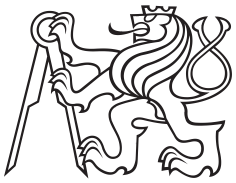
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická

Distribuované úložiště dat - uživatelské rozhraní

Artem Grigorian

Vedoucí: Ing. Macejko Peter
Květen 2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Grigorian** Jméno: **Artem** Osobní číslo: **452981**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Distribuované úložiště dat - uživatelské rozhraní

Název bakalářské práce anglicky:

Distributed data storage - desktop UI

Pokyny pro vypracování:

Navrhněte a implementujte uživatelské rozhraní (webovou nebo desktop aplikaci) pro práci s daty v distribuovaném úložišti (viz literatura).
Aplikace musí poskytovat alespoň tyto funkce:
1) Práci se soubory a adresáři (kopírování, přesouvání, přejmenování, nahrání, stáhnutí, smazání).
2) Práci s metadaty datových objektů (počet kopií, nastavení sdílení).
3) Zabezpečení dešifrovacího klíče (nesmí opustit aplikaci).
K implementaci komunikace se systémem sdílení dat můžete využít vznikající Java knihovnu. V případě potřeby upravte funkci zbývajících částí systému sdílení dat (viz literatura).
Své řešení otestujte na pilotní implementaci datového úložiště.

Seznam doporučené literatury:

[1] Dyntar, T.: Distribuované úložiště dat - správa metadat. [Diplomová práce]. Praha: ČVUT FEL, Katedra počítačů, 2014. 71 s.
[2] Janura, J.: Distribuované úložiště dat - klientská část. [Diplomová práce]. Praha: ČVUT FEL, Katedra počítačů, 2014. 66 s.
[3] Kudrnáč, M.: Distribuované úložiště dat - správa dat. [Diplomová práce]. Praha: ČVUT FEL, Katedra počítačů, 2014. 63 s.
[4] Tuček Petr, Distribuované úložiště dat, diplomová práce FEL ČVUT, Praha, červen 2012
[5] Volf Martin, Distribuované úložiště dat, diplomová práce FEL ČVUT, Praha, leden 2014
[6] George Coulouris, Jean Dollimore, Tim Kindberg, Gordon Blair, Distributed Systems: Concepts and Design, Addison-Wesley, 2011, 978-0132143011

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Peter Macejko, katedra telekomunikační techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **26.03.2019** Termín odevzdání bakalářské práce: **24.05.2019**

Platnost zadání bakalářské práce: **19.02.2021**

Ing. Peter Macejko podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Poděkování

Chtěl bych poděkovat vedoucímu své bakalářské práce panu Ing. Petru Macejkovi za cenné rady, nápovědy a konzultace. Dále bych chtěl poděkovat své rodině, která mě podporovala po celou dobu studia.

Prohlášení

Prohlašuji, že jsem předloženou bakalářskou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, ...

.....

Artem Grigorian

Abstrakt

Cílem této bakalářské práce je navrhnout a následně implementovat grafické uživatelské rozhraní v podobě desktop aplikace, zajišťující přístup jednotlivých klientů k uloženým datům v distribuovaném úložišti. Výsledkem práce je funkční desktop aplikace, která slouží k přístupu do systému s ohledem na bezpečnostní požadavky.

Klíčová slova: distribuované úložiště, grafické uživatelské rozhraní, desktop aplikace

Vedoucí: Ing. Macejko Peter
Katedra telekomunikační techniky,
Technická 1902/2,
160 00 Praha 6

Abstract

The goal of this bachelor thesis is to design and subsequently implement a graphical user interface in the form of a desktop application, ensuring access of individual clients to stored data in a distributed storage. As a result of this thesis the desktop application communicating with a system with respect to security requirements is presented.

Keywords: distributed storage, graphical user interface, desktop application

Title translation: Distributed data storage - desktop UI

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

V této práci se budu zabývat analýzou existujících řešení cloudových úložišť s pohledu funkcionality a struktury uživatelského rozhraní. Konkrétním cílem je navrhnout a následně realizovat grafické uživatelské rozhraní klientské aplikace pro správu souborového úložiště za použití Java knihovny, která slouží jako backend aplikace. Tato Java knihovna se bude starat o komunikaci s distribuovaným úložištěm samotným [1] . Vývojem knihovny se zabývá kolega Bogdan Grigorian [2] .

1.1 Specifikace cíle

Cílem této práce je se seznámit s několika aktuálně existujícími řešeními problémů správy souborových úložišť a na základě výsledků rešerše navrhnout a implementovat klientskou desktop aplikací s grafickým uživatelským rozhraním, která bude sloužit jako součást již existujícího systému distribuovaného úložiště dat. Tento systém je rozdělen do tří částí - první částí je práce Martina Kudrnáče [1] , která se věnuje implementaci samotného ukládání dat; druhou částí je práce Jana Janury [3] , která se zabývá komunikační bránou (přístupový server čili Access server); třetí částí je práce Bogdana Grigoriana [2] , obsahující návrh a následnou implementaci knihovny v jazyce Java, která bude sloužit jako backend budoucí desktopové klientské aplikace.

1.2 Struktura práce

V této kapitole je popsána struktura práce. První část práce se zabývá rešerší momentálně existujících řešení a jejich analýza z pohledu struktury a funkcí uživatelského rozhraní. Podívám se na cloudové systémy pro správu souborů jako Google Drive, OneDrive a Dropbox.

V další části si popíši strukturu a komponenty již existujícího distribuovaného úložiště a Java knihovny, která bude sloužit jako backend budoucí desktopové klientské aplikace.

V třetí části se zaměřím na definici výhod a nevýhod zvoleného programovacího jazyka a zvolím vhodnou platformu pro realizaci uživatelského rozhraní. Navíc tato část bude obsahovat popisy základních myšlenek a návrhy designu

uživatelského rozhraní, kroky implementace samotné klientské aplikace a způsob využívání Java knihovny, kterou implementuje ve své práci kolega Bogdan Grigorian [2] .

Čtvrtá část se bude věnovat testování implementované aplikace.

V závěru budou shrnuty výsledky práce a ohodnoceno splnění cílů práce. Tamtéž bude popsán další postup vývoje aplikace.

Kapitola 2

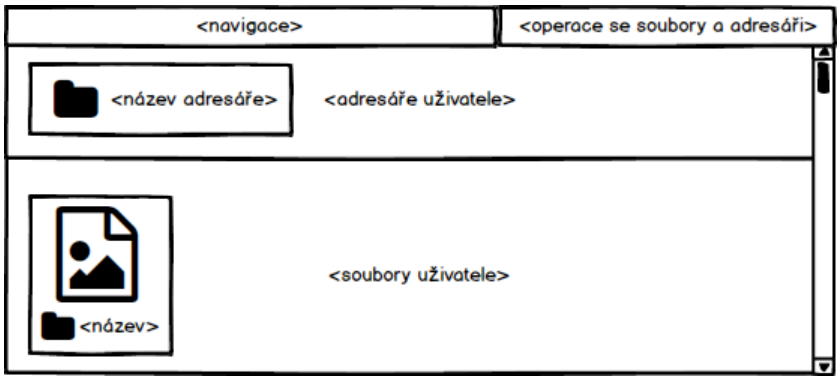
Analýza existujících cloudových řešení

Tato kapitola se věnuje rešerši existujících cloudových úložišť. Smyslem této kapitoly je provest analýzu uživatelských rozhraní webových aplikací a vyjmenovat operací se soubory a adresáři, které této systémy poskytují svým uživatelům. Pro každý systém je vytvořen zjednodušený nákres prvků uživatelského rozhraní a jejich rozložení.

2.1 Google Drive

2.1.1 Uživatelské rozhraní

V první řadě bych se chtěl podívat na webové uživatelské rozhraní aplikace Google Drive [4] a nakreslit zjednodušené schéma rozložení prvků, které je znázorněno na následujícím obrázku 2.1 .



Obrázek 2.1: Uživatelské rozhraní webové aplikace Google Drive - režim "Mřížka"

Menu navigace a operací se soubory a adresáři se nachází v horní části pracovního prostoru uživatelského rozhraní. Oblast prohlížení obsahu úložiště je rozdělena do dvou částí - horní část buď reprezentuje adresáře (ikona a vedle název v každém prvku) v aktuálním adresáři, pokud takové existují, nebo je skryta, pokud aktuální adresář neobsahuje žádné podadresáře. Pod tím se nachází seznam všech souborů aktuálního adresáře (náhled a ikona s jménem vedle pod tím).

Při práci s úložištěm existuje možnost přepínání způsobu zobrazení obsahu z režimu "Mřížka", kde všechny objekty jsou rozloženy na mřížce, do režimu "Seznam" a zpátky. Schéma rozložení prvků v režimu "Seznam" nazelzene v příloze A (viz obrázek A.1).

