FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Inventarizace HW ve firmě s proaktivním sledováním rizik
Student:	Lukáš Pokorný
Vedoucí:	Ing. Miroslav Prágl, MBA
Studijní program:	Informatika
Obor / specializace:	Bezpečnost a informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

Pro inventarizaci hardware připojeného do firemní sítě vyberte vhodné prostředky a navrhněte a realizujte nástroj, který umožní:

- sběr dat:
  - program pro lokální sběr na stanicích (např. název PC, verze OS, přihlášený uživatel, MAC adresa),
  - program pro centrální sběr dat ze serverů / síťových prvků (např. DNS, DHCP, switche),
- předzpracování a uložení dat do databáze,
- vyhodnocení: přehledy a výstupy, vhodné např. pro dohledání, jaký uživatel používá jaký počítač či v jakém portu switche (a tedy lokalitě) je hledané zařízení umístěno, pro vyhodnocování případných rizik (např. docházející místo na disku, přehřívání CPU, slabá baterie) apod.

Přesné vymezení, jaká data je třeba sbírat, jaké výstupy generovat a jaká rizika vyhodnocovat konzultujte průběžně s vedoucím práce.

Elektronicky schválil/a prof. Ing. Pavel Tvrdík, CSc. dne 6. února 2023 v Praze.

Bakalářská práce

INVENTARIZACE HW VE FIRMĚ S PROAKTIVNÍM SLEDOVÁNÍM RIZIK

Lukáš Pokorný

Fakulta informačních technologií
Katedra počítačových systémů
Vedoucí: Ing. Miroslav Prágl, MBA
11. května 2023

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Děkuji vedoucímu mé bakalářské práce Ing. Miroslavu Práglovi, MBA za odborné, praktické a cenné rady, které mi pomohly tuto práci zkompletovat.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona

V Praze dne 11. května 2023 .....

Abstrakt

Tato bakalářská práce se zabývá analýzou, návrhem a implementací systému sloužícímu k inventarizaci hardwaru ve firmě. Data, která se z počítačů odesílají na server, jsou šifrovaná algoritmem RSA. Celý systém je rozdělen na inventarizační část a část sledování rizik. V inventarizační části se posílají veškeré informace, které je potřeba o počítači znát. V části sledování rizik se posílají data, která byla vyhodnocena jako indikátor případného rizika a ta jsou odesílána mnohem častěji. Těmito informacemi jsou detekovány případné bezpečnostní, ale i funkcionální problémy. K implementaci celého sytému byl využit programovací jazyk C#, jako databáze byla vyhrazena instance Microsoft SQL Serveru a pro komunikaci s databází byl použit framework Dapper. Jelikož téma práce je velmi komplexní, je v praktické části, po dohodě s vedoucím práce, realizovaná jen vybraná část. O to větší důraz je kladen na možnosti budoucího rozšíření, kterými se zabývá samostatná kapitola v poslední části práce.

Klíčová slova inventarizace hardwaru, sledování rizik, bezpečnostní rizika, C#, síťové rozhraní, .NET Windows Forms, klient-server

Abstract

This thesis deals with the analysis, design, and implementation of a system for hardware inventory in a company. The data sent from computers to the server are encrypted using the RSA algorithm. The entire system is divided into an inventory part and a risk monitoring part. The inventory part sends all the necessary information about the computer, while the risk monitoring part sends information that has been identified as an indicator of potential risk, and this information is sent much more frequently. These pieces of information detect potential security, as well as functional problems. The C# programming language was used for the implementation of the entire system, with an instance of Microsoft SQL Server serving as the database, and the Dapper framework used for communication with

the database. Since the thesis topic is very complex, only a selected part is implemented in the practical part of the work, in agreement with the thesis supervisor. Therefore, greater emphasis is placed on the possibilities of future expansion, which are discussed in a separate chapter in the last part of this thesis.

Keywords hardware inventory, risk monitoring, security risks, C#, network interface, .NET Windows Forms, client-server

Seznam zkratek

BIOS	Basic Input-Output System
CRUD	Create Read Update Delete
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
GUI	Graphical User Interface
NÚKIB	Národní úřad pro kybernetickou a informační bezpečnost
OAEP	Optimal Asymmetric Encryption Padding
ORM	Object-Relational Mapping
OS	Operační Systém
RSA	Rivest–Shamir–Adleman
SQL	Structured query language
SSH	Secure Shell
UNC	Universal naming convention
WMI	Windows Management Instrumentation

Úvod

Počítače a jiná elektronická zařízení jsou dnes součástí každé firmy a vlastně i každodenního života. Jen těžko by si člověk mohl představit život bez těchto zařízení, jelikož většina z nás je využívá jak ke své práci a obživě, tak v osobním životě. V některých profesích se s počítači setkáváme více než v jiných. Mnoho firem, společností, institucí a dalších jiných zařízení jich má již tak mnoho, že mají vlastní IT oddělení, které tyto přístroje spravuje.

Inventarizace je poměrně kritickou součástí správy IT. IT oddělení může spravovat desítky, stovky i tisíce počítačů. Pamatovat si informace o takovém množství zařízení není pro člověka možné. Z tohoto důvodu využívají různé inventarizační softwary, které jim pomáhají jejich práci zefektivnit. Těchto softwarů je velké množství a mají mnoho různých funkcí. Patří do nich například inventarizace hardwaru či softwaru jednotlivých počítačů, inventarizace síťových prvků, provádění auditů, poskytování reportů či automatické plánování skenu počítače. Můžeme se setkat jak s placenými, tak s open-source aplikacemi.

Tématem této práce je vyvinout podobný software pro zvolenou firmu. Tato firma již disponuje programem na inventarizaci hardwaru. Nicméně s postupem jeho využívání v praxi vznikly určité nové požadavky, které tento software nesplňuje. Přínosem této bakalářské práce je vytvoření nové aplikace pro danou firmu a její používání v praxi. Z výše uvedených důvodů jsem se rozhodl pro téma „Inventarizace HW ve firmě s proaktivním sledováním rizik“.

Práce se zabývá analýzou, návrhem a implementací systému, který provádí inventarizaci počítačů a síťových prvků. Celá práce je rozdělená do těchto tří hlavních částí. Jelikož se jedná o velmi komplexní téma, je ve čtvrté části kladen důraz na možná vylepšení.

Na bakalářskou práci bych rád navázal i diplomovou prací, ve které bych vylepšil stávající funkce a rozhraní a aplikaci bych rozšířil o další funkcionality.

Kapitola 1
Cíle

Cílem této práce je navrhnout a realizovat nástroj, který usnadní inventarizaci hardwaru v rámci IT oddělení. Tento program bude sbírat data na lokálních stanicích, serverech a síťových prvcích. Sběr dat bude probíhat v administrátorem definovaných časových intervalech.

Dále se budou data bezpečně, tedy šifrovaně, posílat po interní síti na server, kde se předzpracují, uloží do databáze a vyhodnotí.

Hodnotit se budou i případná rizika, jako je například docházející místo na disku, přehřívání CPU a bezpečnostní rizika jako je vložení flash paměti do počítače.

Mým hlavním cílem je, aby byla aplikace co nejvíce užitečná v praxi a maximálně usnadňovala práci pracovníků IT oddělení. Pro splnění tohoto hlavního cíle jsem si zvolil i několik cílů dílčích. Prvním z nich je provedení analýzy již existujících nástrojů, které slouží k inventarizaci v IT. Následovat bude jejich zhodnocení a důvod, proč tedy implementuji něco již existujícího. Druhý zahrnuje návrh samotného řešení korespondujícího s požadavky. Posledním je samotná implementace podle návrhu.

Kapitola 2
Analýza

Tato kapitola je věnována problematice inventarizace a sledování rizik z pohledu teorie. Dále je soustředěna na již existující řešení, jenž umožňují inventarizaci a jejich porovnání. V kapitole jsou také zadefinovány požadavky, které jsou na aplikaci kladeny. V neposlední řadě je věnována doménovému modelu a bezpečnosti aplikace.

2.1 Inventarizace

Pojem inventarizace definuje Zákon č. 563/1991 Sb. takto: *„Účetní jednotky inventarizací zjišťují skutečný stav veškerého majetku a závazků a ověřují, zda zjištěný skutečný stav odpovídá stavu majetku a závazků v účetnictví“*. [1] Definice je zaměrená především na obor účetnictví, ale tato práce se zabývá invantarizací z jiného úhlu pohledu. Jejím cílem je shromáždit informace o počítačích tak, aby byly snadno dostupné a přehledné.

Rád bych zdůraznil, že inventarizace obecně je důležitou součástí každé společnosti. Počítače jsou potřeba téměř v každé firmě a je nutné je nějakým způsobem evidovat. Jejich evidence je nezbytná, aby bylo zřejmé, z jakých komponent jsou počítače sestaveny, jaké mají komponenty vlastnosti, jaká síťová rozhraní daný počítač obsahuje a jaké je jeho fyzické umístění. Díky inventarizaci lze tedy snadno z jednoho místa zjistit všechny tyto údaje. To je určitě výhodné pro úsporu času zaměstnanců, protože nemusí všechny informace sbírat manuálně na reálných místech napříč celou firmou nebo dokonce napříč mnoha pobočkami, které mohou být rozmístěny po celém světě.

2.2 Sledování rizik

Další částí nástroje je sledování rizik. V tomto bodě je třeba nahlédnout do tématiky bezpečnosti. V této souvislosti se používají tři pojmy, kterými jsou informační,

počítačová a síťová bezpečnost. Tyto pojmy mají mnoho společného a taktéž sledují stejný cíl, avšak drobné rozdíly mezi nimi nalezneme. Jejich cílem je: *„ochrana důvěrnosti, integrity (celistvosti) a dostupnosti informaci“*.[2] Práce se věnuje bezpečnosti počítačové a síťové.

Stav počítačové bezpečnosti je koncepčním ideálem dosaženým použitím tří procesů: prevence, detekce a náprava rizik.[3] Tato práce se věnuje pouze detekci. Následná náprava detekovaných rizik je zajištěna pracovníkem IT oddělení firmy.

Síťová bezpečnost zahrnuje ochranu dat během jejich přenosu. Práce se bezpečnému přenosu věnuje v sekci 2.7.1.

V následujících podkapitolách jsou definována rizika, která bude výsledný nástroj detekovat a upozorňovat na ně. Vzhledem k potenciálu tohoto systému by měl být program navžen tak, aby bylo v budoucnu snazší rozšiřovat sledovaná rizika podle potřeb či požadavků.

2.2.1 Funkcionální rizika

Jako vhodné případy funkcionálních rizik byly vybrány:

1. přesáhnutí určitého procentuálního limitu kapacity disku,
2. přesáhnutí určitého teplotního limitu procesoru,
3. neplánované odpojení disku

2.2.2 Bezpečnostní rizika

Jako vhodné případy bezpečnostních rizik byly vybrány:

1. detekce nového disku (včetně flash disku),
2. přihlášení nového uživatele, který není evidován v databázi,
3. nadměrná síťová komunikace,
4. změna DNS serveru,
5. změna DHCP serveru,
6. změna výchozí brány.

