Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Program pro správu PC a serverů

2020

Martin Štěpánek

Vedoucí práce: RNDr. Jan Konečný, Ph.D

Studijní obor: Aplikovaná informatika, kombinovaná forma

Bibliografické údaje

Autor:	Martin Štěpánek
Název práce:	Program pro správu PC a serverů
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2020
Studijní obor:	Aplikovaná informatika, kombinovaná forma
Vedoucí práce:	RNDr. Jan Konečný, Ph.D
Počet stran:	50
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliograhic info

Author:	Martin Štěpánek
Title:	Software for remote management of PC and servers
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2020
Study field:	Applied Computer Science, combined form
Supervisor:	RNDr. Jan Konečný, Ph.D
Page count:	50
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

V práci analyzuji jednotlivé systémy a existující nástroje pro jejich vzdálenou správu. Následně navrhuji vlastní řešení. Na konkrétních příkladech ukazuji současný stav a řešení pomocí výsledného programu. Úkolem je vytvořit nástroj pro praktické a efektivní využití při každodenní práci administrátora.

Synopsis

In this work I analyze individual systems and existing tools for their remote management. Then I propose my own solution. On specific examples I show the current state and solution using the resulting program. The task is to create a tool for practical and effective use in the daily work of the administrator

Klíčová slova: vzdálená správa; software; programování

Keywords: remote management; software; programming

Děkuji vedoucímu práce doc. RNDr. Janu Konečnému, Ph.D, svým nadřízeným za ochotu a přítelkyni za podporu.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

V prvních kapitolách analyzuji jednotlivé systémy a nástroje pro jejich správu. Následující kapitoly věnuji návrhu a realizaci vlastního řešení za pomocí programovacího jazyka C#. Jedná se především o správu systémů ve světě Microsoft, byť určitě části lze použít i mimo něj.

Smyslem tvorby programu je docílit vyšší efektivity práce pomocí poznatků z praxe, doplnit či úplně nahradit základní nástroje používané např. při práci systémového administrátora, dále si kladu za cíl zjednoduššit jinak složitější úkony.

Program je tvořen pro velké firemní prostředí, v potaz tedy přichází i velký počet koncových stanic a serverů. Použití programu je možno i v menších prostředích na platformě Windows.

2 Analýza systému a současné nástroje

V této části představím jednotlivé systémy a zmíním jednotlivé nástroje pro jejich správu, zaměřím se především na oficiální nástroje od společnosti Microsoft, jelikož program zaměřuji na platformu Windows. U každého z nástrojů zmíním výhody a případně nevýhody na základě vlastní analýzy.

Cílové systémy jsou především z oblasti základní infrastruktury firemního prostředí, ve kterém je potřeba nějakým způsobem spravovat uživatele, počítače a další objekty v rámci firmy. Pro tyto učely slouží služba Active Directory, kterou představím v první podkapitole. Druhá podkapitola se věnuje jmenné službě – DNS, která mimo jiné překládá názvy počítačů na IP adresy a tedy s Active Directory úzce souvisí. Třetí podkapitola pojednává o systému, který přiděluje IP adresy a další potřebné údaje pro komunikaci na síti. Poslední podkapitola se věnuje databázovému systému, který není pro základní infrastukturu kritický, přesto se jedná o velmi důležitou a běžnou část firemního prostředí.

Žádný z těchto systémů není autonomní a vyžaduje tedy údržbu, nebo zásahy v nastavení, pro tyto případy máme programy pro jejich správu, nebo-li nástroje, jejichž prostřednictvím spravujeme jednotlivé objekty, záznamy, nebo databáze dle typu systému.

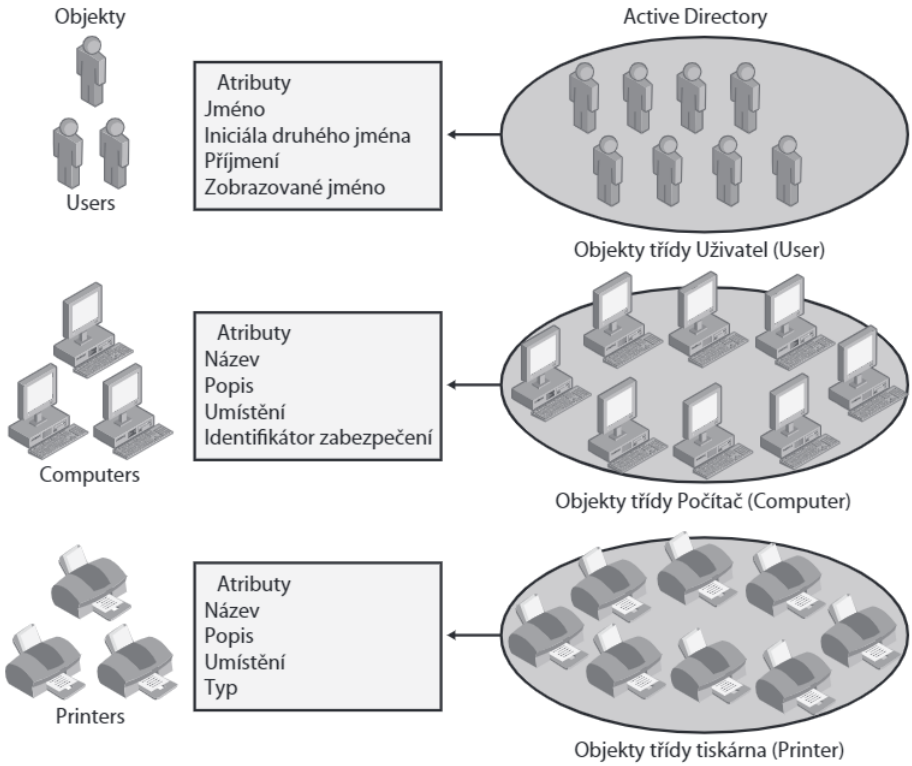
V kapitolách lze narazit na pojem audit, popř. auditovací funkce. Pod tímto pojmem rozumím přehled stavu daného systému, především případné možné nedostatky. Příkladem může být výpis prázdných skupin v Active Directory, na které nás auditovací funkce upozorní. Prázdná skupina sama o sobě není chybou, přesto je vhodné upozornit na existenci takových skupin, aby uživatel zvážil jejich smysl.

2.1 Active Directory

Před samotnou analýzou a představením nástrojů pro službu Active Directory, je potřeba vysvětlit k čemu Active Directory vlastně slouží:

> Active Directory je rozšiřitelná adresářová služba, která umožňuje centralizovanou správu síťových prostředků. Umožní vám snadno přidávat, odebírat nebo přemisťovat účty pro uživatele, skupiny a počítače, stejně jako jiné typy prostředků. Téměř každá úloha správy, kterou provedete, nějakým způsobem souvisí se službou Active Directory. Služba Active Directory je založena na standardních internetových protokolech a její návrh vám pomůže jasně identifikovat fyzické a logické součásti struktury vaší sítě[**1**].

Správou Active Directory tedy rozumíme správu objektů (viz obr. 1) konkrétně uživatelů, skupin a počítačů. Nástroj umožňuje vyhledávání, vytváření, úpravu či mazání objektů. Celé téma Active Directory je poněkud obsáhlejší, Active Directory obsahuje logickou strukturu rozdělenou na domény, stromy, organizační jednotky aj.[**1**]. Pro potřeby mého programu lze však uvést pouze tyto skutečnosti a nezacházet příliš do hloubky.



Obrázek 1: Objekty v Active Directory (Zdroj: [**1**])

Jedním z nástrojů pro správu Active Directory je program: „Uživatelé a počítače služby Active Directory“ (viz obr. 2). Používá se pro správu již zmíněných objektů tedy uživatelů, skupin a počítačů[**1**]. Nástroj se instaluje jako součást balíčku „RSAT: Active Directory Domain Services a Lightweight Directory Tools“, který je možné stáhnout z oficiálních stránek společnosti Microsoft.

Následuje výčet výhod a nevýhod tohoto nástroje. Tento výčet je subjektivním názorem, přesto jsem se rozhodnul jej prezentovat, pro přiblížení motivace tvorby vlastního programu.

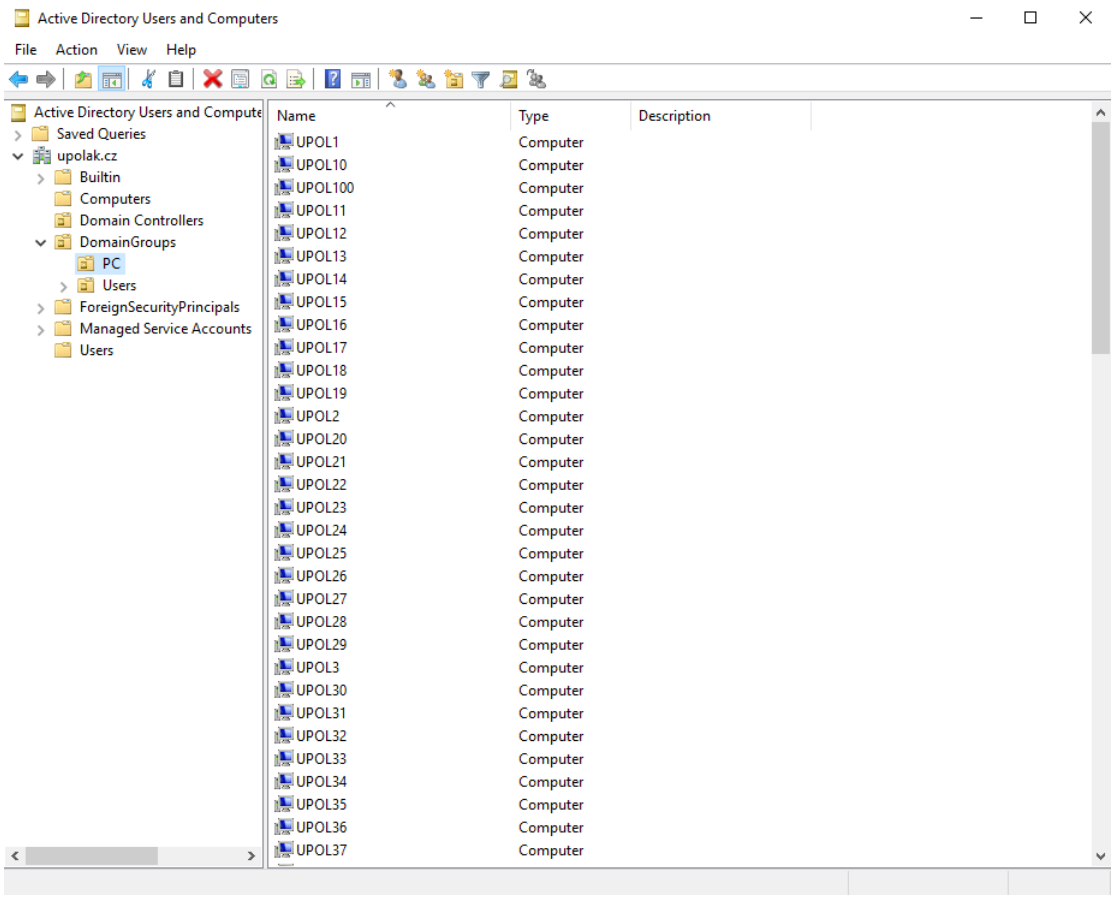
Mezi výhody nástroje patří:

- jednoduchost,
- oficiální nástroj,
- obecná známost,
- přehledné a přívětivé uživatelské rozhraní.

Mezi nevýhody patří:

- nedostatečné vyhledávání,

- absence pokročilých funkcí, např. auditu.



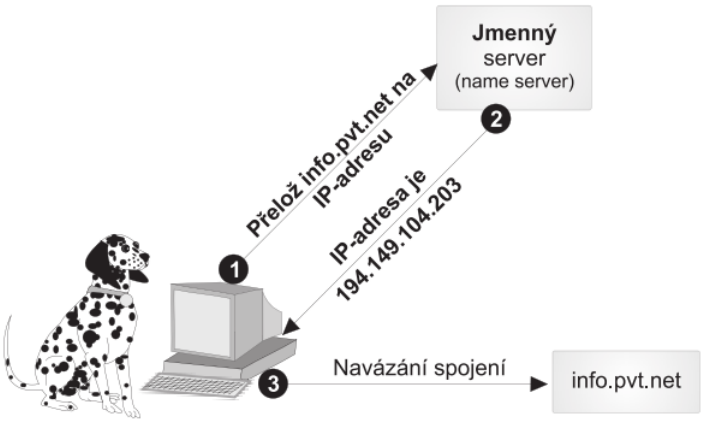
Obrázek 2: Ukázka programu „Uživatelé a počítače služby Active Directory“.

Nástroj považuji na jednu stranu za dostačující, co se týče základních operací, na druhou stranu mi chybí auditovací funkce typu: „Vyhledej prázdné skupiny“, „Zobraz neaktivní počítače“. Postrádám také propojenost s dalšími systémy, uznávám však, že poslední připomínka je otázkou pohledu na vývoj.

2.2 DNS

V této kapitole zmíním základní terminologii a nastíním systém překladu názvů – DNS.

> Klienti služby Active Directory používají službu DNS k nalezení prostředků. Služba DNS překládá snadno čitelné názvy hostitelů na číselné adresy protokolu IP (Internet Protocol).
> Každému počítači v doméně je přiřazen plně kvalifikovaný název domény (FQDN), například server34.microsoft.com[**1**].



Obrázek 3: Překlad jména na IP adresu. (Zdroj: [**10**])

Příklad překladu jména na IP adresu vidíme na obrázku 3. Pro překlad z IP adresy na jméno slouží tzv. reverzní záznam[**10**]. Záznamu je možné vytvořit synonymum, označujeme CNAME, někdy také alias[**10**].

DNS server nabízí množství typu záznamů, které pro naši potřebu není potřeba uvádět, důležité je vědět, k čemu DNS server slouží a jaký je jeho účel.

Pro správu DNS se používá stejnojmenný nástroj, který je ,stejně jako nástroj v předchozí podkapitole, součástí balíčku RSAT.

Mezi výhody nástroje patří:

- jednoduchost,
- oficiální nástroj,
- obecná známost,
- velké množství nastavení a konfigurace.

Mezi nevýhody patří:

- nedostatečné vyhledávání,
- absence auditovacích funkcí.

Nástroj je více než dostačující, přesto lze vytknout například nedokonalé dohledávání konkrétních záznamů.

2.3 DHCP

Tato podkapitola pojednává konkrétně o *Microsoft DHCP Server*. DHCP služba slouží pro automatické přidělování IP adresy, masky podsítě, výchozí brány a DNS serverů[**2**]. Výhodou je fakt, že konfiguraci klienta není nutné provádět manuálně. DHCP využívá tzv. scopes neboli rozsahy adres, které definují síťový segment[**2**]. Jednotlivé rozsahy lze vytvářet a následně konfigurovat[**2**].

Princip fungování DHCP lze popsat následovně[**2**]:

- klientům je po žádosti propůjčena IP adresa, tzv. lease,
- žádost po určitém čase vyprší a IP adresa je volná k dalšímu přiřazení,
- propůjčení IP adresy lze převést na tzv. rezervaci, čímž zajistíme permanentní přiřazení IP adresy.

Službu DHCP lze rozdělit na DHCP server a DHCP nástroj, DHCP server má na starosti interakci s klienty, přičemž DHCP nástroj nabízí uživatelské prostředí pro správu serveru[**2**].

Správou DHCP tedy rozumíme správu jednotlivých záznamů a rozsahů. Nástroj umožňuje vytváření, úpravu či mazání záznamů a rozsahů tzv. scopes. Ukázku nástroje vidíme na obrázku 4. Nástroj pro správu DHCP je stejně jako nástroj pro AD a DNS součástí balíčku RSAT.

