Na tomto místě bude oficiální zadání vaší práce

- Toto zadání je podepsané děkanem a vedoucím katedry, anebo taky ne no
-
- musíte si ho vyzvednout na studiijním oddělení Katedry počítačů na Karlově náměstí,
- v jedné odevzdané práci bude originál tohoto zadání (originál zůstává po obhajobě na katedře),
- ve druhé bude na stejném místě neověřená kopie tohoto dokumentu (tato se vám vrátí po obhajobě).

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Diplomová práce

Archivační systém pro cloudové prostředí

Bc. Martin Mudra

Vedoucí práce: Ing. Martin Klíma, Ph.D.

Studijní program: Otevřená informatika, Navazující magisterský

Obor: Softwarové inženýrství

10. května 2013

Poděkování

Děkuji svému vedoucímu práce Ing. Martinu Klímovi, Ph.D. za všechny jeho cenné rady a připomínky.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

Hostivici dne 5. 5. 2013 .....

Abstract

Presented work deals with issues of file backup and synchronization implemented in a cloud computing technology. This work also describes problems of file versioning and long term storage of backuped files including their versions. One of the main effort in long-term file storage is maintained data minimalization on the server side. Data processing on the server side is required to support asynchronnous run of the whole process. Main goal is creation of reliable system with universal architecture, which can be easily extended with new features.

Abstrakt

Tato práce se zabývá problematikou zálohování a synchronizace souborů implementované zejména pomocí technologie cloud computing. Práce je zaměřena na řešení problematiky verzování jednotlivých souborů a dlouhodobého uchování zálohovaných souborů, včetně příslušných verzí. Při dlouhodobém uchování verzovaných dat je hlavní snahou minimalizovat množství udržovaných dat na cloudové platformě. U samotného zpracování dat je také dbáno na asynchronní běh celého procesu na serverové straně. Hlavním požadavkem je vytvoření systému, který bude snadno rozšiřitelný o další funkce, a bude mít co nejvíce univerzální architekturu.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

1.1 Motivace

Zálohování souborů a jejich údržba je úkol, který řeší v současné době každý uživatel počítače. S pokročilými možnostmi a výkonem počítačů roste každoročně množství dat, které uživatel počítače vyprodukuje. Pouze jejich údržba se stává pro uživatele čím dál tím větší přítěží. Kupříkladu množství vyrobených dat na jednotýdenní dovolené může ve fotografiích a různých videosekvencí dosáhnout snadno několika desítek GB, což je kapacita, která byla před dvaceti lety nepředstavitelná. Takovýto přírůstek dat však také vyžaduje značné množství času od vlastníka dat na jejich organizaci, uložení, zálohování a případně verzování dat. Vlastník dat má velmi často také více zařízení (mobil, stolní počítač, notebook, tablet apod.) a uživatel velmi často potřebuje mít data k dispozici na více zařízeních zároveň. Současné běžně dostupné internetové připojení také vybízí k různému sdílení dat mezi uživateli, protože jsou uživatelé schopni přenést velký objem dat v rozumném čase. Přesto všechny tyto úkoly zůstávají uživatelsky časově náročné, a proto vhodným řešením může být použití systému pro automatizovanou správu dat.

Společnosti jako Amazon, Google nebo Microsoft a mnoho dalších nabízí pronájem výkonných počítačů, úložného prostoru a výpočetního výkonu. Těmto službám jsou souhrnně nazývány „cloud“. Pro jejich zákazníky to přinášní několik výhod, mezi které patří především:

- Bezúdržbovost celého systému
- Škálovatelnost (v případě aktuální potřeby vyššího výkonu již uživatel nemusí přikupovat další počítače, ale stačí, když si pronajme další výkon)
- Zálohování (o data uložená se stará poskytovatel úložného prostoru, který zároveň ručí nějakým licenčním ujednáním za jejich uchování a ochranu proti ztrátě)

Uvedené výhody nahrávají možnosti ukládat data uživatelů do některého z těchto cloudů pomocí nějakého automatizovaného systému.

1.2 Cíl práce

Hlavní cíl práce je vytvořit systém, který umožní jednotlivým uživatelům zálohování jejich souborů do prostředí cloudu Microsoft Azure, a který bude uložené soubory zpracovávat s cílem minimalizace objemu ukládaných dat.

1.3 Scénáře použití

Typický příklad pro použití takovéhoto systému jsou soubory typu fotografie a videa jednotlivých uživatelů. Velikost jedné fotografie narůstá s postupným zlepšováním cenově dostupných fotoaparátů, které jsou často schopny nahrávat i videa o velikém rozlišení. Jednotlivé fotografie si však uživatel často prohlédne několikrát po jejich pořízení a následně je potřebuje dlouhodobě uchovat, aby se k nim mohl kdykoliv vrátit. Použití nějakého vhodného systému pro ukládání dat je pro uživatele tedy velmi pohodlné a usnadní to práci, protože se o své data nebude muset již starat, ale stále k nim bude mít přístup ze všech zařízení, která mají přístup k internetu. Pro vlastníka zařízení by bylo také pohodlné, kdyby mohl své soubory verzovat, neboli mít přístup ke všem změnám u každého souboru. Uživateli by měla tak být zachována možnost kdykoliv se vrátit ke starší verzi jednotlivých souborů nebo celých složek.

1.3.1 Operace uživatelských účtů

1. Uživateli je umožněno registrovat se pomocí klienta
2. Uživateli je umoženo přihlásit se pomocí registrovaného účtu

1.3.2 Základní zálohování souborů

1. Uživatel se přihlásí k systému pomocí klienta.
2. Uživatel si zvolí složku, která bude nadále hlídána
3. Veškeré změny v této složce budou okamžitě nahrány do serverové strany systému běžící v Windows Azure.
4. Veškeré změny provedené se soubory z jiného klienta jsou okamžitě reflektovány i na soubory ve vybrané složce.

1.3.3 Webový přístup

1. Uživatel se přihlásí k systému prostřednictvím webové stránky
2. Uživatel si může procházet své soubory ve formě souborového systému
3. Uživatel si může u libovolného souboru prohlédnout jeho nahrané verze a stáhnout si libovolnou z nich do svého počítače.

4. Uživatel může se soubory provádět základní operace jako například:
   - Přejmenování
   - Smazání
   - Přesun
   - Kopírování

1.4 Návaznost na jiné práce

1.4.1 Původní stav

Podobný systém již byl implementován v rámci několika diplomových a semestrálních prací. Tomuto systému předcházel výzkum v Laboratoři interoperability Microsoft na ČVUT[1]. Tento systém však byl implementačně nadále neudržovaný. Prostředí Windows Azure od doby implementace tohoto systému také pozměnilo a původní systém tedy nemohl být použit. Tento systém však velmi dobře posloužil jako funkční prototyp a jako prozkoumání dostupných možností. Původní rozdělení komponent je zobrazeno na obrázku 1.1.

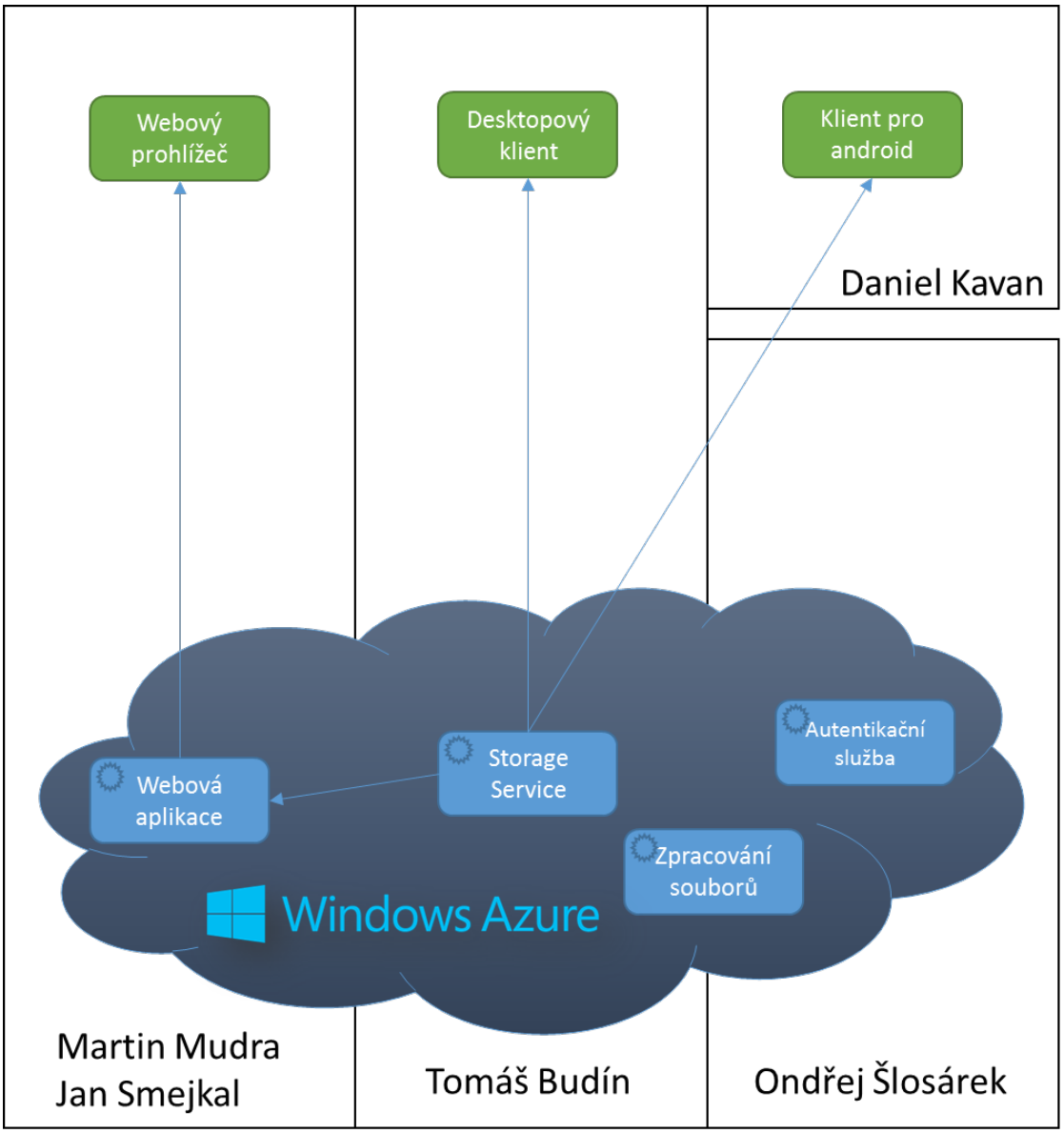
1. Bc. Tomáš Budín(*Diplomová práce*)
   - Storage Service (služba zajištující ukládání souborů)
   - Desktopový Klient
   - Část služby pro zpracování souborů (komunikace s prostředím)
2. Bc. Ondřej Šlosárek(*Diplomová práce*)
   - Zpracování souborů
   - Autentizační služba
   - Část služby pro zpracování souborů (verzování souborů)
3. Bc. Daniel Kavan(*Diplomová práce*)
   - Klient pro Android telefony
4. Bc. Martin Mudra a Bc. Jan Smejkal(*Semestrální práce z předmětu Vývoj webových aplikací 2* [2])
   - Webový klient pro prohlížeč

[1] Laboratoř interoperability Microsoft (HomePage)<http://czm.fel.cvut.cz/IOL/default.aspx>
[2] Vývoj webových aplikací 2(Stránka předmětu) <http://dcgi.felk.cvut.cz/cs/study/wa2>

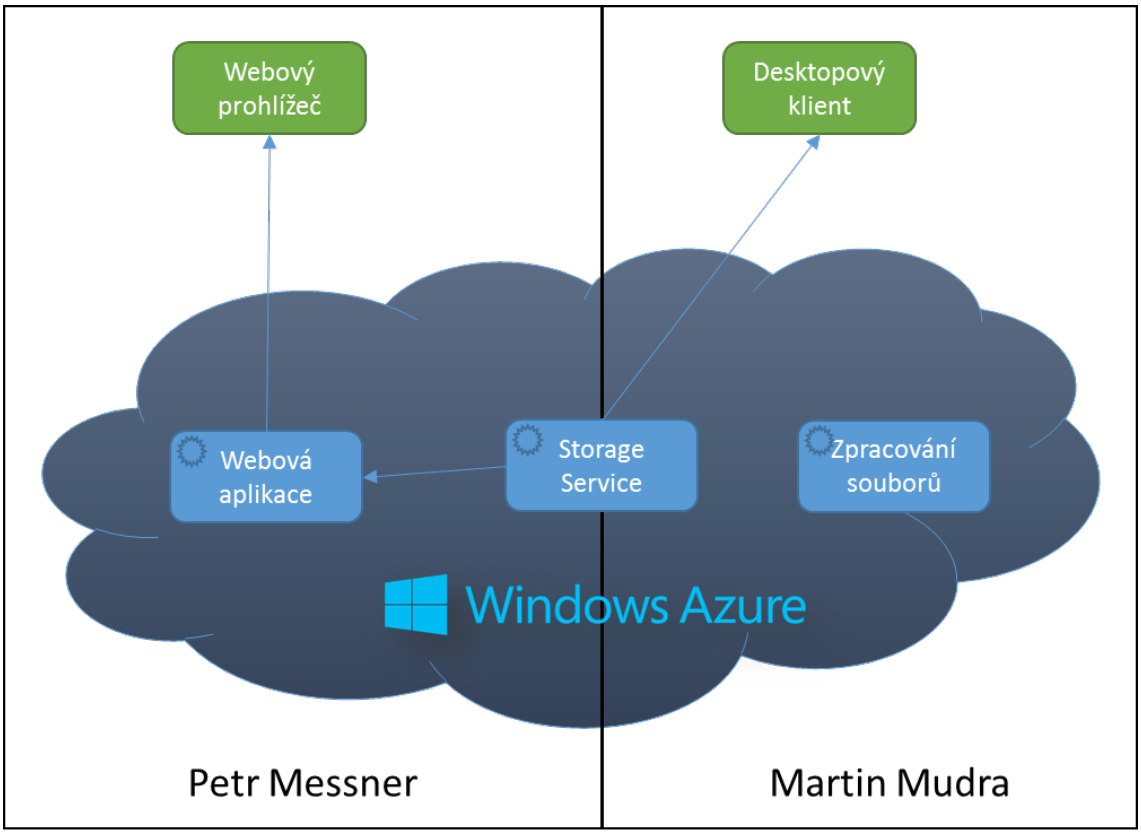
1.4.2 Současný stav

Nová implementace systému je po komponentách rozdělna mezi dva lidi, jak je zobrazeno na obrázku 1.2

1. Bc. Martin Mudra(*Diplomová práce*)
    - Storage Service (služba zajištující ukládání souborů)
        - Architektura služby
        - Základní operace se souborovým systémem na serverové straně
            * Nahrát soubor
            * Nahrát novou verzi souboru
            * Smazání souboru
            * Vytvoření složky
        - Komunikace se službou zpracovávající soubory
        - Verzování entit v databázi
    - Zpracování souborů na serveru
    - Desktopový klient se základními operacemi
2. Petr Messner(*Bakalářká práce*)
    - Webová aplikace
    - Storage Service
        - Řešení konfliktů
        - Distribuce aktualizací jednotlivým klientům pomocí pollingu
        - Pokročilé operace se souborovým systémem
            * Rekurzivní smazání složky
            * Kopírování souboru bez nutnosti nahrávání dat
            * Přesun souboru mezi složkami
        - Pokročilé operace se souborovým systémem
            * Kopírování souborů file (bez nahrávání dodatečných dat)
            * Přesun souborů mezi složkami
            * Smazání celé složky v souborovém systému



Obrázek 1.1: Rozdělení práce v původním projektu



Obrázek 1.2: Rozdělení práce projektu

Kapitola 2

Analýza

2.1 Porovnání s konkurencí

V současné době je na trhu mnoho produktů umožňujících synchronizaci a nahrávání souborů. Je velmi těžké je všechny detailně porovnat, protože fungování serverové strany je u většiny hlavních konkurenčních produktů utajené. Můžeme však vycházet z předpokladu, že většina z nich využívá nějakou formu minimalizace ukládaných dat. Budou porovnány pouze největší podobné produkty, protože počet produktů nabízejících obdobnou funkcionalitu je mnoho a jich počet se neustále mění. Porovnání hlavní konkurenčních produktů je platné k datu: 10. května 2013.

2.1.1 Dropbox

Dropbox[1] je velmi dobře známá aplikace pro zálohování a synchronizace souborů, protože byla jedna z prvních, které se podařilo prosadit mezi širokou veřejnost. Dropbox umožňuje základní verzování souborů ale tato funkce je dostupná pouze s některými klienty. Dropbox také umožňuje sdílení souborů a to i s uživateli, kteří dropbox jinak nevyužívají a nemají založený ani uživatelský účet [6] Dropbox je však silně zaměřen na jednotlivé uživatele a menší společnosti v business sféře.

Dropbox nabízí v současné době[5]:

- Webové rozhraní (verzování pouze read-only)
- desktopového klient (plná funkčnost, částečná funkcionaliza, odkazuje na webové rozhraní)
- Windows Store klient (částečná funkčnost, pouze read-only)
- Klient pro Linux
- Klient pro Os X
- Mobilní klient pro všechny dostupné platformy smartphone

[1] Dropbox (HomePage)<https://www.dropbox.com/>

2.1.2 Skydrive

Skydrive[2] je sdílení a zálohovací řešení od společnosti Microsoft. Skydrive nabízí velmi rozsáhlou integraci do nových produktů společnosti Microsoft a nabízí například přímé ukládání dokumentů do skydrive z kancelářského balíku Office ve verzi 2013. Do Skydrive se přihlašuje pomocí Live Id, které je rozšířené do celé sady produktů Microsoft(Outlook.com, Windows 8 apod.) Skydrive umožňuje také sdílení souboru nebo celé složky i uživatelům, kteří nemají vlastní Live Id pomocí unikátního odkazu pro webový prohlížeč. Skydrive podporuje verzování dokumentů, avšak v současné verzi uchovává pouze posledních 25 verzí [2]. Skydrive poskytuje v současné době [26]:

- Webové rozhraní (plná funkčnost)
- Desktopového klient (plná funkčnost)
- Windows Store klient (částečná funkčnost, pouze read-only)
- Mobilní klient pro všechny dostupné platformy smartphone

2.1.3 Google Drive

Google drive[3] je poměrně mladý produkt publikovaný 24. dubna 2012 [22]. Google drive je produkt striktně zaměřený na samostatné uživatele. Google drive využívá Google account, který je použit v celé řadě dalších dokumentů od Google (např. Gmail, Google Calendar apod.). Google drive je zaměřen hlavně na online provoz a poskytuje pouze omezenou možnost offline editaci a synchronizaci souborů [9]. Mezi hlavní přednosti google drive však patří pokročilý online editor dokumentů (Google Docs). Google drive poskytuje omezené verzování souborů [8] a maže automaticky starší verze po 30 dnech [8][22]. Google drive poskytuje v současné době [22]:

- Webové rozhraní (plná funkčnost)
- Desktopového klienta (omezená funkctionalita v offline režimu)
- Mobilní klienty pro Android a iOS [22]

2.2 Analýza požadavků

2.2.1 Funkční požadavky

- Systém musí být schopný synchronizovat vybranou složku mezi různými klienty.
- Systém musí podporovat verzování souborů
- Systém musí podporovat stáhnutí libovolné starší verze souboru
- Systém musí umět řešit konflikty (řešeno v Bakalářské práci Petra Messnera)

[2] Skydrive (HomePage)<https://skydrive.live.com/>
[3] Google drive (HomePage)<https://drive.google.com/>

- Systém musí podporovat více uživatelů
- Synchronizace souboru by měla proběhnout co nejdříve po zaznamenání změny uživatelem
- Systém musí minimalizovat množství ukládaných dat do cloudového úložiště (ppři rozumné časové dostupnosti souborů uživateli)
- Uživatelské role celého systému musí být následující:
  - Koncový uživatel s klientem na Windows
  - Koncový uživatel s klientem pro webový prohlížeč
  - Administrator serverové strany
  - Administrator virtualizačního systému
  - Provozovatel celého systému (Poskytovatel cloudové platformy)

2.2.2 Nefunkční požadavky

- Systém musí být jednoduše škálovatelný, aby byl schopen obsluhovat narůstající počet uživelů
- Systém musí být elastický
- Desktopový klient musí podporovat instalaci do windows pomocí instalátoru
- Desktopový klient by měl být snadno použitelný
- Desktopový klient by měl zatěžovat uživatelský počítač svým během jen minimálně
- Systém by měl být rozdělen do několika služeb kvůli možné škálovatelnosti v prostředí cloudu
- Systém by měl mít vrstevnatou architekturu
- Systém by měl běžet v cloudu (kvůli dostupnosti a dalším výhodám)
- Systém by měl mít co nejméně bottlenecků

2.3 Analýza současného stavu

2.3.1 Analýza architektury stávajícího systému

Na obrázku 2.1 je zobrazena stávající architektura serverové strany a jedotlivé komunikace mezi službami. Klienti komunikují pomocí rozhraní s RESTful službou *StorageService*, která jim zprostředkuje veškerou dostupnou funkcionalitu serveru. Tato služba zpracuje údaje o souboru do Azure SQL databáze a zároveň zapíše data souboru do Azure BLOB úložiště. Následně *Storage Service* podle verze nahraného souboru zaznamená do fronty zpráv (Azure

Queues) požadavek na zpracování nově příchozího souboru a pomocí Windows Communication Foundation (dále nazýváno pouze WCF) rozhraní notifikuje neustále běžící serverovou roli *FileProcessor*, která se stará o zpracování požadavků od Storage Service.