2.1.2 Operace se soubory a adresáři

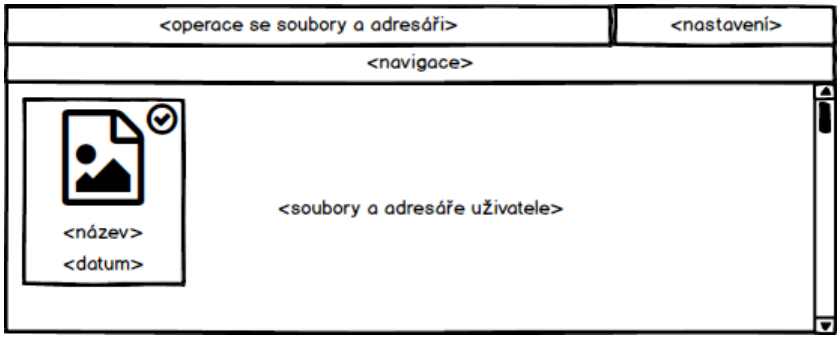
Webové rozhraní pro správu adresářů a souborů poskytuje velkou sadu funkcí:

- Operace s adresáři:
  - vytváření nových adresářů
  - mazání existujících adresářů
  - sdílení existujících adresářů
  - stahování existujících adresářů (v zip)
  - přejmenování existujících adresářů
  - kopírování existujících adresářů
  - přesouvání existujících adresářů
- Operace se soubory:
  - vytváření nových souborů
  - nahrávání souborů ze systému uživatele
  - stahování existujících souborů
  - mazání existujících souborů
  - sdílení existujících souborů
  - přejmenování existujících souborů
  - kopírování existujících souborů
  - přesouvání existujících souborů
- Další funkce:
  - změna barvy ikony adresáře nebo souboru
  - otevření souborů a adresářů v externí aplikace
  - označení adresářů a souborů "hvězdičkou" - přidání do oblíbených

2.2 OneDrive

2.2.1 Uživatelské rozhraní

Dále bych se chtěl podívat na uživatelské rozhraní webové aplikace OneDrive [5] . Struktura rozhraní se skládá ze čtyř základních částí - menu navigace, menu nastavení, menu operací se soubory a adresáři a oblast prohlížení obsahu aktuálního adresáře (viz obrázek 2.2).



Obrázek 2.2: Uživatelské rozhraní webové aplikace OneDrive - režim “Mřížka”

Každý prvek (adresář nebo soubor) má přehled obsahu nebo ikonu, název objektu a datum poslední změny.

Stejně, jako u aplikace Google Drive existuje možnost přepínání zobrazení mezi dvěma základními režimy - “Mřížka” a “Seznam”. Schéma rozložení prvků v režimu “Seznam” nazelzene v příloze A (viz obrázek A.2).

2.2.2 Operace se soubory a adresáři

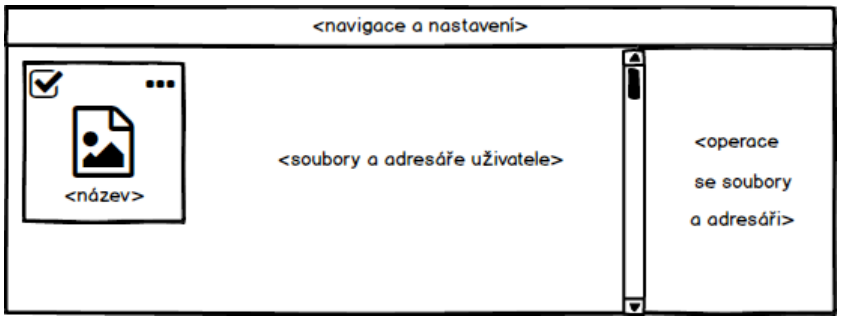
Následující seznam reprezentuje všechny funkce pro práci se soubory a adresáři, které webové rozhraní OneDrive poskytuje svým uživatelům:

- Operace s adresáři:
  - vytváření nových adresářů
  - mazání existujících adresářů
  - sdílení existujících adresářů
  - stahování existujících adresářů (v zip)
  - přejmenování existujících adresářů
  - kopírování existujících adresářů
  - přesouvání existujících adresářů
- Operace se soubory:
  - vytváření nových souborů
  - nahrávání souborů ze systému uživatele
  - mazání existujících souborů
  - sdílení existujících souborů
  - přejmenování existujících souborů
  - kopírování existujících souborů
  - přesouvání existujících souborů
- Další funkce:
  - přehled všech verzí vybraného souboru
  - vytvoření fotoalb z vybraného adresáře

2.3 Dropbox

2.3.1 Uživatelské rozhraní

Dalším cloudovým systémem je webové úložiště, které se jmenuje Dropbox [6] . Jedná se o službu, která je provozována společností Dropbox, Inc. Stejně, jako u předchozích systémů vytvořím schéma rozložení prvků grafického rozhraní (viz obrázek 2.3).



Obrázek 2.3: Uživatelské rozhraní webové aplikace Dropbox - režim "Mřížka"

Interface je hodně podobný OneDrive - každý prvek, reprezentující adresář nebo soubor, má ikonu a název objektu. Navíc v horním rohu je umístěno tlačítko, které slouží pro zobrazení kontextového menu, obsahujícího seznam dostupných operací pro daný objekt.

Aplikace také poskytuje svým uživatelům možnost zobrazit obsah adresáře dvěma způsoby - "Mřížka" a "Seznam". Schéma rozložení prvků v režimu "Seznam" nazelzene v příloze A (viz obrázek A.3).

2.3.2 Operace se soubory a adresáři

Stejně jako předchozí systémy, Dropbox poskytuje všechny základní funkce, potřebné běžnému uživateli pro správu souborů a adresářů:

- Operace s adresáři:
  - vytváření nových adresářů
  - mazání existujících adresářů
  - sdílení existujících adresářů
  - stahování existujících adresářů (v zip)
  - přejmenování existujících adresářů
  - kopírování existujících adresářů
  - přesouvání existujících adresářů
- Operace se soubory:
  - vytváření nových souborů

- nahrávání souborů ze systému uživatele
- mazání existujících souborů
- sdílení existujících souborů
- přejmenování existujících souborů
- kopírování existujících souborů
- přesouvání existujících souborů

- Další funkce:
  - označení adresářů a souborů “hvězdičkou” - přidání do oblíbených

2.4 Zhodnocení

Všechny tři systémy poskytují svým uživatelům funkcionality, které splňují většinu požadavků běžného uživatele webové aplikace. Pro každý systém byla vytvořena zjednodušená schémata jejich grafických uživatelských rozhraní, která budou využita při návrhu systému klientské desktop aplikace.

Kapitola 3

Analýza a popis systému distribuovaného úložiště

V této kapitole jsou stručně popsané části systému distribuovaného úložiště dat.

3.1 Datové centrum

Datové centrum představuje subsystém, který tvoří vlastní úložiště dat a který se skládá z jednotlivých datových center spojených do strukturované DHT sítě. Na uzlích (nodes) v této síti jsou uložena data a každý z nich je schopen obsluhovat požadavky od přístupového serveru (Access Server), o kterém si povíme v další kapitole 3.2 . Návrhem a vývojem tohoto subsystému se zabýval ve své práci Martin Kudrnáč [1] .

Klientská aplikace s uzly zmíněného subsystému napřímo komunikovat nebude. K tomuto účelu slouží přístupový server (Access server).

3.2 Access server

Access server neboli přístupový server je aplikace, která má za úkol zprostředkovávat komunikaci klientské aplikace s vlastním datovým centrem. Hlavním účelem tohoto serveru je odstínění vlastního úložiště dat od klientské části. Podrobnější analýzu a popis Implementace přístupového serveru naleznete v diplomové práci Jana Janury [3] .

3.3 Klientská aplikace

Klientská desktop aplikace, u které se v této práci zabývám návrhem a realizací, je určena pro koncového uživatele systému distribuovaného úložiště a bude mít grafické uživatelské rozhraní. Jejím hlavním úkolem kromě interakce s uživatelem a správy metadat je také komunikace s přístupovou bránou (Access server).

Jeden z pokusů o vytvoření klientské aplikace, která je integrovaná do systému pomocí knihovny FUSE a nemá grafické uživatelské rozhraní, je

popsán v diplomové práci Jana Janury [3] .

3.4 Klientská Java knihovna

Java knihovna, kterou bude realizovat ve své práci kolega Bogdan Grigorian [2] , bude sloužit jako součást klientské aplikace a měla by se starat o komunikaci s přístupovým serverem a o správu metadat. Tato knihovna je implementací business vrstvy a bude plnit funkci backend části desktopové klientské aplikace, kterou budu implementovat ve své práci.

Kapitola 4

Návrh řešení

V této kapitole jsou definované funkční a nefunkční požadavky, popsané zvolené technologie pro tvorbu grafického uživatelského rozhraní a označené případy užití aplikace. Nakonec bylo navrženo grafické uživatelské rozhraní a vytvořen model oken aplikace.

4.1 Specifikace požadavků

4.1.1 Funkční požadavky

Funkční požadavky uvádějí, jaké možnosti má systém poskytovat. Desktop aplikace, kterou v této práci implementuji a pro kterou navrhuji uživatelské rozhraní pro práci s daty v distribuovaném úložišti by měla poskytovat alespoň tyto základní funkce:

- Operace se soubory:
  1. kopírování souborů
  2. přesouvání souborů
  3. přejmenování souborů
  4. změna počtu replikací souborů
  5. změna nastavení sdílení souborů
  6. nahrání souborů ze souborového systému uživatele
  7. stáhnutí souborů a uložení do lokálního souborového systému uživatele
  8. smazání souborů
- Operace s adresáři:
  1. kopírování adresáře
  2. přesouvání adresáře
  3. přejmenování adresáře
  4. změna nastavení sdílení adresáře
  5. smazání adresáře

Při práci se soubory a metadaty musí být zajištěno šifrování a dešifrování objektů na klientské straně (na straně aplikace). Tím pádem musí být zajištěno zabezpečení dešifrovacího klíče, který nesmí opustit aplikaci.

K implementaci core funkcionality aplikace, která se týká práce s daty, může být využita Java knihovna JavaLib vznikající v rámci práce kolegy Bogdana Grigoriana [2] .

Zmíněné funkční požadavky byly zadané vedoucím práce a vznikly z úvodních konzultací.

4.1.2 Nefunkční požadavky

Na základě vlastního rozmýšlení a z konzultací s vedoucím práce byly definované následující kvalitativní (nefunkční) požadavky:

- zajistit možnost použití aplikace bez počítačové myši, pomocí klávesnice (alespoň pro základní operace se soubory a adresáři)
- zajistit možnost použití klávesových zkratek pro základní operace se soubory a adresáři

4.2 Zvolený programovací jazyk

Jako jazyk pro implementaci jsem použil programovací jazyk Java. Zvolil jsem ho z několika důvodů: je jednoduchý, objektově orientovaný, dostatečně rychlý a výkonný a paměť se spravuje pomocí automatického garbage collectoru, což umožňuje rychlejší implementaci. Navíc, knihovna JavaLib, která bude použita jako backend část aplikace je napsaná v jazyce Java, což je také vážným rozhodujícím faktorem při výběru. Ale nejvýznamnější vlastnostmi pro mě jsou nezávislost na architektuře a přenositelnost (vytvořená aplikace běží na libovolném operačním systému/architektuře, podporující Java Runtime Environment).

