Masarykova univerzita
Fakulta informatiky

Návrh a implementace virtuálního souborového systému pro systém Celebrio

Bakalářská práce

Jan Volmut

Brno, jaro 2011

Prohlášení

Prohlašuji, že tato bakalářská práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Vedoucí práce: Mgr. Michal Kolínek

Poděkování

Rád bych touto cestou poděkoval vedoucímu své bakalářské práce, panu magistru Kolínkovi za vedení práce a za poskytnutí cenných rad při jejím psaní. Také bych rád vyslovil dík všem svým kolegům z týmu Celebrio, především těm, kteří se svou částí podíleli na vývoji virtuálního souborového systému. Jmenovitě Pavlu Smolkovi, který se přímo podílel na vývoji a Petru Kuncovi a Martinu Novákovi, kteří položili základy při návrhu virtuálního souborového systému a byli vždy ochotni přispět svou radou. V neposlední řadě bych také rád vyjádřil svůj dík panu docentu Pitnerovi, který projekt Celebrio zaštiťuje svou osobou na Fakultě informatiky.

Shrnutí

Práce se zabývá návrhem a implementací virtuálního souborového systému. Jedná se především o uchovávání informací o souborech a jejich fyzického ukládání na specifická úložiště. Práce obsahuje požadavky, kterými jsem se při implementaci řídil, dále obsahuje informace o návrhu i samotné implementaci.

Klíčová slova

Virtual File System, Celebrio Software, Windows Azure,

Obsah

Kapitola 1

Úvod

Téměř všechny aplikace potřebují nějakým způsobem pracovat s uloženými daty. Pro většinu z nich postačuje klasický souborový systém, který jim dává možnost ukládat a načítat data z lokálních i vzdálených disků. Zároveň tento souborový systém ctí práva přístupu k datům, s nimiž pracuje. Systém Celebrio ovšem potřebuje zpracovávat data dle vlastní politiky práv a na základě vlastních detailů.

Celebrio Software[1] je začínající studentská firma, která se v současné době zaměřuje na vývoj aplikace, která bude fungovat jako webový operační systém ve vzdáleném prostředí. Na jádru tohoto systému je momentálně postaven webový operační systém pro seniory, který jim zpřístupňuje mnohé aplikace, dosažitelné a využitelné bez větších znalostí a vynaloženého úsilí. Jádro jako takové je složeno z několika částí, modulů, mezi které patří právě i virtuální souborový systém.

Celebrio je systém, na kterém pracují studenti především z Fakulty informatiky Masarykovy univerzity, ale také z Fakulty informačních technologií Vysokého učení technického v Brně a Univerzity Tomáše Bati ve Zlíně. Tito studenti tvoří tým, který se podílí na vývoji. Spolupracují v mnoha ohledech, a proto i na virtuálním souborovém systému se určitou měrou podílelo více lidí.

1. Web Celebrio Software: <http://www.celebriosoftware.com>

Kapitola 2

Požadavky

Pro vnitřní potřeby systému Celebrio je nutné vytvořit virtuální souborový systém, který zajišťuje správu souborů a operace nad nimi. Mezi takové operace patří ukládání, mazání, načítání a část dalších operací, které poskytují klasické souborové systémy. Virtuální souborový systém bude mezivrstva mezi systémem Celebrio a různými úložišti. Požadavkem je, aby Celebrio mohlo komunikovat s různými datovými úložišti a centrální virtuální souborový systém zajišťoval perzistenci dat vzhledem k těmto úložištím. Uživatel by potom měl mít možnost používat virtuální souborový systém tak, jako klasický (klasický ve smyslu klasické adresářové struktury) i přes to, že soubory jsou reálně na různých místech.

2.1 Databáze

Metadata o souborech budou uložena v databázi. Ta bude zajišťovat ucelenost souborového systému a bude obsahovat virtuální adresářovou strukturu. Virtuální adresářová struktura bude stejná jako na jiných klasických souborových systémech, tedy stromová s jedním kořenem, a jednotlivé úrovně mezi sebou budou odděleny jednotným odělovačem („/“). V databázi dále budou drženy ostatní údaje o souborech, jako je jejich velikost, mimetype, druh úložiště a podobně.

2.2 Úložiště

Systém Celebrio se snaží vyjít vstříc co největšímu počtu uživatelů. V dnešní době je trend ukládat data do vzdálených úložišť, a proto i Celebrio bude nabízet ukládání na vzdálené servery. V této fázi vývoje konkrétně na servery Windows Azure. Stejně tak je ale důležité umožnit uživateli ukládání na lokální úložiště. V této práci se budu zabývat především umožněním ukládání souborů z virtuálního souborového systému jednak na lokální úložiště a také na servery Windows Azure.

Kapitola 3

Souborové systémy

3.1 Souborový systém

Souborový systém je prostředek pro ukládání souborů na datová úložiště. Počítačový soubor je kolekce dat, která dohromady tento soubor tvoří. Datová úložiště využívají souborové systémy, aby tyto soubory mohly být uloženy a rozeznány od sebe. Soubory se často navíc ukládají tak, aby byly „human-readable“ tedy jednoduše čitelné pro člověka, který se souborovým systémem pracuje. To například znamená, že jsou soubory ukládané pod jmény, které člověk dokáže od sebe odlišit. Na základě jmen také může se soubory dále pracovat. Tato jména souborů jsou propojena v alokační tabulce s indexy či uzly označující jednotlivé soubory.

Aby uložení souborů bylo více lidsky čitelné, mohou být uspořádány hierarchicky a tvořit logickou strukturu ukládaných souborů. Osoba užívající souborový systém je tak schopna jednoduše členit a rozlišovat soubory nejen na základě jmen, ale také na základě jejich hierarchického rozložení. Je pak možno členit soubory do dalších logických celků. Hierarchická struktura ale není pravidlem. Existují i souborové systémy, které s hierarchií uložených souborů pracují jen omezeně, nebo nemusí pracovat vůbec.

Soubory v souborovém systému mívají doprovodné informace, které jsou pro každý daný soubor specifické. Jedná se typicky o délku souboru vyjádřenou například v bytech a čas poslední modifikace. Jsou různé další atributy, které si jednotlivé souborové systémy mohou ukládat, jako je čas založení souboru, idetifikační údaj o vlastníkovi souboru a skupině, kódování u textových souborů, příznak smazání, příznak spustitelnosti, či zařízení, kde je soubor uložen. Časové údaje (modifikace, založení) mohou být uloženy více způsoby včetně timestampu.[8]

3.1.1 Oprávnění k přístupu

Souborový systém může být využíván jen jedinou osbou, jako je tomu často u osobních počítačů, ke kterým přistupuje jediný uživatel, nebo může být být sdílen s několika uživateli najednou. Ve druhém případě musí být implementována opatření zabraňující uživateli přistupovat k souborům druhého uživatele bez jeho vědomí. Naopak také musí být umožněno uživateli, aby své soubory mohl poskytnout uživateli jinému. V souborovém systému tyto problémy řeší model oprávnění. Ten zajišťuje práva a privilegia uživatelů přistupovat k jednotlivým souborům. Mezi způsoby, které zajišťují tato oprávnění, se řadí také Access

Control List (ACL) a Role-Based Access Control (RBAS).

