Mendelova univerzita v Brně
Provozně ekonomická fakulta

Optimalizace datového úložiště

Diplomová práce

Vedoucí práce:
RNDr. Zuzana Prišťáková, Ph.D.

Bc. Barbora Aulehlová

Brno 2015

Na tomto místě bych ráda poděkovala RNDr. Zuzaně Priščákové, Ph.D. za vstřícný přístup, ochotu a poskytnutí cenných informací při vedení této práce.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Optimalizace datového úložiště**
vypracovala samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*
Jsem si vědoma, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.
Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 20. prosince 2015 .....

Abstract

Aulehlová, B. Optimalization of data storage. Diploma thesis. Brno: Mendel University in Brno, 2015.

The theoretical part of this thesis is devoted to familiarization with the technology of cloud computing, virtualization technology, data storage problems and Petri nets. Followed by the practical part, the first part discusses the various technologies which are VMware ESX, Citrix Xen, Hyper -V, KVM and oVirt. The following part is a summary of the characteristics of each technology and an output of this summary is a decision tree which is intended to help to select the appropriate virtualization technology. The second part is dedicated to the implementation of an application used for storing data in the database depending on the sensitivity of data. The thesis includes the final economic evaluation and discussion as well.

Keywords

cloud computing, virtualization, Petri nets, data, hypervisor, VMware ESX, Citrix Xen, Hyper-V, KVM, oVirt, C#, MySQL, database, client-server

Abstrakt

Aulehlová, B. Optimalizace datového úložiště. Diplomová práce. Brno, 2015

Teoretická část této práce je věnována seznámení se s technologií cloud computing, virtualizační technologií, problematikou ukládání dat a Petriho sítím. Následuje praktická část, jejíž první část pojednává o jednotlivých virtualizačních technologiích a to VMware ESX, Citrix Xen, Hyper-V, KVM a oVirt. Následuje souhrn vlastností jednotlivých technologií a výstupem je poté rozhodovací strom pro volbu vhodné virtualizační technologie. Druhá část je věnována implementaci aplikace určené pro ukládání dat do databáze v závislosti na citlivosti. Součástí práce je závěrečné ekonomické zhodnocení a diskuze.

Klíčová slova

cloud computing, virtualizace, Petriho sítě, data, hypervizor, VMware ESX, Citrix Xen, Hyper-V, KVM, oVirt, C#, MySQL, databáze, klient-server

Obsah

Seznam obrázků

1 Úvod a cíl práce

1.1 Úvod do problematiky

V dnešní moderní době jsou virtualizační techniky prudce na vzestupu. Virtualizační techniky přináší firmám výrazné snížení nákladů a zefektivnění celkového provozu. Existuje mnoho řešení, ať už placená, či volně dostupná a každá firma si může vybrat řešení dle svých preferencí. Tato práce se bude zabývat právě volbou vhodného virtualizačního prostředí především s ohledem na bezpečnost.

Každá firma disponuje velkým množstvím dat. Tyto data jsou kritickou složkou, neboť v případě jejich zneužití může být ovlivněna existence celé firmy. U dat je třeba brát ohled především na zachování důvěryhodnosti, integrity a dostupnosti. Výstupem této práce bude aplikace sloužící k ukládání dat do databáze s ohledem na jejich citlivostní stupeň.

Data jsou v dnešní době uchovávána především elektronicky v datových úložištích, často za využití cloud computingu. Cloud computing je také poměrně novou technologií, umožňující firmám i jednotlivcům využívat zdroje bez nutnosti jejich pořízení. Jedná se v podstatě o placený přístup k požadovaným službám, ať už se jedná o software, či datové úložiště. Tato technologie opět přináší snížení nákladů a zvýšení pohodlí pro uživatele. Cloud computing úzce souvisí s virtualizací, a proto o něm bude v této práci také pojednáváno.

1.2 Cíl a metodika práce

Cílem této práce je porovnat dostupná virtualizační řešení. Vybrané virtualizační řešení bude poté nainstalováno, otestováno a namodelováno pomocí Petriho sítě, přičemž budou zkoumány jeho případné nedostatky. Součástí porovnání bude také rozhodovací strom určený pro společnosti a jednotlivce pro ulehčení rozhodování při výběru virtualizační technologie.

Dalším cílem je implementovat aplikaci, která bude schopna ukládat uživatelem zvolená data do databáze v závislosti na citlivostním stupni dat. Analytická část aplikace bude prováděna pomocí nástroje Enterprise Architekt a samotná implementace bude provedena v jazyce C# a v databázi MySQL.

Práce je zaměřena především na malé a středně velké společnosti, s čímž koresponduje i zvolená virtualizační technologie a implementovaná aplikace.

2 Struktura práce

Tato práce se dělí na dva větší logické celky – teoretickou část a praktickou.

V první části, nazvané Literární přehled, definujeme cloud, jeho základní typy a modely nasazení. Následně se zaměříme na virtualizaci, definování základních pojmů a opět se zaměříme na typy virtualizace a techniky. V další části jsou popsána data, a to především jejich životní cyklus, ukládání v cloudu a jejich bezpečnostní atributy. Poslední částí jsou Petriho sítě, kde jsou opět definovány základní pojmy a typy Petriho sítí. Součástí každé této části je shrnutí a nastínění využití dané problematiky v praktické části.

Další kapitola bude zaměřena na bezpečnost. První část této kapitoly se bude týkat bezpečnosti v rámci modelu TCP/IP, přičemž budou analyzovány bezpečnostní prvky v rámci vrstev tohoto modelu. Následovat bude popis bezpečnostních protokolů SSL a TLS a jejich srovnání.

Následovat bude kapitola týkající se porovnání dostupných virtualizačních technologií. Konkrétně se bude jednat o technologie VMware ESX, Citrix Xen, Hyper-V, KVM a oVirt. Každá tato technologie bude popsána a zaměříme se také především na výhody a nevýhody dané technologie. Výstupem této části bude poté rozhodovací strom určený pro usnadnění rozhodování při výběru virtualizační technologie.

Následující kapitola se bude zabývat již zvolenou virtualizační technologií, která bude nainstalována, otestována a její instalace bude namodelována pomocí Petriho sítě.

Stěžejní kapitolou bude popis implementace aplikace pro ukládání dat na základě citlivosti. Problém aplikace bude definován pomocí formální i neformální specifikace, budou popsány použité nástroje a aplikace samotná.

Závěrem práce bude ekonomické zhodnocení a diskuze zaměřená především na zhodnocení kladů a záporů zvoleného řešení a navržení možných rozšíření.

3 Literární přehled

3.1 Virtualizace

Hlavní myšlenkou virtualizace je schopnost přenést fyzický server do virtuálního stroje. Základy virtualizace položila společnost IBM v roce 1960 a ve svém článku poté popsala principy a vlastnosti virtuálních strojů, které jsou využívány dodnes. Dle jejich definice umožňuje virtuální stroj virtualizaci všech hardwarových zdrojů, a to včetně procesorů, pamětí, úložišť a síťových připojení. Virtuální stroj je velmi podobný klasickému fyzickému serveru. Podporuje operační systém a obsahuje množinu zdrojů, ke kterým mohou aplikace na virtuálním stroji přistupovat. Mezi základní pojmy v oblasti virtualizace patří hostitel a hypervizor. Hostitel je server poskytující hardwarové prostředky, zejména CPU a RAM, a hypervizor je virtualizační vrstva, která umožňuje běh virtuálního stroje nad hardwarovými prostředky hostitele. Hypervizor může jet přímo na hardwaru serveru bez potřeby operačního systému, nebo se může jednat o klasickou aplikaci běžící na operačním systému (Portnoy, 2012). Představiteli hypervizorů jsou: Vmware ESX, Citrix, Hyper-V či KVM.

Virtualizace je významně spjata s cloud computingem, neboť mění data centra na soběstačné, vysoce škálovatelné a dostupné zdroje.

Typy virtualizace

- **Serverová virtualizace:** Hypervizor poskytuje fyzickou vrstvu k využití virtualizovanému serveru či stroji. Hypervizor je zde nainstalován přímo na serveru, a to bez potřeby operačního systému, kde jsou virtualizační stoje poté zaváděny (bootovány). Hypervizor se pak stává rozhraním mezi hardwarovými prostředky serveru a virtuálními prostředky virtuálního stroje. Zatímco hypervizory jsou základy pro virtuální prostředí, virtuální stroje jsou základem pro aplikace. Virtuální stroje obsahují vše co jejich fyzické protějšky, tedy operační systém, aplikace, přístup k úložišti nebo síťové připojení. Virtuální stroje mohou být navíc naopak od fyzických klonovány, upgradovány, či přemísťovány na libovolná místa bez toho, aniž by si toho uživatel všimnul. V klasickém pojetí jeden server zastává jednu funkci, u serverové virtualizace je ovšem několik serverů provozováno pouze jedním fyzickým serverem. Virtualizované servery sdílejí zdroje jednoho stroje skrz různá prostředí, což má za následek potřebu méně fyzických serverů, a tím pádem snížení nákladů a zvýšení využití hardwaru (Portnoy, 2012).
- **Klientská (desktopová) virtualizace:** Virtuální klienti (desktopy) běží na serverech v datacenterech, přičemž tyto servery obsahují lepší a spolehlivější hardware, než klasické počítače. Aplikace, na které se uživatelé připojují, taktéž běží v datacentrech, tudíž je celý proces efektivnější. Klasické počítače s sebou přinášejí mnoho obtíží, ať už se jedná softwarové aktualizace, záplaty (patche), či hardwarovou podporu. A proto je tento typ virtualizace je nasa-

zován především za účelem ulehčení správy počítačů a zvýšení bezpečnosti. Virtuální klient je dostupný skrz tzv. thin klienta, což je nízkonákladový, centrálně spravovaný počítač, který spoléhá na serverovou výpočetní sílu. Jak již bylo řečeno, vytvoření obrazů klientů (desktopů) jakožto virtuálních strojů s sebou přináší mnohé výhody. Několik obrazů může být sdíleno mezi stovkami uživatelů, záplaty a aktualizace jsou aplikovány přímo na obraz, čímž se okamžitě dostávají k uživateli. Pokud dojde k nečekanému poškození obrazu, stačí jej nahradit jeho originálem. Co se bezpečnosti týče, existují speciálně navržené virtuální stroje, které běží na serveru a zabezpečují tak všechny virtuální klienty, které zde běží (Portnoy, 2012).

- **Aplikační virtualizace:** Za virtualizací aplikací stojí dva hlavní důvody – ulehčení nasazení a vzájemné působení aplikací mezi sebou. Díky tomuto typu virtualizace není třeba instalovat aplikaci na každý počítač zvlášť a následně na každém počítači provádět případné aktualizace. Navíc je zde omezena možnost nefunkčnosti aplikace zapříčiněná jinou aplikací (Portnoy, 2012).

Techniky virtualizace

- **Plná virtualizace:** Jedná se o virtualizaci za použití binárního překladu a přímého provedení. To zajišťuje úplnou virtualizaci, protože hostovaný operační systém je od základního hardwaru zcela oddělen virtualizační vrstvou. Hostovaný operační systém tedy nepociťuje žádnou změnu a není si vědom toho, že je virtualizován, stejně tak jako aplikace nepotřebují žádné modifikace. Hypervizor překládá za chodu instrukce operačního systému a odchytává jejich výsledky pro pozdější využití. Plná virtualizace poskytuje nejlepší izolovanost a bezpečnost v rámci virtuálních strojů a usnadňuje migraci a přenositelnost. Nevýhodou ovšem je, že při plné virtualizaci je prakticky nemožné dosáhnout plného výkonu. Příkladem této virtualizace jsou produkty VMware a Microsoft Virtual Server (Matyska, 2007).
- **Paravirtualizace:** Tato virtualizace má za úkol zvýšení výkonu a efektivnosti. Vzhledem k tomu, že zde není možné využít nemodifikované operační systémy, klesá její kompatibilita a přenositelnost. Virtualizace v tomto případě tedy není úplná, některé vlastnosti mohou být omezeny a operační systém je schopen rozpoznat, že běží ve virtuálním prostředí. Při paravirtualizaci nepřistupuje virtualizovaný stroj přímo k procesoru a instrukce jsou prováděny skrze virtuálního managera (VMM). Zástupci této virtualizace jsou VMware workstation a Xen (Matyska, 2007).
- **Hardwarová virtualizace (hardwarová podpora virtualizace):** Jedná se o dvě podobné technologie společností Intel (virtualizace Intel VT) a AMD (virtualizace AMD-V), které se snaží optimalizovat výkon procesoru pro běžnou virtualizaci za pomoci překladu instrukcí a paměťových adres. Hypervizor má za úkol kontrolovat procesor, paměť a další komponenty tím, že umožňuje několika různým operačním systémům běžet na stejném serveru bez potřeby zdrojového

kódu. Operační systémy se pak jeví, jako kdyby měly vlastní procesor, paměť a ostatní komponenty (Reys, 2008).

Život virtuálního stroje

Všechny stroje, ať už se jedná o virtuální, nebo fyzické, mají životní cyklus. Životní cyklus virtualizovaného stroje se od fyzického částečně liší. V dynamických datacentrech jsou pouze hostované servery fyzickými stroji, což zkracuje a zjednodušuje jejich životní cyklus. Virtuální služby naopak vyžadují rozsáhlejší životní cyklus. Základem tohoto životního cyklu jsou čtyři fáze – plánování, příprava a nasazení, provoz, vyřazení (Ruest, Ruest, 2009).

- **Plánování:** Jedná se o fázi zaměřenou na identifikaci požadavků a přípravu řešení pro nasazení. Základní výhodou u virtuálních serverů je absence koupě fyzického serveru.
- **Příprava a nasazení:** Tato fáze se zaměřuje na technickou architekturu procesu a může nastat ve stejnou dobu jako proces plánování. Technická architektura specifikuje technické parametry využívané během instalace. Tato fáze dále zahrnuje získávání a vytvoření balíčků, konfigurace, instalace a testování nasazených strategií.
- **Produkce:** V rámci této fáze nastává správa problémů, změn a optimalizace a administrativní management v rámci sítě. Patří sem i smlouva s koncovým uživatelem, tzv. SLA (service level agreement). SLA se soustředí především na výkonnostní a kapacitní analýzu, dostupnost, spolehlivost a reakční dobu služeb.
- **Vyřazení:** Jakmile stroj dosáhne jistého stupně zastarání, je třeba ho vyřadit a odstranit tak zastaralé technologie a procesy.

Shrnutí

V této kapitole byl definován pojem virtualizace, její typy, techniky a život virtuálního stroje. Virtualizace je schopnost přenést fyzický server do virtuálního stroje, čímž především snižuje náklady společnosti a zefektivňuje správu strojů. Virtualizace může být serverová, klientská, nebo aplikační. Dále je třeba zvolit techniku virtualizace, které se dělí na virtualizaci plnou, paravirtualizaci a hardwarovou. V praktické části bude využíván fyzický server, na kterém bude virtualizován opět server, a proto bude zvolena virtualizace plná serverová.

3.2 Cloud

Základní myšlenkou cloudu je využívání vzdáleně hostovaných výpočetních zdrojů, které jsou zprostředkovávány skrze Internet. Přesná definice cloudu je ovšem o něco složitější problém a její interpretace se liší. Nejuznávanější definice byla uveřejněna Národním institutem standardů a technologie (NIST) v roce 2011: „Cloud computing je model pro umožnění vyhovujícího, na vyžádání umožněného přístupu ke sdí-

lenému prostoru výpočetních zdrojů (např. sítě, servery, úložiště, aplikace a služby), které mohou být rychle poskytovány a uvolňovány s minimálním úsilím ze strany managementu nebo poskytovatele.“ Bohužel ovšem ani tato definice není všeobecně přijata a mezi experty se vyskytují neshody (Rhoton, Haukioja, 2013).

Další možnou definicí cloudu je definování pomocí jeho vlastností:

- **Multitenancy (sdílené zdroje):** Na rozdíl od předchozích výpočetních modelů, ve kterých jsou zdroje využívány pouze jedním uživatelem, je cloud založen na business modelu, ve kterém jsou zdroje sdílené. Tento fakt umožňuje dosažení škálovatelnosti a úspory nákladů (Rhoton, Haukioja, 2013), (Mather, Kumaraswamy, Latif, 2009).
- **Elasticita:** Základním přínosem cloudu je škálovatelnost poskytovatele služeb, která je k dispozici koncovým uživatelům. Škálovatelnost je navíc umožněna oběma směry dle potřeby, tedy uvolnění zdrojů v případě jejich nevyužívání, nebo naopak využívání dalších nových zdrojů. Cloud computing navíc umožňuje škálovatelnost v rámci desítek tisíců systémů, šířky pásma a úložného prostoru (Rhoton, Haukioja, 2013), (Mather, Kumaraswamy, Latif, 2009).
- **Flexibilní platby:** Různé využívání zdrojů kombinované se službami na vyžádání umožňuje různé možnosti plateb zákazníků. Základem a velikou výhodou pro uživatele je účtování plateb pouze za vyžádané zdroje a čas, ve kterém dané zdroje opravdu využívají. Co se plateb týče, u cloudového řešení odpadají uživatelům fixní náklady a je jim umožněna lepší optimalizace nákladů (Rhoton, Haukioja, 2013), (Mather, Kumaraswamy, Latif, 2009).

Mezi další vlastnosti pak patří například doručitelnost služeb, kdy služby jsou nejčastěji doručovány ve formě programových rozhraní, jakožto doplněk k rozhraní uživatelskému, nebo univerzální přístup, díky němuž jsou zdroje přístupné jakékoliv autorizované entitě, navíc se uživatel ke službě může připojit nezávisle na místě (Rhoton, Haukioja, 2013).

Cloud service model

Cloud service model neboli SPI model, představuje tři servisní modely pro cloud computing, které se liší v rozsahu sílení prostředků svým uživatelům.

- **Cloud Software-as-a-Service (SaaS):** U klasického přístupu je zákazník nucen si k požadovanému softwaru zakoupit licenci a nainstalovat si tento software na svůj hardware. To může ovšem přinášet různé komplikace, neboť je třeba brát ohled na kompatibilitu s operačním systémem nebo instalace aktualizací. Další nevýhodou je, že zákazník, který potřebuje software využít pouze nárazově, zaplatí ve většině případů za licenci stejně jako uživatel, který chce software využívat dlouhodobě a intenzivně. Tyto problémy řeší právě SaaS. Jak již název napovídá, jedná se o model poskytující svým uživatelům aplikace, běžící v cloudové infrastruktuře. Tyto aplikace jsou přístupné skrze rozhraní různých klientů a není tedy vyžadován žádný speciální hardware. Uživatel tedy nemá přístup k samotné cloudové infrastruktuře, síti, úložišti či serverům. Typickým

příkladem tohoto modelu je Salesforce.com, který poskytuje svým zákazníkům software pro řízení vztahu se zákazníky (CRM) (Mather, Kumaraswamy, Latif, 2009), (Winkler, 2011). Obměnou SaaS je STaaS (Storage-as-a-Service), ve kterém poskytovatel pronajímá uživateli svá úložiště.