2.3 Existující řešení

Na trhu je k dispozici mnoho softwarových produktů, které provádí správu a monitorivání IT infrastruktury. Každý z těchto produktů je unikátní a disponuje trochu odlišnými řešeními. Některé provádí správu pouze sítí, jiné hardwaru či softwaru, ale lze se setkat i s produkty, které nabízí jejich kombinaci. Tyto produkty jsou většinou placené, ale nalezneme i bezplatné zkušební verze nebo produkty, které jsou zcela zdarma. Pro účely této práce byly zvoleny následující příklady:

SolarWinds Server & Application Monitor je založen na serveru a monitorujících aplikacích, které poskytují automatizovanou inventarizaci. Jsou sledovány klíčové informace, jako je místo na disku, CPU, paměť a síťová rozhraní. Kromě hardwaru jsou poskytovány i reporty softwaru. Tento produkt je ideální pro velké podniky.[4]

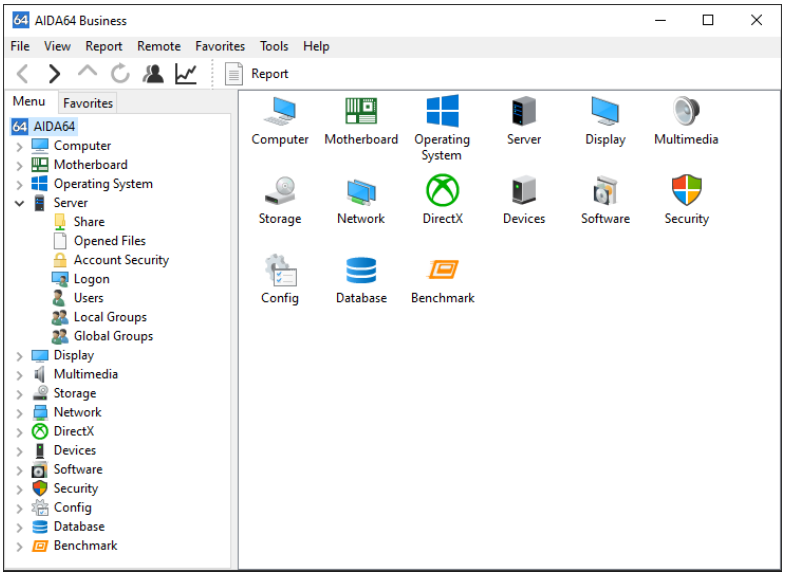
Atera je založen na cloudové platformě, která obsahuje balíček nástrojů pro vzdálený monitoring a správu klientů. Pracovník IT oddělení tedy může k programu přistupovat odkudkoli prostřednictvím webového prohlížeče. Jsou zaznamenávána data o hardwaru i softwaru.[4]

ManageEngine Endpoint Central je multiplatformní systém. Může být využíván stanicemi s Windows, macOS i Linux. Za příplatek je nabízena i správa mobilních zařízení se systémem iOS a Android. Je poskytována inventarizace hardwaru i softwaru a zahrnováno automatizované sledování výkonu s výstrahami. Tento produkt je ideální pro střední a velké organizace, nicméně nabízí bezplatnou edici pro 25 koncových stanic, takže je atraktivní i pro malé podniky.[4]

Spiceworks Inventory je bezplatný nástroj pro správu hardwaru i softwaru. Dokáže automaticky detekovat zařízení připojená k síti. Je kompatibilní pouze s Windows a macOS. Je velmi vhodný pro malé podniky z důvodu jednoduchosti, a protože je zcela zdarma.[4]

AIDA64 je poslední příklad nástroje pro inventarizaci, který je popsán, jelikož firma, kde bude nástroj testován, tento produkt vlastní. Je nabízen jak produkt pro domácí počítače, tak i do organizací. V době, kdy je práce psána, jsou nabízené čtyři různé produkty. Jejich hlavním produktem je AIDA64 Business, který nabízí diagnostiku hardwaru, síťový audit, vzdálený přístup a správu změn.[5]

Na obrázku 2.1 je zobrazeno základní okno po spuštění aplikace. Je zde nastíněno, jaké všechny okruhy informací program *AIDA64 Business* zjišťuje. Kromě hardwaru zobrazuje i informace o operačním systému a jeho různá nastavení. Uživatel si také může zobrazit informace o softwaru jako jsou všechny programy, plánované úlohy či programy, které se spouští po spuštení počítače.



Obrázek 2.1 Základní okno programu AIDA64 Business[6]

Nástroj *AIDA64 Business* kromě skenu jednoho počítače nabízí i možnost auditu. Uživatel si může zvolit, které počítače oskenovat a program poté zobrazí výsledek ve statistice, kde jsou uvedeny výsledky. Například se může jednat o informaci, kolik procent počítačů disponuje jakým operačním systémem.[6]

Ve firmě je prováděna inventarizace programem AIDA64. Postupem času a zapojením programu do praxe byly nalezeny nedostatky, především v části sledování rizik. Program, který je vytvářen v rámci této práce, má za cíl tyto nedostatky odstranit. Zejména z důvodu správy a bezpečnosti počítačů je sledování rizik velmi užitečné. Mohou se detekovat potenciální bezpečnostní hrozby a také funkcionální problémy, které mohou způsobit nefunkčnost některého z počítačů.

2.4 Konkrétní požadavky od firmy

Úkolem práce je napsat nástroj pro inventarizaci hardwaru ve firmě. Tento nástroj bude testován v nejmenované firmě, která již disponuje podobným programem, konkrétně programem AIDA64. Tento program je popsán v sekci 2.3.

Proto byly požadavky upraveny tak, aby aplikace obsahovala funkcionality, kterými program AIDA64 nedisponuje. Mezi tyto funkcionality patří zejména část sledování rizik. Celý systém bude implementován pouze na operační systém Windows 10 a novější, jelikož je nainstalován na většině počítačů ve firmě.

Oficiální požadavky jsou rozděleny na dvě části, a to požadavky funkční a nefunkční.

2.4.1 Funkční požadavky

S vedoucím práce a vedoucím IT oddělení firmy jsme se shodli na následujících funkčních požadavcích.

2.4.1.1 F1 - Evidence počítačů

Systém bude inventarizovat hardwarové prvky počítačů jako jsou disky, paměti, procesory, ale i sériová čísla, názvy počítačů a operační systémy včetně jejich verzí. Data budou sbírána klientem a následně posílána na server, kde se uloží do databáze.

2.4.1.2 F2 - Evidence uživatelů

V sytému je potřeba i evidence uživatelů, kteří se na počítač přihlásí, především kvůli zajištění bezpečnosti. Systém bude tuto evidenci umožňovat. Data o přihlášení budou ukládána do databáze.

2.4.1.3 F3 - Evidence síťových rozhraní

Program bude provádět inventarizaci síťových rozhraní. Evidována budou síťová rozhraní na jednotlivých počítačích, jejich název, popis, MAC adresa, IPv4 adresa, maska sítě, DHCP server, všechny jeho DNS servery, a pokud počítač má, tak IPv6 adresa. Tyto prvky budou opět uloženy v databázi.

2.4.1.4 F3 - Evidence síťových prvků

Program bude provádět inventarizaci switchů Dell X1052P, jelikož právě tyto prvky firma vlastní. Uživatel zadá přiřazenou IP adresu, jméno a heslo pro přihlášení a program následně zjistí přiřazení MAC adres k určitým portům.

2.4.1.5 F4 - Vyhodnocování rizik

Systémem budou sledována rizika jak funkcionální, tak zároveň bezpečnostní. Všechna tato rizika jsou popsána v samostatné kapitole 2.2.1. Data budou načtena z databáze a poté vyhodnocena.

2.4.1.6 F5 - Upozornění na rizika

Aplikací budou rizika po vyhodnocení zobrazována tak, aby pracovník IT oddělení dokázal co nejrychleji rozpoznat problém a okamžitě na něj zareagovat.

2.4.2 Nefunkční požadavky

Nefunkční požadavky jsou požadavky, které definují omezení a rozsah aplikace. Shodli jsme se na následujících.

2.4.2.1 N1 - Klient-server

Existuje mnoho způsobů, jak provádět inventarizaci. S vedoucím práce jsme se shodli na tom, že celý systém bude implementován na principu klient-server. Část klienta bude spuštěna na počítačích a data budou odesílána na část serveru. Zde se budou zpracovávat, ukládat a vyhodnocovat.

2.4.2.2 N2 - Zabezpečení přenosu dat

Data budou posílána po interní síti, ale i přesto je třeba, aby byl přenos zabezpečen a aby nemohla být nikým sledována/podvržena.

2.4.2.3 N3 - Velikost dat

Systém bude spuštěn nepřetržitě několik let. Za tu dobu bude objem přijatých dat obrovský, a proto je nutné, aby se data, vzhledem k velikosti, ukládala co nejefektivněji.

2.5 Případy užití

V této kapitole jsou popsány případy užití nástroje na inventarizaci, které vycházejí z požadavků uvedených v předešlé kapitole. Jedná se o funkcionality, které budou systémem poskytovány účastníkům.

2.5.1 Účastníci

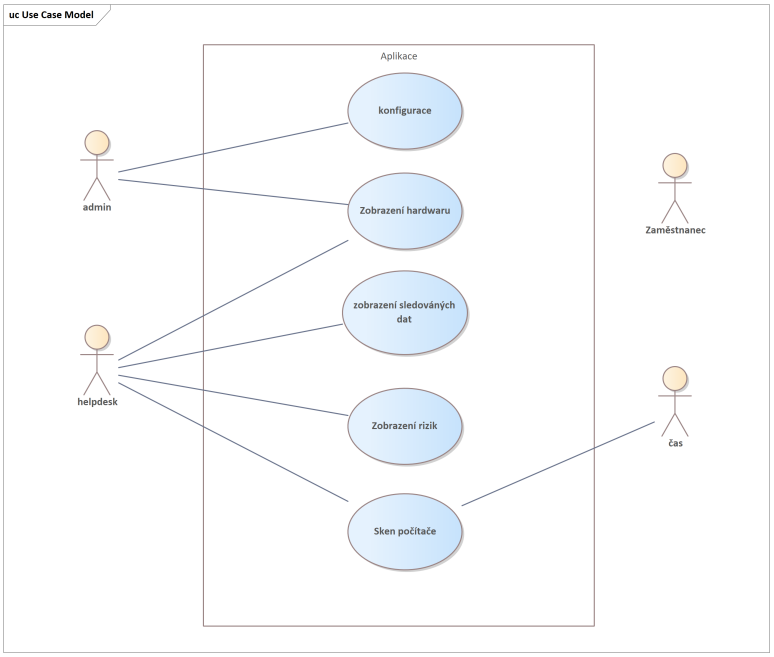
Nejprve je třeba definovat účastníky (role), kteří budou systém využívat.