3.1.1.1 Access Control List

Access Control List je seznam záznamů, ve kterém jsou zanesena práva a oprávnění. Nejčastěji to bývá tabulka, která spojuje zdroje (soubory) s uživateli a oprávněními.

Př.

Uživatel Bob chce přistupovat k souboru "prava.txt", který vytvořila Alice. Proto mu Alice nastaví oprávnění na prohlížení daného souboru. Změna se zanese do seznamu oprávnění zhruba jako (Bob, prava.txt, r). Bob nyní může číst soubor, který vytvořila Alice, ale Alice mu to nejdříve musela povolit. Pokud by k soubru "prava.txt" chtěla přistupovat Eva, musela by jí to Alice také dovolit (nebo by Eva musela použít nějaký jiný způsob z jejího repertoáru, aby se k souboru dostala)

Jednotlivé záznamy v ACL se nazývají Access Control Entries (ACE) - položky řízení přístupu. Tyto položky vypadají podobně jako v příkladu výše a jsou v nich zaneseny právě informace o souboru, uživateli a právu přístupu. Základní práva jsou R pro čtení (read), W pro zápis (write) a X pro spuštění (execute). Nicméně různé systémy používají různý přístup k oprávnění a tak je možné najít i další, zjemňující prvky.[7]

V ACL mohou nastat dva základní případy - uživatel může mít právo přistupovat k sobouru nebo jej mít nemusí. Souborů, ke kterým uživatel nemá oprávnění přistupovat, bývá často několikrát více než těch, ke kterým přístup má. Jak je možné vidět, matice seznamu řízení přístupu může být velmi řídká, tedy mnoho záznamů by bylo prázdných. Z toho důvodu ukládáme jen existující oprávnění a nikoli celou matici. V zásadě pokud oprávnění existuje, přístup je povolen a pokud neexistuje, přístup je zamítnut.[6]

3.1.1.2 Role-Based Access Control

Role-Based Access Control (RBAC), neboli řízení přístupu založené na roli, počítá s přidělování rolí jednotlivým uživatelům. Klasické ACL používá k ukládání záznamů o oprávněních uživatele trojci uživatel-zdroj-privilegium. RBAC rozšiřuje tento způsob přístupu o role. Tabulka uložení práv k přístupu je uložena jako role-zdroj-privilegium.

Přiřazováním rolí k uživatelům můžeme docílit lepší správy řízení přístupu, než kdybychom nastavovali každému uživateli jeho vlastní práva. Takto nastavujeme práva přístupu k souboru celé skupině uživatelů a nemusíme řešit nastavení pro každého uživatele zvlášť. Některé systémy dovolují uživateli být ve více skupinách a tedy stačí jen uživateli přidat roli, aby mohl pracovat s dalšími dokumenty, které ovšem ostatním členům z předešlé role nejsou k dispozici. Zároveň mu samozřejmě zůstává role původní. Nyní tedy přistupuje k souborům obou rolí.

3.2 Virtuální souborový systém

Existuje mnoho druhů souborových systémů a často je potřeba přistupovat k nim najednou z jednoho systému. Tento přístup zajišťuje virtuální souborový systém, který vytváří abstraktní vrstvu nad různými klasickými souborovými systémy. Z pohledu člověka, který tuto vrstvu využívá, to vypadá, jakoby pracoval s jedním souborovým systém. Ve skutečnosti však pracuje s různými souborovými systémy, které virtuální souborový systém zaštiťuje. Implementace virtuálního souborového systému spočívá v ukládání informací o jednotlivých úložištích, ale především také v implementacích rozhraní k jednotlivým souborovým systémům. Existuje pak jednotná stuktura virtuálního souborového systému a podle uložených odkazů se virtuální souborový systém odkazuje na různá úložiště.

Virtuální souborový systém se využívá například pokud je potřeba překlenout rozdíly mezi souborovými systémy různých operačních systémů anebo při přístupu ke vzdálenému úložišti. Jaká datová úložiště bude umět virtuální souborový systém ovládat záleží na implementaci jeho rozhraní.

Kapitola 4

Použité technologie

Při práci na praktické části byly použity následující technologie, které byly potřebné pro zajištění cílů práce a kompatibility se stávajícím jádrem systému. Kromě níže vyjmenovaných technologií bylo použito také MySQL, jako databázová vrstva a PHP coby hlavní jazyk, ve kterém je Celebrio napsáno.

4.1 Nette Framework

Jádro systému Celebrio je napsáno v jazyce PHP, respektive jeho objektově orientované podobě. Jako podpůrný prostředek byl zvolen Framework Nette a databázová vrstva Dibi.

Nette je relativně mladý framework, který je nejvíce rozšířený v České a Slovenské republice. Jeho výhodou je silné zabezpečení proti různým útokům, které mohou poškodit či kompromitovat obsah uložený na webu. Mezi takové útoky patří například Cross-site scripting (XSS), který zneužívá neošetřený vstup. Útočník pak může podvrhnout část kódu svým vlastním maligním kódem. Další útoky, které Nette odstiňuje, jsou útoky spojené s vkládáním špatných vstupů do zadávacích polí (URL attack, control codes, invalid UTF-8) a také útoky spojené se session (Session hijacking, session stealing, session fixation).[5]

Dalším důvodem pro užití Nette Frameworku je jeho podpora MVC modelu, principu KISS a dalších moderních technologií.[4]

Nette Framework obsahuje některé funkcionality, které vývoj usnadňovaly. Především je to statická třída `Nette\Environment`, pomocí které lze přistupovat k některým vlastnostem prostředí, ve kterém Celebrio funguje. Například lze takto zjistit, který uživatel (pokud je přihlášený) zaslal požadavek, který se právě zpracovává. Díky tomu lze jednodušeji implementovat řízení přístupu. Ke zjištění přihlášeného uživatele slouží metoda `getUser()`, která vrací instanci třídy `User`. Ta v sobě nese informace o uživateli včetně autorizačního handleru (Authorization handler), pomocí kterého se dá pracovat s oprávněními uživatele.[1]

Další technologií využité ve virtuálním souborovém systému, která vychází z Nette, je systém řízení přístupu. Nette obsahuje třídu `Permission`, která zajišťuje přístup k oprávněním a je jednoduše rozšiřitelná o další funkcionalitu. Třída `Permission` vlastně agreguje jednolivá oprávnění a pokud je nutné zjistit, jestli má uživatel právo přistupovat k určité části systému, směřuje systém tyto požadavky právě na tuto třídu.

1. Nette\Web\User: <http://doc.nette.org/cs/nette-web-user>

4.2 Dibi

Dibi je abstraktní databázová vrstva pro PHP verze 5 a je použita i ve frameworku Nette verze 0.9. Celebrio využívá dibi pro komunikaci systému s databází. V kódu aplikace jsou volány metody abstraktní třídy `dibi`, které překládají pseudo SQL jazyk zapsaný programátorem na dotazy pro jednotlivé databáze. Díky tomu je systém schopen fungovat na více druzích databází, aniž by existovalo několik verzí systému pro jednotlivé implementace databázové vrstvy.