- **Cloud Platform-as-a-Service (PaaS):** Tento model umožňuje svým zákazníkům nasazení jimi vytvořené či získané aplikace do cloud infrastruktury. Podmínkou pro tuto aplikaci ovšem je, aby byla vytvořena pomocí programovacích jazyků a nástrojů podporovaných poskytovatelem. Součástí tohoto přístupu bývá i poskytnutí vývojového prostředí pro vývojáře aplikací ve formě služby. Vývojáři tak mohou vyvíjet aplikace bez nutnosti instalace jakýchkoliv nástrojů na jejich hardware. Uživatel zde opět nemá přístup k síti, úložišti či serverům, a nemůže spravovat či kontrolovat cloudovou infrastrukturu, ovšem má kontrolu nad nasazenými aplikacemi včetně jejich konfigurací. Příkladem tohoto modelu je Google App Engine, který umožňuje uživatelům sestavit a provozovat aplikace na Google infrastruktuře (Mather, Kumaraswamy, Latif, 2009), (Winkler, 2011).
- **Cloud Infrastucture-as-a-Service (IaaS):** U klasického přístupu je v případě hostovaných aplikací dána uživateli k dispozici celá infrastruktura. Často zde také nastává nutnost nákupu, či pronájmu speciálního hardwaru pro dané aplikace. Model IaaS dává zákazníkovi největší volnost z výše zmíněných modelů, neboť mu poskytuje úložiště, sítě a další počítačové zdroje, kde může uživatel umístit a spustit libovolný software, ať už se jedná o operační systémy nebo aplikace. Tento model je vysoce škálovatelný a služby jsou přidávány, odebírány libovolně dle požadavků uživatele. Opět je zde výhodou platba pouze za opravdu využité zdroje. Uživatel opět nemá oprávnění spravovat či kontrolovat cloudovou infrastrukturu, ovšem může provádět kontroly skrze operační systémy, úložiště a nasazené aplikace a částečně i skrze síťové komponenty. Typickým představitelem je Amazon Elastic Compute Cloud, což je webová služba nabízející cloudem hostované služby (Mather, Kumaraswamy, Latif, 2009), (Winkler, 2011).

Realita je ovšem jiná a existuje tak mnoho modelů, které nerespektují vymezené hranice a nespadají tak přesně ani do jedné výše uvedené kategorie (Rhoton, Haukioja, 2013).

Modely nasazení

Modely nasazení nám říkají, jakých způsobem je cloud poskytován.

- **Veřejné cloudy:** Tento přístup je nejklasičtějším pojetím cloudu. Zdroje jsou zde poskytovány široké veřejnosti dynamicky třetí stranou skrze Internet. Prodejce má v tomto případě k dispozici jedno, či více data center, ze kterých své služby poskytuje. Služba je poskytována více uživatelům zaráz skrz běžnou infrastrukturu. Vzhledem k tomu, že se jedná o nejčastější model nasazení, jsou tyto cloudy nejčastěji dostupné přes Internet, čímž vycházejí vstříc běžným uži-

vatelům. Vzhledem k tomu, že jsou ukládána na běžných datových úložištích, hraje zde identita uživatele, kontrola přístupu a šifrování velice důležitou roli. Poskytovatel služeb je zodpovědný za bezpečnost a každodenní operace týkající se poskytovaných služeb, a tím uživateli těchto služeb zůstává minimální kontrola nad bezpečnostními aspekty (Mather, Kumaraswamy, Latif, 2009), (Winkler, 2011).

- **Soukromé cloudy:** V tomto případě se jedná o provozování cloud computingu v privátních sítích. Na rozdíl od veřejných cloudů, které jsou dostupné všem, kteří si o jeho služby zaplatí, je soukromý cloud k dispozici pouze dané organizaci, která si ho koupí a zdroje tedy nejsou využívány více uživateli zároveň. Tato organizace je poté i zodpovědná za operace prováděné v tomto cloudu. Ačkoliv data v tomto cloudu jsou přístupná pouze dané organizaci, mívají tato data přísná opatření, aby se zajistila izolace dat pouze v rámci soukromého cloudu. Vzhledem k tomu, že jsou tyto cloudy soukromé, nejsou zde aplikována některá bezpečnostní opatření, která můžeme nalézt u cloudů veřejných. Ovšem jak již bylo zmíněno výše, to, že se jedná o soukromý cloud, nezaručuje samo o sobě jeho bezpečnost. Soukromé cloudy se dále dělí na dedikované (cloud je hostován na datovém centru organizace a je spravován IT oddělením organizace) a řízené (infrastruktura je vlastněna zákazníkem, ovšem správu provádí prodejce) (Mather, Kumaraswamy, Latif, 2009), (Winkler, 2011).
- **Komunitní cloudy:** Jedná se o sdílení v rámci neveřejného cloudu. Cloud je v tomto případě hostován na zařízeních prodejce a zároveň je prodejcem spravován. Může být navíc sdílen mezi několika organizacemi zároveň. Ideální je tento model pro organizace, které podléhají stejným regulačním předpisům nebo právním omezením. Komunitní cloud bývá občas zahrnován mezi soukromé cloudy (Mather, Kumaraswamy, Latif, 2009), (Winkler, 2011).
- **Hybridní cloudy:** Jedná se o kombinaci veřejného a soukromého, případně komunitního cloudu. Jednotlivé cloudy jsou poté propojeny standardizovanou technologií umožňující přenos dat a aplikací. V hybridním cloudu mohou organizace provozovat kritické aplikace, citlivá data v cloudu soukromém a ostatní v cloudu veřejném. Typickým využitím veřejného cloudu může být testování a zabezpečování jakosti v době, kdy jsou na privátním cloudu prováděny aktualizace (Mather, Kumaraswamy, Latif, 2009), (Winkler, 2011).

Shrnutí

V této kapitole byl definován cloud, včetně cloud service modelu a modelu nasazení. Cloud service model v sobě zahrnuje tři modely, a to SaaS, PaaS a IaaS. SaaS je určen pro zprostředkovávání aplikací uživatelům skrze cloud, PaaS umožňuje uživatelům nasazení jejich aplikace do cloudové infrastruktury a IaaS poskytuje uživateli v rámci cloudu úložiště, sítě a další počítačové zdroje. V praktické části bude využívána obdoba SaaS, a to STaaS, což ve kterém poskytovatel místo aplikací nabízí uživatelům pronájem úložišť. V modelu nasazení byly rozebrány veřejné, soukromé,

komunitní a hybridní cloudy. Využití jednotlivých modelů ve velké míře závisí na velikosti podniku. Podniky rozlišujeme na mikropodniky (do 10 osob), malé podniky (do 50 osob), střední (do 250 osob) a veliké. Vzhledem k tomu, že je praktická část práce zaměřena na středně velké podniky, byl zvolen soukromý (privátní) cloud. Středně velké firmy většinou disponují poměrně velkým množstvím citlivých informací, jejichž únik na veřejnost by měl na firmu kritický dopad, a proto je zvolen soukromý cloud, který je přístupný vždy pouze dané organizaci.

3.3 Data ve virtualizovaném prostředí

Na začátku této kapitoly je třeba si definovat základní pojmy. Data jsou znaky, signály, které subjekt získává vnímáním reality, jsou to atributy objektů. Z dat vznikají zprávy, což jsou data vyznačující se syntaxí a sémantikou. Informace je pak zpráva, která snižuje míru neznalosti o daném jevu. Praktické poznatky a teorie získané z informací pak souhrnně tvoří znalosti (Konečný, 2002).

Životní cyklus dat v klasickém prostředí

Aby data naplnila svou podstatu, je třeba s nimi vhodně pracovat od jejich vzniku, až po jejich archivování, či případné zničení. Nejlepším možným přístupem v rámci správy dat je dodržování jistých osvědčených postupů, standardů a norem, které jsou konzistentní a předem definované. Data jsou základem každé společnosti, a proto je třeba s nimi pracovat efektivně.

- **Vytvoření:** Data mohou být vytvořena různými způsoby, typickým příkladem je výzkum, sběr dat (experiment, pozorování, měření, simulování), odchytávání, koupě dat, či vytváření metadat. V rámci vytvoření může ještě nastat fáze plánování, ve které jsou stanoveny postupy a práce s daty (Mather, Kumaraswamy, Latif, 2009).
- **Zpracování:** Data je po vytvoření třeba zpracovat, aby mohla být dále využívána. Mezi procesy zpracování patří digitalizování dat (v případě papírové, či jiné neelektronické podoby), kontrola dat, validace, pročištění dat. Některá data může být třeba anonymizovat, pokud se jedná o citlivé údaje. Jak nezpracovaná, tak zpracovaná data vyžadují kompletní metadata pro případ možné kopie. Konečnou fázi je poté zpracování dat a uložení. Celou fázi je třeba důkladně dokumentovat, aby se tak zajistila potřebná integrita dat (Mather, Kumaraswamy, Latif, 2009).
- **Analýza:** Abychom z dat získali potřebné informace, je potřeba data správně interpretovat, odvodit a sestavit vhodné výstupy. Klasickým obsahem analýzy může být popsání faktů, nalezení vzorů, testovací hypotézy, statistická analýza, či modelování. Konečným výstupem je poté interpretování výsledků dané analýzy. Data je v této fázi také potřeba připravit na další uchování (Mather, Kumaraswamy, Latif, 2009).

- **Uchování (uložení):** Data jsou uchovávána za účelem přístupu k datům po určitou dobu a pro jejich případné budoucí využití. Před uchováním dat je třeba data převést do nejvhodnějšího formátu z hlediska uložení. Dále je třeba zvolit vhodné ukládací médium v závislosti na citlivosti dat, potřebné délce jejich uložení a dalších faktorech. Ačkoliv většina dnešních úložišť může být považována za bezpečná, vždy je vhodné data zálohovat na více místech. K uloženým datům je také vhodné vytvořit metadata a dokumentaci (Mather, Kumaraswamy, Latif, 2009).
- **Přidělení přístupů:** Před udělením přístupů je potřeba vhodně určit, která data je možné sdílet dál, která poskytnout k dispozici omezenému okruhu uživatelů a která nechat bez přístupu. Autorská data je dále vhodné opatřit ochranou v podobě autorského práva. V případě sdílení dat je třeba zajistit kompletní metadata a je třeba zajistit bezpečný přenos dat (Mather, Kumaraswamy, Latif, 2009).
- **Opakované využití:** Data mohou být v budoucnu využita opakovaně, a to například při následujících výzkumech, porovnávání výsledků s novými výsledky, nebo při učení (Mather, Kumaraswamy, Latif, 2009).

Životní cyklus dat v cloudu

Informace prochází několika fázemi – vznik, použití, přenos, transformace, uložení, archivování a destrukce. Každá fáze se navíc skládá z několika částí.

- **Vznik informace:** Řeší vlastnictví informace, tedy kdo ve společnosti informaci řeší a jak je řešeno vlastnictví v případě, že společnost využívá cloud computing. Dále je zde řešena klasifikace informace, tedy kdy a jak je informace klasifikována a existují-li omezení v případě použití cloud computingu. Posledním bodem je řízení, které řeší, zda existuje jistá struktura řízení, která zajišťuje, aby byla informace v cloudu správně spravována a chráněna po celou dobu životního cyklu (Mather, Kumaraswamy, Latif, 2009).
- **Použití informace:** Řeší především, zdali je informace používána interně, nebo externě, tedy použití v rámci organizace (např. v soukromém cloudu), nebo mimo (např. ve veřejném cloudu). Dále je řešeno použití informace třetí stranou, tedy jestli je informace sdílena s dalšími organizacemi. Další částí je ujistit se, že použití informace koresponduje s účelem, ke kterému byla určena a jestli její použití v cloudu koresponduje s ustanoveními společnosti. Poslední částí je zajištění, aby informace v cloudu byla spravována tak, aby podléhala právním požadavkům (Mather, Kumaraswamy, Latif, 2009).
- **Přenos informace:** V prvé řadě řeší, jestli je informace přenášena v rámci soukromé, nebo veřejné sítě. V případě veřejné sítě je totiž třeba zajistit řádné zabezpečení proti úniku, či zneužití přenášených informací. Přenášenou informaci je dále s ohledem na bezpečnost nutné šifrovat. Některé zákony dokonce vyžadují šifrování v případě přenosu osobních údajů skrze veřejné sítě. Posledním

bodem je kontrola přístupu v rámci cloudu, tedy zajištění, že přístup k datům budou mít pouze ověření uživatelé (Mather, Kumaraswamy, Latif, 2009).

- **Transformace informace:** Je důležité, aby byla zachována původní omezení a ochrana informace, aby byla zachována integrita dat a aby v případě agregace dat nebyla informace, případně osobní údaje, stále vázány k jisté osobě (Mather, Kumaraswamy, Latif, 2009).
- **Uložení:** Zde je potřeba opět zajistit kontrolu přístupu, tedy aby k datům neměly přístup neoprávněné osoby. Dále je třeba zajistit takové uložení dat, které by umožňovalo organizaci přístup k datům a jejich správu v budoucnosti. Je také třeba zajistit základní bezpečnostní atributy, což je integrita, důvěrnost a dostupnost dat a dále je zde opět třeba zajistit vhodné šifrování informace (Mather, Kumaraswamy, Latif, 2009). Právě ukládání dat je hlavní náplní této práce.
- **Archivace:** U archivace informace je třeba mít na paměti, že informace, jako jsou například osobní údaje, mohou mít specifické požadavky co se jejich uložení a archivace týče. Dále je třeba zajistit, aby informace byla archivována na médiu, na které bude umožněn přístup i v budoucnu a jak dlouho mohou být data uložena v případě, že je využíváno úložiště mimo organizaci (Mather, Kumaraswamy, Latif, 2009).
- **Zničení:** Posledním krokem je zničení informace. Informaci je třeba zajistit její bezpečné zničení, aby se předešlo případnému úniku informace a je třeba informaci kompletně (Mather, Kumaraswamy, Latif, 2009).

Jednotlivé kroky se mohou lišit v závislosti na využití, či nevyužití cloudu, případně typu cloudu a také na povaze informace, kde právě osobní údaje vyžadují nejvyšší stupeň bezpečnosti.

Při ukládání dat v cloudu musíme myslet na několik klíčových rizik. Prvním rizikem je fakt, že ve většině cloudů je úložiště implementováno centrálně (v rámci jednoho místa), a tím se stává zajímavým cílem pro trestné činy. Trestným činem může být v tomto případě jak napadení hackerem, tak přímo útok na fyzické zařízení například v podobě krádeže. Na druhou stranu centralizace umožňuje poskytovatelům lepší možnost monitorování a s tím i rychlejší detekování potencionálních problémů. Dalším rizikem je „multi-nájem“ (multitenancy), tedy sdílení zdrojů mezi více uživateli. S tím jsou spojeny možné problémy především při zálohování dat a při operacích týkajících se vracení změn. Kromě těchto problémů je i riziková možnost, že data jednoho uživatele budou smíchány s daty jiného uživatele. Posledním rizikovým faktorem je samotný software a hardware. V nejhorších případech může pak dojít ke zničení dat, či odhalení dat zákazníkům navzájem. Mezi další méně významná rizika patří například poskytovatelé, kteří sice nevyužívají centralizovaný systém úložišť, ovšem každé úložiště mají v jiné lokalitě, což může představovat i jinou zem. S jinou zemí přichází jiné zákony a tak i možnost přístupu k datům od vyšších autorit této země (Winkler, 2011).

Jak již bylo psáno výše, základní ochranou uložených dat je jejich šifrování. Šifrování se liší v závislosti na využitém typu modelu. Při využití IaaS je šifrování

k dispozici a s ohledem na daný model je silně doporučováno. Ovšem u modelů PaaS a SaaS nemusí být šifrování realizovatelné, neboť šifrování by znemožnilo indexování či vyhledávání dat. Navíc data z těchto modelů jsou míšena s daty ostatních uživatelů, a přestože jsou zde aplikace navrhovány tak, aby bránily neautorizovanému přístupu, jistá rizika zde stále jsou. Někteří poskytovatelé tak vyvíjí aplikace ve spolupráci se třetí stranou, aby tak zajistili větší bezpečnost jejich služeb. Šifrování se ovšem netýká pouze dat. Další využití v cloudu jsou kontrola přístupu k rozhraním vedoucím ke zdrojům, kontrola přístupu k administraci virtuálních strojů a obrazům operačních systémů a kontrola přístupu k aplikacím (Mather, Kumaraswamy, Latif, 2009), (Winkler, 2011).

Při ukládání dat do cloudu v rámci organizace je třeba mít zaznamenáno kdy a kam byly data uložena s ohledem na předpisy společnosti a pro případné audity. K ukládání dat se váží tři základní bezpečnostní atributy, a to důvěrnost (přístup k datům pouze pro oprávněné osoby), integrita (zajištění neporušitelnosti dat) a dostupnost (možnost pro uživatele mít možnost ke svým datům kdykoliv přistoupit).

- **Důvěrnost:** U důvěrnosti ve veřejném cloudu musíme zvážit především dva aspekty, a to kontrolu uživatelského přístupu a jakým způsobem jsou data uložená v cloudu chráněna. Kontrola přístupu by se měla skládat z autentizace (ověření identity uživatele pomocí jména a hesla nebo pomocí biometrických údajů či certifikátu) a autorizace (práva nebo výsady přidělené danému uživateli či procesu). Bohužel poskytovatelé cloudových služeb využívají ve většině případů klasický autorizační postup, a to například jméno a heslo a ani autentizační mechanismy nebývají silné. Pro běžného uživatele může být tato kontrola přístupu dostačující, ovšem pro větší společnosti se může stát kritickou. Ukládání dat v cloudu by v každém případě mělo zahrnovat jejich šifrování. U šifrování nás zajímá především jaký šifrovací algoritmus je využíván a jak silný je šifrovací klíč tohoto algoritmu. Bohužel někteří poskytovatelé svým uživatelům šifrování dat při ukládání nenabízí a uživatel si tak musí zajistit šifrování sám, což může být pro většinu uživatelů výrazný problém a mnoho uživatelů se pak smíří s nešifrovanými daty. Jak již bylo řečeno, pokud poskytovatel využívá šifrování dat, je důležité, aby šifrovací algoritmus, který využívá, byl opravdu bezpečný. V každém případě je třeba využívat symetrické šifrovací algoritmy, u kterých je využíván stejný klíč jak pro šifrování, tak pro dešifrování. Pouze symetrické algoritmy totiž disponují potřebnou rychlostí a efektivností pro šifrování většího množství dat. U symetrického šifrování dále platí, že čím delší klíč je použit, tím je šifrování bezpečnější. Vzhledem k tomu, že je klíč stejný pro šifrování i dešifrování, o to větší důraz je potřeba klást na jeho uchovávání. V ideálním případě by měl mít uživatel klíč uložen u sebe a nenechávat tento klíč uložen v cloudu v místě, kde jsou uložena data (Mather, Kumaraswamy, Latif, 2009).
- **Integrita:** Důvěrnost v sobě ovšem nezahrnuje integritu ani dostupnost. Namísto klasického šifrování využívá integrita MAC, Message authentication code, což je kryptografická funkce zajišťující kromě integrity i autentizaci dat. Nejjed-

nodušším využitím MAC je využití blokového symetrického algoritmu. Dalším důležitým faktorem je možnost kontroly integrity dat bez nutnosti stáhnout si tyto data z cloudu a až poté je zkontrolovat. Z tohoto důvodu je třeba matematická možnost kontroly integrity, a to například pomocí CRC (cyklického redundantního součtu) (Mather, Kumaraswamy, Latif, 2009).