4.3 Zvolené vývojové prostředí

Při výběru vývojového prostředí byla zvolená platforma IntelliJ IDEA (verzi 2018.3.3 Ultimate Edition) [7] . Podle mého názoru toto prostředí má všechny funkce, potřebné pro kódování a poskytuje sadu zabudovaných editorů/prohlížečů pro různé formáty souborů, jako například SQL, HTML, XHTML, CSS, XSL, XPath a JSON.

IntelliJ IDEA je dostupná ve dvouch verzích: Ultimate Edition a Community Edition. Community Edition je open-source a má jenom základní funkcionality.

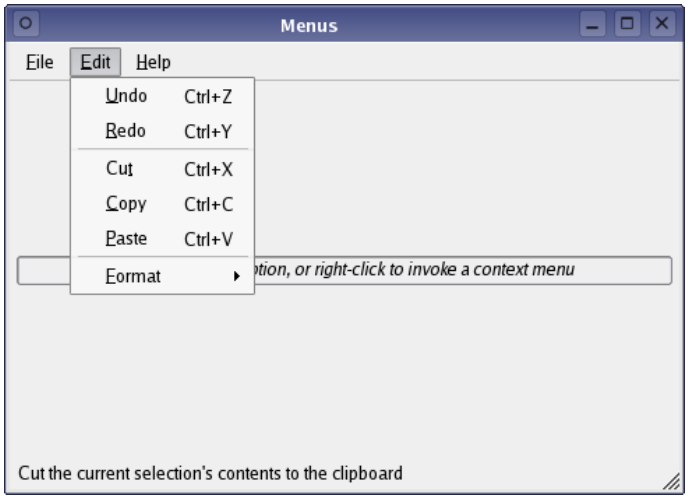
Ultimate Edition je plně vybavené vývojové prostředí, poskytující velké množství nástrojů a podporuje integraci s několika VCS: Team Foundation Server, ClearCase, Perforce a Visual SourceSafe, CVS, Subversion, Git, GitHub, Mercurial.

4.4 Zvolená UI technologie

V současné době existuje velké množství knihoven pro tvorbu grafického uživatelského rozhraní v jazyce Java. V této kapitole bude popsáno několik nejznámějších z nich [8] a ve výsledku zvolená nejvhodnější pro vývoj aplikace.

4.4.1 Qt Jambi

První framework, na který bych se podíval je Qt Jambi [9] . Jedná se o speciální verzi C++ knihovny Qt [10] pro platformu Java, která je momentálně pod LGPL licencí. Za důležitý rozhodující faktor při výběru knihovny pro implementaci považuji to, že Qt Jambi je zastaralá a chybí jí oficiální dokumentace (je dostupná jenom její archivní verze [11]). Navíc tento framework nebude fungovat bez nativních DLL knihoven, což omezuje podporu některých platforem. Na druhou stranu, podle mých zkušeností, tato knihovna je rychlejší v porovnání se standardními Java knihovnami (jako například Swing). Na dalším obrázku 4.1 je znázorněn příklad vzhledu grafických prvků.



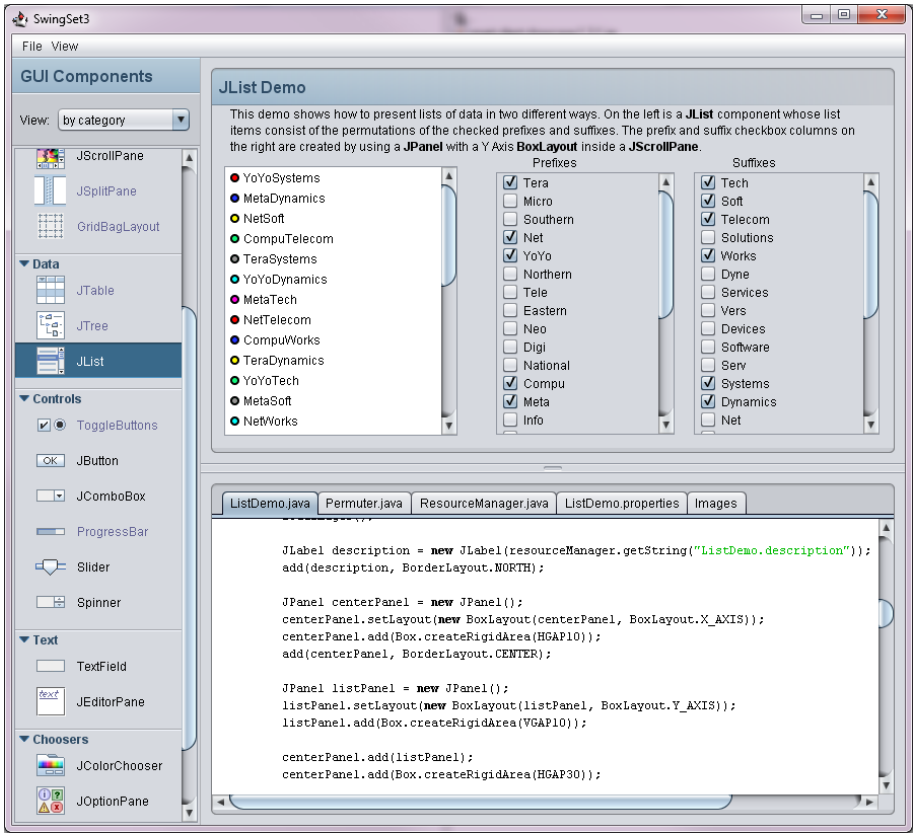
Obrázek 4.1: Příklad vzhledu grafických prvků Qt Jambi [12]

4.4.2 AWT a Swing

Knihovna Swing [13] je součástí Java Foundation Classes, která se používá pro vytváření okenních aplikací. Swing je postavený na AWT API [14] , je nativní a kompletně napsán v jazyce Java, což je jeho hlavní výhodou. Na druhou stranu, zásadní nevýhodou je to, že obě tato řešení jsou zastaralá. Příklad vzhledu grafických prvků je znázorněn na obrázku 4.2 .

4.4.3 JGoodies

JGoodies [16] je postavená na Swing a AWT, a je podobná knihovně SwingLabs čili SwingX [17] . JGoodies je šířen pod licencí BSD Open Source

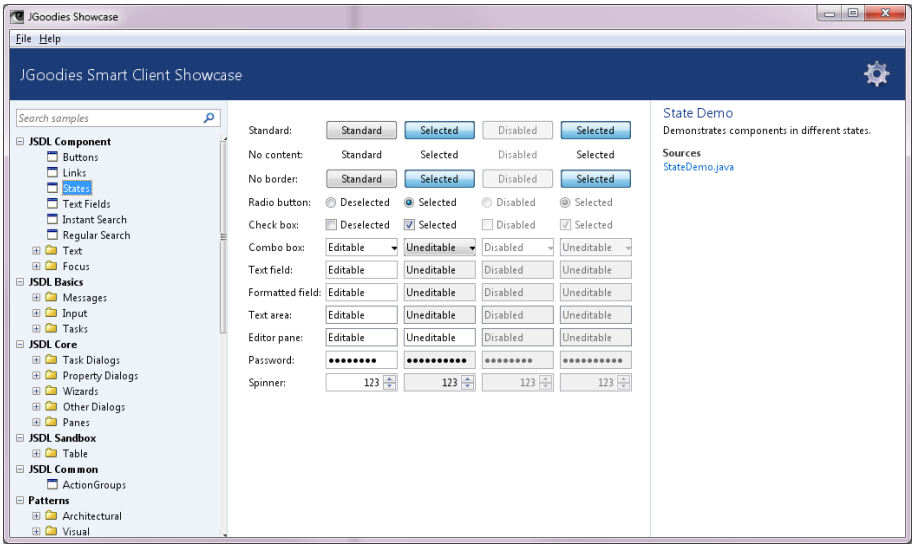


Obrázek 4.2: Příklad vzhledu grafických prvků Swing [15]

License. Z mého pohledu grafické prvky vypadají dost moderně, elegantně a profesionálně (viz obrázek 4.3). Na druhou stranu zásadní nevýhodou pro účely mého projektu je tzv. “steep learning curve” [18] , což znamená že má velmi složitý proces učení. Myslím, že tato knihovna je spíš vhodná pro velké dlouhodobé projekty a pro programátory, kteří mají dobré zkušenosti s knihovnou Swing.