Dibi obsahuje i některé konstrukce, díky nimž je skládání dotazů do databáze pohodlnější

```
$arr = array(
    'pole' => 'hodnota',
    'bit'  => TRUE,
);
dibi::query('INSERT INTO [table]', $arr);

//výsledkem bude dotaz odpovídající následující podobě
INSERT INTO table (pole, bit) VALUES ('hodnota', TRUE);
```

Kromě používání polí jako argumentů obsahuje dibi tzv. dibiFluent. Jedná se o řetězení příkazů, které ve výsledku vedou k vykonání stejného dotazu, ale jejich použití se více blíží objektovému návrhu.

```
dibi::select("*")->from("tabulka")->where("parametr = %s", "hodnota");
//Výsledkem bude
SELECT * FROM tabulka WHERE parameter="hodnota";
```

4.3 Windows Azure Storage

Windows Azure Storage je služba, kterou poskytuje platforma Windows Azure. Jedná se o úložiště velkých binárních objektů, které je umístěno na serverech Windows Azure. Pro přístup k datům je využíváno rozhraní REST.

Struktura ukládání souborů je u Windows Azure Storage relativně plochá, jako jediný technologický prostředek, jak členit soubory, je jejich ukládání do speciální struktury zvané kontejner. Kontejnery si lze představit jako složky, které se nacházejí přímo pod kořenovým adresářem. Nicméně tyto kontejnery se nemohou do sebe zanořovat. Existuje tedy jakýsi kořenový adresář (resp. kontejner) a pokud chceme data nějak logicky dělit, vytváříme pro ně další kontejnery. Každý soubor navíc musí být umístěn v nějakém kontejneru. V distribuovaném prostředí Windows Azure je problém v tom, že data mohou být rozmístěna na různých místech. Ukládání do společných kontejnerů by mělo zajistit, aby data spolu související byla na jednom místě a tak poskytovala možnost rychlejšího přístupu, než kdyby byla na různých úložištích.

Technologie Windows Azure Storage nicméně také podporuje zanořování pomocí oddělovače „/“. Je tedy možné ukládat soubory jako „cesta/adresar/soubor“ a později se dotázat na všechny soubory, které obsahují ve svém jménu „cesta/adresar/“. Je nutné upozornit, že toto členění se promítá jen ve jménech souborů, nevytváří adresářovou strukturu jako takovou, jak by se na první pohled mohlo zdát.

Soubor se v terminologii Azure Storage označuje jako „blob“, který může být dvojího typu.

`Block blob` Blob je složen z bloků dat, které mají velikost 4 MB. Tyto části souboru se na server odesílají sekvenčně a po odeslání je zavolána funckce „commit“, která potvrdí uložení na serveru za předpokladu, že byl přenos úspěšný. Block blob může mít velikost až 200 MB.

`Page blob` Používá se pro objemnější soubory a jeho velikost může být až 1 TB.

Kromě již zmíněného REST rozhraní poskytuje Windows Azure také vývojové balíčky (SDK - Software Development Kit) pro některé rozšířené jazyky jako je C#/.NET, Java a PHP.[2]

Kapitola 5

Návrh

Navrhování datové struktury, struktury tříd a celkové podoby virtuálního souborového systému pro systém Celebrio sestával z různých pohledů na jeho celkové uchycení a použití. Při návrhu byly řešeny otázky týkající se počtu atributů u držených souborů, oprávnění přístupu k souborům, ukládání na různá datová úložiště a také kontroly přístupu a logování.

5.1 Datový návrh

Datový model virtuálního souborového systému čítá v současné době deset tabulek, přičemž využívá další dvě systémové tabulky (`users` pro ukládání vlastníků a `roles`, kde jsou uloženy role přiřazené uživatelům). V této práci jsou tabulky datového návrhu rozčleněny do několika logických skupin tak, aby se daly strukturovaně popsat jejich funkce. Tyto logické celky by se daly pojmenovat jako „Centrální část“, „Část pro aplikace“ a „Správa přístupu k souborům“. Celkový diagram datového modelu je umístěn v příloze A.

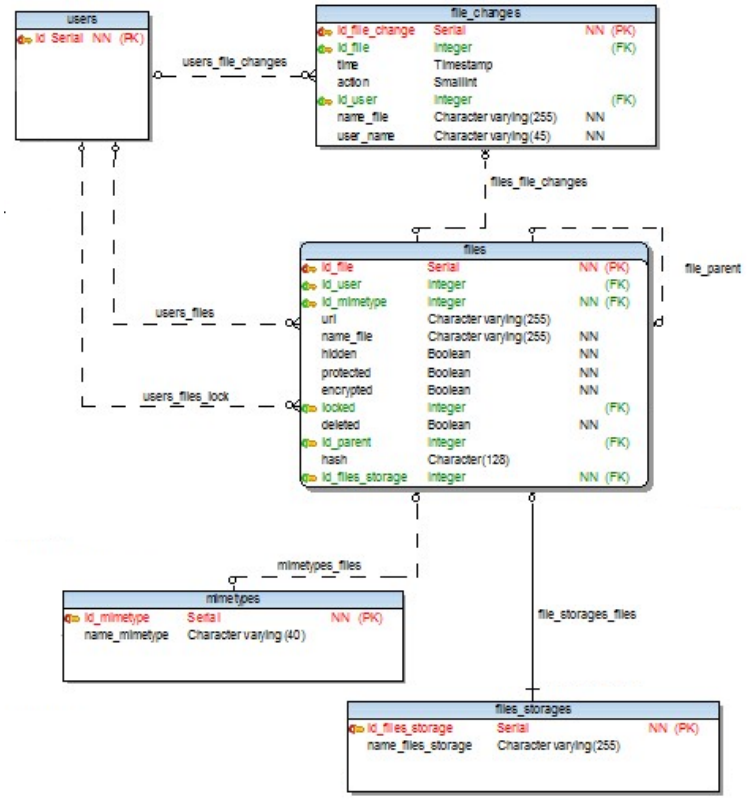
Z důvodu snahy Celebria být modulárním systémem nejsou některé tabulky zahrnuty v uváděných diagramech. Jsou to především tabulky, které sice souvisí s virtuálním souborovým systémem, ale při jeho zavádění nemusí existovat a dokonce nemusí být ani později použity. Týká se to především částí systému pracujících se souborovými úložišti, které se aktivují až při zavedení funkcionality obsluhující tyto moduly.

Centrálním bodem při návrhu datové struktury virtuálního souborového systému byla tabulka `files` (obrázek 5.1), kde jsou uloženy informace o souborech. Obsahuje cestu a jméno souboru, jeho mimetype, odkaz na vlastníka souboru, atributy souboru a také odkaz na úložiště, kde je soubor fyzicky uložen. Identifikátor uživatele je brán z tabulky `users`, která je standardní částí systému a spojuje soubor s jeho vlastníkem.