Hlavní výhodou této architektury je vysoká škálovatelnost služby StorageService pro klienty. Protože se jedná o službu, která nemá vnitřní stav, může být nasazeno mnoho instatncí této služby zároveň podle počtu klientů a jejich aktuální potřeby. Všechny tyto instance však přistupují ke společnému úložišti ve formě Azure SQL databáze, na kterou však není při jednotlivých operacích vyvíjena veliká zátěž, protože se zde ukládají jenom metainformace pro jednotlivé soubory.

Zároveň všechny tyto instance zapisují do stejného datového úložitě data souborů. Typ datového úložiště je typu Windows Azure BLOB, které je však pro takovéto úkoly určeno a je schopné poskytnout dostatečnou výkonnost na základě předplacené škálovatelnosti [12][17].

Služba *FileProcessor* je zodpovědná za minimalizaci ukládaných dat v cloudovém prostře pomocí počítání rozdílových verzí algoritmem pro hledání maximální společné podsekvence.

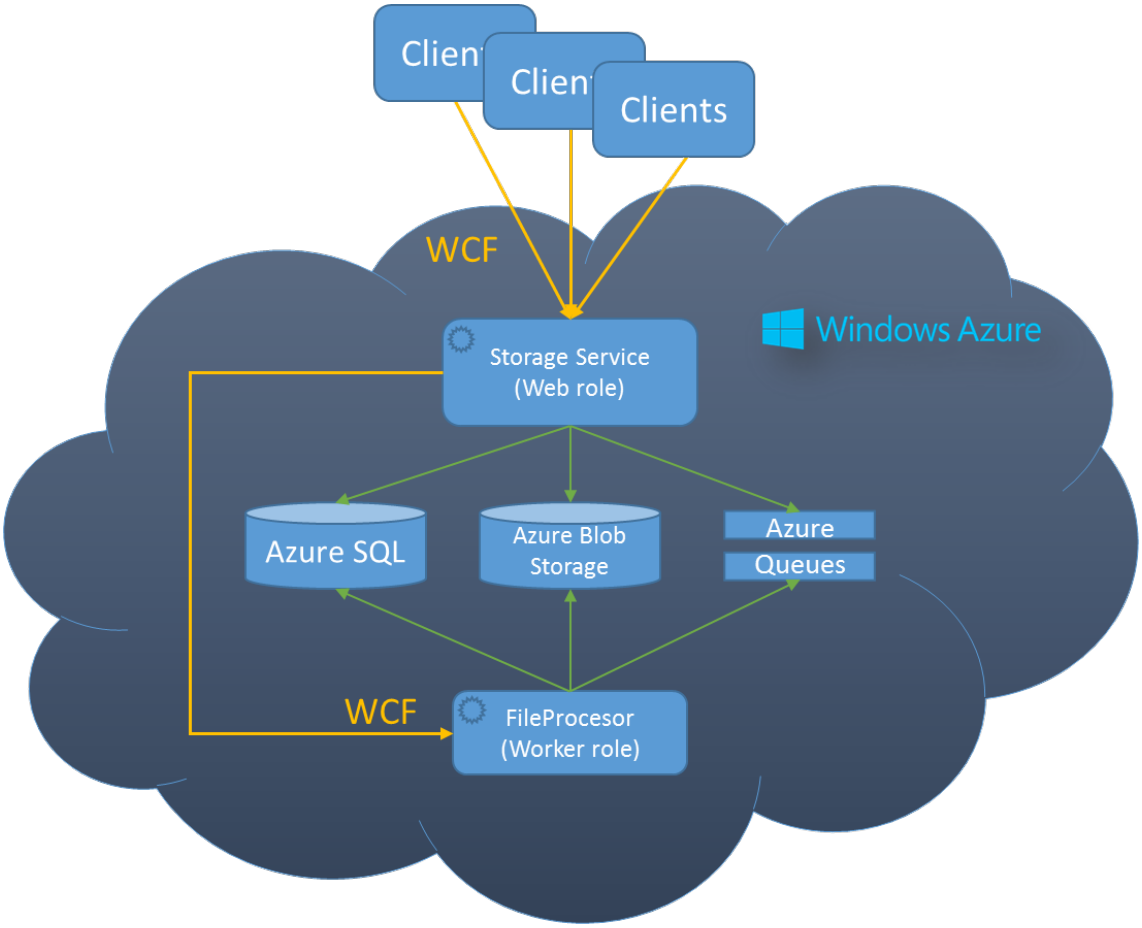
Služba *FileProcessor* běží neustále. Po přijetí požadavku přes WCF notifikaci zkontroluje fronty zpráv a zpracuje první požadavek dostupný ve frontě. Služba *FileProcessor* zapíše výsledek zpracování do datového BLOB úložiště a pozmění příslušná data o zpracování požadavku do Azure SQL databáze. Následně také smaže požadavek z fronty. Oddělení služby *FileProcessor* od hlavní služby je z důvodu časové náročnosti operací potřebných k minimalizaci dat.

Hlavním omezením tohoto řešení je však přílišná vzájemná závislost služeb *Storage Service* a *FileProcessor*. Aby celý systém fungoval, musí dojít k notifikaci služby služby *FileProcessor* službou *Storage Service*. V případě selhání této notifikace pomocí WCF rozhraní může dojít k případu, kdy se služba *FileProcessor* nedozví o novém úkolu ve frontách a ke zpracování tohoto úkolu vůbec nedojde. V opačném případě, kdy je úložištěm požadováno velké množství požadavků, může dojít k situaci, kdy přenosový kanál mezi *Storage Service* a *FileProcessor* přestane stíhat přenášet požadavky.

V případě využití vzájemných notifikací je také zbytečné použití přenosu dat přes Azure Queues, protože veškerá potřebná data k zadání úkolu FileProcesor službě mohou být serializována a zaslána jako součást notifikace přes WCF rozhraní.

Vzájemná závislost těchto služeb také značně snižuje škálovatelnost služby FileProcessor, protože při procesu přidávání nové instance FileProcesor musí systém zajistit, aby se o této službě dozvěděly všechny stávající instance služby StorageService a začaly posílat notifikace pomocí WCF o nových úkolech. Dokud není toto zajištěno, není nově přidaná instance FileProcesor zapojena do procesu zpracování souborů.

V diplomové práci Bc. Tomáše Budína je popsána architektura systému, která spojení mezi službami *Storage Service* a *FileProcessor* pomocí WCF neobsahuje a komunikace úložiště Azure Queues[3]. Tato varianta eliminuje prakticky všechny nedostatky architektury, protože umožní nezávislou škálovatelnost obou služeb. Architektura vymyšlená a popsaná Bc. Tomášem Budínem byla tedy vyhodnocena jako vyhovující požadavků a po menších zodpovědností jednotlivých služeb je možné ji použít.



Obrázek 2.1: Architektura serverové strany současného stavu a vyznačené komunikace a závislosti

2.3.2 Aktuální stav implementace

Současná implementace projektu trpěla mnoha, převážně implementačními, nedostatky, jako jsou například neošetřené výjimky, špatné návratové hodnoty, nedodržování best practices a doporučeného názvosloví pro jazyk C# [28]. Jednotlivé implementační komponenty byly na sobě příliš závislé a záměna částečné funkčnosti nebyla implementačně jednoduše umožněna. RESTful služba pro ukládání souboru poskytovala komunikační rozhraní pomocí WCF, které však nesplňovalo specifikace WCF, a automatické generování Web Services Desctiption Language (dále nazýváno pouze WSDL) služby selhávalo. Nebylo tak možné vyhodnotit komunikační rozhraní jinak než zpětnou analýzou kódu. Současná spolupráce více lidí najednou také vyústila v nefunkčnost některých požadovaných funkcionalit systému. Inicializace celého systému do prvotního stavu byla provedena pomocí unit testu, který však selhával a nebylo tedy možné celý systém bezproblémově nainicializovat do fungujícího stavu.

Služba zpracovávající soubory používala algoritmus pro hledání nejdelší společné podsekvence ve dvou porovnávaných souborech, který potřeboval kvadratické množství paměti a

porovnání s délkou vstupních souborů [24], což se u větších souborů projevilo jako nedostatečné. Po aktualizaci verze Azure Software Development Kitu (dále jen Azure SDK) na verzi 1.8 se služba zpracovávající soubory stala nefunkční a oprava této služby byla vyhodnocena jako netriviální.

Na základě těchto poznatků byl tedy kód označen jako nadále nerozšiřitelný a bylo potřeba celý systém reimplementovat znovu, za použití znalostí nabytých ze současného systému.

2.4 Přehled technických řešení pro implementaci

Na základě poznatků z vývoje stávajícího systému a zhodnocení požadavků na systém bylo rozhodnuto, že jako implementační jazyk bude použit **C♯** , byla však použita jeho novější verze 5.0, která přinesla vylepšení v implementaci asynchronního běhu aplikace.

Jako aplikační struktura celého řešení byl použit **klient-server model**, který umožňuje oddělení aplikační logiky pro uživatele od operací se samotnými soubory. Klientská část je tvořena aplikací pro windows nebo webovou stránkou,

Při výběru klientské platformy byla zvážena aplikace typu Windows Store App, která běží na platformě *Modern UI* jež je součástí operačního systému Windows 8. Při zkoumání možností této platformy byly však nalezeny stávající limity této platformy, které ji v současné době činí velmi těžko použitelnou pro klienta systému. Zejména se jedná o velmi omezený přístup k souborovému systému celého počítače[16] a možnost použít pouze omezenou část .NET frameworku. Jako platforma pro klienta tedy bylo vybrán standartní desktopové rozhraní. Pro vytvoření okna aplikace bylo rozhodnuto použít technologii **WinForms**, která je součástí .NET frameworku již od verze 2.0. Bylo však rozhodnuto připravit systém co nejvíce na klienta i pro platformu *Modern UI*.

Desktopová aplikace je tlustý klient, kopie souborů má tedy uložené u sebe a od serveru vyžaduje synchronizační funkce. Aplikace, která bude zajišťovat monitorování souborů u klienta a bude vše vzdáleně nahrávat na server. Webová aplikace je kvůli omezení webových technologií tenký klient a vyžaduje tedy od serveru specifickou funkcionalitu. Serverová strana systému tedy musí podporovat zároveň tenké i tlusté klienty včetně jejich vzájemné kompatibility.

Jako cloudové prostředí, na kterém běží serverová strana, byl vybrán **Windows Azure**, který poskytuje veškeré potřebné služby pro implementaci celého systému. Vzájemná komunikace bude zprostředkována pomocí WCF, které je součástí .NET frameworku od verze 3.0 [21].

Metadata o souborů jsou ukládána do relační SQL databáze **Azure SQL** poskytovaná cloudovou platformou Windows Azure. Pro načítání informací z databáze byl použit open-source Object-Relational Mapping (dále jen ORM) framework **NHibernate**[4], který je portací *Hibernate* frameworku pro C♯ a je uveden pod licencí *GNU Lesser General Public License* [25].

[4] NHibernate (HomePage)<http://nhforge.org/>

Pro monitorování změn, testování a usnadnění vývoje je použit framework **Apache log4net**[5] pro umožňující snadné logování, který je součástí projektu *Apache Logging Services*.

Pro výpočet rozdílových informací mezi jednotlivými soubory je použit nástroj **bsdiff** [6] v jeho portaci pro jazyk C♯ , která je distribuována pod licencí *BSD Protection License from Feb. 2002* [4][1]. Výpočet rozdílového souboru v této implementaci pracuje s paměťovou náročností[1]:

$$max(17 * n, 9 * n + m) + O(1)$$

, kde n je velikost původního souboru a m je velikost nového souboru. Časová složitost výpočtu pak probíhá se složitostí

$$O((n + m)log(n))$$

Paměťová náročnost na zpětné aplikování rozdílového souboru na nový soubor je následující[1]:

$$n + m + O(1)$$

, kde n je velikost původního souboru a m je velikost nového souboru. Časová složitost procesu aplikace je pak:

$$O(n + m)$$

Pro snížení závislostí komponent na sobě byl použit Inversion of Control (dále také jako IOC) framework **WindsorCastle**[7] ve verzi 3.1, který je součástí projektu *Castle Project*. Project Castle je distribuován pod licencí *Apache License 2.0* [20].

2.4.1 Technika Inversion of Control

Inversion of Control je technika programování, která snižuje vzájemnou závislost jednotlivých komponent na sobě. Jednotlivé komponenty jsou na sobě v ideálním případě minimálně závislé, ale přesto jsou schopny vzájemně spolupracovat. Tohoto stavu se však ve větších projektech těžko dosahuje. V tradičním objektovém programování je běh funkční logiky určen objekty, které jsou staticky přiřazené vzájemně a zároveň jsou předávány nově tvořeným komponentám, což ústí ve vzájemnou závislost komponent. Použití IOC vzájemnou závislost komponent snižuje, protože jednotlivé závislosti komponent nejsou určeny v kódu tvůrcem komponenty, ale třetí stranou, která závislosti určuje podle stanovených pravidel. Touto třetí stranou bývá typicky nějaký IOC Container, což je komponenta absolutně nezávislá na všech ostatních částech systému. Závislosti také nejsou určovány při vývoji, ale až při samotném běhu programu [23], kdy dochází k sestavení grafu závislostí a k následnému injektování potřebných komponent. Tento proces lze uživatelsky ovlivnit, což může vést v ideálním případě k totální změně chování aplikace, bez rekompilace aplikace. Při procesu injektování komponent je také možné předávat vlastní hodnoty jednotlivým komponentám, což je vhodné například pro vytvoření centralizované konfigurace celé aplikace. IOC Container se také může starat o životní cyklus jednotlivých komponent, čímž se částečně deduplikuje kód komponent a usnadní se vývoj programátorům.

[5] Apache log4net (HomePage)<http://logging.apache.org/log4net/>
[6] bsdiff (HomePage) <http://sites.inka.de/tesla/others.html>
[7] WindsorCastle (HomePage)<http://www.castleproject.org/>

Použití IOC nemá však jen výhody. Jednou z hlavních nevýhod je delší start aplikace, protože musí dojít k sestavení grafu závislostí. Sestavení grafu závislostí je typicky časově velmi náročný proces, kde jeho složitost stoupá s počtem komponent použitých v IOC. použití IOC Containeru je tedy vhodné na místech, kde nevadí delší start nebo se neočekává velmi časté restartování celé aplikace. Použití techniky IOC eliminuje statickou analýzu kódu, protože jsou závislosti tvořeny až při běhu programu. Použití techniky IOC je náchylné na vytvoření cyklické závislosti komponent, o které však tvůrce nemusí vědět, a to až do chvíle použití libovolné komponenty z tohoto cyklu [23].

Výhody IOC:

- Snížená vzájemná závislost komponent.
- Zřetelnější a vynucenější rozdělení zodpovědností jednotlivých komponent.
- Výměna komponenty neovlivní ostatní části systému.
- Možnost konfigurace tvoření komponent.
- Centralizovaná konfigurace životních cyklů jednotlivých komponent

Nevýhody IOC:

- Delší start aplikace.
- Nemožná statická analýza kódu.
- Náchylnost na cyklické závislosti jednotlivých komponent

2.4.2 Analýza cloudového prostředí Windows Azure

Windows Azure je cloudová platforma a infrastruktura poskytovaná společností Microsoft založená na virtualizaci jednotlivých instancí. Na obrázku 2.2 je zobrazeno cílení platformy Windows Azure a rozsah poskytovaných služeb[13].

Platforma je cílená jako PAAS (Platform As A Service), ale poskytuje i některé prvky z IAAS (Infrastructure As A Service 2.2). Uživatel si může například vybrat jeden z dostupných operačních systémů, který bude poskytován jako virtuální hostované prostředí. Dostupné operační systémy k datu 10. května 2013[7]:

- Windows Server 2012
- Windows Server 2008 R2 SP1
- OpenLogic Cent OS 6.3
- Ubuntu Server 12.04 LTS
- Ubuntu Server 12.10
- SUSE Linux Enterprise Server 11 SP2

U operačních systémů Windows Server 2012 a 2008 R2 také existuje na výběr několik variant nainstalovaných aktualizací. Virtuální prostředí pro uživatele je tvořeno virtualizací pomocí upravené Hyper-V virtualizece známé jako Windows Azure Hypervisor [19]. Některé další služby poskytované cloudovou platformou Windows Azure jsou:

- Web Sites - Hostování základních webových stránek vytvořených pomocí .NET, PHP nebo Node.js [19]
- Cloud services - Jedná se o hostované služby ve virtualizovaném prostředí. Windows Azure v současné době nabízí sedm výkonnostních variant rozdílných v množství dostupné paměti RAM a počtu dostupných virtuálních výpočetních jader procesoru, které jsou rozlišeny podle ceny pro hostování jednotlivých instancí. Cloud services se nadále dělí podle druhu běhu na dvě kategorie:
  - Web Role - Jedná se o službu hostovanou pomocí interní Internet Information Service (dále jen IIS) běžícího ve virtualizovaném prostředí. Verze IIS se liší podle operačního systému běžícího na virtuáním počítači. Jako webová role může být nasazena webová stránka, služba vytvořená v .NET frameworku nebo libovolná aplikace hostovaná pomocí IIS.
  - Worker Role - Aplikace tohoto typu běží ve vlastním procesu a má standardní aplikační průběh. Worker Role byly umožněny za účelem zpracování časově náročných operací, které potřebují velké množství paměti a výpočetních prostředků.
- Data management - Možnosti dlouhodobého uchování dat. V současné době jsou poskytovány tyto možnosti:
  - Azure SQL - Plnohodnotná SQL databáze s podporou podmnožiny Transact SQL (dále jen TSQL). Oproti SQL Serveru má však několik omezení:
    * Omezení příkazů pro interní správu databází na serveru
    * Spouštění distribuovanýc transakcí
    * Kompresi dat
    * Replikaci SQL serveru
    * Mirroring databáze
    * Autentizace uživatelů - Azure SQL databáze nepodporuje autentizaci přístupů pomocí doménových jmen a je nutné použít autentizaci přístupů pomocí vnitřní SQL server autentizace
    * Nemožnost spustit procesy SQL Agent, který slouží ke spuštění vzdálených operací a naplánovaných úloh.
  - BLOB Storage - Úložiště určené pro velký objem dat. Data jsou chráněné proti ztrátě redundancí.
  - Azure Tables - Základní úložiště založené na tabulkovém principu, které nepodporuje složitější operace s daty. Hlavní výhodou Azure Tables je však velká podpora škálovatelnosti a replikace. Data v Azure tables jsou také chráněna proti ztrátě redundancí.

- Messaging - Zprostředkovaná komunikace mezi jednotlivými Cloud services. Windows Azure v současné době poskytuje dva druhy zprostředkované komunikace:
  - Azure Queues - Fronty zpráv založené na REST rozhraní, které umožňuje pouze tři:
    1. GET
    2. PUT
    3. PEEK

    Jedná se o škálovatelné komunikační úložiště, které má omezenou životnost zpráv na 7 dní [10]. Do zprávy je možné uložit text nebo data o maximální velikosti 64KB (do Azure SDK verze 1.6 pouze 8KB [18])
  - Azure Service Bus - Zprostředkované zpravodajství front komunikujících přes plné WCF rozhraní. Umožňuje přihlásit se k odběru určitých zpráv. Zpráva je omezena na 256KB [10]. Limit fronty je pak stanoven na 5GB. Exkluzivní přístupový mód je řešen pomocí zamykaní [10].
- Azure Scheduler - Plánovač pro spuštění periodické činností s určitý intervalem

Služby Azure Blob, Azure Tables a Azure Queues jsou částí služby Azure Storage a mají společné vlastnosti ukládání a přístupu k datům. Data uložená v těchto úložištích jsou chráněna proti ztrátě automatickou redundancí. U redundance jsou poskytnuty dvě varianty.

