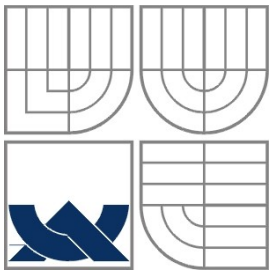
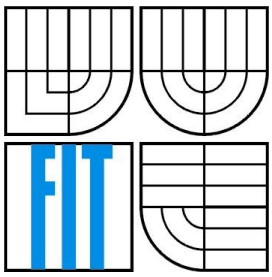


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ONLINE SOUBOROVÝ MANAŽER

ONLINE FILE MANAGER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL ŠVEC

VEDOUCÍ PRÁCE
SUPERVISOR

DOC. RNDR. PAVEL SMRŽ, PH.D.

BRNO 2009

Abstrakt

Práce popisuje návrh a implementaci dvoupanelového souborového manažera s oddělenou klientskou a serverovou částí.

Abstract

Thesis describes design and implementation of dual-panel online file manager, with client application separated from server app.

Klíčová slova

správce souborů, klient, server, XHTML, Javascript, PHP, internet, JSON, jQuery,

Keywords

file manager, client, server, XHTML, Javascript, PHP, Internet, JSON, jQuery

Citace

Online souborový manažer

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Doc. RNDr. Pavla Smrže Ph.D.
Další informace mi poskytli Bc. František Vybíral, CEO společnosti asisio s.r.o.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Michal Švec
19.5.2009

Poděkování

Rád bych poděkoval Doc. RNDr. Pavlu Smržovi Ph.D. za vedení práce a Bc. Františku Vybíralovi za rady při návrhu a vývoji.

Obsah

Obsah....1
1 Úvod....3
2 Teoretický úvod....4
2.1 PHP....4
2.2 Javascript....4
2.3 AJAX....5
2.4 API....6
2.5 JSON....6
2.6 jQuery....6
3 Specifikace....8
3.1 Požadavky....8
3.2 Zvolené technologie....8
4 Serverová aplikace....10
4.1 Implementace....10
4.2 Komunikace s klientskou aplikací....11
4.3 API....12
4.3.1 Seznam souborů....12
4.3.2 Přesun a kopírování souborů....13
4.3.3 Mazání souborů....13
4.3.4 Stažení souboru....13
4.3.5 Vytvoření adresáře....14
4.4 Odpověď API....14
4.5 Zabezpečení....15
5 Klientská aplikace....16
5.1 Požadavky....16
5.2 Uživatelské rozhraní....16
5.3 Implementace....17
5.3.1 Operace se soubory....17
5.3.2 Komunikace se serverem....18
5.3.3 Načítání seznamu souborů....18
5.3.4 JS knihovny a pluginy....19
6 Srovnání s dostupnými alternativami....21
6.1 Dostupné alternativy....21
6.2 Výhody a nevýhody....21

7 Závěr....22
7.1 Uživatelské hodnocení....22
7.2 Možná rozšíření....25
7.3 Výsledné zhodnocení....25
Literatura....26
Seznam příloh....27

1 Úvod

AJAXové aplikace dobývají svět Internetu. Čím dál tím více služeb implementuje AJAX do svých stránek. Ať už se jedná o našeptávače u vyhledávání, dynamické načítání obsahu stránek nebo kontrolu a odesílání formulářových prvků. AJAX dal také vzniknout velmi populárnímu pojmu WEB 2.0, který bývá označován jako budoucnost Internetu. Jedná se právě o webové stránky využívající AJAX a obecně Javascript na většinu operací, které by jinak vyžadovaly opětovné načtení celé stránky, čímž zlepšují celkový dojem uživatele z prohlížení stránky. Na českém Internetu je bohužel dobrých WEB 2.0 stránek málo. Proto je nutné rozšiřovat řídké řady AJAXových aplikací využívající model klient-server, kde klienta reprezentuje právě WEB 2.0 aplikace pracující se serverovým API.

Tato bakalářská práce se věnuje implementaci dvoupanelového online správce souborů využívajícího výhod WEBu 2.0. Ten najde své využití v mnoha typech online aplikací. Reálné nasazení ho čeká v redakčním systému, může být však použit i v elektronickém obchodu pro výběr obrázků k produktům, pro správu souborů na FTP bez možnosti přístupu k FTP klientovi atd.

V další kapitole jsou rozvedeny a popsány použité technologie, což přispívá k lepšímu porozumění problému a pochopení implementace.

Třetí kapitola shrnuje požadavky na danou aplikaci. Obsahuje výčet základních a volitelných funkcí, které by neměly v implementaci chybět. Rozebírá také důvody pro zvolení použitých technologií.

Ve čtvrté kapitole je podobně rozebrána serverová aplikace. Popisuje implementaci této aplikace, způsob komunikace s klientskou aplikací a obsahuje také popis API spolu s informacemi o jeho zabezpečení.

Klientská aplikace je podrobně popsána v páté kapitole. Jsou v ní vysvětleny požadavky na uživatelské rozhraní i funkčnost, rozebrána implementace komunikace, funkcí a operací se soubory. Nachází se zde i popis použitých pluginů a knihoven.

V šesté kapitole je výsledný program porovnán s ostatními dostupnými řešeními. Diskutovány jsou jak jeho výhody, tak jeho nedostatky.

Závěrečná, sedmá kapitola poskytuje výsledky uživatelského testování, jsou v ní uvedeny také nápady na další vylepšení a rozšíření aplikace z hlediska uživatelského rozhraní i funkčnosti.

2 Teoretický úvod

2.1 PHP

PHP je skriptovací jazyk určený původně pouze pro web. Vyvinul se však tak, že umožňuje skriptování z příkazového řádku nebo může být použit v samostatných desktopových aplikacích. Byl vytvořen v roce 1995 dánem Rasmusem Lerdorfem. Nyní je vyvíjen skupinou s názvem *The PHP Group*. PHP je volně dostupný software vydávaný pod PHP licencí (je kompatibilní s GNU GPL licencí, avšak obsahuje omezení na používání názvu PHP), používaný převážně pro vývoj webových aplikací. Zdrojové kódy mohou být vkládány přímo do HTML, které je zpracováváno webovým serverem. Ten zpracuje PHP kód a vrátí webovou stránku jako výstup. PHP interpret může být umístěný na většině webových serverů a na téměř všech operačních systémech a platformách zdarma.

Zdrojové kódy jsou uzavřeny mezi značky *<?php* a *?>*. Cokoliv mimo ně je vypsáno rovnou na výstup. PHP má dva typy omezovacích znaků *<?php* a *<?*. Použití kratší varianty však není doporučeno kvůli horší přenositelnosti. Syntaxe a klíčová slova PHP jsou podobná jazykům se syntaxí vycházející z jazyka C. Podmínka *if* a cykly *for* a *while* jsou stejné jako u jazyků C, C++ nebo Java. Proměnné jsou v PHP zapisovány se znakem *$* na začátku.

Podpora objektového programování byla do PHP přidána ve verzi 3 a vylepšena ve verzi 4. Ve verzi 5 byla kompletně přepsána a dnes je na velmi dobré úrovni. PHP má také velmi kvalitní podporu knihoven pro zpracování textu a grafiky, poskytování přístupu k databázím (mj. MySQL, Oracle, PostgreSQL) a podporu mnoha internetových protokolů (HTTP, SMTP, POP3, IMAP, FTP, LDAP atd.). Velmi známá je kombinace operačního systému Linux, webového serveru Apache, databáze MySQL a PHP nazývaná LAMP.*[1]*

2.2 Javascript

Javascript je multiplatformní, objektově orientovaný skriptovací jazyk. V dnešní době se používá jako interpretovaný programovací jazyk pro webové stránky. Je vkládaný přímo do HTML stránky nebo jako odkaz na externí javascriptový soubor. Obvykle slouží pro ovládání různých interaktivních prvků grafického uživatelského rozhraní, nebo pro vytváření animací a efektů. V poslední době je však masivně využíván v technologii AJAX (viz kapitola 2.3).

Autorem Javascriptu je Brendan Eich ze společnosti Netscape. Původní název byl Mocha, který se později změnil na LiveScript a poté na Javascript. Poprvé byl představen v roce 2005 spolu s příchodem prohlížeče Netscape 2.0. Standardizovaná verze Javascriptu se jmenuje ECMAScript.