Pro logování přístupu k souborům slouží tabulka `file_changes`. Obsahuje přístupy uživatelů k danému souboru se specifickou akcí. Zaznamenávají se vždy jen poslední přístupy, tedy kdy a kým byl soubor naposledy upravován.

ATRIBUTY TABULKY FILES

`url`, `name_file` Virtuální cesta a virtuální jméno souboru. Na základě těchto atributů lze dohledat soubor. Kombinace jména a cesty musí být unikátní.



Obrázek 5.1: Datový model - centrální část virtuálního souborového systému

`id_file, id_user, id_mimetype, id_files_storages` Identifikátor souboru a reference na tabulky uživatelů, typu souboru a úložiště.

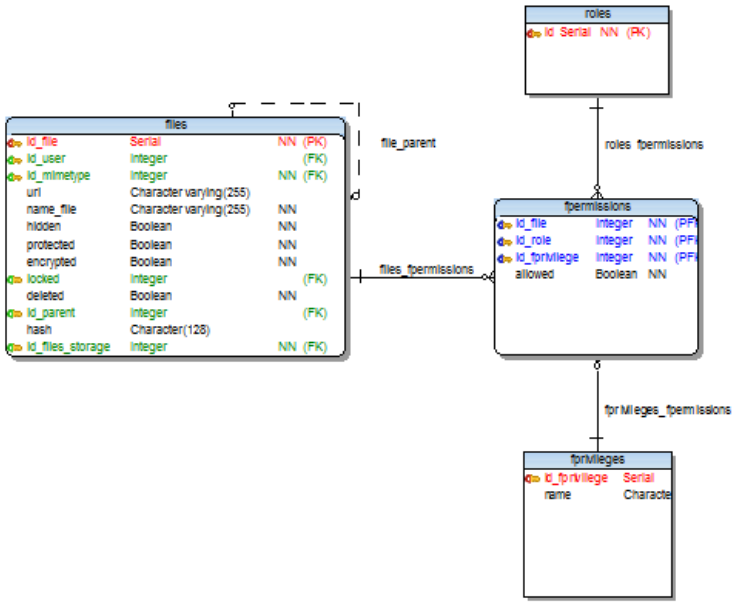
`id_parent` Odkaz na rodiče souboru. Pomáhá udržovat stromovou strukturu souborového systému. Díky tomu se souborové systémy různých úložišť mohou prolínat a například složka na lokálním úložišti může obsahovat soubory na úložišti vzdáleném.

`hidden, protected, encrypted, locked` Atributy souboru, které obsahují údaje, zda je soubor skrytý, chráněný, šifrovaný nebo zamčený.

Druhá část diagramu souvisí s oprávněními a přístupem k souborům a je znázorněna na obrázku 5.2. Na tabulku `files` se váže tabulka `fpermissions`, kde jsou uloženy záznamy o oprávněních uživatelů přistupovat k souborům. Práva jsou ukládána jako ve schématu ACL podporující RBAC zmíněné v kapitole 3.1.1.

Role jsou přejímány z tabulky `roles`, která je částí systému Celebrio a jsou v ní uloženy jednotlivé role. Ve virtuálním souborovém systému se tak nepřidělují práva jednotlivým uživatelům, ale rolím, které jsou těmto uživatelům přiřazeny. Samotná práva jsou uložena v tabulce `fprivileges`. Pro virtuální souborový systém byla zavadena klasická práva pro čtení, zápis a spuštění a navíc i práva číst a zapisovat atributy, číst oprávnění a také práva pro převod vlastníka souboru.

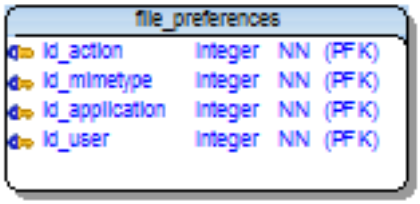
Jednotlivá oprávnění jsou jednoznačně identifikována trojicí role-soubor-právo. Každá položka v tabulce oprávnění má navíc příznak **allowed**, který slouží k povolení, případně k zakázání oprávnění přístupu pro danou roli. Pro správce je pak jednoduší odebírat a přidávat oprávnění, protože po jejich odejmutí zůstane záznam uložen a v případě potřeby se oprávnění znovu povolí a nemusí se vytvářet znovu.



Obrázek 5.2: Datový model - přístup k souborům

Poslední část datového modelu zachycuje propojení virtuálního souborového systému s tabulkami aplikací. Systém Celebrio bude využívat své vlastní aplikace i aplikace třetích stran. Ačkoli tato část systému není ještě zcela hotova, již nyní je inicializována společně s virtuálním souborovým systémem, jelikož s ním nepřímo souvisí (většina aplikací nějakým způsobem využívá souborový systém). Středem této části datového modelu je tabulka `file_preferences` vyobrazena na obrázku 5.3. Je to agregační tabulka, která spojuje tabulku uživatelů, aplikací povolených Celebriem, typů souborů a akcí. Tyto čtyři atributy tabulky tvoří její primární klíč a vytváří záznam o tom, jaká akce se má provést s aplikací podle uživatelských nastavení. Například pro prohlížení obrázku (akce read) může sloužit aplikace pro prezentaci obrázků, ale pro jeho editaci může být nastavena jiná aplikace, která otevře obrázek pro editaci. Jednotlivá nastavení se váží na mimetype souboru. Tak je zaru-

čeno, že soubory se stejným typem (např. image/jpeg) se budou pro jednu akci otevírat ve stejné aplikaci.



Obrázek 5.3: Datový model - tabulka file_preferences

5.2 Návrh funkcionality

Virtuální souborový systém Celebria sestává z několika částí, jak již bylo naznačeno v kapitole 2. Daly by se rozdělit na zpracování požadavku, správu databázových záznamů a operace nad souborovými úložištěmi. Virtuální souborový systém poskytuje veřejné rozhraní - API (Application programming Interface). Součástí tohoto API jsou operace nad soubory, které se mají vykonat. Třída implementující toto rozhraní zaštiťuje celý virtuální souborový systém, tedy funkcionalita databáze a obsluha úložiť by měla být před ostatními skryta, respektive každý požadavek, který přijde z vnějšku, musí projít přes tuto třídu. Tato politika je zavedena zejména kvůli kontrole práv a také kvůli záznamu přístupů. Při zpracování požadavku dochází k ověřování práv a k volání tříd, které souvisí s úpravou záznamů o souborech v databázi. Poté se volají služby pro ukládání na jednotlivá datová úložiště.

Nedílnou součástí virtuálního souborového systému jsou i jednotlivá úložiště, kam se soubory ukládají. Je potřeba především lokální úložiště, kde jsou soubory standardně uloženy, pokud je systém Celebrio nainstalován na klasické firemní servery nebo i na vzdálený server Celebria. Dále je zapotřebí rozhraní pro úložiště Windows Azure Storage, jelikož se nasazení Celebria plánuje i na platformu Windows Azure.[1]Je potřeba, aby existovalo jednotné rozhraní, které respektují všechny implementace úložišť. Musí poskytovat standardní operace, které jsou nezávislé na vnitřním chování virtuálního souborového systému. Návrh rozhraní počítá s funkcionalitou poskytující základní operace nad souborovými systémy. Především se jedná o uložení, načtení, smazání, přesunutí, přejmenování a překopírování souboru.