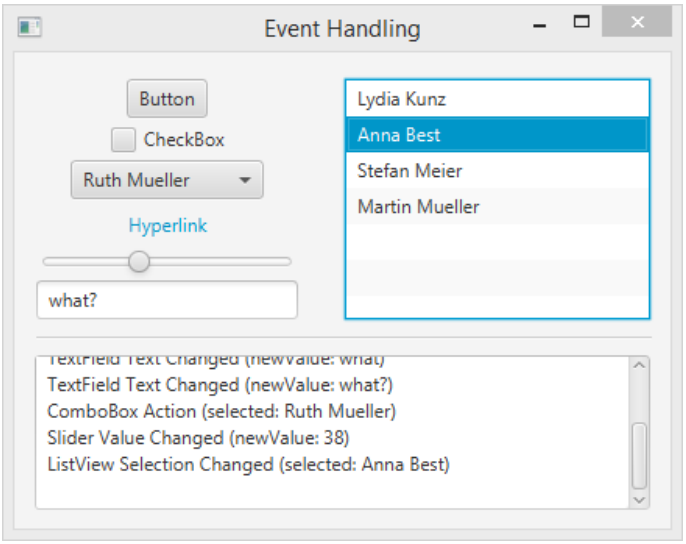
4.4.4 JavaFX

Další zajímavou softwarovou platformou pro vývoj desktop aplikací je JavaFX [20] , kterou vyvinula společnost Sun Microsystems. Tato platforma je především zaměřena na vývoj RIA aplikací (v překladu Bohatá internetová aplikace), což jsou webové aplikace, které především plní funkce tradičních desktopových aplikací [21] . JavaFX je náhradou pro zastaralý Swing, která umožňuje rychlý a snadný vývoj pro internetové prohlížeče a desktopy. Tato knihovna nabízí dobrou podporu pro používání multimediálních prvků (jako video, audio, atd.). Podle mého názoru, největší výhodu přináší přenositelnost mezi platformami - aplikace naprogramovaná na platformě JavaFX se dá spouštět na všech prostředích, podporujících JRE (Java Runtime Environment). Navíc existuje možnost zabalit aplikaci pomocí Native Packaging [22] , což umožní instalaci a spuštění bez nutnosti nainstalovaného JRE na cílovém



Obrázek 4.3: Příklad vzhledu grafických prvků JGoodies [19]

zařízení. Obrázek 4.4 znázorňuje vzhledu grafických prvků JavaFX.



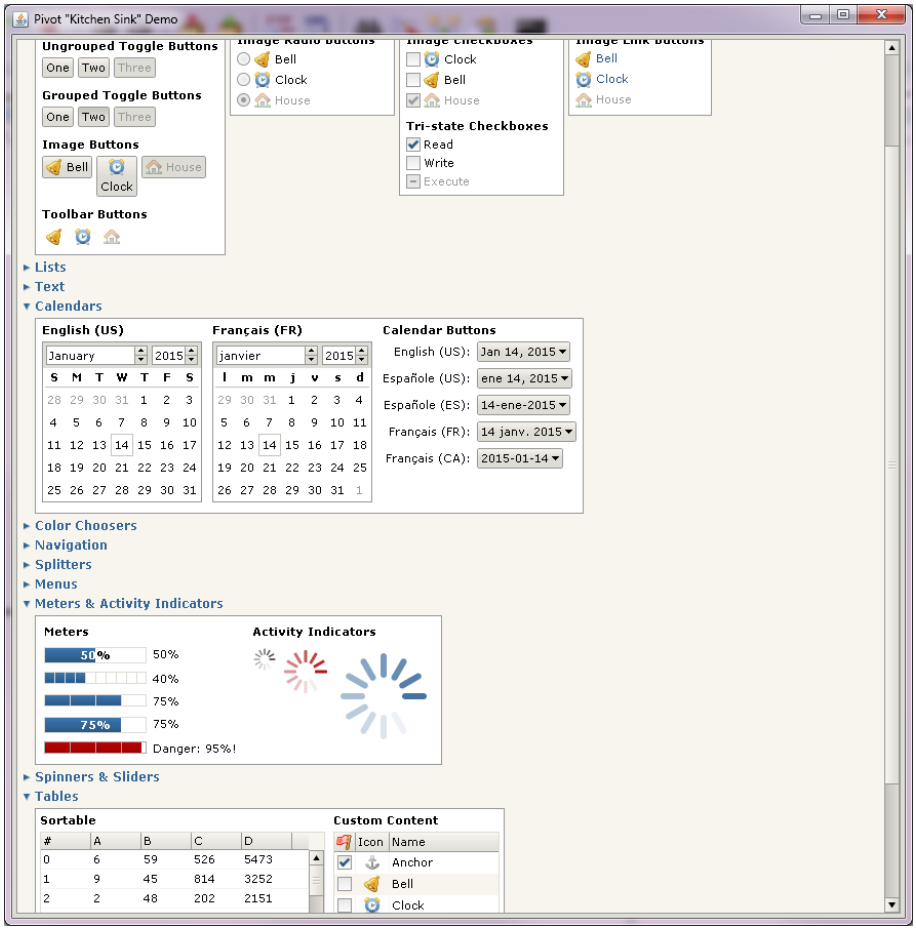
Obrázek 4.4: Příklad vzhledu grafických prvků JavaFX [23]

4.4.5 Apache Pivot

Projekt Apache Pivot [24] pochází z open source a je šířen pod licencí Apache Software Foundation. Silnou stránkou této knihovny je rozsáhlá dokumentace a velké množství příkladů s živými ukázkami, které pomůžou snadno začít její používání.

V porovnání s knihovnami Swing a AWT Apache Pivot má podle mého názoru modernější vzhled komponent (viz obrázek 4.5). Stejně jako u JavaFX a Swing rozložení elementů UI lze definovat pomocí markup jazyka BXML,

založeného na XML.



Obrázek 4.5: Příklad vzhledu grafických prvků Apache Pivot [25]

4.4.6 Výsledek

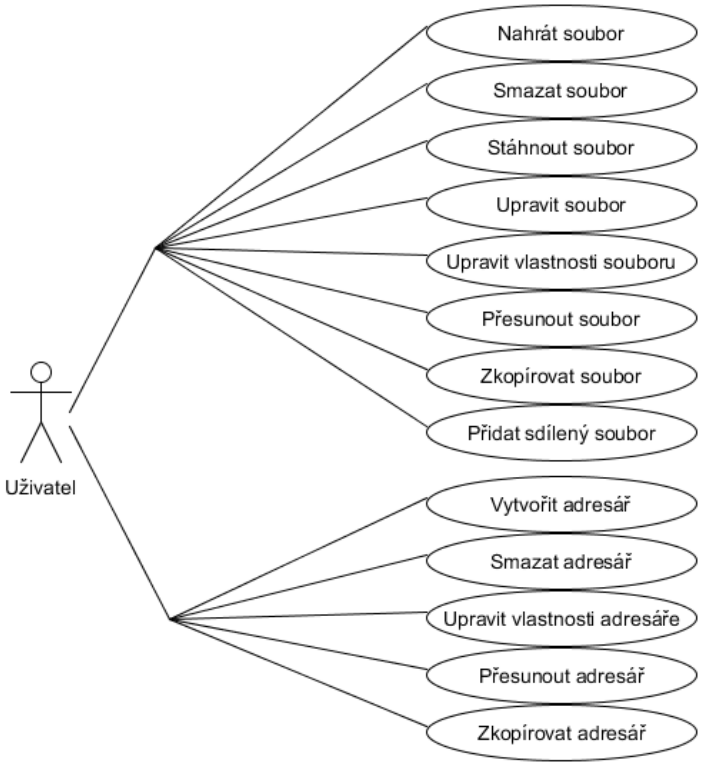
Na základě provedené analýzy jsem zvolil pro implementaci uživatelského rozhraní knihovnu JavaFX z následujících důvodů: není zastaralá (na rozdíl od např. Swing), je přenositelná, používá jazyk podobný XML pro popis rozložení prvků a podporuje CSS styly. Navíc JavaFX má moc dobrou a rozsáhlou dokumentaci, což je pro mě velkým rozhodujícím faktorem, protože s touto platformou nemám žádné zkušenosti.

4.5 Případy užití

Případ užití (angl. use case) popisuje, jak uživatel používá systém k dosažení určitého cíle. Diagram užití popisuje model systému a pomáhá definovat jeho hranice. Diagram případu užití se skládá ze systému, souvisejících případů použití a aktérů [26] .

4.5.1 Diagram případů užití

V systému existuje jenom jedna uživatelská role, kterou nazývám Uživatel. Uživatel má přístup ke všem funkcionalitám systému a může provádět operace s objekty systému, které mu patří, nebo které s ním jsou sdílené. Následující diagram užití 4.6 znázorňuje základní případy užití aplikace.



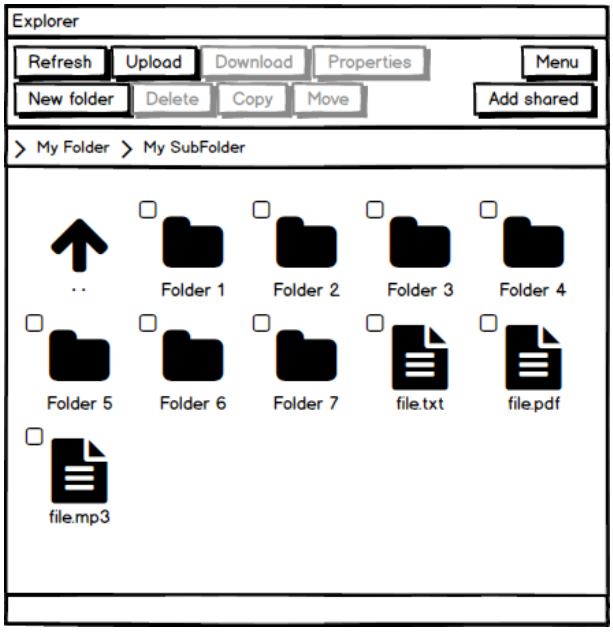
Obrázek 4.6: Diagram případů užití

4.6 Návrh uživatelského rozhraní

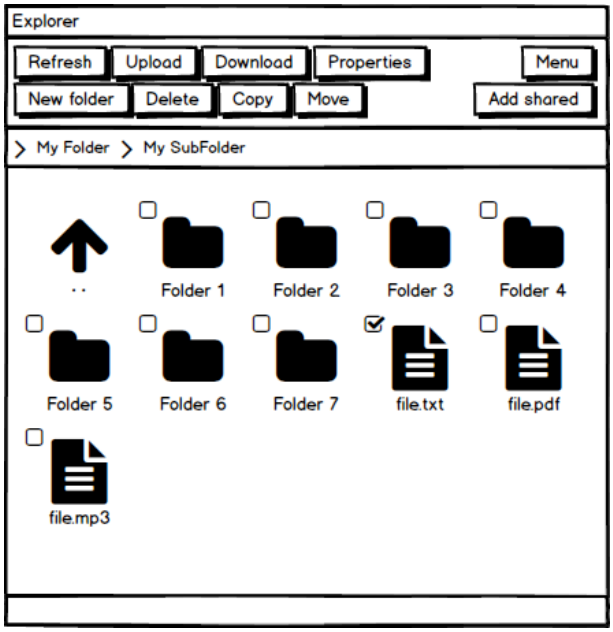
Na základě rešerše existujících cloudových úložišť a zmíněných požadavků jsem navrhnul uživatelské rozhraní klientské aplikace. Pro vytváření návrhů jsem použil aplikaci Balsamiq Mockups 3 [27] , která poskytuje velkou sadu nástrojů pro vytváření grafických rozhraní mobilních, webových a desktop aplikací.