Pracovník IT oddělení — role reprezentující pracovníka IT oddělení. Především on bude s celým nástrojem pracovat a bude reagovat na rizika, která budou nástrojem vyhodnocena.

Admin — reprezentuje roli pracovníka IT oddělení, který bude mít za celý systém odpovědnost.

Čas — role reprezentující čas.

Zaměstnanec — tato role je zastoupena zaměstnancem firmy, který není pracovníkem IT oddělení. V modelu je jen pro zdůraznění, že s celým systémem nebude nijak interagovat.



Obrázek 2.2 Use case model

2.5.2 Konfigurace

Admin bude moci konfigurovat nastavení celého systému. Lze nastavit zvlášť jak stranu klienta, tak stranu serveru. Na klientovi lze konfigurovat cesta ke klíči, který bude použit k šifrování, IP adresa a port serveru. Na serveru lze nastavit cesta k soukromému klíči, databázi, IP adresu a port, na kterém bude server spuštěn. Dále je možné nakonfigurovat limity pro hlášení jednotlivých incidentů.

2.5.3 Zobrazení informací o hardwaru

Pracovník si bude moci vyhledat konkrétní počítač a zobrazit si všechny jeho údaje. Zobrazí se nové okno detailu počítače, kde budou základní údaje počítače jako jsou disky, paměti, procesor, síťová rozhraní a jiné informace, které je vhodné do tohoto okna umístit.

2.5.4 Zobrazení sledovaných dat

Pomocí zobrazení sledovaných dat bude možné vidět data, která slouží ke sledování rizik. V tabulce lze například sledovat teplotu procesoru v čase, využití disku v čase či sledování historie přihlášených uživatelů na daném počítači.

2.5.5 Zobrazení rizik

Pracovník IT oddělení si bude moci zobrazit všechna rizika, které program vyhodnotil a určitým způsobem na ně zareagovat.

2.5.6 Sken počítače

Sken počítače se bude provádět v definovaném časovém intervalu.

2.6 Doménový model

Pro vytvoření popisu, definování a vazeb mezi entitami je využíván doménový model. Slouží také k přiblížení problematiky tématu a na jeho základě je postaven celý systém. Model byl z důvodu přehlednosti rozdělen na 2 části — inventarizační část a část sledování rizik.

2.6.1 Inventarizace

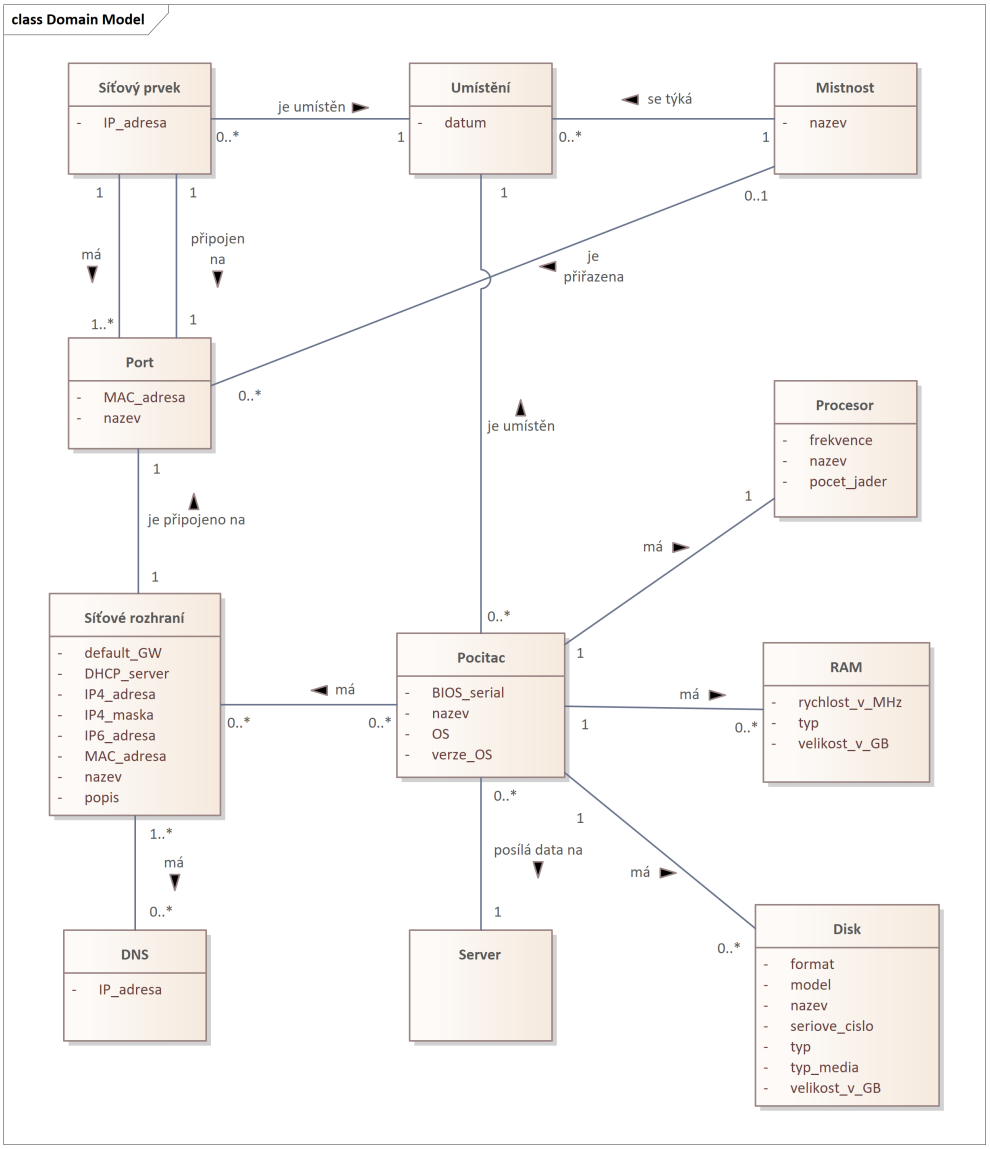
V této části modelu je příblíženo, jaká data se na jednotlivých počítačích získávají. Jde o základní informace o počítači a především o nejdůležitější data týkající se disků, pamětí a procesoru. Mezi další důležitá sledovaná data patří informace o síťových rozhraních na jednotlivých počítačích, kde je sledován jejich počet a nejdůležitější údaje, jako je například IP adresa, výchozí brána a mnoho dalších (viz obrázek 2.3). Ke každému rozhraní se z důvodu bezpečnosti sbírají DNS servery.

Inventarizací se provádí i evidence switchů. Ze switche program získává informace, na kterém portu je přiřazená jaká MAC adresa. Díky evidenci portů přiřazených k jednotlivým místnostem ve firmě můžeme namapovat konkrétní umístění počítačů.

2.6.2 Sledování rizik

V druhé části modelu je nastíněno, jaká data se získávají v části sledování rizik. Jedná se o teplotu procesoru, jejíž hranici může nastavit role admin v konfiguračním souboru. V případě, že teplota překročí stanovenou hranici, je hlášen nový incident, který se vytvoří pouze jednou jako prevence proti zahlcení databáze duplicitními informacemi.

Kromě teploty je sledována velikost volného místa na každém z disků. Stejně jako u teploty je možné nastavit procentuální hranici využitého místa na disku v konfiguračním souboru. V případě jejího překročení dojde k nahlášení nového incidentu. Zároveň lze zaslanými informacemi identifikovat vložení či odebrání některého z disků. Toto bezpečnostní opatření zahrnuje i vložení či odebrání flash



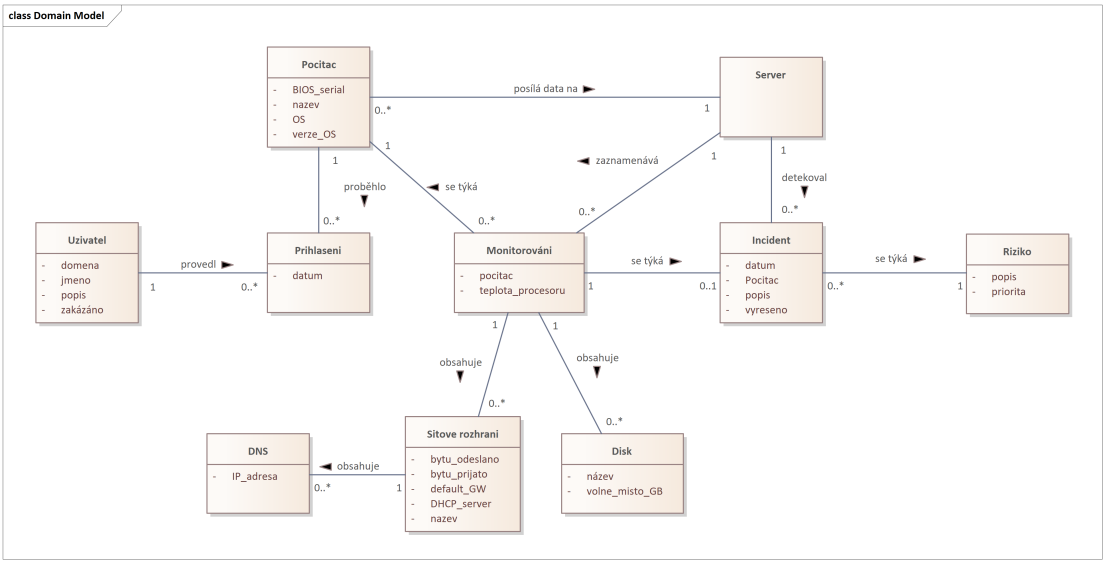
Obrázek 2.3 Inventarizace v doménovém modelu

disku, které patří mezi velká bezpečnostní rizika. Tímto lze rozpoznat i nefukčnost některého z disků, což může velmi usnadnit práci pracovníka IT oddělení. Data, která jsou zaznamenávána jak o teplotě, tak o volné kapacitě, lze sledovat v průběhu časové osy. To umožňuje přehledně a efektivně využít data v rámci statistického pozorování.

Další data, která jsou v této části aplikace sledována, se týkají aktuálního stavu síťových rozhraní. Tímto monitorováním jsou získávány například informace o možné nadměrné komunikaci, způsobené potenciálně nebezpečným stahováním či odesíláním dat. To může být například způsobeno v případě, že počítač obsahuje nějaký malware, který odesílá citlivá data mimo firmu. Také je kontrolována

výchozí brána, DHCP server a DNS servery a programem je upozorněno na případnou změnu jakýchkoliv sledovaných dat. Tyto informace však nejsou v databázi uloženy, jelikož by byly často opakované a docházelo by tak ke zbytečné duplikaci dat. Program by tak měl být schopen rozpoznat například man-in-the-middle útok[1] či jiný problém způsobený změnou některého z údajů.

V neposlední řadě jsou monitorována data o uživatelích, kteří se na počítač přihlásili. Vždy je sledováno poslední přihlášení aktuálního uživatele a v případě, že se liší od posledního záznamu, dojde k uložení nové informace. To znamená, že v případě, že je uživatel přihlášen po celý den, je uložen pouze jeden záznam o jeho přihlášení například v 9 hodin, který se stále opakuje. V případě, že se ale během dne odhlásí a znovu přihlásí, dojde k zaznamenání nové informace a následuje její uložení.