1. Windows Azure storage: <http://www.microsoft.com/windowsazure/storage/>

Kapitola 6

Implementace

6.1 Centrální funkcionalita

Jak již bylo řečeno výše (kapitola 5.2), je v implementaci nutné zahrnout jednu třídu, která bude zaštiťovat zbytek virtuálního souborového systému. Tímto pomyslným štítem je v implementaci třída `VirtualFileSystem`. Tuto třídu je možné volat uvnitř systému podle potřeby, ale z vnějšího prostředí systému k tomu bude docházet pomocí REST rozhraní, pomocí které se budou přijímat požadavky. Jednotlivé části virtuálního souborového systému spolu komunikují na základě předávání referencí na objekty speciálně pro to vytvořené. Jedná se o třídy `DataFile` a `VirtualFile`, společně s jejich předkem, od kterého dědí společnou funkcionalitu `GenericFile`.

`DataFile` Entita sloužící pro předávání standardních informací o souboru. Obsahuje cestu a jméno k souboru, úložiště, na kterém je soubor uložen, jeho typ a data souboru. Také obsahuje atribut `fileInfo`, ve kterém jsou uloženy další atributy souboru. S touto entitou pracuje virtuální souborový systém při komunikaci s okolním prostředím včetně tříd obsluhující datová úložiště.

`VirtualFile` Entita, která reprezentuje uložení dat v databázi. Virtuální souborový systém ji využívá pro interní potřeby komunikace s databází a při sestavování výsledné podoby entity `DataFile`, která bude vrácena uživateli.

Cesta od zadání požadavku uživatelem až po zobrazení výstupu se skládá z několika procedur. Z HTTP požadavku je sestaven `DataFile`, který musí obsahovat název souboru a cestu k němu (ve virtuální podobě). Volitelnými parametry je mimetype souboru a jeho úložiště. Inicializace třídy `DataFile` pak může vypadat takto:

```
$dataFile = new DataFile(
                "jmeno",
                "/dloua/cesta",
                "typ/souboru",
                "druh_uloziste"
                 );
```

Poté dochází k inicializaci třídy `VirtualFileSystem`, na které se zavolá metoda s názvem požadavku, který přišel společně s daty souboru (například akce save - uložit nebo load - načíst). Je nutné uvést, že třída `VirtualFileSystem` obsahuje konstruktor vyžadující třídu `User`, která není součástí samotného Celebria, ale frameworku Nette (viz kapitola 4.1). Třída `User` obsahuje informace o uživateli, který je právě přihlášen.

Veřejné metody ve třídě `VirtualFileSystem` obsluhují celou práci nad virtuálním souborovým systémem. Dalo by se říci, že je to správce této části systému. Nejdříve dojde k převedení entity `DataFile` na entitu `VirtualFile`. Následně jsou volány třídy související s databázovou vrstvou (viz 6.2), kde se ověří práva a provedou se potřebné úpravy v databázi. Pokud vše skončí správně, vytvoří se instance třídy obsluhující datové úložiště pomocí tvz. tovární třídy (Factory class). Té se předá reference na původní instanci třídy `DataFile`, která byla předána do virtuálního souborového systému a provedou se tak změny v daném úložišti.

Příklad načtení textového souboru text.txt, uloženém na lokálním úložišti ve virtuální složce /home/volmut/:

Požadavek na soubor se převede na entitu `DataFile`, kterému se nastaví cesta „/home/volmut“ a jméno „text.txt“. Ten se společně se současným uživatelem předá virtuálnímu souborovému systému (třídě `VirtualFileSystem`). Dojde k vytvoření entity `VirtualFile`, založené na entitě `DataFile`. Virtuální soubor se předá třídě `VfsEngine` (viz 6.2), kde se zkontroluje oprávnění pro přístup k souboru „/home/volmut/text.txt“, a doplní se mu potřebná data (především na jakém úložišti se soubor nachází, tedy „local“). Na základě těchto informací načte virtuální souborový systém třídu `Local`, která pracuje s lokálními soubory. Odešle mu `DataFile`, na jehož základě se dohledá soubor na lokálním disku a nastaví entitě `DataFile` jeho obsah. Virtuální souborový systém poté vrátí již komplentí `DataFile` rozhraní, které odešle data pro zpracování uživateli, či aplikaci.[3]

Příklad 6.1.1: Příklad práce virtuálního souborového systému

Následující část kódu uvádí příklad, jak je možné komunikovat s virtuálním souborovým systémem navazující na předchozí příklad:

```
//inicializace virtuálního souborového systému a předání mu současného
//uživatele
$vfs = new VirtualFileSystem(Environment::getUser());
//vytvoření datového souboru
$dataFile = new DataFile("text.txt", "/home/volmut");
//nastaveni obsahu souboru
$dataFile->setData("Pokusny text");

$vfs->save($dataFile); //uložení souboru

//načtení souboru /home/volmut/text.txt, load vrací opět DataFile
$result = $vfs->load($dataFile);
```

```
//odeslání obsahu textového souboru do prohlížeče,
//kde se zobrazí text "Pokusny text"
echo $result->getData();
```

Přestože je v příkladu použit textový soubor, funguje stejný systém i s binárními soubory. Pokud je soubor pro uložení například obrázek, bude atribut `data` entity `DataFile` obsahovat reprezentaci binárních dat souboru. Ilustrační schéma pohybu dat od zadání požadavku až po uložení souboru je vyobrazen v příloze B.

Pro aktivaci virtuálního souborového systému je zapotřebí některých dalších tříd systému Celebrio a také některých tabulek v databázi. Také potřebuje některé součásti frameworku Nette a databázové vrstvy dibi. Z toho důvodu jsou třídy, které reprezentují modul virtuálního souborového systému, samostatně nefunkční, ačkoli v systému Celebrio fungují a jsou aktivně využívány.

6.2 Databázová vrstva

Systém Celebrio si klade za cíl být velmi interoperabilním systémem a proto podporuje několik databází. Jsou to databáze MySQL, PostgreSQL, MSSQL a Windows Azure SQL. Pro všechny tyto datábaze existují skripty, které vytvoří tabulky v databázi a Celebrio je schopno na těchto databázích fungovat. Virtuální souborový systém má také databázové skripty pro zmíněné databáze, ovšem nejsou součástí skriptů systému, z důvodu modularity Celebria. Pro lepší provázanost systému s databázemi využívá Celebrio databázovou vrstvu dibi, která unifikuje způsob dotazování do databáze.[1]

Virtuální souborový systém využívá databázi pro zpřístupnění dat o souborech (z angl. metadata) uložených v systému. Databázovou vrstvu zastupuje třída `VfsEngine`, ve které je obsažena funkcionalita, propojující virtuální souborový systém s databází. Provádění dotazů nad databází předchází samotné manipulaci se soubory na datových úložištích. Z tohoto důvodu a také z důvodu, že některé akce nemusejí k datům na úložištích vůbec přistupovat, je třída `VfsEngine` místo, na kterém dochází ke kontrole přístupových práv.