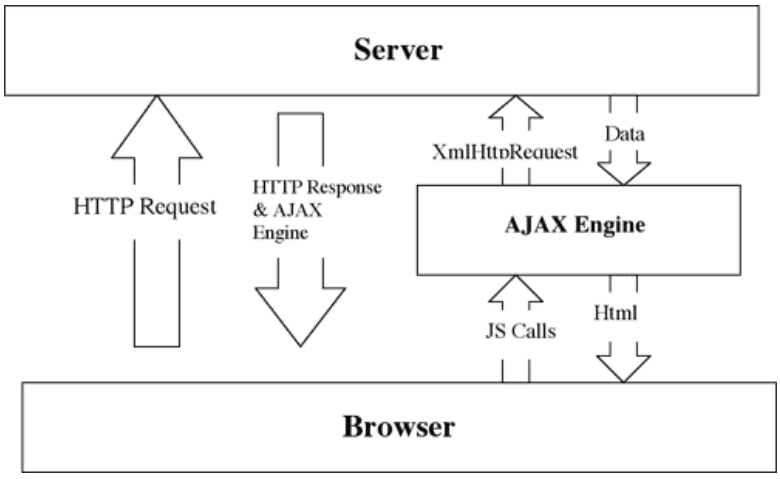
Z ní byly odvozeny další implementace, jako je například ActionScript, známý především ve spojení s Macromedia Flash. Firma Microsoft představila jako reakci na Javascript svůj skriptovací jazyk JScript, který je s Javascriptem kompatibilní a je podporován platformou .NET.

Javascript je syntaxí (stejně jako PHP) odvozený od jazyka C. Je také jako PHP slabě typovaný a k jeho běhu je potřeba interpret. Ten je v případě JavaScriptu většinou přímo v internetovém prohlížeči. Existují však aplikace, které dokáží spouštět JavaScript samostatně mimo webové stránky a umožňují tak nahrazení dávkových souborů javascriptovými skripty.

Vzhledem k tomu, že je Javascript spouštěný na uživatelském počítači, je potencionálním zdrojem bezpečnostních rizik. Proto má několik omezení. Nemůže například pracovat se soubory, a také nemůže přistupovat na jiné stránky, než ze kterých je spouštěný. Celý Javascriptový kód je spouštěný v takzvaném sandboxu. To je bezpečnostní mechanismus, který se používá pro oddělený běh programů. Program spouštěný v tomto režimu má omezenou a kontrolovanou sadu prostředků pro svůj běh, jako např. přístup k paměti, místo na disku, operační paměť atp.*[2][3]*

2.3 AJAX

AJAX (Asynchronous JavaScript and XML) je obecné označení pro technologie vývoje interaktivních webových aplikací, které mění obsah svých stránek bez nutnosti jejich znovunačtení. Na rozdíl od klasických webových aplikací poskytují uživatelsky příjemnější prostředí, ale vyžadují použití moderních webových prohlížečů. Podobně jako LAMP nebo DHTML není AJAX technologie, ale skupina používaných technologií. Zahrnuje technologie HTML (popřípadě XHTML), CSS, DOM a Javascript pro práci s HTML dokumentem a metodou pro asynchronní výměnu dat s webovým serverem. Typicky se používá *XMLHttpRequest*, ale je možné použít například i *Iframe*. Pro výměnu dat nemusí být použito pouze XML, ale i JSON nebo HTML.



Obrázek 1: Princip funkce AJAXu[7]

Hlavní výhodou AJAXu je odstranění nutnosti načítat znovu celou stránku kvůli změně dat. Tím se redukuje zmenšuje a datová náročnost. Dalším plusem je možnost např. interaktivně kontrolovat formulářové prvky. Při použití AJAXu pro změnu obsahu stránky se redukuje také počet připojení na server. CSS soubory a soubory se skripty jsou načteny pouze jednou.

Nevýhody AJAXu jsou zřejmé. Požadavky nejsou registrovány prohlížečem do historie a nereagují tak na tlačítko zpět. Většina vyhledávacích robotů nepodporuje skriptovaní, a tak nejsou stránky načtené AJAXem registrovány do vyhledávače. Dalšími nevýhodami je složité záložkování načtených stránek, špatná podpora těchto technologií v prohlížečích pro PDA a mobilní telefony a bezpečnostní rizika, která nemusela být vývojáři webových aplikací zohledněna při navrhování API pro AJAXové požadavky.*[4]*

2.4 API

Jedná se o rozhraní pro podporu programování aplikací. API je soubor rutin, datových struktur, tříd a objektů nějaké knihovny, programu nebo například jádra operačního systému, které mohou být volány a používány z jiného programu. API přesně definuje vstupní a výstupní hodnoty a způsoby volání.*[8]*

2.5 JSON

JFormát pro výměnu dat nezávislý na jakékoliv platformě. Dají se jím reprezentovat data organizovaná v polích nebo objektech (asociativních polích). Hlavními datovými entitami jsou seznam hodnot (alternativa pole) a dvojice *název:hodnota*. Všechny hodnoty jsou reprezentovány řetězci, jelikož JSON je textový formát. Hlavním nedostatkem JSON je, že neumožňuje definovat znakovou sadu, a také neumožňuje zapsat znak konce řádku. Alternativou k JSON je značkovací jazyk XML. Oproti němu má však JSON výhodu v jednoduchosti značek, které nezabírají tak velký počet znaků.

JSON je nejvíce používaný pro přenos dat přes síť v procesu zvaném serializace. Jeho hlavní použití je ve spojení s AJAXovými aplikacemi. Přestože byl původně podmnožinou jazyka Javascript a je používán hlavně tímto jazykem, je to formát dat nezávislý na programovacím jazyku. *[5][6]*

2.6 jQuery

jQuery je odlehčená javascriptová knihovna, která ulehčuje interakci mezi Javascriptem a HTML. Je navržená tak, aby oddělovala chování XHTML stránky, stejně jako CSS odděluje vzhled stránky. To je také někdy nazýváno jako „nevtíravý Javascript“. jQuery umožňuje vybírání HTML elementů

ze stránky, definování události, manipulování s CSS, efekty a animace, práci s AJAXem, rozšiřitelnost a použití pluginů. Pro selekci má také podporu Xpath.

Mezi hlavní výhody jQuery patří propracovaná dokumentace a příklady, snadné použití, zajímavé funkce a to, že se drží striktně ve svém jmenném prostor. Nemodifikuje nic, co mu nepatří. V porovnání s ostatními javascriptovými frameworky jako je např. Django, Dojo nebo MooTools nepostrádá jQuery žádnou funkci. Jediná omezení nebo nedostatky má jQuery oproti frameworku Dojo v kreslení grafických prvků, grafů a statistik a nedostatečné podpoře standardu WAI-ARIA, která je však plánována v další verzi.[9][10]

3 Specifikace

3.1 Požadavky

Online souborových manažerů je v dnešní době dostupných velké množství. Drtivá většina z nich zvládá základní operace se soubory jako je kopírování, přesun, mazání, nové adresáře atd. To jsou samozřejmé funkce, které by neměly chybět u žádného softwaru pro práci se souborem. Dalším prvkem navíc pak může být například archivace označených souborů, či rozbalení archivu. Souborový manažer, který je předmětem této práce, je určen pro nasazení v systémech typu elektronický obchod nebo redakční systém. Ty mají svá specifika, od kterých se odvíjí požadavky na návrh a chování celé aplikace.

Tak jako všechny ostatní manažery musí i popisovaná aplikace umožňovat standardní sadu příkazů pro práci se soubory. To znamená přejmenování, kopírování, přesouvání a mazání. Tyto operace musí umět provádět jak se soubory, tak i adresáři. Dále musí umožňovat označení více souborů najednou a provedení stejných operací, jako s jednotlivým souborem. Další základní vlastností je vytvoření nové složky. Nadstandardními funkcemi je archivace označených souborů a filtrování výpisu. To musí být prováděno nad celým název souboru tak, aby se daly filtrovat i soubory jednoho typu podle koncovky.

Pro potřeby elektronického obchodu, kde se v jedné složce s obrázky produktů může nacházet velké množství souborů, musí být efektivně vyřešena práce s velkým seznamem souborů. To je provedeno postupným načítáním podle pozice posuvné lišty. U obrázků musí být vyřešen jejich náhled .

