření bylo použito statistických metod. Z naměřených časů jsou vypočteny dva výsledné časy. Jeden odpovídá aritmetickému průměru naměřených časů, zatímco druhý je jejich mediánem. Pro naměřené časy je rovněž vypočtena směrodatná odchylka [62]. Urychlení udává kolikrát rychlejší bylo stahování ve více vláknech oproti stahování v jednom vlákně.

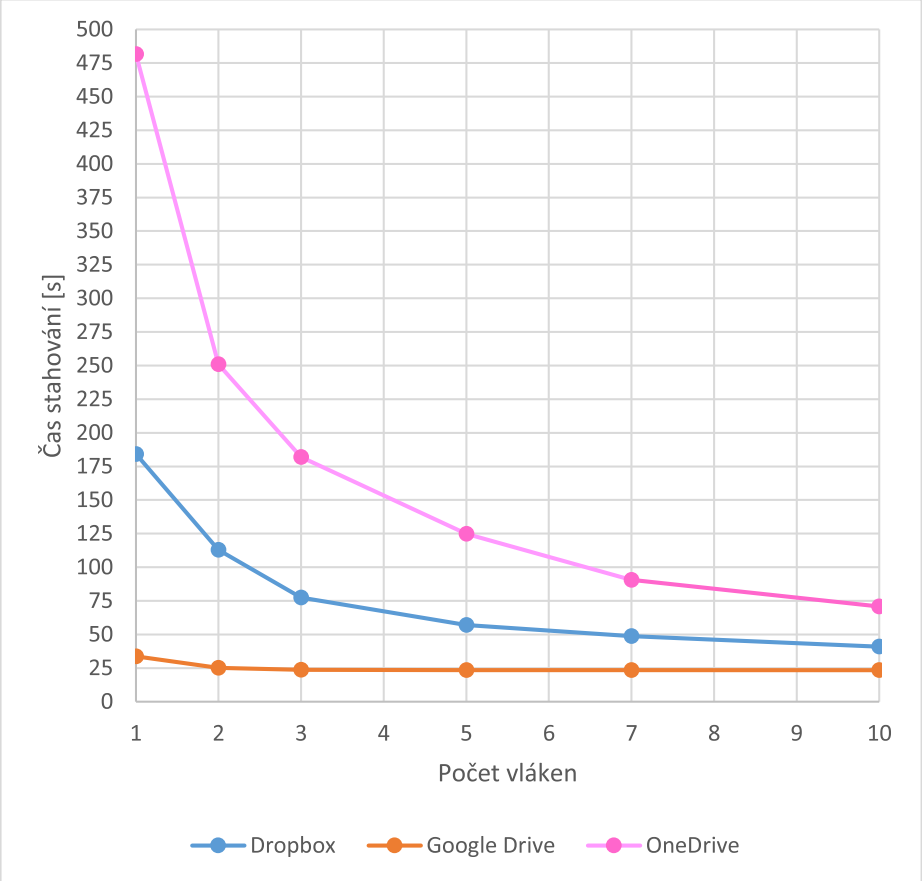
Vláken	**Úložiště**	**Výsledný čas [s]**		**Odchylka [s]**	**Urychlení**
		Průměr	**Medián**		
1	Dropbox	184,20	186,72	±8,12	———
	Google Drive	33,71	31,18	±6,39	———
	OneDrive	481,49	477,63	±19,56	———
2	Dropbox	112,80	113,06	±6,88	∼ 1,64×
	Google Drive	25,26	24,95	±0,89	∼ 1,30×
	OneDrive	250,92	250,63	±9,66	∼ 1,91×
3	Dropbox	77,32	75,36	±4,35	∼ 2,43×
	Google Drive	23,81	23,85	±0,32	∼ 1,36×
	OneDrive	181,87	182,41	±3,88	∼ 2,63×
5	Dropbox	56,95	58,09	±5,62	∼ 3,22×
	Google Drive	23,51	23,46	±0,28	∼ 1,38×
	OneDrive	124,86	122,60	±6,72	∼ 3,88×
7	Dropbox	48,65	48,66	±4,63	∼ 3,81×
	Google Drive	23,43	23,40	±0,19	∼ 1,39×
	OneDrive	90,61	89,91	±3,60	∼ 5,31×
10	Dropbox	40,90	41,56	±3,65	∼ 4,50×
	Google Drive	23,42	23,42	±0,16	∼ 1,39×
	OneDrive	70,80	70,35	±4,40	∼ 6,80×

Tabulka 6.1: Měření rychlosti stahování.

Z výsledků měření v tabulce 6.1 je vidět, že paralelní přístup k souborům v úložišti je možný a lze jej využít pro rychlejší stahování souborů. Navýšení rychlosti stahování pro Google Drive se zastavilo už při použití třech vláken.

Při použití více vláken se již jen stabilizovaly časy na hodnotě kolem 23, 45 sekund. Toto bylo způsobeno plným vytížením linky a měření tak bylo nutné zopakovat na lince s kapacitou 1 Gb/s. Při testování na 1 Gb/s lince se však ukázalo, že i přes navýšení kapacity linky byla rychlost stahování přibližně stejná.

Vynecháme-li výsledky pro Google Drive, můžeme říct, že s přibývajícími vlákny rostla i rychlost stahování. Při použití 10-ti vláken je rychlost stahování pro Dropbox navýšena na více než čtyřnásobek původní rychlosti. Pro OneDrive se navýšení rychlosti dostalo až téměř na sedmínásobek. Pro Google Drive je navýšení rychlosti stahování podstatně nižší, avšak i tak se jedná o navýšení oproti hodnotám dosaženým pro jedno vlákno. V tomto ohledu splnila knihovna veškerá očekávání.



Obrázek 6.3: Graf urychlení stahování.

7 Implementace klientské aplikace

Základem klientské aplikace je implementovaná knihovna, která poskytuje aplikaci základní funkcionalitu a některé rozšiřující funkce. Cílem implementace aplikace tak je poskytnout uživateli grafické uživatelské rozhraní (GUI) pro práci s úložišti a rozšířenou funkcionalitu, která není součástí knihovny.

7.1 Rozšířená funkcionalita

V rámci aplikace je implementována rozšířená funkcionalita, která je tvořena kontrolou konzistence dat, synchronizací složek a transferem souborů mezi úložišti.

7.1.1 Kontrola konzistence dat

Jádrem kontroly konzistence dat je třída `ChecksumProvider`, která má na starosti správu kontrolních součtů a jejich přiřazování k metadatům souborů z úložišť. Pro uchovávání kontrolních součtů je na lokálním disku vytvořen soubor `.multicloud`. Kontrolní součty jsou uchovávány ve formátu JSON. Stejný soubor je vytvořen i v kořenovém adresáři na každém cloudovém úložišti spravovaném v aplikaci. V případě poškození nebo smazání lokálního souboru je tak možné získat soubor zpět.

Další výhodou uložení souboru na cloudových úložištích je možnost jejich přenositelnosti na další uživatele a stroje. Pokud by tedy přistupoval uživatel ke svému úložišti z jiného stroje, získá kontrolní součty stažením jednoho souboru. Aby byly kontrolní součty přenositelné, je u každého záznamu namísto lokálního názvu účtu použito identifikátoru uživatele na úložišti.

V případě souběžného přístupu k úložišti z více instancí aplikace zároveň, je před téměř každou operací kontrolována aktuálnost souborů s kontrolními součty. Pokud je na některém z úložišť detekován novější soubor s kontrolními součty, je tento soubor stažen a sloučen s lokálním souborem.

Sloučení probíhá tak, že novější soubor je použit jako základ. Ze staršího souboru jsou pak postupně vybírány jednotlivé záznamy a kontrolována jejich

existence v novějším souboru. Pokud již v novějším souboru existují, jsou zahozeny. Pokud nikoliv, jsou do novějšího souboru přidány. Cílem sloučení je udržovat jen nejnovější verze záznamů, ale přitom nepřijít o žádné záznamy.

Naplnění souboru s kontrolními součty probíhá při každé operaci nahrávání souboru a transferu souborů mezi úložišti. Dále pak byla vytvořena speciální operace, která stáhne soubor z úložiště do dočasné složky a spočte jeho kontrolní součet. Stažený soubor je následně smazán.

Přiřazení kontrolního součtu probíhá na základě shody mezi uloženými metadaty v záznamu kontrolního součtu a metadaty získanými z úložiště. Kontrolované atributy souboru jsou popsány v sekci 5.3. Kontrolní součty pak hrají důležitou roli při stahování souboru z více úložišť zároveň. Postup stahování probíhá v tomto případě tak, že uživatel nejprve zvolí soubor, který chce stáhnout. Tento soubor je následně nutné najít i v dalších úložištích.

- Pokud vybraný soubor měl spočtený kontrolní součet, vyhledávají se v dalších úložištích pouze soubory se shodným kontrolním součtem nebo soubory bez kontrolního součtu se shodnou velikostí. Pokud je uživatelem vybrán soubor bez kontrolního součtu, je uživatel na tuto skutečnost před stahováním upozorněn. Obsah staženého souboru by ve výsledku nemusel odpovídat.
- Pokud vybraný soubor neměl spočtený kontrolní součet, je možno vybírat ze souborů se shodnou velikostí jako má původně vybraný soubor. Pokud ale některé z vybraných souborů mají kontrolní součty spočtené a navzájem se liší, není soubor možné stáhnout, protože některé ze souborů obsahují odlišná data.

7.1.2 Synchronizace složek

Další rozšířenou funkcionalitou, která je založena na kontrolních součtech, je synchronizace složek. Pro synchronizaci složek byl zvolen jednorázový běh synchronizačního procesu vzhledem k jeho jednodušší implementaci a prioritu dostala lokální složka. V nastavení programu je tak možné zvolit si složku z lokálního disku, která bude synchronizována. Součástí synchronizace je i možnost výběru, na které úložiště bude který soubor synchronizován. Výchozím stavem pro každý soubor pak je prázdná množina úložišť, na které bude synchronizován.

Proces synchronizace je zahájen zjištěním struktury souborů v lokální složce. Následně je pro tyto soubory vypočten kontrolní součet. Poté se pokračuje zjištěním struktury souborů ve složkách na vzdálených úložištích a dopočtení chybějících kontrolních součtů. Po těchto operacích může proběhnout porovnání kontrolních součtů a zjištění změn souborů.

V případě nalezení změny lokálního souboru, je tento soubor připraven k nahrání na úložiště. Pokud pak soubor v úložišti je beze změn, může proběhnout nahrání lokální verze na úložiště. Pokud je ale detekována změna na vzdáleném souboru, je vyžadován zásah uživatele, který může rozhodnout, který ze souborů bude přepsán, případně může rozhodnout o přeskočení synchronizace souboru. Jelikož byla zvolena strategie s prioritní lokální složkou, změna souboru ve vzdáleném úložišti nevyvolá automaticky stahování vzdáleného souboru. Namísto toho je opět na uživateli rozhodnout, který ze souborů bude přepsán.

Podobně je tomu i v případě mazání souborů. Smazání souboru v úložišti nemá za následek smazání souboru na lokálním disku. Pokud by tedy uživatel smazal soubor z úložiště a spustil proces synchronizace, soubor by byl na úložiště znovu nahrán. Obdobně je tomu i při smazání lokálního souboru. Při jeho smazání nedojde ke smazání souboru v úložišti. Důvodem pro toto chování je to, že k souboru v úložišti může být přistupováno z více zařízení. Pokud by tak byl smazán soubor v úložišti, byl by při synchronizaci dalšího zařízení opět na úložiště nahrán.

7.1.3 Transfer souborů

Tato rozšiřující funkcionalita je implementována pomocí operací `CUT`, `COPY` a `PASTE`. V případě vyvolání operace `CUT` nebo `COPY` jsou uložena do paměti metadata souboru, nad kterým byla tato operace vyvolána. Společně s metadaty je uchováno i úložiště, nad kterým byla tato operace provedena a typ operace. Operací `PASTE` nad jiným úložištěm je spuštěn proces transferu souborů mezi úložišti.

Tento proces probíhá tak, že je nejprve soubor stažen na lokální disk. Poté je vypočten jeho kontrolní součet a nakonec je soubor nahrán na cílové úložiště. Pokud se jednalo o přesun souboru, je původní soubor smazán. Operace mazání je prováděna jako poslední pro případ možné chyby v přenosu souboru.

7.2 Rozvržení aplikace

Aplikace je rozdělena do dvou hlavních částí. Základem je grafické rozhraní, které poskytuje uživateli okamžitou odezvu na prováděné akce. Aby však toto bylo možné, je na pozadí vytvořeno druhé vlákno (dále referováno jako *worker*), které má na starosti práci s knihovnou a volání jejích metod. Worker pak po dokončení operace posílá získaná data zpět grafické části, která je zobrazí.

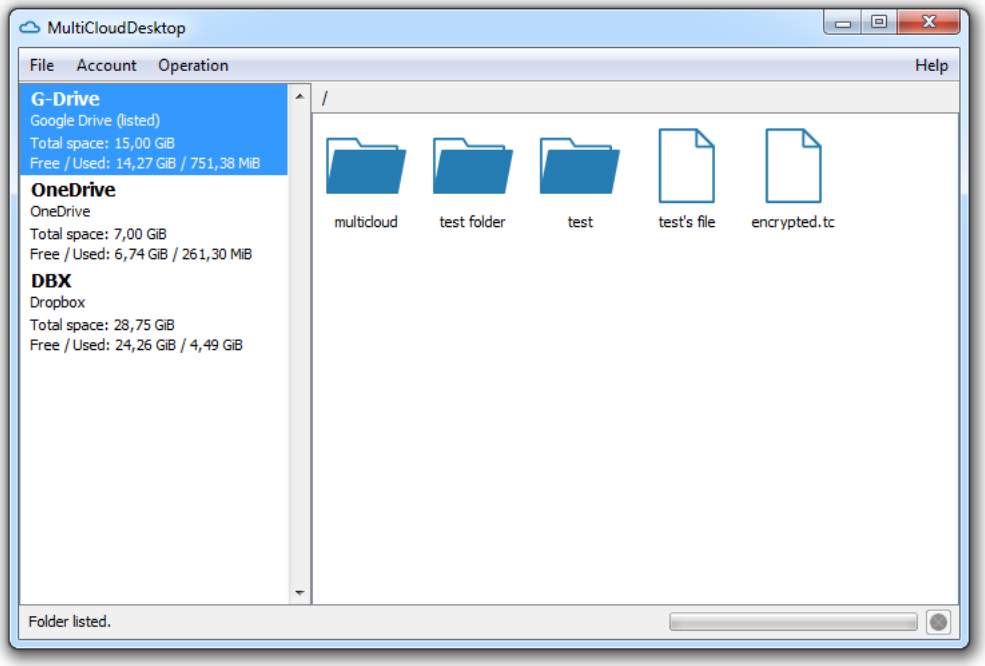
Celkově je aplikace rozdělena do 6 balíků, které obsahují dohromady 63 souborů. V těchto souborech je obsaženo celkem 59 tříd, 3 enumerace a jedno rozhraní. Rozsah aplikace podle metriky LOC převyšuje 11 tisíc řádek kódu. Z tohoto množství je přibližně 33% obsaženo v balíku `data`, 27% v balíku `dialog`, 16% v balíku `action`, 14% ve výchozím balíku a 5% v každém z balíků `renderer` a `callback` [66]. Následuje stručný popis jednotlivých balíků:

- `action` – Obsahem tohoto balíku jsou akce vyvolávané uživatelem z grafického rozhraní aplikace.
- `callback` – Tento balík obsahuje rozhraní pro zpětná volání z vlákna workera.
- `data` – Obsahem tohoto balíku jsou třídy pro reprezentaci dat a třídy pro zpracování dat. V tomto balíku je obsažena třída workera a třída pro práci s kontrolními součty.
- `dialog` – Již podle názvu balíku je patrné, že jeho obsahem jsou dialogová okna zobrazovaná aplikací.
- `renderer` – Tento balík v sobě zahrnuje třídy, které mají na starost zobrazování položek v seznamech typu `JList` a stromech typu `JTree`.