- **Dostupnost:** Posledním bezpečnostním atributem je dostupnost. Dostupnost nejvíce ohrožují tři faktory, a to síťové útoky, dostupnost samotného poskytovatele služeb a nakonec poskytované služby. Proti síťovým útokům je třeba bránit se především pomocí specializovaného hardwaru (firewall, IDS, IPS) a softwaru (antivir, spyware). Tyto metody ovšem nikdy nebudou dostačující v případě, že nebudou dodrženy základní bezpečnostní aspekty, jako je například fyzické zabezpečení serverů či znemožnění přístupu do sítě neautorizované osobě. Co se dostupnosti poskytovatelů týče, téměř žádný nenabízí známých 99,999%. Klasickým příkladem je u poskytovatelů cloudových služeb dostupnost 99,9% času, což představuje necelých 9 hodin nedostupnosti ročně. Co se dostupnosti poskytovatelů týče, je třeba brát v potaz i samotnou budoucí existenci daného poskytovatele. U poskytovaných služeb je časté, že poskytovatel sice data ukládá, ovšem nezajišťuje jejich zálohy, případně je zajišťuje za další poplatky (Mather, Kumaraswamy, Latif, 2009).

Všechny tyto tři bezpečnostní atributy by měly být zahrnuty ve smlouvě mezi zákazníkem a poskytovatelem cloudových služeb, ovšem i přesto by měl zákazník věnovat zvýšenou pozornost tomu, u koho svá data nechává uložená. S ukládáním dat také blízce souvisí ochrana údajů. V cloudu je tato ochrana o to důležitější, neboť zde třetí strany provádějí správu a operace nad našimi daty, a tím k nim v podstatě přistupují. Pro organizace je situace složitější, neboť některé musejí s ohledem na právní požadavky zajistit, že jejich poskytovatel cloudových služeb zajišťuje správnou ochranu údajů. Navíc na data uložená v cloudu se váží stejná omezení jako na běžně uložená data a je možné je využívat k účelu, ke kterému byla primárně určena. S osobními daty může jejich vlastník ve většině příkladů manipulovat, tedy upravovat je a mazat, a toto jeho právo by mělo i při uložení dat na cloud zůstat a měl by si ho vlastník dat zkontrolovat při podepisování smlouvy s poskytovatelem cloudových služeb (Mather, Kumaraswamy, Latif, 2009).

Klasifikace dat a bezpečnostní stupně

Procedury a zásady klasifikace dat slouží k definici kategorií a kritérií, které by měla organizace při klasifikaci dat využívat. Jakmile je klasifikační schéma vytvořeno, měly by být vytvořeny bezpečnostní standardy, které budou definovat správné zacházení s daty každé kategorie a standardy pro ukládání dat, které definují požadavky v rámci životního cyklu dat. Klasifikační schéma by mělo být srozumitelné a jednoduché natolik, aby každý zaměstnanec byl schopen určit, do které kategorie se daná data řadí (Rouse, 2007). Nejčastěji se pak jedná o následující kategorie:

- Kategorie 4: vysoce citlivé firemní a zákaznické údaje, které by v případě zveřejnění mohly způsobit společnosti peněžní či právní problémy
- Kategorie 3: citlivé interní data, která v případě zveřejnění mohou mít na společnost negativní dopad
- Kategorie 2: interní data, která nejsou určena pro veřejnost
- Kategorie 1: data, která mohou být volně šířena mezi veřejnost

V některých literaturách bývají kategorie 3 a 2 spojeny do jedné.

3.4 Příprava dat pro cloud

Klasifikace dat představuje základ pro určování a následně přidělování relevantních hodnot k datům. Proces klasifikace dat pak pomáhá organizacím kategorizovat jejich uložená data dle citlivosti a celkového dopadu na společnost a zároveň bere v potaz riziko spjaté s danými daty. Co se terminologie týče, pro klasifikaci dat se využívá následujících dvou modelů přičemž praktická část se bude řídit terminologií prvního modelu.

Citlivost	**Model 1**	**Model 2**
Vysoká	Důvěrná	Omezená
Střední	Pro interní potřebu	Citlivá
Nízká	Veřejná	Neomezená

Tabulka 1: Terminologie klasifikace dat (Simorjay, 2014)

- **Důvěrná data:** Mezi tato data jsou řazena data, která by měla v případě ztráty či zneužití katastrofický dopad na firmu či jednotlivce. Tyto data mohou zahrnovat: osobní data, finanční záznamy, obchodní materiály, právní data, data obsahující ověřovací údaje (Simorjay, 2014).
- **Data pro interní potřebu:** Tato data mají střední citlivost a zahrnují se sem data, která by na firmu či jednotlivce v případě ztráty, či zneužití neměla tak veliký dopad. Tato data mohou zahrnovat: emaily, dokumenty a složky, které neobsahují citlivé údaje. V podstatě sem lze zahrnou všechna nedůvěrná data (Simorjay, 2014).
- **Veřejná data:** Jak již název napovídá, tato data nepředstavují pro firmu či jednotlivce žádnou hrozbu v případě jejich ztráty či zneužití. Patři sem například data, která byla na veřejnost vystavena například z marketingových důvodů. Do této skupiny patří i spamové e-maily (Simorjay, 2014).

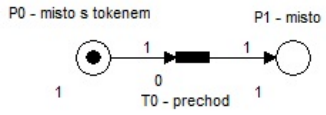
3.5 Petriho sítě

Petriho sítě představují grafický a modelovací nástroj použitelný v mnoha systémech. Slouží k popisu systémů, které jsou definovány jako souběžné, asynchronní, distribuované, paralelní, nedeterministické a/nebo stochastické. Tyto charakteristiky budou nyní blíže vysvětleny. Souběžný systém je schopný komunikovat s ostatními

systémy, zatímco sám provádí jiné činnosti. Asynchronní systémy nečekají na dokončení požadavku a jsou schopny dále pokračovat ve své činnosti. Distribuované systémy provádějí několik procesů zaráz přičemž tyto procesy spolu komunikují pomocí zpráv. Paralelní systémy jsou schopny paralelně zpracovávat více samostatných úloh. Nedeterministický systém je takový systém, u nějž pro každý vstup existuje několik možných výstupů a výstup tak nelze předpokládat. Stochastický systém je pak opakem deterministického systému a probíhají zde stochastické (náhodné) procesy (Murata, 2002).

V grafickém vyjádření slouží Petriho sítě jako vizuálně komunikační pomůcka stejně tak, jako slouží například diagramy toků. Matematické vyjádření zas představuje stavové rovnice, matematické rovnice a další matematické modely popisující chování systému. Petriho sítě mají díky jejich generalizaci a shovívavosti velké pole využití. V podstatě můžou být aplikovány na libovolnou aplikaci, či systém, který je možné popsat diagramem toků. Problémem mohou ovšem být velké sítě, které je potom složité analyzovat. U takových sítí je potom třeba zavést jisté omezení a modifikace za účelem zjednodušení (Murata, 2002).

Petriho síť představuje graf, ve kterém existuje počáteční uzel (počáteční stav). Tento graf je orientovaný (graf, jehož množina hran je uspořádaná), ohodnocený (graf, jehož hrany jsou ohodnoceny) a bipartitní (graf, jehož množinu uzlů je možné rozdělit na dvě disjunktní množiny tak, že mezi nimi nevede žádná hrana) a skládá se ze dvou typů uzlů – místa a přechody. Místa jsou pak značeny kruhy, přechody čtverci/obdélníky. Hrany mohou vést buď od místa k přechodu, nebo od přechodu k místu a jsou ohraničeny váhami, což jsou zde celá kladná číslo. Ke každému místu je poté přiřazen stav (značení). Pokud je toto značení nezáporné číslo k, potom říkáme, že místo je značeno k tokeny. Do daného místa je poté zaznačeno k černých teček představujících tokeny.



Obrázek 1: Základní značení v Petriho sítích

V modelování je poté často využíván koncept podmínek a událostí, kde místa reprezentují podmínky a přechody události. U podmínek platí, že pokud se v místě (podmínce) nachází token, je podmínka pravdivá (Murata, 2002). Jedná se pak o C/E Petriho sítě, o kterých budeme mluvit později.

Petriho síť je pětice PN = (P, T, F, W, M_0) kde:

P = $\{p_1, p_2, ..., p_m\}$ je konečná množina míst,
T = $\{ t_1, t_2, ..., t_m\}$ je konečná množina přechodů,
$F \subseteq (P \times T) \cup (T \times P)$ je konečná množina hran,
$W : F \to \{1, 2, 3, ...\}$ je váhová funkce,
$M_0 : P \to \{0, 1, 2, 3, ...\}$ je počáteční značení,

$P \cap T = \emptyset$ a $P \cup T = \emptyset$.

U mnohých systémů lze na základě jejich stavů a změn popsat jejich chování. Pro uskutečnění přechodu jsou v Petriho sítích stanovena následující pravidla:

- Přechod t je považován za uskutečnitelný v tom případě, když každé vstupní místo p tohoto přechodu je značeno minimálně w(p, t) tokeny, kde w(p, t) je váha hrany vedoucí z p do t. Přechod může být uskutečněn tedy pouze, pokud jsou všechny vstupní hrany nasyceny.
- I když je přechod považován za uskutečnitelný, nemusí se uskutečnit. Záleží totiž na tom, zdali v tomto místě nastala potřebná událost.
- Uskutečnění přechodu t odstraní w(p, t) tokenů z každého vstupního místa p přechodu t a přidá w(t, p) tokenů každému výstupnímu místu p přechodu t , kde w(t, p) je váha hrany vedoucí z t do p. Opět tedy platí, že hrany (v tomto případě výstupní) musí být nasyceny.

Přechod, který nemá vstupní místa se nazývá zdrojový přechod a naopak přechod, který nemá žádné výstupní místa, se nazývá norový. Vzhledem k absenci vstupních hran je zdrojový přechod vždy uskutečnitelný. Norový přechod pak naopak zkonzumuje všechny tokeny a žádné dále neprodukuje. Další speciální případ je pak smyčka (self-loop). Jedná se o takové místo, které je zároveň vstupním i výstupním místem přechodu. Petriho síť, která neobsahuje smyčky, se nazývá čistá (Murata, 2002).

Vlastnosti Petriho sítí

První vlastností je **životnost**. Petriho síť je považována za živou v případě, že jsou všechny její přechody živé. Živý přechod je takový přechod, který může být proveden vícekrát. Je definováno několik stupňů životnosti – mrtvý (L0-live, přechod nikdy nebude proveden), L1-live (přechod bude proveden alespoň jedenkrát), L2-live (mějme dané číslo $k \in N$, přechod pak bude uskutečněn minimálně k-krát), L3-live (přechod je proveden nekonečněkrát), L4-live (přechod je stupně L1-live pro každé značení $M \in R$, tedy je proveden při každém kroku). Tyto stupně se také nazývají hladiny životnosti. Životnost je také spojována s neexistenci uváznutí (deadlocku) v síti. Síť neobsahuje uváznutí pokud je pomocí každého dosažitelné značení možno dosáhnout jistého přechodu (Desel, Esparza, 1995).

Další vlastností je **omezenost**. Značení míst v Petriho sítích může být buď neomezené (může nabývat libovolné hodnoty), nebo omezené (může obsahovat maximálně pouze daný počet značek — tokenů). Petriho síť je omezená, pokud jsou všechna její místa omezená. Omezenost tedy znamená, že v každém stavu sítě existuje v každém místě maximálně k značení (tokenů), a to včetně stavu vstupního (Desel, Esparza, 1995).

S omezeností souvisí **bezpečnost** sítě. Pokud je každé místo Petriho sítě bezpečné, pak je i síť označena jako bezpečná. Bezpečnost sítě znamená, že počet značek v každém místě je menší nebo rovno 1. Bezpečnost je tedy případem omezenosti (Desel, Esparza, 1995).

Konzervativnost nám zaručuje, že se počet značek v sítích při provádění přechodů nemění. Značky tedy neubývají ani nepřibývají (Desel, Esparza, 1995).

Důležitým pojmem v rámci Petriho sítí je **dosažitelnost**, teda zda je dané místo dosažitelné z jakéhokoliv dosažitelného značení. Problém dosažitelnosti patří mezi základní problémy Petriho sítí. S tímto problémem souvisí problém pokrytí. Je zde řešeno, zda je možné dosáhnout značení, jehož složky jsou větší než daná hodnota. Pokrytí je tedy další vlastností Petriho sítí (Češka, 2015).

Typy Petriho sítí

- **P/T Petriho sítě:** jedná se o sítě složené z míst (place), označené kruhy, přechodů (transition), označené čtverci/obdélníky, a orientovaných hran. Udáním kapacity místa definujeme maximální počet tokenů, který se může v daném místě nacházet. Udáním váhy hrany definujeme maximální počet tokenů, který může hrana pojmout. Počáteční značení nám pak udává počet tokenů pro každé místo v síti. Hrana bez hodnocení se nazývá jednoduchá a má kapacitu 1, ovšem místo bez ohodnocení má neomezenou (nekonečnou) kapacitu. Kapacita místa a váha hrany se v teorii grafů nazývá ohodnocení uzlů a hran. Stav sítě je pak určen počtem tokenů v každém místě (Markl, 2006).
  Jak již bylo psáno výše, k uskutečnění přechodu v P/T Petriho sítích jsou definována pravidla, která jsou v některých zdrojích označována jako evoluční. Přechod je proveditelný, jestliže:
  - Každé vstupní místo přechodu obsahuje minimálně tolik tokenů, kolik je váha hrany vedoucí z daného místa do daného přechodu.
  - Každé výstupní místo přechodu obsahuje počet tokenů zvětšený o násobnost hrany vedoucí z přechodu do dané hrany a zároveň nepřevyšuje kapacitu tohoto místa.

  Po provedení přechodu se změní značení sítě. Počet tokenů v každém vstupním místě daného přechodu se zmenší o váhu hrany spojující dané místo a přechod a počet tokenů v každém výstupním místě daného přechodu se zvětší o váhu hrany spojující toto místo s přechodem. V každém případě je tedy hrana nasycena (Markl, 2006).
  Speciálním případem P/T Petriho sítí jsou C/E Petriho sítě, Obyčejné Petriho sítě (kapacita každého místa je nekonečná a kapacita každé hrany je 1) a Elementární Petriho sítě (každý přechod má alespoň jedno vstupní a alespoň jedno výstupní místo a zároveň se jedná o čistou síť, což je síť bez smyček) (Markl, 2006).
- **P/T (Place/Transition) Petriho sítě s inhibičními hranami (inhibitory):** Jedná se o síť, ve které se mohou vyskytovat tzv. inhibiční hrany. Inhibiční hrana je specifickým případem hrany grafu, směřuje pouze od míst k přechodům a je zobrazována orientovanou úsečkou zakončenou kroužkem. Hlavním rozdílem oproti klasickým hranám je, že po inhibičních hranách se nepřesouvají tokeny. Jedná se o negativně pojaté testovací hrany. Výpočetní síla Petriho

sítě s inhibičními hranami je srovnatelná s výpočetní silou Turingova stroje. Inhibiční hrany upravují proveditelnost přechodů a při tom neovlivní následné značení (Markl, 2006):

- Místo patří do vstupní inhibiční množiny, jestliže z daného místa vede inhibiční hrana do daného přechodu.
- Přechod je proveditelný, jestliže pro každé vstupní místo platí, že obsahuje méně tokenů, než kolik činí váha hrany vedoucí z daného místa do dané hrany.
- Po uskutečnění přechodu se změní značení stejným způsobem jako v příslušné P/T síti.

P/T Petriho sítě s prioritami: Ke každému přechodu sítě je přiřazeno číslo udávající prioritu přechodu. Přechod je pak povolen, je-li proveditelný v odpovídající P/T síti bez priorit a je proveditelný, jestliže je povolen a žádný jiný povolený přechod nemá vyšší prioritu. Změna stavu je pak totožná s danou P/T sítí (Markl, 2006).

- **C/E (Condition/Event) Petriho sítě:** Jak již bylo zmíněno výše, jedná se o sítě složené z podmínek (condition), označené kruhy, událostí (event), označené čtverci/obdélníky, a orientovaných hran. Značení (tokeny) zde vyjadřují logickou hodnotu podmínek, tedy pokud je token přítomen, je podmínka pravdivá, pokud není, je podmínka nepravdivá. Každá podmínka je teda splněna, nebo nesplněna. Ke změnám stavů podmínek dochází uskutečňováním událostí. Rozlišujeme zde dva typy podmínek — vstupní a výstupní.
  - Podmínka c je vstupní podmínkou (precondition) události e, jestliže vede hrana od podmínky c k události e. Podmínka tedy platí při provedení události.
  - Podmínka c je výstupní podmínkou (postcondition) události e, jestliže vede hrana od události e k podmínce c. Podmínka tedy začíná platit až po provedení události.

  Je tedy zřejmé, že podmínka nemůže být ovlivněna danou událostí, pokud s ní není spojena hranou. Pro provedení události je třeba, aby všechny její vstupní podmínky byly splněny a všechny výstupní byly nesplněny. Taková událost se pak nazývá proveditelná. Po uskutečnění proveditelné události jsou všechny její výstupní podmínky splněny a naopak všechny její vstupní podmínky nesplněny. Celkový stav sítě je zadán pomocí množiny podmínek, které jsou v dané chvíli splněny.

  V těchto sítích může také nastat konflikt, a to tehdy, když existují dvě události, které jsou v určitém okamžiku proveditelné, ovšem mají jisté vstupní podmínky společné. Provedení těchto událostí se pak vylučuje a události jsou tak v konfliktu.

  Speciálním případem C/E Petriho sítí je konečný automat, kde každá událost má pouze jednu vstupní a jednu výstupní podmínku. V celé síti se pak nachází pouze jeden token určující aktuální stav automatu. Dalšími speciálními případy C/E Petriho sítí jsou tzv. čisté sítě (sítě bez smyček) a elementární sítě

(sítě, kde každá událost má alespoň jednu vstupní a alespoň jednu výstupní podmínku a zároveň se jedná o čistou síť) (Markl, 2006).

- **Časované Petriho sítě (Timed Petri Nets — TPN):** U předchozích typů sítí bylo předpokládáno, že provedení přechodu je okamžité. Provedení přechodu bylo závislé pouze na jeho umístění v síti. Trvání dějů (změny stavů) lze charakterizovat: deterministicky (časy jsou konstanty), stochasticky (časy jsou náhodné), kombinovaně. Časové charakteristiky mohou být spojeny s libovolnými prvky Petriho sítě tedy přechody, místy, hranami, tokeny. Čas je pak reprezentován dvěma reálnými čísly, kde první znační minimální a maximální čas pro provedení (Markl, 2006).
- **Barevné Petriho sítě (Coloured Petri Nets — CPN):** Místo jednoho druhu tokenů je zde několik druhů. Každý token spadá do jisté třídy (třídy barev) a tokeny téže třídy pak považujeme za tokeny stejného typu. Každému místu je pak přiřazen typ tokenů, které se mohou v daném místě nacházet. Pomocí inskripčních jazyků pak lze vyjádřit také podmínky nad barvami (Markl, 2006).
- **Hierarchické Petriho sítě (Hierarchical Petri Nets — HPN):** U klasického jednoúrovňového způsobu návrhu riskujeme ztrátu přehledu, špatné zobrazení vnitřní struktury systému, či pracný návrh a následnou malou spolehlivost navrženého systému. Hierarchický způsob návrhu tyto problémy odstraňuje. Síť zde představuje množina nehierarchických Petriho sítí (stránek). Stránka A je pak podstránkou B, jestliže A rozvíjí některý prvek z B. Jsou zde pak využívány hierarchizační konstrukty: substituce přechodů, substituce míst, volání přechodů, slučování přechodů a slučování míst (Markl, 2006).
- **Objektové Petriho sítě (Object Oriented Petri Nets — OOPN):** Tokeny v těchto sítích představují instance objektových tříd a přechody představují metody aplikované na objekty. Tokeny pak skrze přechody většinou pouze procházejí a neničí se tak vstupní tokeny, ani se nevytváří výstupní. Hrany popisují možné toky objektů v systému. Tato síť může být klasická, nebo hierarchická (Markl, 2006).