Celá architektura bude řešena jako klient-server. To znamená dvě úplně oddělené aplikace. Klientská aplikace bude k serveru přistupovat pomocí API a jeho výsledky bude zobrazovat v okně prohlížeče. Je však možné ji realizovat i jako desktopovou aplikaci. Server musí být napsaný v programovacím jazyce, který může být spouštěn na webovém serveru a dovoluje přístup přes Internet. Je tak možné použít například PHP, Ruby, Python, ASP nebo Javu.

3.2 Zvolené technologie

Pro implementaci byla vybrána kombinace programovacích jazyků PHP (serverová část) a XHTML a Javascript (klientská část). Pro výměnu dat byl místo XML, které je sice dnes velmi populární, ale pro tyto účely velmi těžkopádné, zvolen formát JSON. Ten je nativně podporován jazykem Javascript a je velmi nenáročný na výslednou velikost přenášených dat.. Javascript byl zvolen kvůli jeho skvělé

podpoře ve všech webových prohlížečích, vzrůstající popularitě a možnosti provádět s ním úpravy XHTML obsahu stránky. To celé v technologii AJAX je prostředek, který se pro tyto potřeby perfektně hodí.

Pro snadnější práci s Javascriptem byla zvolena knihovna jQuery, která je v dnešní době jednou z nejlepších dostupných prostředků pro usnadnění vybírání, úprav, doplňování XHTML obsahu a práce s AJAXem. Umožňuje také přidávat na stránky animace a efekty.

Jazyk PHP byl zvolen pro jeho velké rozšíření, popularitu a jednoduchost . Je volně dostupný, nativně podporuje všechny potřebné operace se soubory. Je možné ho provozovat na všech dnešních webových serverech (Apache, Lighttpd, IIS) a poskytuje ho drtivá většina webhostingových společností. Má také dobrou podporu formátu JSON (rozšíření *JSON*, které je také ve většině konfigurací aktivováno).

4 Serverová aplikace

Serverová aplikace je pouze skript umístěný na webovém serveru zpracovávající požadavky od klientských aplikací definované v API. Po vyřízení požadavku se běh skriptu ukončí.

4.1 Implementace

Ke spuštění a správné funkci je potřeba mít v nastavení PHP povoleno rozšíření *JSON* a upravit konfigurační soubor *server.ini*. Ten má standardní syntaxi ini souborů a jsou v něm nastavené následující parametry:

- **defaultDir** – adresář na serveru, který bude pro klientskou aplikaci představovat kořenový adresář.
- **dirFirst** – booleovská hodnota, která v případě *true* zajistí, že adresáře budou ve výpisu souborů zobrazované dříve než soubory. V případě *false* budou položky zobrazeny podle abecedy bez ohledu na to, jestli se jedná o soubor nebo adresář.
- **listCount** – standardní počet souborů pro výpis.

Serverová aplikace je implementována jako několik PHP tříd.

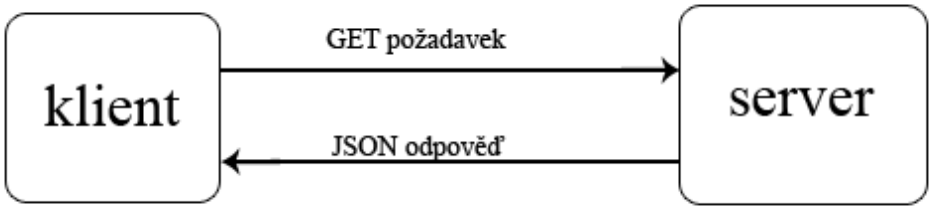
- Třída *ofmServer* se stará o zpracování příchozích požadavků a o práci se souborem.
- Třída *zipfile* umožňuje vytváření zip archivu.
- Třída *imgThumb* poskytuje možnost vytváření zmenšenin obrázků pro náhledy v aplikaci.

Hlavní třídou je *ofmServer*, která se stará o veškerou činnost. Funkce *processApi()* je prováděcí funkce, která zpracovává parametry URL a podle nich rozhoduje o provedené akci. V případě standardních akcí se soubory je volána jednotná funkce *doStuff()* pro mazání, kopírování a přesun. Ty mají totiž téměř totožné parametry a rozlišují se pouze voláním příslušné operace. Další funkce pro manipulaci se soubory je *doRename()*, která má rozdílné parametry oproti předchozím operacím, proto je volána přímo a ne přes pomocnou funkci *doStuff()*. Pro vytváření adresáře slouží funkce *makeDir()*, seznam souborů vrací funkce *fileList()*, která však slouží pouze pro zpracování chyb a ošetření vstupních parametrů. Skutečné procházení složky zajišťuje funkce *readDirectory()*, která má na starosti filtrování, řazení a zpracovávání atributů souborů. Stažení souboru zajišťuje funkce *getFile()*, která podle koncovky detekuje MIME typ souboru a nastaví podle něj parametr HTTP hlavičky *Content-type*: na správnou hodnotu. Dále nastaví *Content-Disposition: attachment,* což způsobí, že prohlížeč provede akci asociovanou s daným typem souboru. U obrázků to znamená otevření a u jiných typů souborů jejich stažení. To samé se provede u funkce *zipIt()*, která vrátí zkomprimované zadané soubory a složky. Tato funkce využívá třídu *zipfile*, jejímž autorem je Joshua Townsend (http://www.gamingg.net). Všechny tyto operace používají standardní funkce PHP. Není

použito funkce pro spouštění příkazů terminálu, takže je kód plně funkční pod jakýmkoliv operačním systémem.

4.2 Komunikace s klientskou aplikací

Všechny požadavky jsou přijímány metodou GET. To znamená, že parametry jsou zadány přímo v URL. Tento přístup má dvě nevýhody. Všechny tyto požadavky mohou být získány z historie prohlížeče, proxy serveru, logů web serveru nebo http hlavičky Referer. To zvyšuje bezpečnostní rizika. Druhou nevýhodou je, že Microsoft Internet Explorer má maximální povolenou délku URL 2083 znaků.*[11]* Při složitějších požadavcích tak může aplikace přestat fungovat. Tyto dva problémy by bylo možné vyřešit použitím požadavku typu POST. V tomto případě je ale problém na straně klienta, jelikož podpora POST požadavků na jinou doménu, u kterých dochází k automatickému zpracování JSON odpovědi je velmi špatná. Naopak pro GET požadavky stejného typu má jQuery přímo funkci, která zajistí vše potřebné. Ta využívá požadavku JSONP, který může být proveden i na jinou doménu než je aktuálně běžící kód. To klasický AJAX XmlHttpRequest nedokáže. JSONP se od klasické JSON odpovědi liší přidáním callback funkce k odpovědi. Ve výsledném programu je tedy ponecháno řešení s požadavky typu GET, které však může být jednoduše změněno s příchodem verze jQuery, která bude umět jednoduše pracovat s POST požadavky na jiné domény.



Obrázek 2: Model komunikace

Aplikace ve většině případů vrací asociativní pole, které obsahuje všechny potřebné parametry (viz. podkapitola API), které jsou zakódovány funkcí *json_encode()* a poslány na standardní výstup funkcí *echo()*. Pokud klientská aplikace podporuje komprimované spojení metodou gzip (parametr HTTP_ACCEPT_ENCODING v superglobální proměnné *$_SERVER*), je celý výstupní buffer zpracováván metodou *ob_gzhandler()*, která zajistí poslání příslušných hlaviček a zkomprimování výsledných dat.

4.3 API

Akce se definuje parametrem *mode*. Dostupných je několik akcí. Ty rozlišují počet, formát dalších parametrů a výstupní data. Parametry API jsou navrženy přímo pro potřeby správce souborů s operacemi v jedné složce. To v praxi znamená, že pokud se provádí operace s několika soubory, je vždy určen adresář ve výchozí adresářové struktuře a seznam všech souborů v něm, se kterými se operace provádí. Neuvádí se tedy celá cesta k souborům, ale pouze relativní cesta vůči zadanému adresáři. Tím pádem nelze standardně pracovat se soubory z jiných adresářů než je zvolený adresář, nebo adresáře jemu podřazené. Lze to však obejít použitím „..“. Tento přístup má výhodu v případě omezení délky URL prohlížečem nebo webovým serverem. Parametry a jejich názvy jsou navrženy tak, aby byly i u rozdílných operací co nejvíce podobné a intuitivní. Výchozí adresář je tedy vždy označen jako *src,* cílový jako *dest*, seznam souborů je vždy *files* atd..