7.2.1 Grafické uživatelské rozhraní

Důležitou částí desktopové aplikace je právě grafické rozhraní, které umožňuje uživateli ovládat knihovnu a pracovat tak se soubory v cloudových úložištích. Další funkcí grafického rozhraní je zobrazovat uživateli data získaná z cloudových úložišť.

Podoba grafického rozhraní je zachycena na obrázku 7.1. V levé části tohoto okna je seznam uživatelských účtů, zatímco v pravé jsou zobrazeny soubory a složky vzdáleného úložiště. V horní části je pak nabídka operací, které je možné v aplikaci provádět, a v dolní části se zobrazují informace o výsledku operací a jejich průběhu. Kompletní uživatelská příručka aplikace je pak obsažena v příloze B.1.



Obrázek 7.1: Hlavní okno desktopové aplikace.

Hlavní třídou grafického rozhraní je třída `MultiCloudDesktop`, která rozšiřuje třídu `JFrame`. Třída `JFrame` je třídou z knihovny grafických prvků Swing a slouží pro vytvoření okna aplikace [67]. Obsahem tohoto okna jsou další grafické prvky z téže knihovny.

V horní části okna je nabídka aplikace, která je tvořena různými operacemi, které jsou rozřazeny do několika kategorií. Kategorie *Account* v sobě zahrnuje operace pro práci s uživatelskými účty, zatímco *Operation* obsahuje operace pro práci se soubory a složkami. Stejné nabídky se zobrazují i při kliknutí pravým tlačítkem na nějakou položku v příslušném seznamu pod nabídkou. Nabídka *File* umožňuje vyvolat nastavení aplikace a spustit proces synchronizace. Poslední nabídka *Help* v sobě nabízí položku pro zobrazení informací o aplikaci. Každé položce v nabídce je přiřazena jedna akce, která je provedena při aktivaci položky. Aktivaci některých položek je

taktéž možné provést pomocí klávesových zkratek, které jsou zobrazeny vedle názvu položky v hlavní nabídce. Spuštěná akce pak ve většině případů vyvolá dialogové okno.

Dialogová okna jsou součástí balíku `dialog` a slouží pro zobrazování a zadávání různých typů dat. Pomocí dialogových oken jsou tak shromážděna data potřebná pro vyvolání některé z metod knihovny. Shromážděná data jsou pak předána vláknu workera, který provede volání metody knihovny.

Zpětnou vazbou od workera jsou předána zpět data, která je potřeba uživateli zobrazit. K tomuto účelu slouží balík `renderer`, který obsahuje třídy pro vykreslování obsahu grafických prvků typu `JList` a `JTree`.

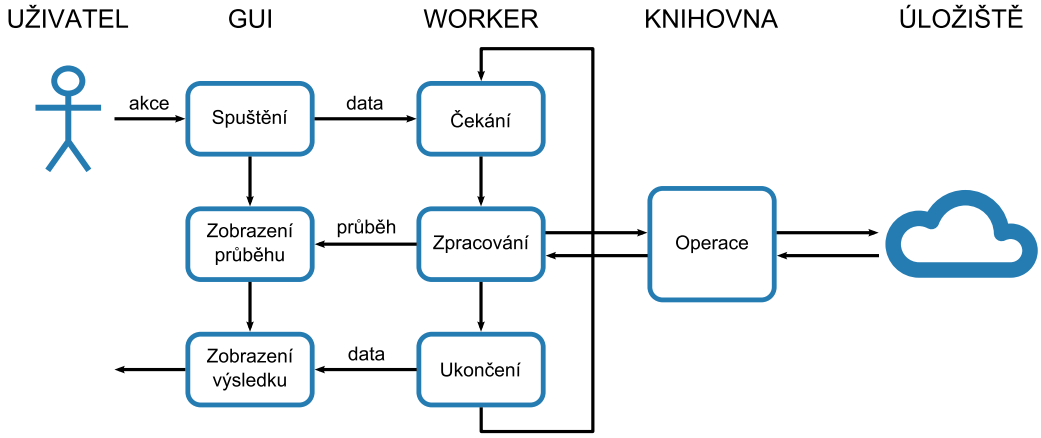
7.2.2 Zpracování na pozadí

Na pozadí běhu aplikace je spuštěno vlákno workera, který je pak zpětně provázán s grafickým rozhraním pomocí rozhraní pro zpětné vyvolání `BackgroundCallback<T>`, kde typovým parametrem `T` je typ návratových dat. Worker tak po dokončení volání knihovní metody vyvolá metodu `onFinish` poskytnuté implementace tohoto rozhraní a předá tak data grafickému rozhraní. Balíkem, ve kterém jsou obsažena návratová volání, je balík `callback`. Samotný worker je pak součástí balíku `data`, jehož obsahem jsou pak ještě třídy pro ukládání a správu dat aplikace.

Vlákno workera není aktivní po celou dobu. Worker je aktivován pouze v případě, kdy je uživatelem vyvolána některá akce vyžadující zpracování na pozadí. V tu chvíli jsou vláknu předána potřebná data a vlákno je probuzeno. Následuje zpracování dat vláknem a navrácení výsledku grafickému rozhraní. Po dokončení zpracování dat je vlákno opět uspáno a převedeno do stavu čekání na další data. Vláknu workera není možné předávat data v případě, že zrovna probíhá zpracování jiných dat. Zpracování dat je ale možné přerušit.

Na obrázku 7.2 je zobrazen schematický postup zpracování akce vyvolané uživatelem. Vyvoláním akce jsou na úrovni grafického rozhraní získána potřebná data prostřednictvím dialogových oken. Data jsou následně předána vláknu workera ke zpracování. Souběžně s tím je uživateli zobrazen grafický prvek pro sledování průběhu operace. Vlákno workera je probuzeno a zahajuje zpracování dat. Při zpracování jsou volány metody knihovny, prostřednictvím které je přistupováno k úložišti. Během zpracování předává worker grafickému rozhraní data o průběhu zpracování operace. Po dokon-

čení operace jsou grafickému rozhraní předána data s výsledkem operace. Výsledná data jsou následně prezentována uživateli. Vlákno workera je v tomto bodě opět uspáno a čeká na další data ke zpracování.



Obrázek 7.2: Zpracování akce vyvolané uživatelem.

8 Implementace mobilního klienta

Mobilní klient je rovněž postaven na implementované knihovně. Aby však bylo možné knihovnu použít i pro mobilní aplikaci, bylo nejprve nutné ověřit její funkčnost na zvolené platformě, tj. Android. Při prvním testování však byla zjištěna nekompatibilita knihovny s platformou Android. Tato nekompatibilita se týkala HTTP klienta použitého v knihovně. Problémem bylo, že na zvolené platformě je již dostupný HTTP klient od Apache, bohužel však v odlišné verzi než na jaké je závislá knihovna. Nebylo tak možné využít již existujícího HTTP klienta, jak bylo navrženo v sekci 6.1 a bylo nutné tento problém vyřešit jiným způsobem.

Při zkoumání HTTP klienta na platformě Android byla zjištěna verze odpovídající přibližně verzi `4.0-beta-1` [63, 64]. Knihovna však je závislá na verzi klienta `4.3.3`. Možností řešení tohoto problému bylo více. První možností byla změna závislostí knihovny a přepsání jejího kódu pro verzi HTTP klienta dostupné na platformě Android. Tato možnost by byla ale značně náročná na úpravy. Kromě knihovny by bylo nutné i upravit již vytvořenou desktopovou aplikaci a ověřit její funkčnost.

Druhou možností bylo stažení oficiálního balíku pro podporu novějších HTTP klientů na platformě Android přímo od Apache [64]. Tento balík pracuje tak, že novější třídy HTTP klienta verze `4.3.x` jsou přejmenovány a tudíž je možné se k nim dostat pod jiným názvem. Volba této možnosti také nebyla ideální, protože by opět bylo potřeba upravovat kód knihovny a případně i desktopové aplikace.

Poslední možností, která byla v rámci zkoumání tohoto problému nalezena, bylo použití knihovny Jar Jar Links během vytváření výstupního balíku knihovny. Knihovna Jar Jar Links při překladu a kompletaci výstupního `JAR` archivu dokáže vzít přeložené třídy změnit `package`, do kterého třídy patří [65]. Tato možnost nevyžaduje žádné úpravy existujícího kódu, proto také byla zvolena. Balík (`package`), do kterého patří třídy HTTP klienta, tak byl přejmenován z `org.apache.*` na `xorg.apache.*`. Výsledkem se tak stala knihovna nezávislá na implementaci HTTP klienta dostupného na platformě Android. Pro vytvoření výstupní verze knihovny byla použita verze `1.4` knihovny Jar Jar Links.

8.1 Vybraná funkcionalita

V rámci implementace klientské aplikace pro mobilní platformu Android byla zvolena pouze omezená množina navržených funkcí. Stejně tak byla omezena funkčnost některých funkcí. Důvodem této volby byly především omezené výpočetní prostředky platformy.

Pro implementaci byly zvoleny základní funkce pro procházení adresářové struktury vzdálených úložišť, nahrávání a stahování souborů a manipulaci se soubory a složkami v úložišti. Funkce pro kopírování, přesun a vyhledávání souborů byly vynechány. Z rozšířené funkcionality byly zvoleny funkce pro výpočet kontrolních součtů a synchronizaci složek. V rámci synchronizace je využito i nahrávání na více úložišť a stahování ve více vláknech. Rovněž je dostupná selektivní synchronizace. Rozšířená funkcionalita je v případě mobilního klienta převzata od desktopové aplikace (viz sekce 7.1) a poupravena pro použití na mobilní platformě.

8.2 Rozvržení aplikace

Mobilní aplikace je rozdělena do 4 balíků, s přibližným celkovým rozsahem 6000 řádek kódu dle metriky LOC [66]. Součástí těchto 4 balíků je 38 souborů, jejichž obsahem je 34 tříd, 2 rozhraní a 2 enumerace. Význam balíků je následující:

- `display` – Tento balík obsahuje grafické a datové prvky aplikace. Součástí balíku jsou třídy pro vykreslování položek v seznamech typu `ListView`, třídy implementující dialogová okna pro nastavení aplikace a třída zprostředkující ostatní dialogová okna aplikace.
- `fragment` – Obsahem tohoto balíku jsou jednotlivé fragmenty aplikace. Je zde umístěn fragment pro zobrazení uživatelských účtů, souborů a fragment pro nastavení aplikace.
- `tasks` – V tomto balíku jsou obsaženy asynchronní úlohy implementující jednotlivé operace prováděné nad úložištěm.
- *Výchozí balík* – Ve výchozím balíku jsou umístěny především třídy pro zpracování kontrolních součtů a třídy s aktivitami aplikace.

Rozložení kódu pak je přibližně 36% v balíku `display`, 28% ve výchozím balíku, 28% v balíku `tasks` a nakonec 8% v balíku `fragment`.

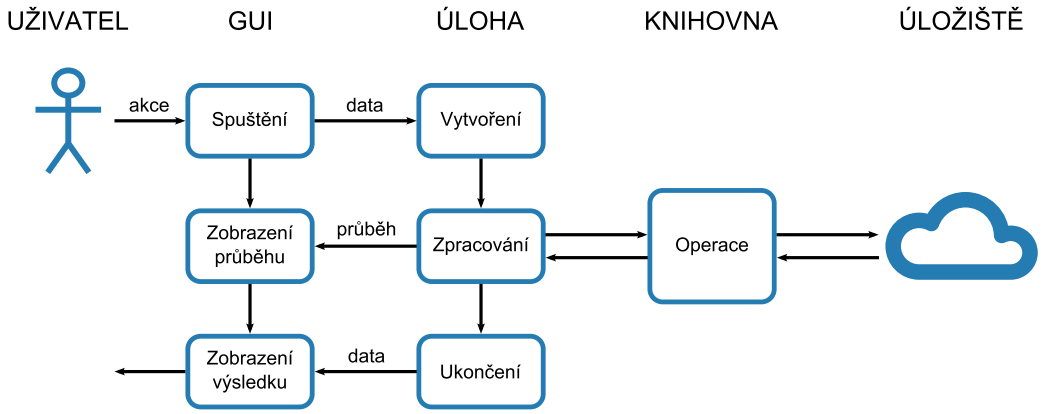
Základem mobilní aplikace jsou dvě aktivity. První aktivitou je hlavní aktivita, která jako svou součást zobrazuje fragmenty se seznamem uživatelů a seznamem souborů v úložišti. Současný návrh aplikace dovoluje zobrazení pouze jednoho fragmentu a tak jsou fragmenty dynamicky střídány. Druhou aktivitou je aktivita pro nastavení aplikace. Součástí této aktivity je pouze jeden fragment s nastavením.

Prvním z fragmentů hlavní aktivity je fragment sloužící pro zobrazování informací o uživatelských účtech. Z tohoto fragmentu je možné účty vytvářet, autorizovat, přejmenovávat a mazat. Ke každému účtu je možné rovněž vyvolat operaci pro získání informací o uživateli a dostupném prostoru. Druhý z fragmentů je určen pro procházení adresářové struktury na úložišti. Z tohoto fragmentu je možné nahrávat a stahovat soubory, mazat a přejmenovávat soubory i složky a vytvářet nové složky. Dále je zde možnost spuštění výpočtu kontrolního součtu souboru.

Oba fragmenty rozšiřují `ListFragment` a využívají vlastní `layout` pro zobrazení položek seznamu. U obou fragmentů je rovněž využito zobrazení nabídky s ikonami ve spodní části obrazovky a kontextové nabídky vyvolané dlouhým stiskem položky seznamu.