- Lokálně redundantní - Redundantní data jsou uložena ve stejném datacentru, jako originální.
- Geo redundantní - Redundantní data jsou uložena v rozdílném datacentru než originální

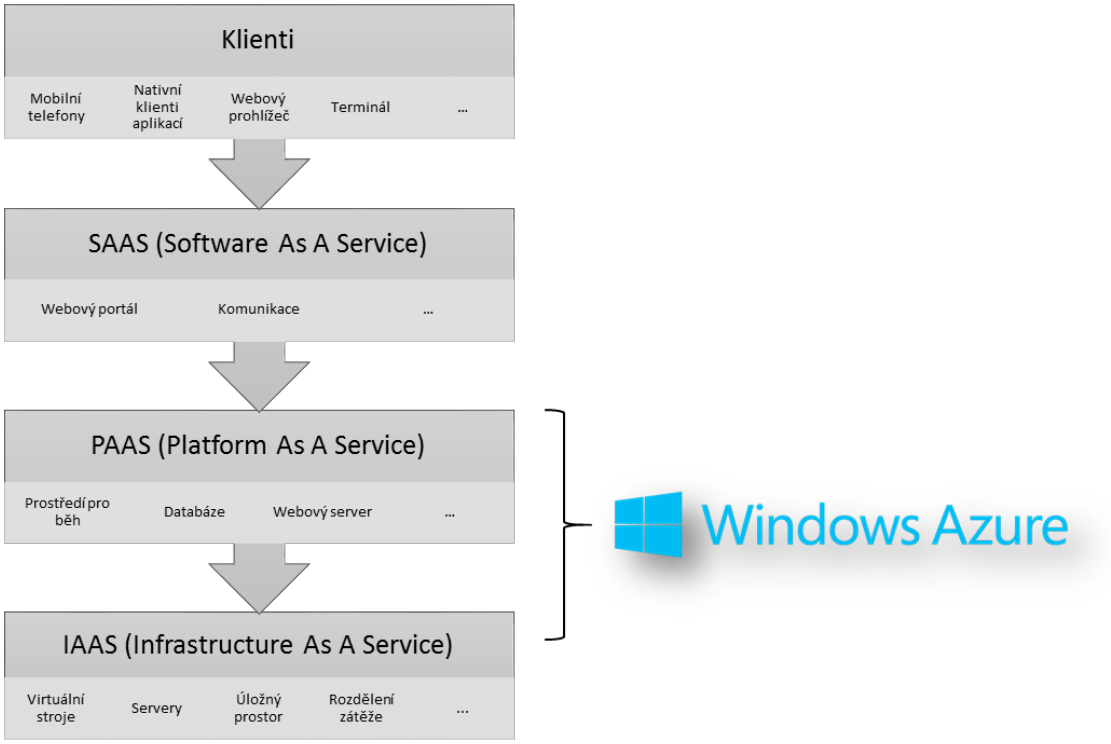
2.4.2.1 Ceny Windows Azure

Všechny ceny zde uvedené jsou platné k datu 10. května 2013a jsou určeny podle tarifu pay-as-you-go, který neobsahuje žádné závazky na delší dobu. Windows Azure podporuje ještě předplatné na 6 nebo 12 měsíců, kde jsou pak ceny nižší [11].

V tabulce 2.1 jsou zobrazeny ceny a parametry jednotlivých instancí pro hostování jednotlivých cloud services. Cena za jednotlivé instance je počítána na hodinové bázi a je účtováno za každou započatou hodinu, kdy byla cloud service nasazena, a to i když se nachází v zastaveném stavu [11].

Azure SQL databáze je účtována na denní bázi a odvíjí se podle velikosti samotné databáze. V tabulce 2.2 jsou zobrazeny ceny podle velikosti databáze. Ceny jsou uváděny v měsíčních poplatcích za databázi, avšak cena je vždy dělena počtem dní v aktuálním měsíci [11].

Azure BLOB, Azure Tables a Azure Queues jsou účtovány společně, protože spadají do služby Azure Storage. U všech služeb Azure Storage je počítáno množství dat, které jsou aktuálně uloženy, a zároveň počet transakcí, kterými je k těmto datům přistupováno. Množství uložených dat je počítáno po jednotlivých dnech. Transakcí se v tomto případě myslí



Obrázek 2.2: Cílení služby Windows Azure

operace čtení, nebo zápisu. Při změně libovolných dat, jsou tedy účtovány dvě transakce, protože musíme data přečíst a následně zapsat. Cena transakcí byla stanovena na **$0.01** za 100000 transakcí. Počet transakcí není nijak omezen [11].

Účtování za službu Azure Service Bus je založeno na počítání počtu uložených zpráv a počtu hodin, kdy je k odběru zpráv přihlášen jeden posluchač. Cena za zprávy je stanovena na **$0,01** za každých započatých 10000 zpráv. Zpráva má v tomto případě velikostní limit 64KB a v případě přesahu tohoto limitu je odečtena další zpráva. Stále však platí technologicky stanovený limit 256 KB na zprávu a maximálně 5 GB dat aktuálně uložených. Cena za každou hodinu a každého posluchače je stanovena na **$0,10** za každých započatých 100 hodin.

Další poplatky jsou účtovány při využití odchozích dat z cloudové platformy. Cena za odchozí data je účtována podle objemu a podle geologického umístění rozděleného do dvou zón[18]:

- Zóna 1 - Východní Spojené Státy, Západní Spojené Státy, Jižní centrální Spojené Státy, Severní Evropa, Západní Evropa
- Zóna 2 - Východní Asie a Jihovýchodní Asie

Ceny odchozích připojení jsou v tabulce 2.3. Pro větší objem odchozích dat je nutné se dohodnout přímo se společností Microsoft [11]. Veškerá příchozí data jsou zdarma.

Jméno instance	Počet virtuálních jader	Paměť RAM	Cena za hodinu
Extra Small (A0)	Sdílené jádro	768 MB	$0,02
Small (A1)	1	1,75 GB	$0,08
Medium (A2)	2	3,5 GB	$0,16
Large (A3)	4	7 GB	$0,32
Extra Large (A4)	8	14 GB	$0,64
A6	4	28 GB	$0,90
A7	8	56 GB	$1,80

Tabulka 2.1: Ceny a parametry instancí pro hostování cloud services [11]

Velikost databáze	Cena za databázi za měsíc	Cena za každý další GB velikosti
0 až 100 MB	$4.995	-
100 MB až 1 GB	$9.99	-
1 GB až 10 GB	$9.99 (za první GB)	$3.996
10 GB až 50 GB	$45.954 (za prvních 10 GB)	$1.996
50 GB až 150 GB	$125.874 (za prvních 50 GB)	$0.999

Tabulka 2.2: Ceny a parametry SQL databází podle velikosti uložených dat [11]

Množství odchozích dat za měsíc	Zóna 1	Zóna 2
Prvních 5GB	Zdarma	Zdarma
5 GB až 10 TB	$0.12 za GB	$0.19 za GB
dalších 40 TB	$0.09 za GB	$0.15 za GB
dalších 100 TB	$0.07 za GB	$0.13 za GB
dalších 350 TB	$0.05 za GB	$0.12 za GB

Tabulka 2.3: Ceny a parametry dat odchozích z platformy Windows Azure

Kapitola 3

Návrh řešení

3.1 Návrh architektury

Ze získaných poznatků ze současného stavu byla navržena třívrstvá architektura celého systému, která vychází ze systémové architektury navržené v Diplomové práci Bc. Tomášem Budínem a využívá veškerých výhod škálovatelnosti.

Architekturu aplikace lze rozdělit na tři vrstvy:

- Storate layer
- Business logic layer
- Client layer

Na obrázku 3.1 jsou názorně zobrazeny jednotlivé vrstvy a přílušné komponenty, které každá vrstva obsahuje. První dvě vrstvy následně tvoří serverovou stranu v architektonickém vzoru klient-server.

3.2 Storage Layer

Pro splnění požadavků na celý systém bylo potřeba rozhodnout, jaká data je zapotřebí uložit na serverové straně. Po analýze požadavků byla tato data rozděleny do několika sekcí.

- Samotná data souborů
- Metadata uložených souborů a složek (velikost, typ, čas vytvoření, název souboru atd.)
- Data pro uložení struktury souborového systému (názvy složek, hierarchie složek, hierarchie souborů, vlastník)
- Data uživatelských účtů
- Data pro komunikaci služeb mezi sebou

U všech těchto dat bylo třeba rozhodnout, do jakého typu poskytovaného úložiště budou uložena. Samotná data souborů musí být ukládána do služby typu Azure BLOB, protože ta jediná je schopna poskytnout dostatečně škálovatelné úložiště velkého objemu dat.

Metadata souborů a data pro uložení struktury souborového systému jsou si z hlediska druhu informací podobná a navíc jsou vzájemně provázána závislostmi hierarchie souborového systému (např. příslušnost data do složek atd.). Z tohoto důvodu bylo rozhodnuto, že budou ukládána do stejného úložiště. Jedná se o velmi malé množství informací, ke kterým ale bude velice často přistupováno.

Úložiště poskytovaná na platformě Windows Azure, která jsou schopna rozumně (technologicky dostupná data ke čtení a psaní) pojmout data tohoto formátu, jsou dvě: Azure SQL a Azure Tables. Protože ale při každé změně dat u klienta je nutné změnit příslušně data i na serverové straně, je nevhodné použít službu Azure Tables, u které je účtováno za přístupové transakce (viz 2.4.2). Z tohoto důvodu byla zvolena služba Azure SQL. Za těchto podmínek bylo nutné rozhodnout o datech uživatelských účtů, které také mají vazby na hierarchii souborového systému uloženého na serveru. Z důvodu možnosti automatizovaného hlídání konzistentnosti těchto dat pomocí relací mezi daty byl zvolen systém Azure SQL i pro uložení těchto informací. Zároveň se sníží nutnost potřeby hlídat v business logice vztah dat uživatelských účtů a hierarchie souborového systému, protože může položením správného dotazu tyto relační vztahy vyřešit přímo Azure SQL databáze a navrátit výsledky již filtrované.

Pro vzájemnou komunikaci cloud services v *business layer* mezi sebou je doporučeno přímo společností Microsoft [14] používat Azure Queues nebo Azure Service Bus, které jsou poskytovány právě za tímto účelem. Protože jsou však tyto dvě služby účtovány podle rozdílných pravidel, bylo nutné zvážit předpokládané zatížení a využití komunikace služeb. Azure Service Bus umožňuje odběr zpráv pomocí automatizovaného odebírání(viz 2.4.2), což je velice pohodlný system upozorňování na nové položky ve frontě, je však účtován za započtenou hodinu, kdy je k odběru přihlášen alespoň jeden posluchač. V navržené architektuře se počítá se vzájemnou spoluprácí obou služeb dohromady a je nezbytné používat nepřetržitou komunikaci mezi nimi. To určuje využití Azure Service Bus, jako finančně nevýhodnou variantu. Bylo tedy rozhodnuto využití Azure Service Queues s periodickým zjišťováním nových položek. Protože jsou Azure Service Queues účtovány podle přístupových transakcí (viz kapitola 2.4.2), bylo nutné zavést administračně upravitelný interval, ve kterém jsou fronty pollovány na nově čekající zprávy.

3.3 Business logic layer

Vrstva *Business logic layer* obsahuje hlavní logiku celé serverové strany a zároveň zajištuje obsluhu jednotlivých klientů. Analýzou původního stavu se prokázalo, že rozdělení do dvou služeb podle funkcionality se jeví jako vyhovující řešení a bylo proto zachováno. Vrstva obsahuje dvě služby:

- Storage Service
- File Space Service

3.3.1 Storage Service

Hlavní zodpovědností služby *Storage Service* je obsluha veškerých klientských požadavků. Služba *Storage Service* je navržena tak aby poskytovala RESTful web Application Programming Interface (dále jen web API), přes které budou přistupovat k serverovým operacím klienti tenkého i tlustého typu. Rozhraní služby je vystavené pomocí technologie WCF, která umožňuje vytvořit rozhraní komptatibilní i s různými dalšími platformami, například *Modern UI* platformy, mobilními telefony atd. Zároveň technologie umí zajistit šifrování přenášených dat pomocí běžně dostupných a používaných šifrovacích algoritmů a postupů. Rozhraní služby je vystavené pomocí technologie WSDL a z hlediska budoucí použitelnosti služby je nezbytné, aby bylo komunikační rozhraní co nejjednodušší a jednotlivé operace působily probíhaly jako atomické.

Služba *Storage Service* je zodpovědná za správu virtuálních souborových systémů ukládaných na server uživateli a správu oprávnění jednotlivých uživatelů.

Operace, které jsou klientům vystaveny, jsou podle funkcionality rozděleny do tří skupin:

1. Operace s uživateli - Operace podporující funkčnost více uživatelů - například operace registrace nového uživatele nebo operace pro přihlášení uživatele

2. Operace se složkami - Operace pro kontrolu virtuálního souborové systému uloženého na serveru

3. Operace se soubory - Operace pro správu souborů na serveru. Spadají sem například operace nahrání nového souboru, nahrání nové verze souboru apod.

Operace z kategorie 1 a 2 jsou časově nenáročné, protože veškeré operace z těchto kategorií mohou být vyřízeny ihned. Operace z kategorie 3 mohou být časově náročné, protože je nutné přenést samotná data souborů.

Služba je navržena tak, aby požadavek na server byl co nejrychleji vyřízen a nebyl zdržován libovolnými operacemi, které nejsou pro vyřízení požadavků nezbytně nutné a proto operace z kategorie 3 vyřídí interní operace se souborem v nezměněné podobě a ihned odpoví klientovi.

Služba přímo komunikuje s vrstvou *Storage layer*, kam zapisuje a čte informace, které jsou následně zprostředkovány klientům. Služba *Storage Service* je také vytvořena pomocí techniky IOC (viz kapitola 2.4.1), umožňující rychlou a bezproblémovou záměnu libovolné části. Například ukládání informací do vrstvy *Storage layer*. Tímto je zajištěna určitá úroveň modularity celého systému.

3.3.2 Uživatelská konta a autorizace požadavků

Při řešení požadavku více uživatelů je nutné dodržet bezestavovost samotné služby *Storage Service* z důvodů neomezené škálovatelnosti. Stav jednotlivých uživatelů tedy nemůže být udržován v paměti služby. Uživatelé jsou identifikováni podle unikátních tokenů, přenášených při každém požadavku z klienta na server.

Klientské účty jsou vytvořeny procesem registrace, který jako jediný neobsahuje žádnou identifikaci požadavku na server. Při registračním procesu jsou, po ověření unikátnosti, uloženy údaje z požadavku do databáze Azure SQL, čímž je uživatel považován za známého. Při

procesu přihlašování zašle klient v požadavku na přihlášení své přihlašovací jméno a otisk přihlašovacího hesla na službu *Storage Service*. Ta ověří otisk hesla uloženého při registraci a příchozího oproti otisku z požadavku na přihlášení. V případě shody je uživateli vygenerován službou unikátní token, který obsahuje informace o platnosti a uživateli. Token je následně uložen do databáze Azure SQL, čímž je uživatel považován za přihlášeného. Unikátní token je následně uživateli navrácen v odpovědi na požadavek na přihlášení. Klient si navrácený token uloží a zasílá ho s každým požadavkem na server, čímž identifikuje uživatele. Zároveň je tím zaručena atomicita jednotlivých požadavků na server, protože již není požadován proces dalšího ověření. Úspěšný proces registrace a získání autorizačního tokenu je zobrazeno na obrázku 3.2.

Služba *Storage Service* při každém požadavku identifikuje uživatele podle tokenu, který je součástí příchozácích parametrů a porovná je s informacemi, které načte z databáze Azure SQL.

3.3.2.1 Verzování

Služba *Storage Service* řeší při příchodu požadavku na nahrání souboru nezbytné operace spojené s verzováním souborů a složek tak, aby bylo zajištěno korektní uchování informací o časové návaznosti jednotlivých verzí souborů a složek. Při nahrávání souborů uživatelem dochází ke dvěma možným situacím, které jsou přes jednotlivé klienty reflektovány na službu *Storage Service*:

1. Uživatel nahrává nový soubor nebo vytváří novou složku
2. Uživatel mění soubor již nahraný nebo mění složku již vytvořenou

Při operaci nahrávání nového souboru je nutné zaznamenat údaje o nově příchozím souboru do *Storage layer* a služba *Storage Service* tedy musí vykonat několik operací v přesně danné posloupnosti.

1. Služba načte údaje o uživateli z databáze Azure SQL
2. Služba ověří identitu z požadavku a porovná jí s identitou uživatele načtenou v předchozím kroku. V případě, že identity nesouhlasí, je soubor odmítnut a s klientem je ukončena komunikace s příslušnou chybou.
3. Služba načte z databáze Azure SQL informace o složce, do které se klient snaží soubor nahrát, a ověří oprávnění uživatele pro zápis do této složky. V případě nesouhlasu oprávnění je soubor odmítnut a s klientem je ukončena veškerá komunikace s příslušnou chybou.
4. Požadavek je nyní ověřen a služba *Storage Service* zprostředkuje přenos dat od klienta do úložiště Azure BLOB.
5. Po úspěšném přenosu obsahu souboru služba vygeneruje unikátní ID souboru, které slouží k jednoznačné identifikaci souboru na serveru. Následně jsou z požadavku načtena metadata o souboru, která jsou uložena do databáze Azure SQL společně s odkazem na přenesená data v Azure BLOB a unikátním ID souboru.

6. Služba poznamená do úložiště Azure Storage Queues informace o nově příchozím souboru.

7. Uživateli je zpětně navrácen výsledek operace přidávání, jehož součástí je i vygenerované ID souboru, pod kterým soubor vystupuje na serveru.

Operace změny souboru je složitější, protože zde dovhází k několika situacím, které mají vzájemně podobný průběh a za kterých dochází k vytvoření nové verze souboru. Soubor musí být při těchto operacích identifikován svým vygenerovaným unikátním ID, které bylo předáno při vytvoření souboru nebo výpisu souborů na serveru klientovi. Nová verze souboru může vznikat v následující situaci:

- Jsou k dispozici nová data souboru
- Soubor je přejmenován
- Soubor je smazán
- Soubor je přesunut do jiné složky
- Případně různé kombinace těchto operací (Např. jsou k dispozici nová data a soubor byl přejmenován)

Nejsložitější situací je možnost, kdy jsou k dispozici nová data, či případně kombinace operací, která tuto možnost obsahuje. Při operaci aktualizace dat souboru je proces průběhu serverové operace složitější:

1. Služba načte údaje o uživateli z databáze Azure SQL

2. Služba ověří identitu z požadavku a porovná ji s identitou uživatele načtenou v předchozím kroku. V případě, že identity nesouhlasí, je soubor odmítnut a s klientem je ukončena komunikace s příslušnou chybou.

3. Služba načte z databáze Azure SQL informace o souboru, identifikovaného pomocí unikátního identifikátoru, kterému se klient snaží změnit obsah, a vyhodnotí oprávnění k zápisu do souboru a příslušné složky na serveru. V případě neúspěchu je veškerá komunikace s klientem ukončena s příslušnou chybou

4. Požadavek je nyní ověřen a služba *Storage Service* zprostředkuje přenos dat od klienta do úložiště Azure BLOB.

5. Požadavek klienta je nyní testován na konfliktní návaznost verzí a je nutné konflikt vyřešit. Tato oblast je popsána v bakalářské práci Petra Messnera.

6. Služba označí metainformace o původní verzi souboru v databázi Azure SQL jako neaktuální

7. Služba vytvoří nové metainformace o souboru na základě příchozích změn z požadavku klienta a původních informací o souboru. V nových metainformacích o souboru je změněné umístění obsahu nově uloženého v Azure BLOB.

8. Služba poznamená do úložiště Azure Storage Queues zprávu s informací o nově vzniklé verzi souboru společně s unikátním ID souboru.

9. Požadavek na změnu dat je nyní klientovi navrácen jako vyřízený. Při tomto procesu není nutné klientovi navracet jiné informace, protože veškerá data o souboru zaslal klient a ostatní informace souboru se nezměnily.

Průběh ostatních scénářů vzniku nových verzí je velmi podobný, protože se jedná o procesy, u kterých nemusí docházet k přenosu dat, ale vznikají pouze nové verze souboru se změněnými metadaty jednotlivých souborů:

1. Služba *Storage Service* načte údaje o uživateli z databáze Azure SQL

2. Služba ověří identitu z požadavku a porovná jí s identitou uživatele načtenou v předchozím kroku. V případě, že identity nesouhlasí, je soubor odmítnut a s klientem je ukončena komunikace s příslušnou chybou.

3. Služba *Storage Service* načte z databáze Azure SQL informace o souboru, identifikovaného pomocí unikátního identifikátoru, nad kterým se uživatel snaží provést operaci přejmenování, přesunu do jiné složky nebo smazání. Informace navrácené z databáze vyhodnotí služba a vyhodnotí oprávnění uživatele k zápisu do souboru a všech příslušných složek. V případě nedostatečného oprávnění je s uživatelem ukončena komunikace s příslušnou chybou.

4. Požadavek klienta na změnu souboru je nyní testován na konfliktní návaznost verzí a je nutné konflikt vyřešit. Tato oblast je popsána v bakalářské práci Petra Messnera.

5. Služba *Storage Service* označí původní metainformace o souboru v databázi Azure SQL jako neaktuální. V případě, že klient požadoval operaci smazání souboru zde process končí úspěchem.