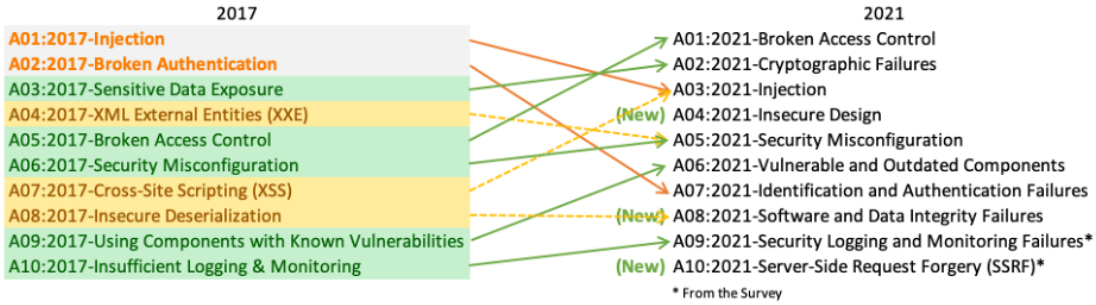


Obrázek 2.4 Sledování rizik v doménovém modelu

2.7 Bezpečnost aplikace

Proto, aby byl celý nástroj použitelný v praxi, je třeba zanalyzovat a minimalizovat případná bezpečnostní rizika. V této části jsou popsány nejčastější zranitelnosti, které by mohl podle [7] systém obsahovat (viz obrázek 2.5).

[1] Komunikace mezi 2 stranami, které se domnívají, že spolu komunikují, ale jejich komunikace prochází přes útočníka, který zprávy odposlouchává, přeposílá a případně mění.



Obrázek 2.5 Nejčastější zranitelnosti podle [7] pro roky 2017 a 2021

2.7.1 Šifrování přenosu dat

Na druhé příčce [7] se nachází zranitelnost nazvaná jako „Cryptographic Failures“. Jelikož systém bude obsahovat citlivá data ohledně stavu počítačů v celé firmě, je velmi důležité zajistit, aby byla ochrana těchto dat dostatečná a bezpečná.

Přestože se informace o počítačích budou posílat přes interní síť, je nutné data zašifrovat. Potencionální útočník může například přes zranitelnost jakékoli jiné aplikace odposlouchávat komunikaci ve vnitřní síti.

Rozhodl jsem se, že v systému budou data přenášena šifrovaně pomocí algoritmu RSA. Tento algoritmus je založen na asymetrické kryptografii, což znamená, že existuje pár klíčů. Říká se jim privátní a veřejný klíč. Veřejný klíč je použit pro šifrování dat. Jak už název napovídá, veřejný klíč může znát kdokoli, kdo chce šifovaná data poslat, proto je možné ho bezpečně distribuovat. Naopak privátní klíč slouží k dešifrování dat a v ideálním případě ho zná pouze jeho majitel.

RSA se stal nejčastěji používaným šifrovacím systémem, jelikož nabízí zajištění ochrany, platnosti a nezměnitelnosti šifrovaných dat. [8]

2.7.1.1 Princip šifrování

1. Máme dvě stejně velká prvočísla p a q.
2. Definujeme $n = pq$ a $\phi(n) = (p-1)(q-1)$.
3. Vybereme náhodné číslo e, aby platilo $gcd[e, \phi(n)] = 1$ a $1 < e < \phi(n)$.
4. Najdeme exponent d v mezích $1 < d < \phi(n)$ aby platilo: $ed = 1 mod \phi(n)$
5. Dvojice (e,n) je veřejný klíč, dvojice (d,n) je soukromý klíč.

Zašifrování zprávy m pak vypadá: $E(m) = |m^e|_n = c$ a dešifrování poslané zprávy c vypadá: $D(c) = |c^d|_n = |m|_n$. [8]

2.7.2 SQL Injection

Na třetím místě [7] se nachází zranitelnost „Injection“. Tento typ útoku hrozí v případě, že uživatel zadá vstup použitý v SQL dotazu, který se předtím nekontroluje. Uživatel tedy může vložit místo očekávaného vstupu řetězec, díky kterému se aplikace chová jinak, než programátor zamýšlel.[9, 10] Například tento jednoduchý autentizační dotaz očekává od uživatele jeho jméno a heslo.

```
sql = "SELECT id FROM users WHERE username='" +
            user + "' AND password='" + pass + "'"
```

Tento dotaz působí neškodně, avšak v případě, že uživatel zadá jako heslo řetězec „password’ OR ’5’=’5“ vznikne tento dotaz:

```
SELECT id FROM users
WHERE username='user' AND password='password' OR '5'='5'
```

Jelikož rovnost 5=5 je vždy pravdivá, dojde k tomu, že celá podmínka bude splněná a příkaz vrátí první ID z tabulky uživatelů. Tím navíc často bývá administrátor, což znamená, že se útočník dostane do aplikace bez autentizace, a navíc i s administrátorskými právy.[11]

Dále je možné uvést příklad parametrizovaného dotazu:

```
SELECT * FROM Directory WHERE LastName LIKE '${NAME}';
```

Při zadání příslušného vstupu vznikne následující dotaz. Tímto dotazem se dostaneme ještě o stupeň dále.

```
SELECT * FROM Directory WHERE LastName LIKE 'frank'
    OR 1=1
    UNION SELECT user, password
    FROM mysql.user WHERE 'q'='q'
```

V případě, že jsou dostatečná práva pro uskutečnění tohoto dotazu, tak tento upravený příkaz načte úplný seznam tabulky uživatelů MySQL včetně hesel. Naštěstí tento dotaz právě z důvodu oprávnění velmi pravděpodobně selže. Nicméně útočník tento vstup může vyzkoušet, protože stále existují aplikace běžící s plnými právy.[9]

Tento typ útoků umožňuje útočníkovi nejen odhalit všechny informace o databázi, ale také číst, aktualizovat, měnit nebo odstraňovat data, která jsou v databázi uložena. Mezi obranné mechanismy proti tomuto typu útoků patří mimo jiné dodržování principu nejmenších oprávnění, kontrola vstupů nebo používání placeholderů. [10]

2.7.3 Další zranitelnosti

Dále [7] uvádí i další zranitelnosti a doporučení, jak jim zabránit. Jde například o selhání autentizace, kde je nutné, aby se znemožnilo automatizovanému prolomení hesel (hrubou silou). Tomu lze zabránit jednodušše tím, že například po třech neúspěšných přihlášeních se systém na minutu zablokuje. Také je třeba, aby se hesla držela různých doporučení. Například NÚKYB vydal vyhlášku [12], kde jsou definována doporučení pro minimální požadavky na tvorbu hesla. Bezpečnost hesel je ale velmi komplexní téma, takže není možné ho celé zahrnout do této práce.

Dále [7] říká, že je důležité pracovat s nejnovějšími verzemi všech komponent, a to jak na straně klienta, tak na straně serveru, včetně operačního systému, systému správy databází a všech knihoven.

Kapitola 3
Návrh

Tato kapitola je zaměřena na návrh aplikace. Její součástí je výběr technologií, návrh databázového modelu a modelu architektury.

3.1 Výběr technologií

Při výběru technologií je nutné zohlednit mnoho kritérií. Správný výběr totiž může velmi ovlivnit úspěch celé práce, jelikož zvolené technologie, ať už vhodně či nevhodně, mohou mít velký dopad na vývoj a případnou údržbu systému.

3.1.1 Programovací jazyk

Existuje mnoho vhodných kandidátů na volbu programovacího jazyka. Je třeba zohlednit, aby server obsahoval grafické rozhraní, jelikož i vzhled aplikace může usnadnit její používání. Dalším kritériem je požadavek, aby nástroj byl kompatibilní s operačním systémem Windows 10. Nejen proto jsem se rozhodl pro objektově orientovaný programovací jazyk C#. Byl vybrán z několika důvodů:

- znalost a zkušenost autora,
- široká škála knihoven a frameworků,
- kompatibilita s OS Windows 10.

Tento jazyk byl zvolen jak pro klietnskou část, tak i část serveru. Klientská část bude programována jako *konzolová aplikace* a část serveru jako *aplikace Windows Forms*.

3.1.2 Způsob uložení dat

Jako databázový systém byla vybrána relační databáze *Microsoft SQL Server*. Jedná se o rozšířený a osvědčený systém, který nabízí spolehlivost, snadnou správu

a hlavně standardizovaný jazyk SQL. Navíc Microsoft SQL Server je dobře kompatibilní s nástroji, které jsem se rozhodl použít. Jako velkou nevýhodou relačních databází je to, že vyžadují mapování objektů do tabulek, což může být časově náročné a složité, zejména u větších a komplexnějších systémů.

3.1.3 Přístup k datům

Z důvodu nevýhody relačních databází, zmíněné v předchozí podkapitole, je vhodné využít nějaký existující framework, který umožňuje mapovat výsledky SQL dotazů na objekty v jazyce C#.

3.1.3.1 Dapper

Pro implementaci přístupu k datům v databázi bylo zvažováno několik možností, jako jsou *Entity Framework*, *Dapper* nebo *NHibernate*. Nakonec jsem zvolil použití framework Dapper. Dapper byl vyvinut týmem, který mimo jiné stojí za vznikem velmi známé stránky *Stack overflow*, která funguje jako fórum převážně v IT sféře.[13, 14]

Dapper je open-source ORM framework pro .NET a .NET Core aplikace. Umožňuje vývojářům rychlý a snadný přístup k datům z databází bez nutnosti psát zdlouhavý kód. Dapper nabízí mimo jiné možnost mapovat výsledky na objekty a spouštět uložené procedury. Je k dispozici jako NuGet balíček.[14]

Dapper je snadno ovladatelný a výkonný vzhledem k jeho jednoduchosti. Na rozdíl od ostatních ORM řešení nevyžaduje žádnou složitou konfiguraci během vývoje. Jednou z jeho největších předností je rychlost, která byla i hlavním cílem při vývoji. Další výhodou je podpora parametrizovaných dotazů, kterými je systém chráněn před útoky SQL Injection (viz sekce 2.7.2). Velmi důležitou výhodou vzhledem k jeho častému využívání v programech je jeho kompatibilita s mnoha databázovými poskytovateli.[15, 14]

3.1.4 Získávání dat

Vzhledem k tomu, že program bude pracovat s informacemi o hardwaru, je třeba vybrat nástroje, které umožňují tyto informace zjistit. V tomto ohledu existuje mnoho způsobů, jak data získat. Lze použít například nástroj WMI nebo knihovnu OpenHardwareMonitor.