Navržený prototyp hlavního okna aplikace představuje prohlížeč souborů v souborovém systému. Jeho náhled je znázorněn na následujícím obrázku 4.7 .

Nad polem navigace je umístěno horizontální menu, které obsahuje tlačítka pro základní operace s úložištěm a objekty v něm. Tlačítka, která jsou zodpovědná za operace se soubory a adresáři jsou vypnuta, pokud žádný soubor není vybrán (pomocí zaškrtnutí checkboxu). Při výběru alespoň jednoho souboru se tlačítka zapnou, což je znázorněno na dalším obrázku 4.8 .

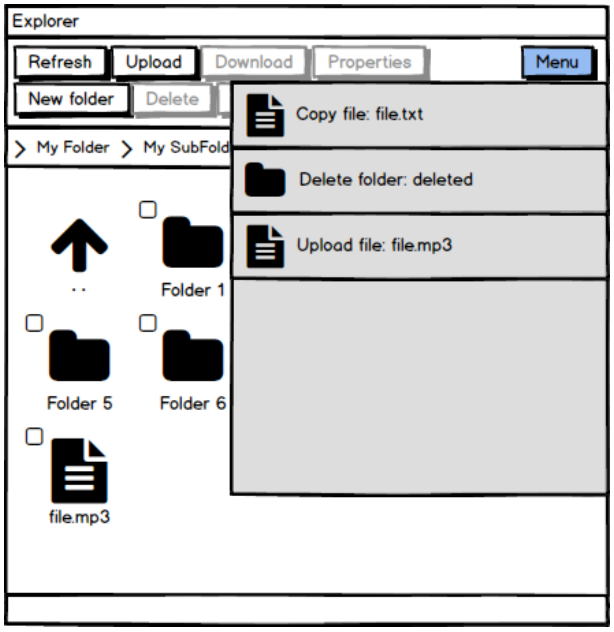


Obrázek 4.7: Model hlavního okna aplikace



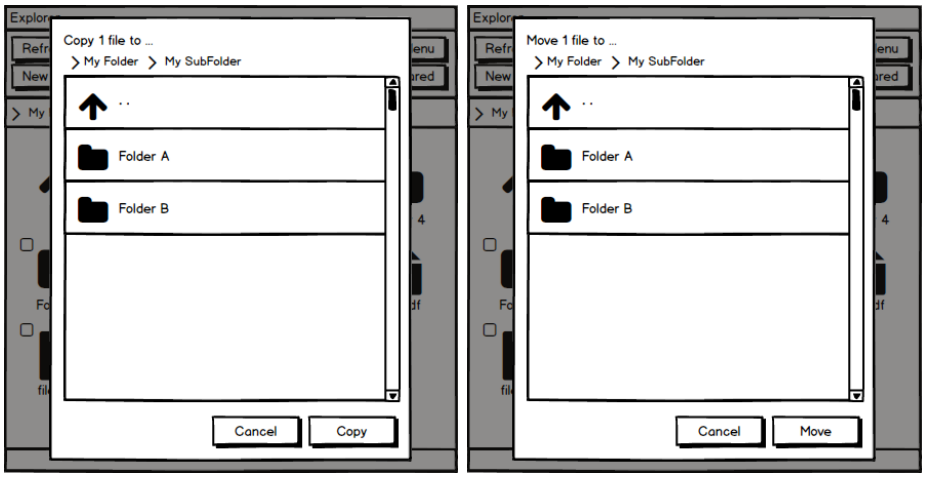
Obrázek 4.8: Model hlavního okna aplikace - jeden soubor je vybrán

Oblast menu také obsahuje tlačítko s názvem “Menu”, které slouží k zobrazení popup okna s frontou probíhajících a ukončených operací v souborovém systému (viz obrázek 4.9).



Obrázek 4.9: Model hlavního okna aplikace - popup menu

Tlačítka “Move” a “Copy” slouží k přemístění a kopírování souborů v souborovém systému. Při zmáčknutí se otevře prohlížeč adresářů, kde uživatel musí vybrat cílový adresář (viz obrázek 4.10).



Obrázek 4.10: Model okna prohlížeče adresářů - operace “move” a “copy”

Pro ostatní operace (“Delete”, “Properties”, “New folder”) byly také vytvořeny prototypy oken (viz. příloha B), která představují dialogová okna, obsahující textová pole a tlačítka pro potvrzení či zrušení operace.

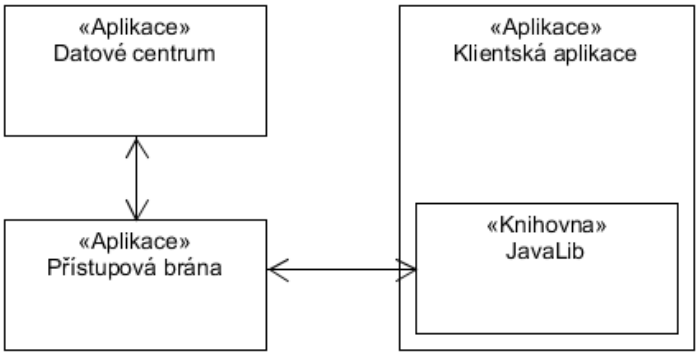
Kapitola 5

Realizace

Tato kapitola se věnuje popisu implementace aplikace a realizací grafického uživatelského rozhraní v jazyce Java na platformě JavaFX.

5.1 Aplikace jako část systému

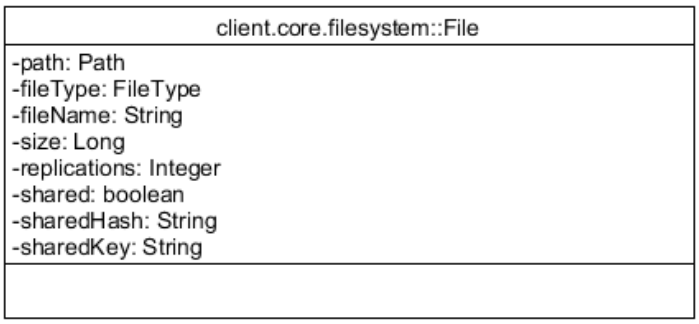
V této kapitole si povíme o systému a roli klientské desktopové aplikace v něm. Následující obrázek 5.1 reprezentuje komunikaci mezi jednotlivými částmi systému.



Obrázek 5.1: Komunikace mezi jednotlivými částmi systému

Datové centrum - subsystém, který tvoří úložiště dat a skládá se z jednotlivých datových center, která jsou spojena do strukturované sítě pomocí DHT. Tento subsystém komunikuje s přístupovým serverem (Access server), který odstiňuje klientskou aplikaci od datového centra a zprostředkovává komunikaci mezi nimi. Veškerá komunikace klientské části s přístupovým serverem probíhá přes knihovnu JavaLib.

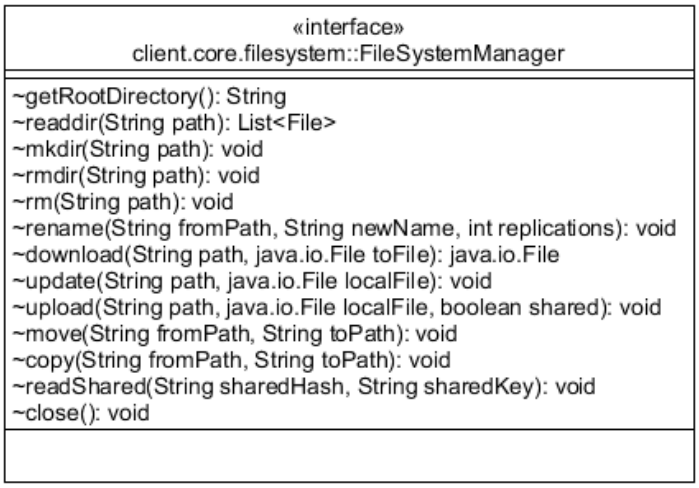
Vzhledem k tomu, že v době realizace klientského programu knihovna JavaLib, která má sloužit jako backend aplikace, nebyla hotová, ve spolupráci s Bogdanem Grigorianem byly navrhnuty základní třídy a rozhraní. Metody tohoto rozhraní by pak měla implementovat zmíněná Java knihovna. Následující třída (viz obrázek 5.2) reprezentuje adresář či soubor.



Obrázek 5.2: Diagram třídy File

Třída File reprezentuje soubor či adresář a zahrnuje 8 vlastností, každá z nich má speciální význam. Pole path je cesta k danému souboru či adresáři v souborovém systému; pole fileType může nabývat jen dvou hodnot: DIR nebo FILE; pole fileName je jméno daného souboru či adresáře; pole size reprezentuje velikost souboru (u adresářů je 0); pole replications je počet replikací souborů (0 pro adresář); pole shared má hodnotu true, když daný soubor či adresář je sdílen, nebo false když není; pole sharedHash a sharedKey obsahuje informace, potřebné pro práci se sdílenými objekty.

Následující objekt (viz obrázek 5.3) reprezentuje rozhraní (interface), obsahující sadu metod.