4 Bezpečnost dat

Definice bezpečnostní architektury dle RFC zní „Bezpečnostní architektura je soubor principů popisujících: bezpečnostní služby, které daný systém musí poskytovat, aby splňoval požadavky uživatelů, systémové prvky nezbytné pro implementování služeb a výkonnostní stupně vyžadované pro dané prvky pro možnost vyrovnání se s hrozbami v daném prostředí.“ (IETF, 2000)

Typickým představitelem je bezpečnostní architektura OSI popisující základní bezpečnostní služby, bezpečnostní mechanismy a vztahy mezi nimi. Také popisuje promítání bezpečnostních služeb na danou síťovou architekturu a umístění těchto služeb v rámci OSI modelu (Oppliger, 2009).

Architektura OSI rozlišuje mezi pěti třídami bezpečnostních služeb – autentizace, kontrola přístupu, důvěrnost dat, integrita dat a nepopiratelnost služeb. Architektura dále vymezuje bezpečnostní mechanismy určené pro implementaci daných služeb jako je například šifrování, nebo mechanismus kontroly přístupu. Typické bezpečnostní protokoly jsou SSL/TLS ovšem tyto protokoly nepředepisují žádný bezpečnostní mechanismus a vše závisí na konkrétní implementaci (Oppliger, 2009).

Při zabezpečování sítě a komunikace v síti je třeba brát zřetel na všechny vrstvy komunikace. Základní protokoly samy o sobě nezajišťují zabezpečení přenášených dat a proto je třeba využívat protokoly zajišťující jejich bezpečnost (Oppliger, 2009). V praxi se v současné době dává přednost modelu TCP/IP před modelem OSI a proto i popis bezpečnostních prvků na jednotlivých vrstvách bude vztažen k TCP/IP modelu.

- **Síťová vrstva:** Na této vrstvě je bezpečnost představována MAC (media access control) security, která zajišťuje MAC rámcům autentizaci, důvěrnost a integritu dat. Dalšími bezpečnostními prvky jsou zde VPN (virtual private network) sítě a bezpečnostní protokoly PPTP (point-to-point tunneling protocol), či L2TP (layer 2 tunneling protocol) (Oppliger, 2009).
- **Internetová vrstva**: V rámci této vrstvy zajišťují bezpečnost především protokoly IP security (IPsec) a Internet Key Exchange (IKE), určené k navázání bezpečného spojení mezi komunikujícími stranami. IPsec/IKE jsou schopné zabezpečit veškeré internetové aplikace (Oppliger, 2009).
- **Transportní vrstva:** Zde je možné využít kryptografické techniky v kombinaci s HTTP a poskytnout základní bezpečnost pro webové transakce. Na této vrstvě jsou umístěny také protokoly SSL/TLS. V pozdější době byla v rámci zlepšení bezpečnosti mezi aplikační a transportní vrstvu přidána Secure Socket Layer (SSL), jejímž úkolem je navázat bezpečné spojení a skrze toto spojení přenášet data. Tato její funkcionalita je ovšem velice podobná protokolům transportní vrstvy a proto bývá SSL zahrnováno do transportní vrstvy (Oppliger, 2009).
- **Aplikační vrstva:** Na této vrstvě je opět možné využít HTTP v kombinaci s kryptografickými technikami, nebo systém pro ověřování a distribuci klíčů (Oppliger, 2009).

4.1 Secure Socket Layer (SSL)

SSL je klient/server protokol umožňující aplikacím bezpečný přenos dat mezi komunikujícími stranami za pomoci autentizačních služeb, kontroly důvěrnosti a integrity přenosu. Využívá šifrování pomocí veřejného klíče, nezajišťuje nepopiratelnost služeb a je „socketově“ orientovaný, což znamená že všechna data přijatá/odeslaná do/z daného socketu jsou šifrována stejným způsobem. K zajištění správného příjemce zprávy je využívána technologie PKI (public-key infrastructure). SSL je stavový protokol, což znamená, že klient i server se vždy musí nacházet v jistém stavu a musí mít jistou stavovou informaci. Základním rozdělením stavů je stav aktuální a stav čekající, které se dále dělí na stav čtení a zapisování. Protokol rozděluje data na menší jednotky — fragmenty, s kterými poté dále pracuje individuálně. Každý fragment pak může být zkomprimován, je autorizován pomocí MAC a zašifrován. Následně je k němu přidána hlavička (header) a je odeslán příjemci. Příjemce poté každý fragmet dešifruje, ověří, dekomprimuje a opět spojí do jedné zprávy. Až poté je zpráva předána vyšší vrstvě (Oppliger, 2009).

Protokol SSL je logicky možné rozdělit na dvě části. První část je umístěna nad spolehlivým protokolem transportní vrstvy, jako je např. TCP a je zde vykonávána první část představující SSL Record Protocol, zajišťující např. zapouzdření dat. Nad SSL Record Protokolem se pak nachází druhá část, skládající se že čtyř protokolů:

- SSL Handshake Protocol: Je jádrem celého SSL a umožňuje komunikujícím stranám provést vzájemné ověření, vyjednání typu šifrování a případné komprimace.
- SSL Change Cipher Spec Protocol: Umožňuje komunikujícím stranám upozornit na případnou šifrovací změnu.
- SSL Alert Protocol: Umožňuje komunikujícím stranám upozornit na potenciální problémy a výměnu výstražných zpráv.
- SSL Application Data Protocol: Má za úkol předání dat vyšší vrstvě (Oppliger, 2009).

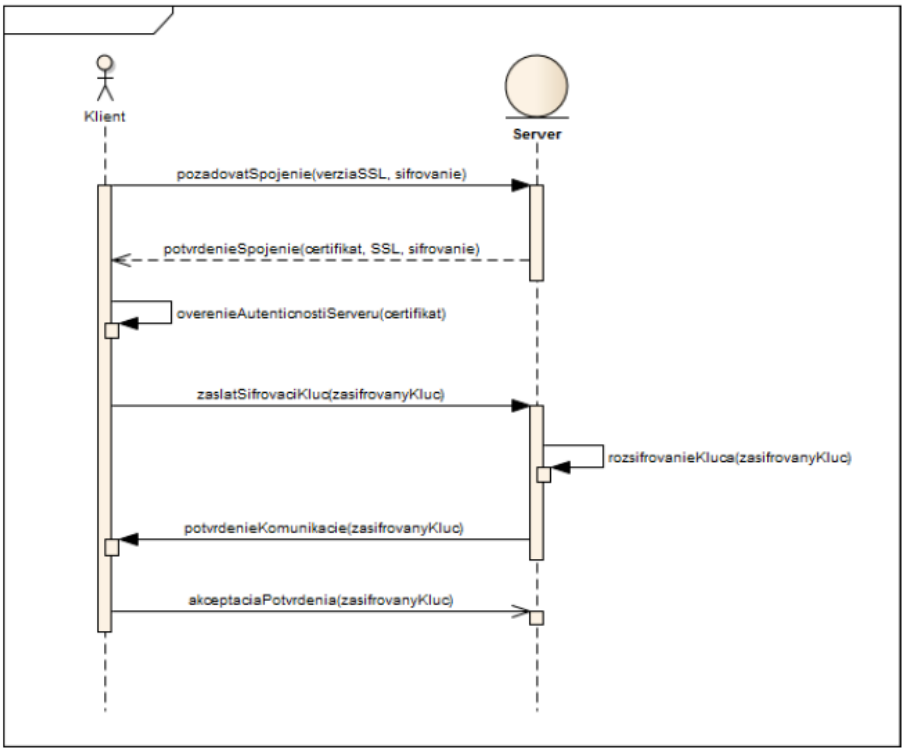
SSL protokol zakládá a využívá SSL spojení a SSL relace, přičemž spojení i relace jsou stavové a protokol tedy pro ně drží dané informační prvky. SSL spojení je využíváno pro skutečný přenos dat mezi komunikujícími stranami. SSL relace odkazuje na spojení mezi komunikujícími stranami vytvořené SSL Handshake protokolem a definuje šifrovací a jiné parametry využívané SSL spojením. V jedné relaci pak může být zahrnuto jedno a více spojení (Oppliger, 2009).

Před využitím šifrovacích technik je vždy stanoven „klíčový materiál“. Pro výměnu klíčů jsou na výběr tři algoritmy generující 48-bytový klíč tzv. premaster secret a to: RSA, Diffie-Hellman a FORTEZZA.

- **RSA:** Při použití tohoto algoritmu klient vygeneruje premaster secret, zašifruje ho veřejným klíčem serveru a odešle zašifrovaný text serveru. Server poté využije soukromý klíč k dešifrování premaster secret.
- **Diffie-Hellman:** Tento algoritmus využívá Diffie-Hellman výměnu klíčů a výsledná Diffie-Hellman hodnota představuje premaster secret.

- **FORTEZZA:** U tohoto algoritmu proces výměny klíčů získá tzv. TEK, který umožňuje bezpečně přenášet náhodně zvolený premaster secret (Oppliger, 2009).

Jakmile je stanoven premaster secret, je možné ho využít pro tvorbu tzv. master secret, což je 48-bytové speciální číslo sdílené mezi klientem a serverem. Master secret také reprezentuje jeden z SSL relačních stavových prvků (Oppliger, 2009).



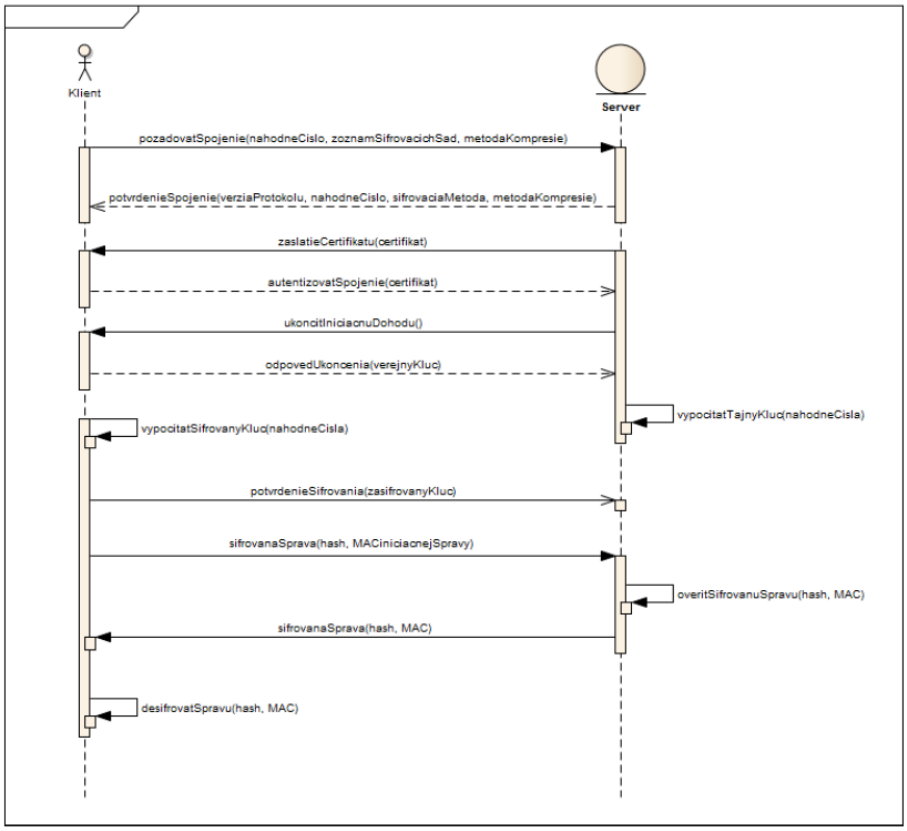
Obrázek 2: Inicializace spojení mezi klientem a serverem při použití protokolu SSL (Priščáková, 2015)

4.2 Transport Layer Security (TLS)

TLS je nástupcem protokolu SSL. Také se jedná o klient/server protokol, který se nachází nad spolehlivým protokolem transportní vrstvy a skládá se ze stejných vrstev a protokolů jako protokol SSL. Nejvyužívanější verzí tohoto protokolu je v současné době TLS 1.2 (Oppliger, 2009).

Na spodní vrstvě se nachází protokol TLS Record Protocol, který opět fragmentuje, komprimuje a šifruje data. TLS record se skládá z hlavičky (header) obsahující typ protokolu, verzi a délku dat a samotných dat. Každý TLS record má navíc přidělené unikátní 64-bitové číslo, které ovšem není odesíláno druhé straně. Vyšší vrstva zahrnuje opět protokoly TLS Change Cipher Spec Protocol, TLS Alert Protocol, TLS Handshake Protocol a TLS Application Data Protocol. Každý protokol má jednoznačně identifikován typ svého obsahu, přičemž je možné definovat typy

nové. Stavy jsou totožné s protokolem SSL tedy stav aktuální a čekající a u každého dále stav čtení a zápis (Oppliger, 2009).



Obrázek 3: Inicializace spojení mezi klientem a serverem při použití protokolu TLS (Priščáková, 2015)

Vzhledem k vysoké podobnosti s SSL budou nyní uvedeny již pouze rozdílnosti mezi SSL a TLS protokolem. TLS rozlišuje mezi bezpečnostními parametry a stavovými elementy, zatímco SSL toto rozlišení nedělá. Dalším rozdílem je způsob generování klíčů. Kromě RSA jsou zde využívány dvě podoby Diffie-Hellmanova algoritmu v kombinaci s RSA, a to Ephemeral Diffie Hellman (DHE) a Ephemeral elliptic curve Diffie-Hellman (ECDHE), který se od klasického Diffie Hellman liší využitými matematickými základy (Ristić, 2014).

Základní jádro protokolů ovšem zůstalo téměř stejné. První změnu od SSL 3 přineslo TLS 1.1 opravou bezpečnostních problémů. Následovalo TLS 1.2, které zavedlo autentizované šifrování a využívání bezpečnějšího hashování (Ristić, 2014). Jak již bylo řečeno, TLS 1.2 je v současné době nejvyužívanější verzí tohoto protokolu.

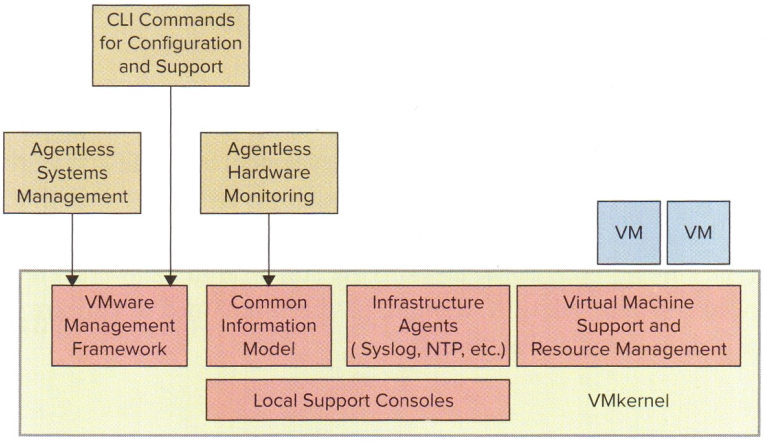
5 Srovnání virtualizačních prostředí

V dnešní době existuje, co se řešení virtualizace týče, několik možností, od operačních systémů umožňující uživateli systém rozdělit na několik (např. Sun Solaris Zones, BSD, PowerVM), přes systémy, které virtualizují operační systém, který může být sdílen mezi všemi uživateli (např. Parallels Virtuozzo), po serverovou virtualizaci, kterou se bude tato část zabývat.

5.1 VMware ESX

Společnost VMware byla první, která přišla s komerční serverovou virtualizací. Jejich první produkt umožňoval práci s virtuálními stroji na klientech s operačními systémy Windows a Linux. Dalšími produkty byly ESX 1.0 a GSX 1.0. Jedná se o hypervizory typu 1 (ESX 1.0) a typu 2 (GSX 1.0) a tyto řešení jsou dostupná dodnes a jsou stále vylepšována, přičemž GSX nyní nese název VMware Server a je dostupný zadarmo. Originální ESX bylo tvořeno hypervizorem a konzolí založenou na Linuxu. Od tohoto modelu bylo ovšem upuštěno, neboť konzole byla značně větší než hypervizor (přibližně 30x) a hrozila zde možnost útoků na hypervizor právě skrze konzoli. Nástupce ESX – ESXi tedy ponechat stejný hypervizor, ovšem již nebyla obsažena konzole a hypervizor je ovládán klasicky skrze příkazový řádek, či přes integrace do aplikací třetích stran (Portnoy, 2012).

Jedním ze základních rozdílů mezi VMware a ostatními řešeními je, že základem modelu VMwaru je VMkernel, obsahující všechny procesy nezbytné k podpoře virtuálních strojů a správě a dostupnosti hardwarových zdrojů. Navíc poskytuje služby jako logování, integrace s VMware nástroji a moduly třetích stran (hardwarové ovladače a monitorovací nástroje) (Portnoy, 2012).



Obrázek 4: ESX architektura (Portnoy, 2012)

V roce 2003 představila společnost produkt VMotion, který za běhu umožňuje migraci virtuálního stroje z jednoho fyzického stroje na jiný bez nutnosti přeru-

šit chod operačního systému či aplikací zde běžících (Portnoy, 2012). VMware se v dnešní době drží na prvních příčkách trhu za čímž stojí několik výhod této platformy.

VMware virtualizace je placená, ovšem nabízí uživateli, oproti ostatním platformám, rozsáhlejší možnosti (VMware High Availability, VMotion, Storage VMotion, Update Manager, Distributed Resource Scheduler, Distributed Power Manager). VMware poskytuje zákazníků vysokou dostupnost a spolehlivost, kterou zajišťuje oněmi rozsáhlejšími možnostmi:

- **VMware Hight Availability (HA):** V případě výpadku hosta je virtuální stroj automaticky nastartován na jiném hostu. Odstávka virtuálního serveru pak trvá pouze tak dlouho, jak dlouho trvá nastartování operačního systému včetně příslušných služeb.
- **VMotion:** Umožňuje migraci běžícího virtuálního stroje mezi jednotlivými hosty (využití především pro případy údržby, správy).
- **Storage VMotion:** Slouží k přesunu virtuálních disků z hosta do úložné sítě SAN (Storage Area Network) a pro přesun disků ze SAN do jiné sítě.
- **Update Manager:** Zajišťuje automatické aktualizace ESX serverů, či operačních systémů a aplikací běžících na virtuálních strojích s ohledem na co nejnižší nedostupnost dané aplikace.
- **Fault Tolerance (FT):** Zajišťuje, že v případě výpadku ESX hosta není zapotřebí restartování operačního systému běžícího na virtuálním stroji.
- **Site Recovery Manager (SRM):** Umožňuje automatizovat a testovat plán obnovy po havárii.