Stejně jako služba Active Directory i služba pro DHCP je obsáhlejší, než jsem v této ukázce prezentoval, můj výsledný program se však zabývá běžnou správou ve funkčním provozu, která se týká především rezervací, pronájmů a rozsahů. Konfigurace pokročilých nastavení a možností probíhá po instalaci DHCP serveru a do konfigurace jako takové se jistě nezasahuje při denní práci administrátora.

Následuje opět výčet výhod a nevýhod nástroje pro DHCP.

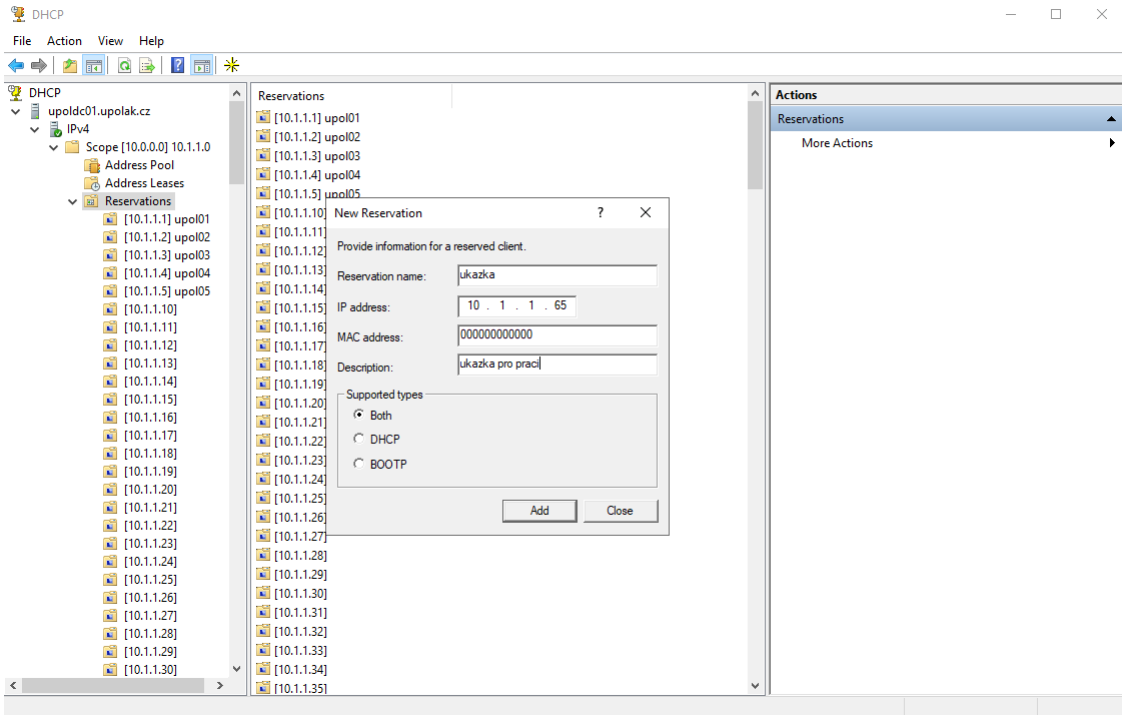
Mezi výhody nástroje patří:

- dostatečný pro základní operace,
- oficiální nástroj,
- obecná známost.

Mezi nevýhody patří:

- absence jednoduchého vyhledávání,
- absence auditovacích funkcí.

Nástroj je v základu dostačující, umožňuje také pokročilé nastavení, některé operace jsou avšak prováděny složitějšími postupy, např. vyhledání konkrétní rezervace považuji za pomalé a nepohodlné.



Obrázek 4: Přidání nové rezervace v nástroji „DHCP“.

2.4 MS SQL Server

MS SQL Server – Je databázový systém od společnosti Microsoft. Databázový systém umožňuje práci s databázemi, jako je čtení, změna a mazání dat[**3**]. V případě MS SQL Serveru mluvíme o relačním databázovém systému, který ukládá data ve formě tabulek – relací[**3**]. Pro dotazy nad databázemi, ale i pro SQL Server obecně, používáme dotazovací jazyk T-SQL. Dotazem získáváme potřebné informace z databáze[**3**].

Pro správu databázového serveru nabízí Microsoft dva nástroje. Databázový server je možno spravovat přes příkazovou řádku, konkrétně pomocí programu *sqlcmd*, který je možno nainstalovat jak samostatně, tak v rámci instalace serveru.

Výhody nástroje *sqlcmd*:

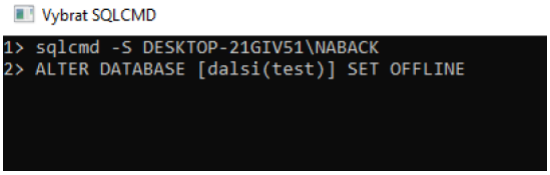
- malá instalace,
- rychlý a silný nástroj,
- poslední záchrana při obnovování master databáze.

Nevýhody nástroje *sqlcmd*:

- nutnost znát příkazy,

- žádné grafické rozhraní[1],
- časově náročnější pro určité operace vzhledem k nutnosti příkazy psát.

Na obrázku 5 vidíme praktickou ukázku přepnutí databáze [dalsi(test)] do OFFLINE režimu v nástroji „sqlcmd“.



Obrázek 5: Přepnutí databáze [dalsi(test)] do OFFLINE režimu v nástroji „sqlcmd“.

Druhým nástrojem je *SQL Server Management Studio*, který byl dříve součástí instalace SQL Serveru. Od verze Microsoft SQL Server 2016 je nutné nástroj stáhnout a nainstalovat samostatně.

SQL Server Management Studio je kombinací dvou starších nástrojů: Enterprise Manager a Query Analyzer.[**3**]. S oběma jmenovanými nástroji jsem se setkal i v praxi a v porovnání se současným nástrojem tj. SQL Server Management Studio musím uznat, že novější nástroj přinesl velkou změnu a vylepšení.

Výhody nástroje *SQL Server Management Studio*:

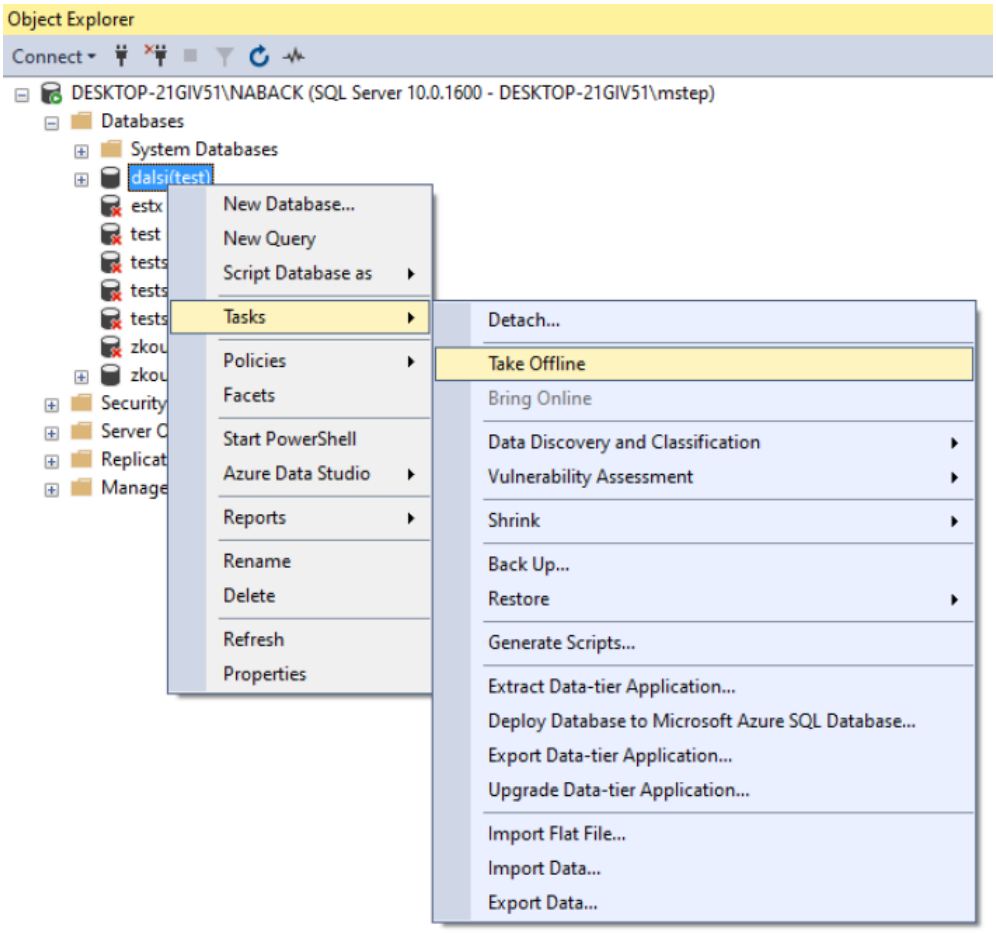
- obsáhlý nástroj,
- přívětivé a přehledné uživatelské rozhraní,
- efektivní a odladěný nástroj.

Nevýhody nástroje *SQL Server Management Studio*:

- přehršel funkcí a informací,
- některé základní funkce chybí.

Obrázek 6 ukazuje totožnou akci tj. přepnutí databáze [dalsi(test)] do OFFLINE režimu, tentokrát v nástroji „SQL Server Management Studio“.

[1]Tato vlastnost nemusí být nutně nevýhodou, záleží na úhlu pohledu a preferencích daného uživatele.



Obrázek 6: Přepnutí databáze [dalsi(test)] do OFFLINE režimu v nástroji „SQL Server Management Studio“.

Oba nástroje mají své použití a místo, které je obtížné nahradit. Nástroje se vzájemně doplňují. Přesto zde vidím prostor pro zlepšení, především u druhého z nich, kde mi chybí důležité funkce, které se musí provádět přes příkazy, např. identifikace a řešení tzv. sirotků.

Sirotek je označení pro uživatele databáze jehož přihlášení, tzv. login, bylo odstraněno nebo neexistuje[**4**]. Prakticky to znamená, že uživatel, jenž je sirotkem , nebude mít možnost se přihlásit k serveru i přes skutečnost, že účet existuje na úrovni databáze. Z praxe bych rád poznačil, že na tento stav se mnohdy zapomíná, a to i u správců databází, kdy po přesunutí databáze na jiný server již nikdo nemyslí na tuto situaci, a z mého pohledu by měl nástroj nabízet upozornění na tento stav.

2.5 Počítače a servery – klient

Pro správu počítače a serveru lze použít nástroje jako *služby*, *správa počítače*, *regedit* aj., které umožňují vzdálené připojení. Žádný z těchto nástrojů nenabízí

pokročilejší možnosti vzdálené správy. Řešením je například použití vlastního scriptu v programu *PowerShell*, v případě zaměření na více počítačů nebo serverů také pomocí *Group Policy*.

Nástroj *Group Policy* slouží opravdu především k centrální správě počítačů, uživatelů, aplikací aj. Zde bych rád zdůraznil slovo centrální, neboť k tomuto užití je nástroj velmi efektivní, v kapitole příklady z praxe ovšem uvidíte proč tento nástroj nemá v některých příkladech vhodné využití.

Pro správu počítače nebo serveru existuje více jednoúčelových nástrojů, žádný z nich nenabízí možnosti jako je vzdálená instalace popř. odinstalace programu, instalace knihoven, zálohování aj. Alternativou je opět *PowerShell* nebo *Group Policy*, tato varianta je opět dle mého názoru nevhodná pro určité situace, nejlépe moji motivaci naznačí příklady z praxe.

System Center Configuration Manager

Zkráceně také SCCM, je pokročilý nástroj pro správu počítačů a serverů. Vyjmenování a popis jednotlivých funkcí by zabralo několik desítek stran, proto představím především základní myšlenku tohoto nástroje a zmíním základní funkce. Tento nástroj se označuje také jako *Microsoft Endpoint Configuration Manager*.

SCCM se skládá z tzv. lokalit a databázového serveru. Lokalitu můžeme chápat jako bod pro připojení a správu klientů, přičemž počet lokalit musí být alespoň jedna, ta se označuje jako primární. Primární lokalitu lze nainstalovat na samostatný server, ale také společně s databázovým serverem, dále je možno instalovat a přídávat další lokality, aby nedocházelo k přetěžování jedné lokality[**5**].

Na jednotlivé klienty, ať už klasické nebo serverové poté distribuujeme agenta, který má za úkol komunikovat s danou lokalitou, sbírat informace a provádět dané úkoly[**5**]. Jedná se tedy o tzv. agentové řešení, které budu zmiňovat i dále v textu.

Dovolím si zmínit, alespoň některé funkce tohoto nástroje. SCCM umožňuje sběr informací o hardware, software nebo také například o tom, kdo se na počítač hlásí a jak často. Nástrojem je možno distribuovat aktualizace operačního systému, instalovat libovolné aplikace, spouštět vlastní scripty, vytvářet a spouštět libovolné reporty. Nabízí tedy i auditovací funkce, mimo jiné dokáže odpovědět na otázky: „Kdo má nainstalovaný program X“, „U kterých počítačů sedí nejčastěji Jan Novák“, „Kdo nerestartoval počítač déle než týden“ aj. Dále zmíním také možnost vzdálené instalace operačního systému, v případě využití antivirového programu Windows Defender také přehled hlášení a napadení[**5**].

Výhody nástroje *System Center Configuration Manager*:

- obsáhlý nástroj,
- velké množství funkcí,
- dokáže do určité míry nahradit Powershell a Group Policy,

- auditovací funkce,
- aktivně vyvíjen,
- oficiální nástroj společnosti Microsoft,
- spousta funkcí na jednom místě.

Nevýhody *System Center Configuration Manager*:

- vyžaduje odbornou instalaci a správu,
- agentové řešení[2],
- vše na jednom místě,
- placený.

Fakt, že je vícero funkcí na jednom místě považuji za výhodu i za nevýhodu, pokud nám lokalita z nějakého důvodu přestane fungovat, přijdeme tím mimo jiné i o aktualizace pro počítače, o dohled nad antivirovým programem, o scripty, nasazení aplikací a spoustu dalšího.

Za normálního stavu je výhoda mít přehledně vše na jednom místě a nemít vše rozprostřeno mezi velký počet jednotlivých nástrojů.

K nástroji je nutno přistupovat se znalostí a obezřetností, možnost vzdálené instalace operačního systému zní například velmi dobře, může se však stát, že po v podstatě dvou kliknutích přeinstalujete počítače a servery v celé firmě, což v některých případech znamená i několik tisíc klientů.

3 Příklady z praxe

Motivací pro návrh vlastního řešení jsou jednoznačně příklady z praxe, některé z nich zde představím, ukáži jaké je jejich řešení s dosud představenými nástroji. Po návrhu a vytvoření vlastního nástroje představím řešení těchto konkrétních problémů ve vlastním nástroji.

3.1 Problém č.1: Instalace programu na 10 klientů

Klientem zde rozumíme jak počítač, tak server. Na první pohled jednoduchý požadavek. Při takovém požadavku se nabízí více řešení. Prvním je zajisté *Group Policy*, tj. vytvoření pravidla nad skupinou klientů. Problém nastává, pokud jsou klienti rozřazeni do různých skupin v Active Directory, tedy do tzv. OU, někdy není takové rozlišení klientů jednoduché a řešení si vyžaduje více času.

[2]Jelikož se jedná o agentové řešení tj. agent-server, je nutné také případně řešit problémy s komunikací a celkovou distribuci, některé úkony jsou také zdlouhavé oproti přímému zásahu na počítači.