6. Služba *Storage Service* vytvoří nové metainformace s příslušnými změnami podle typu požadavku z klienta a uloží je do databáze Azure SQL.

7. Služba poznamená do úložiště Azure Storage Queues zprávu s informací o nově vzniklé verzi souboru společně s unikátním ID souboru.

8. Požadavek je nyní klientovi navrácen jako vyřízený.

Obdobný proces verzování je aplikován i u klientských požadavků obsahující operace se složkami.

Proces verzování ústí v následující lineární řadu jednotlivých verzí, která je zobrazena na obrázku 3.3. Při dotazu na aktuální verzi jednotlivých verzovaných entit virtuálního souborového systému je uživatelům zprostředkována vždy pouze poslední metadatverze a klient je tedy od verzování entit kompletně odštítěn, a to až do chvíle, kdy jsou specificky klientem vyžádány veškeré informace o historii jednotlivé entity. Služba *Storage Service* je schopna navrátit informace o libovolné verzi včetně dat, která verze obsahuje.

3.3.2.2 Práva

Služba *Storage Service* řeší oprávnění na úrovni jednotlivých uživatelů. Každý adresář virtuálního souborového systému na serveru má v každé své verzi poznamenán vlastníka. Zároveň při verzování jednotlivých entit je zaznamenáno unikátní ID uživatele, který změnu provádí.

3.3.3 Řešení konfliktů mezi soubory

Oblast řešení konfliktů je řešena v bakalářké práci Petra Messnera.

3.3.4 File Space Service

Služba *File Space Service* je součástí business logiky a je zodpovědná za minimalizaci ukládaných dat ve *Storage layer*. *File Space Service*. Služba provádí úkoly počítání rozdílových dat mezi dvěma staršími verzemi souborů, za účelem ušetření místa v úložišti Azure Storage BLOB. Zároveň však služba umí konvertovat verzi s rozdílovými daty na verzi s plnými daty.

Služba monitoruje nově uložené zprávy v úložišti Azure Queues a v případě, že nalezne ve frontě zpráv informaci o nově příchozí verzi určitého souboru, snaží se minimalizovat uložená data jednotlivých verzí v řadě verzí tohoto souboru. Zprávy o nově příchozích souborech a jejich verzích jsou do Azure Queues přidávány službou *Storage Service*, která tím zasílá ostatním službám v *business logic layer*, určité útržky informací o chodu a aktuálním stavu celého systému.

Proces minimalizace uložených dat v Azure Storage BLOB probíhá vždy nad celou řadou verzí u souborů, ke kterým byla nahrána nová verze. Minimalizace probíhá počítáním rozdílových verzí mezi dvěma verzemi tohoto souboru.

3.3.4.1 Verzovací politika

Počítání rozdílových souborů zahrnuje dva předpoklady:

- Často používaná verze každého souboru bude vždy jen ta poslední.
- Obsah dat mezi jednotlivými verzemi bude vzájemně podobný.

Oba body vycházejí z uživatelské tvorby souborů tvořené převážně inkrementálním přidáváním informací do souboru.

Z těchto předpokladů je určena verzovací politika složená z následujících pravidel:

- Určený počet posledních verzí souboru je vždy plný
- Pro zrychlení dopočítávání plných dat ke všem verzím souboru je nutná občasná existence plných verzí vždy po určitém počtu dat.

Pravidla systému jsou přizpůsobitelná na základě nastavení služby *File Space Service*

Na obrázku 3.4 je příklad načtené řady souborů, na kterou nebyla aplikována technologie minimalizace dat pomocí dopočítávaných rozdílových verzí. Obrázek 3.5 zobrazuje stav po aplikaci minimalizačního procesu s nastavením verzovací politiky následujícím způsobem:

- Poslední dvě verze souboru budou plné
- Vždy každé čtvrté verzi souboru budou ponechána plné data pro pro rychlejší dopočítávání plných verzí souborů (počítáno vždy od poslední plné verze).

3.3.4.2 Proces diffování souborů

Proces vytváření rozdílových dat v souboru je následující:

1. Zpráva o nově příchozí verzi souboru je načtena z úložiště Azure Queues a následně je z úložiště odebrána.
2. Ze zprávy je rozpoznáno unikátní ID souboru, ke kterému byla přidána verze.
3. Pro rozpoznané ID je z databáze Azure SQL načtena kompletní historie verzí.
4. Z načtené lineární historie verzí jsou aplikací verzovacích pravidel určeny verze ke konverzi.
5. Pro každou verzi určenou ke konverzi na rozdílový soubor jsou provedeny následující kroky:
   (a) Jednou z dostupných metod je dopočítán rozdílový soubor z verze souboru v lineární řadě cílovému souboru následující
   (b) Rozdílová data jsou zapsána do úložiště Azure BLOB
   (c) Do databáze Azure SQL je zapsán záznam o výpočtu rozdílových dat, který obsahuje informace o verzi, ze které byl rozdílový soubor vytvořen, a algoritmus kterým byla data počítána.
   (d) V metadatech příslušné verze je přepsána cesta k nově uloženým datům a je označeno, že data v úložišti Azure BLOB jsou rozdílová
   (e) Původní plná data jsou z úložiště BLOB smazána

   Analogicky pro verze určené ke konverzi na plný soubor jsou provedeny následující kroky:

   (a) Z databáze Azure SQL je načten záznam o výpočtu rozdílových dat pro požadovaný soubor. Z načteného záznamu je určen algoritmus, kterým byla rozdílová data souboru vytvořena a verzi souboru, ze kterého byla rozdílová data počítána.
   (b) Aplikací rozdílových dat na plná data verze, ze kterých byla spočítána, je vytvořena plná verze dat, která je zapsána do Azure BLOB.
   (c) V metadatech příslušné verze je přepsána cesta k nově vytvořením plným datům a je označeno, že se jedná o data plná. Takto změnená data jsou uložena do databáze Azure SQL.
   (d) Původní rozdílová data jsou z úložiště Azure BLOB smazána.
   (e) Z databáze Azure SQL je smazán záznam o výpočtu rozdílových dat.

3.3.5 Dočasně plná data

Po aplikaci procesu minimalizace ukládaných dat může dojít k situaci, kdy klient požádá službu *Storage Service* o data starší verze souboru, ke které neexistují aktuálně plná data. V této situaci je nutné data souboru pro klienta převést z minimalizované formy zpátky do plné verze. Proces převodu zpět je časově, paměťově i výkonostně náročný a proto je vhodné, aby ho plnila služba *File Space Service*. Z tohoto důvodu byla vytvořena druhá fronta zpráv v úložišti Azure Queues, kam služba *Storage Space* ukládá zprávy s požadavky na soubory, které vyžadují klienti ke stažení a které nemají plnou verzi. Služba *File Space Service* prioritně hlídá zprávy i z této fronty a pokud se vyskytne požadavek v této frontě, okamžitě začíná proces zpětného převodu potřebné verze. Cílová verze není převedena na plnou verzi souboru, protože při dalším přidání verze souboru do řady verzí tohoto souboru, by byla převáděna zpět na dočasnou verzi. Místo toho jsou plná data zapsána do úložiště Azure BLOB a v databázi Azure SQL je verzi souboru označeno místo, kde leží plná data. Klientovi je mezitím službou *Storage Space* navrácena odpověď, s informací, že aktuálně plná data nejsou k dispozici. Je na klientovi, aby po nějaké době požádal o data verze znovu. Tento process je zobrazen na obrázku 3.7.

Pokud jsou data verze již k dispozici ve formě dočasně plných dat, jsou data službou *Storage Space* poskytnuta klientovi v odpovědi ihned. Na obrázku 3.6 je znázorněn sekvenční diagram požadavku v případě, že jsou plná data k dispozici. V případě, že data stále nejsou k dispozici, je opět navrácena informace o nedostupnosti dat a celý proces se opakuje.

3.3.6 Databáze

Na obrázku 3.8 je zobrazen model relační databáze Azure SQL. Model je navržen tak, aby splnil požadavky třetí normální formy [27]. Informace o uložených souborech jsou uloženy v tabulce **FileVersion** a **File**. Tabulka **FileVersion** je verzovanou entitou, která využívá verzování pomocí platnosti řádků určených časovou známkou začátku platnosti záznamu a časovou známkou konce platnosti záznamu. **FileVersion** také obsahuje referenci do tabulky **File**, která obsahuje neverzované záznamy o unikátních ID souborů nahraných na server. Obdobný systém verzování pomocí časových známek platnosti je aplikován také na informace o složkách v tabulce **FolderVersion**. V této tabulce jsou veškeré verzované informace o složkách virtuální serverové struktury adresářů. K lineární řadě verzovaných informací o složkách existuje vždy jeden záznam v tabulce Folder, kde je také uložené unikátní ID složky na serveru. Virtuální souborový systém je složen pomocí vztahu nadřazené složky ParrentFolderId, který obsahuje jak verzované informace o souborech (**FileVersion**), tak verzované informace o složkách (**FolderVersion**). Verzovnáním tohoto vztahu je umozněno udržovat dohledatelnou informaci o přesunech ve virtuálním souborovém systému. Každému uživateli je při registraci vytvořena základní složka, do které jsou mapovány veškeré podsložky. Za založení kořenové složky je zodpovědná služba *Storage Service*. Uživatelské oprávnění k zápisu či čtení je následně kontrolováno rekurzivní kontrolou celého souborového systému začínajího v kořenové složce uživatele, pro kterého jsou práva kontrolována. Uživatelské informace jsou uloženy v tabulce **User**, na kterou je navázána verzovaná tabulka **Share**, která slouží pro sdílení složek mezi uživateli. Pro přístup do databáze je použita SQL autorizace samotného serveru.

3.3.7 Škálovatelnost a úzká místa

Při analýze stavající architektury i při návrhu nové architektury bylo velmi důležité zvážit možnosti škálovatelnosti. Při škálování existují dvě možná řešení škálovatelnosti, která je potřeba zvážit.

- Scale-up - jedná se o zvýšení výkonnosti současného řešení pomocí výměny komponent za silnější. Na platformě Windows Azure tento druh škálovetalnosti odpovídá přechodu na silnější instanci virtualizovaného prostředí hostující příslušnou Cloud service.
- Scale-out - škálování je docíleno přidáním další instanci služby. V terminologii Windows Azure se jedná o přikoupení nové instance pro příslušnou Cloud service.

Škálovatelnost služeb Azure BLOB a Azure Queues je spravována přímo platformou Windows Azure a výkonnost této komponenty je omezena technologicky na společný limit kapacity pro jeden storage account účet. Limitní kapacity jsou společné a jsou [12]:

- Maximální kapacita 100 TB
- Maximální počet přístupových transakcí 5000/s
- Maximální přenosová rychlost 3 Gb/s

Současná architektura nepočítá s využitím více Windows Azure Storage účtů, ale omezení počtu transakcí by se nemělo stát limitním faktorem celého systému. Protože podstata celého systému není v ukládání velkého množšství malých informací, ale menšího množství větších souborů jednotlivých uživatelů. Limit přenosové rychlosti by se mohl stát omezujícím v případě, kdy přenosová rychlost všech současně nahrávaných souborů by dosáhla tohoto limitu. To však při běžně dostupném internetovém připojení domácností odpovídán několika tisícům uživatelů najednou. Nejvíce limitní je tedy kapacita 100 TB, která limituje maximální velikost všech uložených souborů všech uživatelů dohromady. Kapacita 100 TB je však pro použití několik mnoho uživatelů v současné době více než dostatečná. Jednoduchou úpravou celého systému by však šlo použít úložiště blob z jiného storage účtu Windows Azure, čímž by se kapacita zdvojnásobila. Úprava systému by musela být následující: do relační databáze Azure SQL, která obsahuje umístění dat každého souboru, by musela být přidána informace, která by obsahovala použití jiného storage accountu Windows Azure. Azure SQL databáze je omezena maximální velikostí databáze na 150 GB (viz 2.4.2). Velikostní limit by však neměl být dosažen protože do Azure SQL databáze jsou ukládány pouze metainformace o souborech uložených na serveru a informace o klientech, kteří k systému přistupují. Výkonnost Azure SQL databáze je škálovatelná pouze v omezeném měřítku pomocí použití federace, která nebyla v architektuře systému použita a její zavedení by vyžadovalo změny všech cloud services, které k databázi přistupují. Výkonostní nedostatečnost databáze Azure SQL není očekávána, protože každý uživatel jednotlivé soubory vždy chvilku přenáší mezi serverem a klientem, což tvoří určité omezení celého systému.

Architektura systému je navržena tak, aby mohla těžit z výhod obou druhů škálování. Platforma Windows Azure je přímo připravena na použití obou druhů škálování jednotlivých Cloud services. Použitím libovolného škálování docílíme teoreticky téměř lineárního nárůstu výkonnosti až do limitů služeb vrstvy *Storage layer*, které jsou popsány výše.

3.4 Client Layer

Vrstva *Client layer* obsahuje veškeré možnosti přístupu, které budou přímo přístupné koncovým uživatelům systému. Vrstva je tvořena Webovou aplikací, která má vlastní serverovou stranu tvořenou webovým serverem typu IIS. Tato serverová strana webové aplikace je ale v architektuře celého systému tenkým klientem, který pouze přeposílá požadavky mezi koncovým uživatelem a službou *Storage Service*. Součástí *Client layer* je i nativní aplikace pro Windows.

3.4.1 Webový klient

Tato část systému je řešena v bakalářské práci Petra Messnera

3.4.2 Windows Klient

Nativní klient pro Windows je program pro platformu Windows desktop. Klient užmožňuje uživateli využít možnosti synchronizace souborů se serverovou stranou za jeho minimální účasti. Desktopový klient se bude chovat jako tlustý klient, který umožní uživateli jednu jeho lokální složku namapovat do vzdáleného systému, a veškeré změny v této složce jsou okamžitě reflektovány na server. Pokud koncový uživatel změní data na serveru z jiného umístění jsou tyto změny co nejrychleji reflektovány do vybrané složky.

Desktopová aplikace musí plnit následující úlohy:

1. Prvotní synchronizace souborových systémů při startu, či výpadku spojení
2. V synchronizovaném stavu je zbytečné provádět plné porovnání souborových systémů a je možné synchronizovat pouze pomocí inkrementálních změn na obou stranách. Proces inkrementální synchronizace tedy musí plnit následující úkoly:
    - Monitorování lokálních změn v souborovém systému
    - Reflektace lokálních změn na server
    - Monitorování změn v souborovém systému na serverové straně
    - Reflektace změn ze serveru na lokální souborový systém

Prvotní synchronizace souborových systémů je časově velmi náročný proces, protože je nutné detekovat v jakém stavu je lokální souborový systém i v jakém stavu je vzdálený virtuální souborový systém a následně oba tyto stavy porovnat a vzájemně reflektovat změny. Tento proces je plně automatizovaný a probíhá vždy při startu desktopové aplikace nebo výpadku spojení se službou. Úkol vyhodnocení stavu lokálního systému spočívá v rekurzivní průchod monitorované složky a zaznamenání veškerých dat o složkách a zaznamenání metada o souborech, které jsou přítomny. Ke zjištění stavu vzdáleného souborového systému je poskytována službou *Storage Service* obdobná operace, jejíž výsledek je rekurzivní průchod virtuálního soborového systému do maximální možné hloubky a následně je navrácen výsledek klientovi. Operace porovnání následně prochází načtené stavy a hledá veškeré změny, které jsou okamžitě reflektovány na příslušnou stranu. Po dokončení tohoto úkolu jsou oba systémy považovány za synchronizované a desktopový klient obsahuje v tuto chvíli kopii

souborů ze serverové strany uloženou v monitorované složce. V tuto chvíli je spuštěn proces inkrementální synchronizace

Při procesu inkrementální synchronizace je nutné monitorování lokálních změn ve složce vybrané uživatele. Seznam změn nutných k detekci je následující:

- Vytvoření souboru nebo složky
- Smazání souboru nebo složky
- Přejmenování souboru nebo složky
- Přesun souboru nebo složky
- Kopírování souboru nebo složky
- Změna dat souboru

Při detekci libovolné z těchto akcí je volána příslušná metoda na serverové straně, která změnu ihned reflektuje na souborový systém uložený v databázi. Z důvodů bezestavosti služby *Storage Service*, která tvoří pro klienty vstupní bod serverové strany, je nutné detekovat změny souborového systému uloženého na serveru pomocí metody pollingu. Metoda pollingu je založena na periodicky opakovaných dotazech klienta na službu. Na obrázku 3.9 je zobrazeno periodické polování změn klientem ze serveru a následné získání metada o složce se změněným obsahem. Tento obsah je následně porovnán s lokálním obsahem a jsou provedeny příslušné operace, kterými jsou:

- Vytvoření lokální složky.
- Rekurzivní smazání celé složky.
- Vytvoření nového souboru a stažení jeho dat.
- Stažení nových dat souboru a jejich zápis do již existujícího souboru.
- Smazání lokálního souboru
- Přesun souboru

Ve chvíli provádění lokálnch změn aktualizacemi ze serveru je nutné vyřadit veškeré změny prováděné procesem synchronizace ze monitorovaných událostí lokálního souborového systému, protože jinak by došlo k cyklickému vzájemnému vytváření nových verzí klientem na serveru.

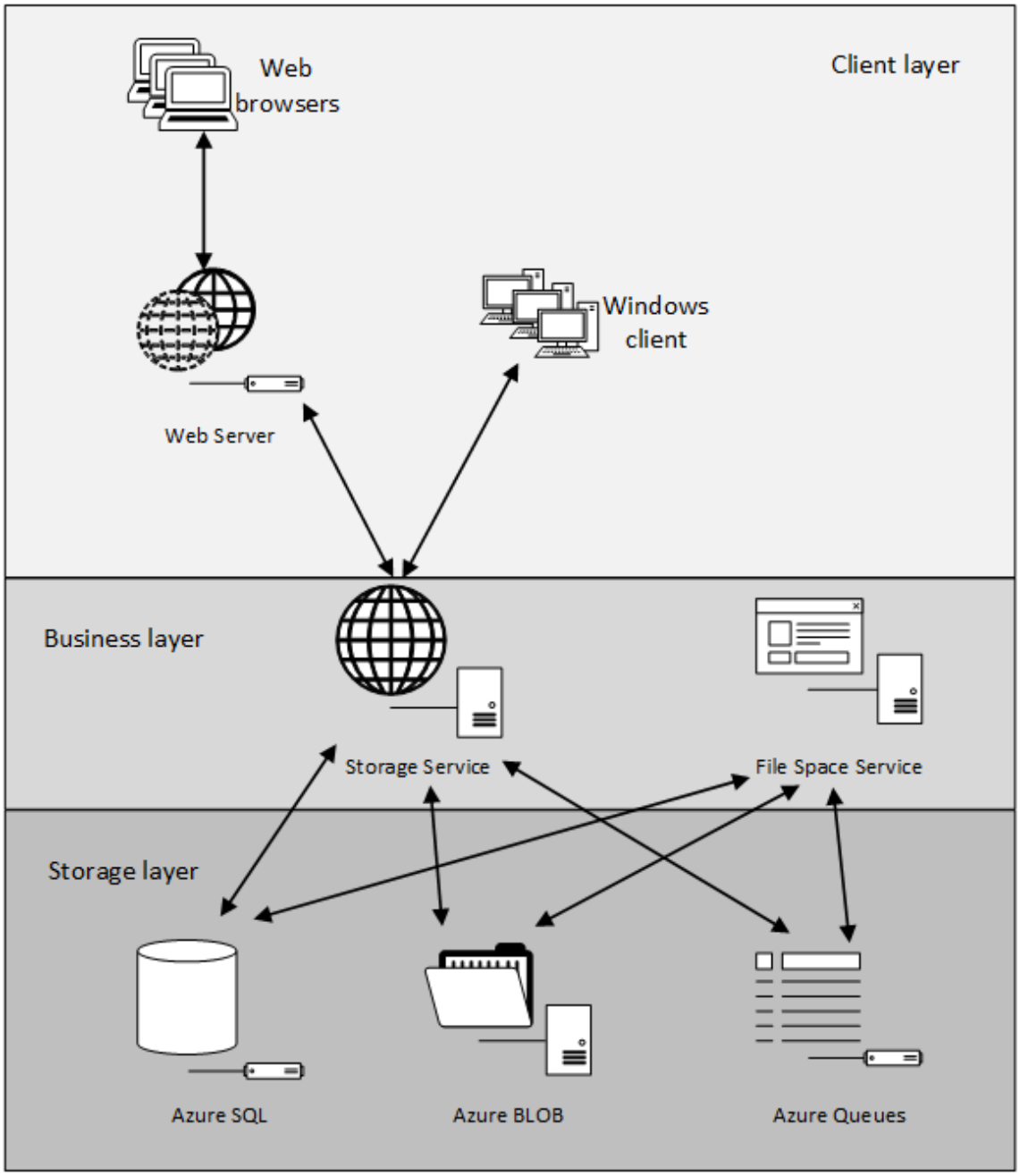
3.4.3 Schéma nasazení do cloudového prostředí Windows Azure