Obrázek 5.3: Diagram rozhraní FileSystemManager

Interface FileSystemManager popisuje základní operace pro práci se souborovým systémem. Dále bude stručně popsána funkcionalita každé metody.

- String getRootDirectory();
  Metoda getRootDirectory vrací cestu ke kořenovému adresáři systému.

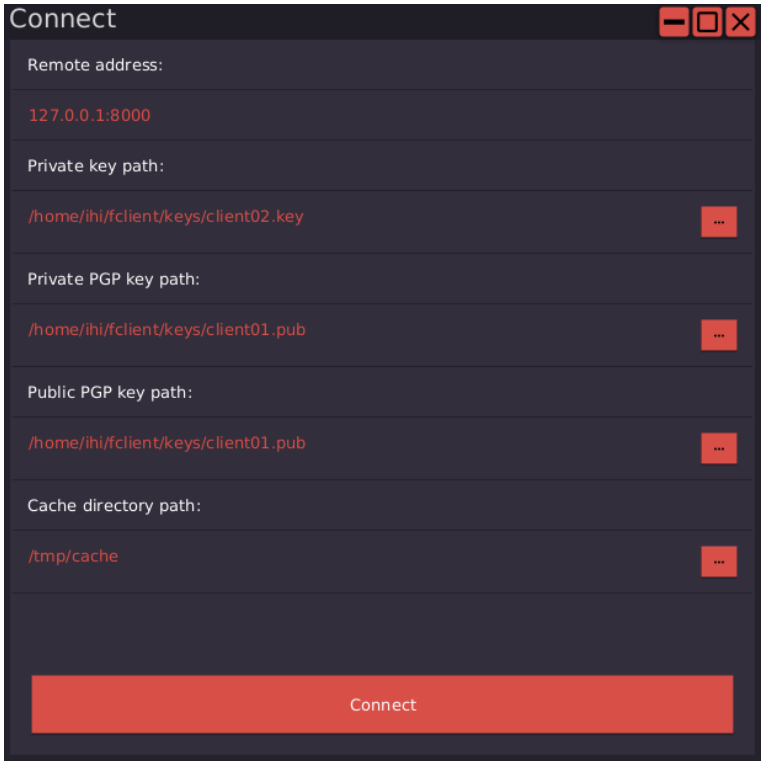
- List<File> readdir(String path);
  Tato metoda přečte obsah adresáře path a vrací jeho obsah jako seznam objektů typu File.
- void mkdir(String path);
  Tato metoda se stará o vytváření nových adresářů.
- void rmdir(String path);
  Metoda rmdir smaže adresář uvedený v parametru.
- void rm(String path);
  Tato metoda má stejný účel, jako rmdir, ale pracuje se soubory.
- void rename(String fromPath, String newName, int replications);
  Metoda rename slouží k přejmenování objektu a změně počtu replikací v systému (pro soubory).
- java.io.File download(String path, java.io.File toFile);
  Tato metoda stáhne uvedený soubor a přesune jeho obsah do cílového objektu, uvedeného v parametru toFile.
- void update(String path, java.io.File localFile);
  Tato metoda nahraje data souborů z lokálního souborového systému a aktualizuje uvedený, v prvním parametru již existující, datový objekt.
- void upload(String path, java.io.File localFile, boolean shared);
  Tato metoda vytvoří prázdný datový objekt a nahraje do něj data souborů z lokálního souborového systému.
- void move(String fromPath, String toPath);
  Tato metoda přesune uvedený soubor či adresář.
- void copy(String fromPath, String toPath);
  Tato metoda zkopíruje uvedený soubor či adresář.
- void readShared(String sharedHash, String sharedKey);
  Tato metoda přidá do kořenového adresáře sdílený soubor.
- void close();
  Tato metoda smaže obsah cache a měla by být zavolaná při ukončení běhu aplikace.

5.2 Uživatelské rozhraní

Struktury a rozložení prvků všech oken uživatelského rozhraní aplikace jsou uloženy do FXML souboru. Jedná se o speciální formát JavaFX, založeném na XML. Tyto soubory se načítají a upravují podle potřeby dynamicky za běhu programu (runtime). Pro každý soubor existuje vlastní CSS soubor se stejným názvem, ve kterém jsou popsané styly grafických prvků. Pro ty FXML soubory, které definují strukturu uživatelských oken, existují speciální

“ovladače” (controllers), které slouží k inicializaci prvků, navěšení posluchačů událostí a volání core metod.

Startovací okno aplikace (viz obrázek 5.4) představuje dialogové okno, ve kterém uživatel musí zadat údaje, potřebné ke správnému běhu programu. Všechna vstupní políčka se validují a pokud nějaké údaje jsou zadané špatně (například byl uveden neexistující soubor, nebo adresa přístupového serveru má špatný formát), uživateli se zobrazí upozornění s chybou.



Obrázek 5.4: Klientská aplikace - Startovací okno

Po vyplnění údajů a zmáčknutí tlačítka Connect se otevře hlavní okno klientské aplikace (viz obrázek 5.5). Toto okno je prohlížečem souborového systému a poskytuje svému uživateli přehled o souborech a podadresářích aktuálního adresáře. Nad polem navigace je umístěno horizontální menu, obsahující tlačítka pro operace s objekty v souborovém systému. V pravém horním rohu je umístěno tlačítko Menu, pro které existuje klávesová zkratka "Tab". Po zmáčknutí se otevře popup okno, znázorňující operace se soubory a adresáři (viz obrázek 5.5).

Pro veškeré elementy horizontálního menu existují klávesové zkratky. Dále bude vyjmenován seznam klávesových zkratek, které jsou k dispozici pro uživatele aplikace:

- CTRL+1 nebo CTRL+R - tlačítko Refresh
- CTRL+2 nebo CTRL+N - tlačítko Create new directory
- CTRL+3 nebo CTRL+U - tlačítko Upload file

Obrázek 5.5: Klientská aplikace - Hlavní okno a popup menu

- CTRL+4 nebo CTRL+D - tlačítko Download file
- CTRL+5 nebo CTRL+I - tlačítko Properties
- CTRL+6 nebo DEL - tlačítko Delete
- CTRL+7 nebo CTRL+X - tlačítko Move
- CTRL+8 nebo CTRL+C - tlačítko Copy
- CTRL+9 - tlačítko Add shared file
- TAB - tlačítko Menu
- CTRL+A - vybrat všechny objekty aktuálního adresáře
- CTRL+Q - zrušit výběr pro všechny objekty aktuálního adresáře

Následující seznam vyjmenuje a stručně popíše existující FXML soubory projektu:

- alert.fxml - okno pro zobrazení upozornění pro uživatele
- confirm.fxml - potvrzující dialog
- editor.fxml - editor textových souborů
- explore.fxml - hlavní okno aplikace, prohlížeč souborového systému
- folders.fxml - prohlížeč adresářů souborového systému
- image_viewer.fxml - prohlížeč obrázků
- menu.fxml - okno popup menu s frontou operaci
- prompt.fxml - dialog se vstupním textovým polem
- properties.fxml - okno pro úpravu vlastností souboru
- start.fxml - startovací okno aplikace

5.3 Použité technologie

V této kapitole jsou vyjmenovány použité technologie pro vývoj aplikace. Každá z těchto technologií je stručně popsána.

5.3.1 Gradle

Gradle [28] je open source systém pro správu build a závislosti (dependencies) Java projektu. Tato technologie svými účely je podobná tradičnímu Apache Ant [29] a Apache Maven [30] , kteŕé používájí pro konfiguraci technologii XML. Gradle ale používá ve svých konfiguračních souborech (settings.gradle a build.gradle) DSL jazyk, založený na jazyce Groovy.

5.3.2 Java Native Access

JNA nebo Java Native Access [31] je knihovna, která se používá pro přístup k "native shared libraries" bez nutnosti psaní dalšího kódu. JNA umožňuje volání metod "native" knihoven tradičním voláním Java metod a používá na to knihovnu JNI. Tato knihovna nebude v tomto projektu použita explicitně, ale její přítomnost znamená nutnost mít zkompilovanou "shared library" pro každou platformu, na kterých desktopová aplikace bude spuštěna. O její využití se bude starat knihovna JavaLib.

5.3.3 JavaFX

JavaFX je softwarovou platformou pro vývoj uživatelských rozhraní pro desktopové, web a mobilní aplikace v jazyce Java. Tato platforma používá technologie RIA, která plní funkce tradičních desktopových aplikací běžících na zařízení uživatele (ne na serveru). Tato technologie plní všechny hlavní požadavky, včetně bezpečnostních.

5.3.4 Git

Git [32] je jedním z distribuovaných systémů správy verzí (DVCS z angl. Distributed Version Control Systém), který zaznamenává všechny změny souborů, a uživatel se tak může kdykoliv vrátit ke konkrétní verzi (tzv. verzování). Pro verzování zdrojového kódu v rámci implementace aplikace byl použit verzovací systém Git. Vzdálený repozitář projektu byl umístěn na serveru katedry telekomunikační techniky kttmine.fel.cvut.cz

Kapitola 6

Testování

V této kapitole je popsán postup a použité způsoby testování.

6.1 Automatické testy

K napsání automatických testů jsem použil knihovnu TestFX [33] , která umožňuje jednoduché a zároveň spolehlivé testování JavaFX aplikace. Pro testování taky byly použity knihovny JUnit [34] a Mockito [35] .

Pomocí unit a integračních testů byly pokryty následující části aplikace:

- AlertWindowTest - testuje "Alert" okno aplikace, které je určeno k zobrazení upozornění pro uživatele.
- ConfirmWindowTest - testuje "Confirm" okno aplikace, které je potvrzujícím dialogem a poskytuje uživateli na výběr kladnou nebo zápornou odpověď.
- MenuPopupWindowTest - testuje popup okno "Menu", které obsahuje frontu probíhajících a ukončených operací.
- PromptWindowTest - testuje "Prompt" dialogové okno vyžadující vstup uživatele.
- PropertiesWindowTest - testuje dialogové okno, pomocí kterého uživatel mění vlastnosti adresářů a souborů.
- StartWindowTest a StartWindowWithPropertiesTest - testuje startovací okno aplikace, které je zobrazeno jako první při spuštění aplikace. V něm uživatel musí zadat údaje, potřebné pro připojení k ostatním částem distribuovaného úložiště a správnému běhu programu.