Další částí aplikace je nabídka s nastavením aplikace. Pro nastavení počtu vláken, synchronizační složky a selektivní synchronizace bylo nutné vytvoření vlastních dialogových oken. Nejdůležitějším dialogovým oknem je okno pro výběr lokálních souborů a složek. Vlastností tohoto dialogu je možnost procházení lokální adresářové struktury. Tento dialog je v pozměněné formě použit i pro operace nahrávání a stahování souboru.

Všechny operace implementované v aplikaci jsou vytvořeny jako asynchronní úlohy běžící na pozadí. Na popředí je vždy zobrazeno dialogové okno informující o průběhu operace, pomocí kterého je operaci možné přerušit. Asynchronní úlohy tak rozšiřují třídu `AsyncTask`, která je dělena do třech fází. V první fázi je provedena příprava operace. Druhou fází je provedení operace na pozadí. Poslední fází je pak ukončení operace. První a poslední fáze navíc mohou přistupovat ke grafickému rozhraní a zobrazovat tak uživateli informace o výsledcích operace. Pro zobrazování průběhu operace je pak připravena metoda `onProgressUpdate`.



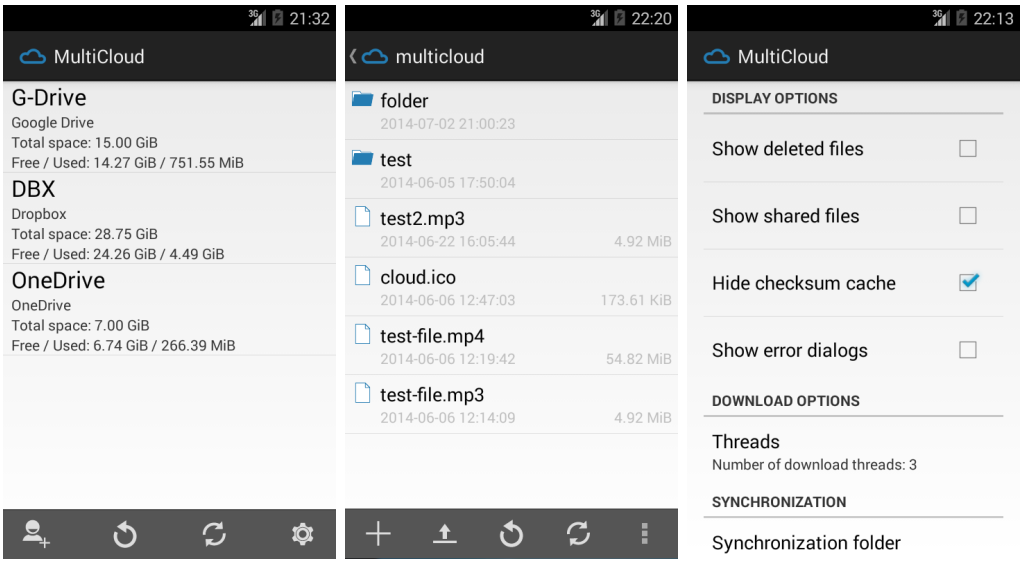
Obrázek 8.1: Zpracování akce vyvolané uživatelem v mobilním klientu.

Obrázek 8.1 slouží jako ilustrace průběhu zpracování akce vyvolané uživatelem. Vyvolaná akce spustí skrze grafické rozhraní dialogové okno pro zobrazení průběhu operace a asynchronní úlohu na pozadí. Úloze jsou předána potřebná data pro provedení operace. Úloha pak spustí svůj běh, během kterého vyvolává metody knihovny a zároveň informuje uživatele o průběhu operace. Po dokončení operace předá úloha grafickému rozhraní získaná data a ukončí se. S každou akcí uživatele je tak vyvolána nová úloha.

Vzhledem k omezené velikosti paměti dostupné pro aplikaci byl maximální počet vláken omezen na 5. Další omezení je aplikováno na operaci synchronizace, u které není vyvoláván dialog v případě konfliktů mezi soubory. Namísto toho je pouze zobrazeno dialogové okno po dokončení synchronizace, ve kterém jsou vypsány soubory, které nebylo možné synchronizovat. Na uživateli je pak ponecháno vyřešení problémů s těmito soubory.

8.3 Ukázka aplikace

Na obrázcích 8.2 jsou zobrazeny ukázky z mobilní aplikace. Obrázek 8.2(a) zachycuje fragment pro zobrazení uživatelských účtů. Na obrázku 8.2(b) je ukázka fragmentu pro zobrazování obsahu vzdáleného úložiště. Obrázek 8.2(c) pak představuje obrazovku s nastavením aplikace. Uživatelská příručka aplikace je pak obsažena v příloze B.2.



(a) Uživatelské účty. (b) Prohlížení souborů. (c) Nastavení.

Obrázek 8.2: Ukázky obrazovek mobilní aplikace.

9 Možnosti rozšíření

Vytvořenou knihovnu i aplikace by bylo možné dále rozšiřovat. Některá zde navržená rozšíření by bylo možné implementovat buď na straně knihovny nebo aplikace, jiná nikoliv. V závěru je navrženo rozšíření celého konceptu o webovou službu.

Knihovna

Možným rozšířením na straně knihovny by byla implementace transferu souborů mezi úložišti v rámci knihovny nebo alespoň jeho částí. Částí pro transfer souborů se rozumí podpora sekvenčního stahování souboru, jehož výstupem by byl proud dat, který by bylo možné rozdělit do proudů pro výpočet kontrolního součtu a proudu pro nahrávání souboru. Další takovou částí by pak nutně byla i podpora pro nahrávání datového proudu na úložiště.

Dalším rozšířením knihovny by mohla být podpora zpracování více operací souběžně. V dosavadní implementaci je podpora pouze jedné probíhající operace. Aby bylo možné takto knihovnu rozšířit, bylo by potřeba zajistit, aby neprobíhaly souběžně dvě operace nad jedním úložištěm, resp. jedním souborem. Výlučný přístup k úložištím je zvláště důležitý především pokud jde o operace nahrávání a stahování souborů.

Pro rozšíření možnosti znovupoužitelnosti knihovny by bylo knihovnu možné rozdělit do několika menších knihoven. Mohla by tak vzniknout například samostatná knihovna s implementací OAuth 2.0 autorizace.

Desktopová aplikace

Pro desktopovou aplikaci by bylo možné navrhnout a implementovat celou řadu nových rozšíření. Jako součást aplikace by mohl například vzniknout synchronizační klient, který by sledoval změny na lokálním disku a propagoval je na vzdálená úložiště. V aplikaci by pak mohla vzniknout podpora plánování synchronizačních procesů. Vhodné by pro tento účel bylo časové plánování, u kterého by bylo možné zvolit datum a čas příští synchronizace. Možností by bylo i například synchronizovat soubor až po uplynutí předem stanovené doby od jeho poslední změny. Upravovaný soubor by tak nebyl nahrán na úložiště při každém uložení změn, ale mohl by být nahrán až po dokončení práce s ním.

Rozšířit aplikaci by bylo možné i mimo rámec cloudových úložišť. Zde se naskýtá možnost využití jiných protokolů pro přístup a manipulaci se soubory. Jedním takovým protokolem je například FTP. Aplikace by tak získala možnost přístupu jak k veřejným, tak i privátním FTP serverům. Stejně tak by mohla vzniknout například podpora protokolu OAuth 1.0 pro autorizaci. Realizaci těchto rozšíření by pak bylo vhodné provést přiložením dalších knihoven implementujících zvolené protokoly.

Mobilní klient

Možností rozšíření mobilní aplikace by byla implementace dosud vynechané funkcionality. Pro toto rozšíření by bylo ovšem vhodné nejprve rozšířit knihovnu o podporu přenosu souborů v datových proudech. Bez této podpory by mohl být v určitých případech přenos velkých souborů problematický.

Pro mobilní aplikaci by bylo rovněž možné rozšíření o další protokoly pro přístup k úložištím a manipulaci se soubory.

Vhodné rozšíření pro mobilní aplikaci by bylo sledování stavu baterie a dostupného přípojení k síti Internet. V případě nízkého stavu baterie by tak například nebylo možné spustit synchronizaci. Naopak, při připojení mobilní platformy ke zdroji napájení a k bezdrátové síti WiFi by mohla být provedena synchronizace automaticky.

9.1 Webová služba

Po vzoru existujících aplikací třetích stran pro přístup ke cloudovým úložištím by bylo možné vytvořit webovou službu, se kterou by komunikovaly obě aplikace. Účelem webové služby by byl přesun a kopírování souborů mezi úložišti. Z klientské aplikace by tak stačilo odeslat požadavek na spuštění vybrané operace a ta by byla provedena bez nutnosti přenosu souboru skrze klienta.

Další funkcionalitou, kterou by webová služba mohla mít, by byla synchronizace souborů mezi úložišti. Tato synchronizace by navíc mohla být spouštěna dle předem nastaveného časového plánu bez nutnosti zásahu uživatele. Součástí webové služby by tak mělo být i webové rozhraní pro přístup prostřednictvím prohlížeče s možností nastavení všech potřebných parametrů.

Společně se synchronizací by ve službě mohla být udržována i aktuální metadata a služba by tak mohla řídit i stahování a nahrávání souborů klienta. V případě stahování by tak stačilo z klientské aplikace poslat požadavek s vybraným souborem a služba by se postarala o nalezení shodného souboru i v dalších úložištích. Aplikaci by pak byly vráceny pouze zdroje, ze kterých má soubor stahovat. Pro nahrávání souboru na více úložišť by pak mohlo být implementováno nahrání souboru pouze na webovou službu, která by soubor sama rozkopírovala na vybraná úložiště.

Vytvořením webové služby by se tak mohl snížit celkový datový tok klientské aplikace. To by bylo zvláště výhodné pro mobilní klienty, kde bývají datové přenosy limitované signálem, operátorem a výdrží baterie.

10 Závěr

Diplomová práce se zabývala problematikou cloudových úložišť. Nejprve byla prozkoumána vybraná cloudová úložiště se zaměřením na principy jejich fungování. Následoval průzkum dostupných knihoven pro přístup k úložištím. Na základě jejich důkladné analýzy pak bylo zvoleno vlastní řešení pro přístup ke cloudovým úložištím.

V rámci práce byla vytvořena knihovna pro jednotný přístup ke všem vybraným úložištím. Tato knihovna není závislá na žádné jiné knihovně pro přístup ke cloudovým úložištím. Výhodou této knihovny je především možnost jejího použití pro další cloudová úložiště bez nutnosti úpravy zdrojového kódu knihovny. Další výhodou oproti existujícím knihovnám je možnost souběžného nahrávání souborů na více úložišť zároveň. Součástí knihovny je i implementace paralelního stahování souborů, které je popsáno v sekci 5.2. Funkcionalita paralelního stahování byla následně ověřena v sekci 6.4.

S využitím vytvořené knihovny byla dále navržena a implementována klientská aplikace pro přístup k vybraným cloudovým úložištím. V rámci klientské aplikace byla implementována i další navržená rozšíření z kapitoly 5. Kromě základních operací se soubory podporuje klientská aplikace kontrolu konzistence dat, synchronizaci složek a transfer souborů mezi jednotlivými úložišti. Implementovanou funkcionalitou se aplikace vyrovná existujícím řešením, které v některých aspektech dokonce předčí.

Pro mobilní platformu Android byla vyřešena kompatibilita knihovny. Na základě knihovny pak byla vytvořena mobilní aplikace s vybranou funkcionalitou ze sekce 8.1. Výsledná mobilní aplikace v sobě kombinuje vybranou množinu základních a rozšiřujících funkcí.

Implementací byla ověřena funkcionalita na zvolených platformách. Výstupem práce je tak jednotná knihovna použitelná na desktopové i mobilní platformě a dvě aplikace využívající knihovnu na zvolených platformách. Knihovnu i aplikace by bylo dále možné rozšířit o další funkcionalitu, případně by bylo možné vytvořit webovou službu, která by dále rozšiřovala možnosti správy souborů v cloudových úložištích.

Přehled zkratek

AES (Advanced Encryption Standard)
Algoritmus pro šifrování dat, který využívá symetrického klíče.

API (Application Programming Interface)
Rozhraní pro programování aplikací, často rozhraní pro volání funkcí knihovny.

CBC (Cipher block chaining)
Mód operace blokové šifry, kdy šifrování bloků je zřetězeno.

CRUD (create, read, update, delete)
Množina operací definovaná pro manipulaci s daty pomocí RESTu.

CTR (Counter mode)
Mód operace blokové šifry, u kterého je bloková šifra převedena na proudovou, a využívá inkrementačního čítače.

ECB (Electronic Codebook)
Mód operace blokové šifry, kde každý block je šifrován zvlášť.

ECMA (European Computer Manufacturers Association)
Organizace zabývající se tvorbou standardů informačních a komunikačních systémů.

FTP (File Transfer Protokol)
Protokol pro přenos souborů.

GUI (Graphical User Interface)
Grafické rozhraní pro interakci mezi uživatelem a aplikací.

HTTP (Hypertext Transfer Protokol)
Aplikační protokol pro výměnu hypertextových dokumentů.

IaaS (Infrastructure as a Service)
Model cloudové služby, kdy poskytovatel poskytuje infrastrukturu.

IETF (Internet Engineering Task Force)
Organizace zabývající se tvorbou Internetových standardů a vývojem protokolů.

IT (Informational Technology)
Technické odvětví zaměřující se na počítače a jejich fungování po technické stránce.

ITU (International Telecommunication Union)
Mezinárodní telekomunikační unie, připravuje specifikace pro telekomunikační systémy, sítě a služby.

JSON (JavaScript Object Notation)
Jednoduchý, textový formát pro výměnu dat nezávislý na programovacím jazyce.

LOC (Lines of Code)
Metrika pro měření rozsahu zdrojového kódu programu založená na počítání počtu řádků kódu.

MIME (Multipurpose Internet Mail Extensions)
Identifikátor formátu přenášených dat.

OFB (Output feedback mode)
Mód operace blokové šifry, u kterého je bloková šifra převedena na proudovou, a využívá pro šifrování i dešifrování operace exkluzivní disjunkce.

PaaS (Platform as a Service)
Model cloudové služby, kdy je poskytovatelem poskytnuta platforma s předem připravenými prostředky pro tvorbu aplikací.

PHP (Hypertext Preprocesor)
Hypertextový preprocesor je skriptovacím jazykem používaným na webových serverech pro tvorbu dynamických stránek.

POJO (Plain Old Java Object)
Jednoduché Java objekty pro uchovávání dat, které jsou mapované JSON parserem.

REST (Representational State Transfer)
Architektura rozhraní pro přístup k datovým zdrojům.