Služba *Storage Service* je nasazena do cloudového prostředí jako Web role, protože je požadován její běh až ve chvíli libovolného požadavku klienta. Služba běží v interním poskytovaném serveru IIS (Internet Information Service), který je součástí služby Cloud Services typu Web role. Služba *File Space Service* je nasazena jako Worker role, protože monitoruje přidávané

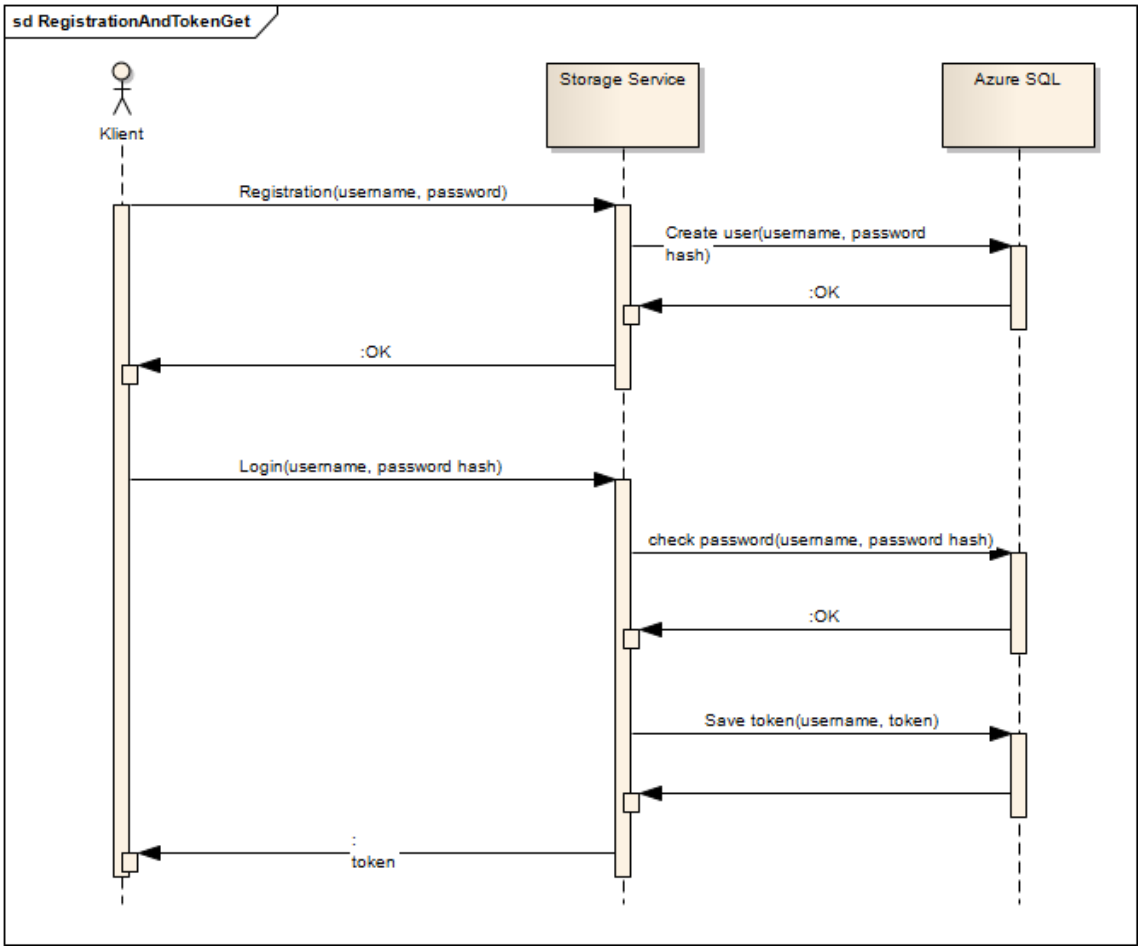
zprávy v Azure Queues. U této služby se také očekává její velké vytížení a časová, paměťová i výpočetní náročnost.

V cloudovém prostředí Windows Azure je přítomna také služba *Web GUI application*, která tvoří serverovou stranu celého webového klienta. Služba *Web GUI application* je nasazena jako Web Role a běží v poskytovaném webovém serveru IIS, protože obsluhuje požadavky spojené s komunikací internetového prohlížeče u koncového uživatel, které následně ve formě klienta přeposílá pomocí WCF rozhraní na službu *Storage Service*. Více o webovém rozhraní v bakalářské práci Petra Messnera.

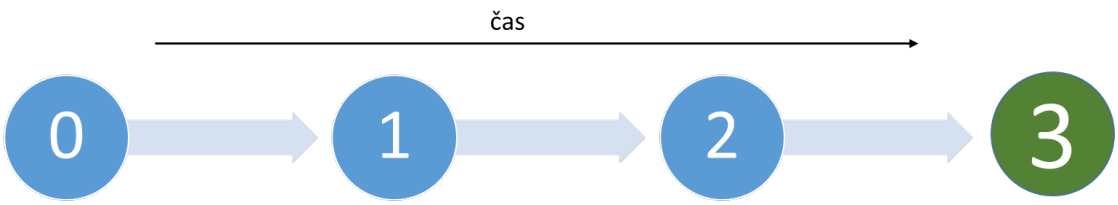
Na obrázku 3.10 je zobrazen schéma nasazení a typ jednotlivých služeb typu Cloud Services a jejich vzájemná komunikace.



Obrázek 3.1: Navržená třívrstvá architektura systému

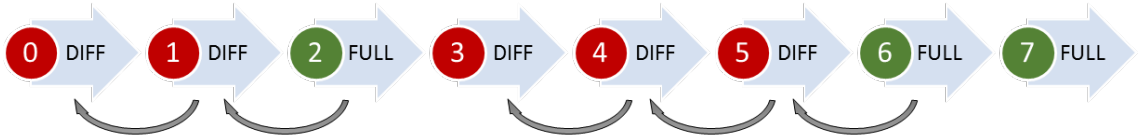


Obrázek 3.2: Diagram volání při registraci a následné získání autorizačního tokenu pro požadavky pomocí přihlášení

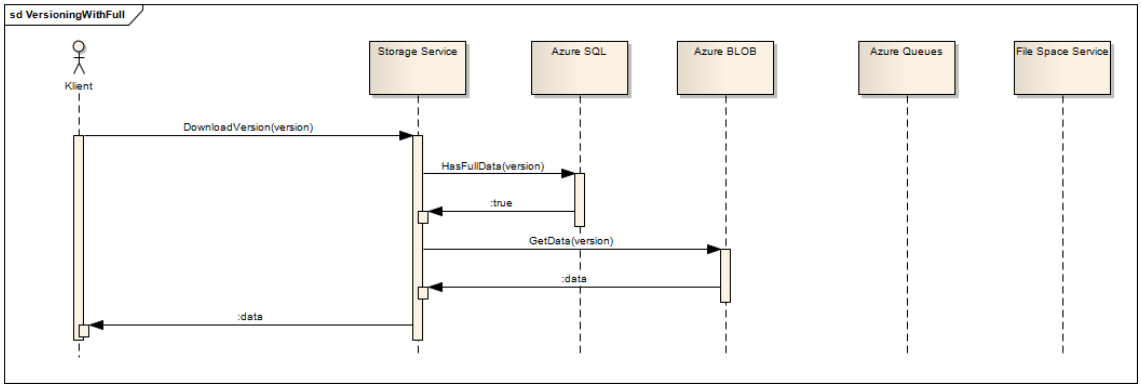


Obrázek 3.3: Příklad řady verzí jedné entity (souboru nebo složky) a vyznačená aktuální verze

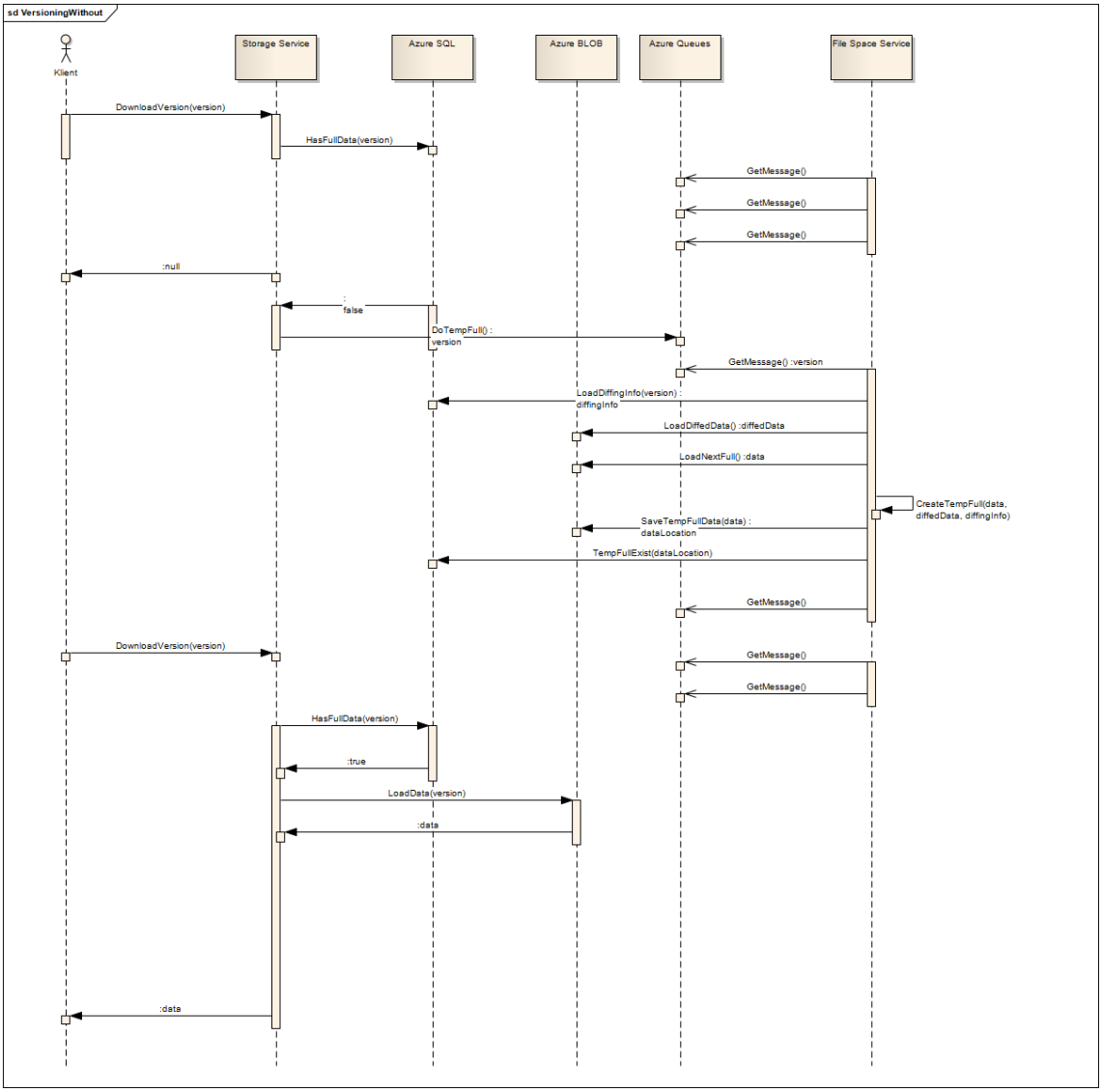
Obrázek 3.4: Příklad řady verzí jednoho souboru před aplikací minimalizačního procesu. Všechny verze souboru obsahují plné verze dat uložených na serveru. Verze 7 je poslední nahraná verze souboru.



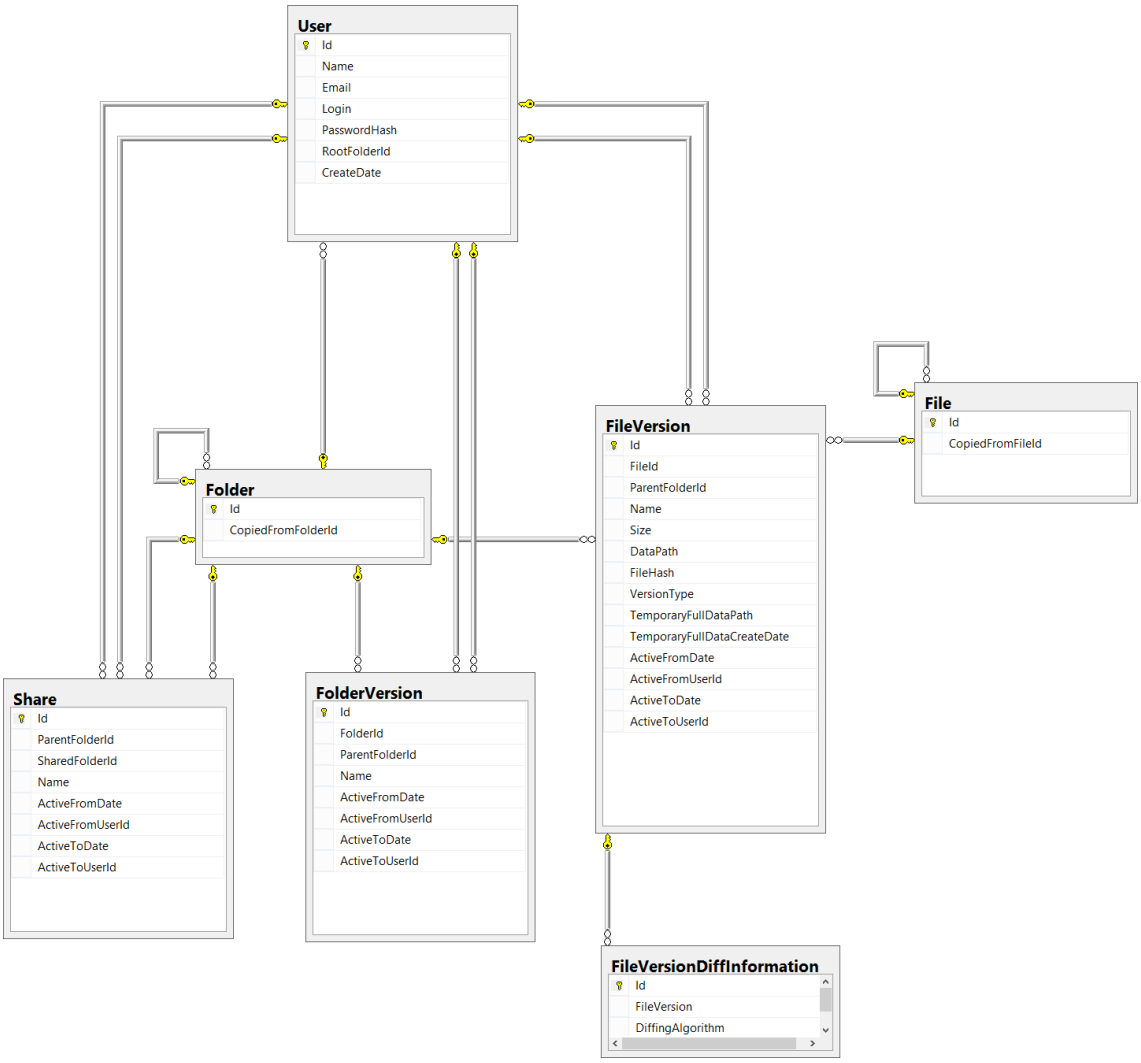
Obrázek 3.5: Příklad aplikace minimalizačního procesu na původní řadu souboru s konfigurací pravidel následující [Počet posledních plných verzí souboru: 2, Počet verzí po kterých je ponechána FULL verze: 3(směrem od poslední verze)] černé šipky značí verzi ze které lze dopočítat plnou verzi dat



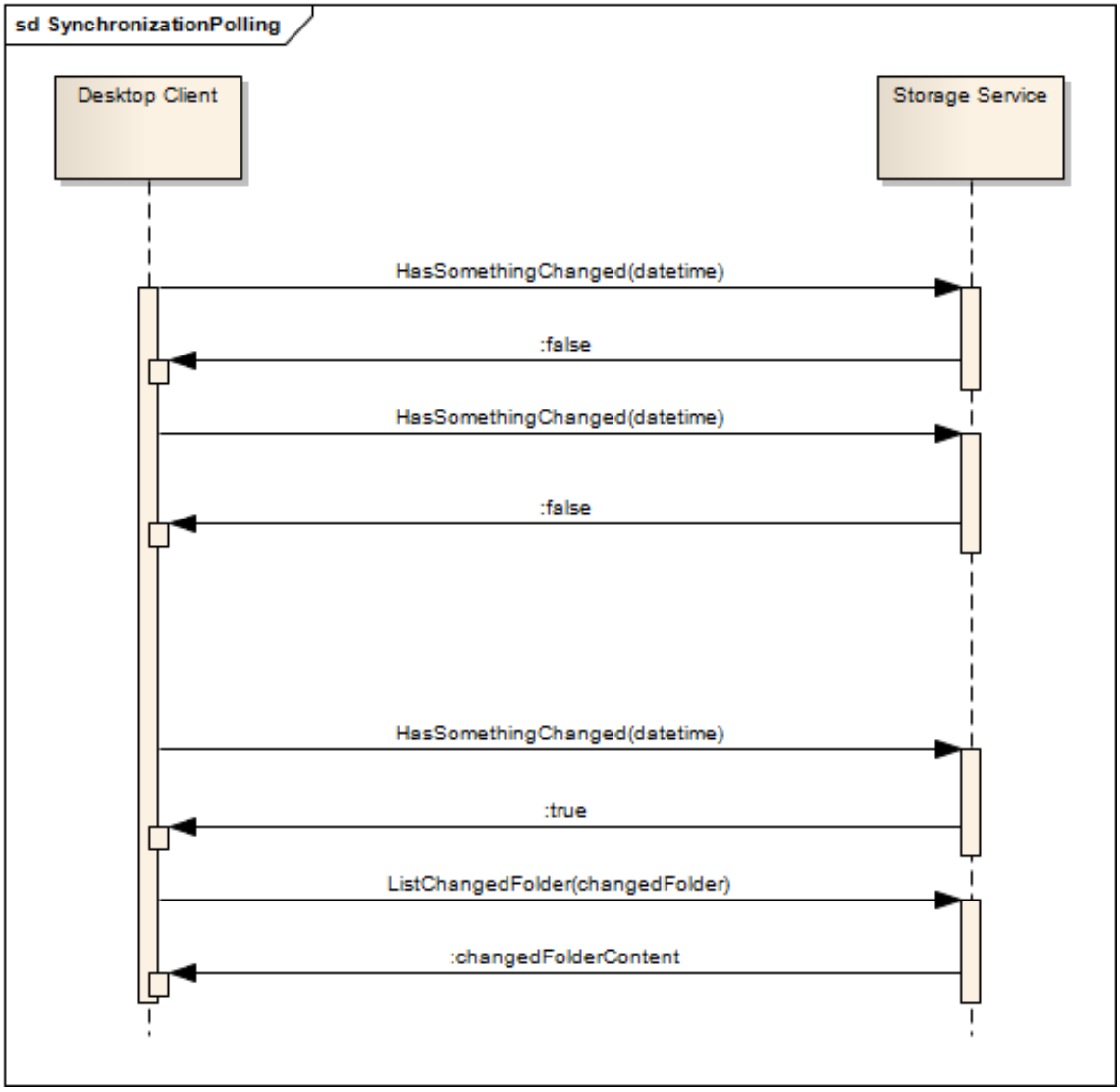
Obrázek 3.6: Sekvenční diagram volání v případě, kdy klient vyžaduje plná data, která jsou k dispozici okamžitě a jsou mu ihned zprostředkována službou *Storage Space*