Další výhodou je vysoká hardwarová a aplikační kompatibilita produktů VMware. Aplikace běžící na hostu nemusí být kompatibilní s virtuálním serverem, neboť tyto aplikace neví, že jsou virtualizovány. Dostačuje klasická kompatibilita s daným operačním systémem běžícím na hostu (Davis, 2009).

Co se bezpečnosti týče, VMware disponuje bezpečnostním frameworkem VMsafe určeným pro prodejce třetích stran za účelem tvorby bezpečných produktů pro VMware infrastrukturu a produkt Vshield Zone určený pro VMware administrátory k tvorbě bezpečných zón mezi virtualizovanými hosty (Davis, 2009).

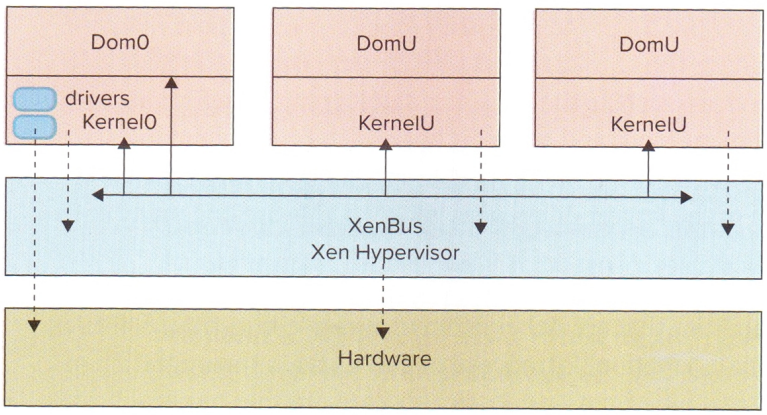
Neposlední výhodou je, že díky vysoké rozsáhlosti platformy VMware je o této problematice dostupné široké množství knih, článků, videí a celkově je tím i zajištěna rozsáhlejší podpora.

5.2 Citrix Xen

Hypervizor Xen začal být vyvíjen na počátku 90. let. V roce 2002 byl kód zpřístupněn jakožto open source projekt a každý se tak mohl podílet na jeho vylepšování. V dnešní době je Xen hypervizor součástí nabídky produktů společností Red Hat, Novell a Sun (Portnoy, 2012).

Hypervizor je umístěn přímo na hardwaru fyzického stroje, případně na „čistém kovu“, ovšem architekturou se od VMware liší. Model Xen obsahuje speciálního

hosta znaného Domain 0 (Dom 0), na kterém běží operační systém a který je nastartován (bootován) zároveň s nastartováním hypervizoru a na rozdíl od ostatních hostů, má tento host právo na správu všech vstupů a výstupů ostatních hostů. Také je tento host schopný obsluhovat ovladače hardwarového zařízení. Všechny žádost od hostů v Xen architektuře prochází skrz hypervizor a Domain 0, odkud je požadavek vyslán k požadovanému zdroji, který výsledek posílá opět skrze Domain 0 a hypervizor (Portnoy, 2012).



Obrázek 5: Xen architektura (Portnoy, 2012)

Xen technologie bývá označována jakožto nejrychlejší a nejbezpečnější a dokáže využívat plného potenciálu hardwaru společností Intel a AMD. Xen obsahuje také komplexní správu pomocí XenCenter, které umožňuje rychlou adaptaci virtualizace pro konsolidaci serverů, zajištění kontinuity provozu, vývoje a testování softwaru. Další vlastnotí je XenMotion, který obstarává stejnou funkcionalitu jako výše zmíněný VMotion, tedy umožňuje migraci běžících virtuálních strojů mezi fyzickými servery bez přerušení dodávky služeb. Pokud navíc na virtuálním stroji běží Citrix XenApp, stroje mohou být přesunuty pouhým „táhnutím“ bez toho, aniž by ovlivnili uživatele. XenApp dále umožňuje vytvářet z virtuálních strojů šablony, díky kterým může být nový stroj vytvořen během pár sekund (Kelar Pacific, 2008).

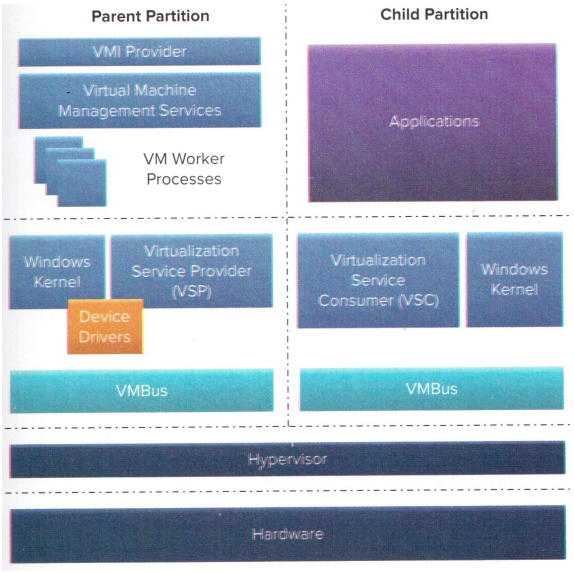
5.3 Microsoft Hyper-V

Prvním virtualizačním počinem společnosti Microsoft byl produkt Virtual Server. Virtual Server je hypervizorem typu 2 a v dnešní době je dostupný zadarmo. V roce 2008 byl na trh uveden Microsoft Hyper-V jako instalovatelná součást operačního systému Windows Server 2008 (Portnoy, 2012).

Hyper-V je hypervizor typu 1, tedy kód hypervizoru je umístěn přímo v hardwaru. V rámci Hyper-V jsou hosté nazývání „oddíly“ (partition). Stejně jako u modelu Xen zde existuje speciální oddíl, který má přímý přístup k hardwarovým zdrojům a jede na něm operační systém (Windows Server 2008). Tento oddíl vytváří a spravuje své potomky (oddíly) a spravuje systémové funkce a ovladače zařízení.

Ačkoliv Hyper-V funguje na operačním systému Windows Server, operační systémy virtuálních strojů mohou být libovolné (Portnoy, 2012).

Na rozdíl od platformy VMware, Hyper-V využívá techniku mikrokernel, což umožňuje ovladačům každého virtuálního stroje jet nezávisle na ostatních. Tato technika navíc přináší další vylepšení bezpečnosti, kterým VMware nedisponuje. Při případném útoku musí útočník napadnout každý virtualizovaný stroj zvlášť, není možné se jednoduše nabourat do hypervizoru pomocí jeho API (BackupAssis, 2013).



Obrázek 6: Hyper-V architektura (Portnoy, 2012)

Oddělenost virtuálních strojů s sebou přináší také zlepšení stability. V případě poruchy virtuálního stroje není narušena celá virtualizační platforma, ale pouze kritický host. V rámci Hyper-V Windows Serveru 2012 byla představena i nová funkcionalita — Hyper-V Replica. Tato funkcionalita umožňuje jednoduše klonovat hosty skrze WAN (wide area network) či bezpečné VPN (virtual private network) a zároveň vytváří náhradní pro případný výpadek (BackupAssis, 2013).

Spekulativní potom může být založení Hyper-V na Windows. Pro mnohé uživatele znamená platforma Windows známe prostředí a tudíž rychlejší zorientování se v celé virtualizační platformě. Výhodou pro uživatele operačního Windows Server je fakt, že Hyper-V je zahrnuto v rámci licence operačního systému a nejsou tedy vyžadovány další výdaje. Nevýhodou může být, že Hyper-V je dostupné pouze pro 64-bitové počítače a tudíž je operační systém Windows Server 2008 R2 nabízen pouze v 64-bitové verzi. Ovšem je třeba také brát v potaz, že Hyper-V je mladou virtualizační platformou a větší pokroky se tudíž dají teprve očekávat (BackupAssis, 2013).

5.4 Kernel-based Virtual Machine (KVM)

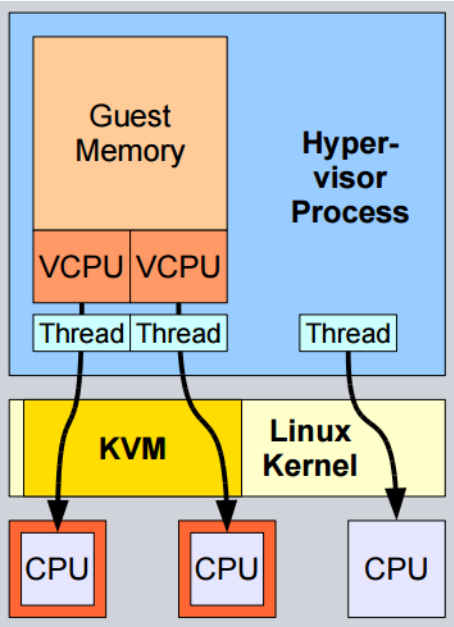
KVM je open source hypervizor založený na Linuxu, ovšem dostupný i pro systém Windows. KVM je součástí operačních systémů Red Hat Enterprise Linux a výš, SUSE Linux Enterprise Server a Canonical Ubuntu (IBM Corporation, 2012).

KVM poskytuje hostovaným systémům emulované, nebo para-virtualizované zařízení. Para-virtualizovaná zařízení jsou softwarovou implementací hardwarových zařízení, kde hypervizor a hostovaný operační systém využívají optimalizované vstupní a výstupní rozhraní pro efektivnější komunikaci. Příkladem je para-virtualizovaný ovladač v hostovaném operačním systému. Na rozdíl od emulovaných zařízení, para-virtualizovaná zařízení vyžadují modifikovaný operační systém, čímž se snižuje jejich kompatibilita. Para-virtualizované zařízení zajišťuje nižší latenci a vyšší propustnost vstupních a výstupních operací hostovaných operačních systémů. Pro para-virtualizované zařízení poskytuje KVM VirtIO API. VitIO API specifikuje mezi virtuálním stroje a hypervizorem rozhraní, které je na hypervizoru nezávislé. Para-virtualizované zařízení jsou vhodné především pro hostované operační systémy, které provádějí těžké vstupní a výstupní operace a aplikace (IBM Corporation, 2012).

Emulovaná zařízení bývají součástí plné virtualizace. V takovém případě pak hypervizor vytváří softwarovou implementaci hardwaru. V KVM poskytuje emulovaná zařízení modifikované QUEMU (hostovaný hypervizor provádějící virtualizaci hardwaru). Hypervizor pak zachycuje všechny vstupní a výstupní požadavky od hostovaného operačního systému a napodobuje operace reálného hardwaru. Hostovaný operační systém pak nerozpozná, že nekomunikuje s reálným hardwarem. Ačkoliv emulovaná zařízení disponují větší kompatibilitou, výkonnostně jsou nižší než para-virtualizovaná (IBM Corporation, 2012).

V KVM může jakožto lokální úložiště hostovaných operačních systémů sloužit blokové zařízení nebo soubory. Soubory jsou myšleny obrazy disků, neboť tyto soubory jsou dostupné pro hypervizory a představují velkokapacitní úložiště pro hostované operační systémy ve formě virtuálních hard disků. Velikost souboru pak představuje největší možnou velikost úložiště. Bloková zařízení ovšem bývají výhodnější. Při práci s blokovým zařízením je třeba průchod menším počtem vrstev než u souborů, a tím je zajištěna nižší latence a vyšší propustnost systému (IBM Corporation, 2012).

Jak již bylo řečeno, technologie KVM hypervizoru je založena na Linuxovém jádru. Každý systém hostovaný pomocí KVM je proces složený z několika vláken a vzhledem k tomu, že Linux je multi-taskový operační systém, je schopen přepínat provádění mezi procesy a může provádět více procesů zároveň za použití simultánního multi-processing (SMP) hardwaru (IBM Corporation, 2012).



Obrázek 7: KVM architektura (Kiszka, 2010)

Základní výhodou oproti ostatním virtualizačním technologiím je, že KVM je nabízeno zadarmo a dokonce je součástí některých Linuxových distribucí. KVM disponuje vysokým výkonnostním stupněm, kterým v některých případech převyšuje i VMware. KVM dokáže na jednom x86 serveru rozjet až 552 virtuálních strojů. Vzhledem k Linuxovému základu je KVM schopno využívat bezpečnostní vlastnosti Security Enhanced Linuxu (SELinux). Toto zabezpečení umožňuje KVM kontrolu přístupu k virtuálním strojům (Open Virtualization Alliance, 2015).

Další vlastností odvozenou z Linuxu je možnost QoS politiky, která umožňuje virtuálním strojům nastavit mezní hodnoty procesoru, paměti, sítě, vstupů a výstupů disku a garantované QoS pro daný virtuální stroj. Jak již bylo zmíněno, KVM je open source a tudíž není uživatel omezován, co se týká typu partnerů a řešení (Open Virtualization Alliance, 2015).

5.5 oVirt

oVirt je komplexní, multifunkční, flexibilní nástroj pro vytváření a správu virtualizační infrastruktury založený na Linuxu a Libvirt. Libvirt je nástroj pro správu virtualizační infrastruktury umožňující spravovat stroje hostované na Quemu/KVM, Xen, VirtualBox a LXC, přičemž oVirt se specializuje konkrétně na Quemu s KVM. oVirt je vhodný především v případech: kombinování různých typů hardwaru v rámci jedné virtualizační platformy, zajištění centrální správy virtuálních strojů skrze grafické uživatelské rozhraní, usnadnění správy většího množství virtuálních strojů, automatizace klastrování a vyrovnání zatížení virtuálních strojů, automatizace převzetí služeb v případě výpadku při živé migraci (Lesovsky, 2013).

oVirt je, stejně jako KVM, open source řešení a pracuje na operačních systémech CentOS nebo RHEL. Je základem pro Red Hat Enterprise Virtualization (RHEV), což je komerční virtualizační produkt založený na KVM. Platforma oVirt obsahuje následující komponenty (Lesovsky, 2013):

- **oVirt jádro (engine):** množina softwaru a služeb sloužící pro správu globální konfigurace virtualizační infrastruktury včetně správy virtuálních strojů, úložišť a síťových nastavení. Jedná se o jádro celé platformy a poskytuje rozhraní ostatním součástem infrastruktury.
- **oVirt uzly (node):** představují virtualizační jednotky jedoucí přímo na virtuálních strojích. Jedná se o Linuxové servery s nainstalovaným démonem libvirt a VDSM (Virtual Desktop and Server Manager).
- **Úložiště a síťová infrastruktura:** může se jednat o lokální nebo síťové úložiště jako je DAS, nebo NAS, nebo se může jednat o sítě úložišť (SAN — Storage Area Networks). Diskové jednotky v sobě mají uložené obrazy virtuálních strojů a obrazy nainstalovaných operačních systémů. Síťová zařízení pak propojují jádro, uzly a úložiště.

Platforma oVirt tedy umožňuje správu virtualizační infrastruktury, obsahuje nástroje pro migraci živých virtuálních strojů mezi fyzickými hosty a nástroje pro budování virtuálních strojů odolných v případech výpadku. Díky plánovači zdrojů je možné efektivně rozdělit zdroje mezi jednotlivé stroje, redukovat energii a stanovovat limity pro zdroje. Stejně jako u ostatních platforem je zde možné tvořit šablony virtuálních strojů, díky nimž pak nasazení nového stroje trvá pár sekund. Nevýhodou může být, že díky používání na operačních systémech Linux je vyžadována alespoň základní znalost Linuxové příkazové řádky (Lesovsky, 2013).

5.6 Porovnání

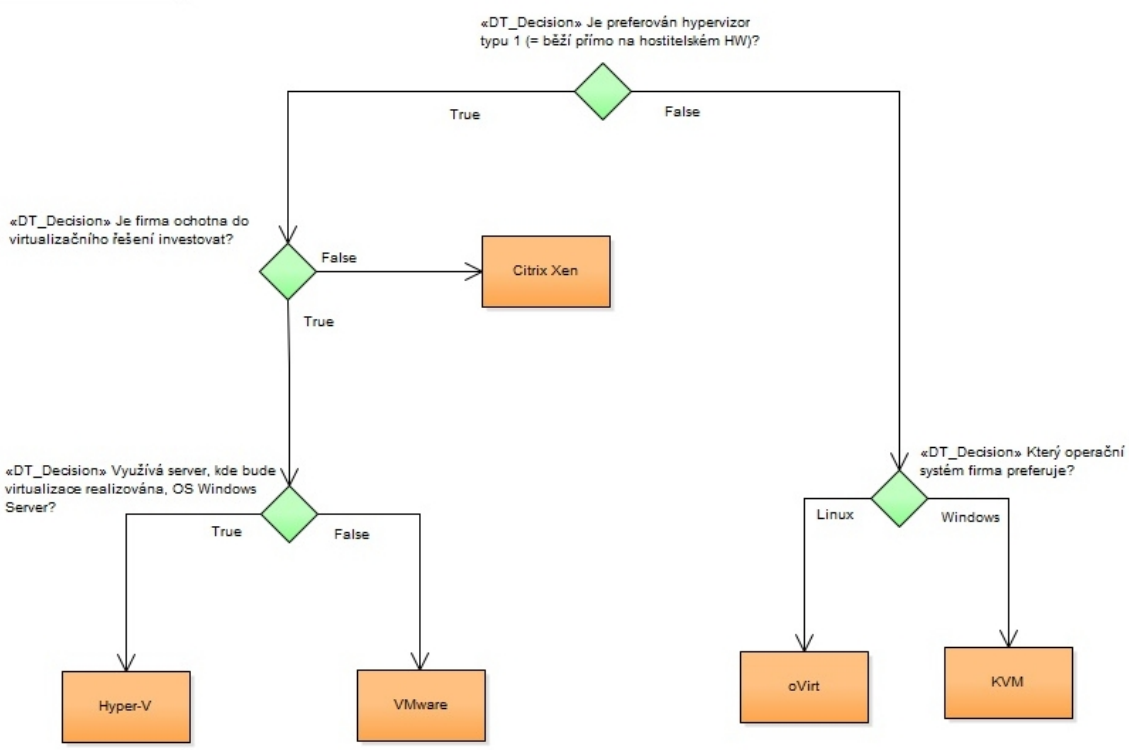
Všechny důležité aspekty budou nejdříve uvedeny v tabulce a následně bude z těch, které jsou pro společnost, či jednotlivce při rozhodování nejdůležitější, sestaven rozhodovací strom.

	Placené	**Bázový OS**	**Živá migrace**
VMware	Ano	-	Ano
Citrix Xen	Ne	-	Ano
Hyper-V	Ano	Windows Server	Ano
KVM	Ne	Windows/Linux	Ano
VMware	Ne	Linux	Ano

	Tvorba šablon	Oficiální podpora	Typ hypervizoru
VMware	Ano	Ano	1
Citrix Xen	Ano	U placené verze	1
Hyper-V	Ano	Ano	1
KVM	Ano	Ne	2
VMware	Ano	Ne	2

Tabulka 2: Vlastnosti virtualizačních technologií

Jak je možné vidět, živou migraci a tvorbu šablon poskytují všechny zkoumané virtualizační metody, proto nebudou do výsledného rozhodovacího stromu zahrnuty. Stejně tak nebude zahrnuta ani oficiální podpora, neboť bylo zjištěno, že by na výsledek rozhodovacího stromu neměla vliv.