RFC (Request for Comments)
Dokumety popisující Internetové protokoly, které jsou vydávány jako doporučení a podle kterých se řídí většina Internetu.

SaaS (Software as a Service)
Model cloudové služby, kdy je poskytovatelem poskytnut přístup k hotové aplikaci.

SDK (Software Development Kit)
Balík nástrojů pro vývoj aplikací.

URI (Uniform Resource Identifier)
Jednotný identifikátor zdroje sloužící k přesnému určení zdroje informací, především v rámci sítě Internet.

XML (Extensible Markup Language)
Značkovací jazyk užívaný pro výměnu dat mezi aplikacemi.

Literatura

[1] Sosinsky, B., *Cloud Computing Bible*. Wiley, 2010, 532 s. ISBN: 978-0-470-90356-8.

[2] Kulkarni, G.; Waghmare, R.; Palwe, R.; Waykule, V.; Bankar, H.; Koli, K., Cloud storage architecture, *Telecommunication Systems, Services, and Applications (TSSA), 2012 7th International Conference on* [online], pp. 76-81, 30-31.10.2012 [cit. 11.3.2014]. DOI: 10.1109/TSSA.2012.6366026 Dostupné z: `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6366026`.

[3] Telecommunication standardization sector of ITU, Focus Group on Cloud Computing Technical Report, *Part 1: Introduction to the cloud ecosystem: definitions, taxonomies, use cases and high-level requirements* [online], 69 s., 2012 [cit. 12.3.2014]. Dostupné z: `http://www.itu.int/ITU-T/newslog/Cloud+Computing+And+Standardization+Technical+Reports+Published.aspx`.

[4] Dropbox, Inc., *Terms - Dropbox* [online]. 2014 [cit. 20.4.2014]. Dostupné z: `https://www.dropbox.com/privacy`.

[5] Google Inc., *Smluvní podmínky společnosti Google – Ochrana soukromí a smluvní podmínky – Google* [online]. 2014 [cit. 20.4.2014]. Dostupné z: `http://windows.microsoft.com/cs-cz/windows/microsoft-services-agreement`.

[6] Microsoft, *Smlouva o poskytování služeb společnosti Microsoft - Microsoft Windows* [online]. 2014 [cit. 20.4.2014]. Dostupné z: `https://www.google.com/intl/cs/policies/terms/`.

[7] Dropbox, Inc., *Will joining someone else's shared folder use my quota? - Dropbox* [online]. 2014 [cit. 20.6.2014]. Dostupné z: `https://www.dropbox.com/help/59/en`.

[8] Google Inc., *Your storage limit - Drive Help* [online]. 2014 [cit. 26.4.2014]. Dostupné z: `https://support.google.com/drive/answer/6558`.

[9] Dropbox, Inc., *Is there a limit or maximum to how big my files can be? - Dropbox* [online]. 2014 [cit. 16.5.2014]. Dostupné z: `https://www.dropbox.com/help/5/en`.

[10] Dropbox, Inc., *Developer guide - Dropbox* [online]. 2014 [cit. 21.4.2014]. Dostupné z: https://www.dropbox.com/developers/reference/devguide.

[11] Google Inc., *Google Docs, Sheets, and Slides size limits - Drive Help* [online]. 2014 [cit. 16.5.2014]. Dostupné z: https://support.google.com/drive/answer/37603.

[12] Microsoft, *OneDrive Maximum Individual File Size - Microsoft Community* [online]. 2014 [cit. 16.5.2014]. Dostupné z: http://answers.microsoft.com/en-us/onedrive/forum/sdfiles-sdsync/onedrive-maximum-individual-file-size/2f8b752b-cf16-481c-961e-42cfb57801cb.

[13] D. Hardt, Ed., *RFC 6749 - The OAuth 2.0 Authorization Framework* [online]. Oct. 2012 [cit. 9.4.2014]. ISSN: 2070-1721. Dostupné z: http://tools.ietf.org/html/rfc6749.

[14] *OAuth 2.0 - OAuth* [online]. [cit. 9.4.2014]. Dostupné z: http://oauth.net/2/.

[15] E. Hammer-Lahav, Ed. *RFC 5849 - The OAuth 1.0 Protocol* [online]. Apr. 2010 [cit. 9.4.2014]. ISSN: 2070-1721. Dostupné z: http://tools.ietf.org/html/rfc5849.

[16] M. Jones, D. Hardt, *RFC 6750 - The OAuth 2.0 Authorization Framework: Bearer Token Usage* [online]. Oct. 2012 [cit. 9.4.2014]. ISSN: 2070-1721. Dostupné z: http://tools.ietf.org/html/rfc6750.

[17] Ugo Cei, Piergiorgio Lucidi, *Overview of REST Concepts and Developing your First Web Script using Alfresco | Packt Publishing* [online]. 2010 [cit. 23.5.2014]. Dostupné z: http://www.packtpub.com/article/overview-rest-concepts-developing-first-web-script-alfresco.

[18] M. Malý, *REST: architektura pro webové API | Zdroják* [online]. 3.8.2009 [cit. 23.5.2014]. Dostupné z: http://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/.

[19] *JSON* [online]. 2012 [cit. 10.5.2014]. Dostupné z: http://json.org/.

[20] T. Bray, Ed., *RFC 7159 - The JavaScript Object Notation (JSON) Data Interchange Format* [online]. Mar. 2014 [cit. 10.5.2014]. ISSN: 2070-1721. Dostupné z: http://tools.ietf.org/html/rfc7159.

[21] *The JSON DataInterchange Format* [online]. 2013 [cit. 10.5.2014]. Dostupné z: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

[22] Dropbox, Inc., *Dropbox - Core API* [online]. 2014 [cit. 22.4.2014]. Dostupné z: https://www.dropbox.com/developers/core.

[23] Google Inc., *Google Drive SDK – Google Developers* [online]. 2014 [cit. 22.4.2014]. Dostupné z: https://developers.google.com/drive/web/downloads.

[24] Microsoft, *OneDrive Developer Downloads - OneDrive Dev Center* [online]. 2014 [cit. 22.4.2014]. Dostupné z: http://msdn.microsoft.com/en-us/onedrive/dn630256.aspx.

[25] Microsoft, *Interaktivní sada Live SDK* [online]. 2014 [cit. 22.4.2014]. Dostupné z: http://isdk.dev.live.com/dev/isdk/Default.aspx.

[26] Microsoft, *OneDrive development - OneDrive Dev Center* [online]. 2014 [cit. 22.4.2014]. Dostupné z: http://msdn.microsoft.com/en-us/library/dn641952.aspx.

[27] Dropbox, Inc., *Dropbox - Core API - endpoint reference* [online]. 2014 [cit. 22.4.2014]. Dostupné z: https://www.dropbox.com/developers/core/docs.

[28] Google Inc., *API Reference - Google Drive SDK – Google Developers* [online]. 2014 [cit. 22.4.2014]. Dostupné z: https://developers.google.com/drive/v2/reference/.

[29] Microsoft, *REST reference - OneDrive Dev Center* [online]. 2014 [cit. 22.4.2014]. Dostupné z: http://msdn.microsoft.com/en-us/library/dn631844.aspx.

[30] Microsoft, *OneDrive API (Live Connect)* [online]. 2014 [cit. 22.4.2014]. Dostupné z: http://msdn.microsoft.com/en-us/library/hh826521.aspx.

[31] Microsoft, *Using the REST API - OneDrive Dev Center* [online]. 2014 [cit. 22.4.2014]. Dostupné z: http://msdn.microsoft.com/en-us/library/dn659752.aspx.

[32] The Apache Software Foundation, *Licenses* [online]. 2012 [cit. 26.4.2014]. Dostupné z: http://www.apache.org/licenses/.

[33] The Apache Software Foundation, *Apache HttpComponents - Apache HttpComponents* [online]. 2014 [cit. 12.5.2014]. Dostupné z: `http://hc.apache.org/`.

[34] Yidong Fang, *json-simple - JSON.simple - A simple Java toolkit for JSON - Google Project Hosting* [online]. 2012 [cit. 12.5.2014]. Dostupné z: `https://code.google.com/p/json-simple/`.

[35] FasterXML, LLC, *Jackson JSON Processor - Home* [online]. 2014 [cit. 12.5.2014]. Dostupné z: `http://jackson.codehaus.org/`.

[36] Hobara Rei, *Rei's Shed - WebDAV Client CarotDAV -* [online]. 2014 [cit. 14.6.2014]. Dostupné z: `http://rei.to/carotdav_en.html`.

[37] Vehera LTD, *Storage Made Easy* [online]. 2014 [cit. 14.6.2014]. Dostupné z: `http://storagemadeeasy.com/`.

[38] CloudKafé, *CloudKafé | Organizing your Cloud!* [online]. 2013 [cit. 14.6.2014]. Dostupné z: `https://www.cloudkafe.com/`.

[39] Jolicloud, *Jolicloud* [online]. 2014 [cit. 14.6.2014]. Dostupné z: `http://www.jolicloud.com/`.

[40] Primadesk Inc., *Primadesk* [online]. 2012 [cit. 14.6.2014]. Dostupné z: `https://www.primadesk.com/`.

[41] MultCloud, *Manage, Move, Copy, and Migrate Files Between Cloud Storage Services with MultCloud* [online]. 2014 [cit. 14.6.2014]. Dostupné z: `https://www.multcloud.com/`.

[42] cloudHQ, *cloudHQ - Sync and Integrate Google Drive, Gmail, Dropbox, Box, OneDrive, Evernote, Basecamp* [online]. 2014 [cit. 14.6.2014]. Dostupné z: `https://www.cloudhq.net/dropbox`.

[43] IFTTT Inc., *Welcome - IFTTT* [online]. 2014 [cit. 14.6.2014]. Dostupné z: `https://ifttt.com/`.

[44] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, *RFC 2616 - Hypertext Transfer Protocol – HTTP/1.1* [online]. Jun. 1999 [cit. 22.4.2014]. Dostupné z: `http://tools.ietf.org/html/rfc2616`.

[45] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee, *RFC 2068 - Hypertext Transfer Protocol – HTTP/1.1* [online]. Jan. 1997 [cit. 22.4.2014]. Dostupné z: `http://tools.ietf.org/html/rfc2068`.

[46] R. Fielding, Ed., Y. Lafon, Ed., J. Reschke, Ed. *RFC 7233 - Hypertext Transfer Protocol (HTTP/1.1): Range Requests* [online]. Mar. 2014 [cit. 14.6.2014]. ISSN: 2070-1721. Dostupné z: `http://tools.ietf.org/html/rfc7233`.

[47] Katz, Jonathan, Lindell, Yehuda, *Introduction to modern cryptography.* Boca Raton: Chapman & Hall/CRC, 2008, xviii, 534 s. ISBN 978-1-58488-551-1.

[48] National Institute of Standards and Technology, *ADVANCED ENCRYPTION STANDARD (AES)* [online]. 26.11.2001 [cit. 1.5.2014]. Dostupné z: `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`.

[49] Gary C. Kessler, *An Overview of Cryptography* [online]. 2014 [cit. 1.5.2014]. Dostupné z: `http://www.garykessler.net/library/crypto.html`.

[50] John J. G. Savard, *Basic Block Cipher Modes* [online]. 2003 [cit. 1.5.2014]. Dostupné z: `http://www.quadibloc.com/crypto/co040601.htm`.

[51] Javamex UK, *Block modes with Java ciphers* [online]. 2012 [cit. 1.5.2014]. Dostupné z: `http://www.javamex.com/tutorials/cryptography/block_modes.shtml`.

[52] Google Inc., *google-http-java-client - Google HTTP Client Library for Java - Google Project Hosting* [online]. 2014 [cit. 26.5.2014]. Dostupné z: `https://code.google.com/p/google-http-java-client/`.

[53] Google Inc., *google-gson - A Java library to convert JSON to Java objects and vice-versa - Google Project Hosting* [online]. 2014 [cit. 23.5.2014]. Dostupné z: `https://code.google.com/p/google-gson/`.

[54] *Json Java parsers / generators microbenchmark Myšlenky dne otce Fura* [online]. 18.3.2012 [cit. 22.5.2014]. Dostupné z: `http://blog.novoj.net/2012/02/05/json-java-parsers-generators-microbenchmark/`.

[55] Adamek, M., *JSON Parsers Performance on Android (With Warmup and Multiple Iterations) - Martin Adamek* [online]. 2013 [cit. 22.5.2014]. Dostupné z: `http://martinadamek.com/2011/02/04/json-parsers-performance-on-android-with-warmup-and-multiple-iterations/`.

[56] *Java JSON Libraries Comparison* [online]. 21.1.2011 [cit. 22.5.2014]. Dostupné z: `http://geokoder.com/java-json-libraries-comparison`.

[57] *Open Source Web Servers in Java* [online]. [cit. 5.5.2014]. Dostupné z: `http://java-source.net/open-source/web-servers`.

[58] *Simple 5.1.6* [online]. [cit. 5.5.2014]. Dostupné z: `http://www.simpleframework.org/index.php`.

[59] Dropbox, Inc., *App Console - Dropbox* [online]. 2014 [cit. 27.4.2014]. Dostupné z: `https://www.dropbox.com/developers/apps`.

[60] Google Inc., *Google Developers Console* [online]. 2014 [cit. 27.4.2014]. Dostupné z: `https://console.developers.google.com/project`.

[61] Microsoft, *Overview - Windows* [online]. 2014 [cit. 27.4.2014]. Dostupné z: `https://account.live.com/developers/applications/index`.

[62] Reif, J., *Metody matematické statistiky.* Fakulta aplikovaných věd, Plzeň, 2004, 288 s. ISBN: 80-7043-302-7.

[63] *What version of Apache HttpClient is used in Android 4.2.2 SDK? - Stack Overflow* [online]. 2013 [cit. 12.6.2014]. Dostupné z: `http://stackoverflow.com/questions/15658678/what-version-of-apache-httpclient-is-used-in-android-4-2-2-sdk`.

[64] The Apache Software Foundation, *Apache HttpComponents - HttpClient for Android* [online]. 2014 [cit. 12.6.2014]. Dostupné z: `https://hc.apache.org/httpcomponents-client-4.3.x/android-port.html`.

[65] Google Inc., *jarjar - Embedding Java libraries since 2004 - Google Project Hosting* [online]. 2014 [cit. 12.6.2014]. Dostupné z: `https://code.google.com/p/jarjar/`.

[66] *LocMetrics - Source Code Line Counting Tool* [online]. 2010 [cit. 10.5.2014]. Dostupné z: `http://www.locmetrics.com/`.

[67] Oracle, *Trail: Creating a GUI With JFC/Swing (The Java™ Tutorials)* [online]. 2014 [cit. 23.4.2014]. Dostupné z: `http://docs.oracle.com/javase/tutorial/uiswing/index.html`.