4.3.1 Seznam souborů

Seznam souborů je pole struktur obsahujících veškeré potřebné informace o souborech. Spolu se seznamem se vrací ještě cesta k aktuálnímu adresáři (z důvodu její úpravy do správného tvaru při zadání znaku pro vyskočení v adresářové struktuře o jednu položku výš „..“), status, zda byla operace úspěšná a celkový počet souborů v adresáři.

parametry	**volitelnost**	**datový typ**	**Popis, hodnota**
mode	povinný	řetězec	*list*
dir	volitelný	řetězec	Cesta ke zvolenému adresáři. V případě, že není zadaný se použije cesta ke kořenovému adresáři.
from	volitelný	celé číslo	Index prvního souboru, od kterého se začíná vypisovat. Pokud není zadaný, začíná se od 0.
cnt	volitelný	celé číslo	Počet souborů, které se vypisují. Standardně je nastaveno na 30.
filter	volitelný	řetězec	Filtrovací řetězec. Výpis souborů bude obsahovat pouze položky obsahující tento podřetězec. Aplikuje se na celý název souboru i s příponou.

Tabulka 1: Parametry operace výpis souborů

Výsledná URL tak může vypadat například takto:
http://www.example.com/?mode=list&dir=/&from=45&cnt=15
Veškeré hodnoty by měly být upraveny funkcí pro převedení všech nealfanumerických znaků na zástupné symboly začínající %. V PHP je to funkce *urlencode()*, v javascriptu funkce *escape()*.

Serverová aplikace automaticky tyto hodnoty dekóduje. Cesta k adresáři je kontrolována, a v případě, že ukazuje do adresáře, kam uživatel nemá přístup, je vráceno chybové hlášení.

4.3.2 Přesun a kopírování souborů

Tyto operace mají shodné parametry, proto jsou uváděné dohromady. Seznam souborů je zde uváděn jako pole z důvodu minimalizace počtu požadavků na server při operaci s více soubory.

parametry	**volitelnost**	**Datový typ**	**Popis, hodnota**
mode	povinný	řetězec	*copy* nebo *move*
src	povinný	řetězec	Zdrojový adresář.
dest	povinný	řetězec	Cílový adresář.
files	povinný	pole řetězců	Pole jmen souborů, se kterými je prováděna akce.

Tabulka 2: Parametry operací přesun a kopírování

Výsledná URL tak může mít např. následující tvar:
http://www.example.com/?mode=copy&src=/&dest=/tmp&files[]=foo.bar&files[]=foo.baz

4.3.3 Mazání souborů

Tato operace je totožná s operacemi přesun a kopírování, ale nevyžaduje parametr dest.

parametry	**volitelnost**	**Datový typ**	**Popis, hodnota**
mode	povinný	řetězec	*delete*
src	povinný	řetězec	Zdrojový adresář.
files	povinný	pole řetězců	Pole jmen souborů, se kterými je prováděna akce.

Tabulka 3: Parametry operace mazání

4.3.4 Stažení souboru

Operace využívaná také při zobrazování obrázků má jediný parametr, kterým je název souboru. Server sám zjistí jaký je MIME typ a podle toho nabídne příslušný soubor ke stažení se správnou hlavičkou.

parametry	**volitelnost**	**Datový typ**	**Popis, hodnota**
mode	povinný	řetězec	*get*
file	povinný	řetězec	Zdrojový adresář.

Tabulka 4: Parametry operace stažení souboru

Výsledná URL pro stažení obrázku je např.:
http://www.example.com/?mode=get&file=foo.png

4.3.5 Vytvoření adresáře

Vytvoření nového adresáře na zadané cestě.

parametry	volitelnost	Datový typ	Popis, hodnota
mode	povinný	řetězec	*mkdir*
src	povinný	řetězec	Adresář ve kterém bude nový adresář vytvořen.
name	povinný	řetězec	Název nového adresáře.

Tabulka 5: Parametry operace vytvoření nového adresáře

Ukázková URL:

http://www.example.com/?mode=mkdir&src=/foo&name=bar

4.4 Odpověď API

Po provedení operace vrací server asociativní pole kódované do formátu JSON, které obsahuje následující položky:

status	Řetězec reprezentující úspěšnost požadované operace. Může nabývat následujících hodnot *EOK* -operace proběhla úspěšně *OUTEMPTY* – nejsou dostupné další položky k načítání *LISTEMPTY* – prázdný adresář *EWRONGPATH* – zvolen špatný adresář *EDATAMISS* – chybí některý z parametrů *ECOPY* – nelze zkopírovat *EMOVE* – nelze přesunou *EDEL* – nelze vymazat
dir	Při výpisu souboru je v této hodnotě uložena cesta k aktuálnímu adresáři
items	Dvojrozměrné pole obsahující zobrazené soubory. Každá položka tohoto pole obsahuje následující parametry: **ext** – koncovka souboru (v případě adresáře obsahuje hodnotu [dir]) **name** – název souboru bez koncovky **time** – čas poslední změny souboru ve formátu YYYY-mm-dd **rights** – práva k souboru v číselném formátu, např. 775 **size** – velikost souboru převedená na nejvhodnější formát, např. 2MB, 12kB atd.
count	Celkový počet souborů v adresáři

Tabulka 6: Položky pole, vráceného serverovou aplikací

4.5 Zabezpečení

Zabezpečení na serverové straně je velmi důležité V případě špatného ošetření zobrazovaného adresáře by bylo možné číst a zobrazovat systémové nebo soukromé složky. Toto je zde ošetřeno použitím PHP funkce *realpath()*, která např. řetězec „*/var/log/../../etc/passwd*“ převede na „*/etc/passwd*“. Ta je použita na celou cestu k požadovanému adresáři a je porovnána s adresářem nastaveným v konfiguračním souboru serveru. Porovnání se provádí průchodem obou cest po jednotlivých adresářích. Při první neshodě je detekována chyba a výpis souborů neproběhne. Jiná bezpečnostní rizika nebyla na straně serveru zaznamenána.