3.1.4.1 WMI

Pro zjištění některých informací o hardwaru byl nakonec zvolen nástroj WMI, který umožňuje přistupovat k informacím o výkonu, stavu a konfiguraci operačního systému, aplikacích a hardwarových prvcích. WMI lze také použít k získání všech potřebných dat, jako jsou například data o procesoru, pamětích, sériovém čísle

BIOSu či informace o přihlášených uživatelích. Informace o některých použitých třídách lze nalézt v [16].

Například pro zjištění operačního systému a jeho verze lze vybrat atributy *Caption* a *Version* z třídy „Win32_OperatingSystem“.[16] Tento způsob umožňuje získat mnohem více dalších informací o počítači.

3.1.4.2 Jiné způsoby

Jako programovací jazyk byl vybrán C#, který obsahuje velkou škálu knihoven, jenž jsou s operačním systémem Windows kompatibilní. Z tohoto důvodu bylo možné využít široké spektrum knihoven k získávání informací o počítači. Například lze využít třídu *NetworkInterface*, která umožňuje získat všechna potřebná data o aktivních síťových rozhraních, včetně statistik o síťovém provozu. Tato třída je součástí knihovny „System.Net.NetworkInformation“.

K získání informací o discích lze použít třídu *DriveInfo* z knihovny „System.IO“. Je možné tak jednoduše získat veškerá potřebná data včetně volné kapacity.

Dále například v třídě *Environment* jsou obsaženy informace o běhovém prostředí. Z této třídy je získán název počítače a jméno uživatele. Třída je součástí knihovny „System“.

3.2 Návrh architektury

Pro návrh systému byla zvolena třívrstvá architektura, která je tvořená prezentační, business a datovou vrstvou. Tento přístup umožňuje snadnou rozšiřovatelnost a udržovatelnost kódu. Celý systém je rozdělen na klienta a server. Výsledný model je znázorněn na obrázku 3.2.

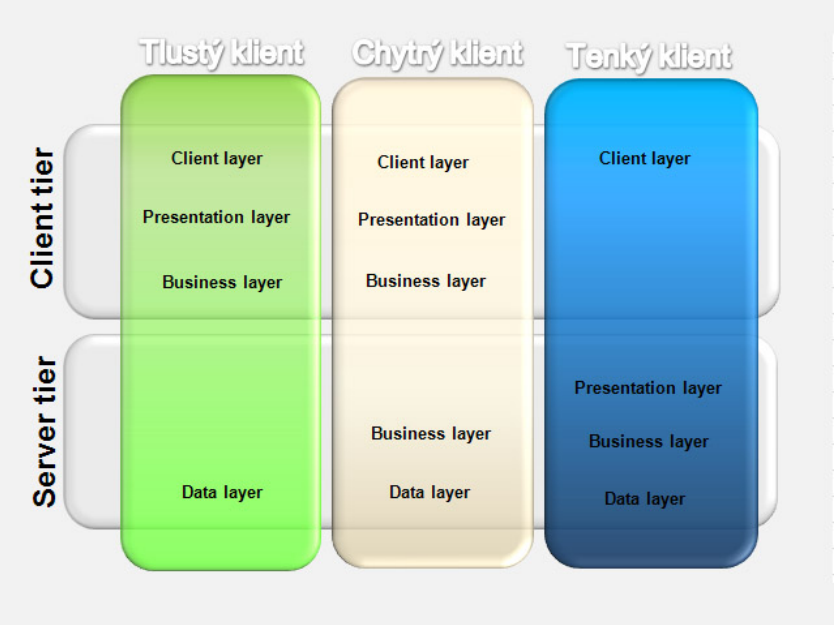
3.2.1 Klient

Pro optimální výkon a efektivitu systému je potřeba správné rozložení architektury. Podle [17] existují následující možnosti, které jsou taktéž znázorněny na obrázku 3.1.

tenký klient — označení systému, kde na straně klienta neprobíhá žádná rozhodovací logika. Nároky na instalaci jsou minimální. Nejčastějším příkladem je klient spuštěn v internetovém prohlížeči,

tlustý klient — opak tenkého klienta, bývají potřeba větší nároky na hardware a připojuje se přímo k databázovému nebo jinému serveru,

chytrý klient — kombinace výhod tenkého a tlustého klienta.



Obrázek 3.1 Rozložení výpočetních sil dle [17]

Jedna z vlastností, která je od klienta žádaná, je, aby byl program spuštěn na pozadí, jelikož pro běžného uživatele ve firmě není při jeho práci důležité, že firma inventarizaci provádí. Jeho práce by však neměla být neustále narušována pravidelným zobrazováním programu na obrazovce, čemuž se právě zabrání jeho spuštěním na pozadí. Druhou možností je nainstalování klienta jako služby, což není v této práci obsaženo, ale jedná se o jedno z možných vylepšení (viz kapitola 5).

Proto jsme se s vedoucím práce shodli, že nejlepší možností je použití tenkého klienta, který bude spuštěn na pozadí, bude zjišťovat informace pouze o konkrétní stanici a bude data odesílat na server.

3.2.2 Server

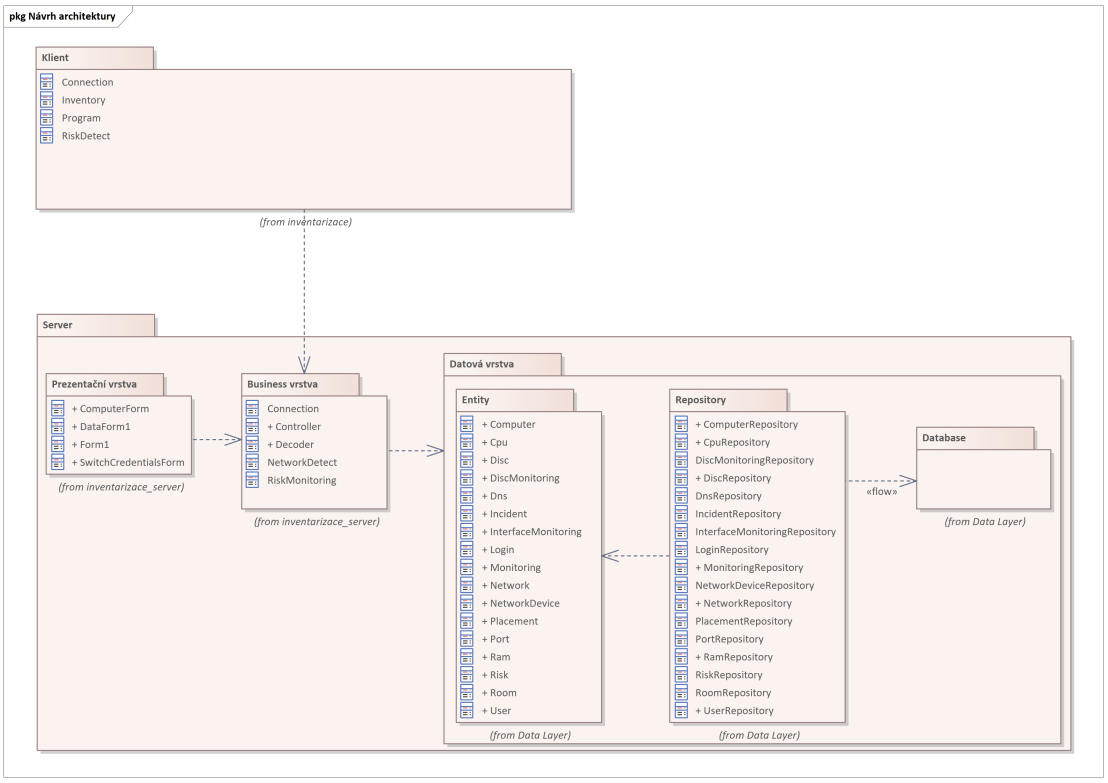
Na obrázku 3.1 je vidět, že server bude obsahovat tři vrstvy. První vrstvou je vrstva prezentační, která je zodpovědná za zpracovávání požadavků uživatele a uživatelského rozhraní. V tomto případě je server tvořen využitím Windows Forms v programovacím jazyku C# a tato vrstva bude obsahovat GUI (grafické uživatelské rozhraní).

Druhou vrstvou je business vrstva, která je odpovědná za řízení systému. Jsou v ní obsaženy třídy, které reagují na konkrétní situace a zprostředkovává komunikaci s datovou vrstvou.

Poslední vrstvou je vrstva datová, která je zodpovědná za přímou komunikaci s databází. Zprostředkovává základní CRUD operace s daty, tedy ukládání dat do databáze a jejich správu.

3.2.3 Databáze

Poslední komponentou celého systému je databáze. Na základě konzulace s vedoucím IT oddělení firmy jsme se rozhodli, že nejlepší variantou bude vytvoření lokální instance některé z moderních databázových systémů. Důvody a volba konkrétního databázového systému jsou podrobněji popsány v sekci 3.1.2. Tato volba nám umožní snadnou správu a zároveň zajistí dostatečnou škálovatelnost pro budoucí rozšíření funkcí systému.

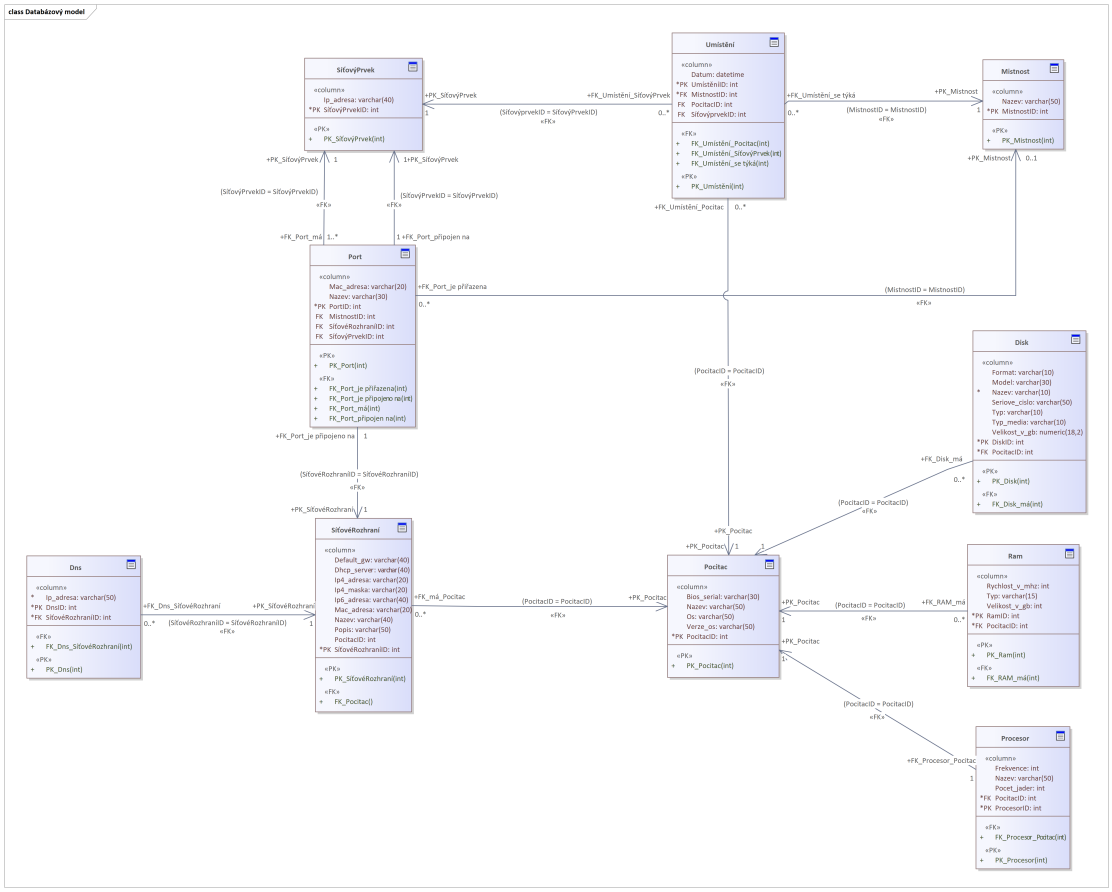


Obrázek 3.2 Model návrhu architektury

3.3 Návrh databáze

V databázovém modelu je znázorněna struktura a vztahy mezi daty, které jsou do databáze ukládány.