Přílohy

Příloha A

A.1 Dokumentace knihovny

V této příloze jsou dokumentovány poskytované metody knihovny. Dále jsou zde popsána rozhraní, která knihovna poskytuje, a způsob jejich využití. Nedílnou součástí knihovny je i popis konfiguračního souboru pro přístup k úložištím. Aktuální verzi knihovny je možno získat na adrese:

`https://github.com/stanek0j/MultiCloud`

A.1.1 Metody knihovny

Základem pro využití metod knihovny je vytvoření instance třídy `MultiCloud`. Tato třída poskytuje dva konstruktory, kde první je prázdný a druhému je předáván parametr s nastavením knihovny. Nastavení knihovny se provádí předáním instance třídy `MultiCloudSettings`, jejímž obsahem jsou objekty pro správu účtů, pro správu tokenů a pro správu definic úložišť (viz sekce A.1.3).

Metody knihovny je možno rozdělit do několika kategorií podle jejich zaměření. Jsou tak přítomny metody pro nastavení knihovny, správu uživatelských účtů nebo přístup k úložišti. Následuje rozdělení metod podle jejich kategorického zaměření. Po tomto rozdělení následuje výčet všech metod knihovny s předávanými parametry a návratovými hodnotami.

Nastavení knihovny

- getSettings

Správa uživatelských účtů

- abortAuthorization
- authorizeAccount
- createAccount
- deleteAccount
- refreshAccount
- renameAccount
- validateAccounts

Přístup k úložišti

- abortOperation
- accountInfo
- accountQuota
- addDownloadSource
- addUpdateSource
- addUploadSource
- copy
- createFolder
- delete
- downloadFile
- downloadMultiFile
- getLastError
- getListener
- listFolder
- metadata
- move
- rename
- search
- setListener
- updateFile
- updateMultiFile
- uploadFile
- uploadMultiFile

Popis jednotlivých metod

- `void abortAuthorization()`
  Metoda pro přerušení běžícího autorizačního procesu.

- `void abortOperation()`
  Metoda pro přerušení probíhající operace.

- `AccountInfo accountInfo(String accountName)`
  Metoda pro získání základních informací o uživateli, mezi které patří jméno uživatele a jeho identifikátor. Parametrem je název uživatelského účtu. Návratovou hodnotou jsou základní informace o uživateli.

- `AccountQuota accountQuota(String accountName)`
  Metoda pro získání informací o kapacitě prostoru v úložišti. Parametrem je název uživatelského účtu. Návratovou hodnotou jsou informace o kapacitě prostoru v úložišti.

- `void addDownloadSource(String accountName, FileInfo sourceFile)`
  Tato metoda přidává zdrojový soubor pro stahování z více úložišť zároveň. Parametry tvoří název uživatelského účtu, ze kterého má být stahováno, a metadata souboru, který má být stažen.

- `void addUpdateDestination(String accountName, FileInfo destination, FileInfo destinationFile, String destinationName)`
  Tato metoda přidává cílový soubor pro nahrávání na více úložišť

zároveň. Parametry tvoří název uživatelského účtu, na který má být nahráváno, metadata cílové složky, metadata cílového souboru a název cílového souboru. Při využití této metody je přepsán obsah cílového souboru.

- `void addUploadDestination(String accountName, FileInfo destination, String destinationName)`
  Tato metoda přidává cílový soubor pro nahrávání na více úložišť zároveň. Parametry tvoří název uživatelského účtu, na který má být nahráváno, metadata cílové složky a název cílového souboru. Při využití této metody je na úložiště nahrán nový soubor.

- `void authorizeAccount(String accountName, AuthorizationCallback callback)`
  Pomocí této metody je vyvolán proces autorizace uživatelského účtu. Parametry volání jsou název uživatelského účtu, který má být autorizován, a implementace rozhraní pro interakci s uživatelem během procesu.

- `FileInfo copy(String accountName, FileInfo file, FileInfo destination, String destinationName)`
  Metoda pro kopírování souborů v rámci úložiště. Parametrem je název uživatelského účtu, na kterém má ke kopírování dojít, metadata zdrojového souboru, metadata cílové složky a název cílového souboru. Návratem jsou metadata souboru vytvořeného kopírováním.

- `void createAccount(String accountName, String cloudStorage)`
  Metoda pro vytvoření uživatelského účtu. Předávanými parametry jsou název uživatelského účtu a název úložiště, ke kterému bude pomocí vytvořeného uživatelského účtu přistupováno.

- `FileInfo createFolder(String accountName, String folderName, FileInfo parent)`
  Metoda pro vytvoření nové složky v úložišti. Parametry tvoří název uživatelského účtu, název složky a metadata cílové složky, ve které má být nová složka vytvořena. Výstupem metody jsou metadata nově vytvořené složky.

- `FileInfo delete(String accountName, FileInfo file)`
  Metoda pro smazání souboru nebo složky v úložišti. Parametry tvoří název uživatelského účtu a metadata souboru či složky. Po vykonání metody jsou navrácena metadata smazaného souboru či složky.

- `void deleteAccount(String accountName)`
  Metoda pro smazání uživatelského účtu. Parametrem je pouze název uživatelského účtu.

- `File downloadFile(String accountName, FileInfo sourceFile, File destination, boolean overwrite)`
  Metoda pro stažení souboru z jednoho úložiště v jednom vlákně. Parametry volání jsou název uživatelského účtu, metadata stahovaného souboru, lokální soubor, který bude vytvořen, a zda je povolen přepis lokálního souboru v případě jeho existence. Návratem metody je lokální soubor, který byl stažen.

- `File downloadFile(String accountName, FileInfo sourceFile, String destination, boolean overwrite)`
  Stejná metoda jako předešlá, akorát s tím rozdílem, že namísto lokálního souboru je dodán pouze jeho název.

- `File downloadMultiFile(File destination, boolean overwrite)`
  Metoda pro stažení souboru z více úložišť ve více vláknech. Tato metoda využívá zdroje připravené metodou `addDownloadSource`. Vstupem metody je cílový lokální soubor a zda může dojít k přepisu existujícího souboru. Výstupem je pak lokální soubor.

- `File downloadMultiFile(String destination, boolean overwrite)`
  Tato metoda je stejná jako předešlá, akorát namísto lokálního souboru má na vstupu název lokálního souboru.

- `OperationError getLastError()`
  Metoda pro získání poslední chyby, která se vyskytla v knihovně.

- `ProgressListener getListener()`
  Metoda pro získání implementace rozhraní pro sledování průběhu stahování a nahrávání.

- `MultiCloudSettings getSettings()`
  Metoda pro získání aktuálního nastavení knihovny.

- `FileInfo listFolder(String accountName, FileInfo folder, boolean showDeleted, boolean showShared)`
  Metoda pro získání obsahu vzdálení složky. Parametry tvoří název uživatelského účtu, metadata složky, jejíž obsah má být získán, a zda je povoleno zobrazení smazaného a sdíleného obsahu. Výstupem z metody jsou metadata složky rozšířena o její obsah.

- `FileInfo listFolder(String accountName, FileInfo folder, boolean showDeleted)`
  Stejná metoda jako předešlá, pouze s menším počtem parametrů.

- `FileInfo listFolder(String accountName, FileInfo folder)`
  Stejná metoda jako předešlá, bez parametrů ovlivňujících výpis obsahu.

- `FileInfo metadata(String accountName, FileInfo file)`
  Metoda pro získání metadat o souboru nebo složce. Vstupními parametry jsou název uživatelského účtu a metadata souboru či složky. Metadata na vstupu musí obsahovat pouze atributy pro jednoznačné určení souboru či složky, zbytek je voláním metody doplněn. Výstupem jsou tak kompletní metadata.

- `FileInfo move(String accountName, FileInfo file, FileInfo destination, String destinationName)`
  Metoda pro přesun souboru v rámci úložiště. Parametry jsou název uživatelského účtu, metadata zdrojového souboru, metadata cílové složky a název cílový souboru. Výstupem jsou nová metadata přesunutého souboru.

- `void refreshAccount(String accountName, AuthorizationCallback callback)`
  Metoda pro získání nového přístupového tokenu využitím obnovovacího tokenu. Vstupem jsou název uživatelského účtu a rozhraní pro interakci s uživatelem během procesu.

- `FileInfo rename(String accountName, FileInfo file, String fileName)`
  Metoda pro přejmenování souboru či složky. Parametry volání jsou název uživatelského účtu, metadata souboru či složky a nový název. Návratovou hodnotou jsou nová metadata souboru či složky po přejmenování.

- `void renameAccount(String accountName, String newName)`
  Metoda pro změnu názvu uživatelského účtu. Vstupem je název uživatelského účtu a název, na který má být uživatelský účet přejmenován.

- `List<FileInfo> search(String accountName, String search, boolean showDeleted)`
  Metoda pro vyhledávání souborů a složek. Vstupem jsou název uživatelského účtu, řetězec, který má být vyhledán, a zda je povoleno

zobrazení smazaných souborů či složek. Návratem z metody je seznam s metadaty nalezených souborů a složek.

- `void setListener(ProgressListener listener)`
  Metoda pro nastavení implementace rozhraní pro naslouchání průběhu operace nahrávání a stahování.

- `FileInfo updateFile(String accountName, FileInfo destination, FileInfo destinationFile, String destinationName, File data)`
  Metoda pro přepsání obsahu souboru na jednom úložišti. Vstupem metody jsou název uživatelského účtu, metadata cílové složky, metadata cílového souboru, název cílového souboru a lokální zdrojový soubor. Výstupem pak jsou nová metadata upraveného souboru.

- `FileInfo updateMultiFile(File data)`
  Touto metodou je přepsán obsah všech vzdálených souborů připravených metodou `addUpdateDestination`. Vstupem je zde pouze lokální zdrojový soubor. Výstupem pak jsou metadata upraveného souboru.

- `FileInfo uploadFile(String accountName, FileInfo destination, String destinationName, boolean overwrite, File data)`
  Metoda pro nahrání souboru na jedno úložiště. Vstupem metody jsou název uživatelského účtu, metadata cílové složky, cílový název souboru v úložišti, zda může být přepsán existující soubor a lokální zdrojový soubor. Výstupem jsou metadata nové nahraného souboru.

- `FileInfo uploadMultiFile(boolean overwrite, File data)`
  Metoda pro nahrání souboru na více úložišť zároveň. Využívány jsou cílové destinace připravené metodou `addUploadDestination`. Vstupními parametry metody jsou nastavení přepisu existujících souborů a lokální zdrojový soubor. Výstupem pak jsou metadata nahraného souboru.

- `void validateAccounts()`
  Tato metoda slouží pro odstranění nevalidních spojení mezi uživatelskými účty a přístupovými tokeny a pro odstranění nevyužívaných tokenů.

A.1.2 Výjimky

Součástí knihovny jsou i zde popsané výjimky, které jsou standardně vyvolávány při výskytu problémů uvnitř knihovny.

- `AbortedException` – Tato výjimka rozšiřuje výjimku `MultiCloudException` a její využití je pro signalizaci přerušení operace.
- `MultiCloudException` – Hlavním účelem této výjimky je signalizace problémů, které vznikly nesprávným užíváním knihovny.
- `OAuth2SettingsException` – Tato výjimka je vyvolávána v případě výskytu problémů během autorizačního procesu.

A.1.3 Rozhraní

Knihovna obsahuje několik rozhraní, která jsou určena pro implementaci při jejím použití.

- `AuthorizationCallback` – Toto rozhraní je určeno pro vyvolání grafického uživatelského rozhraní během autorizačního procesu. Cílem tohoto rozhraní je tak poskytnout uživateli platformě závislou metodu pro autorizaci přístupu k úložišti.
- `AccountManager` – Toto rozhraní specifikuje metody pro správu uživatelských účtů. Součástí knihovny je i jeho základní implementace pracující se soubory.
- `CloudManager` – Toto rozhraní specifikuje metody pro správu definic cloudových úložišť. Součástí knihovny je implementace pracující se soubory.
- `CredentialStore` – Toto rozhraní slouží pro ukládání a získávání přístupových tokenů. Součástí knihovny je implementace tohoto rozhraní pracující se soubory a její rozšíření implementující šifrování dat.

A.1.4 Abstraktní třídy

Důležitou abstraktní třídou knihovny je třída `ProgressListener`, která tvoří základ pro implementaci sledování průběhu operací nahrávání a stahování souboru. Abstraktní metodou, kterou je třeba implementovat, je metoda `onProgress`. Tato metoda slouží pro periodické zobrazení průběhu operace. Minimální interval, po kterém je metoda vyvolána, je předán třídě pomocí konstruktoru. Metoda nemá žádné vstupní parametry a neobsahuje žádný výstup. Aby bylo možné stahovat a nahrávat soubory, musí být knihovně předána instance třídy tuto třídu rozšiřující.

A.1.5 Konfigurační soubor

Konfigurační soubory pro knihovnu jsou napsány ve formátu JSON a mají následující podobu. Hodnoty typu `{request_obj}` symbolizují objekty pro specifikaci HTTP požadavků, které jsou popsány dále.

Každý konfigurační soubor má povinný parametr `name`, který symbolizuje název úložiště. Popis úložiště je pak hodnotou parametru `description`.

Následují parametry `authorize` a `token`, které definují HTTP požadavky na autorizační server v průběhu autorizace. Parametr `grant_type` specifikuje typ použitého autorizační grantu a `grant_class` pak třídu pro *extension grant* v případě jeho použití. Parametry `client_id`, `client_secret`, `username` a `password` jsou předávanými parametry autorizačních grantů, přičemž vyplněny musí být pouze ty parametry, které autorizační grant používá. Přesměrování po procesu autorizace je provedeno na `redirect_uri`. Posledním autorizačním parametrem je `scope`, kterým je specifikován požadovaný rozsah přístupu k úložišti.

Následuje parametr pro specifikaci kořenové složky úložiště `root_folder`. Tento parametr obsahuje objekt, kterému je možno specifikovat identifikátor nebo cestu ke kořenové složce.