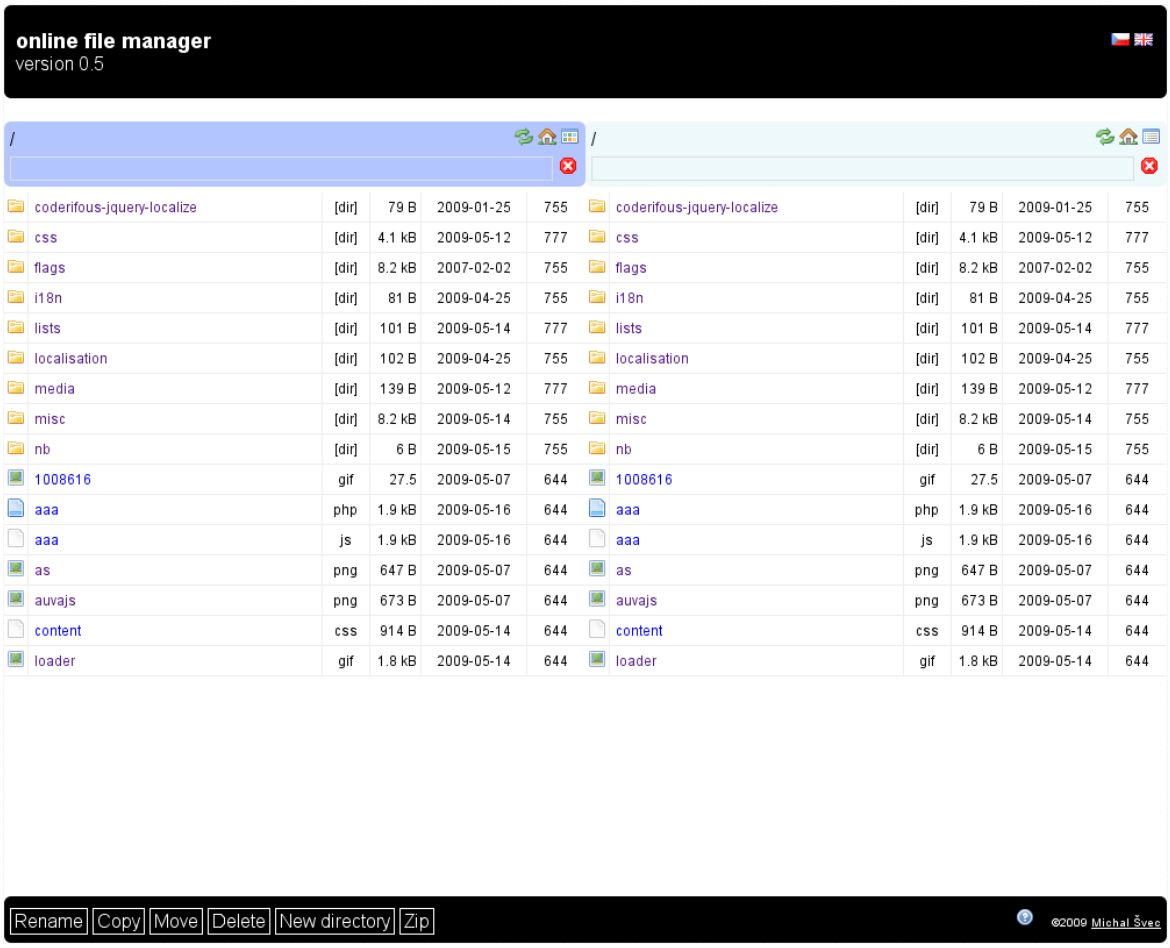
5 Klientská aplikace

5.1 Požadavky

Základním požadavkem na klientkou aplikaci byla její snadná pozdější úprava a rozšiřitelnost o další funkce a jednoduchá začlenitelnost do již hotového řešení, a to buď použitím jen její části, nebo jednoduchou úpravou designu. Správce by měl obsahovat dva samostatné panely s možností manipulace souborů mezi dvěma právě otevřenými složkami.

5.2 Uživatelské rozhraní

Uživatelské rozhraní je navrženo tak, aby bylo co nejjednodušší, přehledné, snadno upravitelné, ale přesto vzhledově přijatelné. Skládá se ze tří hlavních částí: horní lišta, panely se soubory a spodní lišta..



Obrázek 3: Layout aplikace

Horní lišta je pouze informativní. Na levé straně je zobrazen název aplikace, popř. logo. Vpravo je panel pro přepínání jazyků. Na tuto lištu je možné umístit další systémové funkce, jako je např. menu zbytku webu, formulář pro nahrání nového souboru, nebo ji kompletně změnit tak, aby odpovídala zbytku celého webu. Ve zdrojovém kódu odpovídá divu s identifikátorem *top*.

Panely se soubory jsou jádrem celé aplikace. Nad výpisem souborů se nachází informační lišta, kde je zobrazena cesta k aktuálně zobrazenému adresáři a funkční ikony, kterými je možno se dostat snadno na kořenový adresář, obnovit seznam souborů nebo přepnout režim zobrazení. V poslední verzi má aplikace dva režimy zobrazení. Standardně je aktivované tabulkové zobrazení. Druhý režim je dlaždicové zobrazení. Tyto zobrazení také určují, jaké informace o souboru uživatel uvidí. Dlaždicové zobrazení je stručné a zobrazuje pouze název a ikonu souboru, zatímco tabulkové zobrazení poskytuje veškeré informace.

Spodní lišta je jako ve většině dvoupanelových správců souborů určena tlačítkům pro operace se soubory. Zvolená operace se provádí vždy nad aktivním panelem, který je volen jako zdrojový a druhý neaktivní panel je volen jako cílový.

5.3 Implementace

Základ klientské aplikace je napsán ve značkovacím jazyce XHTML a nachází se v souboru *index.html*. V něm je definováno rozložení celé aplikace. Vzhled aplikace je určen v souboru *main.css* ve složce *css*. Ta obsahuje ještě další pomocné soubory s definicí vzhledu některých pluginů. Chování je definováno v souboru scripts.js, který se nachází ve složce *scripts*. Tak je docíleno oddělení všech tří základních částí aplikace – obsahu, vzhledu a chování. Vzhled je k obsahu přiřazen pomocí CSS tříd a identifikátorů, chování je určeno pomocí jQuery selektorů. Funkce jsou navázány jak na XHTML elementy (obrázky, tlačítka) tak na události (pohyb posuvné lišty). Pro interakci s uživatelem, např. při výpisu chybové hlášky nebo dotazu na název nové složky, jsou použity nastylované XHTML dialogy místo klasických javascriptových dialogů.

5.3.1 Operace se soubory

Operace se soubory se dělí na dva typy – operace s označenými soubory a operace s jedním souborem. Operace s jedním souborem jsou definovány ve funkci *setItemEvent()*, která se provádí nad všemi načtenými položkami v seznamu souborů. V ní je definováno označení a odznačení souboru při kliknutí a také je v ní navázána událost zobrazení překryvného okna (tzv. *lightBox*) s náhledem obrázku, pokud se jedná o obrázek. V této funkci by se tak měly nacházet všechny podobné události. Pro operaci s více označenými soubory slouží funkce *doStuff()*, která detekuje aktivní panel a pomocí funkce *getSelectedItems()* z něj získá seznam označených souborů. Poté

sestaví příslušnou URL a odešle požadavek na server. Dále pokud je to nutné znovu načte seznam souborů v jednom nebo obou panelech. Operace posledního zmiňovaného typu jsou prováděny po stisknutí tlačítek ve spodní liště.

Další možností práce se soubory je pomocí funkce drag&drop. Ta je implementována pomocí jQuery UI pluginů. Konkrétně se jedná o části *draggable* a *droppable*. Ty zajišťují podporu operací drag a drop. Metoda draggable je přiřazena ke všem položkám seznamu souborů. Je nastavena tak, aby přesouvání bylo omezeno pouze na část okna s panely. Položka se po vyvolání akce přesouvání klonuje, což zajistí, že je možné s ní táhnout i mimo oblast seznamu souborů i přesto, že má nastaveno skrývání přesahujících prvků. Toho je docíleno volbou *appendTo*, která způsobí vložení naklonovaného prvku mimo seznam souborů. Dále je nastavena prodleva mezi klinutím a posouváním, která zamezuje problémům rozlišení kliknutí a posouvání. Metoda droppable je nastavena dvakrát. Poprvé na celý panel, což umožňuje zkopírování souboru do aktuálně otevřeného adresáře. Podruhé na každou položku seznamu souborů. Pokud se jedná o adresář, kopíruje se přímo do něj. Pokud je položka soubor, kopíruje se do aktuálního adresáře.

Označení a přesun většího počtu souborů je v tomto případě nemožné implementovat. Je to z důvodu složitosti aplikace a špatné kompatibility pluginů selectable a droppable. Proto je možné posouvat pouze jednu položku. Operace s více soubory tak musí být prováděna pomocí funkčních tlačítek na spodní liště.

5.3.2 Komunikace se serverem

Načítání dat a zasílání požadavků je řešeno pomocí funkce *makeRequest()*, která jako parametr přebírá URL s parametry požadavku a funkci pro zpracování příchozích dat. Nakonec je k URL připojen parametr *jsoncallback*, který umožňuje posílat požadavky i na jiné domény, než je aktuálně běžící skript. Tato funkce je volána jednak z funkce *getItems()*, kde zajišťuje načtení seznamu souborů, a také z funkcí pro operace se soubory, kde pouze zajišťuje zaslání požadavku a navrácení chybového kódu.

5.3.3 Načítání seznamu souborů

Stahování seznamu souborů je řešeno poněkud netradičně. Při inicializaci panelu nebo změně adresáře je nejprve zjištěna výška viditelné části seznamu souborů (výška divu třídy *list*) a z ní je vypočítaný počet souborů, které je nutné načíst, aby byl celý pohled zaplněn. Z těchto údajů je pak sestaven výsledný dotaz a je odeslán požadavek na server. Po přijetí odpovědi na požadavek je naplněn celý *div* seznamem souborů prázdnými položkami (jejich počet je jedním z parametrů odpovědi). Tím je dosaženo toho, že se v případě potřeby zobrazí posuvná lišta, se kterou je možné

posouvat pohled na další soubory. Aktuální pohled je poté naplněn údaji o jednotlivých souborech. Pokud uživatel nepohne posuvnou lištou nebo se jinak neposune v pohledu dolů, neprovádí se žádná akce a zůstává načtený jen aktuální viditelný pohled. Zbytek je vyplněn prázdnými položkami. Tím se zmenšuje objem dat přenesených ze serveru a zvyšuje rychlost načítání prvních položek ve složce.