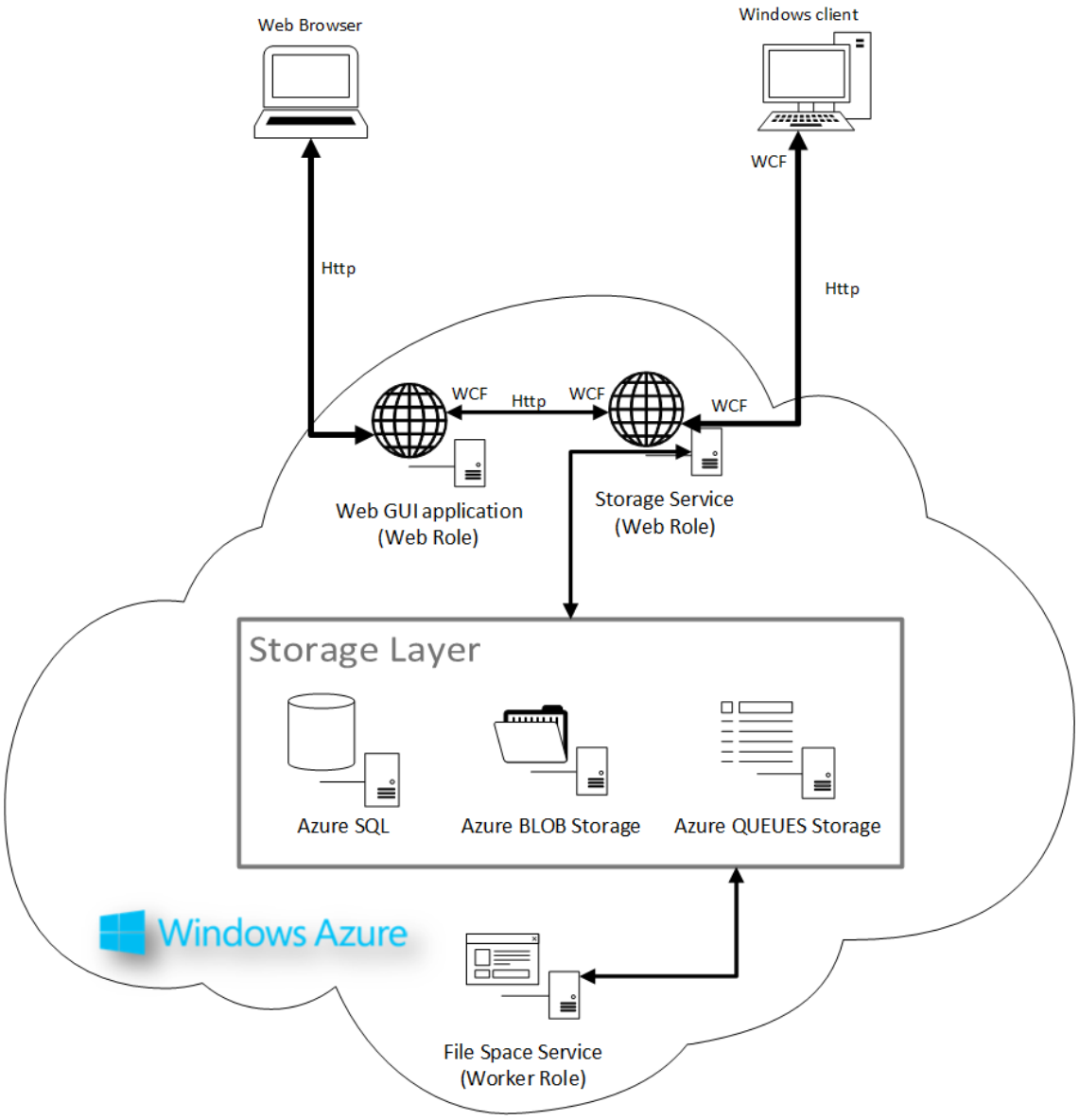
Obrázek 3.7: Sekvenční diagram volání v případě kdy klient vyžaduje data pro verzi, která je v minimalizované formě. Diagram zobrazuje zaznamenání požadavku do fronty Azure Queues, monitoring fronty, dopočítání dočasně plných dat a zápis do dočasně plných dat do Azure BLOB. Při následujícím požadavku klienta na tuto verzi jsou data předána klientovi



Obrázek 3.8: Diagram modelu relační databáze Azure SQL



Obrázek 3.9: Sekvenční diagram pollingu změn klientem a následné získání obsahu změněného adresáře na serverové straně



Obrázek 3.10: Diagram nasazení rolí v cloudovém prostředí a jejich vzájemná komunikace

Kapitola 4

Realizace

4.1 Platforma

Protože bylo vybráno cloudové prostředí Windows Azure poskytované společností Microsoft, byl za jazyk implementace vybrán C♯ společně s .NET frameworkem. Jazyk C♯ je vyspělým objektově orientovaným jazykem, pro který je dostupné příjemné vývojové prostředí Microsoft Visual Studio 2012. Tato kombinace má v prostředí Windows Azure rozsáhlou podporu například v automatizovaném nasazování služeb přímo z vývojového prostředí, nebo například vzdálené debugování některých operací přímo v ekosystému Windows Azure.

4.2 Software for development

Vývoj celého systému probíhal na platformě Microsoft Windows 8 Pro, který obsahoval kompletní sadu aplikací a nástrojů pro vývoj celého systému:

- Microsoft .NET Framework ve verzi 4.5
- Microsoft SQL Service 2012 Developer edice
- Microsoft Visual Studio 2012 Ultimate
- Microsoft Azure SDK ve verzi 1.8
- IIS ve verzi 8

4.3 Rozdělení kódu do komponent

Z důvodů deduplikace kódu byla vytvořena Dynamic Linked Library (dále jen dll), která obsahuje veškeré objekty spojené se vzájemnou komunikací mezi serverovou a klientskou stranou. Tato knihovna je veřejně uvolněna formou API zaštiťující veškerou komunikaci se serverovou stranou. V případě potřeby je možné kompletní implementaci této knihovny vygenerovat z vystaveného rozhraní služby *Storage Service*, které je v jazyce WSDL, pomocí

nástroje *svcutil.exe*, dodávaného jako součást .NET frameworku, případně alternativy pro jiný jazyk.

Dále byl kód rozdělen do několika dalších komponent tvořených DLL knihovnami. Široké použití *IOC Container* také umožňuje přidání komponent a jejich následného začlenení do celého procesu zpracování souborů i bez přístupu ke zdrojovému kódu celého systému. Protože však IOC eliminuje možnosti statické analýzy kódu a závislosti jednotlivých komponent jsou určovány až po samotném spuštění kódu, není možné určit přesný diagram závislostí. Navíc se jednotlivé závislosti komponent mohou měnit na základě uživatelské konfigurace bez rekompilace aplikace. Komponenty jsou vytvořeny podle logické návaznosti funkcí, které obsahují, což zároveň rozděluje kód do jednotlivých vsrtev. Tyto vrstvy spolu komunikují přímo, ale jejich závislost je určena konfigurací *IOC Containeru*.

Vytvořené komponenty a stručný popis jejich obsahu:

- NGMsBox.Core - obsahuje funkce tvořící logiku celého systému
- NGMsBox.DataContracts - obsahuje komunikační kontrakty, po kterých probíhá komunikace se *Storage Service*. Tato komponenta je také zpřístupněna klientům, případně její kód může být vygenerován z rozhraní vystaveného ve WSDL. Při implementaci této komponenty bylo také dbáno na možnost jejího použití pro implementaci podmnožinou .NET frameworku, která je dostupná na platformě Windows Store.
- NGMsBox.Entities - obsahuje funkce pro kompletní přístup k vrstvě *Storage layer*
- NGMsBox.Utils - obsahuje pomocné funkce, které mohou být zpřístupněny klientům, jako například způsob generování otisku hesla pro přesnos apod.
- NGMsBox.FileSpace.Core - obsahuje funkce tvořící logiku celého systému pro minimalizaci ukládaných dat, včetně pravidlel pro tvorbu verzovací politiky.
- NGMsBox.FileSpace.Diffing - obsahuje algoritmy pro počítání verzí jednotlivých souborů

4.4 Storage Service

Služba využívá *IOC Containeru Windsor Castle* a její závislosti jsou injektovány až ve chvíli běhu programu na základě konfigurace, která zároveň nastavuje připojení a parametry pro komunikaci se *Storage layer*.

Ukázka konfigurace *IOC containeru*, na které jsou vidět konfigurační parametry a definice jedné třídy v containeru:

```
<castle>
  <properties>
<!--connection strings here-->
    <defaultConnectionString>
     Server=localhost;Database=NGMsBoxDB;User Id=admin;Password=admin;
    </defaultConnectionString>
```

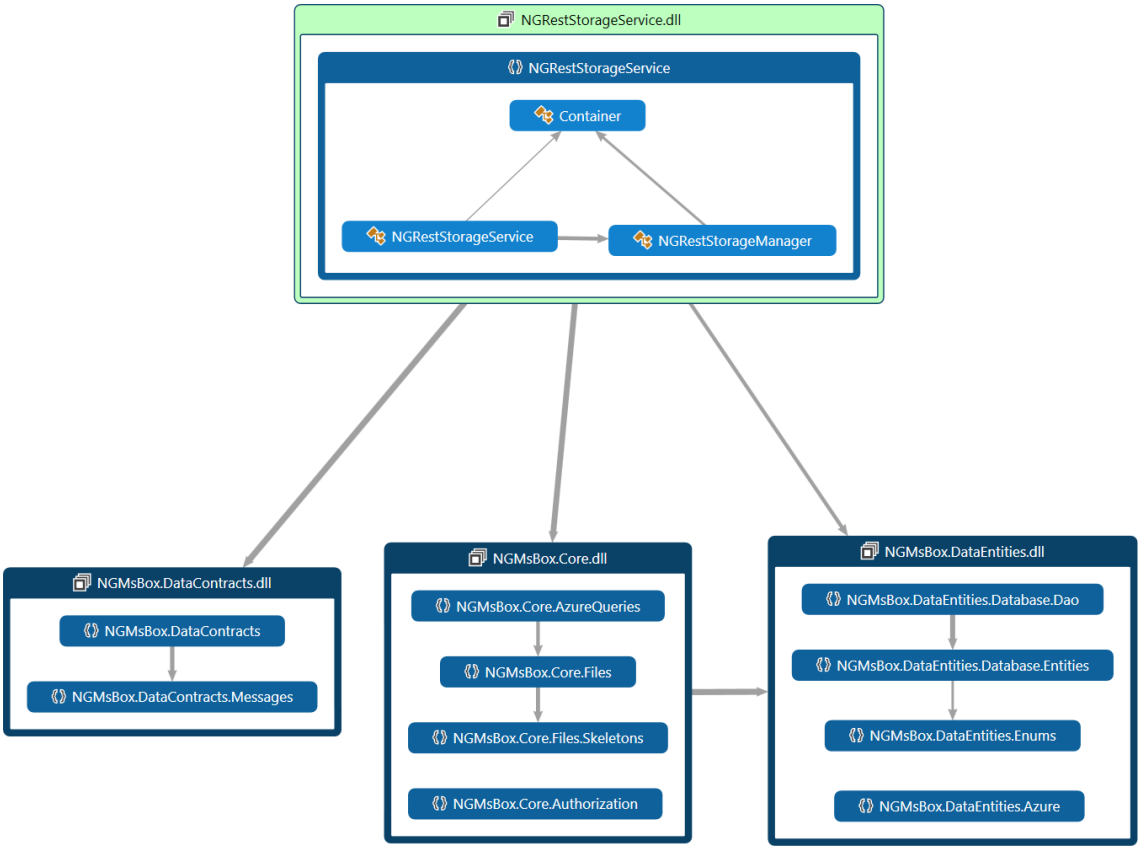
```
<!--AZURE QUEUE NAMES-->
    <newFileQueueName>newfilequeue</newFileQueueName>
    <tempFullQueueName>tempfullrequestqueue</tempFullQueueName>
  </properties>

  <components>
    <component id="NGRestStorageManager"
               type=" NGRestStorageService.NGRestStorageManager,
               NGRestStorageService"/>

   </components>
</castle>
```

Parametry pro nastavení jednotlivých komponent pak mohou být použity v konfiguraci úplně jiné komponenty. Tímto způsobem je zaručeno znovupoužití jednotlivých komponent ve více službách při zachování konfigurovatelnosti pro každou službu zvlášť. Ukázka použití prametrů v konfiguraci jiné komponenty:

```
    ...
    <component id="AzureFileQueueClient"
               type="NGMsBox.Core.AzureQueries.AzureFileQueueClient,
               NGMsBox.Core">
      <parameters>
        <azureConnectionStringName>
         #{azureConnectionStringName}
        </azureConnectionStringName>
        <newFileQueueName>
         #{newFileQueueName}
        </newFileQueueName>
        <tempFullQueueName>
         #{tempFullQueueName}
        </tempFullQueueName>
      </parameters>
    </component>
    ...
```

Služba využívá logovacího frameworku *Apache log4net*, který je velmi silně konfigurovatelný. Ukládání logů z této služby je nastaveno do souborů, které jsou vytvářeny na serveru, na němž je služba nasazena. Není však problém logování změnit tak, aby byly logované záznamy zasílány například emailem na libovolnou adresu. Při změně konfigurace není vyžadována rekompilace služby.

Na obrázku 4.1 je zobrazeno využití jednotlivých komponent ve službě *Storage Space*. Služba samotná obsahuje pouze Container a implementaci komunikačního rozhraní, jehož funkčnost je ale již rozdělena do jiných komponent. Jednotlivé vztahy mezi komponentami jsou řízeny objektem Container, který je implementací IOC Containeru Windsor Castle.



Obrázek 4.1: Diagram využití komponent ve službě *Storage Space* a částečný výpis obsahu komponent.

4.5 File Space Service

Tato služba také využívá IOC Containeru, což umožnilo implementovat vysokou míru konfigurovatelnosti služby bez rekompilace aplikace. Jedná se zejména o dynamické přidávání pravidel pro verzovací politiku a jejich vzájemnou spolupráci. Za spuštění verzovacích pravidel je zodpovědná jedna třída, která však nemá přímé závislosti na implementacích jednotlivých pravidel. Tato třída si při startu aplikace projde všechny assembly, které jsou jí předány, a hledá implementace jednotlivých pravidel, ze kterých si následně vytvoří instance a při požadavku zajistí spuštění všech pravidel podle jejich nakonfigurovaných priorit. Konfigurace jednotlivých pravidel je umožněna pomocí konfigurace IOC Containeru. Tímto procesem je zajištěna maximální míra abstrakce a je dosaženo snadného dynamického přidávání pravidel například pomocí dodání další DLL knihovny. Obdobným procesem jsou přidávány algoritmy pro počítání rozdílových souborů, takže při implementaci dalšího algoritmu není nutné kompilovat celou aplikaci. Služba využívá logování ve formě *Apache log4net*, která má stejné možnosti jako popsané u služby *Storage Service* - viz kapitola 4.4.

Diagram na obrázku 4.2 zobrazuje využití jednotlivých komponent, ze kterého je zřejmé použití stejné komponenty pro přístup k *Storage layer* jako ve službě *Storage Service*. Kon-

figurace komponenty pro přístup může být však odlišná.

Ukázka konfigurace jednotlivých pravidel:

```
...
<!--Version rules-->
 <component id="RuleLastVersionIsFull"
           type="NGMsBox.FileSpace.Core.Vesioning.RuleLastVersionIsFull,
           NGMsBox.FileSpace.Core"
           service="NGMsBox.FileSpace.Core.Vesioning.IVersionRule,
           NGMsBox.FileSpace.Core">
  <parameters>
    <lastFullVersionsCount>1</lastFullVersionsCount>
    <priority>1</priority>
  </parameters>
</component>

<component id="RuleFullVersionAfter"
           type="NGMsBox.FileSpace.Core.Vesioning.RuleFullVersionAfter,
           NGMsBox.FileSpace.Core"
           service="NGMsBox.FileSpace.Core.Vesioning.IVersionRule,
           NGMsBox.FileSpace.Core">
  <parameters>
    <fullVersionEach>3</fullVersionEach>
    <priority>2</priority>
  </parameters>
</component>
...
```

4.6 Windows desktop klient

Při implementaci desktopového klienta bylo důkladně zváženo použití IOC Containeru, protože zde by mohl zbytečně brzdit start aplikace. Protože ale desktopový klient neobsahuje velké množství komponent, je zdržení při startu aplikace zanedbatelné. Nakonec byl tedy Windsor Castle použit na implementaci IOC containeru také. Desktopový klient obsahuje pouze jedno okno, které slouží pro nastavení přihlášení uživatele (případně pro registraci) a pro nastavení monitorované složky. Uživateli není umožněno přes klienta stáhnout starší verze souboru a pro tuto operaci doporučuji webového klienta vytvořeného a popsaného v bakalářské práci Petra Messnera. Zároveň desktopový klient řešeší veškeré konflikty formou *Last-one-wins*, protože oblast řešení konfliktů je také součástí bakalářské práce Petra Messnera. Změny souborového systému jsou monitorovány pomocí low level zpráv o operacích se souborovým systémem, zaštítěné komponentou FileSystemWatcher. Tato komponenta má však některá omezení [15]:

- Hlídání souborového systému NTFS - plná podpora

- Hlídání souborového systému FAT32 - částečná podpora, fungující na hlavním lokálním disku počítače. Není možné hlídat připojená úložiště (přenosný flash disk, paměťové karty, mobilní telefony, fotoaparáty apod.)
- Hlídání vzdáleného souborového systému připojeného pomocí protokolu SAMBA - omezená funkčnost
- Minimální verze operační systém Microsoft Windows 98
- Hlídání vzdáleného souborového systému jiného typu - není vyzkoušeno
- Hlídání souborového systému jiného typu - není vyzkoušeno

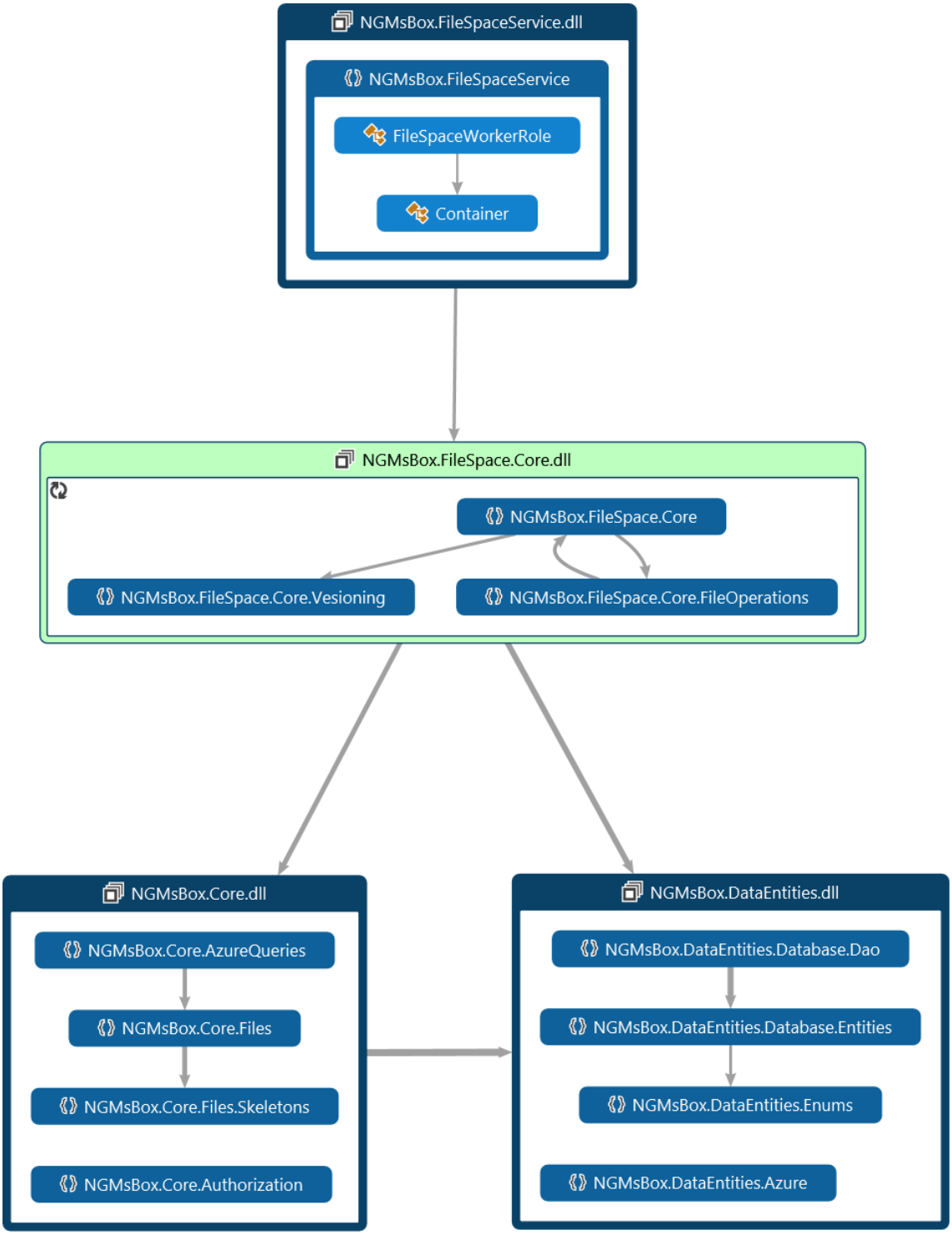
Použití komponenty FileSystemWatcher na hlídání vzdáleného souborového systému připojeného pomocí protokolu SAMBA by však bylo možné, za předpokladu úpravy kódu a konfigurací. I přesto by však byla omezena funkčnost. Desktopový klient tedy funguje bez problémů na souborových systémech NTFS a případně FAT32 (za splnění druhé omezující podmínky).