Pomocí unit testů byly otestované pomocné třídy a volání metod třídy pro přístup k souborového systému.

6.2 Uživatelské testování

Uživatelského testování se zúčastnili celkem tři účastníci, kteří měli k dispozici funkční prototyp aplikace a seznam úkolů, které bylo potřeba splnit:

1. Nastartujte aplikaci.
2. Vytvořte dva nové prázdné adresáře v kořenovém adresáři a pojmenujte je “S1” a “S2”.
3. V adresáři “/S1” vytvoříte nový prázdný adresář s jménem “SS1”.
4. Do adresáře “/S2” nahrajte textový soubor a přejmenujte ho na “TXT.txt”
5. Do adresáře “/S1/SS1” nahrajte soubor s obrázkem a přejmenujte ho na “PIC.pic”
6. Do souboru “/S2/TXT.txt” napište libovolný text a uložte změny.
7. Stáhněte soubor “/S2/TXT.txt” na svůj počítač a smažte adresář “/S2”.
8. Otevřete soubor “/S1/SS1/PIC.pic”

Testování probíhalo v knihovně NTK na počítači s operačním systémem Debian, spuštěném ve virtualizačním nástroji Oracle VM VirtualBox [36] . Ostatní části systému (Datové Centrum a Access server) byly spuštěny na lokálním stroji. Po splnění úkolů jsem od uživatelů dostal zpětnou vazbu.

Průběh testování (číslování odpovídá seznamu úkolů):

1. Uživatel bez problémů nastartoval aplikací.
2. Uživatel vytvořil adresář “/S1”. Při vytváření adresáře “/S2” použil klávesovou zkratku “CTRL+N”.
3. Uživatel pomocí klávesové zkratky “CTRL+N” vytvořil adresář “/S1/SS1”.
4. Uživatel otevřel adresář “/S2” a nahrál tam soubor “test_text.txt”. Uživatel přejmenoval soubor “test_text.txt” na “TXT.txt”.
5. Uživatel otevřel adresář “/S1/SS1” a nahrál tam soubor “test_image.png”. Uživatel přejmenoval soubor “test_image.png” na “PIC.pic”
6. Uživatel otevřel adresář “/S2” a kliknutím na soubor “TXT.txt” otevřel editor souborů a přidal řádek textu. Uživatel uložil změny kliknutím na tlačítko “Save”.
7. Uživatel stáhnul soubor “TXT.txt” na svůj počítač a zkontroloval, jestli jeho změny byly uloženy. Uživatel otevřel kořenový adresář, zaškrtnul checkbox adresáře “S2” a smazal adresář pomocí klávesy “DEL”
8. Uživatel otevřel adresář “/S1/SS1” a kliknutím na soubor “PIC.pic” spustil prohlížeč obrázků.

Poznámky od účastníků testování (číslování odpovídá seznamu úkolů):

1. “Aplikace se příliš pomalu načítá.”
5. “Chybí možnost ručně upravovat cestu otevřeného adresáře.”
6. “Chybí možnost hledání souborů či adresářů podle jména.”

6.3 Zhodnocení

Během vývoje aplikace se testovala ručně a pomocí automatických testů. Byly nalezeny a opraveny drobné chyby, které vznikly během implementace; žádné zásadní chyby se neobjevily. Nakonec aplikace byla otestována z pohledu koncového uživatele a ukázaly se některé kvalitativní nedostatky, neovlivňující funkcionalitu aplikace. Jejich oprava se v této práci řešit nebude kvůli nedostatku času.

Kapitola 7

Závěr

Cílem práce bylo navrhnout a implementovat grafické uživatelské rozhraní v podobě desktop aplikace pro práci s daty v distribuovaném úložišti. V rámci realizace projektu se mi podařilo vytvořit funkční program, který poskytuje veškeré požadované funkcionality pro práci se soubory i adresáři a s metadaty a který splňuje naznačené bezpečnostní požadavky. Funkce jádra programu plní Java knihovna s názvem JavaLib, kterou navrhnul a implementoval ve své práci Bogdan Grigorian [2] .

Výsledné řešení bylo úspěšně manuálně otestováno na pilotní implementaci datového úložiště a také pomocí automatických unit a integračních testů.

V budoucnu se plánuje přidání dalších funkcionalit do aplikace, kterými jsou: možnost ručně upravovat cestu otevřeného adresáře, možnost hledání souborů či adresářů podle jména, seřazení zobrazených objektů podle jejich vlastností a rozšířené možnosti sdílení souborů a adresářů.

Literatura

[1] M. Kudrnáč, "Distribuované úložiště dat - správa dat," 2014, diplomová práce. Praha. ČVUT FEL, Katedra počítačů.

[2] B. Grigorian, "Distribuované úložiště dat - java knihovna," 2019, bakalářská práce. Praha. ČVUT FEL, Katedra počítačů.

[3] J. Janura, "Distribuované úložiště dat - klientská část," 2014, diplomová práce. Praha. ČVUT FEL, Katedra počítačů.

[4] "Google drive: Free cloud storage for personal use," https://www.google.com/drive/, accessed : 15/05/2019.

[5] "Microsoft onedrive," https://onedrive.live.com/, accessed : 15/05/2019.

[6] "Dropbox," https://www.dropbox.com/, accessed : 15/05/2019.

[7] "Intellij idea: The java ide for professional developers by jetbrains," https://www.jetbrains.com/idea/, accessed : 15/05/2019.

[8] S. community. (2013) Java gui frameworks. what to choose? swing, swt, awt, swingx, jgoodies, javafx, apache pivot? [Online]. Available: https://stackoverflow.com/questions/7358775/java-gui-frameworks-what-to-choose-swing-swt-awt-swingx-jgoodies-javafx

[9] "Qt jambi · github," https://github.com/qtjambi, accessed : 15/05/2019.

[10] "Qt | cross-platform software development for embedded & desktop," https://www.qt.io/, accessed : 15/05/2019.

[11] "Qt jambi reference documentation," https://doc.qt.io/archives/qtjambi-4.5.2_01/com/trolltech/qt/qtjambi-index.html, accessed : 15/05/2019.

[12] N. C. and/or its subsidiary(ies), "Menus example," 2009. [Online]. Available: http://artem.gratchev.com/2015/01/choosing-java-gui-for-raspberry-pi-dashboard-application/

[13] "Package javax.swing," https://docs.oracle.com/javase/8/docs/api/javax/swing/package-summary.html, accessed : 15/05/2019.

[14] “Package java.awt,” https://docs.oracle.com/javase/8/docs/api/java/awt/package-frame.html, accessed : 15/05/2019.

[15] artem, “Swing,” 2015. [Online]. Available: http://artem.gratchev.com/2015/01/choosing-java-gui-for-raspberry-pi-dashboard-application/

[16] “Jgoodies – we make java look good and work well,” http://www.jgoodies.com/, accessed : 15/05/2019.

[17] “Swing labs,” http://www.swinglabs.com/, accessed : 15/05/2019.

[18] K. Lentzsch. (2004) The jgoodies forms framework. [Online]. Available: https://ptolemy.berkeley.edu/ptolemyII/ptII8.1/ptII/doc/whitepaper.pdf

[19] artem, “Jgoodies,” 2015. [Online]. Available: http://artem.gratchev.com/2015/01/choosing-java-gui-for-raspberry-pi-dashboard-application/

[20] “Javafx,” https://openjfx.io/, accessed : 15/05/2019.

[21] T. community. (2014) Rich internet application (ria). [Online]. Available: https://www.techopedia.com/definition/2531/rich-internet-application-ria

[22] “Deploying javafx applications: Self-contained application,” https://docs.oracle.com/javafx/2/deployment/self-contained-packaging.htm, accessed : 15/05/2019.

[23] C. community, “Our example application,” 2014. [Online]. Available: https://code.makery.ch/blog/javafx-8-event-handling-examples/

[24] “Apache pivot,” https://pivot.apache.org/, accessed : 15/05/2019.

[25] artem, “Apache pivot,” 2015. [Online]. Available: http://artem.gratchev.com/2015/01/choosing-java-gui-for-raspberry-pi-dashboard-application/

[26] V. paradigm community. (2016) What is use case diagram? [Online]. Available: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/

[27] “Balsamiq for desktop | balsamiq,” https://balsamiq.com/wireframes/desktop/, accessed : 15/05/2019.

[28] “Gradle build tool,” https://gradle.org/, accessed : 15/05/2019.

[29] “Apache ant - welcome,” https://ant.apache.org/, accessed : 15/05/2019.

[30] “Maven – welcome to apache maven,” https://maven.apache.org/, accessed : 15/05/2019.

[31] “Java native access (jna) - github,” https://github.com/java-native-access/jna, accessed : 15/05/2019.

[32] “Git,” https://git-scm.com/, accessed : 15/05/2019.

[33] “Github - testfx/testfx: Simple and clean testing for javafx,” https://github.com/TestFX/TestFX, accessed : 15/05/2019.

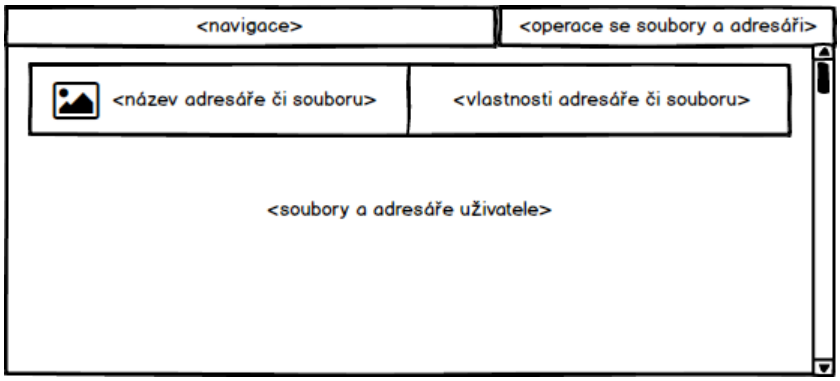
[34] “Junit – about,” https://junit.org/junit4/, accessed : 15/05/2019.