Obrázek 8: Rozhodovací strom

Praktická část této práce je věnována především malým a středním firmám. Ideální je tedy produkt, který bude dostupný zadarmo a nebude tak vyžadovat dodatečné výdaje. Dále se dá předpokládat, že takovýmto společnostem bude více vyhovovat hypervizor typu 2, protože bude ve většině případů třeba virtualizovat pouze několik hostů a tudíž nebude třeba zvýšený výkon. Není jisté, zda budou všechny firmy ochotné k virtualizaci využívat operační systém Linux a proto je výslednou zvolenou virtualizační metodou KVM. KVM je dostupné zadarmo, jedná

se o hypervizor typu 2 a je dostupné jak pro operační systém Linux, tak pro operační systém Windows.

6 Instalace a modelování vybrané technologie

Jako virtualizační řešení vhodné pro malé a střední firmy bylo vybráno KVM. KVM bylo testováno na serveru s operačním systémem Debian, který bývá díky své jednoduchosti, rychlosti a rychlé odezvě s velkým výkonem využíván pro fyzické servery s virtualizací často. Počáteční konfiguraci je nutné provést skrze příkazovou řádku, proto je nutné, aby uživatel znal základy Linuxové příkazové řádky, pokud KVM instaluje právě na operační systém Linux.

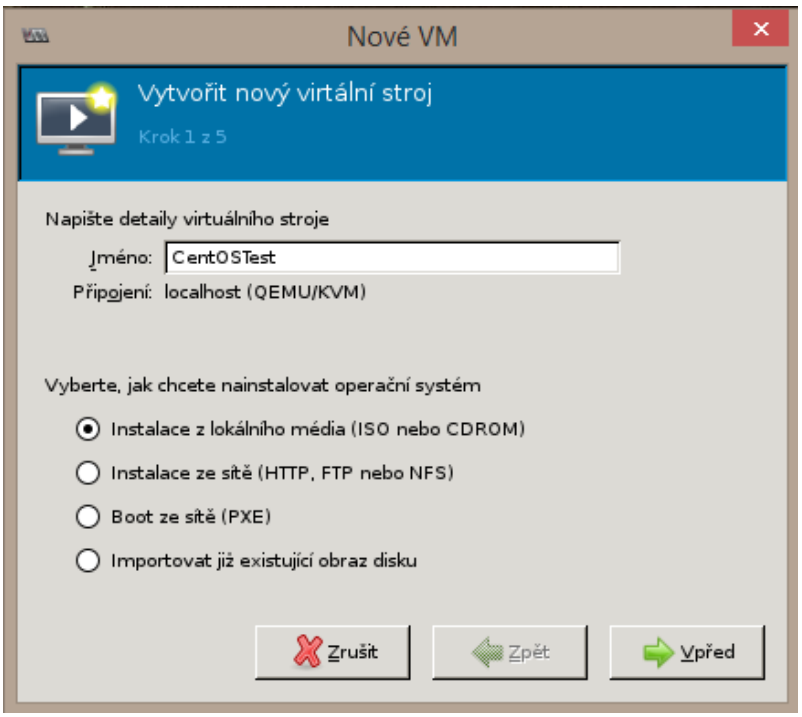
V prvním kroku je třeba zkontrolovat, zda fyzický stroj, na který bude KVM instalováno, obsahuje rozšíření pro virtualizaci, ať už Intel VT, nebo AMD-V. Poté je možné přikročit k samotné instalaci potřebných částí. Mezi tyto části patří qemu-kvm, libvirt-bin, virtinst a případně virt-manager, pokud chce uživatel spravovat virtuální stroje skrze grafické rozhraní. Po úspěšné instalaci je třeba přidat uživatele, kteří budou mít právo spravovat virtualizaci, do skupit kvm a libvirt. Dále je možné zvolit komunikaci skrze protokol TLS. Tato možnost vyžaduje upravit soubor /etc/libvirt/qemu.conf a umístění TLS certifikátu a soukromých klíčů. Pokud bude využíváno grafické rozhraní, je potřeba provést další úpravy. Následně je nutné restartovat službu libvirt. V dalším kroku je třeba nakonfigurovat bridge, který má za úkol propojení virtuálního síťového rozhraní s fyzickým síťovým rozhraním. Před samotnou konfigurací je nainstalován balíček pro správu bridge — bridge-utils a bridge samotný je vytvořen. Posledním krokem, co se bridge týče, je úprava souboru etc/network/interface do následující podoby:

```
1  # The primary network interface
2  auto eth0
3  iface eth0 inet manual
4
5  auto br0
6  iface br0 inet static
7          address 192.168.123.99
8          netmask 255.255.255.0
9          network 192.168.123.0
10         broadcast 192.168.1.255
11         gateway 192.168.123.165
12         dns-nameservers 8.8.8.8 8.8.4.4
13         bridge_ports eth0
14         bridge_stp off
```

Po úpravě souboru je nutné restartovat službu networking, aby se změny projevily. Po těchto krocích je možné přistoupit k samotné tvorbě virtuálního stroje.

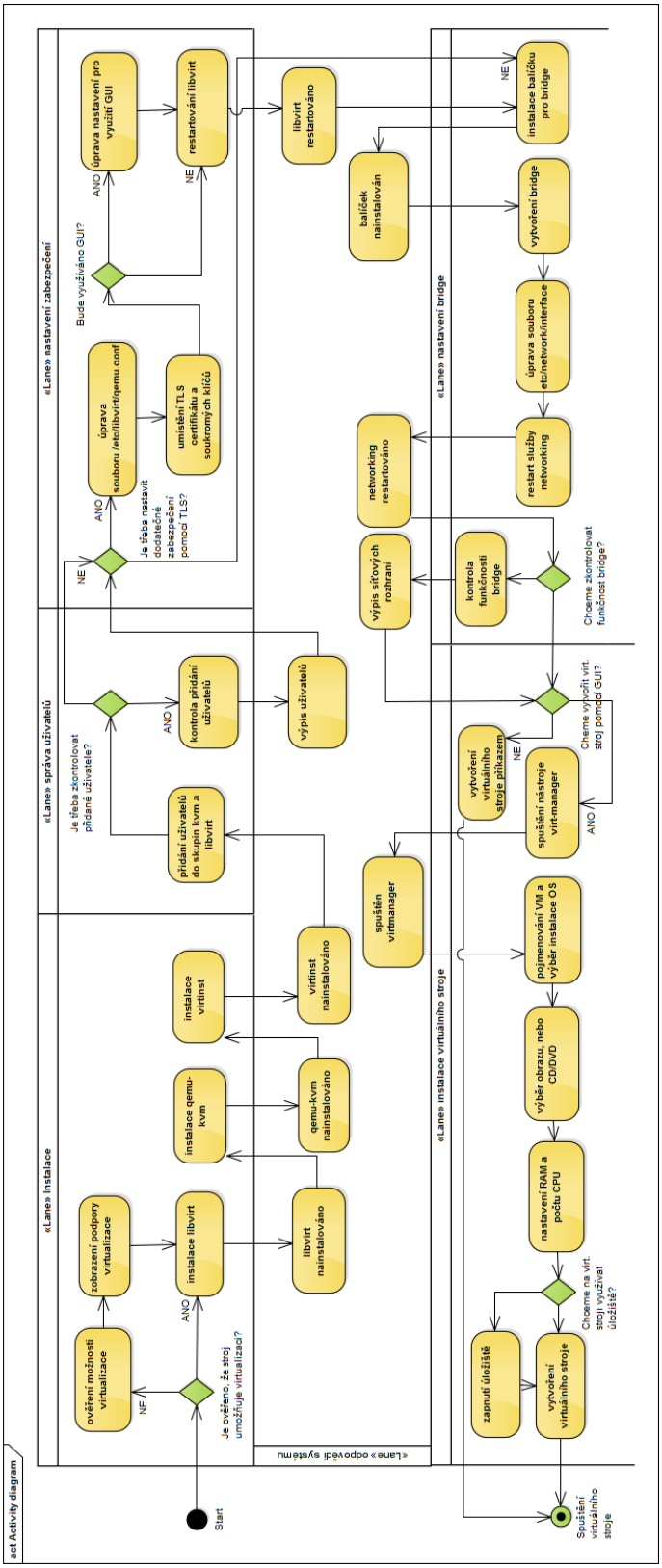
Jak již bylo psáno výše, virtuální stroj je možné vytvořit z příkazové řádky, nebo za pomoci grafického rozhraní. Grafické rozhraní virt-manager je dostačující pro základní tvorbu virtuálního stroje. Pokud ovšem chce uživatel specifikovat více vlastností virtuálního stroje, je vhodnější využít k tvorbě příkazovou řádku. Následující příklad vytvoří virtuální stroj se systémem CentOS, který bude mít operační paměť o velikosti 512MB, 2 jádra a disk o velikosti 10GB.

```
1 virt-install --connect qemu:///system -n CentOSTest -r 512 --vcpus=2 --disk
    path=/var/lib/libvirt/images/CentOSTest.img,size=10 -c
    CentOS-7-x86_64-Minimal-1503-01.iso --vnc --noautoconsole --os-type
    linux --os-variant rhel6 --accelerate --network=bridge:br0 --hvm
```

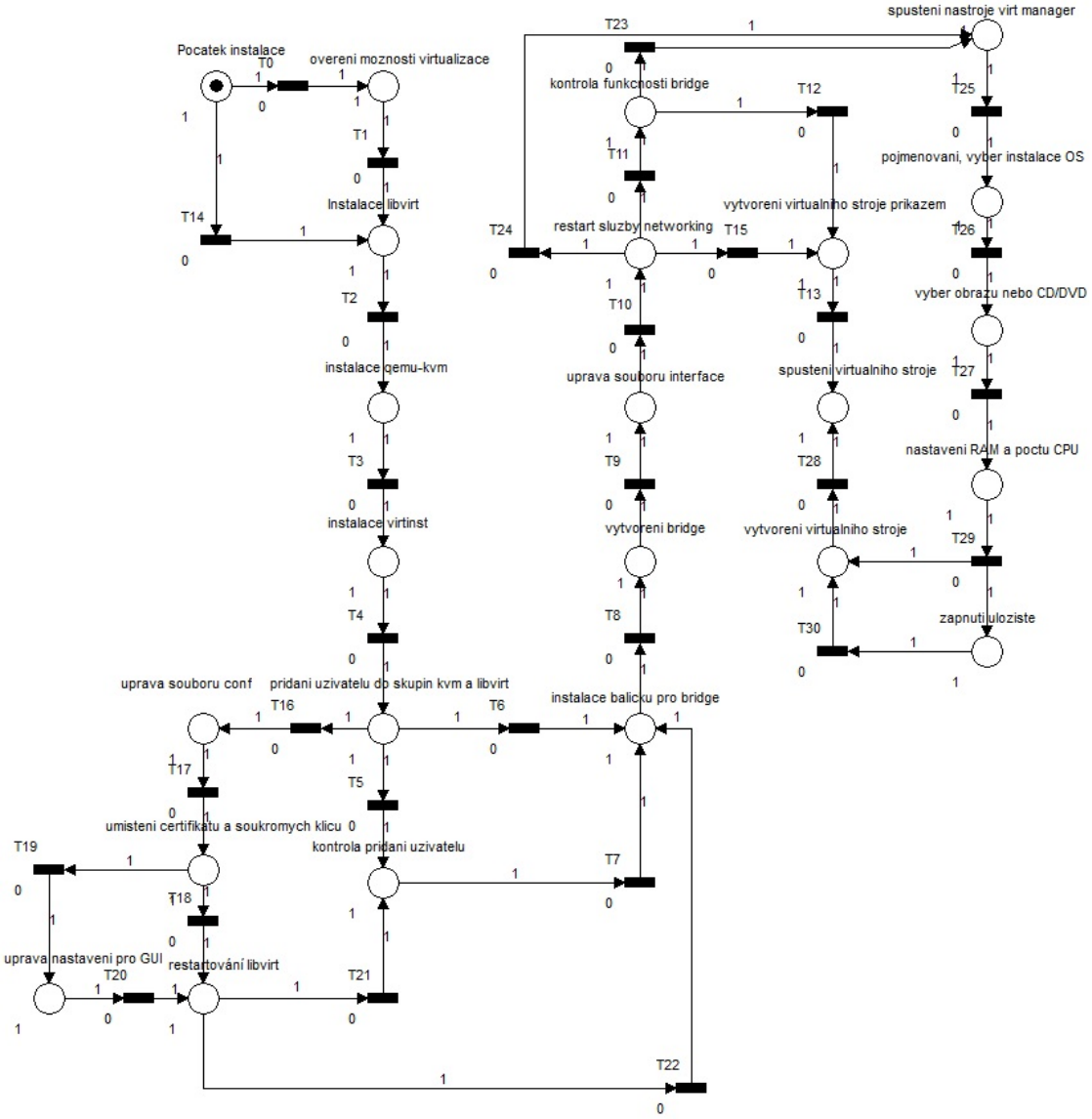


Obrázek 9: Okno programu virt-manager

Jednotlivé kroky instalace byly převedeny do diagramu aktivit, který posloužil k vytvoření Petriho sítě. Diagram aktivit obsahuje všechny činnosti prováděné uživatelem a činnosti prováděné systémem. Tyto činnosti jsou v digramu logicky rozdělené na sekce — instalace, správa uživatelů, nastavení zabezpečení, nastavení bridge, instalace virtuálního stroje a odpovědi systému. Petriho síť poté modeluje pouze činnosti ze strany uživatele.

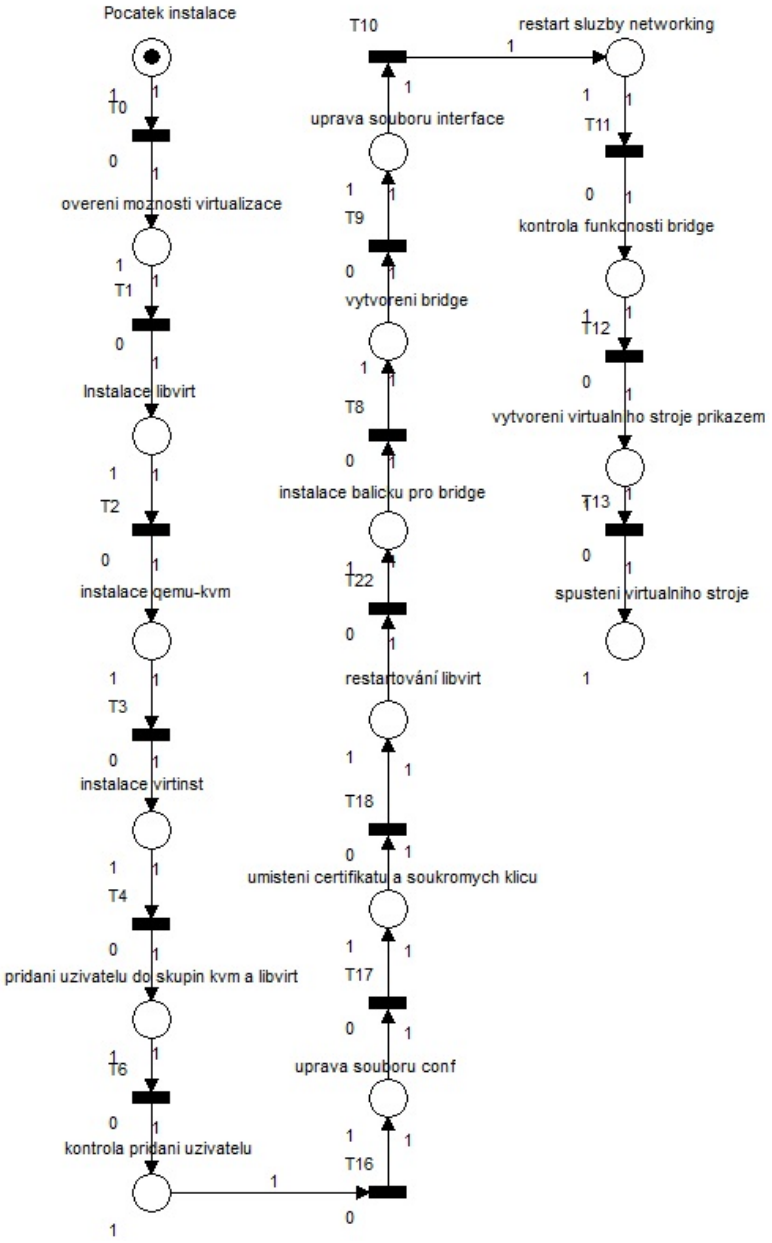


Obrázek 10: Diagram aktivit



Obrázek 11: Petriho síť

Následně byla namodelována optimalizovaná Petriho síť, tedy byl vybrán takový postup, který je nejvhodnější z hlediska bezpečnosti. Byly prováděny veškeré kontroly (kontrola uživatelů, funkčnosti) a především bylo provedeno zabezpečení skrze protokol TLS. Co se samotné instalace virtuálního stroje týče, za vhodnější byla zvolena metoda vytvoření skrze příkaz. Tato metoda nabízí uživateli pokročilé možnosti jako je rozšířené nastavení procesoru, popis stroje, pokročilé nastavení úložiště, nebo nastavení sítě.



Obrázek 12: Optimalizovaná Petriho síť

7 Aplikace

7.1 Specifikace problému

Specifikace se vytváří na základě požadavků uživatele a je třeba především stanovit cíl řešení, požadovaný výsledek, případnou cenu, podmínky dodání a podrobně zdokumentovat cílový stav. Specifikace je zpravidla vytvářena jako neformální a formální. Neformální specifikace představuje zadání projektu a popis problému, formální specifikace pak detailněji popisuje problém a dělí se na funkční a nefunkční požadavky. Funkční požadavky definují požadovanou funkcionalitu systému a nefunkční definují požadavky na řešení (programovací jazyk, požadavky na standardy) a vnější požadavky (legislativa, organizační struktura) (Rábová, 2008).

Neformální specifikace

Koncového uživatele aplikace představují především malé a střední firmy s požadavkem možnosti nahrání dat do databáze s ohledem na jejich citlivost. V každé firmě musí být především dbáno na bezpečnost dat a proto bývají data uložena na vzdáleném serveru, ke kterému bude třeba se skrze aplikaci připojit a vybrat soubory určené k uložení do databáze. Vzhledem k citlivosti některých dat bude třeba zajistit kontrolu přístupu uživatelů. Data je dle jejich citlivosti možné klasifikovat na několika skupin, a to data důvěrná, data pro interní užití a data veřejná. S každou skupinou bude pak třeba nakládat jinak.