Při návrhu je důležité definovat entity, atributy a vztahy mezi nimi. Mnoho jich známe již z doménového modelu (sekce 2.6), nicméně je do databázového modelu třeba zakomponovat primární a cizí klíče a provést případnou dekompozici m:n vazeb. Popis tabulek je pro přehlednost, stejným způsobem jako v doménovém modelu, rozdělen do dvou částí, přestože fyzicky budou všechny tabulky uloženy v jedné databázi.



Obrázek 3.3 Databázový model inventarizace

Obrázek 3.4 Databázový model sledování rizik

Kapitola 4
Implementace

V této kapitole je popsána implementace celého systému, její rozdělení na dané vrstvy a požadavky, které jsou na počítače kladeny, aby byl systém funkční. Také je zde zadefinován průběh komunikace mezi klientem a serverem. Součástí kapitoly jsou i ukázky kódu.

4.1 Komunikace

Pro komunikaci mezi klientem a serverem byla použita třída „Socket“ z knihovny *System.Net.Sockets*, která umožňuje vytvoření spojení socketů pomocí protokolu TCP/IP. Socket lze na obou stranách použít jak pro přijímání, tak pro odesílání dat. Bezpečnost přenosu dat je zajištěná zašifrováním pomocí veřejného klíče serveru algoritmem RSA. Více je o tomto algoritmu popsáno v sekci 2.7.1.

Následuje ukázka kódu použitého během šifrování a dešifrování v jazyce C# pomocí knihovny *System.Security.Cryptography*. Šifrování probíhá metodou *Encrypt*, dešifrování metodou *Decrypt*, jejichž prvním parametrem jsou data k šifrování/dešifrování a druhým parametrem je mód zarovnání. V ukázkách 1 a 2 je použit mód *OaepSHA1*, který reprezentuje objekt představující šifrovací standard OAEP s hashovacím algoritmem SHA1.[18]

Během komunikace klient odesílá buď data o počítači, která slouží k inventarizaci, nebo data, která slouží ke sledovánní rizik.

4.1.1 Definování struktury odeslaných dat

Definování pořadí odeslaných dat je znázorněno na obrázcích 4.1, 4.2 a 4.3. Důležitý je první záznam „ID“, protože díky němu je rozlišeno, zda se jedná o data týkající se inventarizace, či data, která slouží ke sledování rizik. Po rozpoznání a odeslání dat jsou informace dále zpracovávány, což je popsáno v sekcích 2.6.1 a 2.6.2.

Data se posílají jednotlivě oddělená znakem „`\n`“. Posledním znakem, který značí konec přenosu je znak „`\0`“. Tento znak se posílá buď na konci tokenu, nebo

na úplném konci přenosu, tedy po zaslání všech informací.

```
using System.Security.Cryptography;
private readonly string publickeyPath;

private byte[] encrypt(byte[] data)
{
    RSA rsa = RSA.Create();

    //načtení obsahu souboru .pem
    string publicKey = File.ReadAllText(publickeyPath);

    //naimportování klíče
    rsa.ImportFromPem(publicKey);

    //vrácení výsledku šifrování s paddingem
    return rsa.Encrypt(data, RSAEncryptionPadding.OaepSHA1);
}
```

Výpis kódu 1 Průběh šifrování dat

```
using System.Security.Cryptography;
private readonly string privatekeyPath;

private string decrypt(byte[] buffer)
{
    RSA rsa = RSA.Create();

    //načtení obsahu souboru .xml
    string privateKey = File.ReadAllText(privatekeyPath);

     //naimportování klíče
     rsa.FromXmlString(privateKey);

    //vrácení výsledku dešifrování
    return Encoding.ASCII.GetString(
        rsa.Decrypt(buffer, RSAEncryptionPadding.OaepSHA1))
}
```

Výpis kódu 2 Průběh dešifrování dat

4.1.1.1 Data inventarizace

Inventarizační data jsou označena pomocí ID rovnající se 1. Po rozpoznání ID následuje přenos několika základních informací o počítači a následně informací o síťových rozhraních. Nejdříve je odeslán počet síťových rozhraní, následuje počet informací a pak samotné informace. Stejným způsobem jsou odeslána data o discích, pamětích a uživatelích.

Data o procesoru se zasílají před informacemi o uživatelích a jejich struktura je mírně odlišná. Vždy je odeslán počet informací a následují sledovaná data. Důvodem odlišnosti je, že počítač obsahuje pouze jeden procesor.

Celá struktura je podrobněji znázorněna na obrázku 4.1.

ID	Název	Sériové číslo BIOSu	Operační systém	Verze OS	Síťová rozhraní	Disky	Paměti	Název CPU	Frekvence CPU	Počet jader CPU

Obrázek 4.1 Struktura odeslaných dat inventarizace

4.1.1.2 Data sledování rizik

Data ke sledování rizik jsou označena ID rovnající se 2. Po rozpoznání ID následuje odeslání názvu počítače a teploty procesoru na server. Dále jsou odesílána data o discích a síťových rozhraních. To probíhá stejným způsobem jako odesílání dat o discích a síťových rozhraních během inventarizace, liší se ale jejich typ.

Následuje odeslání informace o posledním přihlášení aktuálního uživatele. Nejprve je posláno jméno uživatele a poté datum a čas jeho posledního přihlášení.

Celá struktura je podrobněji znázorněna na obrázku 4.2 a podrobnější struktura, jak jsou odeslána data síťových rozhraní je znázerněno na obrázku 4.3.

ID	Název počítače	Teplota CPU	Síťová rozhraní	Disky	Uživatel	Datum posledního přihlášení

Obrázek 4.2 Struktura odeslaných dat ke sledování rizik

Počet síťových rozhraní	Počet informací	Název rozhraní	Bytů přijato	Bytů odesláno	Defaultní brána	DHCP server	DNS server 1	DNS server 2	...	Počet informací	Název rozhraní	...

Obrázek 4.3 Podrobnější struktura odeslaných dat síťových rozhraní

4.2 Klient

Klient je prozatím implementován jako konzolová aplikace. Nicméně jak už bylo zmíněno, je zde možnost vylepšení programu tím, že bude naistalován jako služba (viz kapitola 5). Klient pravidelně zasílá informace o hardwaru počítače na server.

Při jeho spuštění si program nejdříve načte z konfiguračního souboru všechny potřebné informace jako jsou IP adresa, port a veřejný klíč serveru, který se později použije pro šifrování komunikace.

Poté vytvoří novou instanci třídy *Connection*. Tato třída se stará o komunikaci se serverem. Má pouze 2 metody a to *connectTo* a *sendData*. Jejími paramentry jsou IP adresa a port, na kterém server běží. V případě, že se nepodaří na server připojit, klient se o to pokusí periodicky znovu.

Dále vytvoří novou instanci třídy *Inventory*. Konstruktoru této třídy se jako parametr předá objekt třídy *Connection*, který se vytvořil dříve. Tato třída má pouze jednu veřejnou metodu a to *inventoryComputer*. Ta se kompletně stará o získání a následně poslání všech sledovaných údajů o počítači. Pro poslání dat se použije objekt třídy *Connection*, který je předán jako parametr konstruktoru.

Každý počítač se na začátku každé komunikace autentizuje přiřazeným unikátním tokenem. Tento token je pro každý počítač vygenerovaný serverem během první komunikace. Klient pošle serveru neplatný token a server mu vygeneruje token platný. Poté následuje poslání všech sledovaných dat jako během každé další komunikace.

V části sledování rizik se vytvoří instance třídy *RiskDetect*, která je velmi podobná třídě *Inventory*. Také přijímá jako parametr objekt třídy *Connection* a má pouze jednu veřejnou metodu *detect*, která se stará o získání a zaslání všech potřebných informací k detekci rizik. Taktéž pro poslání dat používá objekt třídy *Connection*.

4.3 Server

Server je o mnoho komplexnější část než část klienta. Navíc, jelikož se na server budou připojovat desítky možná až stovky počítačů, je navržen tak, aby se při připojení každého klienta vytvořilo jedno nové vlákno, které zpracuje přijatá data, a když vše proběhne bez chyby, tak se data uloží do databáze.

4.3.1 Prezentační vrstva

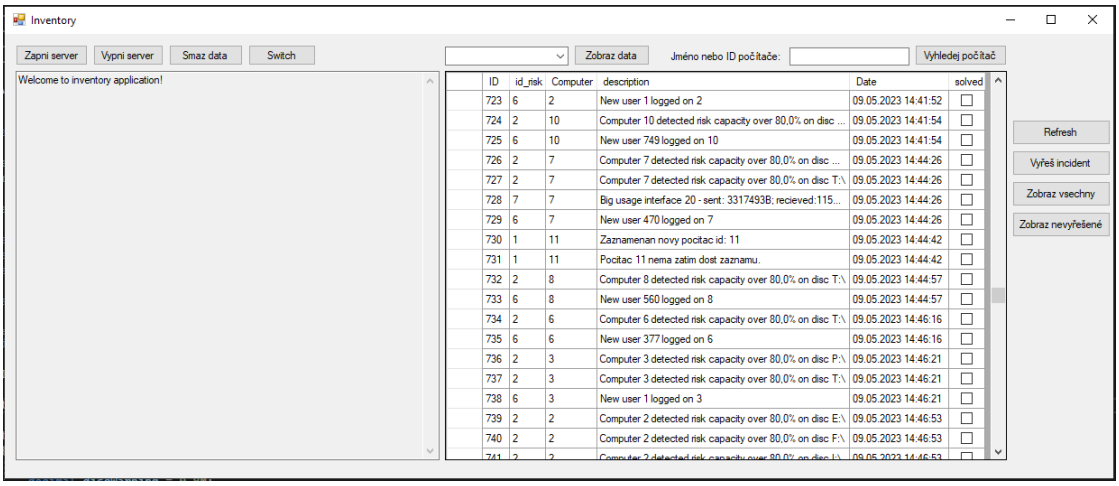
Prezentační vrstva obsahuje rozhraní, se kterým bude pracovník IT oddělení interagovat. Je implementována jako .NET Windows Forms aplikace.