Pokud uživatel popojede s posuvnou lištou směrem dolů, je vyvolána událost *scroll()*, na tu je navázána opět funkce *getItems()*, která však nezačne plnit soubory ze začátku, ale vypočítá pozici první zobrazené položky pohledu z aktuální pozice posuvné lišty (metoda *scrollTop()*) a pošle požadavek na načtení dalších souborů. Po navrácení odpovědi jsou podle identifikátoru v atributu *rel* doplněny požadované soubory.

Výpis souborů může být omezován filtrem, který se zadává do textového pole nad seznamem souborů. Filtrování se aktivuje stisknutím klávesy *Enter*. Tím se nastaví příznak filtrování na *true* a veškeré výpisy jsou od té doby filtrovány až do kliknutí na červenou ikonu vedle textového pole. Shoda filtru se hledá v celém názvu souboru. Tím pádem je možné filtrovat například pouze soubory určitého typu podle jejich koncovky.

5.3.4 JS knihovny a pluginy

K základnímu jQuery je zde použito několik pluginů, které ulehčují některé operace a přidávají několik efektů, čímž zlepšují ovládání a vzhled aplikace. Je použita také externí, na jQuery nezávislá šablonovací knihovna.

Seznam pluginů:

jQuery impromptu: http://trentrichardson.com/Impromptu/index.php

Plugin, který vylepšuje javascriptové funkce alert, prompt a confirm. Umožňuje stylovat jejich dialogy pomocí CSS nebo například používat formuláře uvnitř těchto funkcí. Tento plugin je použit u přejmenování souborů.

jQuery cookie: http://plugins.jquery.com/project/cookie

Plugin umožňující pohodlné zapsání, čtení a smazání souboru cookie, do kterých se ukládá nastavení zobrazení panelu a jazyka.

jQuery corners: http://www.atblabs.com/jquery.corners.html

Vytváření kulatých rohů u XHTML elementů. Funguje se všemi moderními prohlížeči. Pro jeho funkci nejsou potřeba žádné dodatečné obrázky, pouze přidání třídy *rounded* k danému elementu.

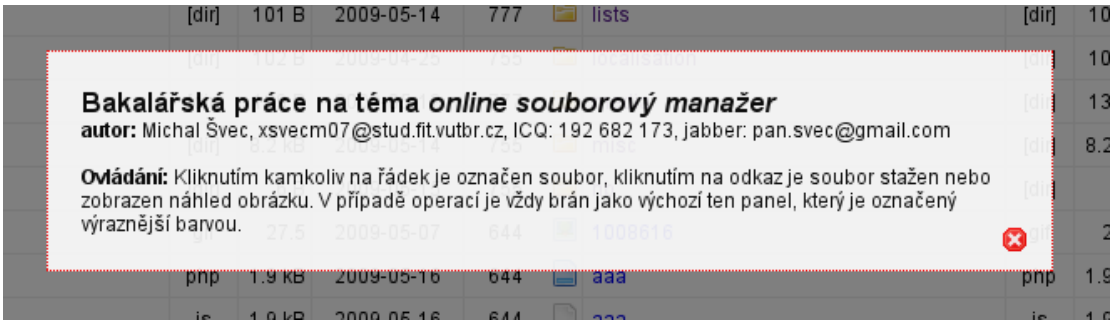
jQuery lightBox: http://jquery.com/plugins/project/jquerylightbox_bal

Velmi známý plugin pro překrytí celé stránky náhledem obrázku, který je použitý u náhledů obrázků.

jQuery localize: http://github.com/coderifous/jquery-localize/tree/master

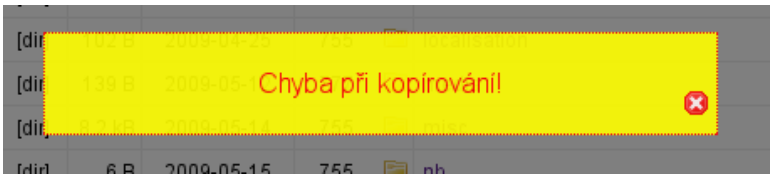
Jednoduchý lokalizační plugin. Načítá soubory s lokalizačními JSON řetězci z adresáře lang. Pomocí atributu *rel* mění obsah elementů se stejným atributem *rel* podle zadaného jazyka.

Seznam souborů je implementován pomocí javascriptových šablon. Použitý je templatovací systém TrimPath. Nejedná se pouze o šablonovací systém, ale o javascriptový framework. Pro potřeby tohoto souborového manažeru je ale použita pouze jeho šablonovací část. Je napsána kompletně v Javascriptu, podporuje všechny moderní prohlížeče v jejich nejnovějších verzích a má syntaxi podobnou nejpoužívanějším šablonovacím systémům jako je např. Smarty. Šablony jsou definované ve skrytých textových polích na konci XHTML dokumentu. Bylo nutné definovat dvě šablony. Jednu pro prázdný řádek výpisu a druhou pro obsah řádku naplněného informacemi o souboru. To je z důvodu plnění celého seznamu pouze prázdnými prvky a doplňováním informací až v druhé fázi. V šablonách je definováno několik podmínek pro zobrazení, které rozlišují adresář od souboru, nebo vypisují „*Prázdná složka*“ v případě, že složka neobsahuje žádné soubory. Zpracování šablony provádí metoda *processDOMTemplate()*, které je v parametrech předán identifikátor textového pole se zápisem šablony a objekt s daty, která budou do šablony doplněna. Výstupem je kompletní XHTML kód položky seznamu.



Obrázek 4: Ukázka informativní hlášky

Pro zobrazování chybových hlášek je použita funkce *showMsgBox()*. Ta pracuje s XHTML elementem třídy *msg-box* a *msg-box-bg*. *Msg-box-bg* slouží k překrytí celé stránky poloprůhledným objektem, který tak zvýrazní chybovou hlášku. Pro výpočet rozměrů vzniklého objektu je použita funkce *getPageSize()*, převzatá ze serveru *quirksmode.com*. Rozlišují se dva typy informačních boxů. Jeden pro chybové hlášky a druhý pro informativní hlášky jako je například nápověda. Styl výpisu se odlišuje druhým parametrem funkce (*type*). V případě, že je *type* nastaven na *error*, použije se hláška s chybovým hlášením, která je definovaná v prvním parametru. Pokud se má vypsat obsah nějakého XHTML elementu, nastaví se *type* na *div* a v třetím parametru se přidá identifikátor vypisovaného elementu (atribut *id*). Pokud není použita ani jedna možnost, vypíše se informativní hláška s obsahem zadaným prvním parametrem funkce.



Obrázek 5: Ukázka chybové hlášky

6 Srovnání s dostupnými alternativami

6.1 Dostupné alternativy

V dnešní době je k dispozici několik kvalitních a moderních souborových manažerů. Všechny ale mají jednu společnou vlastnost. Jsou to serverové aplikace, které používají Javascript pouze pro zlepšení práce s uživatelským rozhraním. Správce souboru s modelem klient-server však není dostupný ani jeden. Téměř všechny nejlepší aplikace obsahují základní sadu operaci kopírování, přesun, mazání a přejmenování. Pokročilejší správci souborů nabízejí také náhled obrázků nebo editaci textových souborů. Méně častá je možnost stáhnout složku či soubory jako archiv. Složité manažery mají navíc funkce jako záložky na oblíbené, nebo často používané složky a správu uživatelů a práv.

6.2 Výhody a nevýhody

Souborový manažer, který je předmětem této zprávy, se řadí svou koncepcí a funkcemi mezi průměrné produkty. Některé jeho vlastnosti ho ale ve specifických případech staví nad ostatní. Jeho hlavními výhodami jsou práce s velkým počtem souborů, filtrování souborů a jednoduchost. Nevýhodou je absence pokročilých funkcí, mezi které patří úprava textových souborů nebo ovládání klávesami.