Funkcionalita řízení přístupu je v samostatné třídě `PermissionChecker`. Pro její inicializaci je potřeba předat konstruktoru třídu specifikující aktuálního uživatele (viz 4.1, třída `User`). Zde je nutné poznamenat, že třída `User` v systému obsahuje autorizační handler, který obsahuje oprávnění pro přístupy k jednotlivým sekcím v systému jako takovém, ale neobsahuje informace o právech k souborům. Proto ve třídě `PermissionChecker` dochází ke zkopírování předané třídy `User` (pomocí metody `clone`)[2] a k výměně autorizačního handleru za nový, který specifikuje oprávnění přístupu k souborům. Toto opatření je nezbytné, aby přístup k oprávněním uživatele ohledně virtuálního souborového systému nenarušilo jeho oprávnění k přístupu do celkového systému. Nový handler je vytvořen pomocí již zmíněných prostředků frameworku Nette (viz 4.1) a za pomoci tříd `FileAcl` a `FileAclModel`, které záznamy o oprávněních vybírají z databáze. Přestože funkcionalita

1. Dibi: <http://dibiphp.com/>
2. Metoda clone: <http://php.net/manual/en/language.oop5.cloning.php>

ověřování oprávnění pro přístup k souborům je funkční, není zatím implementována v systému z důvodu probíhajícího vývoje a testování. Implementací se zde rozumí volání třídy `PermissionChecker` v momentech zásahu do databázové struktury virtuálního souborového systému.

Třída `VfsEngine` zajišťuje správnou manipulaci s daty uloženými v databázi. Především řeší změny, které se projevují i na úložištích, jako je vytváření nebo mazání. Virtuální cesta a jméno souboru se mění vzhledem k požadavkům uživatele stejně, jako by se měnila adresářová struktura v klasických souborových systémech. Některé akce ale nepotřebují, aby se se soubory jakkoli manipulovalo. Příkladem může být načítání obsahu adresáře, kdy přijme virtuální systém entitu `DataFile`, která odkazuje na adresář a požaduje na tomto adresáři akci load (načtení). V takovém případě se provede dotaz do databáze a díky návrhu struktury ukládání dat (každá položka v databázové tabulce `files` obsahuje i odkaz na svého rodiče) bude výsledkem dotazu kolekce potomků dotazovaného adresáře.

6.3 Rozhraní pro úložiště

Datová úložiště jsou místa, kde jsou uložena data souboru. Tato úložiště mohou mít svůj vlastní souborový systém, který je nutné propojit s funkcionalitou Celebria, zejména virtuálního souborového systému. Úložiště mohou být různých typů, ať již lokální nebo vzdálené. Příklad vzdáleného úložiště může být Windows Azure Storage nebo Google Docs. V současné době jsou implementována rozhraní pro dvě úložiště - lokální a vzdálené na serveru Windows Azure. Všechny třídy implementující funkcionalitu pro nějaké datové úložiště musí zároveň implementovat a splňovat rozhraní `IDriver`, které předepisuje základní funkcionalitu, která bude po těchto třídách vyžadována.

Rozhraní `IDriver`:

- load(DataFile $dataFile)
- save(DataFile $dataFile)
- move(DataFile $source, DataFile $destination)
- delete(DataFile $dataFile)
- exists(DataFIle $dataFile)
- onChangeOwner(DataFile $dataFile)

Jednotlivé implementace rozhraní pro datová úložiště již nepotřebují řešit oprávnění přístupu, jelikož je tato problematika řešena již o fázi dříve ve třídě `VfsEngine`. Teoreticky potom mohou být třídy pro datová úložiště používány samostatně, ovšem za předpokladu, že nebudou využívat virtuální souborový systém a budou sloužit jen pro ukládání dat.

6.3.1 Lokální úložiště

Jako ukládání na lokální úložiště je myšleno ukládání souborů na místě, kde je systém Celebrio spušťen. Při zavádění funkcionality pro lokální ukládání je nutné vyplnit cestu, která bude v systému virtuálním kořenovým adresářem. Zadaná cesta musí v souborovém systému přístroje, na kterém Celebrio funguje, existovat. Soubory uložené přes systém Celebrio se nyní budou ukládat do adresáře `celebriofs`, který se vytvoří pod adresářem zvoleným uživatelem.

Cesta v Celebriu	Cesta v lokálním souborovém systému
/home/volmut/text.txt	/cesta/adresar/CELEBRIOFS/home/volmut/text.txt

Tabulka 6.1: Ukázka virtuální cesty k souboru vzhledem k cestě na lokálním úložišti.

Jak je patrné z tabulky 6.3.1, virtuální cesta ve virtuálním souborovém systému od jisté chvíle kopíruje strukturu na datovém úložišti. Informace o cestě, která vede k virtuálnímu kořenu, je uložena v nastaveních, které se vytvářejí při zavádění lokálního úložiště.

Funkcionalita ukládání na lokální úložiště je obsažena ve třídě `Local`. Při aktivování lokálního ukládání v systému se uloží cesta k virtuálnímu kořenovému adresáři a při dalším volání se tato cesta načítá ze systémových nastavení.

Třída `Local` pracuje se dvěma přidruženými třídami, které jsou již v systému Celebrio obsaženy. Jedná se o třídu `File`, obsahující metody pro práci s lokálními soubory a třídu `Folder`, která zpracovává požadavky pro práci s adresáři. Obě třídy nějakým způsobem obalují klasické PHP metody pro práci se soubory a adresáři. Obě třídy tak poskytují rozhraní, díky kterému je možno s těmito funkcemi pracovat jako s metodami tříd.

```
//Vytvoření instance třídy file, zároveň se vytvoří
//spojení na soubor pro zápis
$file = new File("cesta_k_souboru", "w");
//zapsání do souboru
$file->write("Testovací data");
//Uzavření spojení na soubor
$file->close();
```

Kromě klasických metod pro zápis, čtení a mazání obsahuje třída `File` také metody pro zamykání souboru a pro pohyb ukazatele v datech.

Metody ve třídě `Local` pracují na základě dělení entit na adresáře a soubory. Ačkoli pro systém je výhodnější pracovat s entitou, která může zastupovat obě složky, při ukládání do lokálních adresářů je nutné tyto entity rozlišovat z důvodu potřeby jiného chování k adresáři a souboru. Například při přesouvání souboru je třeba rozlišovat, zda se uživatel nesnaží přesunout složku pod soubor. Takovýchto operací, které může uživatel provést, je více. Při vývoji byly tyto operace testovány na Linuxovém prostředí, aby se výsledná funkcionalita co nejvíce přibližovala klasickému souborovému systému. Příkladem může být pokus o přesunutí adresáře na místo souboru, což by vedlo k přepsání tohoto souboru adresářem (obr. 6.1), což není povoleno.

```
qwei@esentia:~/test$ echo "data v a" >> a
qwei@esentia:~/test$ echo "data v b" >> b
qwei@esentia:~/test$ ls
a  b
qwei@esentia:~/test$ mv a b
qwei@esentia:~/test$ ls
b
qwei@esentia:~/test$ cat b
data v a
qwei@esentia:~/test$
qwei@esentia:~/test$ mkdir directory
qwei@esentia:~/test$ ls
b  directory
qwei@esentia:~/test$ mv directory b
mv: cannot overwrite non-directory `b' with directory `directory'
qwei@esentia:~/test$
qwei@esentia:~/test$ mv directory b/
mv: accessing `b/': Not a directory
qwei@esentia:~/test$
```

Obrázek 6.1: Přesunutí souboru na soubor, adresáře na soubor a adresáře pod soubor

Metody pro operace se soubory fungují velmi podobně, jak je tomu v klasickém souborovém systému.