Po spuštění se zobrazí hlavní okno viz obrázek 4.4, jehož nejdůležitější částí je sledování rizik. Zobrazené riziko může pracovník označit a změnit jeho stav na vyřešené. Také si může vyfiltrovat nevyřešené incidenty.

Pracovník může taktéž pomocí vyhledávacího pole zobrazit hledaný počítač v novém okně, kde jsou zobrazeny jeho základní údaje. Počítač lze vyhledat pomocí jeho ID či názvu. Okno se základními údaji je zobrazeno na obrázku v příloze D.1.

Kromě tohoto si pracovník může zobrazit tabulky z databáze. Ze seznamu si vybere tabulku a po stisknutí tlačítka „Zobrazit data“ se zobrazí nový formulář,

kde si může prohlédnout aktuální obsah příslušné tabulky databáze.



ID	id_risk	Computer	description	Date	solved
723	6	2	New user 1 logged on 2	09.05.2023 14:41:52	
724	2	10	Computer 10 detected risk capacity over 80,0% on disc ...	09.05.2023 14:41:54	
725	6	10	New user 749 logged on 10	09.05.2023 14:41:54	
726	2	7	Computer 7 detected risk capacity over 80,0% on disc ...	09.05.2023 14:44:26	
727	2	7	Computer 7 detected risk capacity over 80,0% on disc T:\	09.05.2023 14:44:26	
728	7	7	Big usage interface 20 - sent: 3317493B; recieved:115...	09.05.2023 14:44:26	
729	6	7	New user 470 logged on 7	09.05.2023 14:44:26	
730	1	11	Zaznamenan novy pocitac id: 11	09.05.2023 14:44:42	
731	1	11	Pocitac 11 nema zatim dost zaznamu.	09.05.2023 14:44:42	
732	2	8	Computer 8 detected risk capacity over 80,0% on disc T:\	09.05.2023 14:44:57	
733	6	8	New user 560 logged on 8	09.05.2023 14:44:57	
734	2	6	Computer 6 detected risk capacity over 80,0% on disc T:\	09.05.2023 14:46:16	
735	6	6	New user 377 logged on 6	09.05.2023 14:46:16	
736	2	3	Computer 3 detected risk capacity over 80,0% on disc P:\	09.05.2023 14:46:21	
737	2	3	Computer 3 detected risk capacity over 80,0% on disc T:\	09.05.2023 14:46:21	
738	6	3	New user 1 logged on 3	09.05.2023 14:46:21	
739	2	2	Computer 2 detected risk capacity over 80,0% on disc E:\	09.05.2023 14:46:53	
740	2	2	Computer 2 detected risk capacity over 80,0% on disc F:\	09.05.2023 14:46:53	

Obrázek 4.4 Úvodní okno

4.3.2 Business vrstva

Úkolem business vrstvy je zpracování dat poslaných od klientů a jejich uložení. Obsahuje veškerou logiku serveru včetně části sledování rizik. Poslaná data se rozdělují na inventarizační data a data sloužící ke sledování rizik.

Při první komunikaci s klientem je náhodně vygenerován token. Tento token se během dalších navázaných spojení kontroluje a v případě přijmutí jiného tokenu je spojení ukončeno.

Inventarizační data se všechna uloží do databáze pomocí datové vrstvy. Z informací ke sledování rizik se uloží jen ty, u kterých to má smysl a opodstatnění. Shodli jsme se na nich s vedoucím práce a patří mezi ně například teplota procesoru, jelikož chceme sledovat vývoj teploty v čase. Příkladem informací, které se neukládají, jsou aktuální informace o síťových rozhraních. U těchto dat se detekuje pouze případná změna a upozorní se na ni. Podrobněji je ukládání dat popsáno v sekci 2.6.2.

Další součástí business vrstvy je sledování rizik. Po každém přijmutí dat sloužících ke sledování rizik z konkrétního počítače dojde k vyvolání jejich analýzy. Načtou se poslední 2 záznamy a porovnají se hodnoty teploty procesoru a volná kapacita disků. V případě překročení limitů definovaných v konfiguračním souboru je vytvořen nový incident. Také jsou porovnána data o využití jednotlivých síťových rozhraních a v případě nadměrné komunikace je zaznamenán nový incident. U rozhraních je kontrolována případná změna některého z údajů o výchozí bráně, DHCP serveru a DNS serveru. Posledními monitorivanými informacemi jsou přihlášení na konkrétním počítači. V případě, že se na počítač přihlásí někdo, kdo se ještě nikdy nepřihlásil, je nahlášen nový incident.

Také se pomocí business vrstvy provádí inventarizace switchů. Uživatel je po stisknutí tlačítka „Switch“ dotázán na IP adresu switche a jméno a heslo pro přihlášení. Po potvrzení údajů se program pokusí protokolem SSH navázat spojení s daným switchem. V případě úspěšného spojení dojde ke zjištění MAC adres připojených ke konkrétním portům switche pomocí příkazu „show mac address“. Výsledek se poté zpracuje a data se uloží do databáze.

4.3.3 Datová vrstva

Datová vrstva je zodpovědná za komunikaci s databází. V mé práci byl použit lokální Microsoft SQL Server a pro komunikaci s databází ORM framework Dapper.

Pro každou tabulku v databázi existuje entita, tedy třída ve složce „Entity“, reprezentující daný objekt. Dále pro každou entitu existuje třída odpovídající Dapper repository třídě, která zodpovídá za komunikaci s databází pomocí SQL dotazů. Tato třída umožňuje základní CRUD operace s tabulkou a další jiné SQL dotazy potřebné k fungovaní aplikace. Tyto třídy jsou použity v business vrstvě.

Všechny tyto třídy jsou potomkem abstraktní třídy *Repository*, která je pro ně vytvořena jako šablona. Obsahuje předpis pro všechny CRUD operace, které lze obecně předepsat pro všechny tabulky v databázi.

Tedy například pro počítače existuje v databázi tabulka *Computer*. Této tabulce odpovídá třída *Computer* a pro práci s tabulkou slouží třída *ComputerRepository*, která je potomkem třídy *Repository*. Ukázku kódu této třídy si lze prohlédnout v příloze B. Ukázka kódu s tabukou je uvedena v příloze A.

4.3.4 Průběh komunikace klienta

Připojení klienta na server je možné po jeho zapnutí. Po vytvoření spojení je klientem během první komunikace poslán nevalidní token, který je serverem zachycen. Na to server reaguje vygenerováním nového tokenu, který je zaslán klientovi.

Klientem následně provede inventarizaci počítače a postupně posílá zašifrovaná data oddělená znakem „`\n`“. Server čeká na znak „`\0`“, který je indikátorem konce přenosu. Podle prvního znaku je následně rozhodnuto, že se jedná o inventarizační data. Přijmutá data jsou postupně rozdělena do objektů určitých tříd a pokud vše proběhne v pořádku, tak jsou uložena do databáze.

Stejný postup komunikace je použit i v rámci sledování rizik. Nejprve se program prokáže přiděleným tokenem, poté získá všechny informace a stejným způsobem je odešle na server. Server rozpozná, že se jedná o data sloužící ke sledování rizik a ta jsou rozdělena do příslušných tříd. Pokud vše proběhne v pořádku, jsou uložena do databáze.

Ve chvíli, kdy jsou data ke sledování rizik uložena, dojde ke kontrole rizik pro daný počítač. Při detekci rizika je uložen nový incident, který popisuje a upozorňuje na daný identifikovaný problém tak, aby na něj mohl pracovník IT oddělení reagovat.

4.4 Požadavky na systém

Pro správné fungování části klienta je nutné mít na počítači nainstalovaný operační systém Windows 10 a novější. Také je potřeba mít nainstalovaný .NET Core verze 6.0 a vyšší.

Pro správné fungování části serveru je zapotřebí operační systém Windows 10 a novější, případně OS Windows Server 2019 a novější. Také je třeba mít nainstalovaný .NET Framework verze 4.8 a vyšší. Co se týče databázového systému je potřeba podpora souboru s Microsoft SQL Serverem (přípona .mdf) verze 904 a vyšší. V případě problémů lze nahradit vlastním souborem starší verze a využít *create script* a *insert script*, které inicializují tabulky databáze společně s incicializačními daty. Tento způsob ale nezaručuje plnou funkčnost.

Kapitola 5
Možná vylepšení

Tato kapitola obsahuje popis možných vylepšení a rozšíření aplikace.

Jelikož téma inventarizace je velmi komplexní, je v praktické části, po dohodě s vedoucím práce, realizovaná jen vybraná část.

Možné menší úpravy pro vylepšení systému:

1. Část serveru by bylo nepochybně možné pozvednout o několik úrovní výše pomocí grafického rozhraní. Jeho vzhled by mohl být o moho lepší, kdyby bylo možné klikat na různé ikony, a tím si zobrazovat sledované informace. Rozhodně by se dala také zlepšit intuitivnost rozhraní.
2. Část klienta by bylo možné nainstalovat do OS Windows jako službu. To by umožňovalo mít tento program spuštěn neustále aniž by nějakým způsobem omezoval uživatele počítače. Také by to umožňovalo zjišťovat informaci o tom, zda je uživatel přihlášen. A i přesto, že by uživatel přihlášen nebyl, by bylo možné inventarizaci provádět.
3. Systém byl navržen tak, aby mohl být spuštěn na OS Windows 10 a novějších. V mnoha firmách se používají i jiné operační systémy, například Linux a MacOS, a proto je žádoucí v budoucnu rozšířit systém tak, aby byl multiplatformní.
4. Dále by bylo v případě detekovaného rizika možné pracovníkovi zaslat notifikaci. S tím souvisí i možné rozšíření systému o webové rozhraní. Díky tomu by systém byl pracovníkovi přístupný odkudkoliv i v případě, že není fyzicky v práci.

5.1 Verze klientů

Během testování systému jsem zjistil, že například pro přístup k informaci o teplotě procesoru je třeba oprávnění správce. Jelikož jsou ale prostředí správce a běžného

uživatele oddělená, tak při spuštění jako správce se nedetekují síťové disky.

Tomuto lze zabránit nepoužíváním mapovaných disků (ve formátu C:\cesta). Místo nich je vhodné použít konvence UNC (\\server\share\cesta). Druhým způsobem je změnění registru *EnableLinkedConnections* na hodnotu 0x1.

Dalším řešením je vytvoření 2 specializovaných klientů. První by byl určen pro zjišťování systémových údajů a byl by nainstalován jako služba s vyššími právy, například *LocalSystem*. Druhý by byl použit pro uživatelské informace, tedy například právě pro detekci disků. Na tomto řešení jsme se shodli s vedoucím práce.

5.2 Reporty

V budoucnu by bylo možné systém rozšířit o tvorbu reportů či auditů. To by umožňovalo například tvořit statistiku všech počítačů ve firmě a na základě těchto informací vytvořit grafy a tabulky.