Dalším faktorem je také rychlost této aplikace, klient musí aktualizovat nastavení a zjistit, že má nějaké nové pravidlo, tato aktualizace probíhá po stanoveném časovém intervalu. Instalace také může proběhnout až po restartování klienta, je tedy nutné požádat uživatele o ukončení jeho pracovní činnosti a restartování popř. vyčkat do konce pracovní doby.

Kvůli jednoduché instalaci, jsme také vytvořili pravidlo, které bude potřeba v budoucnu případně odstranit. Výhodou je na druhou stranu situace, při které dojde k odinstalaci programu ať už omylem nebo zámerně, díky Group Policy se program znovu nainstaluje a je tak zajištěna přítomnost programu.

Další řešení je připojit se na každý z klientů, ať už vzdálenou plochou či jiným nástrojem, najít si příslušný instalační soubor a instalaci provést ručně, čímž opět zasahujeme do práce koncového uživatele.

3.2 Problém č.2: Zálohování konkrétní složky z uživatelských profilů

V tomto příkladu máme za úkol odzálohovat např. složku *Plocha* pro každého z uživatelů, přičemž potřebujeme zachovat strukturu složek. Složka plocha je umístěna ve složce `C:\Users\Uzivatel\Desktop`, přičemž uživatelů může existovat několik desítek. Možností je zkopírovat celou složku *Users*, během této akce zkopírujeme i složky, které pro nás nejsou potřebné a které mohou obsahovat značné množství dat, tj. kopírování bude mnohonásobně delší. Další možností je ručně projít každou z uživatelských složek zvlášť, vytvořit novou složku s názvem uživatele a do ní následně překopírovat danou složku. Časová náročnost obou operací je značná.

Příkladem z praxe jsou i programy, které si ukládají konfigurační soubor, nebo jiná data, do uživatelského profilu pro každého uživatele zvlášť, konkrétně do `C:\Users\Uzivatel\.pcedit`, v tomto případě se jedná o program pro monitoring systému potrubní pošty. Při potřebě zálohy takového nastavení narazíme na problémy, viz výše.

3.3 Problém č.3: Přepnutí 30 databází do offline režimu

Pokud budeme mít za úkol přepnout např. 30 z 50 databází do tzv. offline režimu a případně provést další příkaz během této operace nad každou z databází, nezbývá nám nic jiného, než použít příkaz níže, přičemž tento příkaz musíme zkopírovat 29× a změnit název pro každou z databází.

```
USE MASTER
ALTER DATABASE [Zamestnanci]
SET AUTO_UPDATE_STATISTICS_ASYNC OFF
ALTER DATABASE [Zamestnanci]
SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
ALTER DATABASE [Zamestnanci]
SET OFFLINE WITH ROLLBACK IMMEDIATE;
```

První z příkazů, jenž následuje po *USE MASTER*, vypne možnost automatické aktualizace statistik, tedy: *AUTO_UPDATE_STATISTICS_ASYNC*, která by mohla způsobit potíže s přepnutím do *SINGLE_USER* režimu, konkrétně by server obsadil jediné možné připojení a nebyli bychom se schopni připojit[**6**]. Následuje samotné přepnutí do *SINGLE_USER* režimu, tzn. k databázi může být připojen pouze jeden účet. Poslední příkaz je samotné přepnutí do *OFFLINE* režimu. *WITH ROLLBACK IMMEDIATE* navrátí zpět veškeré nedokončené operace a poté dojde k danému příkazu.

Pokud během nebo po úpravě takového množství textu dojde k návrhu na jeho úpravu např. přidaní dalšího příkazu, je nutné opět projít všechny řádky pro 30 databází a změnit nastavení pro každou z nich.

Jak vidíme, tak zde naprosto chybí možnost: „Pro tyto databáze udělej tyto příkazy“.

Pokud je takových databází větší počet, tj. i několik set, tak je tato práce značně nepříjemná a opět časově náročná, rutinní opisování a úprava svádí také k nepozornosti a chybám, které mohou mít fatální dopad na provoz.

Shrnutí: Na těchto krátkých příkladech jsem demonstroval nemožnost automatizace některých úkonů a nutnost rutinní manuální práce. Upozornil jsem na případné problémy, které mohou tímto postupem vzniknout a představil motivaci pro tvorbu vlastního programu níže.

4 Vlastní řešení

V této kapitole přichází na řadu vlastní návrh řešení, jedná se o náhradu či vylepšení současného stavu věci popsaného v předchozích částech. První podkapitola se věnuje využitým technologiím a nástrojům použitých pro vlastní řešení.

4.1 Programovací jazyk a technologie

Vzhledem k zaměření a cílovým systémům, jež jsou všechny v režii firmy Microsoft, se automaticky nabízí programovací jazyk C#, který je produktem této firmy a tedy nepřekvapí svoji propojeností s jednotlivými částmi ekosystému Microsoft.

Společně s .NET Framework umožňuje přístup k jednotlivým systémům například k databázovým systémům pomocí sady třídy ADO.NET.

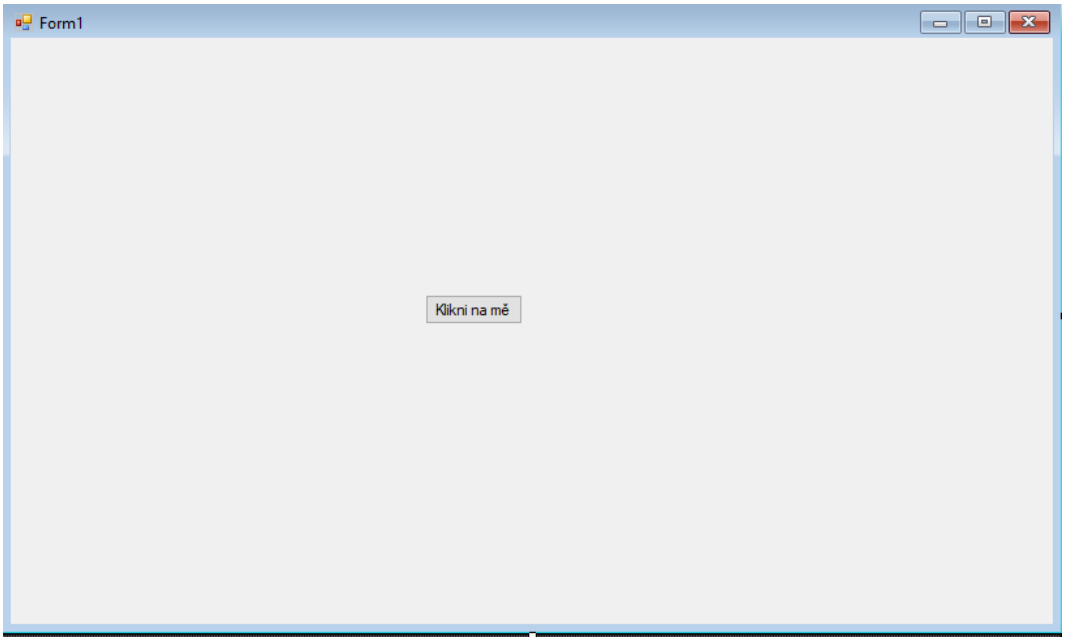
Jazyk C# jsem zvolil také kvůli svým dosavadním zkušenostem s tímto programovacím jazykem. Pro tvorbu programu cílím na .NET Framework 4, vzhledem k plánované podpoře Windows 7, které jsou sice na konci své životnosti, věřím však, že jejich využití stále existuje. Microsoft také umožňuje prodloženou podporu pro organizace s Windows 7, z toho důvodu věřím, že se nejedná o konec systému, byť si uvědomuji brzký zánik podpory.

4.1.1 Grafické prostředí – Winforms vs WPF

Další rozhodnutí tvoří volba mezi grafickým prostředím *Windows Forms App* (zkráceně se označuje jako Winforms) a *Windows Presentation Foundation* (zkráceně WPF). V následující části nastíním pro a proti jednotlivých voleb.

4.1.1.1 Winforms

Winforms je jednoduché uživatelské rozhraní, které je součástí .NET Frameworku, slouží pro tvorbu desktopových aplikací[**3**], nelze tedy použít například pro webové aplikace. Společně s Visual Studiem nabízí jednoduchý návrhář s několika elementy například. Button – tlačítko, Checkbox – zaškrtávací políčko, Textbox – textové pole aj. Obrázek 7 ukazuje reálný vzhled prostředí WinForms, konkrétně formulář s tlačítkem.



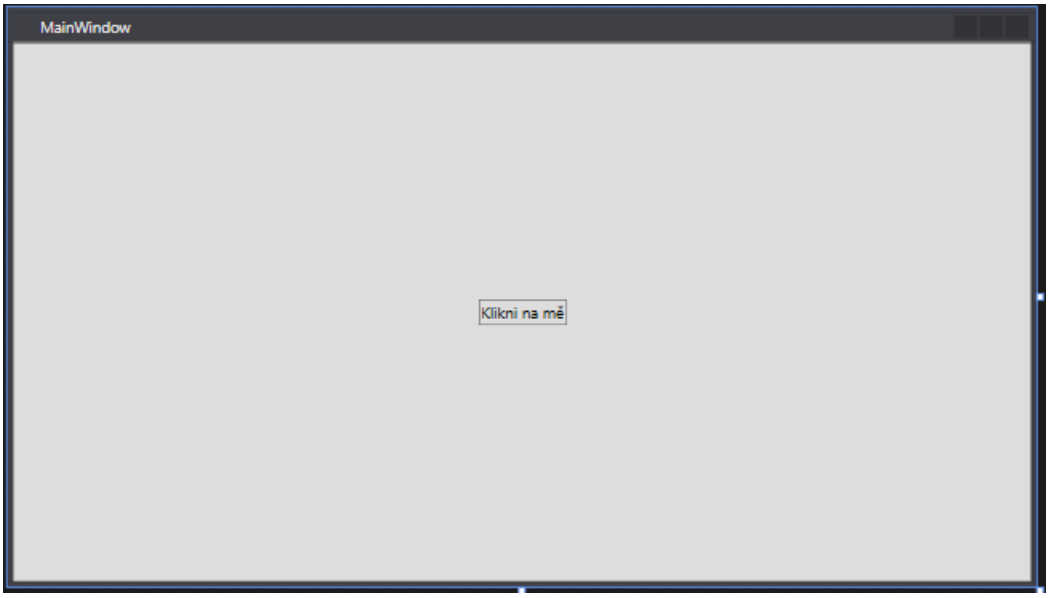
Obrázek 7: Tlačítko „Klikni na mě“ v prostředí Winforms.

4.1.1.2 WPF

WPF používá narozdíl od Winforms tzv. značkový kód (XAML) k implementaci vzhledu aplikace. Aplikaci je možno navrhovat pomocí návrhu v nástroji Visual Studio, nebo pouze pomocí jazyku XAML[**7**]. Například tlačítko je možno do okna přidat pomocí následujících řádků:

```
<Window
    <Button x:Name="button">
    <Button Content="Klikni na mě"/>
    </Button>
/Window>
```

Výsledek tohoto kroku lze vidět na obrázku 8. Pro přidání tlačítka je také možné použít návrh ve Visual Studiu. Předností WPF je také lepší oddělení logiky mezi uživatelským rozhraním a samotným kódem. Jedná se o komplexnější systém a proto je zde také možnost upravit rozhraní dle svých potřeb, narozdíl od WinForms. Naopak ze stejného důvodu je také složitější na naučení a pozdější použití.



Obrázek 8: Tlačítko „Klikni na mě“ v grafickém prostředí WPF.

Verdikt: I přes pokročilost WPF, jsem se rozhodl pro WinForms, pro jeho jednoduchost a okamžitou možnost použití, vzhledem k mým dosavadním zkušenostem a požadavkům na aplikaci, kdy mi tato možnost vystačí, byť uznávám, že WPF má své přednosti a jedná se o novější systém oproti WinForms, které jsou některými považované za zastaralé.

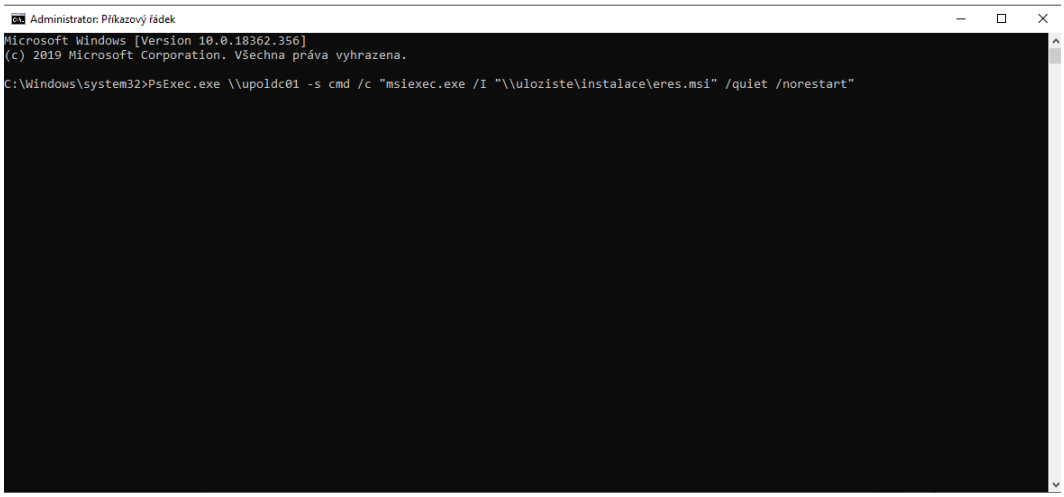
4.1.2 PsExec

Další nástroj, který jsem se rozhodl použít je PsExec. PsExec je nástroj vytvořený Markem Russinovichovem, slouží ke vzdálenému spouštění procesů bez nutnosti instalovat speciálního klienta na cílovém počítači[**8**]. S jeho pomocí je tak možné spustit libovolný příkaz, nebo například instalaci na vzdáleném počítači. PsExec je součástí celé sady nástrojů zvané PSTools od stejného autora.

Princip PsExec je poměrně jednoduchý[**9**]:

- soubor PSEXESVC.exe je nahrán do složky $ADMIN,
- dojde ke vzdálenému vytvoření služby,
- spustí službu.

Služba poté funguje jako wrapper, kdy přesměrovává vstup a výstup mezi hosty přes tzv. named pipes. Díky této službě mohu vzdáleně spouštět příkazy a procesy[**9**], stejně jako kdyby byl na cílovém systému nainstalovaný agent, bez nutnosti instalace a správy agentů (viz obr. 9).



Obrázek 9: Ukázka použití nástroje PsExec, v tomto případě pro instalaci programu ERES na počítač upoldc01.

4.2 Návrh a architektura aplikace

V této kapitole se věnuji samotnému návrhu aplikace a její architektury. Postupně popíši jednotlivé systémy, na které se samotná aplikace zaměřuje a způsob řešení jednotlivých částí. Vysvětlím také důvod proč jsem zvolil daná řešení.

V celém programu a jeho tvorbě si dávám za úkol spojit rychlost a efektivnost potřebného zásahu do cílového systému. Myslím také na fakt, že při takovém zásahu lze brát v potaz větší počet cílových objektů ať už databází, klientů nebo uživatelů.

I vzhledem k těmto požadavkům jsem se rozhodl program pojmenovat **ERES** – zkráceně RS – což znamená Rychlý Systém, v angličtině lze tuto zkratku použít jako Remote Support. Program jako takový jsem pojal primárně v anglickém jazyce z důvodu větší komunity (a tedy případných uživatelů) a především také proto, že anglický jazyk je s naším oborem velmi provázaný. V budoucnu však plánuji dopracovat i českou verzi.