Mezi typy zpráv, které je FileSystemWatcher shopen zpracovávat, patří především:

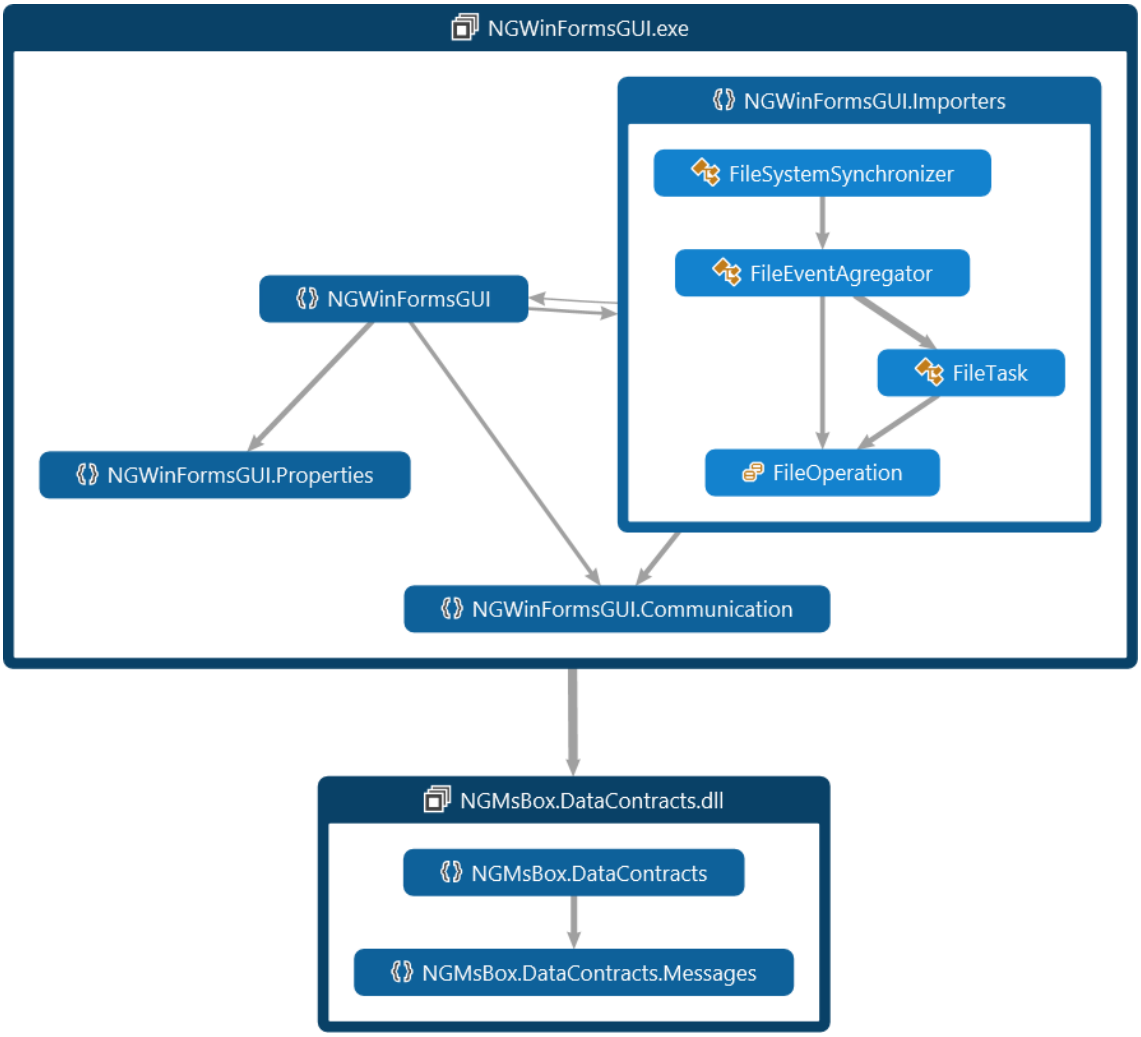
- Změna souboru
- Přejmenování souboru nebo složky
- Smazání souboru nebo složky
- Změny různých atributů u souborů a složek
- Změny posledních přístupů u souborů a složek
- Další informace

Komponenta FileSystemWatcher však dodává low level zprávy přístupu k souborovému systému a je tedy možné, že se některé zprávy vyskytnou vícekrát. Počet zpráv není předem dán, protože závisí na konkrétní implementaci programu, který změnu provádí. Například při ukládání souborů v programu notepad.exe, který je součástí Windows, je vyvolána sekvence zpráv: CREATED - CHANGED. Při provedení stejné operace v programu pomocí Visual Studia 2012 je vytvořena. sekvence zpráv následující: CREATED - CHANGED - CHANGED - CHANGED - CHANGED Zprávy je tedy nutné agregovat do logických celků.

Schéma na obrázku 4.3 zobrazuje vzájemné využití komponent v implementaci desktopového klienta. Pro komunikaci je použita sdílená knihovna obsahující komunikační rozhraní se službou *Storage Service*.



Obrázek 4.2: Diagram využití komponent ve službě *File Space Service* a částečný výpis obsahu komponent.



Obrázek 4.3: Diagram využití komponent ve desktopovém klientu pro Windows a částečný výpis obsahu komponent.

Kapitola 5

Testování

Protože při manipulaci se soubory uživatelů je velmi nutná spolehlivost celého řešení musela být celá složba důkladně otestována. Testování systému nebylo jednoduše technologicky proveditelné, protože pouze málo serverových operací nevyžaduje ke svému běhu služby z *Storage layer*, které nejsou snadno přístupné bez nasazení do cloudového prostředí. Pro odladění celého systému při vývoji bylo zavedené logování ve všech komponentách pomocí *Apache log4net*, kterým je možné zaznamenat nestandartní chování průběhu všech operací.

Z důvodů spolehlivosti bylo rozhodnuto otestovat celou serverovou stranu systémem black box, která je založena na testování jednotlivých požadavků na server a navrácených odpovědí bez znalosti vnitřního fungování serveru. Testování tímto způsobem může být velice snadno zautomatizování pomocí periodického spouštění testů. Cloudová tehcnologie Windows Azure poskytuje službu pro periodické spouštění úkolů jménem *Azure Scheduler*. Pomocí této služby je možné spustit sadu příkazů, skript nebo například program. Z tohoto důvodu bylo rozhodnuto vytvoření jakohosi testovacího mikroframeworku, který bude testovat jednotlivé operace formou blackboxu.

5.1 Storage service testers

Testování bude probíhat pomocí spouštění konzolové aplikace pro každou atomickou operace potřebnou k otestování na serverové straně. Byla vytvořena sada programů, které mohou být spuštěny pomocí skriptu, nebo z příkazové řádky, čehož by mohlo být využito pro automatizaci testování už nasazeného systému v prostředí Windows Azure. Proces testování operací je maximálně automatizovaný tak, aby vyžadoval co nejmenší interakci od uživatele. Jsou testovány jak operace s uživately, tak operace se soubory a složkami. Při testování operací se souborovým systémem je však nutné vytvořit uživatele, pod kterým jsou operace testovány. Před samotným testem operace se soubory je vytvořen unikátní uživatel, který je následně přihlášen a pod kterým probíhají testované operace. Testování jednotlivých operací má navzájem inkrementální závislosti. Například při testování operace nahrání souboru je předpokládáno, že fungují operace pro přihlášení, vytvoření složky a vypsání složky. Všechny tyto předpoklady mají však samostatné testovací programy, které je možné spustu a čímž je tedy zaručená otestovanost systému od menších celků až po větší operace jako je nahrání nové verze souboru apod. Ukázka testování korektního stažení souboru:

```
var fileId = TestUploadFile(token);

    using (var content = File.OpenRead("TestFile.txt"))
    {
        using (var remoteContent =
         m_manager.DownloadFile(token.AuthToken, fileId).FileData)
        {
            return StreamEquals(content, remoteContent);
        }
    }
```

Test nejdříve předpokládá, že operace nahrání souboru je korektně provedena a následně provede stažení souboru ze serveru a porovná obsahy obou souborů a navrátí hodnotu porovnání. Obdobným způsobem jsou implementovány ostatní blackbox testy serveru.

5.2 Testované prostředí

Systém byl otestován na vývojovém prostřední pro platformu Windows Azure, které poskytuje emulaci úložných služeb Azure BLOB, Azure Tables a Azure Queues. Zároveň je poskytován emulátor spouštění instancí jednotlivých Cloud Services. Podporována je emulace Web Rolí i Worker Rolí. Pro zastoupení relační databáze Azure SQL byl použit vývojový server Microsoft SQL Server 2012, který však umožňuje použití některých funkcí navíc oproti poskytovanému Azure SQL. Kontrola na neexistenci použití těchto funkcí probíhala formou kontroly dokumentace při vývoji a při návrhu řešení.

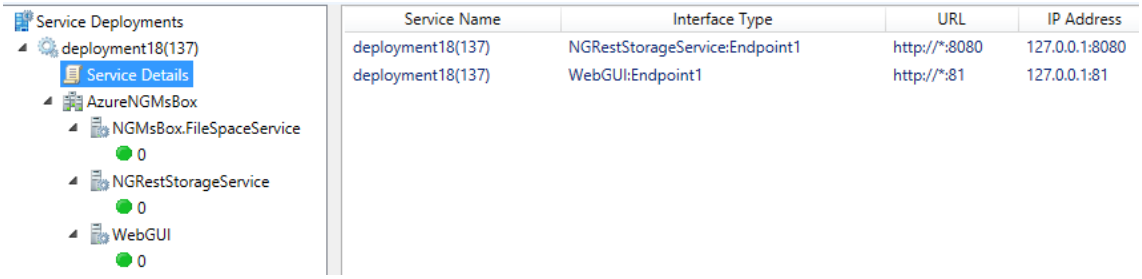
Emulátor běžel na vývojovém počítači, který měl kromě vývojových nástrojů popsaných v 4.2 nainstalován i další programy pro běžné uživatelské užívání. Technická výbava počítače je pak následující:

- Procesor: Core i7 930 4 jádra (8 virtuálních vláken) o frekvenci 2.66 GHz s plnou podporou virtualizace
- Paměť RAM: 18 GB DDR3 paměti
- Grafická karta: 2x AMD Radeon HD 6970 s plnou podporou akcelerace vykreslování grafického prostředí
- Operační systém Windows 8 Pro.

Na obrázku 5.1 je zobrazeno nasazení realizovaného systému do emulátoru Azure Cloud Services a naslouchání na portech emulované instance.

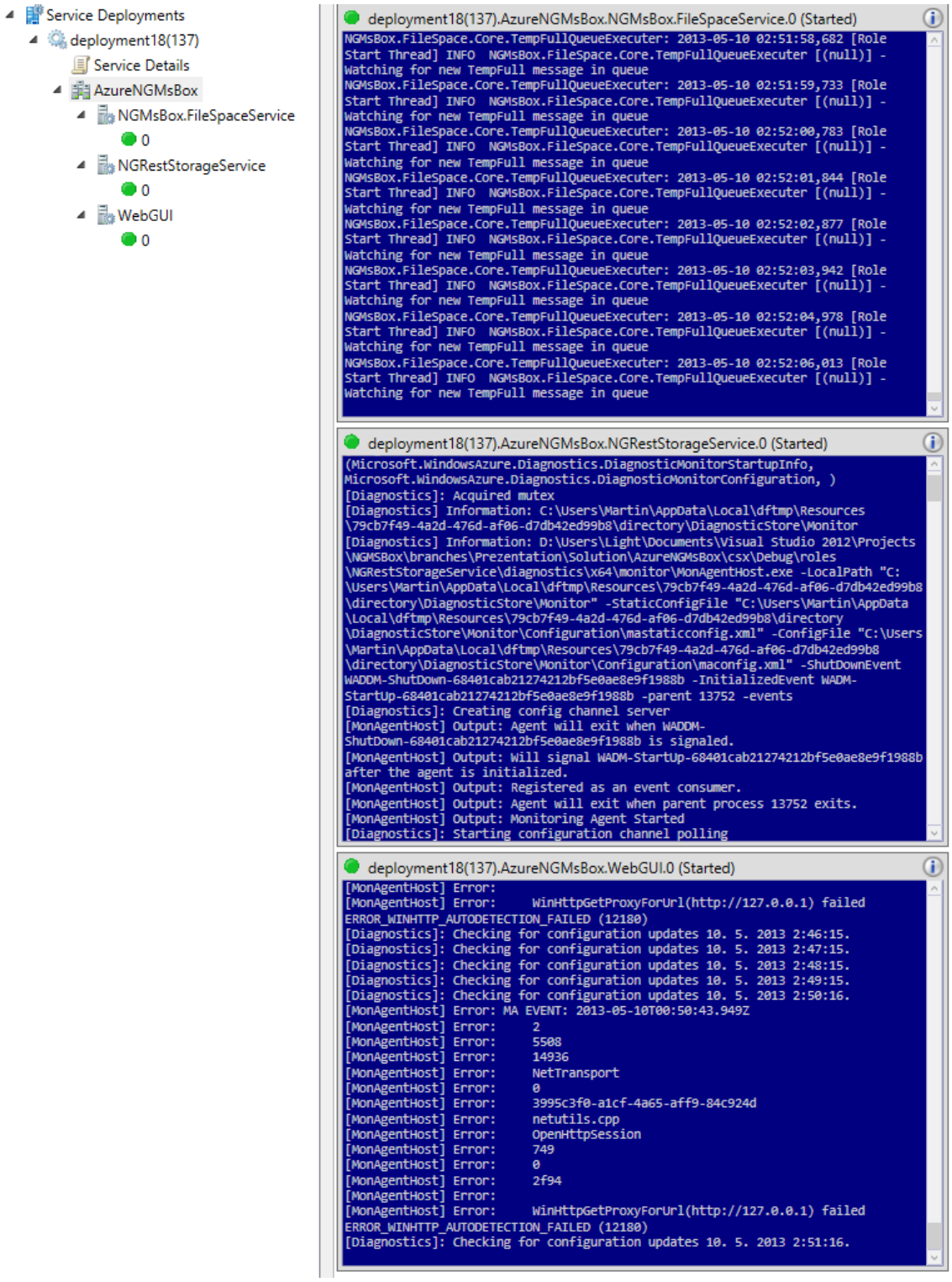
Obrázek 5.2 pak zobrazuje nasazení všech rolí s jednou instancí do emulátoru Azure Cloud Services. Emulátor poskytuje aplikaci pro prohlížení trace logování jednotlivých instancí. Instance *File Space Service* má nastavný logovací framework tak, aby se logování zobrazovalo i v tomto nástroji a na obrázku je vidět monitorání úložiště Azure Queues pomocí periodického dotazování.

Desktopový klient byl testován na stejném počítači, na kterém běžel emulátor s nasazenou serverovou částí systému. Z důvodů důkladnějšího otestování serverové strany celého systému



Obrázek 5.1: Ukázka nasazení systému do emulátoru a poslech na portech v emulovaném instanci virtualizovaného prostředí Azure Cloud Services. Zobrazeny obě dvě Web Role (*Storage Space* i webový klient *Web GUI application*)

nebylo věnováno testování desktopového klienta tolik času a desktopový klient prošel pouze uživatelským testováním. Při psaní dokumentační části diplomové práce běžel desktopový program a byl nasměrován na pozorování složky s obrázky pro dokumentační části, čímž byl uživatelsky otestován pro použití v reálném prostředí.



Obrázek 5.2: Ukázka nasazení všech Cloud Services do emulovaného prostředí windows azure. Nasazeno je po jedné instanci od každé navržené Azure Cloud Service role.

Kapitola 6

Záběry obrazovek

Následující kapitola obsahuje sled záběrů obrazovek pořízených pomocí vývojové verze webového klienta, který je vyvíjen v rámci bakalářské práce Petrem Messnerem. Zobrazení z webového klienta byl zvole z důvodů vyšší názornosti než sobrazení souborů v monitorované složce desktopovým klientem. Obrázek 6.1 ukazuje seznam souborů we webovém klientovi. Soubory byly nahrány na server pomocí dektopového klienta.

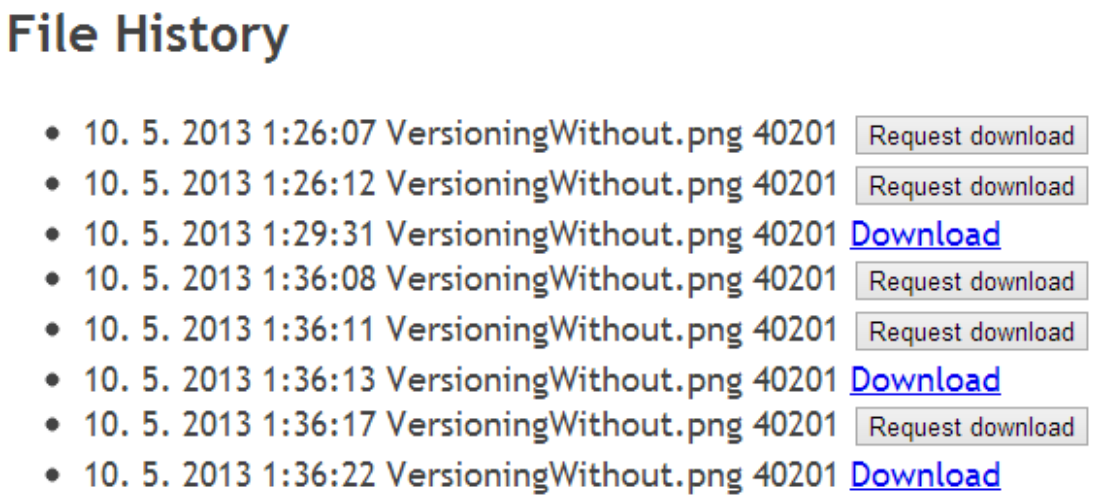
Upload New File | Create Folder | Delete | Move | Copy

Name	Size	Owner	Last Modified	
OldWorkDistrib.png	98869	test	10. 5. 2013 1:29:32	Show history Upload new version
OldArchitecture.png	105170	test	10. 5. 2013 1:29:32	Show history Upload new version
NewWorkDistrib.png	77443	test	10. 5. 2013 1:29:32	Show history Upload new version
NewArchitectureLayers.png	40848	test	10. 5. 2013 1:29:32	Show history Upload new version
NewArchitecture.png	73964	test	10. 5. 2013 1:29:32	Show history Upload new version
LogoCVUT.pdf	5114	test	10. 5. 2013 1:29:32	Show history Upload new version
LogoCVUT.eps	30138	test	10. 5. 2013 1:29:33	Show history Upload new version
FSScomponents.png	93386	test	10. 5. 2013 1:29:33	Show history Upload new version
DatabaseModel.png	54357	test	10. 5. 2013 1:29:33	Show history Upload new version
CurrentCodeNullReference.png	6278	test	10. 5. 2013 1:29:33	Show history Upload new version

Obrázek 6.1: Ukázka seznamu souborů nahraných pomocí desktopového klienta na server a jejich zobrazení ve webovém klientovi.

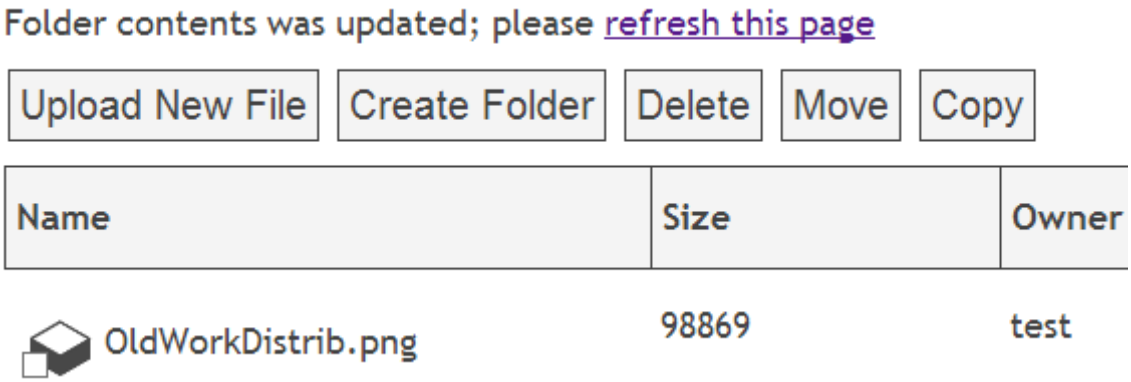
Obrázek 6.2 ukazuje reálné provedení aplikace minimalizační politiky na verzované soubory nad reálnými daty. Na obrázku jsou vidět jednotlivé verze souboru dostupné na serveru, ale pouze některé z nich mají data dostupná ihned. Na ostatní verze je aplikován proces vy-

žádání dočasně plné verze dat. Veškeré verze souboru byly nahrány na server automaticky desktopovým klientem při detekci změn dat v souboru



Obrázek 6.2: Ukázka výsledku verzovací politiky pro reálný soubor. Screenshot je pořízen z webového klienta, který je vyvíjen pro systém v rámci bakalářské práce Petrem Messnerem.

Obrázek 6.3 ukazuje vývojovou verzi detekce změn souborů na serverů pomocí technologie polling. Soubory byly v tomto případě změněny pomocí desktopového klienta. Aktuální verze detekuje pouze změny na jednotlivých souborech, ale serverová strana umožňuje i možnosti detekce na úrovni složek, které využívá desktopový klient.