Zbylé parametry slouží pro specifikaci jednotlivých operací prováděných nad úložištěm. Hodnotou každého z těchto parametrů je objekt specifikující HTTP požadavek na úložiště.

```
{
  "name":"",
  "description":"",
  "authorize":{request_obj},
  "token":{request_obj},
  "grant_type":"",
  "grant_class":"",
  "client_id":"",
  "client_secret":"",
  "username":"",
  "password":"",
  "redirect_uri":"",
  "scope":"",
  "root_folder":
  {
    "id":""
    "path":""
  },
  "account_info_request":{request_obj},
  "account_quota_request":{request_obj},
  "download_file_request":{request_obj},
  "upload_file_begin_request":{request_obj},
  "upload_file_request":{request_obj},
  "upload_file_finish_request":{request_obj},
  "update_file_begin_request":{request_obj},
  "update_file_request":{request_obj},
  "update_file_finish_request":{request_obj},
  "create_dir_request":{request_obj},
  "list_dir_begin_request":{request_obj},
  "list_dir_request":{request_obj},
  "rename_request":{request_obj},
  "copy_request":{request_obj},
  "move_request":{request_obj},
  "delete_request":{request_obj},
  "search_request":{request_obj},
  "metadata_request":{request_obj}
}
```

Objekt pro specifikaci HTTP požadavku je zachycen v následující ukázce. Povinnou součástí tohoto objektu jsou parametry `uri`, který určuje identifikátor vzdáleného zdroje, a `method`, který specifikuje použitou HTTP metodu. Mezi další parametry patří parametr `params`, který obsahuje množinu parametrů použitých při volání požadavku. Dále je možno specifikovat mapování návratových parametrů parametrem `mapping`. Mapování se vztahuje i na parametry předávané uvnitř JSON odpovědi. Parametr `body` slouží pro specifikaci řetězce, který má být v těle požadavku. Parametr `json_body` je pak určen pro specifikaci obsahu těla požadavku ve formátu JSON.

Pokud je požadavku potřeba předávat nějakou proměnnou hodnotu, ať už do parametru `uri`, nebo kamkoliv jinam, je možno ji vložit přímo do řetězce. Vložená hodnota musí být uzavřena mezi znaky `<` a `>`. Knihovnou jsou pak tyto hodnoty rozpoznávány a nahrazovány za příslušné proměnné. Pro každou operaci jsou dostupné jiné takto mapované proměnné.

```
{
  "uri":"",
  "method":"",
  "params":
  {
    "parameter":"value"
  },
  "mapping":
  {
    "property":"mapping"
  },
  "body":null,
  "json_body":
  {
    "property":"value"
  }
}
```

Následuje seznam operací a jejich mapovaných proměnných. Pro každou proměnnou je popsán i její význam.

- `account_info` – žádné proměnné
- `account_quota` – žádné proměnné

- `download_file`
  + `download_url` – URL pro stažení souboru
  + `id` – identifikátor stahovaného souboru
  + `path` – cesta ke stahovanému souboru

- `upload_file` a `update_file`
  + `overwrite` – zda má být soubor přepsán
  + `size` – velikost nahrávaného souboru
  + `id` a `destination_id` – identifikátor cílové složky
  + `file_id` – identifikátor souboru pro přepsání obsahu
  + `path` a `destination_path` – cesta k cílovému souboru
  + `name` – název cílového souboru
  + `offset` – posun uvnitř cílového souboru
  + `session` – identifikátor relace
  + `offsetbuffer` – posun uvnitř cílového souboru včetně aktuálně nahrávaných dat

- `create_dir`
  + `name` – název složky
  + `id` – identifikátor rodičovské složky
  + `path` – cesta k nové složce

- `list_dir`
  + `id` – identifikátor složky
  + `path` – cesta ke složce
  + `deleted` – zda mají být zobrazen smazaný obsah

- `rename`
  + `id` a `source_id` – identifikátor zdroje
  + `path` a `source_path` – cesta ke zdroji
  + `destination_path` – cesta k novému souboru či složce
  + `name` – nový název

- `copy` a `move`
  + `id` a `source_id` – identifikátor zdroje
  + `path` a `source_path` – cesta ke zdroji
  + `destination_id` – identifikátor cílové složky
  + `destination_path` – cesta k cílové složce
  + `name` – nový název

- `delete` a `metadata`
  + `id` – identifikátor zdroje
  + `path` – cesta ke zdroji

- `search`
  + `query` – vyhledávaný řetězec
  + `deleted` – zda mají být zobrazen smazaný obsah

A.1.6 Nasazení knihovny

Aby bylo možné využívat knihovnu, je potřeba kromě konfiguračních souborů ještě zaregistrovat vytvářenou aplikaci v rámci cloudových úložišť. Registrací je získán identifikátor klienta a tajný klíč, které jsou potřeba pro autorizaci aplikace. Pro pohodlnější autorizační proces je možné nastavit v rámci nastavení aplikace na cloudových úložištích URI pro přesměrování po skončení autorizačního procesu. Některá úložiště k tomu vyžadují, aby k aplikaci byla přiřazena unikátní doména. Cílem přesměrování by tak měl být skript, který by provedl následné přesměrování zpět na počítač uživatele.

K tomuto účelu je připraven jednoduchý PHP skript. Pro správné fungování tohoto skriptu je nutné jeho nasazení na webový server s podporou PHP. Jako webový server je možné využít například HTTP server od Apache dostupný z `https://httpd.apache.org/`. Distribuce PHP je pak dostupná z `http://www.php.net/`.

Pro instalaci webového serveru od Apache je doporučeno postupovat podle uživatelské příručky dostupné na `https://httpd.apache.org/docs/2.4/getting-started.html`. Obdobně je tomu i pro PHP, kde uživatelská příručka je umístěna na `http://php.net/manual/en/install.php`.

Po úspěšné instalaci webového serveru s podporou PHP je možné umístit skript na server. Lokace skriptu nehraje roli, podmínkou však je, že tento skript musí být přístupný pro uživatele, který prochází autorizačním procesem. Pokud je tedy výsledná klientská aplikace využívající knihovnu cílena na širokou veřejnost, je nutné mít připravenou veřejnou doménu v síti Internet.

Příloha B

B.1 Uživatelská příručka - MultiCloudDesktop

B.1.1 Instalace a spuštění

Aktuální verzi klientské aplikace je možné stáhnout na následující webové adrese. Doporučené je stažení již připravené aplikace v balíku `.jar` ze složky `/jar`. Druhou možností je stažení `.exe` souboru ze složky `/exe`. Třetí možností je stažení zdrojových souborů aplikace a kompilace pomocí ANT skriptu `build.xml`.

`https://github.com/stanek0j/MultiCloudDesktop`

Spolu s aplikací je důležité stažení definic cloudových úložišť. To se provede stažením celé složky `/definitions` a jejím umístění do stejné složky, ve které je stažena aplikace. Aplikace po spuštění prohledává složku `/definitions` a načítá z ní definice cloudových úložišť.

Pro spuštění programu je potřeba mít instalované běhové prostředí Javy, tzv. Java Runtime Environment (JRE) ve verzi 1.7. Aktuální verzi JRE je možno získat z následující adresy.

`http://www.oracle.com/technetwork/java/javase/downloads/`

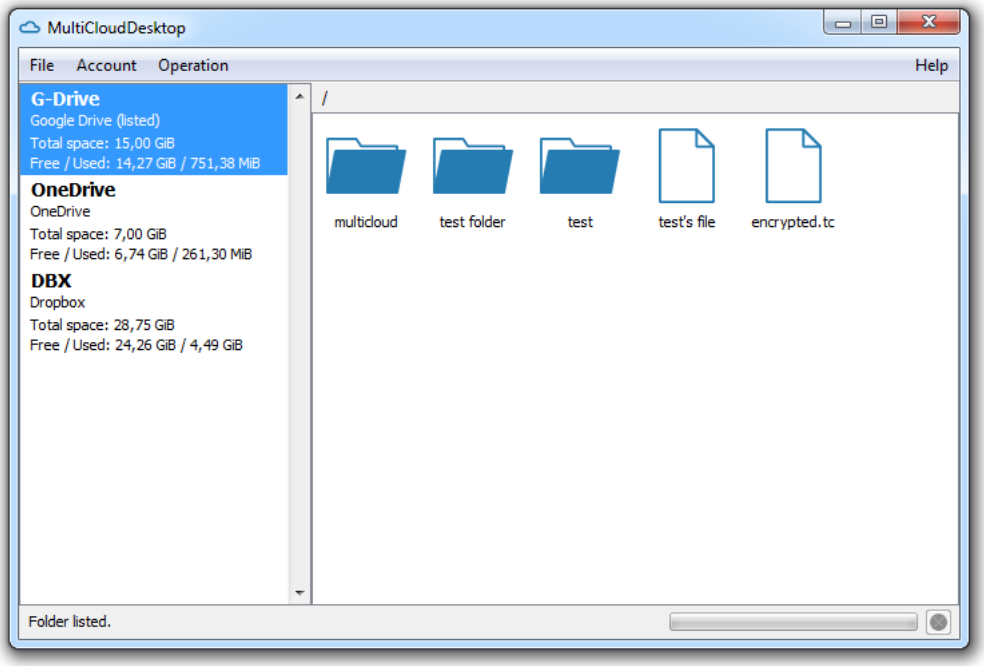
Spuštění aplikace se provede v závislosti na stažené variantě. Spustitelný soubor formátu `.exe` je možno spustit dvojklikem myši. Pro `.jar` soubor je možno použít příkaz z příkazové řádky:

```
java -jar MultiCloudDesktop.jar
```

Po spuštění si aplikace vytváří soubory `.multicloud`, `credentials.dat`, `accounts.json` a `preferences.json`. Tyto soubory jsou potřebné pro správné fungování aplikace a není vhodné je mazat.

B.1.2 Ovládání

Po spuštění aplikace se zobrazí výchozí okno aplikace. V horní části okna je lišta s nabídkou ovládání. V levé části okna je seznam uživatelských účtů a v pravé části pak seznam souborů a složek. Po prvním spuštění aplikace je levý i pravý panel prázdný. Okno aplikace je zachyceno na obrázku B.1.



Obrázek B.1: Okno aplikace.

Jako první je zde zařazen popis a funkce obou panelů v okně a dolní lišty. Poté následuje popis jednotlivých operací z horní nabídky, včetně ukázek dialogových oken, která jsou zobrazována.

Levý panel

V tomto panelu se zobrazují vytvořené uživatelské účty. Ke každému účtu je zobrazen jeho název, typ úložiště, ke kterému přistupuje, a informace o kapacitách úložiště. Dvojklikem levým tlačítkem na účet je provedeno zobrazení jeho kořenové složky v pravém panelu. Pokud však účet nebyl ještě autorizován, je spuštěn proces autorizace, který vyvolá okno webového prohlížeče. Kliknutí pravým tlačítkem vyvolá kontextovou nabídku, která je shodná s horní nabídkou *Account*.

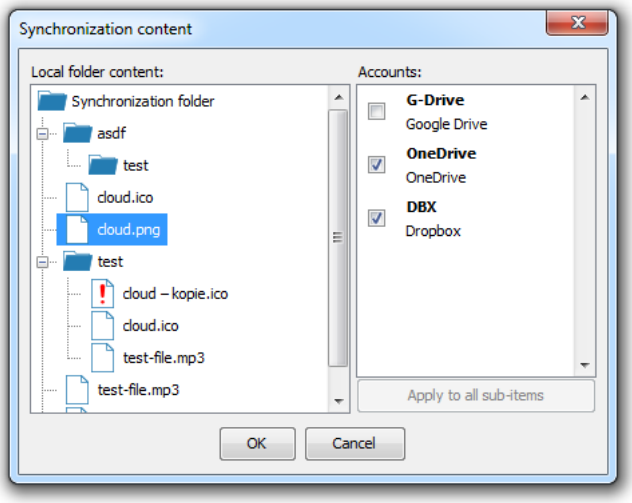
Pro levý panel jsou definovány klávesové zkratky. Stiskem klávesy `Enter` nebo mezerníku je provedena stejná akce jako při dvojkliku levým tlačítkem myši. Klávesou `Delete` je možno mazat účty.

Pravý panel
V pravém panelu je zobrazován obsah složek cloudových úložišť. V horní části panelu se zobrazuje cesta k aktuálně prohlížené složce. Dvojklikem levým tlačítkem na složku je možno složku procházet. Procházením složky s názvem `..` se provede přesun o složku výše (do předchozí složky). Kliknutím pravým tlačítkem je vyvolána kontextová nabídka, která je shodná s horní nabídkou *Operation*. Stiskem klávesy `Enter` nebo mezerníku je rovněž možno procházet složky.

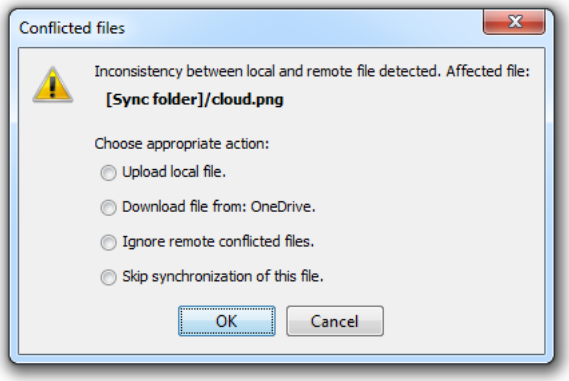
Dolní lišta
V levé části dolní lišty jsou zobrazována informační a chybová hlášení aplikace. V pravé části je zobrazován indikátor postupu operací a tlačítku na jejich přerušení. Pokud zrovna nějaká operace probíhá a není vyvoláno zvláštní dialogové okno, je též možno tuto operaci přerušit stiskem klávesy `Esc`. Následuje popis jednotlivých úkonů, které je možno v aplikaci provádět. Úkony jsou řazeny podle nabídky v horní části okna.

Synchronizace – *File -> Synchronize* – `Ctrl+S`
Tato nabídka spouští proces synchronizace. Před spuštěním procesu je nutné vybrat v nastavení synchronizační složku a určit, které soubory se budou kam synchronizovat. Výběr souborů pro synchronizaci se provádí prostřednictvím dialogového okna, které je pro názornost zachyceno v obrázku B.2. Celý proces synchronizace může být časově velmi náročný. Během synchronizace se může stát, že některý ze souborů nebude v konzistentním stavu a bude vyžadován zásah uživatele prostřednictvím dialogového okna jaké je zobrazeno v obrázku B.3.

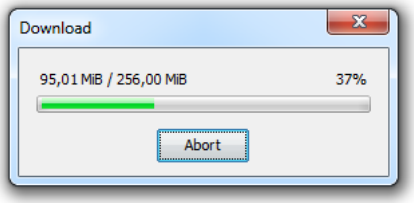
Nastavení – *File -> Preferences* – `Ctrl+P`
Nastavení aplikace je prováděno prostřednictvím dialogového okna zobrazeného v obrázku B.5. V tomto okně je možné nastavit typ zobrazení souborů a složek, synchronizační složku a vybrat soubory pro synchronizaci. Dále je zde umožněno nastavit počet vláken pro stahování a možnosti výpisu obsahu složek. Nechybí zde ani možnost vypnout vyskakování chybových hlášení při chybě.