ERES nemá za úkol dodávat složité funkce a přebujelost. Původní myšlenka tohoto programu vznikla z toho důvodu, že mi některé funkce chyběly v současných nástrojích, nebo mi řešení některých problémů připadalo příliš složité popř. zdlouhavé. Dále jsem chtěl umožnit, aby některé úkony, i ty složitější, mohli provádět i lidé, kteří nejsou zdatní v programu PowerShell nebo v jiných nástrojích. I pokud daný uživatel program PowerShell případně ovládá, může v programu ERES najít zjednoduššení. Pokud určitý script nepoužíváte delší časové období a potřebujete jej použít, tak jej musíte znovu napsat, vyhledat nebo si zpětně nastudovat jaká je jeho funkce, pokud jej máte již uložený. Musíte tedy jeden problém řešit opakovaně.

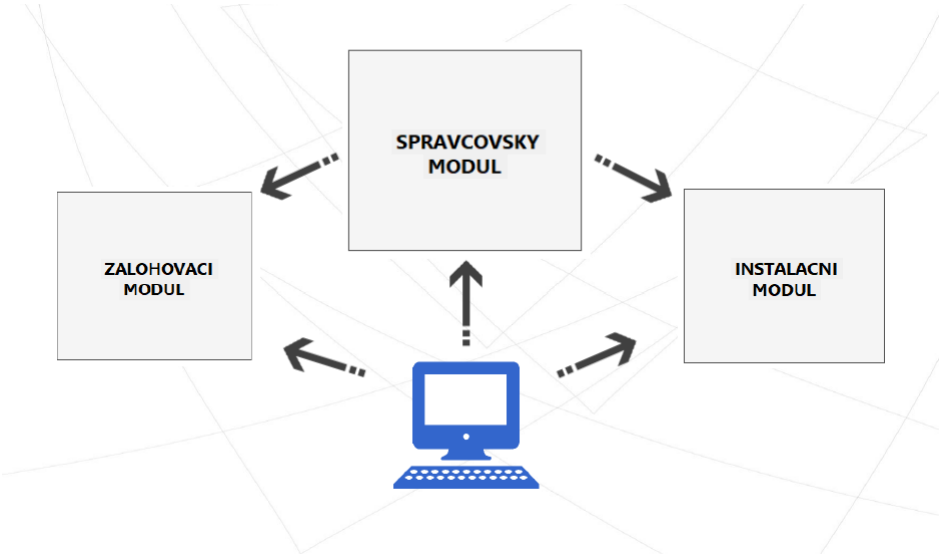
4.2.1 Samostatné celky

Aplikace se zaměřuje na vzdálenou správu několika samostatných celků:

- počítač popř. server jako takový (dále jen klient),
- Active Directory,
- DHCP,
- DNS,
- databázový server (konkrétně MS SQL).

Pro každou z těchto částí je možné vytvořit vlastní třídu a v této třídě obsluhovat jednotlivé systémy. Například pro správu Active Directory můžeme vytvořit třídu *ADManagement.cs*, která bude využívat třídy z oboru názvů *System.DirectoryServices.AccountManagement*, popřípadě ze staršího a složitějšího *System.DirectoryServices*. V tomto případě je tedy nutné si nastudovat jednotlivé třídy a jejich použití.

Rozhodl jsem se jednotlivé systémy rozdělit na tzv. moduly, které budou mít na starosti části funkcionality a budou do jisté míry samostatné. Hlavní výhoda dle mého spočívá v náročnosti na správu takových modulů při co nejmenší spletitosti a také v budoucím vývoji, kdy pro další systémy bude dostatečné vytvoření dodatečného modulu popřípadě využití již stávajících modulů. Na tyto moduly se dá tedy dívat jako na stavební kostky. Příklad modulů je k vidění na obrázku 10. Moduly mezi sebou mohou ale nemusí spolupracovat. Bližší pohled na jednotlivá řešení a výsledek přinesu v následujících podkapitolách.



Obrázek 10: Příklad systému modulů, propojených mezi sebou. Každý modul má na starosti určitou funkci.

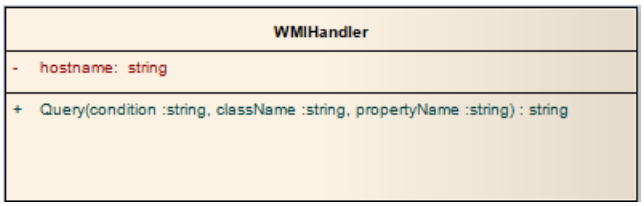
4.2.1.1 Klient

Pro vzdálenou správu počítačů a serverů, dále jen klientů, je potřeba určit cíle a možnosti řešení. Od programu ERES očekávám:

- přehled nainstalovaných aplikací,
- vzdálenou instalaci\odinstalaci aplikace,
- přehled spuštěných služeb,
- vzdálené spuštění\zastavení služby,
- zálohování s funkcí jen a bez*,
- informativní přehled o klientovi.

*V programu označováno jako whitelist a blacklist, tedy zazálohuj jen jmenované, a zazálohuj vše kromě jmenovaného.

Vzhledem k potřebě vyčítání informací bez agentového řešení využívám obor názvů *System.Management*, který umožňuje práci s Windows Management Instrumentation zkráceně WMI. Pomocí WMI lze vyčítat údaje jako je velikost disku, paměti RAM, informace o procesoru aj. Pro tyto potřeby jsem vytvořil třídu *WMIHandler.cs*, která má za úkol vyřizovat WMI dotazy. Diagram této třídy lze vidět na obrázku 11. Tato třída slouží pro obecné účely vyčítání in-

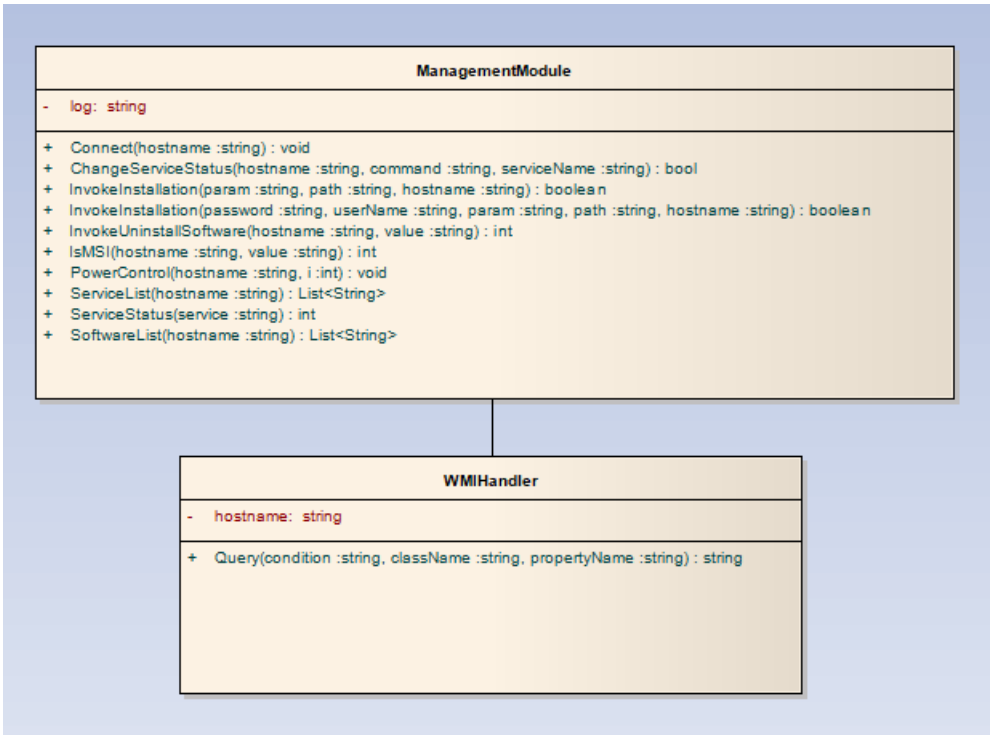


Obrázek 11: Diagram třídy WMIHandler.cs pro vyčítání informací z WMI.

formací z WMI a ne pouze pro správu klientů, proto jsem pro správu klientů vytvořil třídu *ManagementModule.cs*, která mimo jiné doplňuje funkce, které přes WMI nelze řešit, například seznam nainstalovaných aplikací. Seznam aplikací lze vyčíst i přes WMI, bohužel ale získáme pouze aplikace nainstalované pomocí MSI balíčku, vynechá tedy aplikace nainstalované pomocí EXE souboru, což je nedostatečné, proto seznam těchto aplikací vyčítám z registrů, kde nalezneme totožný seznam jako např. v Ovládacích Panelech. Registry obsahují mimo jiné i tzv. *UninstallString*, který obsahuje způsob respektive cestu k odinstalaci programu. Tohoto faktu využiji později k vyvolání odinstalace programu.

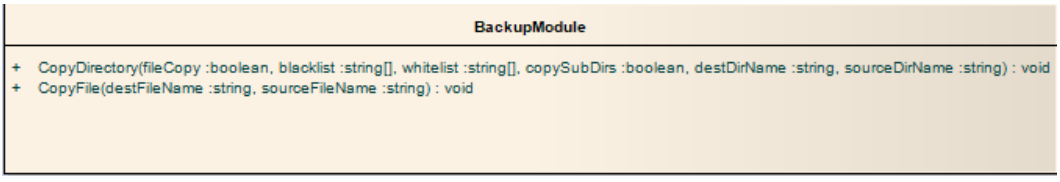
Třída *ManagementModule.cs*, jejíž diagram vidíme na obrázku 12, slouží také k ovládání služeb, respektive ke zjištění stavu, zastavení, spuštění aj. Informace o službách vyčítám mimo jiné i pomocí třídy *WMIHandler.cs*, vzdálená instalace

i odinstalace probíhá díky nástroji PsExec v kombinaci s údaji, které jsem vyčetl z registrů.



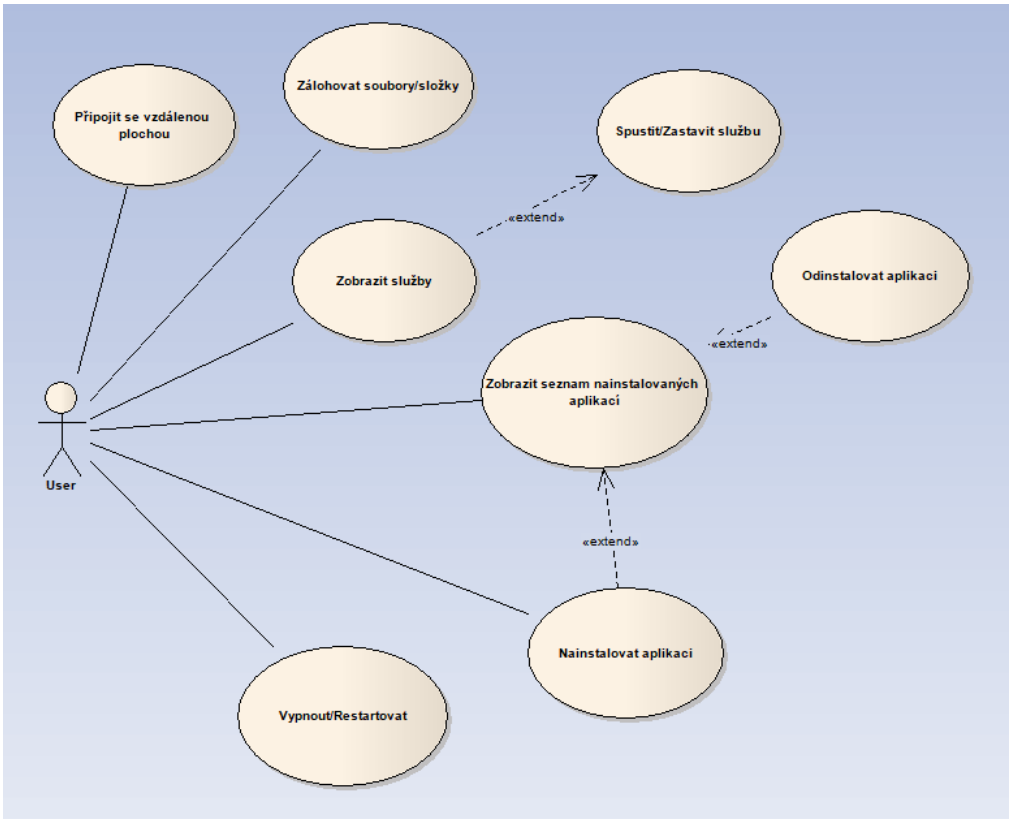
Obrázek 12: Diagram třídy ManagementModule.cs pro správu klientů.

Pro zálohování složek a souborů jsem vytvořil třídu *BackupModule.cs*, která umožňuje zálohování složek a souborů, včetně možnosti tzv. blacklistu a whitelistu, tedy možnosti určení které soubory či složky vynechat, popř. které kopírovat. Tato třída je velmi prostá, jak můžeme vidět na jejím diagramu (viz obr. 13).

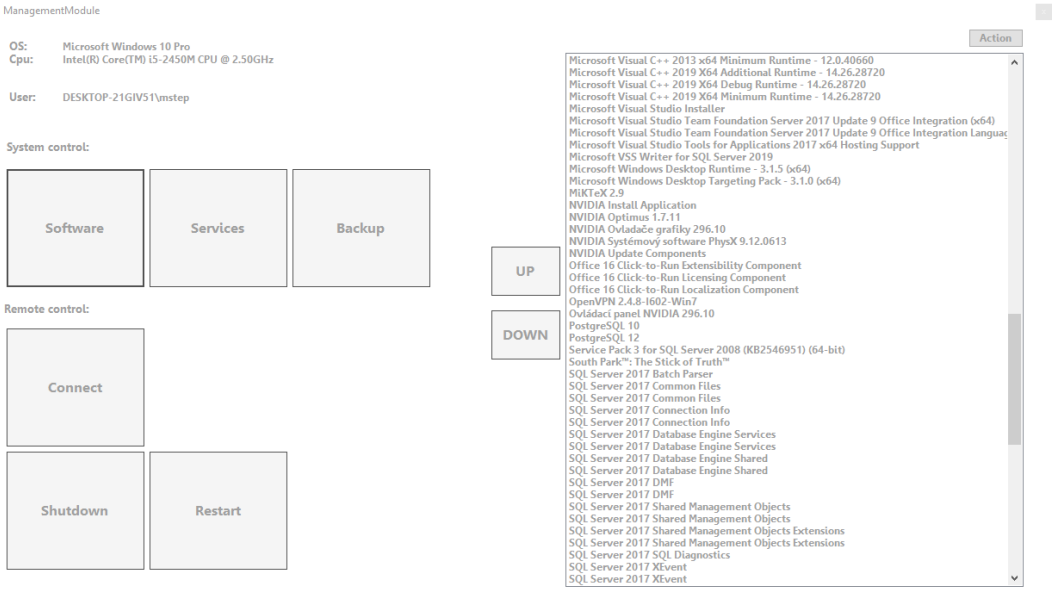


Obrázek 13: Diagram třídy BackupModule.cs pro zálohování souborů a složek.

Tyto moduly jsou společně využívány pro jeden celek, tedy pro správu klientů (viz případ užití na obr. 14). Výhodou tohoto rozdělení je možné využití např. Backup modulu i pro jiné systémy (například kopírování na úrovni souborů by mohl využívat i modul pro správu SQL), popř. pro libovolné volání z různých částí programu. Obrázek 15 ukazuje tuto část v konečném programu – ERES.



Obrázek 14: Případ užití pro správu klientů.



Obrázek 15: Ukázka vzdálené správy v programu ERES.