Obrázek 6.3: Ukázka vývojové verze automatické detekce změny adresáře při změně souborů pomocí jiného klienta.

Kapitola 7

Závěr

7.1 Splnění cílů

V průběhu řešení dané diplomové práce byl nastudován původní existující systém vytvořený Bc. Tomášem Budínem. Původní systém byl důkladně analyzován a byly v něm nalezeny některé nedostatky (převázně v realizační části), pro které byla navržena vylepšení. Byly provedeny úpravy jak na úrovni architektury systému, tak i jednotlivých služeb, včetně obsluhy klientů a samotných operací ukládání dat na serverové straně. Počet vyhodnocených změn vyústil v reimplementaci celého systému, který byl důkladně otestován. Vytvořením nové serverové strany bylo dosaženo univerzálního systému pro zálohování a archivaci souborů v cloudovém prostředí Windows Azure. Upravená architektura umožňuje systému těžit z výhod daných možnostmi poskytované škálovatelnosti služeb na platformě Windows Azure. Nová realizace systému umožňuje dlouhodobou archivaci dat s funkcionalitou verzování entit na úrovni jednotlivých souborů a složek. Pro jednotlivé verze byla navržena nová jednoduše upravitelná a nastavitelná politika pro minimalizaci uložených dat na serveru při zachování jejich dostupnosti a možnosti získání původních dat uživateli. Součástí systému je i nově implementovaná služba, která navrženou politku aplikuje na jednotlivá data. Tím je dosaženo snížení objemu uložených dat. K systému byl také vytvořen nově implementovaný klient pro platformu Windows desktop. Nový systém umožňuje nahrávání uživatelských dat do systému bez potřeby uživatelské interakce.

7.2 Návrhy pro další možná rozšíření systému

Ačkoli v systému byla navržena, implementována a otestována celá řada vylepšení, v průběhu práce bylo odhaleno ještě několik možností pro rozšíření systému o nové funkcionality. Jedná se například o:

- Minimalizaci uložených dat pomocí deduplikace na serverové straně. Jde však o funkci velmi náročnou na výpočetní prostředky a její nasazení může značně komplikovat současné použití minimalizační politiky pro počítání rozdílových dat.

- Klienty pro mobilní platformy nebo uzpůsobení webového klienta pro použití na mobilních telefonech.
- Centralizovanou správu nastavení jednotlivých služeb a sběr souborů s chybovými hláškami z jednotlivých služeb a jejich následné zpracování.
- Centralizovanou správu uživatelských účtů na straně klienta.

Literatura

[1] Andreas John. *Binary diff/patch utility* [online]. 2013. [cit. 6. 5. 2013].

[2] Brian Voo(HongKait.com). *8 Tips To Get The Most Out Of Microsoft SkyDrive* [online]. 2013. [cit. 6. 5. 2013]. Dostupné z: <http://www.hongkiat.com/blog/ms-skydrive-tips/>.

[3] BUDíN, B. T. Backup system capable of data sharing using Microsoft Azure cloud technology. Master's thesis, Czech Technical University in Prague, Faculty of Electrical Engineering, 2012.

[4] Colin Percival. *Binary diff/patch utility* [online]. 2013. [cit. 6. 5. 2013].

[5] Dropbox. *Dropbox install.* https://www.dropbox.com/install, 2013.

[6] Dropbox. *How do I share folders with other people?* https://www.dropbox.com/help/19/en, 2013.

[7] Eric Genesky(DZone.com). *Supported Virtual Machine Operating Systems in the Microsoft Cloud* [online]. 2013. [cit. 6. 5. 2013].

[8] Google. *Google Drive help* [online]. 2013. [cit. 6. 5. 2013]. Dostupné z: <http://support.google.com/drive/bin/answer.py?hl=en&answer=2409045>.

[9] Jiří Macich ml.(lupa.cz). *Google Drive help* [online]. 2013. [cit. 6. 5. 2013]. Dostupné z: <http://www.lupa.cz/clanky/microsoft-skydrive-nebo-google-drive-kam-je-lepsi-ukladat-data/>.

[10] Microsoft. *Windows Azure Queues and Windows Azure Service Bus Queues - Compared and Contrasted* [online]. 2013. [cit. 6. 5. 2013].

[11] Microsoft. *Pricing at-a-glanced* [online]. 2013. [cit. 6. 5. 2013].

[12] Microsoft. *Windows Azure Storage Abstractions and their Scalability Targets* [online]. 2013. [cit. 6. 5. 2013].

[13] Microsoft. *Windows Azure for Enterprises* [online]. 2013. [cit. 6. 5. 2013].

[14] Microsoft. *.NET Multi-Tier Application Using Storage Tables, Queues, and Blobs* [online]. 2013. [cit. 6. 5. 2013].

[15] Microsoft. *FileSystemWatcher does not work on FAT32 or Samba shares?* [online]. 2013. [cit. 6. 5. 2013].

[16] Microsoft. *File access and permissions in Windows Store apps.* Microsoft Corporation, http://msdn.microsoft.com/en-us/library/windows/apps/hh967755.aspx, 2013.

[17] Microsoft. *Microsoft Developer Network Online Documentation.* http://msdn.microsoft.com, 2013.

[18] Microsoft. *Data Transfers* [online]. 2013. [cit. 6. 5. 2013].

[19] Přispěvatelé Wikipedie. *Windows Azure* [online]. 2013. [cit. 6. 5. 2013]. Dostupné z: <http://en.wikipedia.org/wiki/Windows_Azure>.

[20] Přispěvatelé Wikipedie. *Castle Project* [online]. 2013. [cit. 6. 5. 2013]. Dostupné z: <http://en.wikipedia.org/wiki/Castle_Project>.

[21] Přispěvatelé Wikipedie. *.NET Framework* [online]. 2013. [cit. 6. 5. 2013]. Dostupné z: <http://en.wikipedia.org/wiki/.NET_Framework>.

[22] Přispěvatelé Wikipedie. *Google Drive* [online]. 2013. [cit. 6. 5. 2013]. Dostupné z: <http://en.wikipedia.org/wiki/Google_Drive>.

[23] Přispěvatelé Wikipedie. *Inversion of control* [online]. 2013. [cit. 6. 5. 2013]. Dostupné z: <http://en.wikipedia.org/wiki/Inversion_of_control>.

[24] Přispěvatelé Wikipedie. *Longest common subsequence problem* [online]. 2013. [cit. 6. 5. 2013]. Dostupné z: <http://en.wikipedia.org/wiki/Longest_common_subsequence_problem>.

[25] Přispěvatelé Wikipedie. *NHibernate* [online]. 2013. [cit. 6. 5. 2013]. Dostupné z: <http://en.wikipedia.org/wiki/NHibernate>.

[26] Přispěvatelé Wikipedie. *SkyDrive* [online]. 2013. [cit. 6. 5. 2013]. Dostupné z: <http://en.wikipedia.org/wiki/SkyDrive>.

[27] Přispěvatelé Wikipedie. *Third normal form* [online]. 2013. [cit. 6. 5. 2013]. Dostupné z: <http://en.wikipedia.org/wiki/Third_normal_form>.

[28] TROELSEN, A. *Pro C# 2010 and the .NET 4.0 Platform.* New York, NY, USA : Apress, 5th edition, 2009.

Příloha A

Seznam použitých zkratek

WCF Windows Communication Foundation

WSDL Web Services Description Language

PAAS Platform As A Service

IAAS Infrastructure As A Service

SAAS Software As A Service

IIS Internet Information Service

TSQL Transact SQL

SDK Software Development Kit

API Application programming interface

DLL Dynamic Linked Library

Příloha B

Instalační a uživatelská příručka

B.1 Otestování serveru v Azure Emulátoru:

K otestování celého systému je potřebné funkční visual studio verze 2012 a všechny další programy popsány v sekci 4.2. Před samotný spuštěním systému je nutné provést operaci vytvoření relační databáze. Pro vytvoření databáze je nutné provést následující kroky:

- Založit databázi s názvem „NGMsBoxDB“
- K databázi dát administrátorská práva uživateli s názvem admin a heslem admin.
- Spustit připravený skript pro vytvoření databáze, který vytvoří kompletní databázovou strukturu.

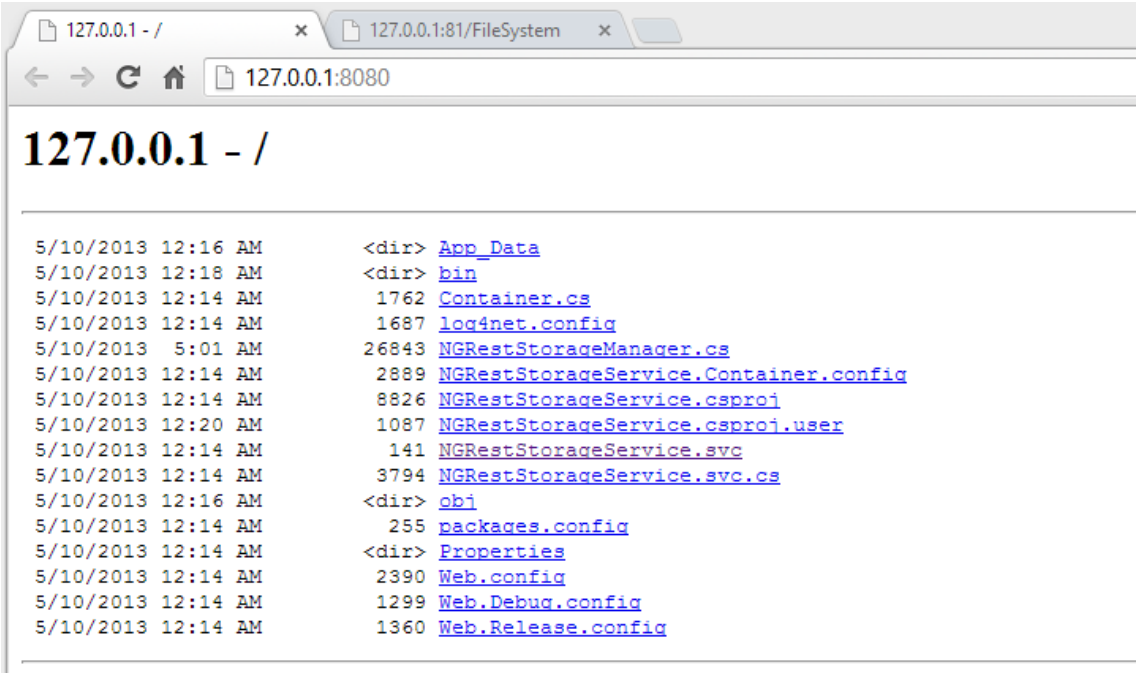
Visual Studio 2012 je nutné spustit jako administrátor. Tento krok je nutný pouze v případě, že celý systém chceme spustit ve Windows Azure emulátoru. V případě nasazování systému přímo na platformu Windows Azure není administrátorský mód Visual Studia požadován. Ve visual studiu nyní otevřeme hlaví solution soubor. V solution exploreru je nutné vybrat projekt s názvem „AzureNGMsBox“, který označíme z kontextového menu jako výchozí. Dalším krokem je již standartní spuštění projektu pomocí klávesy *F5*, nebo případně pomocí zelené šipky, což započne proces nasazování celého systému do emulátoru Windows Azure. Po skončení procesu nasazování jsou automaticky otevřena dvě okna ve webovém prohlížeči. V prvním okně je webový klient s přihlašovací obrazovkou. Ve druhém okně je pak rozhraní služby *Storage Service*. V tuto chvíli je serverová strana úspěšně nasazená a měla by být plně funkční.

B.2 Instalace a otestování desktopového klienta

Pro klienta byl vytvořen instalátor, pomocí kterého je možné klienta normálně nainstalovat do libovolné složky. Nyní je velmi nutné správně nastavit adresu serveru. Protože se jedná o operaci, kterou je nutné provést pouze u nasazení do emulátoru nebyla řešena konfigurace připojení na více serverů uživatelsky příjemnějším způsobem. Tento krok je způsoben chováním samotného emulátoru Windows Azure. U emulátoru není totiž předem jasné, na které

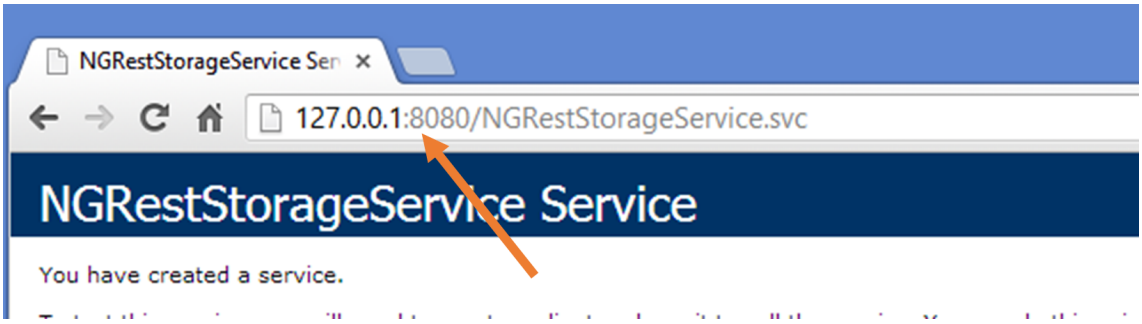
porty a adresy budou instance nasazeny. Proces nastavení klientům správné adresy serveru obsahuje následující kroky:

- Zprovozníme server nasazením do emulátoru pomocí návodu napsaného v sekci B.1.
- Po otevření dvou oken v prohlížeči vyhledáme okno s rozhraním služby *Storage Service*. Obsah tohoto okna by měl vypad následovně jako na obrázku B.1.
- V otevřeném okně rozhraní *Storage Service* vybereme soubor s názvem „NGRestStorageService.svc“, čímž se dostaneme na vystavené rozhraní klientům B.2.
- Nyní je nutné zkopírovat adresu tohoto rozhraní z prohlížeče B.2.
- Zkopírovanou adresu vložíme na příslušné místo do souboru „NGWinFormsGUI.exe.config“, který se nachází v adresáři s nainstalovaným klientem. Na obrázku B.3 je vyznačena pozice adresy v konfiguračním souboru, kterou je nutné nahradit aktuální adresou zkoupírovanou z okna prohlížeče.



Obrázek B.1: Obsah okna s obsahem složky nasazení instance *Storage Service*

Po provedení změny adresy serveru je možné klienta spustit normálním způsobem ze složky instalace, nebo zástupcem programu vytvořeným při instalačním procesu. Po spuštění je uživateli zobrazeno hlavní okno nastavení (obrázek B.4, kde je uživatel vyzván k zadání uživatelských údajů, případně je možné se zde okamžitě zaregistrovat kliknutím na tlačítko registrace (obrázek B.5). Po úspěšném přihlášení je nutné nastavit monitorovanou složku na záložce „Folder Path“ hlavního okna (obrázek B.6). Po vybrání složky je automaticky



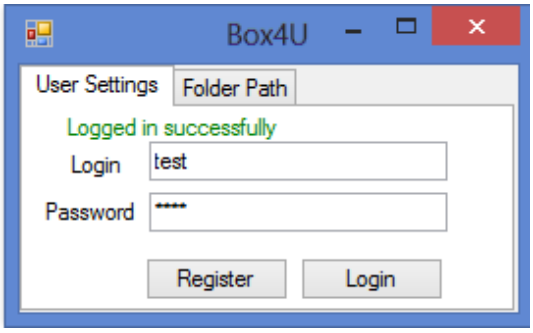
Obrázek B.2: Adresa rozhraní služby *Storage Service* vystavená klientům

```
client>
  <endpoint address="http://localhost:8080/NGRestStorageService.svc"
            binding="basicHttpBinding" bindingConfiguration="NGMsBoxServiceBinding"
            contract="NGMsBox.DataContracts.INGCommunicationContract" name="NGMsBoxServiceBinding" /
/client>
```

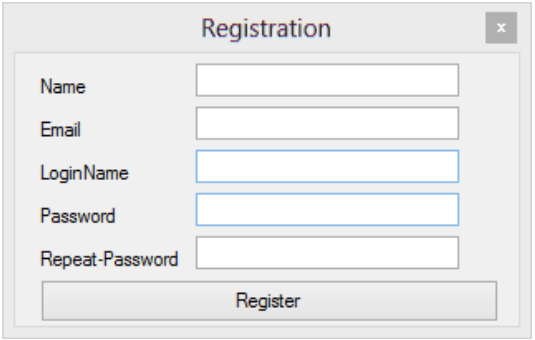
Obrázek B.3: Konfigurace klienta s vyznačeným místem adresy serveru

spuštěno monitorování souborů a jejich synchronizace se serverem. Klienta je možné minimalizovat do systémové lišty vedle hodin (obrázek B.7). Aplikaci můžeme kdykoliv zvětšit zpátky dvojitým poklepáním myši na ikonu v systémové oblasti.

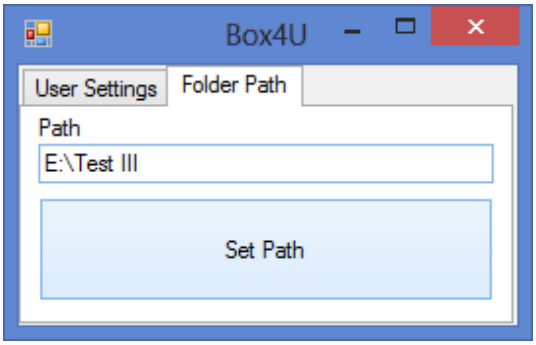
Pro vetší uživatelský zážitek z užívání systému doporučuji klienta pro webový prohlížeč, kterého vyvíjí Petr Messner v rámci Bakalářské práce.



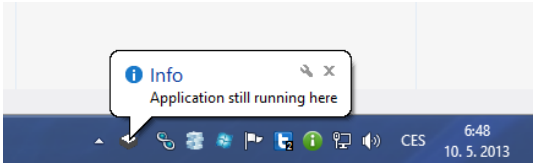
Obrázek B.4: Konfigurace uživatelského účtu v hlavním okně nastavení dekstopového klienta.



Obrázek B.5: Okno desktopového klienta pro registraci nového uživatelského účtu.



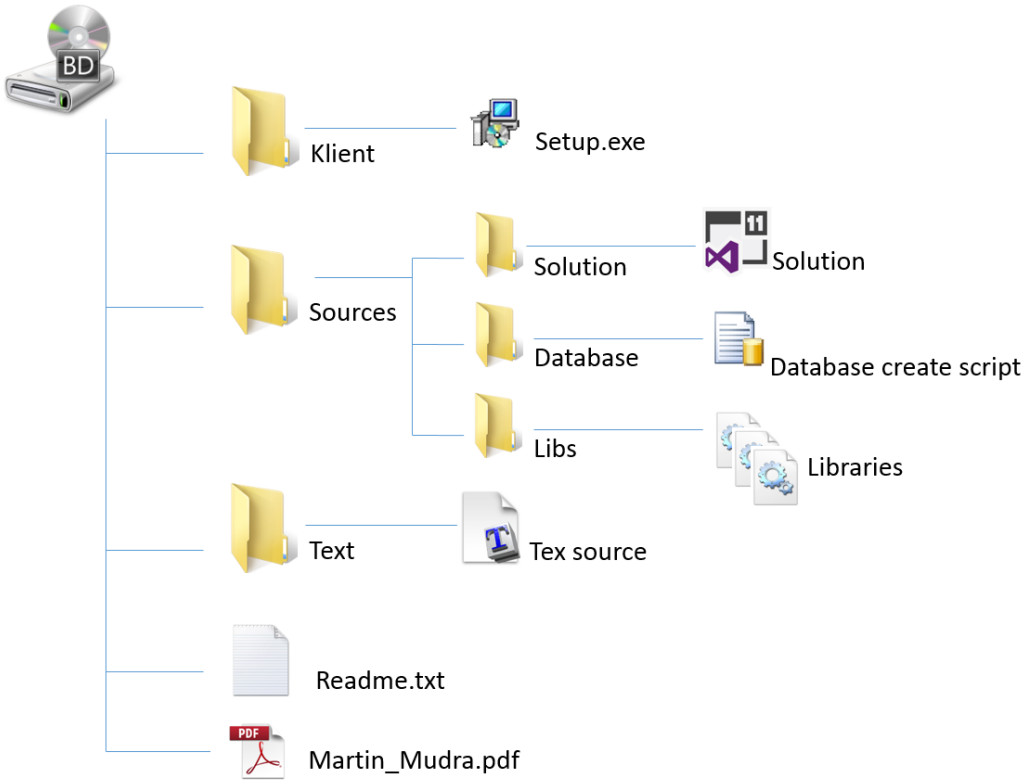
Obrázek B.6: Nastavení monitorované složky v klientovi



Obrázek B.7: Minimalizace klienta do systémové lišty systému Windows

Příloha C

Obsah přiloženého CD



Obrázek C.1: Seznam přiloženého CD