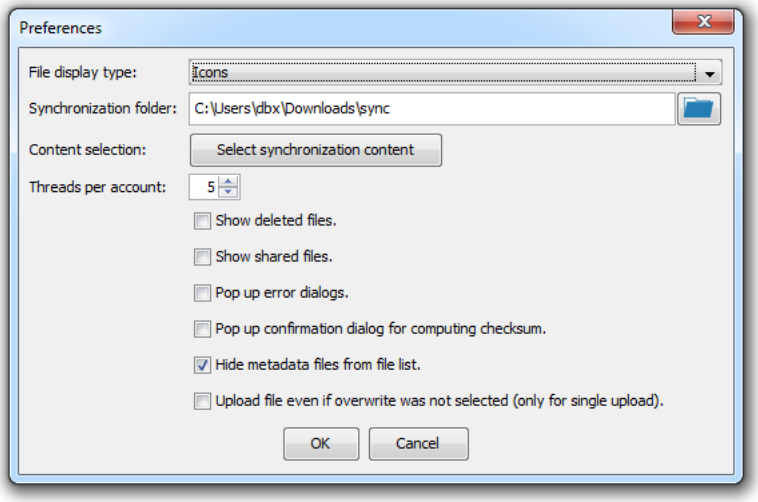
Obrázek B.2: Dialogové okno výběru souborů pro synchronizaci.



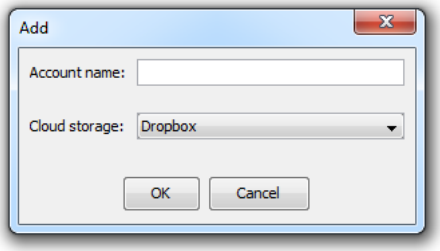
Obrázek B.3: Dialogové okno chyby synchronizace.



Obrázek B.4: Dialogové okno zobrazující průběh operace.



Obrázek B.5: Dialogové okno pro nastavení aplikace.



Obrázek B.6: Dialogové okno pro vytvoření účtu.

Obrázek B.7: Dialogové okno pro vyhledávání.

Ukončení aplikace – *File -> Exit* – `Alt+F4`
Aplikaci je možné ukončit buď z horní nabídky, klávesovou zkratkou, nebo kliknutím na křížek v pravém horním rohu okna.

Vytvoření účtu – *Account -> Add*
Pro vytvoření uživatelského účtu je zobrazeno dialogové okno, do kterého je nutné zadat název účtu a zvolit typ úložiště (viz obr. B.6). Po úspěšném vytvoření účtu je účet zobrazen v levém panelu.

Autorizace – *Account -> Authorize*
Spuštěním procesu autorizace je otevřeno okno webového prohlížeče. Pro úspěšnou autorizaci aplikace pro přístup ke cloudovému úložišti je potřeba postupovat dle pokynů v okně prohlížeče. Po úspěšné autorizaci je skrz aplikaci možno přistupovat k souborům v úložišti.

Informace o účtu – *Account -> Information*
Tato akce vyvolá dialogové okno, ve kterém je zobrazeno jméno uživatele cloudového úložiště a jeho identifikátor.

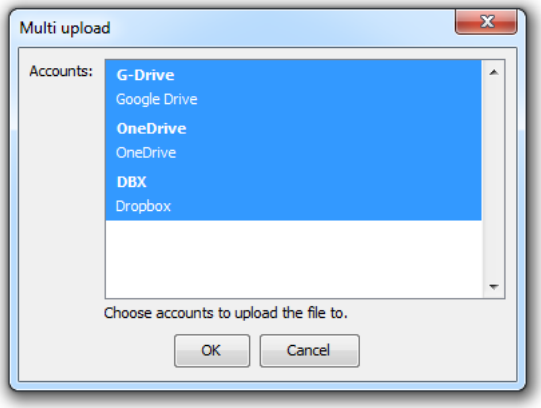
Informace o kapacitě úložiště – *Account -> Quota*
Tato akce vyvolá dialogové okno, ve kterém je zobrazena celková velikost úložiště, obsazený a volný prostor.

Přejmenování účtu – *Account -> Rename*
Tato akce slouží pro přejmenování uživatelského účtu. Pro tento účel je zobrazeno stejné dialogové okno jako při tvorbě účtu, ovšem není možno změnit typ úložiště (viz obr. B.6).

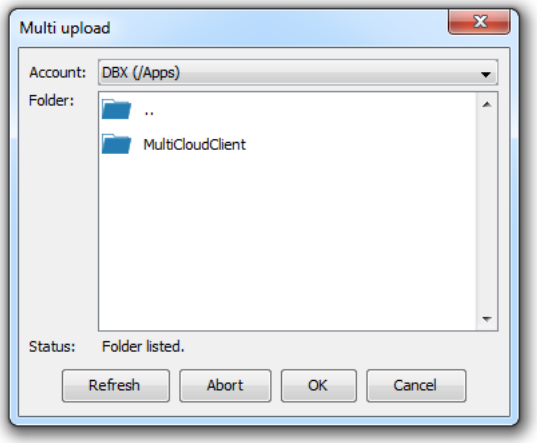
Odstranění účtu – *Account -> Remove*
Tato akce provede smazání vybraného uživatelského účtu. Před smazáním účtu je ještě nutné smazání potvrdit v dialogovém okně.

Obnovení – *Operation -> Refresh* – `F5`
Tato akce obnoví obsah právě zobrazení složky.

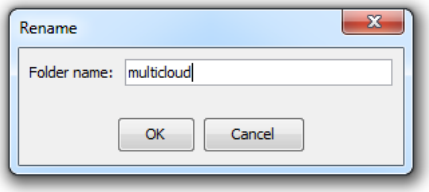
Vyhledávání – *Operation -> Find* – `Ctrl+F`
Tato akce vyvolává dialogové okno vyhledávání (viz obr. B.7). V tomto okně je možné zvolit účet, ve kterém bude vyhledáváno. Pod volbou účtu je nutné zadat hledaný řetězec. Pod tímto zadávacím prvkem je panel pro zobrazení výsledků. Tlačítkem *Find* se provede vyhledání a tlačítkem *Abort* je možné vyhledávání přerušit. Tlačítka *OK* a *Cancel* zavírají dialogové okno.



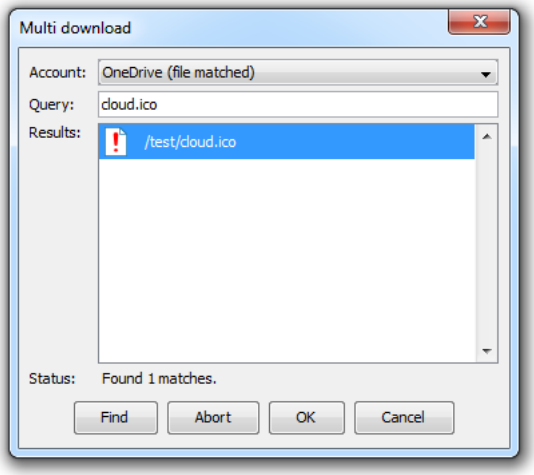
Obrázek B.8: Dialogové okno pro výběr cloudových úložišť.



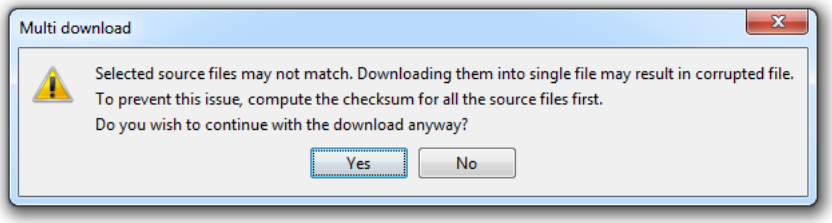
Obrázek B.9: Dialogové okno pro výběr složky v úložišti.



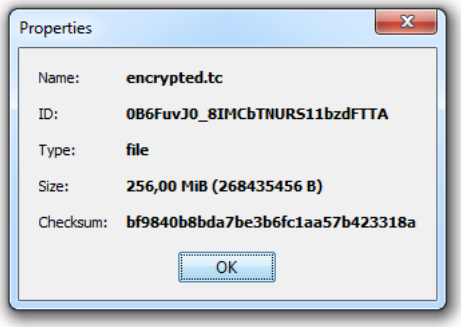
Obrázek B.10: Dialogové okno pro přejmenování souboru či složky.



Obrázek B.11: Dialogové okno vyhledávání v případě neshody souborů.



Obrázek B.12: Upozornění na možnost poškození souboru při stahování.



Obrázek B.13: Dialogové okno zobrazující informace o souboru.

Nahrávání souboru – *Operation -> Upload / Multi upload*
Soubory je možné nahrávat dvěma způsoby. Prvním způsobem je nahrávání souboru na pouze jedno úložiště. Při tomto způsobu nahrávání je zobrazeno dialogové okno pro výběr souboru v počítači. Po jeho vybrání je možné sledovat průběh nahrávání v oknu podobném tomu na obrázku B.4.

Druhým způsobem je nahrávání souboru na více úložišť zároveň. Pro tuto volbu je opět zobrazeno nejprve okno pro výběr lokálního souboru. Po vybrání souboru následuje dialogové okno pro výběr cílových cloudových úložišť (viz obr. B.8). Tento výběr je následován dalším dialogovým oknem, které je zachyceno na obrázku B.9 a slouží pro výběr cílových složek pro soubor na úložištích. Pak již následuje jen okno pro sledování postupu operace z obrázku B.4.

Stahování souboru – *Operation -> Download / Multi download*
Pro stahování souborů je opět na výběr ze dvou voleb. První volbou je stažení vybraného souboru v pravém panelu z jednoho úložiště. Pro tuto volbu je zobrazeno dialogové okno pro výběr lokace v počítači, kam má být soubor uložen. Po volbě lokace následuje zobrazení průběhu stahování (viz obr.B.4).

Při druhé volbě je soubor stahován z více úložišť zároveň. Prvním dialogovým oknem je výběr cloudových úložišť, ze kterých se bude stahovat (viz obr. B.8). Po tomto výběru následuje vyhledání souboru na vybraných úložištích. K tomu slouží dialogové okno pro vyhledávání (viz obr. B.7). Pokud je však nalezen soubor, který by mohl odpovídat vybranému souboru pro stahování, ale není jasné, zda se jedná o stejný soubor, je tento soubor v okně vyhledávání označen vykřičníkem. Tento stav je zobrazen na obrázku B.11. Pokud je nějaký takový soubor vybrán pro stažení, je v dalším kroku zobrazeno upozornění, že by výsledný stažený soubor mohl být poškozen (viz obr. B.12). Předposledním dialogovým oknem je výběr lokální složky, do které má být soubor uložen. Pak již následuje jen okno zobrazující průběh stahování (viz obr. B.4).

Vytvoření složky – *Operation -> Create folder*
Pro vytvoření složky je zobrazeno dialogové okno, do kterého je zadán název složky. Toto okno je shodné s oknem pro přejmenování, které je zobrazeno na obrázku B.10.

Přejmenování – *Operation -> Rename* – `F2`
Tato akce vyvolá dialogové okno pro přejmenování souboru či složky (viz obr. B.10).

Smazání – *Operation -> Delete* – `Delete`
Pro smazání souboru nebo složky je nejprve nutné soubor nebo složku vybrat v pravém panelu. Smazání je nutné potvrdit v dialogovém okně, které je při této akci zobrazeno.

Kopírování a přesun – *Operation -> Cut / Copy / Paste* – `Ctrl+X/C/V`
Tyto akce jsou povoleny pouze pro soubory. Celý proces funguje tak, že je nejprve zvolen nějaký soubor v úložišti. Ten je následně označen pro přesun nebo kopírování. Poté je nutné se dostat v pravém panelu do cílové složky. Při vkládání souboru je pak ještě zobrazeno dialogové okno, ve kterém je možno soubor přejmenovat (viz obr. B.10). Pokud se jedná o transfer souboru mezi dvěma úložišti, je po vložení zobrazeno dialogové okno sledující průběh přesunu nebo kopírování (viz obr. B.4).

Výpočet kontrolního součtu – *Operation -> Compute checksum*
Kontrolní součet je možné počítat pouze pro soubory. Výpočet probíhá tak, že je zobrazeno dialogové okno podobné tomu na obrázku B.4, které sleduje průběh stahování vybraného souboru. Po stažení souboru je vypočten kontrolní součet, který je následně přiřazen k souboru.

Vlastnosti – *Operation -> Properties*
Tato akce vyvolá dialogové okno s informacemi o vybraném souboru nebo složce. Podoba tohoto okna je vidět na obrázku B.13.

O programu – *Help -> About*
Poslední položkou horní nabídky je akce pro zobrazení informací o programu.

B.2 Uživatelská příručka - `MultiCloudAndroid`

B.2.1 Instalace a spuštění

Aktuální verzi mobilní aplikace je možné stáhnout na následující webové adrese. Ve složce `/bin` je umístěn soubor s aplikací a příponou `.apk`. Jelikož není aplikace umístěna na portálu *Google Play*, je nutné pro její instalaci mít povolenou instalaci aplikací z jiných zdrojů. Toto nastavení se provede zatrhnutím nabídky *Nastavení -> Zabezpečení -> Neznámé zdroje*. Tento postup se může lišit pro různé verze systému Android.

`https://github.com/stanek0j/MultiCloudAndroid`

Podmínkou pro úspěšnou instalaci aplikace je systém Android ve verzi alespoň `3.0` nebo novější a přibližně 4 MB volného místa. Instalaci je možno spustit překopírováním balíku do zařízení a jeho spuštěním pomocí libovolného správce souborů. Druhou možností je využití aplikace *adb*, který je součástí Android SDK. Instalace se pak spustí příkazem:

```
adb install MultiCloudAndroid.apk
```

Spuštění aplikace se provede klepnutím na příslušnou položka v menu zařízení. Položka aplikace je pojmenována *MultiCloud* a jako ikonu má modrý symbol oblaku.

B.2.2 Ovládání

Po spuštění aplikace je zobrazena výchozí obrazovka s uživatelskými účty. Podoba této obrazovky je zachycena na obrázku B.14(a). V dolní nabídce jsou zobrazeny ikony pro operace této nabídky. Zleva je to přidání účtu, obnovení informací o účtech, synchronizace a nastavení. V kontextové nabídce pro jednotlivé položky seznamu pak jsou operace autorizace, získání informací o účtu, přejmenování a smazání. Kontextová nabídka je vyvolána dlouhým stiskem položky v seznamu uživatelských účtů.

Přidání účtu
Po stisku ikony se zobrazí dialogové okno z obrázku B.16(a). Povinnými údaji pro vytvoření účtu jsou uživatelské jméno a výběr typu úložiště.

Obnovení informací o účtech
Tímto tlačítkem se spustí operace, která načítá informace ke všem uživatelským účtům. Po dokončení operace jsou obnoveny informace v seznamu uživatelských účtů.

Synchronizace
Pro spuštění synchronizace je nejprve nutné mít nastavenu synchronizační složku. Po spuštění této operace se zobrazí dialog z obrázku B.16(g). Po dokončení této operace je zobrazen dialog podobný tomu v obrázku B.16(h) pouze v případě, že nastal konflikt mezi synchronizovanými soubory. Vyřešení tohoto konfliktu je potřeba provést ručně stažením souboru ze vzdáleného úložiště nebo nahráním lokálního souboru na úložiště.