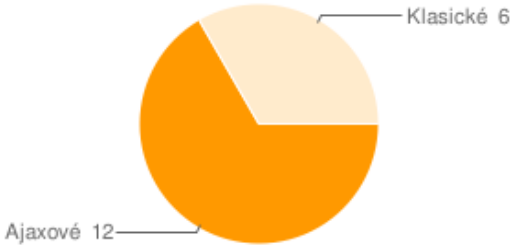
Práce s velkým počtem souborů umožňuje použití např. v elektronických obchodech, kde se vyskytuje velký počet souborů s obrázky produktů, nebo na serveru, kde je velký počet logovacích souborů. Filtrování poskytuje možnost pro rychlejší práci se soubory. Toho se dá velmi dobře také využít u logovacích souborů, pokud je potřeba zobrazit záznamy pouze od jedné aplikace, nebo z jednoho dne. Soubor však musí v názvu obsahovat datum vytvoření. Jednoduchost je využitelná hlavně v případě integrace do hotových řešení. Například do administrace redakčního systému nebo e-shopu pro výběr obrázků a souborů ke stažení.

V případě nevýhod je velkým problémem zachování jednoduchosti a snadné začlenitelnosti do hotových řešení. Podpora klávesových zkratek je záležitostí uživatelského rozhraní a může být implementována v dalších verzích, avšak rozšíření jako autentizace uživatelů, zabezpečení přenosu, nebo nastavení přístupových práv je věc vyžadující složité zásahy do celé aplikace, které vyžadují použití složitějších programovacích technik. To je však v rámci zachování jednoduchosti nemožné.

7 Závěr

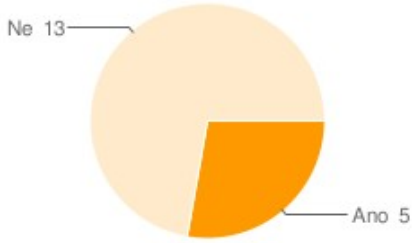
7.1 Uživatelské hodnocení

Při vývoji bylo využito i názoru uživatelů spolu s průzkumem využití a oblíbenosti AJAXových aplikací. Dvě třetiny dotazovaných uživatelů dávají přednost aplikacím využívajícím AJAX.

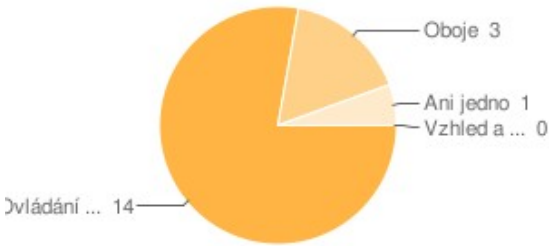


Obrázek 6: Máte raději AJAXové nebo klasické aplikace?

Bohužel však většina uživatelů nepoužívá žádný online správce souborů. Uživatelé, kteří odpověděli kladně používají např. správce souborů v IS FIT, TinyMCE File Explorer nebo správce souborů v emailovém klientu Seznam.cz



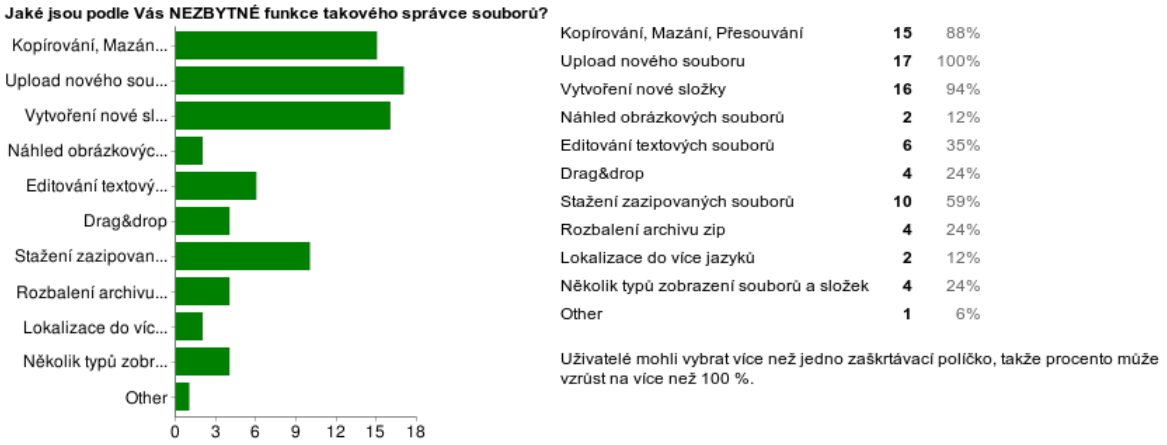
Obrázek 7: Používáte nějaký webový správce souborů?



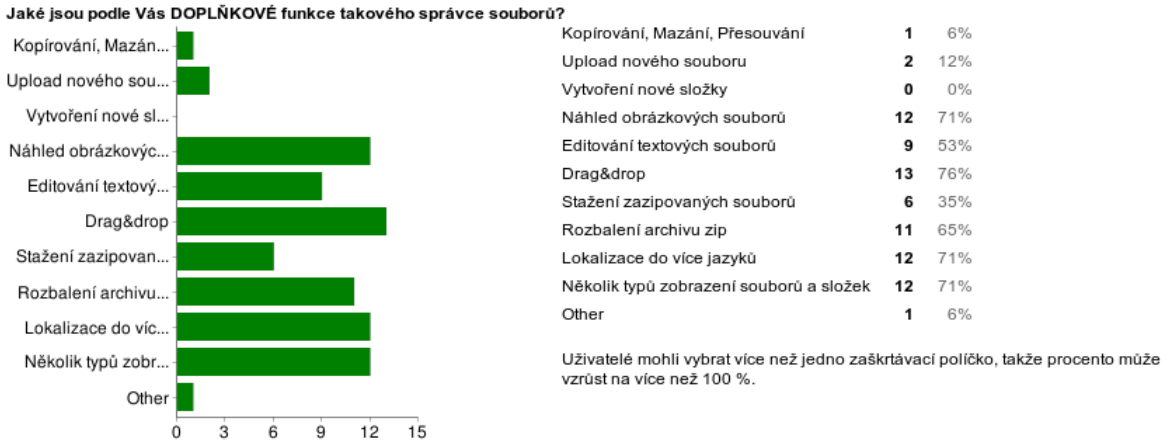
Obrázek 8: Je pro Vás důležité snadné ovládání a přehlednost, či příjemný vzhled a efekty?

Pro drtivou většinu uživatelů je také důležité pohodlné a intuitivní ovládání a přehlednost spíše než vzhled. V tomto ohledu je implementované řešení přesně podle potřeb uživatelů.

Další otázky byly zaměřeny více konkrétně na téma online file manager. První se týkala požadovaných funkcí, které by neměly žádnému file manageru chybět. Další oblibě doplňkových funkcí.



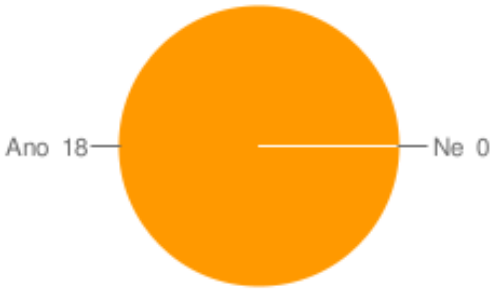
Obrázek 9: Jaké jsou podle Vás NEZBYTNÉ funkce takového správce souborů?



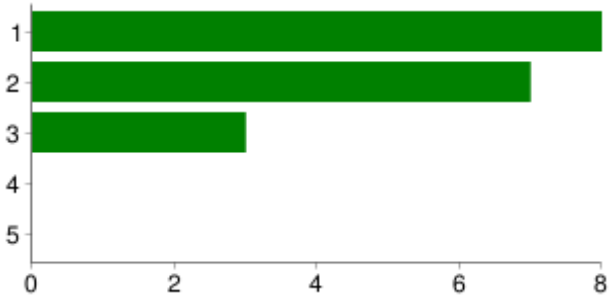
Obrázek 10: Jaké jsou podle Vás DOPLŇKOVÉ funkce takového správce souborů?

Z výše uvedených odpovědí vyplývá, že navržený a implementovaný správce souborů splňuje většinu požadovaných funkcí. Z nejvíce požadovaných doplňkových funkcí jich obsahuje zhruba polovinu. Chybí mu podpora rozbalení archivů a editace textových souborů. Plusem je podpora náhledu u obrázků, stažení zazipovaných souborů, lokalizace a přepínání režimů zobrazení. Současná implementace tak splňuje požadavky běžných uživatelů a může být nasazen pro reálné použití. Nevýhodou je absence některých funkcí.