4.2.1.2 Active Directory

Pro potřebu správy Active Directory v jazyce C# jsem volil mezi dvěma různými přístupy. První je za pomocí *System.DirectoryServices.AccountManagement*, což je obor názvů, se kterým je možné přímo komunikovat s Active Directory, na rozdíl od dalšího oboru názvů tedy *System.DirectoryServices* je jeho použití jednodušší, na druhou stranu tato jednoduchost je vykoupena určitou omezeností při používání pokročilých technik. Dalším možným přístupem je používat oba obory zároveň s tím, že pokud nám nebude stačit první z nich, využijeme druhého.

V první fázi vývoje jsem se uchýlil k první možnosti z nabízených dvou, tedy *System.DirectoryServices.AccountManagement* a byť byl pro mé účely dostatečný, později jsem se rozhodnul celý přístup přehodnotit a přepracovat. Důvodem je, jak také zmíním níže, že systémy jako DHCP a DNS nenabízí podobně kvalitní třídy jako Active Directory v .NET Frameworku a také fakt, že moji snahou je jednoduchý program, jak na správu, tak pro případné programátory. Rozhodnul jsem se všechny tři systémy ovládat pomocí PowerShellu a to z důvodu jednotného systému a možnosti případné uživatele vzdělávat v tomto oboru. Například bude možné provést určitý úkon a uživateli poté představit výsledný script v PowerShellu, tento nápad ohledně tvorby scriptu jsem převzal

z nástroje SQL Server Management Studio a používám ho také v modulu pro správu databázového serveru.

Výsledným rozhodnutím je tedy použít funkcionalitu PowerShellu, byť bych tuto možnost přehodnotil při větším počtu programátorů v týmu. Prakticky využití spočívá v použití oboru názvů *System.Management.Automation*, který umožňuje provádět a volat PowerShellové příkazy. Výhodou je jednotný přístup ke všem částem, rychlejší vývoj vzhledem k tomu, že není potřeba se učit nové třídy pro každý systém zvlášť, stejně tak nové knihovny i od třetích stran. Nevýhodou je dle mého závislost na vnější službě tj. PowerShellu. Vývoj PowerShellu nemám ve svých rukou, funkčnost programu tak závisí na firmě Microsoft a jejím rozhodnutí, také je nutné mít nainstalované moduly v PowerShellu pro správu DNS aj., naštěstí jsou tyto moduly součástí již několikrát zmiňovaného balíčku RSAT, který je téměř nutností pro všechny správce takovýchto systémů a kromě tohoto nástroje tedy není nutné nic dalšího instalovat.

Absence vhodných tříd pro správu DHCP a DNS pro mne na jednu stranu byla překvapením a jistou překážkou, na druhou stranu to program ERES posunulo k větší flexibilitě právě díky modulu pro PowerShell a kromě jmenovaných nevýhod tento směr považuji za vhodný.

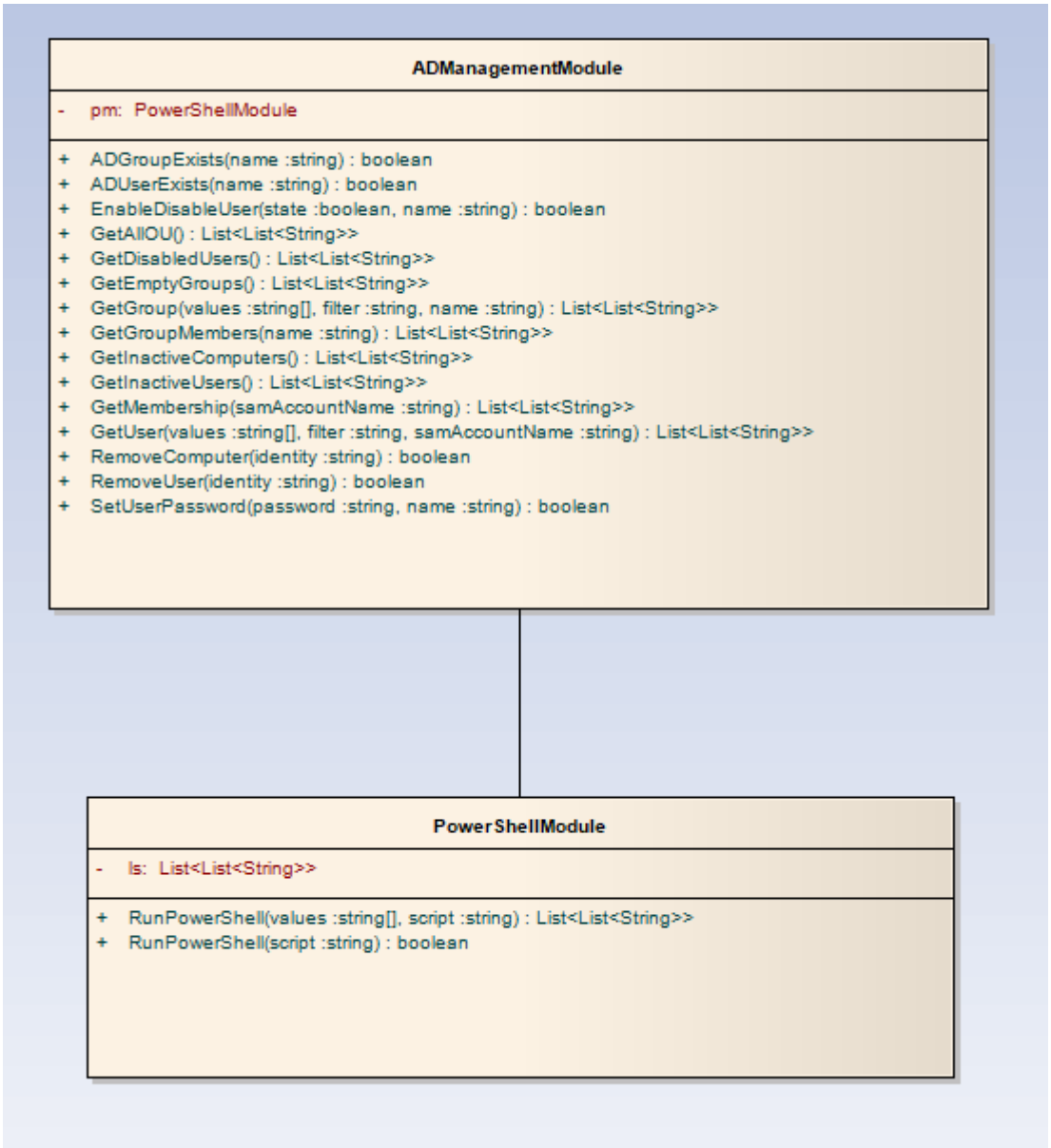
Kromě základní funkcionality jako je vyhledávání, jsem se rozhodnul nabídnout také možnost auditovací funkce pro Active Directory.

Konkrétně je možno auditovat následující:

- neaktivní počítače,
- neaktivní uživatele,
- prázdné skupiny,
- zakázané uživatele.

Za neaktivní je považován počítač či uživatel, který určitý interval nekomunikoval s Active Directory, v mém programu je to konkrétně 60 dní u obou případů. Auditovací funkce budu postupně rozšiřovat.

Pro záležitosti Active Directory jsem vytvořil třídu *ADManagementModule.cs*, která spolupracuje s třídou pro PowerShell modul, konkrétně *PowerShellModule.cs*, obě třídy popisuje blíže diagram tříd na obrázku 16, obrázek 18 následně představuje případ užití pro správu Active Directory.

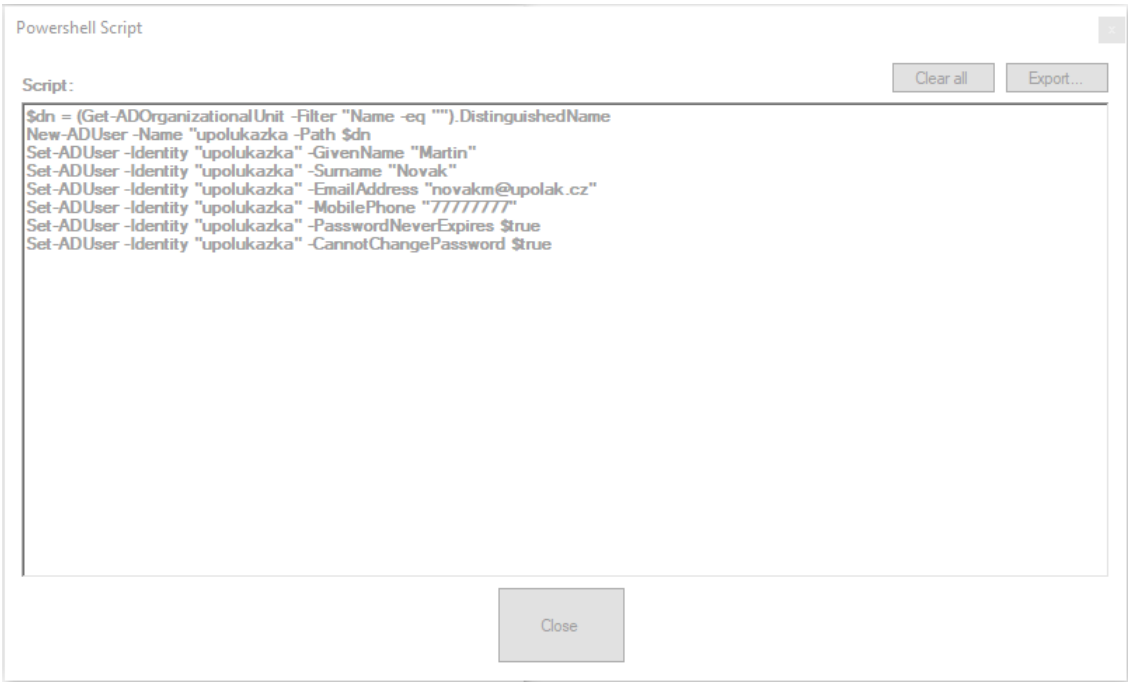


Obrázek 16: Diagram tříd pro Active Directory modul a PowerShell modul.

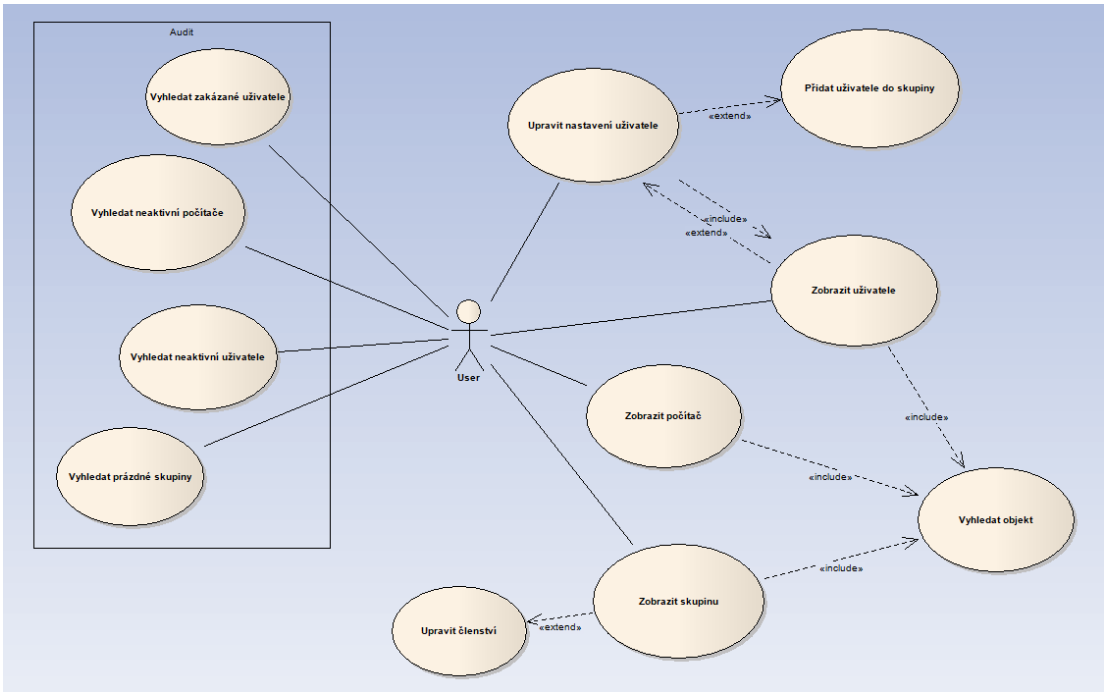
Za výhodu tohoto modulu také považuji fakt, že pokud některá z metod ve třídě *ADManagementModule.cs* není dostačující, může programátor využít *PowerShellModule.cs* pro libovolnou akci a to i pro účely mimo Active Directory.

Nevýhodou by mohl být fakt, že veškerá práce probíhá pomocí PowerShell scriptů a předpokládá se tedy jeho znalost.

Některé z akcí nabízejí také možnost si provedenou činnost tzv. vyscriptovat, tedy vygenerovat PowerShellový script pro daný úkol. Na obrázku 17 vidíme vygenerovaný script pro přidání uživatele.



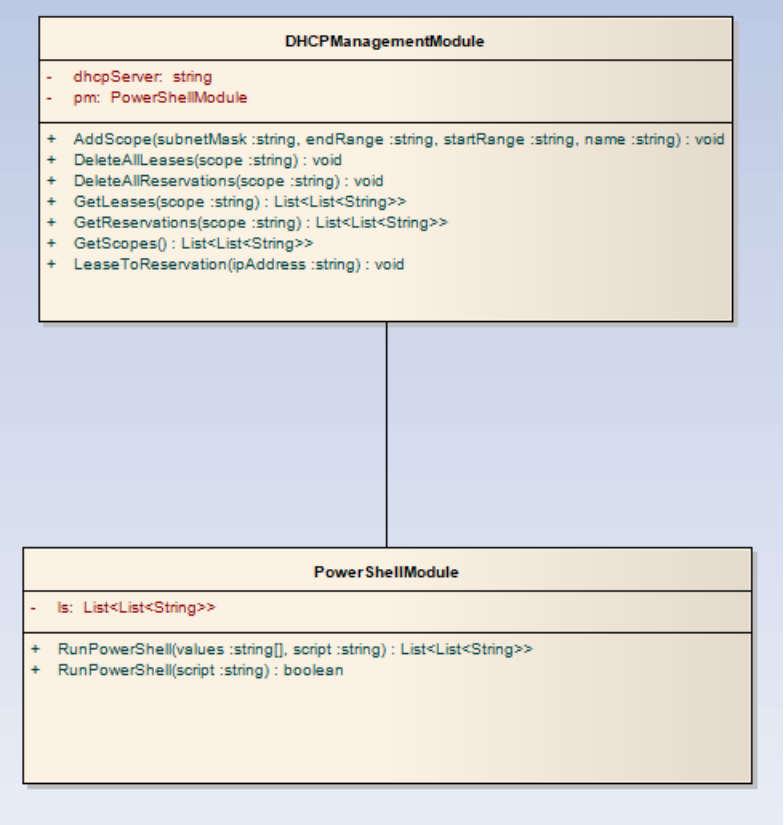
Obrázek 17: Ukázka vygenerovaného scriptu pro přidání uživatele do Active Directory.