Metoda `load` slouží pro načítání obsahu. Při předání složky jako paramentru vrátí výjimku. Nicméně k takovému stavu by ve virtuálním souborovém systému nikdy nemělo dojít, díky tomu, že funkcionalita pro načtení obsahu složky je implementována v databázové vrstvě (třída `VfsEngine`) a tento požadavek do ovladače pro lokální úložiště nepošle. V případě souboru funguje metoda `load` tak, že jej otevře a vrátí jeho obsah. Respektive vrátí entitu `DataFile`, které nastaví atribut `data`.

Pro vytváření a ukládání na lokální úložiště slouží metoda `save`. V případě adresáře i souboru jej vytvoří, pokud neexistují a v případě že jsou již vytvořeny, uloží do souboru data (pokud nějaká přišla).

Metoda `delete` sloužící pro mazání funguje pro soubory i složky téměř stejně, jen s tím rozdílem, že složky jsou smazány rekurzivně. To znamená s veškerým obsahem. Teoreticky je tato funkcionalita nevyužitá, jelikož třída `VfsEngine` odmazává záznamy z adresářů postupně odspoda. Nicméně je tato funkcionalita připravena k budoucímu využití.

Pro přesun souborů slouží metoda `move`. Pomocí ní dochází k přesunům souborů mezi adresáři, ale slouží také k přejmenování. Jak již bylo nastíněno, tato metoda se musí vyvarovat stavů, které jsou klasickým souborovým systémem vnímány jako chybové (viz obr. 6.1).

Metoda `move` slouží také pro přejmenování souborů, stejně jako příkaz **`mv`** v unixovém prostředí. Externí rozhraní tedy může nabízet funkci pro přejmenování, ale v pozadí je tento proces vykonán metodou `move`.

Metoda `exists` slouží pro dotazy, zda soubor (nebo adresář) na lokálním úložišti existuje či ne. Poslední metoda `onChangeOwner` slouží obecně pro akci vyvolanou tehdy, pokud se změní vlastník souboru. U lokálního úložiště zatím není potřeba implementovat žádnou podobnou funkcionalitu. Metoda `on change owner` vznikla teprve nedávno kvůli potřebě reagovat na změnu vlastníka při ukládání na Windows Azure Storage. Přibyla také

do rozhraní, jelikož v budoucnu bude možná potřeba reagovat na změnu vlastníka ve všech implementacích rozhraní pro daná úložiště.

6.3.2 Windows Azure Storage

Windows Azure storage je vzdálené datové úložiště. Jak již bylo napsáno v kapitole 4.3, existuje vývojový balíček pro některé programovací jazyky, včetně skriptovacího jazyka PHP.[3]Tento balíček obsahuje třídy pro připojování k Windows Azure Storage, Windows Azure Queue (implementace fronty zpráv na serverech Windows Azure) a Windows Azure Tables (ukládání strukturovaných dat).

Třída implementující rozhaní pro Windows Azure Storage se jmenuje `AzureCelebio`. Připojení k serveru Windows Azure Storage, kde jsou uloženy soubory, zajišťuje třída z vývojového balíčku `Microsoft_WindowsAzure_Storage_Blob`, která přijímá jako parametry adresu serveru, název účtu a heslo pro zadaný účet. Pomocí těchto údajů se systém autentizuje při zasílání požadavků do úložiště.

```
$storage = new Microsoft_WindowsAzure_Storage_Blob(
              "blob.core.windows.net",
              "jmeno_uctu",
              "heslo"
               );
```

I přesto, že dotazy jsou naimplementovány pomocí PHP, uvnitř tříd pro Microsoft Windows Azure se dotazy vykonávají pomocí rozhraní REST. PHP pak tvoří jen jakýsi most mezi programátorem a HTTP požadavky.

Jak již bylo zmíněno, adresářová struktura Windows Azure Storage je poměrně strohá (viz kapitola 4.3) a není moc prostředků, jak ji implementovat podobně jako v případě lokálního úložiště. Navíc pro tento typ úložiště je požadavek takový, aby každý přihlášený uživatel měl vlastní kontejner, ve kterém budou uchována jeho data. Toto opatření je v Celebriu proto, aby byl možný přehled o tom, kolik místa na vzdáleném serveru uživatel zabírá svými daty. Zároveň toto opatření slouží ke hlídání kvót jednotlivých uživatelů, protože ukládání dat na serverech Windows Azure jsou zpoplatněna.

Z výše jmenovaných důvodů byla implementována funkcionalita, která virtuální jména souborů a jejich virtuální cestu převádí na hash, který slouží jako jméno pro soubory ukládané na úložišti Azure Storage. K ukládání hashovaných jmen slouží tabulka `azure_files`, která je vytvořena při zavádění funkcionality ukládání na úložiště Azure Storage a váže se na tabulku `files`. Dalším důvodem, proč implementovat tento druh ukládání byla absence možnosti přesouvat a přejmenovávat data na úložišti Azure Storage. Při přejmenování či přesunu souborů by navíc docházelo ke zbytečným datovým přenosům, které jsou také zpoplatněny. Proto se tyto změny odehrávají pouze v databázové vrstvě a mění se jen virtuální cesty a jména. Díky tomu, že jsou jména souborů na vzdáleném serveru spojena

3. PHP SDK pro Windows Azure <http://phpazure.codeplex.com/>

s identifikátorem daného souboru v tabulce `files`, může se virtuální cesta volně měnit a soubory v úložišti zůstanou netknuty.

Metody implementované z rozhraní fungují podobně jako u lokálního datového úložiště. Metody `load` a `save` načítají a ukládají data. Tentokrát se jedná pouze o soubory, jelikož ukládání adresářů na Azure Storage působí problémy (Azure Storage uloženou složku označí jako prázdný soubor). Navíc třída `AzureCelebrio` obsahuje metodu pro ukládání velkých souborů `saveLargeFile`. Pomocí této metody se ukládají soubory větší než 64 Mb z důvodu různých postupů při ukládání malých a velkých souborů na Azure Storage (viz kapitola 4.3).

Metoda `delete` funguje pro mazání souborů z úložiště, metoda `exists` funguje stejně jako v případě lokálního úložiště (kapitola 6.3.1). Přesun souborů a jejich přejmenování je zajišťováno pomocí databáze, proto metoda `move` neobsahuje žádnou implementaci.