Formální specifikace

Funkční požadavky:

- aplikace bude kontrolovat přístup uživatelů pomocí zadané e-mailové adresy
- aplikaci umožní přidávat nové uživatele (jednotlivě, či hromadně za pomoci souboru typu .csv), mazat a editovat
- aplikace bude uživatelům zpřístupňovat funkcionalitu na základě jejich typu oprávnění
- aplikace umožní uživateli měnit citlivostní skupiny dat
- aplikace umožní uživateli zobrazit soubory uložené na vzdáleném serveru
- aplikace nahraje uživatelem zvolené soubory do databáze v závislosti na citlivosti těchto souborů

Nefunkční požadavky:

- diagramy k aplikaci budou modelovány v programu Enterprise Architect
- aplikace bude mít klientskou a serverovou část
- aplikace bude vyžadovat připojení k databázi
- databáze bude implementována v jazyce SQL v MySQL systému
- pro tvorbu a práci s databází bude využita aplikace MySQL Workbench
- aplikace bude implementována pomocí jazyka C#
- jako vývojové prostředí aplikace bude využito Microsoft Visual Studio

7.2 Použité technologie a nástroje pro řešení

Návrh aplikace

Pro základní stanovení požadavků na systém byl využit diagram případů užití (tzv. use case) a pro detailnější návrh jednotlivých funkcionalit byl pak využit diagram sekvenční. Pro tvorbu obou těchto diagramů byl využit program Enterprise Architect.

Enterprise Architect

Jedná se o nástroj pro analýzu, návrh a správu podnikových procesů od společnosti Sparx Systems. Software je zaměřen především na UML a dodržuje tak nejnovější UML specifikace včetně rozšíření, jako je např. SysML, BPMN nebo DDS. Kromě analýzy a návrhu umožňuje také sestavení projektu, simulace, testování a generování finální dokumentace (SPARX SYSTEMS, 2014).

Vývoj aplikace

C# a .NET

C# je objektově orientovaný programovací jazyk navržený především pro spolupráci s technologií .NET společností Microsoft. Pomocí tohoto jazyka lze naprogramovat jak klasickou aplikaci, tak dynamickou webovou stránku, či klientskou aplikaci. Společnost Microsoft oficiálně popisuje jazyk C# jako "jednoduchý, moderní, objektově orientovaný a typově bezpečný programovací jazyk, který je odvozen od jazyků C a C + + ". Mezi základní vlastnosti tohoto jazyka patří: plná podpora tříd a objektově orientovaného programování, konzistentní a vhodně definovaná sada základních typů, integrovaná podpora automatického generování dokumentace ve formátu XML, automatické čištění dynamicky přidělované paměti, plný přístup ke knihovně základních tříd .NET, podpora vlastností a událostí, možnost psaní dynamických stránek ASP.NET a webových služeb založených na XML. C# je multiplatformový programovací jazyk, ovšem s jistými omezeními. Například u operačního systému Linux je dostupný v omezené formě, a to pomocí softwarové platformy Mono (Microsoft Developer Network, 2015).

Co se týče samotné aplikace, mezi stěžejní prvky patří formuláře, konkrétně Windows Forms a framework WCF (Windows Communication Foundation). Formulářové prvky představují grafické rozhraní a usnadňují tak uživateli práci s aplikací. WCF pak zajišťuje možnost klient-server aplikace, tedy zajišťuje komunikaci mezi aplikacemi a možnost tvorby servisně orientovaných aplikací (Microsoft Developer Network, 2015).

.NET je platforma k programování pro Windows. Základem této technologie je umožnění zpětné kompatibility. Mezi výhody technologie .NET patří: objektově orientované programování, navrženo vysoce intuitivním způsobem, jazyková nezávislost (Visual Basic, C#, C++), integrovaná podpora webových stránek a webo-

vých služeb, efektivní přístup k datům (ADO.NET), sdílení kódu mezi aplikacemi, zlepšené zabezpečení (Nagel, 2009).

Microsoft Visual Studio

Společnost Microsoft zároveň s C# vyvinula i vývojové prostředí vhodné právě pro tento jazyk pod názvem Microsoft Visual Studio, které bylo také využíváno právě pro vývoj aplikace. Visual Studio samozřejmě není určeno pouze pro vývoj aplikací v jazyce C#, ale umožňuje i vývoj aplikací například v jazycích C++, Python, či Visual Basic, nebo vývoj pro platformy iOS, nebo Android.

MySQL

MySQL je jednou z nejpopulárnějších open source databází. MySQL nevyžaduje speciální požadavky na hardware a tudíž je možné ji rozjet i na klasických uživatelských počítačích, přičemž ani slabší hardware nemá vliv na její rychlost. Instalace je jednoduchá a téměř automatická. Tuto databázi je také možné nalézt u mnohých Linuxových distribucí, kde je automaticky zahrnuta. MySQL poskytuje standardní prostředí a využívání klasického SQL, čímž usnadňuje uživatelům její použití a případný přesun databáze na jiný databázový systém. Díky rozšíření této databáze je uživatelům k dispozici její široká podpora v podobě knížek, internetových fór či přímo specialistů zaměřující se právě na MySQL. Neposlední výhodou je pak lehké napojení MySQL databáze na vyvíjený software, přičemž většina programovacích jazyků disponuje knihovnami s funkcemi pro práci s MySQL databází (Tahaghoghi, Williams, 2007).

V aplikaci je databáze využívána především ze dvou důvodů. Prvním důvodem je potřeba ověřování uživatelů skrze uživatelské údaje (e-mail) a druhým důvodem je ukládání zvolených dat.

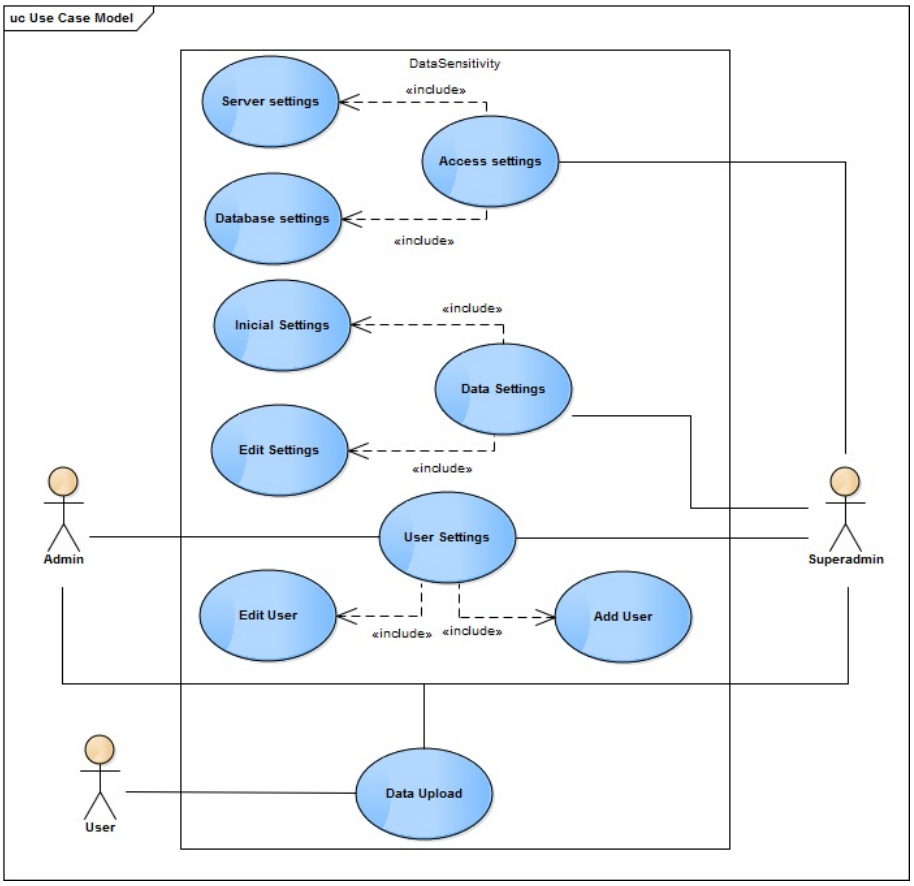
MySQL Workbench

Pro vytvoření a následnou správu databáze byla využita aplikace MySQL Workbench. Tato aplikace umožňuje svým uživatelům navrhnout, namodelovat, vytvořit a spravovat databázi pomocí několika intuitivních integrovaných nástrojů. Pro návrh a modelování slouží Visual Database Design, pro úpravy SQL kódu pak slouží SQL Editor, který odděluje prvky barevnou syntaxí a automaticky doplňuje a napomáhá uživateli. MySQL Workbench pak slouží pro správu databáze, umožňuje správci kontrolovat uživatele připojené k dané databázi, měnit jim oprávnění, či přímo upravovat data v databázi (MySQL, 2015).

7.3 Návrh

Use case

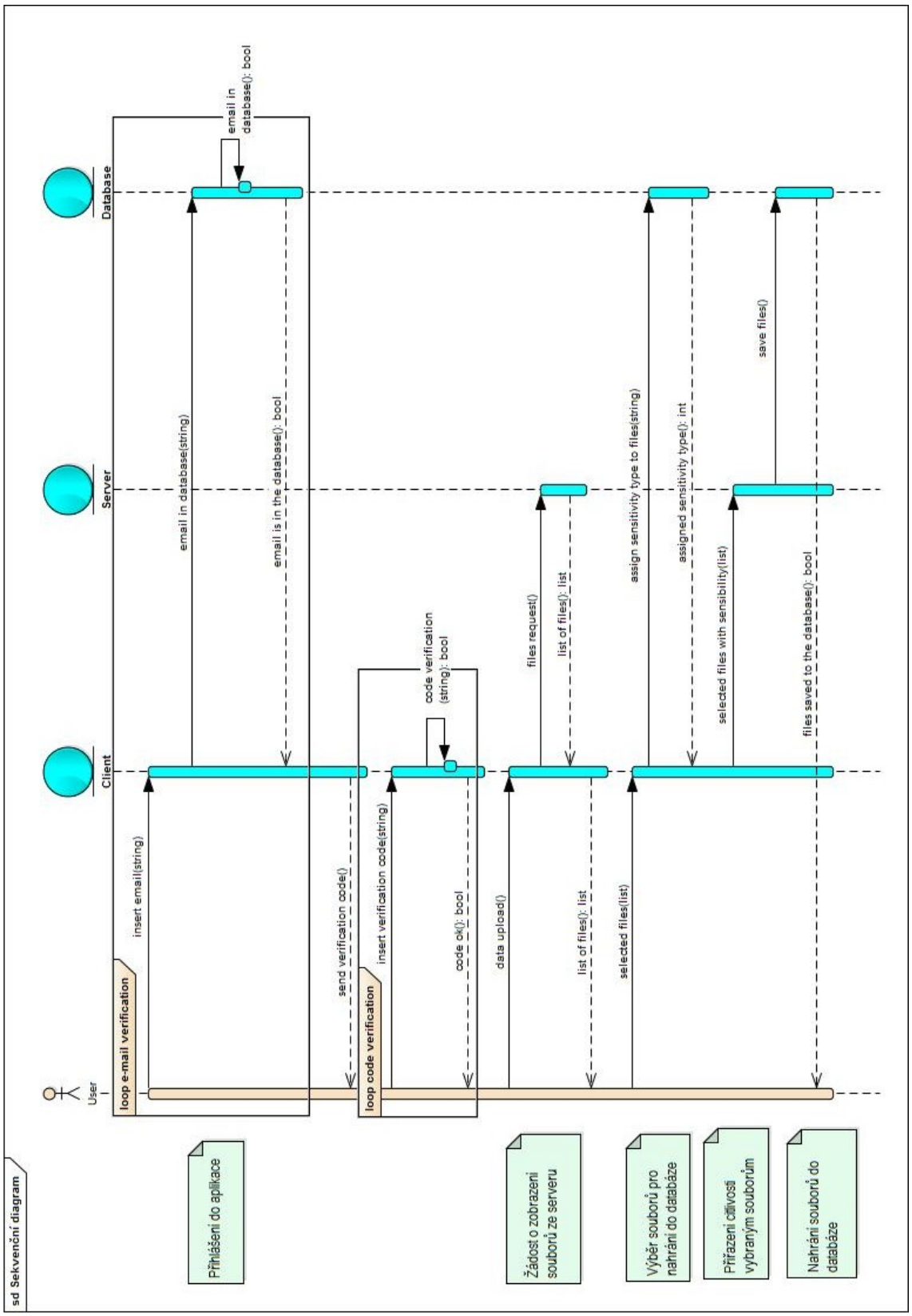
Diagram use case ukazuje chování systému z pohledu uživatele, tedy jaké funkce systém uživateli poskytuje. Říká, co má systém umět, ale ne jak to bude dělat. V podstatě zjednodušeně reflektuje to, co uživatel uvidí v hlavním menu aplikace tedy: nastavení citlivostních stupňů (data settings), nastavení přístupových údajů (access settings), nastavení uživatelů (user settings) a nahrávání souborů (data upload).



Obrázek 13: Use case

Sekvenční diagram pro nahrávání souborů

Následující diagram ukazuje proces nahrávání souborů, což je stěžejní částí této aplikace. Sekvenční diagram zobrazuje chování a spolupráci jednotlivých objektů v rámci jednoho případu užití.



Obrázek 14: Sekvenční diagram pro nahrávání souborů

7.4 Implementace

Základem této aplikace je ukládání dat do databáze na základě jejich citlivostního stupně s ohledem na bezpečnost.

Databáze

V databázi je založeno 5 tabulek - users, extensions, publicFiles, internalFiles a confidentialFiles. První dvě tabulky jsou určeny pro potřeby klienta. V tabulce users je uložen e-mail uživatele, kterým je ověřována jeho totožnost a permissions, což určuje práva uživatele při využívání databáze. V tabulce extensions jsou uloženy uživatelem zvolené koncovky souborů a jejich citlivostní stupně. Do zbývajících 3 tabulek jsou ukládány uživatelem zvolené soubory.

Klientská část

Přihlášení do aplikace

Před prvním přihlášením do aplikace musí uživatel vyplnit údaje pro přihlášení k databázi a serveru. Bez přihlášení k databázi není možné aplikaci využívat.

Při každém další spuštění musí být daný uživatel před přístupem do aplikace ověřen. První ověřením je email, který uživatel vloží po otevření aplikace do příslušného pole. Pokud se tento e-mail nachází v databázi, je uživateli na daný e-mail zaslán verifikační kód, který je náhodně vygenerován. Uživatel má poté 10 minut na to, aby daný kód vložil do aplikace. Pokud je kód správný, nabídne aplikace uživateli menu. Toto menu se liší v závislosti na oprávněních uživatele, které se váží k danému e-mailu.

Menu aplikace

Největší oprávnění má takzvaný superadmin, který má přístup do všech částí aplikace tedy nahrávání souborů, úprava uživatelů, nastavení citlivostních stupňů a nastavení přístupu k serveru a databázi. Dalším oprávněním je admin, který má přístup k nahrávání souborů a úpravě uživatelů a jako poslední oprávnění je běžný uživatel. Ten má přístup pouze k nahrávání souborů.

Nastavení citlivostních stupňů

Superadmin má možnost nastavit parametry, dle kterých se bude rozlišovat citlivostní stupeň jednotlivých souborů. Konkrétně se bude se jednat o parametr koncovka souboru. Aplikace bude dále soubory dělit na základě jejich atributům ovšem k tomuto dělení nebude mít uživatel přístup. Superadmin tedy pouze určí, které koncovky spadají do které skupiny, například .html budou sobory veřejné, .xls soubory důvěrné. Toto nastavení je třeba provést při prvním použití aplikace, jinak nebude možné nahrávat soubory do databáze. Prvotní nastavení je poté dále možné

měnit – mazat jednotlivé koncovky ze skupin, či přidávat nové. Nastavení citlivostních stupňů je uloženo v centrální databázi neboť je třeba, aby k tomuto nastavení měli přístup všichni klienti a všichni měli aktuální informace.

Nastavení přístupu k serveru a databázi

Superadmin musí při prvním spuštění aplikace nastavit adresu serveru a přístupové údaje k databázi. Bez těchto údajů není možné aplikaci využívat, neboť server zprostředkovává klientovi seznam souborů pro nahrání do databáze a vybrané soubory do databáze ukládá. Databáze pak ověřuje samotné uživatele a především jsou v ní uložena pravidla, dle kterých se souborům přiřazují citlivostní stupně.

Úpravy uživatelů

Databáze uživatelů se nachází buď přímo na straně serveru, nebo je od serveru oddělená. Každý uživatel má v databázi uložen e-mail a typ oprávnění (superadmin, admin, user). Superadmin a admin mají možnost uživatele editovat (upravovat a mazat). Pro hromadné nahrávání uživatelů slouží import z CSV formátu, pro přidání menšího množství uživatelů je možné využít klasický formulář v aplikaci.

Nahrávání souborů

Po zvolení položky data upload v menu, vyšle klient požadavek serveru. Server následně klientovi zašle seznam souborů, který se ukáže uživateli v podobě check listu. Uživatel si poté vybere soubory, které chce uložit do databáze a odešle svůj požadavek. Klient nejdříve soubory zanalyzuje – zjistí, jaké mají koncovky a jaké mají atributy a dle těchto parametrů jim přiřadí citlivostní stupeň. Každý vybraný soubor je tedy spárován s citlivostním stupněm a tyto údaje jsou následně odeslány na server. Server poté vybrané soubory nahraje do databáze v závislosti na zvolených citlivostních stupních.

Serverová část

Server má na starost dvě činnosti a to: zprostředkování souborů klientovi a uložení souborů do databáze. Uživatel je na straně serveru povinen nastavit adresu databáze a umístění souborů, které bude server zprostředkovávat klientovi.

8 Ekonomické zhodnocení

Tato práce byla zaměřena na malé a středně velké společnosti. Tyto firmy ve většině případů disponují menší IT infrastrukturou. Dá se předpokládat jeden počítač na zaměstnance a menší počet serverů pro ukládání dat, případně žádné servery. Taková firma pak nemá potřebu volit virtualizační řešení typu VMware, neboť by jeho potenciál naplno nevyužila. Vhodnou volbou bylo v tomto případě tedy open source řešení.

Jako příklad si uvedeme cenu nejlevnějšího VMware řešením, a tím je VMware vSphere Standard. Toto řešení stojí US$ 995.00, tedy přibližně 25000 Kč. V této ceně ovšem není zahrnuta podpora, která stojí US$ 273.00, tedy přibližně 7000 Kč. Pro malou firmu již taková suma může představovat poměrně vysokou částku a vzhledem k tomu, že by s největší pravděpodobností ani nebyl využit celý potenciál tohoto řešení, je open source řešení vhodnější variantou.

Co se cloud computingu týče, voleno bylo mezi hybridním cloudem a cloudem soukromým. Hybridní cloud je kombinací soukromého a veřejného cloudu a vzhledem k důraznosti na bezpečnost byl zvolen soukromý cloud, který veřejnou částí nedisponuje. Bohužel cenový rozdíl není známý, neboť společnosti poskytující cloudové řešení neposkytují ceník svých služeb veřejně. Ovšem dá se předpokládat, že soukromý cloud bude vyžadovat větší náklady, protože musí disponovat větším zabezpečením a spolehlivostí oproti cloudu hybridnímu. Cloud hybridní navíc může umožňovat platby pay-as-you-go, tedy platby pouze za opravdu využívané služby, čímž se náklady společnosti snižují.