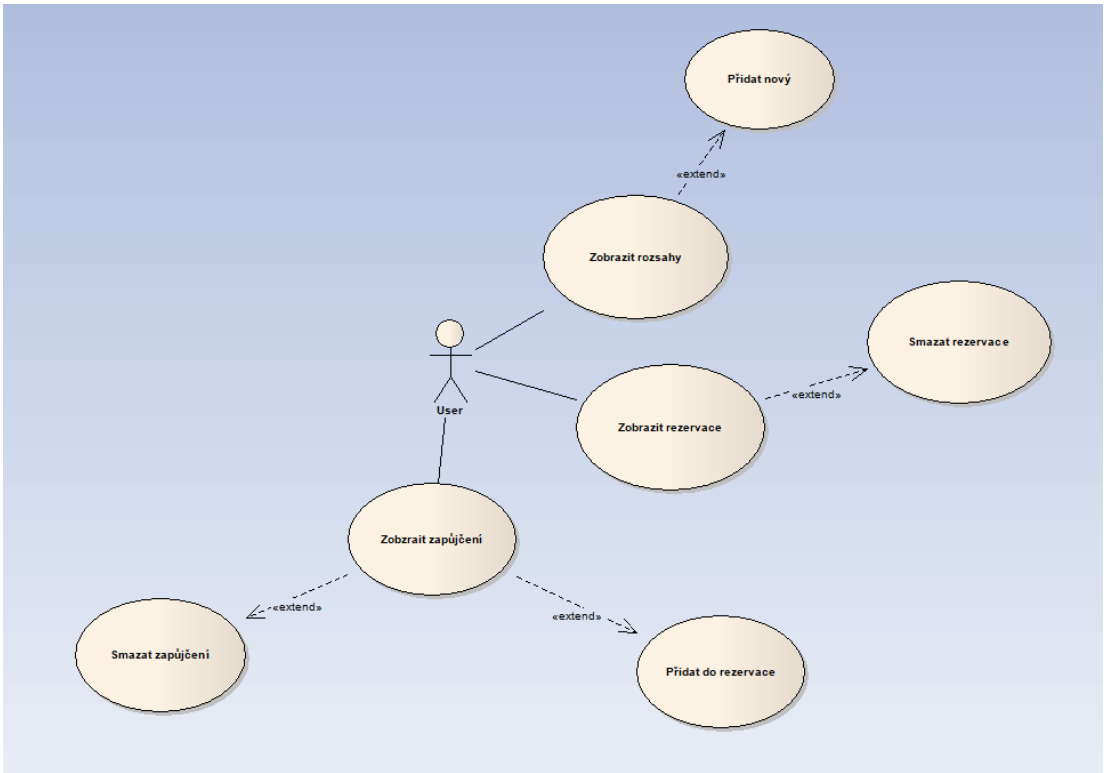
Obrázek 18: Případ užití pro správu Active Directory.

4.2.1.3 DHCP

Pro DHCP .NET nenabízí žádné vhodné třídy k použití, existuje zde ale možnost používat API, konkrétně: *DHCP Server Management API*, díky API je možno se dotazovat DHCP serveru, ale také přidávat záznamy a upravovat konfiguraci DHCP serveru. Jak jsem již zmiňoval výše, zaskočila mne absence knihovny, která by mi pomohla se správou DHCP serveru. API jako takové je použitelné, ale z již zmíněných důvodů výše jsem se rozhodl situaci řešit pomocí PowerShell modulu, který je využíván DHCP modulem (viz diagram tříd na obr. 19). Podotknu, že správou DHCP mám na mysli přehled rezervací popř. vytvoření nové rezervace a tzv. scope, česky by se dalo říct rozsah adres, konkrétní funkce pro správu lze vidět na obrázku 20, který obsahuje případ užití pro DHCP modul.



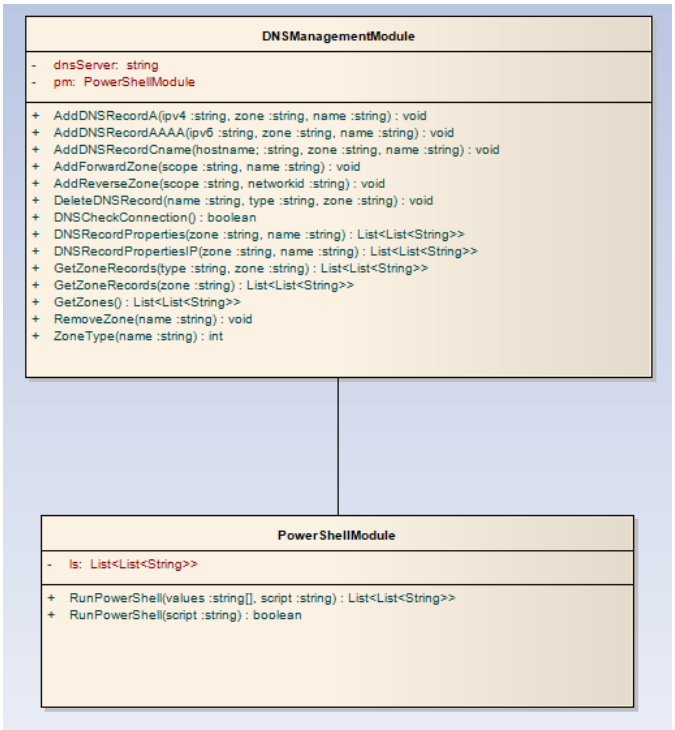
Obrázek 19: Diagram tříd pro DHCP management modul.



Obrázek 20: Případ užití pro DHCP management modul.

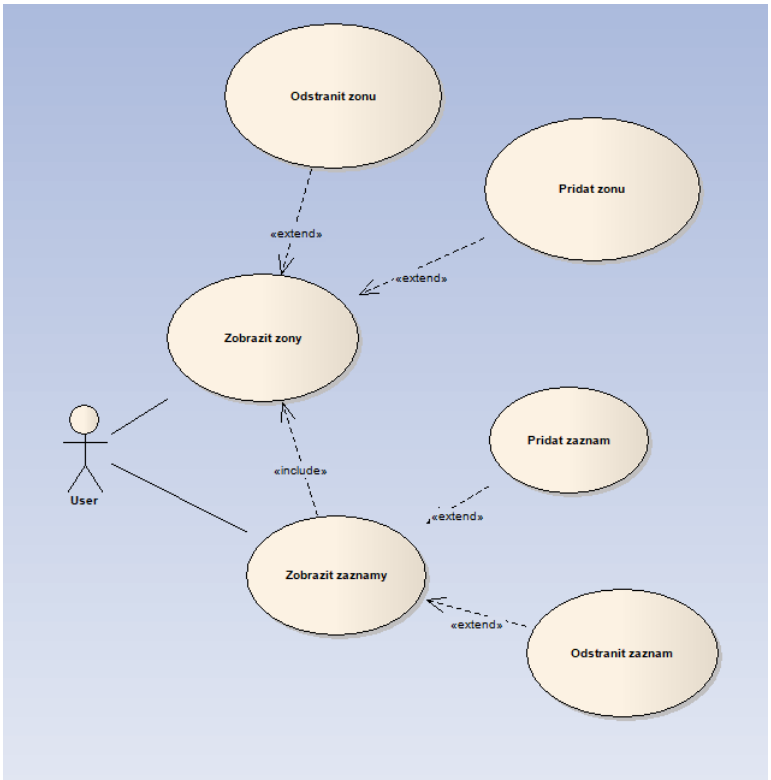
4.2.1.4 DNS

S DNS je to poměrně složitější, .NET nám opět nenabízí žádné třídy ke správě serveru, existuje zde pouze obor tříd *System.net* s třídou Dns, která nám nabízí pouze překlad adres, což je pro můj program nedostatečné. Zajisté existují knihovny třetích stran, těm jsem se rozhodnul vyhýbat do případů absolutní nutnosti, mimo jiné i vzhledem k možnosti případné chybovosti, která by mohla mít dopad na reálný provoz, také mimo oficiální knihovny preferuji vlastní řešení. DNS tedy stejně jako DHCP řeším pomocí PowerShell modulu, který spolupracuje s DNS modulem (viz diagram tříd na obrázku 21).



Obrázek 21: Diagram tříd pro DNS management modul.

Správa DNS pro mě byla ze všech systémů nejnáročnější, narozdíl od jiných částí jako např. SQL jsem správu DNS podcenil s myšlenkou, že se jedná o jednoduchou záležitost. Poměrně rychle jsem narazil na potíže s třídami a rozmanitost všeho nastavení, proto musím přiznat, že funkcionalita této části se nástroji od společnosti Microsoft prozatím nemůže rovnat. Pro potřeby DNS jsem vytvořil třídu *DNSManagementModule.cs*, která staví na dotazech vůči PowerShell modulu. Překážkou pro mě byl způsob vyčítání dat z PowerShellu, kdy jsem potřebná data musel načítat složitěji než u předešlých systémů, například načítání dat z DNS záznamu tzn. IP adresu jsem vyčítal obtížným způsobem. Na obrázku 22 jsou vidět konkrétní funkce, které třída přináší.



Obrázek 22: Případ užití pro DNS management modul.

4.2.1.5 MS SQL

Správa MQ SQL nabízí podobně jako Active Directory dostatečné nástroje již v .NET Frameworku, konkrétně se jedná o třídy ADO.NET, které nám nabízí možnosti propojení s SQL serverem[**3**]. Proces probíhá poměrně jednodušše, na začátku je potřeba vytvořit připojení pomocí tzv. ConnectionStringu, jedná se o string, ve kterém specifikujeme parametry pro připojení, jako je login, zda se jedná o windows login nebo sql login, také můžeme specifikovat výchozí databázi, ke které se připojujeme[**3**]. Na obrázku 23 je vidět ukázka přípojení k serveru SQLUP a instanci SQLSKOLA, toto připojení probíhá pod uživatelem „student“ s heslem „slozitehesloˇ. V tomto případě jsem se rozhodl ponechat přístup přes

```
string connectionString = "Data Source=SQLUP\\SQLSKOLA ;Initial Catalog=master;User ID=student;Password=slozitehesloˇ";
SqlConnection sqlconn = new SqlConnection(connectionString);
```

Obrázek 23: Ukázka připojení k serveru SQLUP\SQLSKOLA pomocí ADO.NET.

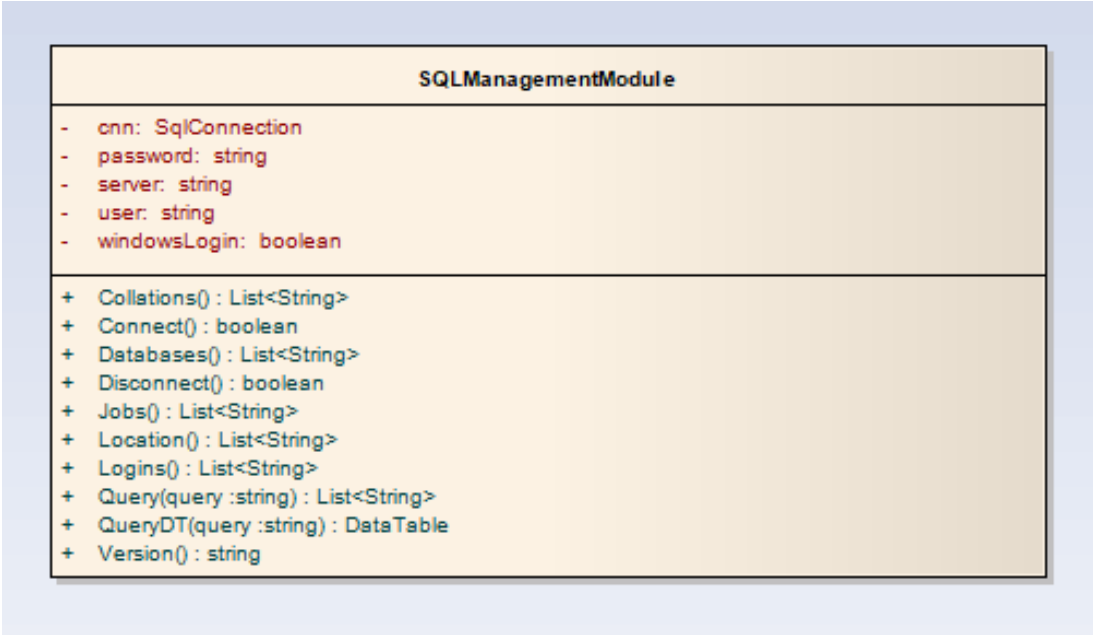
ADO.NET z toho důvodu, že jsem již tuto část měl hotovou a přepracováním do PowerShell modulu bych dle mého spíše ztratil než získal a také proto, že narozdíl od jiných tříd, je ADO.NET jednoduchý na použití.

Vzhledem k tomu, že s databázovým serverem komunikuji pomocí T-SQL je možné výsledné příkazy nabídnout uživateli pro jeho budoucí použití.

Uživatel tedy může zaškrtnout políčko *Script only*, což znamená, že výsledná akce proběhne jen na úrovni tvorby T-SQL scriptu a ten již není poslán k vyřízení.

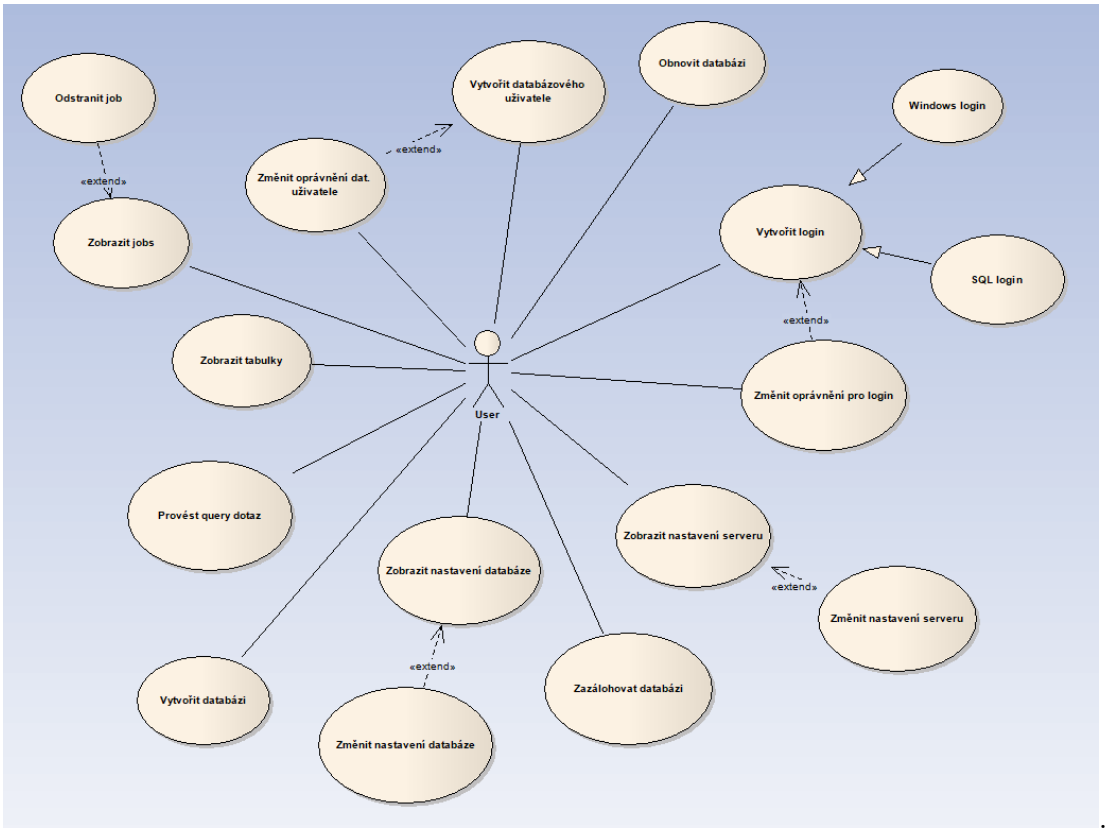
Ke komunikaci s databázovým serverem tedy využívám třídu *SQLManagementModule.cs* místo obvyklého *PowerShellModule.cs*, diagram třídy pro SQL modul lze vidět na obrázku 24. Obrázek 25 představuje jednotlivé funkce, které třída umožňuje.