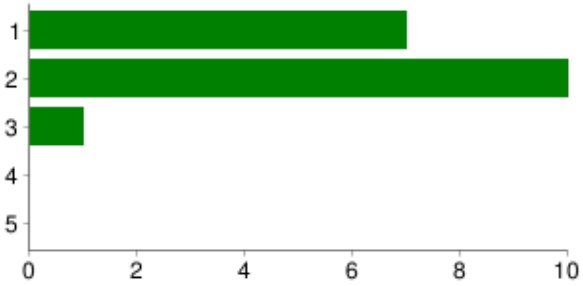
Posledním typem otázek byly otázky zaměřené přímo na konkrétní aplikaci.



Obrázek 11: Je pro vás ovládání aplikace intuitivní?



Obrázek 12: Jak hodnotíte práci se soubory?



Obrázek 13: Jak hodnotíte ovládání aplikace?

Ovládání a práce se soubory byly známkovány jako ve škole, tzn. 1 nejlepší, 5 nejhorší. Aplikace byla hodnocena víceméně kladně. Pro 100% uživatelů má intuitivní ovládání. Tím byl splněn požadavek ze zadání.

7.2 Možná rozšíření

Díky velmi dobré modifikovatelnosti obou aplikací je možné bez zásahu do současného stavu přidat nové funkce. Jelikož aplikace velmi připomíná desktopovou aplikaci, objevilo se mezi požadavky ovládání pomocí klávesnice. K dalším možným vylepšením uživatelského rozhraní se řadí například zobrazení kontextového menu s dostupnými operacemi se souborem.

Co se týče základních funkcí, chybí implementovanému správci souborů upload nových souborů, rozbalení zip archivů a změna přístupových práv k souboru. Doplňkovým rozšířením by mohla být autentizace uživatelů. To by však vyžadovalo výrazné zesložitění jak serverové, tak klientské aplikace. Nejjednodušším řešením by bylo použít http autentizaci, která by však vyžadovala začlenění klientské aplikace např. do PHP skriptu, který by se staral o zajištění autentizace. K serverové aplikaci by bylo potřeba přidat správu uživatelů a jejich databázi. U implementovaného řešení se počítá s tím, že veškeré tyto operace zajistí nadřazený systém (např. redakční systém). Správa uživatelů by také musela řešit přístupová práva jednotlivých uživatelů k souborům a složkám. Stejný problém je se zabezpečením přenosu. Aby vůbec bylo možné přenos šifrovat, musela by klientská aplikace načítat data přes protokol HTTPS. To vyžaduje úpravu webového serveru na kterém serverová aplikace běží.

7.3 Výsledné zhodnocení

Cílem této práce bylo implementovat správce souborů dostupný z Internetu, který by umožňoval začlenění do hotového řešení. To se povedlo a výsledkem je plně funkční souborový manažer podporující všechny nezbytné operace. Výhodou jsou některé další operace, které nepatří mezi obvyklé. Další velkou výhodou je implementace API, což dovoluje změnu klienta za jinou aplikaci. Stejně tak je možné zachovat klientskou aplikaci a serverovou vyměnit za jinou. To poskytuje výslednému programu vysokou flexibilitu. Problémem je různá podpora v prohlížečích. Aplikace byla testována v prohlížeči Firefox 3, ve kterém je bez problémů funkční. Dále funguje bez větších problému v prohlížečích Opera a Safari. Dalším problémem je také výkon Javascriptu a omezení některých funkcí, jako výběr více souborů a operace drag&drop s nimi, které nelze v Javascriptu v tomto případě implementovat. Velkým problémem je také špatná funkčnost na některých serverech webhostingových společnosti, která způsobuje špatné AJAXové načítání položek seznamu.

Literatura

[1] Wikipedie: Otevřená encyklopedie: PHP [online]. c2009 [citováno 16.05.2009]. Dostupný z WWW: <http://cs.wikipedia.org/w/index.php?title=PHP&oldid=3968438>

[2] Wikipedie: Otevřená encyklopedie: JavaScript [online]. c2009 [citováno 16. 05. 2009]. Dostupný z WWW: <http://cs.wikipedia.org/w/index.php?title=JavaScript&oldid=3806344>

[3] Wikipedia: The Free Encyclopedia. JavaScript [online]. c2009 [citováno 16. 05. 2009]. Dostupný z WWW: <http://en.wikipedia.org/w/index.php?title=JavaScript&oldid=290021174>

[4] Wikipedie: Otevřená encyklopedie: Asynchronous JavaScript and XML [online]. c2009 [citováno 16. 05. 2009]. Dostupný z WWW: <http://cs.wikipedia.org/w/index.php?title=Asynchronous_JavaScript_and_XML&oldid=3852735>

[5] Wikipedie: Otevřená encyklopedie: JavaScript Object Notation [online]. c2009 [citováno 16. 05. 2009]. Dostupný z WWW: <http://cs.wikipedia.org/w/index.php?title=JavaScript_Object_Notation&oldid=3911887>

[6] JSON [online]. c2009 [citováno 16. 05. 2009]. Dostupný z WWW: <http://www.json.org/json-cz.html>

[7] 39INAM. An Introduction to AJAX [online]. 2009 [cit. 2009-05-16]. Dostupný z WWW: <http://www.topcoder.com/tc?module=Static&d1=features&d2=071706>.

[8] Wikipedie: Otevřená encyklopedie: Rozhraní pro programování aplikací [online]. c2009 [citováno 16. 05. 2009]. Dostupný z WWW: <http://cs.wikipedia.org/w/index.php?title=Rozhran%C3%AD_pro_programov%C3%A1n%C3%AD_aplikac%C3%AD&oldid=3963639>

[9] Wikipedie: Otevřená encyklopedie: jQuery [online]. c2009 [citováno 16. 05. 2009]. Dostupný z WWW: <http://en.wikipedia.org/wiki/JQuery>

[10] Wikipedia: the free encyclopedia: Comparison of JavaScript frameworks [online]. c2009 [citováno 16. 05. 2009]. Dostupný z WWW: <http://en.wikipedia.org/w/index.php?title=Comparison_of_JavaScript_frameworks&oldid=290335821>

[11] VRÁNA, Jakub. HTTP metody GET a POST [online]. 2005 [cit. 2009-05-16]. Dostupný z WWW: <http://php.vrana.cz/http-metody-get-a-post.php>.

Seznam použitých zkratek

AJAX - Asynchronous JavaScript and XML je technologie pro změnu webových stránek bez jejich znovunačítání

FTP - File Transfer Protocol slouží k přenos souborů mezi počítači

PDA - Personal Digital Assistant je malý kapesní počítač.

JSON – Z anglického JavaScript Object Notation. Datový formát pro přenos dat.

XML - Extensible Markup Language je strulturovaný značkovací jazyk.

XHTML - Extensible HyperText Markup Language je strukturovaný jazyk sloužící pro tvorbu hypertextových stránek

CSS - Cascading Style Sheets označuje jazyk pro popis způsobu zobrazení stránek.

API - Application Programming Interface je rozhraní pro programování aplikací. Označuje sadu tříd, procedur, funkcí.

PHP – Rekurzivní zkratka pro PHP: Hypertext Preprocessor označuje skriptovací jazyk určený především pro tvorbu internetovýh stránek

URL - Uniform Resource Locator je jednotný identifikátor nějakého zdroje na Internetu.

Seznam příloh

Příloha 1. Návod k instalaci
Příloha 2. CD se zdrojovými kódy a elektronickou verzí této práce.

Příloha 1: Instalace

Instalace serverové aplikace

1. Pro zprovoznění aplikace je nutné nahrát serverovou aplikaci do složky webserveru.
2. Zvolit spravovaný adresář a ten nastavit do položky *defaultDir* v souboru *server.ini*. Je možné použít jak absolutní, tak relativní cestu vzhledem k souboru *index.php*.
3. Podle potřeby je možné nastavit konfigurační direktivy *dirFirst* a *listCount*.
4. Rekurzivně změnit práva spravovaného adresáře na 777.

Instalace klientské aplikace:

1. V souboru *client/scripts.js* je nutné nastavit proměnnou *serverURL* na adresu serveru. Např.: *var serverURL = "*http://localhost/OFM/server/*";*

Tím je vše vyřešeno