9 Diskuze

Jak již bylo několikrát zmíněno, tato práce byla zaměřena na malé a středně velké firmy. Malý a střední podnik představují kategorii podniků s malým počtem zaměstnanců. Malý počet je relativní pojem, protože existuje několik hranic pro rozdělení malého, středního a velkého podniku v závislosti na dané zemi, či vyhlášce.

Cílem každé firmy je co nejvyšší zisk, přičemž zisk firmy z velké části plyne z nákladů a firma, tak musí důsledně zvažovat přínosy možných investic. Mezi přední možnosti investice, zefektivnění procesů firmy a snížení nákladů stojí v dnešní době právě cloud computing a virtualizace.

Hlavní předností cloudu je platba pouze za služby, které jsou opravdu využívané, čímž jsou snižovány náklady společností. Co se modelů cloudu týče, nejvyužívanější je SaaS, tedy Software-as-a-service, který uživatelům umožňuje pronájem aplikací. Tato práce je zaměřena na ukládání dat, proto byla zvolena obdoba SaaS, a to STaaS. Jedná se o Storage-as-a-service a uživatelům je zde k dispozici pronájem datového úložiště. Jako model nasazení byl zvolen cloud soukromý. V úvahu přicházel také cloud hybridní, ovšem celá tato práce je zaměřena na bezpečnost a hybridní cloud je složen z cloudu veřejného a soukromého, čímž vzniká větši riziko ztráty či zneužití dat. Dalo by se polemizovat nad tím, jestli není cloud pro menší firmu zbytečností, neboť menší firmy většinou ani nedisponují servery a pro ukládání dat jim slouží klasické externí disky.

Virtualizace umožňuje přenést fyzický server do virtuálního stroje, čímž především snižuje náklady společnosti a zefektivňuje správu strojů. Místo nákupu fyzických serverů tedy stačí tyto servery virtualizovat, přičemž virtualizované servery budou mít stejnou funkcionalitu a výkon jako servery fyzické. Co se virtualizačního řešení týče, požadavkem bylo řešení pro malou a středně velkou firmu, které bude zadarmo a bude se jednat o hypervizor typu 2 a bude jej možné provozovat jak na operačních systémech Linux, tak Windows. Na základě porovnání virtualizačních technologií byl sestaven rozhodovací strom, ve kterém byla následně dle zvolených požadavků nalezena vhodná technologie. Zvolená technologie, tedy KVM, je samozřejmě výkonově slabší, nežli její placené konkurenti, což pro uživatele znamená pomalejší chod a nižší odezvu. Poslední nevýhodou může být složitější instalace, a to především na systému Linux, kde je od uživatelů očekávána znalost příkazové řádky. Tvorba složitějších virtuálních strojů je třeba opět přes příkazovou řádku, což není intuitivní řešení v porovnání s grafickým rozhraním, které ovšem nabízí pouze základní možnosti. Tyto zápory jsou ovšem přijatelné s ohledem na ušetřené náklady a pro potřeby malé a středně velké firmy je toto řešení dostačující. Řešení zadarmo, neboli open source s sebou ovšem může nést i jisté nevýhody.

Open source řešení bývají často méně uživatelsky přívětivá, neboť vzhledem k tomu, že jsou zadarmo, nejsou často plněny požadavky uživatelů. Další nevýhodou je často neexistující podpora k danému produktu a nebo se může stát, že ačkoliv je produkt zadarmo, podpora je již placená. Poslední výraznou nevýhodou je zranitelnost open source produktů. Vzhledem k dostupnosti kódu těchto produktů se

může stát, že útočník objeví chybu v kódu, kterou následně využije ve svůj prospěch, například ke zneužití dat uživatelů, či celých zařízení, na kterých daný software běží. S těmito nevýhodami je nutné počítat, a tak se open source hodí spíše pro menší firmy s menší zranitelností, nežli pro firmy velké, pro které by například ztráta dat měla fatální následky.

10 Závěr

Prvním cílem diplomové práce bylo nalézt optimální virtualizační řešení pro menší a středně velký podnik. Druhým cílem bylo následně implementovat aplikaci, jejíž hlavní činností bude ukládání dat do databáze v závislosti na citlivosti zvolených dat.

Základ této práce byl stavěn na teoretických znalostech, které byly získány z odborné literatury a dalších zdrojů. Teoretická část práce byla zaměřena na několik oblastí — cloud, virtualizace, data a Petriho sítě. V každé části byly rozebrány základní pojmy dané problematiky, případné rozlišení a návaznost dané části na praktickou část práce. Další část práce, která již nebyla čistě teoretického rázu, se týkala bezpečnosti dat. V této části byl analyzován síťový model OSI s ohledem na bezpečnost, tedy byla určena bezpečnostní opatření na každé vrstvě tohoto modelu. Následně se tato část zaměřila na protokoly SSL a TLS umístěné právě na transportní vrstvě OSI modelu umožňující šifrované stavové spojení. Tyto dva protokoly byly popsány a následně bylo provedeno jejich porovnání.

Dále již bylo přistoupeno k části praktické. Jako první bylo třeba zvolit vhodnou virtualizační technologii. V rámci této práce byly porovnávány technologie VMware ESX, Hyper-V, Citrix Xen, KVM a oVirt. Každá tato technologie byla popsána a zvláštní zaměření bylo věnováno jejím výhodám a nevýhodám. Parametry těchto technologií byly zpracovány do tabulky, ze které byl následně vytvořen rozhodovací strom. Bylo předpokládáno, že menší a středně velké společnosti budou mít zájem o open source technologii, nebudou vyžadovat vyšší výkon, tudíž bude dostačující hypervizor typu 2 a tuto technologii bude možné provozovat jak na operačních systémech Linux, tak Windows. Tyto požadavky byly aplikováno na sestavený rozhodovací strom a byla zvolena technologie KVM.

Tato technologie byla testována na fyzickém serveru s operačním systémem Debian, který bývá díky své jednoduchosti, rychlosti a rychlé odezvě s velkým výkonem využíván pro fyzické servery s virtualizací často. Byly testovány veškeré možnosti instalace KVM a tyto možnosti byly namodelovány pomocí diagramu aktivit. Diagram aktivit obsahoval instalace z pohledu uživatele doplněnou o odpovědi systému. Diagram aktivit sloužil k namodelování Petriho sítě. První namodelovaná síť obsahovala všechny činnosti prováděné uživatelem a byla výchozí sítí pro následnou optimalizaci. Optimalizovaná síť pak obsahovala již pouze kroky vedoucí k nejvhodnější instalace s ohledem na požadavky. Hlavním požadavkem byla bezpečnost a proto bylo při instalaci KVM zvoleno zabezpečení pomocí protokolu TLS.

Poslední částí byla implementace aplikace. Před samotnou implementací byla provedena analýza, byly sepsány funkční a nefunkční požadavky, na základě kterých byl sestaven diagram použití a pro stěžejní funkci, tedy ukládání dat, i diagram sekvenční. Diagramy byly modelovány v programu Enterprise Architect, který umožňuje rychlou a intuitivní tvorbu diagramů. Dalším krokem byl návrh a implementace databáze. Byl zvolen databázový systém MySQL, který představuje jeden z nejpoužívanějších databázových systémů. Dále následovala samotná implemen-

tace aplikace. Aplikace byla programována v programovacím jazyce C# v prostředí Microsoft Visual Studio. Aplikace je pojata jako klient-server. Klient slouží k uživatelskému přístupu, tedy přihlášení uživatele, přidávání uživatelů, úprava nastavení a výběr ukládaných souborů. Server pak má na starost především práci s databází tedy ověřování přihlašovaných uživatelů, poskytování dat klientům a ukládání zvolených dat do databáze.

Je tedy zřejmé, že oba cíle této práce byly naplněny. Do budoucna by bylo vhodné otestovat i zbývající virtualizační řešení, což ovšem k jejich placeným verzím často není možné. Co se aplikace týče, bylo možné aplikaci rozšířit o další funkcionality, jako je například detailnější dělení souborů do bezpečnostních skupin na základě parametrů.

11 Literatura

Benefits of KVM. Open Virtualization Alliance [online]. 2015 [cit. 2015-10-02]. Dostupné z: https://openvirtualizationalliance.org/what-kvm/benefits-kvm.

ČEŠKA, Milan, Vladimír MAREK, Petr NOVOSAD a Tomáš VOJNAR. *Petriho sítě: Studijní opora* [online]. Brno: Fakulta informačních technologií VUT, 2009 [cit. 2015-09-30]. Dostupné z: http://www.fit.vutbr.cz/study/courses/PES/public/Pomucky/PES_opora.pdf.

DAVIS, David. *The top 10 cons of VMware virtualization: Challenging VMware critics*. TechTarget [online]. 2009 [cit. 2015-10-02]. Dostupné z: http://searchvmware.techtarget.com/tip/The-top-10-cons-of-VMware-virtualization-Challenging-VMware-critics.

DESEL, Jorg a Gabriel JUHÁS. *What Is a Petri Net?*. In 'H. Ehrig, G. Juhas, J. Padberg, G. Rozenberg (Eds.):' Unifying Petri Nets, LNCS 2128, Springer, pp. 1–25, 2001.

DESEL, Jörg a Javier ESPARZA. *Free choice petri nets*. 1st ed. Cambridge: Cambridge University Press, 1995, viii, 244 s. Cambridge Tracts in Theoretical Computer Science, 40. ISBN 05-214-6519-2.

Five advantages of Hyper-V over other virtualization platforms. BackupAssis [online]. 2013 [cit. 2015-10-02]. Dostupné z: https://www.backupassist.com/blog/news/five-advantages-of-hyper-v-over-other-virtualization-platforms/.

HAUKIOJA, by John Rhoton and Risto. *Cloud computing architected*. 2013 ed. Tunbridge Wells, Kent: Recursive Press, 2011. ISBN 978-095-6355-614.

Kernel Virtual Machine (KVM): Best practices for KVM [online]. IBM Corporation, 2012 [cit. 2015-09-30]. Dostupné z: https://www-01.ibm.com/support/knowledgecenter/linuxonibm/liaat/liaatbestpractices_pdf.pdf.

KISZKA, Jan. *Architecture of the Kernel-based Virtual Machine (KVM)*. Linux-Kongress [online]. In: . Siemens, 2010, 2015-11-20 [cit. 2015-11-20]. Dostupné z: http://www.linux-kongress.org/2010/slides/KVM-Architecture-LK2010.pdf.

KONEČNÝ, Vladimír. *Integrované informační systémy*. Mendelova univerzita Brno, 2002.

LESOVSKY, Alexey. *Getting started with oVirt 3.3 a practical guide to successfully implementing and calibrating oVirt 3.3, a feature-rich open source server virtualization platform*. New Edition. Birmingham, UK: Packt Pub, 2013. ISBN 978-178-3280-070.

MARKL, J. *Petriho sítě: Úvod - neformální výklad* [online]. 2006 [cit. 2015-09-30]. Dostupné z: http://www.cs.vsb.cz/markl/pn/data/NNPN1.pdf.

MATHER, Tim, Subra KUMARASWAMY a Shahed LATIF. *Cloud security and privacy*. 1st ed. Cambridge: O'Reilly, c2009. ISBN 978-059-6802-769.

MURATA, Tadao. *Petri Nets: Properties, Analysis and Applications*. [online] 1989 [cit. 2015-09-30]. Dostupné z: https://inst.eecs.berkeley.edu/ ee249/fa07/discussions/PetriNets-Murata.pdf.

MySQL Workbench. MySQL [online]. 2015 [cit. 2015-10-03]. Dostupné z: https://www.mysql.com/products/workbench/.

NAGEL, Christian. *C 2008: programujeme profesionálně*. Vyd. 1. Brno: Computer Press, 2009, 2 sv. (1126, 772 s.). Programujeme profesionálně. ISBN 978-80-251-2401-7.

OPPLIGER, Rolf. *SSL and TLS: theory and practice*. Boston: Artech House, c2009, xxi, 257 p. Artech House information security and privacy series. ISBN 978-1-59693-447-4.

PORTNOY, Matthew. *Virtualization essentials*. Indianapolis, IN: John Wiley, 2012, xviii, 286 p. ISBN 978-1-118-17671-9.

PRIŠČÁKOVÁ, Zuzana. *Implementace cloud computingu v podnikovém a univerzitním prostředí – utajení, integrita a dostupnost dat*. Brno: Mendelova univerzita, 2015.

RÁBOVÁ, Ivana. *Podnikové informační systémy a technologie jejich vývoje*. V Tribun EU vyd. 1. Brno: Tribun EU, 2008, 139 s. ISBN 978-80-7399-599-7.

RISTIĆ, Ivan. *Buletproof SSL and TLS*. United Kingdom: Feisty Duck Limited, 2014. ISBN 978-1-907117-04-6.

ROUSE, Margaret. *Data classification definition*. TechTarget [online]. 2007 [cit. 2015-10-02]. Dostupné z: http://searchdatamanagement.techtarget.com/definition/data-classification.

RUEST, Nelson a Danielle RUEST. *Virtualization a beginner's guide*. New York: McGraw Hill, 2009. ISBN 978-007-1614-023.

SIMORJAY, Frank. *Data classification for cloud readiness* [online]. Microsoft Corporation, 2014 [cit. 2015-09-30]. Dostupné z: http://download.microsoft.com/download/0/A/3/0A3BE969-85C5-4DD2-83B6-366AA71D1FE3/Data-Classification-for-Cloud-Readiness.pdf.

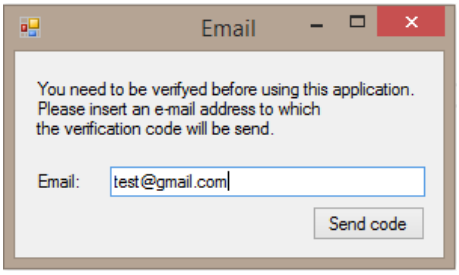
SPARX SYSTEMS. *Enterprise Architect User Guide*. [online] 2014 [cit. 2015-09-30]. Dostupné z: http://www.sparxsystems.com.au/bin/EAUserGuide.pdf.

TAHAGHOGHI, Seyed M a Hugh E WILLIAMS. *Learning MySQL*. Sebastopol, Calif.: O'Reilly, c2007, xvii, 598 p. ISBN 05-960-0864-3.

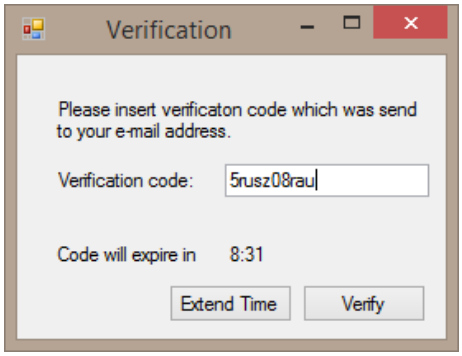
Visual C Language. Microsoft Developer Network [online]. 2015 [cit. 2015-10-03]. Dostupné z: https://msdn.microsoft.com/en-us/library/aa287558(v=vs.71).aspx.

WINKLER, J. *Securing the cloud: cloud computer security techniques and tactics*. Waltham, MA: Syngress/Elsevier, 2011, 290 p. ISBN 978-159-7495-929.

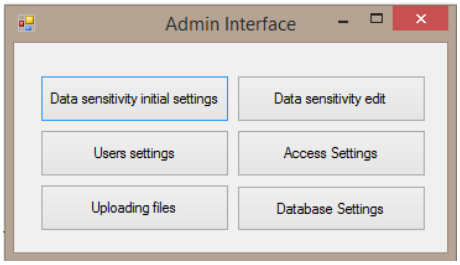
12 Přílohy



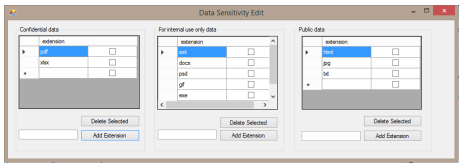
Obrázek 15: Klient — E-mailové ověření uživatele



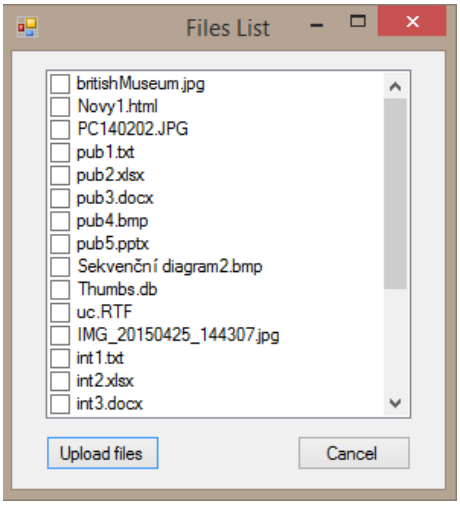
Obrázek 16: Klient — Vložení ověřovacího kódu do aplikace



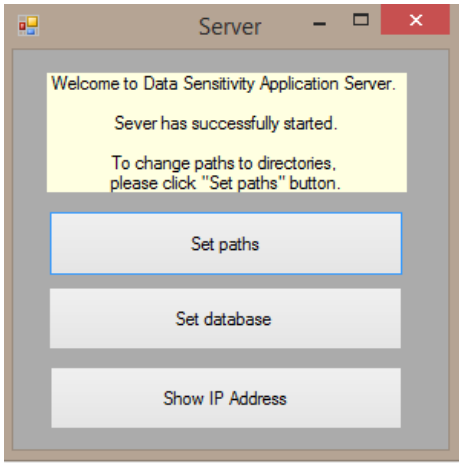
Obrázek 17: Klient — Menu dostupné superadminovi



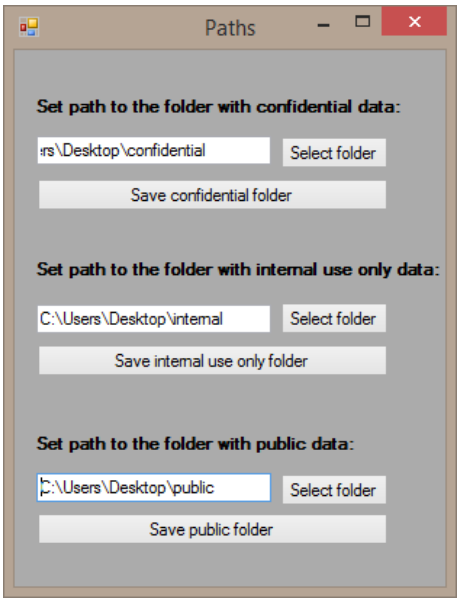
Obrázek 18: Klient — Nastavení citlivostních stupňů



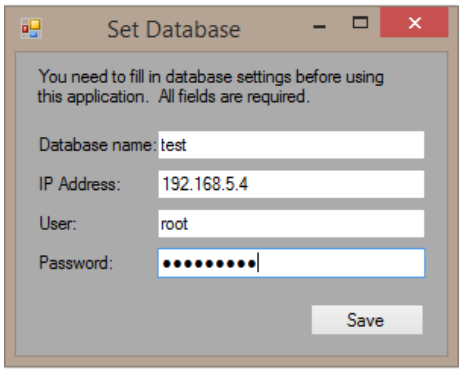
Obrázek 19: Klient — Výpis souborů možných k uložení



Obrázek 20: Server — Menu serveru



Obrázek 21: Server — Nastavení cest ke složkám se soubory



Obrázek 22: Server — Nastavení přístupu k databázi