Další výhodou je fakt, že na dotazy odpovídá databázový server mimo jiné i chybovými hláškami, které specifikují jádro problému a pokud toto hlášení předám uživateli, tak se dozví konkrétní detaily problému a nemusím se tolik soustředit na zachytávání a návrh řešení případných chyb.



Obrázek 24: Diagram třídy pro SQL management modul.

Správa SQL serveru se dá rozdělit na více částí, na server samotný tedy přihlášení na úrovni serveru tzv. login. Nastavení pro záležitosti týkající se serveru jako takového, na úlohy zvané jobs a na databáze samotné, které obsahují vlastní nastavení, vlastní uživatele propojené či nepropojené s loginy.



Obrázek 25: Případ užití pro správu SQL serveru.

4.3 Testovací prostředí

Pro testování jsem původně vytvořil menší prostředí v cloudové službě Azure od společnosti Microsoft, která umožňuje běh virtuálních strojů. Vývoj probíhal delší období a kromě finančních nároků na virtuální prostředí nebyl výkon dostatečný pro mé potřeby, proto jsem se uchýlil k tvorbě domácího prostředí (viz tab. 1). Na jeden z počítačů jsem nainstaloval Windows Server 2019, tento počítač slouží jako hlavní počítač pro služby jako Active Directory, DNS a DHCP, další počítač s Windows 10 obsahuje několik verzí Microsoft SQL Server, na kterých jsem si byl schopný otestovat funkčnost programu i na starších verzích jako je například SQL Server 2008. Poslední počítač obsahuje Windows 7, jedná se tedy o klasickou klientskou stanici, tedy bez serverových instalací a slouží především pro otestování zpětné kompatibility.

Myšlenka podporovat i Windows 7, byť je jeho podpora u konce, vznikla především kvůli rozšířenosti tohoto systému v mém zaměstnání, kdy všechny počítače až na výjimky obsahovaly Windows 7 a využití tohoto programu je mířeno primárně k vlastnímu použití. Bohužel 13. 3. 2020 došlo ke kybernetickému útoku na mého zaměstnavatele, z následného nařízení Národního úřadu pro kybernetickou a informační bezpečnost proběhla reinstalace více jak 3 000 počítačů na Windows 10, tedy hlavní důvod podpory Windows 7 opadnul, přesto doufám,

že program najde využití i na těchto systémech.

Při správě databázové serveru využívám některé techniky, které nemusí být dostupné na SQL Serveru 2008, proto minimální nároky míří na SQL Server 2012. Některé z funkcí programu jsem využil i v praxi, například pokročilé zálohování a správu SQL Serveru.

Tabulka 1: Přehled systémů v testovacím prostředí

Název	Operační systém	Účel
UPOLDC01	Windows Server 2019	AD,DNS,DHCP
UPOL01	Windows 10	SQL
UPOL02	Windows 7	Klient

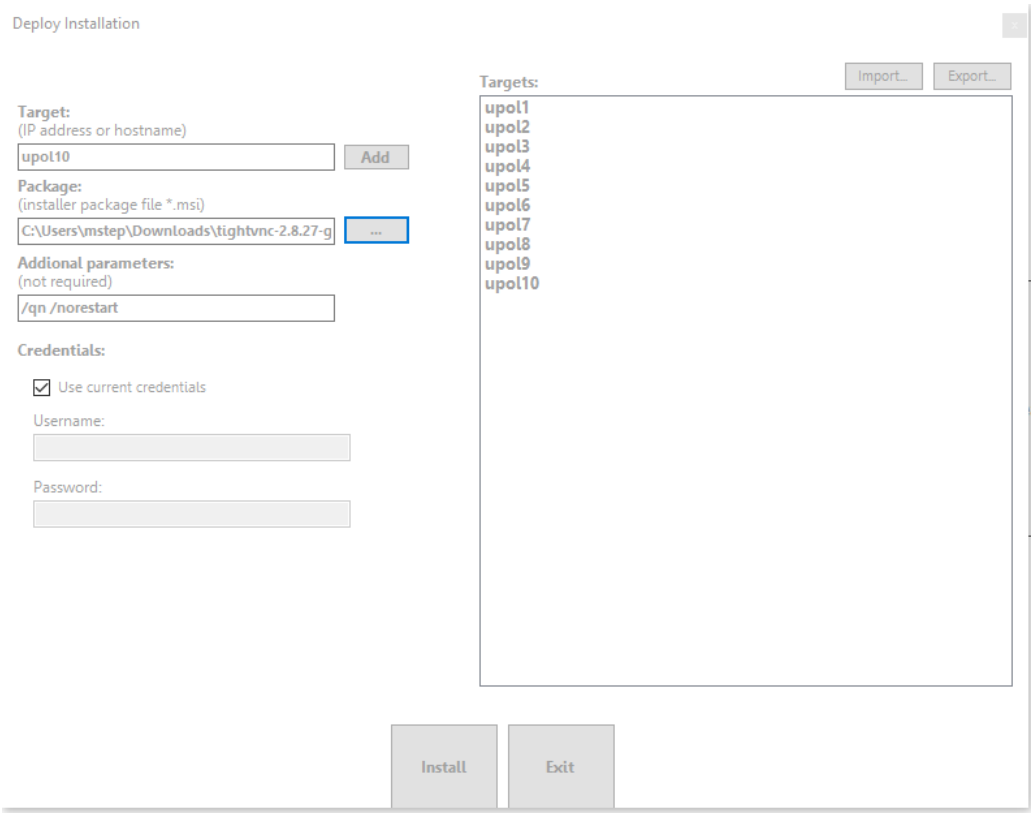
Co se týče funkcionality, běh programu považuji za spolehlivý a v praxi jsem již využil především funkcionality pro klienty a správu SQL, avšak část s DNS považuji za nedokonalou a vhodnou k přepracování nebo velkým úpravám.

5 Příklady z praxe – ukázka řešení

V této kapitole bych rád prezentoval jak se problémy z kapitoly 3, řeší v mém programu – ERES. Jedná se pouze o ukázku, kompletní uživatelská příručka je umístěna v příloze.

5.1 Problém č.1: Instalace programu na 10 klientů

1. Spustíme aplikaci ERES.
2. Vybereme možnost *Deploy Installation*.
3. Po otevření nového okna do kolonky *Target* zadáme název PC.
4. Klikneme na přidat, takto postupujeme pro všech 10 klientů, popř. je možné seznam klientů importovat pomocí TXT souboru tlačítkem *Import....*
5. Do kolonky *Package* zadáme cestu pro MSI soubor. Stav po tomto kroku je zobrazen na obrázku 26.



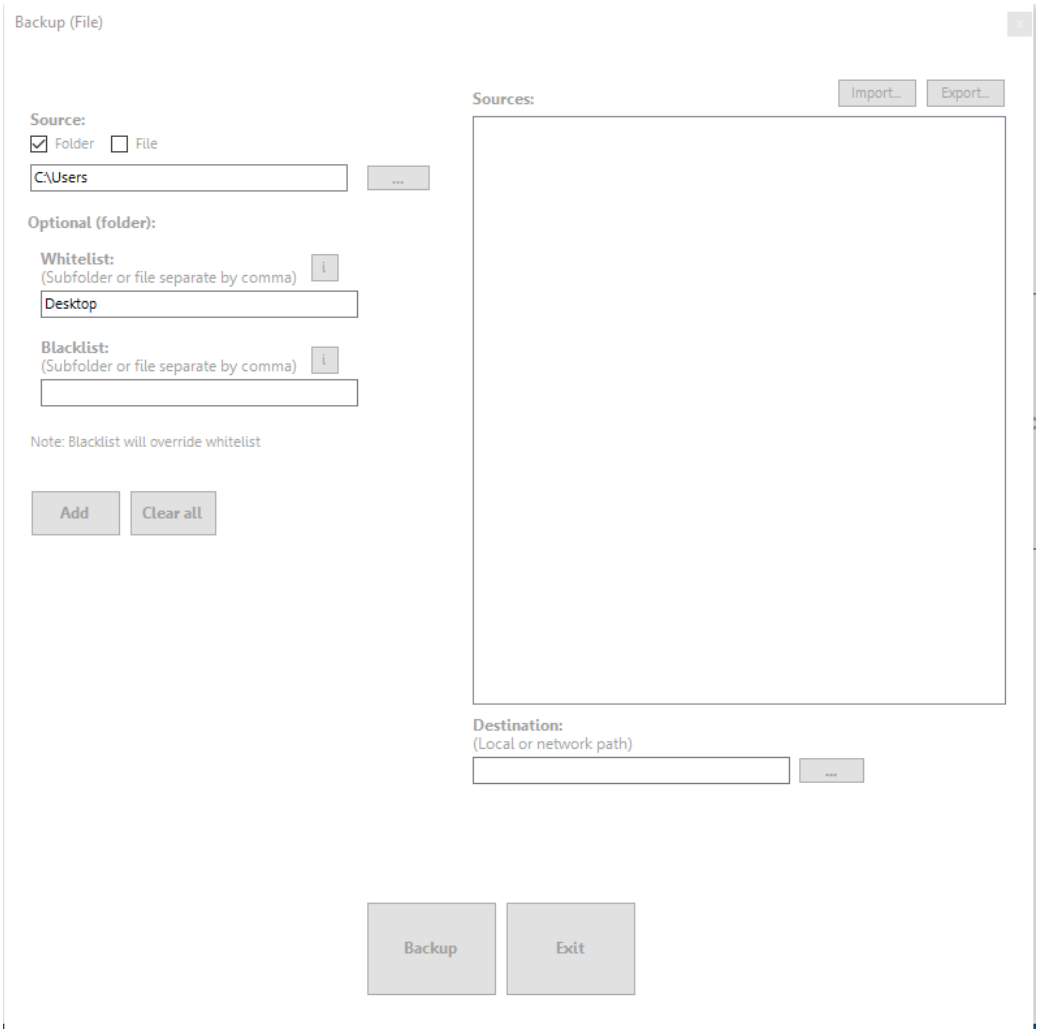
Obrázek 26: Instalace programu TightVNC na 10 cílových klientů.

6. Kliknutí na tlačítko *Install* vyvolá instalaci aplikace na každém z cílových klientů.

5.2 Problém č.2: Zálohování konkrétní složky z uživatelských profilů

1. Spustíme aplikaci ERES.
2. Vybereme možnost *Backup (File)*.
3. Po otevření nového okna do kolonky *Source* zadáme cestu `C:\Users`.
4. Do kolonky *Whitelist* zadáme název složky, kterou chceme zálohovat, tedy *Desktop*. Stav po tomto kroku je zobrazen na obrázku 27.
5. Klikneme na tlačítko *Add*.
6. Do kolonky *Destination* zadáme cestu, do níž se má provést záloha.
7. Klikneme na tlačítko *Backup*.

Ve složce, kterou jsme zadali jako *Destination* se nyní vytvoří složka pro každého uživatele, ve které nalezneme také zálohu složky *Desktop*.

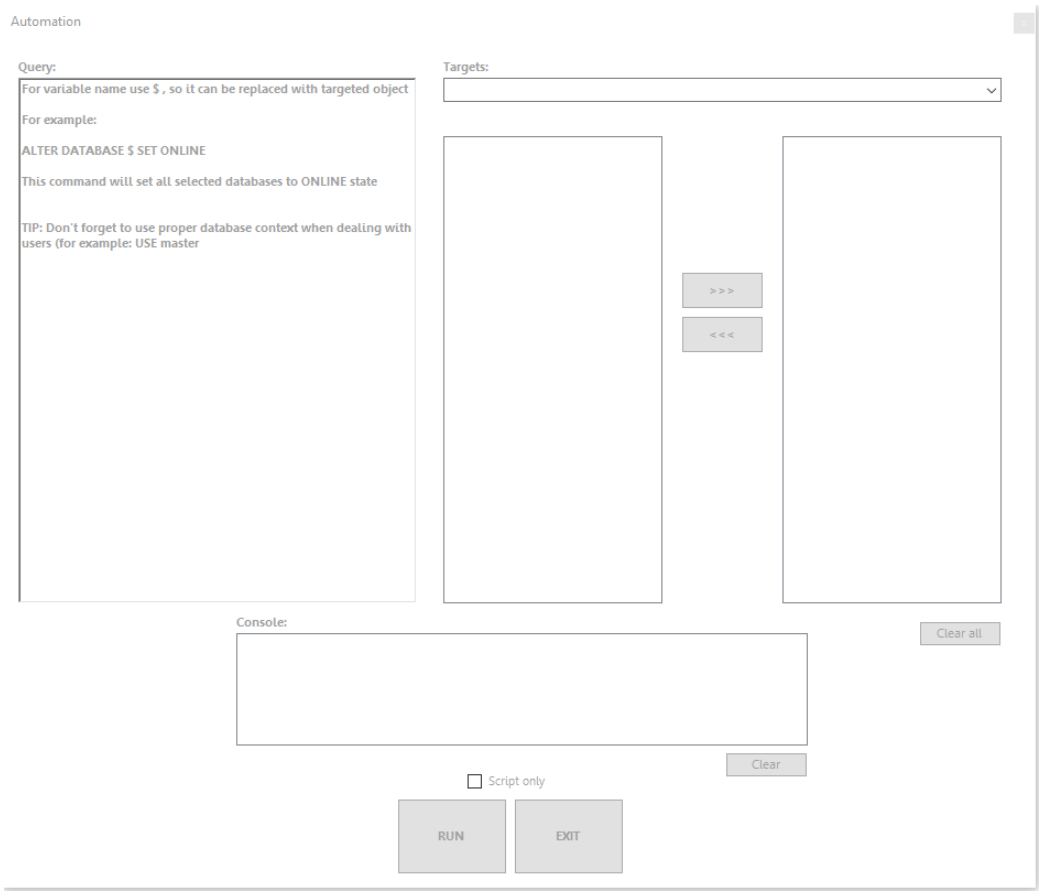


Obrázek 27: Záloha složky Desktop pro všechny uživatele.

5.3 Problém č.3: Přepnutí 30 databází do offline modu

1. Spustíme aplikaci ERES.
2. Do kolonky pro *Search* zadáme název databázového serveru ve formátu SERVER\INSTANCE.
3. Ve vedlejším výběru možností zvolíme možnost *SQL Instance*.
4. Klikneme na tlačítko *Find*.
5. Zobrazí se okno přihlášení, pro SQL login zadáme potřebné údaje, pro Windows login není potřeba zadávat nic.
6. Klikneme na tlačítko *Connect*.
7. Nyní se nám zobrazí několik možností, klikneme na možnost *Automation*.