Nastavení
Stiskem této ikony je vyvoláno okno pro nastavení. Toto okno je zachyceno v obrázku B.14(c) a je popsáno dále v této sekci.

Autorizace
Spuštěním této operace se otevře okno výchozího webového prohlížeče a uživateli je nabídnuta webová stránka úložiště pro autorizaci aplikace. Po průchodu procesem autorizace na webové stránce je možné prohlížeč zavřít. Po jeho zavření je opět otevřena aplikace a dokončen celý proces autorizace.

Informace o účtu
Pomocí této operace je možné zjistit informace o uživateli a dostupné prostoru v rámci úložiště. Po dokončení této operace je zobrazeno dialogové okno zachycené na obrázku B.16(b).

Přejmenování účtu
Přejmenování účtu vyvolá stejný dialog jako vytvoření účtu, s tím rozdílem, že název účtu je již předvyplněn a typ úložiště není možné změnit.

Smazání účtu
Smazání uživatelského účtu probíhá bez zobrazení dialogového okna. Výsledkem je odebrání uživatelského účtu ze seznamu.

Obrazovka adresářové struktury
Při kliknutí na již autorizovaný uživatelský účet je zobrazena obrazovka se soubory v úložišti. Podoba této obrazovky je vidět na obrázku 8.2(b). Opět jsou zde dostupná tlačítka ve spodní části obrazovky. Zleva to jsou vytvoření nové složky, nahrání souboru, obnovení obsahu složky synchronizace a skryté tlačítko nastavení. Tlačítka synchronizace a nastavení fungují stejně jako u

předchozí obrazovky a není nutné je znovu popisovat. Kontextová nabídka obsahuje možnosti stažení souboru, přejmenování, smazání, výpočtu kontrolního součtu a zobrazení vlastností souboru. Procházení složek se provede stisknutím vybrané složky, kterým se spustí načítání jejího obsahu. Pro přechod zpět do rodičovské složky je možno stisknout ikonu aplikace s názvem aktuální složky v levém horním rohu. Tlačítkem zpět je pak možný přechod k obrazovce s uživatelskými účty.

Vytvoření nové složky
Toto tlačítko vyvolá dialog pro vložení jména nové složky. Podoba tohoto dialogu je zobrazena na obrázku B.16(e). Po vytvoření nové složky je obnoven zobrazený obsah.

Nahrání souboru
Pro nahrání souboru je zobrazeno dialogové okno s výběrem lokálního souboru pro nahrání (viz obr. B.16(i). V případě, že v úložišti již existuje soubor se stejným názvem, zobrazí se dialog vyžadující potvrzení připisu souboru. Po potvrzení je zobrazen dialog s průběhem nahrávání souboru, který je podobný tomu v obrázku B.16(d).

Obnovení obsahu složky
Tato operace načte znovu obsah aktuální složky.

Stažení souboru
Spuštěním této operace se zobrazí dialog podobný tomu v obrázku B.16(i). Pomocí tohoto dialogu je možno zvolit lokální složku, do které bude soubor stažen. Dalším dialogem pak je možnost přejmenování souboru před stažením. Pokud je vybrán již existující název souboru v dané složce, je nabídnuta možnost přepsání existujícího souboru. Posledním dialogem této operace je ten na obrázku B.16(d), který zobrazuje průběh stahování. Po dokončení je možné stažený soubor otevřít prostřednictvím libovolného správce souborů.

Přejmenování
Pro přejmenování slouží jednoduchý dialog s polem pro vkládání názvu souboru. Přejmenovat je možné soubory i složky.

Smazání
Před smazáním souboru nebo složky je zobrazen dialog vyžadující potvrzení operace uživatelem. Smazání je provedeno pouze po potvrzení dialogového okna vyobrazeného na obrázku B.16(c).

Výpočet kontrolního součtu
Výpočet kontrolního součtu spustí dialog zobrazující průběh stahování souboru. Jedná se opět o dialog podobný tomu na obrázku B.16(d). Po stažení je vypočten kontrolní součet a stažený soubor je opět smazán.

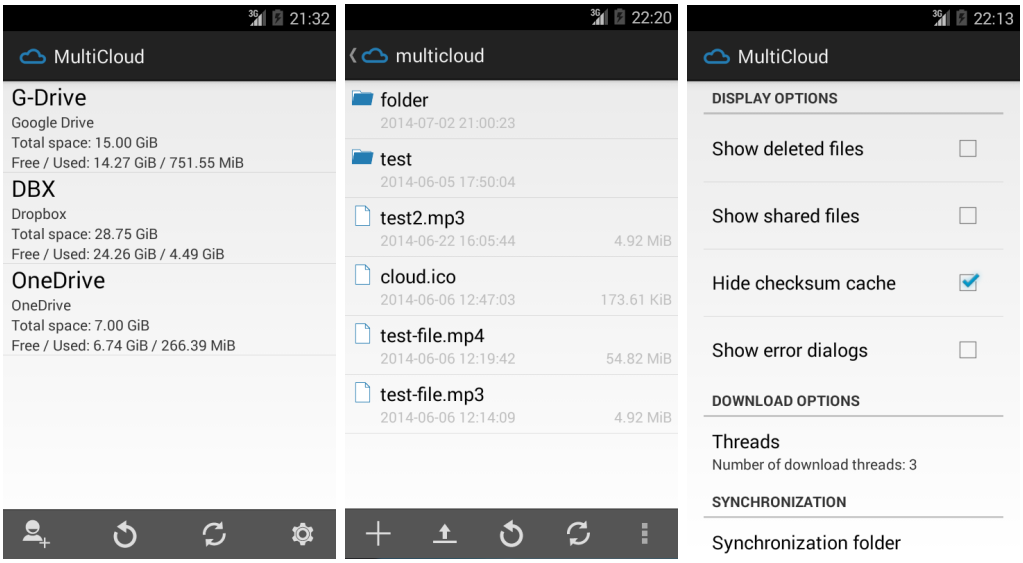
Vlastnosti
Touto operací je zobrazeno dialogové okno s informacemi o vybraném souboru nebo složce. Podoba dialogového okna je zachycena na obrázku B.16(f).

Obrazovka nastavení
Poslední obrazovkou aplikace je obrazovka nastavení. V této obrazovce jsou tři kategorie nastavení. První kategorií je nastavení zobrazení, do kterého patří volby zobrazení smazaných a sdílených souborů, volba skrytí souboru s kontrolními součty a volba zobrazování dialogových oken s chybami. Další kategorii tvoří volby pro stahování, kde je jedinou položkou volba počtu vláken pro stahování. Poslední kategorie se týká synchronizace a je zde možné zvolit synchronizační složku a nastavit selektivní synchronizaci pro jednotlivé soubory v synchronizační složce.

Počet vláken stahování
Touto volbou se zobrazí dialogové okno z obrázku B.15(c), ve kterém je možné zvolit mezi jedním až pěti vlákny. Doporučeným nastavením jsou v tomto případě tři vlákna.

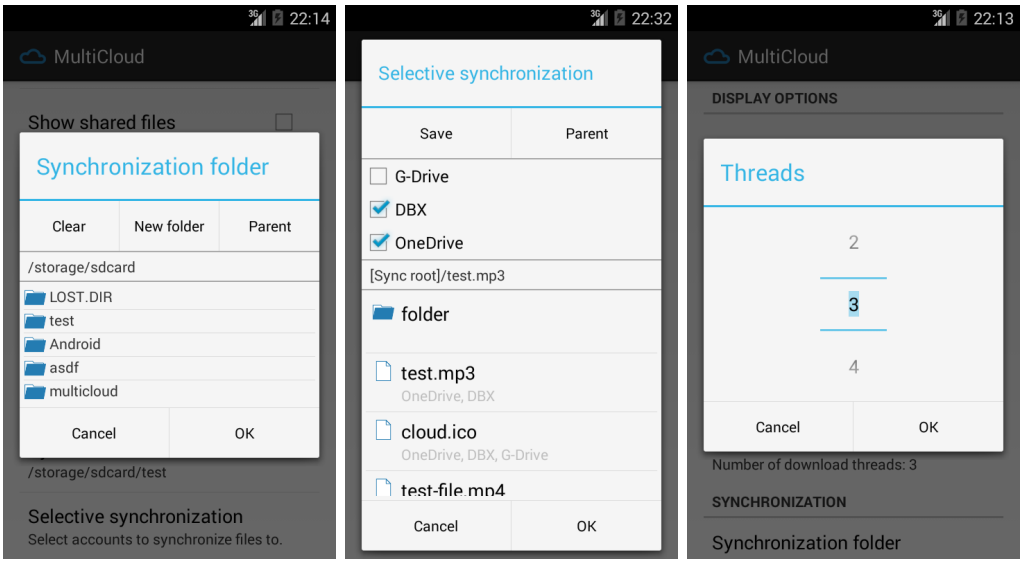
Synchronizační složka
V zobrazeném dialogovém okně zachyceném na obrázku B.15(a) je možné zvolit si složku, která bude synchronizována. V tomto dialogu nejsou zobrazovány soubory obsažené ve složkách. Tlačítkem `Clear` je možné vymazat nastavení složky a zakázat tím možnost synchronizace.

Selektivní synchronizace
Tato volba je dostupná pouze v případě zvolené synchronizační složky. Po stisku této volby se zobrazí dialogové okno zobrazení v obrázku B.15(b). Ve spodní části okna je možné procházet adresářovou strukturu, zatímco v horní části je možné pro jednotlivé soubory zvolit úložiště, na která budou synchronizovány. U každého souboru ve spodní části dialogu je pak zobrazen výčet úložišť, na která bude soubor synchronizován.



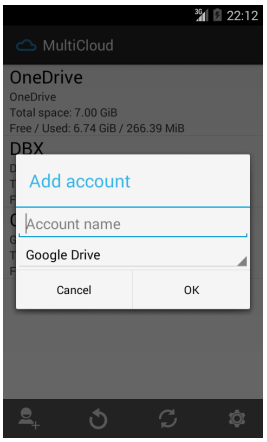
(a) Uživatelské účty. (b) Prohlížení souborů. (c) Nastavení.

Obrázek B.14: Obrazovky mobilní aplikace.

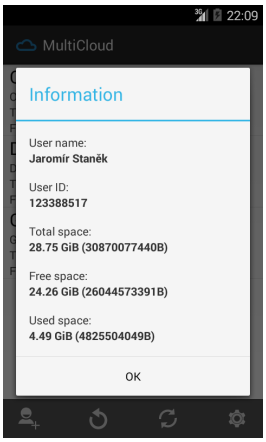


(a) Volba synchronizační složky. (b) Výběr selektivní synchronizace. (c) Volba počtu vláken.

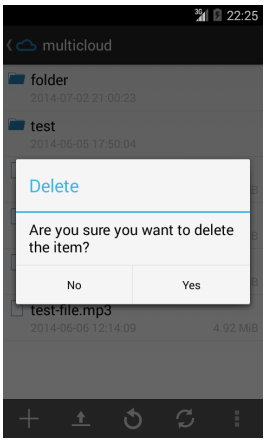
Obrázek B.15: Dialogy nastavení aplikace.



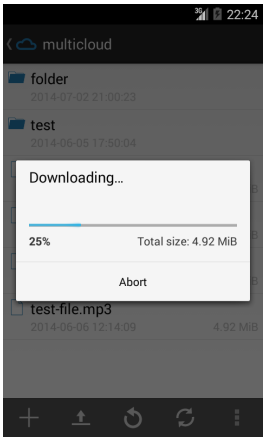
(a) Vytvoření účtu.



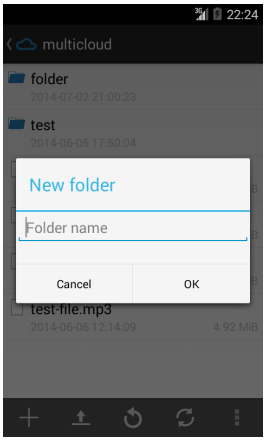
(b) Informace o účtu.



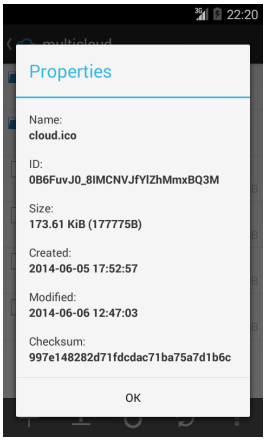
(c) Smazání položky.



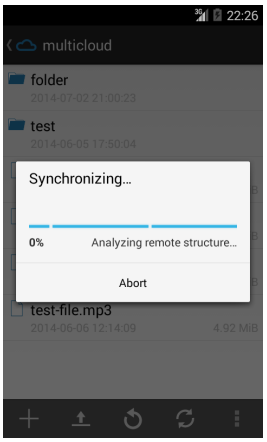
(d) Průběh stahování.



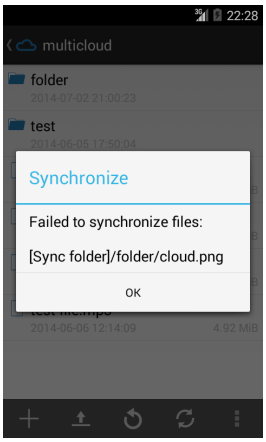
(e) Vytvoření složky.



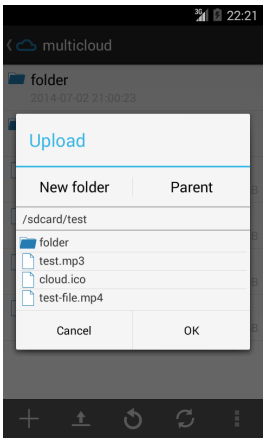
(f) Vlastnosti souboru.



(g) Průběh synchronizace.



(h) Výsledek synchronizace.



(i) Nahrání souboru.

Obrázek B.16: Dialogy mobilní aplikace.

Příloha C

C.1 Obsah CD

Přiložené médium obsahuje tento dokument a realizované programy. Adresářová struktura je následující:

- `dokument` – složka s tímto dokumentem ve formátu `.pdf`
- `program` – složka obsahující realizované programy
  - `MultiCloud` – složka s kompletními soubory implementované knihovny, včetně zdrojových souborů, binárních výstupů a JavaDoc dokumentace.
  - `MultiCloudAndroid` – složka s kompletními soubory implementované mobilní aplikace, včetně zdrojových souborů, binárních výstupů a JavaDoc dokumentace.
  - `MultiCloudDesktop` – složka s kompletními soubory implementované desktopové aplikace, včetně zdrojových souborů, binárních výstupů a JavaDoc dokumentace.
  - `PHP` – složka obsahující PHP skript pro automatickou autorizaci s využitím webové služby.