[35] “Mockito framework site,” https://site.mockito.org/, accessed : 15/05/2019.

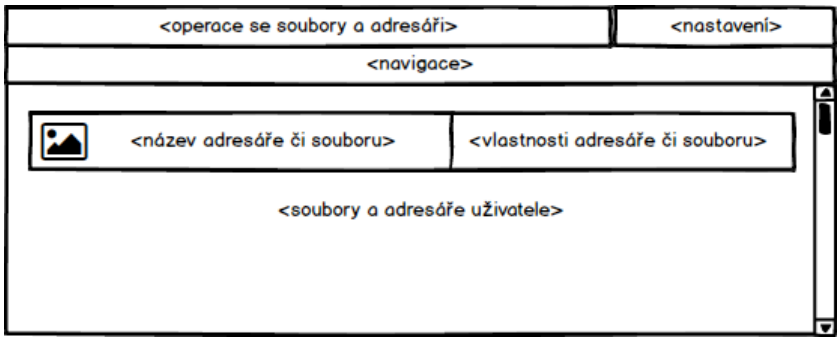
[36] “Oracle vm virtualbox,” https://www.virtualbox.org/, accessed : 15/05/2019.

Příloha A

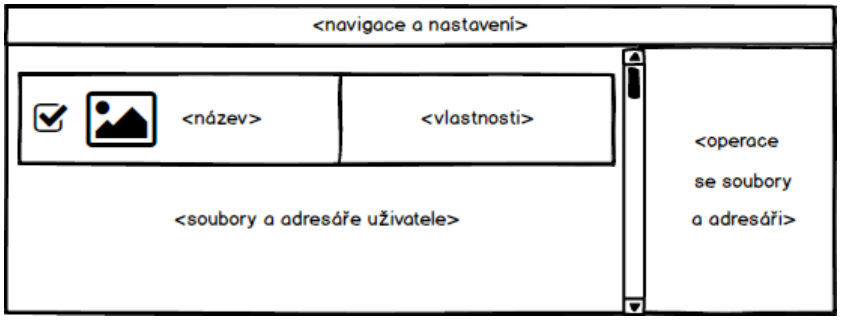
Uživatelské rozhraní webových aplikací



Obrázek A.1: Uživatelské rozhraní webové aplikace Google Drive - režim "Seznam"



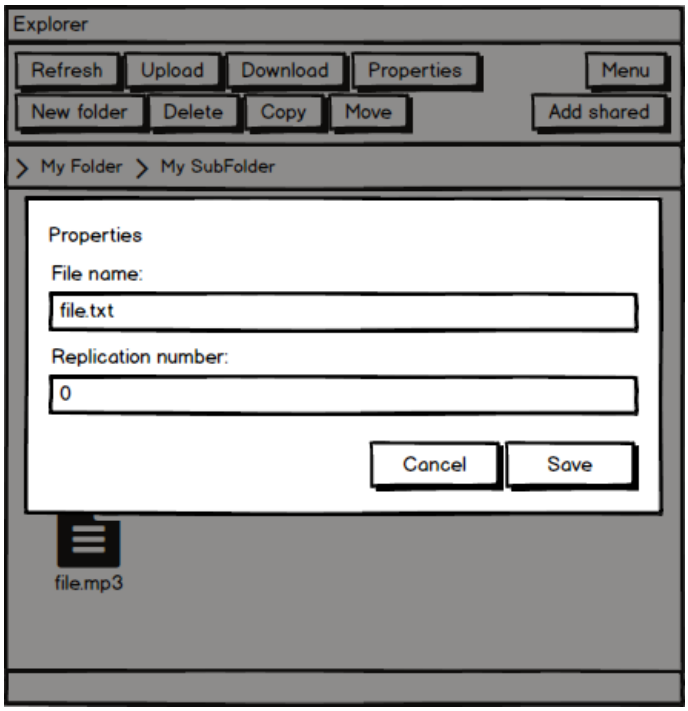
Obrázek A.2: Uživatelské rozhraní webové aplikace OneDrive - režim "Seznam"



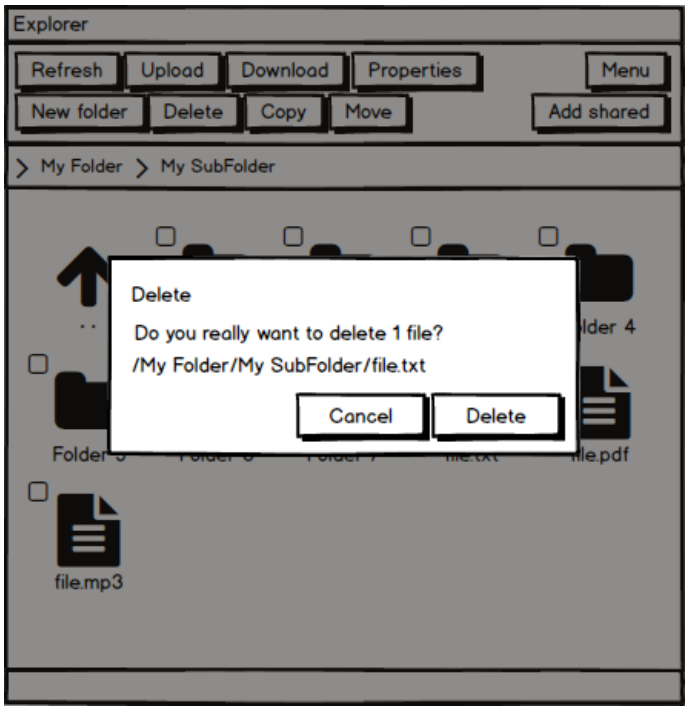
Obrázek A.3: Uživatelské rozhraní webové aplikace Dropbox - režim “Seznam”

Příloha B

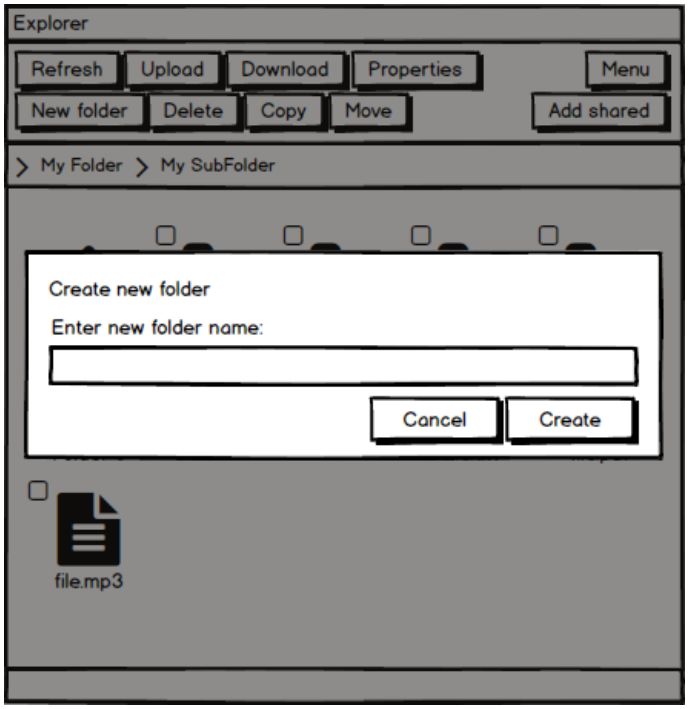
Model oken aplikace



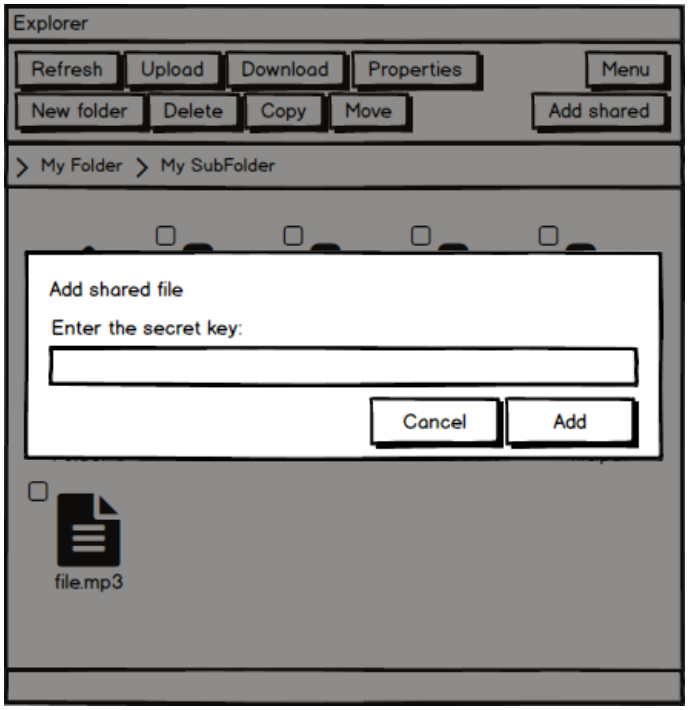
Obrázek B.1: Model dialogového okna “Properties”



Obrázek B.2: Model dialogového okna “Delete”



Obrázek B.3: Model dialogového okna “Create new folder”



Obrázek B.4: Model dialogového okna “Add Shared”

Příloha C

Struktura priloženého CD

/
- `app` ..... zdrojový kód aplikace
- `text` ..... zdrojový kód zprávy ve formátu LaTeX