Zároveň se třídou `AzureCelebrio` byla implementována třída `AzureStorage`, která bude tvořit prostředek pro ukládání souborů na úložiště Windows Azure Storage bez využití virtuálního souborového systému Celebria. Je připravena tak, aby nevyužívala systémového nastavení a nezasahovala tak do dat ukládaných pomocí virtuálního soubrového systému.

Kapitola 7

Závěr

7.1 Využití

Virtuální souborový systém je jedna ze stěžejních částí systému. Bude využívána pro veškerý přístup k souborům uložených přes systém Celebrio. V současné době je dokončováno REST rozhraní pro přístup zvnějšku systému. Zároveň bude virtuální souborový systém sloužit pro načítání a ukládání dat aplikacím navázaných na systém Celebrio. Již nyní je aktivně využíván pro zpřístupnění dat aplikaci Galerie, která je součástí systému Celebrio pro seniory, který vyhrál národní kolo soutěže Imagine Cup 2011 pořádanou firmou Microsoft.

7.2 Další vývoj

Ačkoli je virtuální souborový systém nyní použitelný, objevují se stále nové výzvy, jak jej vylepšit. Jedna z těchto výzev je přepsání funkcionality přenosu dat. Ta se nyní po přenosu do systému ukládají do paměti serveru, kde jsou až do ukončení požadavku. Ukládání do paměti serveru je možné pro relativně malé soubory, které rychle vznikají a rychle zanikají, nicméně pro velké soubory je tato situace z dlouhodobějšího hlediska neúnosná. Následující vývoj se tedy bude zabývat vyřešením přenosu souborů jako proudu dat. Funkcionalita celkového virtuálního souborového systému se téměř nezmění, jen data nebudou součástí datových objektů (`DataFile` a `VirtualFile`), ale bude se s nimi pracovat jako s proudem dat odesílaným po ostatních procedurách.

Literatura

[1] DebianHelp: *ACL(Access Control List) Configuration in Debian*, 2009, <http://www.debianhelp.co.uk/acl.htm>.

[2] Microsoft: *About the Storage Services API - Blob Service*, 2011, <http://msdn.microsoft.com/en-us/library/dd573356.aspx>. 4.3

[3] Novák, M. a Kunc, P. a Smolka, P. a Volmut, J.: *Virtuální souborový systém*, 21. 4. 2011, <https://docs.google.com/a/albireo.eu/document/d/1xv4Puu2nVmP47tuZekZ5UxFPUUaAvqv2TK0g4T89G3Q/edit?hl=cs#> po autentizaci. 6.1

[4] Nette Foundation: *Hlavní přednosti*, 2011, <http://nette.org/cs/hlavni-prednosti>. 4.1

[5] Nette Foundation: *Dokonalé zabezpečení webových aplikací*, 2011, <http://nette.org/cs/hlavni-prednosti/bezpecnost>. 4.1

[6] Matyáš, V. a Říha, Z.: *PV157 - Autentizace a Řízení přístupu: Řízení přístupu*, 13. 4. 2011, <https://is.muni.cz/auth/el/1433/jaro2011/PV157/um/PV157_2011_L10_AC1_final.pdf?fakulta=1433;obdobi=5105;kod=PV157> po autentizaci, 9. 3.1.1.1

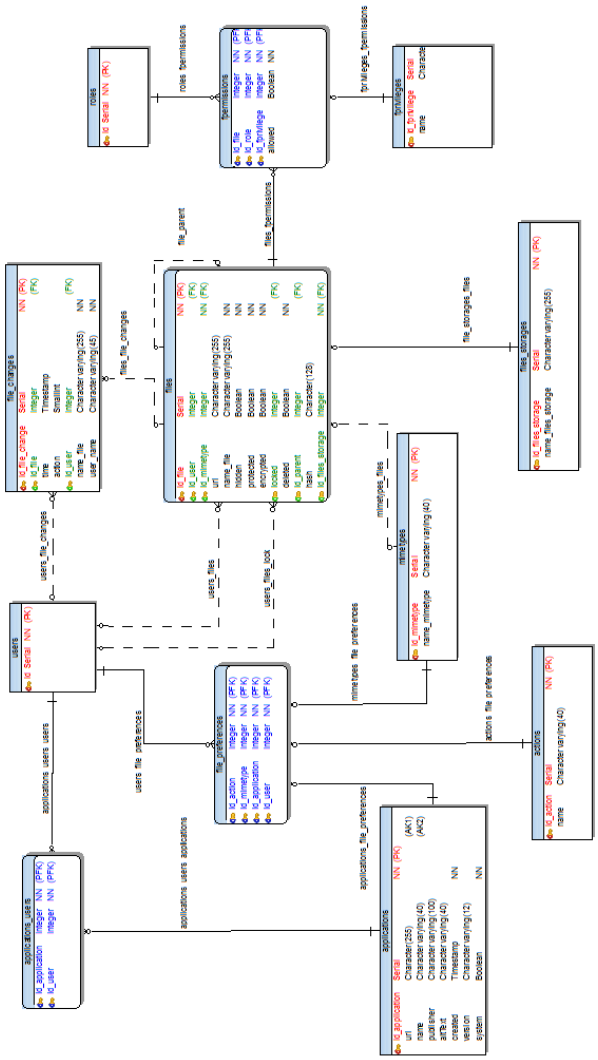
[7] Wikipedia Authors: *Access Control List*, 4. 4. 2011, <http://en.wikipedia.org/wiki/Access_control_list>. 3.1.1.1

[8] Wikipedia Authors: *File system*, 3. 5. 2011, <http://en.wikipedia.org/wiki/Filesystem>. 3.1

Příloha A

Datový model

Ukázka datového modelu použitého při vytváření virtuálního souborového systému

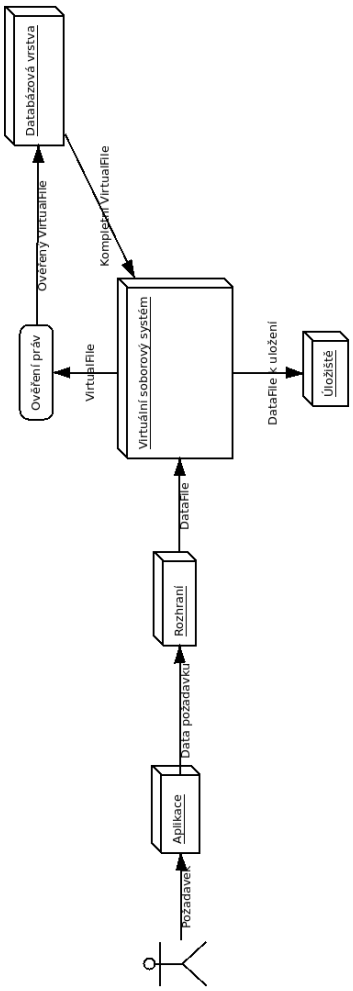


Obrázek A.1: Datový model

Příloha B

Schéma cyklu souboru

Obrázek znázorňuje schéma pohybu dat přes systém od odeslání požadavku uživatelem po uložení dat.



Obrázek B.1: Schéma cesty souboru přes systém