8. Nyní by se nám mělo zobrazit okno viz obrázek 28.

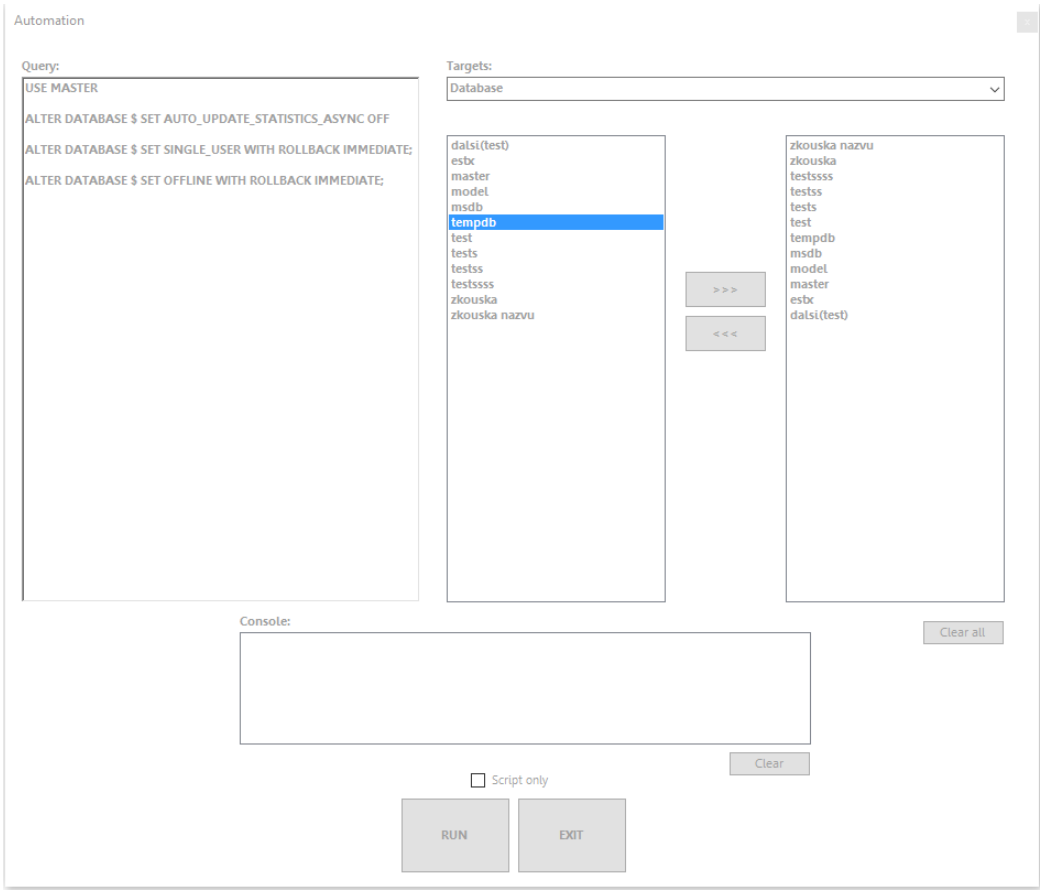


Obrázek 28: Okno pro možnost *Automation* ve správě databázového serveru.

9. Ve výběru možností *Targets* vybereme možnost *Database*, čímž se nám zobrazí seznam databází.

10. Ze seznamu databází vybereme databáze, které si přejeme přepnout do offline modu a tlačítkem $>>>$ potvrdíme tuto volbu.

11. Do textového políčka *Query* napíšeme příkaz, který si přejem vykonat nad každou databází, název databáze nahradíme znakem pro dolar viz obrázek 29.



Obrázek 29: Příprava přepnutí zvolených databází do offline modu v programu ERES.

Závěr

Správa všech jmenovaných systémů je obsáhlé téma a za jednotlivými nástroji k jejich správě stojí velký počet programátorů a dlouhý vývoj. Osobně mě zaskočila komplexnost především správy DNS, respektive možnosti a propracovanost nastavení DNS nástroje, také nástroj pro správu databázového serveru tedy Microsoft SQL Server Management Studio považuji za téměř dokonalý, přesto si myslím, že můj program má své místo a využití, při vší komplexnosti a obrovskému množství funkcí se dle mého zapomíná na jednoduchost a efektivitu při reálném použití.

Program zajisté vyžaduje další vývoj a úpravy, například nejsem spokojený s částí týkající se DHCP, na začátku práce jsem byl skeptický vůči tzv. agentovému řešení, postupně jsem ale přicházel k názoru, že tato varianta má vhodné přínosy a další vývoj by se mohl ubírat i tímto směrem, při řešení agent-server by například nebylo nutností, aby byl cílový počítač zapnutý při vyčítání informací nebo vzdálené instalaci tj. instalace by se spustila jakmile by se spustil počítač a agent si načetl tento požadavek.

Výhody programu vidím v jednoduchosti, snaze zpříjemnit práci a myslet na efektivitu správců systému, výhodou je také, že se jedná o jeden ucelený nástroj a není nutné přepínat mezi vícero nástroji současně. Na příkladech jsem demonstroval jakou časovou úsporu dokáže program přinést i při běžných úkonech.

Další směr vývoje vidím také v centralizaci programu a rozdělení na role. Kdy správce programu bude mít možnost určit role a s nimi i přístupy do programu, respektive jejich omezení dle rolí. Užitečnou funkcí by také byl audit, který by odhalil kdo prováděl konkrétní akce.

Všechny návrhy vývoje však svádí k tomu, porušit původní myšlenku jednoduchého a rychlé systému.

A Příloha č.1: Uživatelský manuál

A.1 Instalace programu ERES

Pro potřeby instalace programu jsou dodávány dva typy instalačních souborů: EXE a MSI. Oba soubory lze použít samostatně, MSI soubor slouží pro potřeby vzdálené instalace. Samotná instalace programu probíhá standardním způsobem.

A.2 Práce s aplikací

1. Spustíme aplikaci ERES.
2. Do kolonky pro *Search* zadáme název cílové stanice, ať už se jedná o databázový server, běžného klienta, DNS, nebo DHCP. Pouze pro potřeby Active Directory není nutné nic zadávat.
3. Ve vedlejším výběru možností zvolíme volbu cílového systému (viz příklad na obr. 30).
4. Klikneme na tlačítko *Find*.
5. Pokud jsme zvolili možnost *SQL Instance*, zobrazí se okno přihlášení, pro SQL login zadáme potřebné údaje, pro Windows login není potřeba zadávat nic.
6. Nyní se zobrazí dostupné moduly pro cílový systém.
7. Po kliknutí na libovolný modul se objeví příslušné okno.

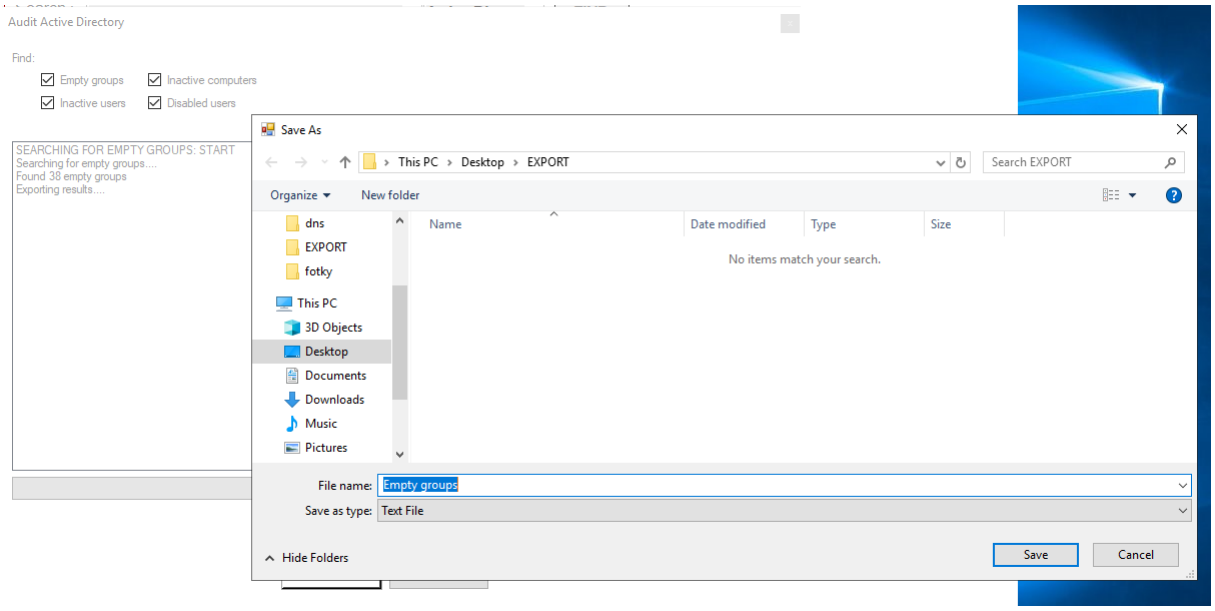
Obrázek 30: Ukázka vyhledávání v programu ERES.

Ovládání jednotlivých modulů je intuitivní a pokročilí uživatelé nenarazí na nic, co by již neznali. Přesto si ukážeme alespoň některé akce. Některé z postupů jsou také k vidění v kapitole 5 *Příklady z praxe – ukázka řešení*

A.2.1 Auditovací funkce pro Active Directory

1. Spustíme aplikaci ERES.
2. Do kolonky pro *Search* zadáme název serveru s nainstalovanou službou pro Active Directory.

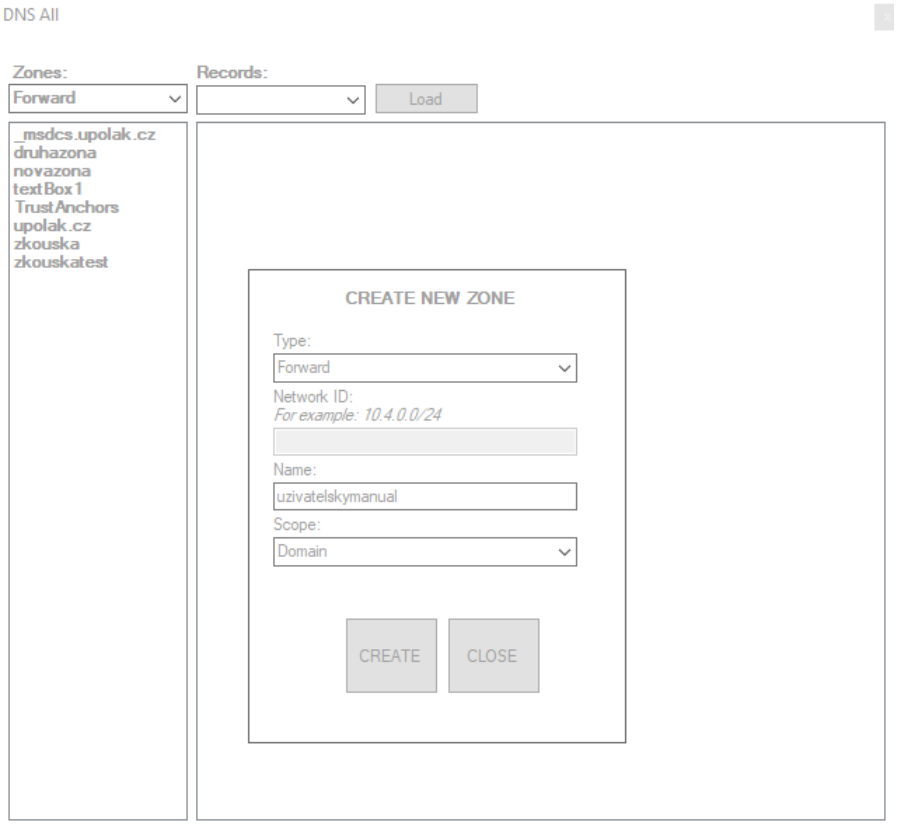
3. Ve vedlejším výběru možností zvolíme volbu *Active Directory*
4. Klikneme na tlačítko *Find*.
5. Pokud jsme vše zadali správně, zobrazí se nám možnost *Audit (AD)*
6. Klikneme na tlačítko *Audit (AD)*
7. Zaškrtneme políčka pro volbu možností, které si přejeme auditovat. (Volbu lze vidět na obrázku 31)
8. Klikneme na tlačítko *ANALYZE*.
9. Pokud nástroj nalezne objekty splňující podmínku naší volby, automaticky nám nabídne export seznamu těchto objektů. Tuto možnost můžeme vidět na obrázku 31.
10. Pokud nástroj nenalezne žádné objekty, export nabídnut není.
11. Po dokončení operace se zobrazí okno informující o jeho konci.
12. Audit Active Directory tímto končí, okno je možné zavřít.



Obrázek 31: Ukázka auditovací funkce pro Active Directory v programu ERES.

A.2.2 Vytvoření DNS zóny

1. Spustíme aplikaci ERES.
2. Do kolonky pro *Search* zadáme název serveru s nainstalovanou službou pro DNS.
3. Ve vedlejším výběru možností zvolíme volbu *DNS*
4. Klikneme na tlačítko *Find*.
5. Pokud jsme vše zadali správně, zobrazí se nám možnost *DNS Management*
6. Klikneme na tlačítko *DNS Management*
7. Objeví se nové okno.
8. Ve volbě pro *Zones*, zvolíme typ zóny.
9. V levém sloupci se zobrazí existující zóny.
10. Klikneme pravým tlačítkem do seznamu.
11. Zobrazí se nám několik možností, vybereme možnost *Create new zone*
12. Objeví se okno pro tvorbu nové zóny.
13. V okně vyplníme nezbytné informace, příklad i okno lze vidět na obrázku 32.
14. Klikneme na tlačítko *CREATE*
15. Zóna byla vytvořena, její přidání můžeme zkontrolovat v seznamu zón.
16. Vytvoření zóny tímto končí, okno i program je možné zavřít.



Obrázek 32: Ukázka tvorby zóny v programu ERES.

B Obsah přiloženého CD/DVD

bin/
: Obsahuje všechny potřebné soubory pro instalaci a běh programu.

doc/
: Text práce v PDF souboru, zdrojový text a všechny přílohy tj. obrázky.

src/
: Obsahuje kompletní projekt včetně zdrojového kódu, knihoven a potřebných souborů pro spuštění programu.

readme.txt
: Obsahuje postup pro instalaci a obsluhu programu včetně požadavků na provoz programu.

Navíc CD/DVD obsahuje:

data/
: Obsahuje video ukázku popisující běh a funkce programu.

Seznam zdrojů

[1] STANEK, William R. *Active Directory: kapesní rádce administrátora*. 1. vydání. Brno: Computer Press, 2009. ISBN 978-80-251-2555-7.

[2] DROMS, Ralph; LEMON, Ted. *The DHCP Handbook*. 2nd edition. Indiana, USA: Sams Publishing, 2002. ISBN 978-0672323270.

[3] AGARWAL, Vidya; HUDDLESTON, James. *Databáze v C 2008: Průvodce programátora*. 1. vydání. Brno: Computer Press, 2009. ISBN 978-80-251-2309-6.

[4] MICROSOFT. *www.microsoft.com*. [online]. [cit. 2020-05-16]. Dostupný také z: ⟨https://docs.microsoft.com/en-us/sql/sql-server/failover-clusters/troubleshoot-orphaned-users-sql-server?view=sql-server-ver15⟩.

[5] MEYLER, Kerrie; SANDYS, Jason; RAMSEY, Greg aj. *System Center 2012 R2 Configuration Manager Unleashed: Supplement to System Center 2012 Configuration*. 1st edition. Indiana, USA: Sams Publishing, 2014. ISBN 978-0672337154.

[6] MICROSOFT. *www.microsoft.com*. [online]. [cit. 2020-05-17]. Dostupný také z: ⟨https://docs.microsoft.com/en-us/sql/relational-databases/databases/set-a-database-to-single-user-mode?view=sql-server-ver15⟩.

[7] MICROSOFT. *www.microsoft.com*. [online]. [cit. 2020-06-05]. Dostupný také z: ⟨https://docs.microsoft.com/cs-cz/dotnet/framework/wpf/introduction-to-wpf⟩.

[8] MICROSOFT. *www.microsoft.com*. [online]. [cit. 2020-06-05]. Dostupný také z: ⟨https://docs.microsoft.com/en-us/sysinternals/downloads/psexec⟩.

[9] MUÑOZ, Daniel. *www.contextis.com*. [online]. [cit. 2020-06-08]. Dostupný také z: ⟨https://www.contextis.com/en/blog/lateral-movement-a-deep-look-into-psexec⟩.

[10] KABELOVÁ, Alena; DOSTÁLEK, Libor. *Velký průvodce protokoly TCP/IP a systémem DNS*. 5. vydání. Brno: Computer Press, 2012. ISBN 978-80-251-2236-5.