5.3 Autentizace

Část serveru by neměla být přístupná nikomu jinému, než danému pracovníkovi IT oddělení. K tomu by mohlo sloužit rozšíření o autentizaci, ve kterém by bylo nutné se před použitím systému přihlásit. Mohlo by zde být tlačítko na odhlášení a z důvodu bezpečnosti by část serveru uživatele odhlásila i po delší neaktivitě.

5.4 Hardware

Systém byl navržen tak, aby bylo možné ho rozšířit o získávání i dalších informací o hardwaru. Bylo by možné například získávat informace o základní desce, monitoru, reproduktorech, mikrofonu či grafických kartách.

Dále je zde možnost rozšíření o zaznamenávání konce záruky u jednotlivých strojů a komponent.

Kromě toho by bylo reálné získávat informace o tiskárnách, které jsou na počítač připojené a systém by mohl zobrazovat případné incidenty jako jsou docházející toner, papír nebo jiné.

5.5 Software

Velkým možným rozšířením systému je rozšíření nejen o sledování hardwaru, ale i softwaru. Podobně, jako to dělá například program *AIDA64 Business* popsaný v sekci 2.3, by program detekoval všechny programy, plánované úlohy či programy, které se spouští po spuštení počítače.

Součástí tohoto rozšíření by mohla být inventarizace různých ovladačů a případně kontrolovat jejich aktuálnosti se současným hlášením incidentů v případě, kdy je ovladač zastaralý.

Také by bylo možné inventarizovat licence, kontrolovat jejich stav a případně hlásit, že se blíží konec jejich platnosti. S tím souvisí i produktové klíče, na které bývají licence často vázané. To by umožňovalo zobrazovat produktové klíče s dalšími podrobnostmi o licencích, jako je jejich počet, datum vypršení platnosti atd.[19]

5.6 Operační systém

Dalším okruhem by mohla být určitá kontrola operačního systému. Pracovník IT oddělení by pomocí nástroje mohl vzdáleně provádět aktualizace systému, nebo dokonce restartovat či vypnout počítač. Případně by u každého počítače mohlo být tlačítko, které na OS Windows spustí program „Připojení ke vzdálené ploše“ a vyplní potřebné údaje. Pracovníkovi by pak jen stačilo zadat heslo a ihned by byl ke vzdálené ploše připojen.

Z důvodu bezpečnosti by musela být vyřešena autentizace, aby například nenastala situace, ve které by kdokoliv mohl vypínat počítače napříč celou firmou.

Kapitola 6
Závěr

Cílem této bakalářské práce bylo navrhnout a realizovat nástroj, který usnadní inventarizaci hardwaru v rámci IT oddělení.

V první části byla zanalyzována existující řešení a bylo zdůvodněno, proč je implementováno něco již existujícího. Následně byly definovány požadavky, které byly na systém kladeny a byl vytvořen doménový model, který čtenáře lépe seznámí s entitami a vazbami mezi nimi. Poté se práce věnovala bezpečnosti aplikace, kde bylo rozebráno šifrování přenosu dat. K tomu byl použit algoritmus RSA. Také byla popsána problematika útoku *SQL Injection*, která byla nastíněna několika příklady.

Další část práce se věnovala návrhu systému. Byly zvoleny vhodné technologie, navržen databázový model a model architektury.

Následující část se zaobírala popisem implementace celého systému. Byla definována struktura odeslaných dat, popsána část klienta, jednotlivé vrstvy serveru a požadavky, které jsou pro spuštění na počítače kladeny. Dále byl popsán průběh komunikace mezi klientem a serverem.

Poslední část se věnovala možnému vylepšení celého nástroje. Na tuto kapitolu byl kladen důraz z důvodu komplexnosti tématu.

Výsledný nástroj měl sbírat data na lokálních stanicích, serverech a síťových prvcích. Data měla být při přenosu šifrována a poté se měla předzpracovat, uložit do databáze a vyhodnotit. Hodnotit se měla i případná funkcionální a bezpečnostní rizika.

Všechny body zadání byly splněny a výsledkem je systém na principu klient-server napsán v programovacím jazyce C#. Celý systém je určen pro OS Windows 10 a novější. Nástroj byl navržen tak, aby bylo možné ho relativně snadno rozšířit o další funkcionality.

Příloha A

Příklad návaznosti entit s tabukou v databázi

	Name	Data Type	Allow Nulls
🔑	Id_computer	int	☐
	compName	nvarchar(50)	☑
	biosSerial	nvarchar(30)	☑
	OS	nvarchar(50)	☑
	OSVersion	nvarchar(50)	☑

Obrázek A.1 Předpis tabulky v databázi

```
public class Computer
{
    public int id_computer { get; set; }
    public string compName { get; set; }
    public string biosSerial { get; set; }
    public string Os { get; set; }
    public string OsVersion { get; set; }

    public Computer(int Id_computer, string compName,
            string biosSerial, string Os, string OsVersion)
    {
        this.id_computer = Id_computer;
        this.compName = compName;
        this.biosSerial = biosSerial;
        this.Os = Os;
        this.OsVersion = OsVersion;
    }
}
```

Výpis kódu 3 Třída *Computer*

```
public class ComputerRepository : Repository<Computer>
{
    public ComputerRepository(string constr) : base(constr) { }

    public override void Insert(Computer item)
    {
        IDbConnection connection = new SqlConnection(constr);
        connection.Open();
        string query = $"INSERT INTO {tableName} " +
            $"VALUES (@id_computer, @name, " +
            $"@biosSerial, @Os, @OsVersion)";
        connection.Execute(query, item);
        connection.Close();
    }
    public override void Update(Computer item)
    {
        IDbConnection connection = new SqlConnection(constr);
        connection.Open();
        string query = $"UPDATE {tableName} SET " +
            $"biosSerial=@biosSerial, OS=@Os, " +
            $"OSVersion=@OsVersion " +
            $"WHERE Id_computer=@id_computer";
        connection.Execute(query, item);
        connection.Close();
    }
    public Computer GetByName(string name)
    {
        IDbConnection connection = new SqlConnection(constr);
        connection.Open();
        string query = $"SELECT * FROM {tableName}" +
            $" WHERE compName=@name";
        Computer result = connection.
            QuerySingleOrDefault<Computer>(query, new { name });
        connection.Close();
        return result;
    }
}
```

Výpis kódu 4 Třída *ComputerRepository*

Příloha B
Třída Repository

```
public abstract class Repository<T> where T : class
{
    protected readonly string constr;
    protected string tableName = typeof(T).Name;
    protected string pk = "id_" + typeof(T).Name;

    protected Repository(string constr)
    {
        this.constr = constr;
    }

    public virtual int NextId()
    {
        IDbConnection connection = new SqlConnection(constr);
        connection.Open();
        string query = $"SELECT MAX({pk}) FROM {tableName}";
        int lastId = connection.QueryFirstOrDefault<int>(query);
        connection.Close();
        return lastId + 1;
    }
```

Výpis kódu 5 Abstraktní třída *Repository* 1/2

```
    public virtual T GetById(int id)
    {
        IDbConnection connection = new SqlConnection(constr);
        connection.Open();
        string query = $"SELECT * FROM {tableName} WHERE {pk} = @Id";
        T result = connection.QuerySingleOrDefault<T>(query, new { Id = id });
        connection.Close();
        return result;
    }

    public bool Contains(int id)
    {
        return GetById(id) != null;
    }

    public virtual IEnumerable<T> GetAll()
    {
        IDbConnection connection = new SqlConnection(constr);
        connection.Open();
        string query = $"SELECT * FROM {tableName}";
        IEnumerable<T> result = connection.Query<T>(query);
        connection.Close();
        return result;
    }

    public virtual void Delete(int id)
    {
        IDbConnection connection = new SqlConnection(constr);
        connection.Open();
        string query = $"DELETE FROM {tableName} WHERE {pk} = @Id";
        connection.Execute(query, new { Id = id });
        connection.Close();
    }

    public abstract void Insert(T item);

    public abstract void Update(T item);
}
```

Výpis kódu 6 Abstraktní třída *Repository* 2/2

Příloha C

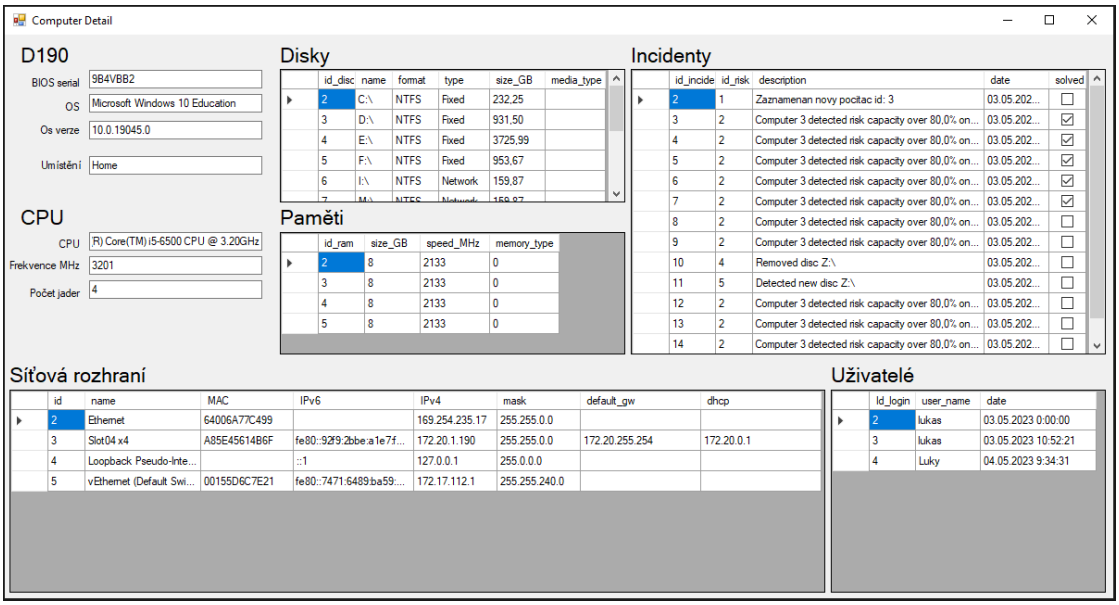
Ukázka kódu úvodní komunikace

```
Connection connection = new Connection(ipAddress, port, publicKeyPath);
Inventory inventory1 = new Inventory(connection);
try {
    connection.connectTo();

    //poslání tokenu
    connection.sendData(token + "\0");
    //čekání na přiřazený token
    token = connection.recieve();
    //provedení inventarizace počítače
    inventory1.inventoryComputer();
    connection.sendEnd();

    //čekání na odpověď; pokud je špatná, program vypíše error
    string tmp = connection.recieve();
    if (tmp != "SERVER OK") {
        Console.WriteLine("Error");
    }
    connection.closeConnection();
}
catch {
    Console.WriteLine("Error");
    Console.ReadLine();
    return;
}
```

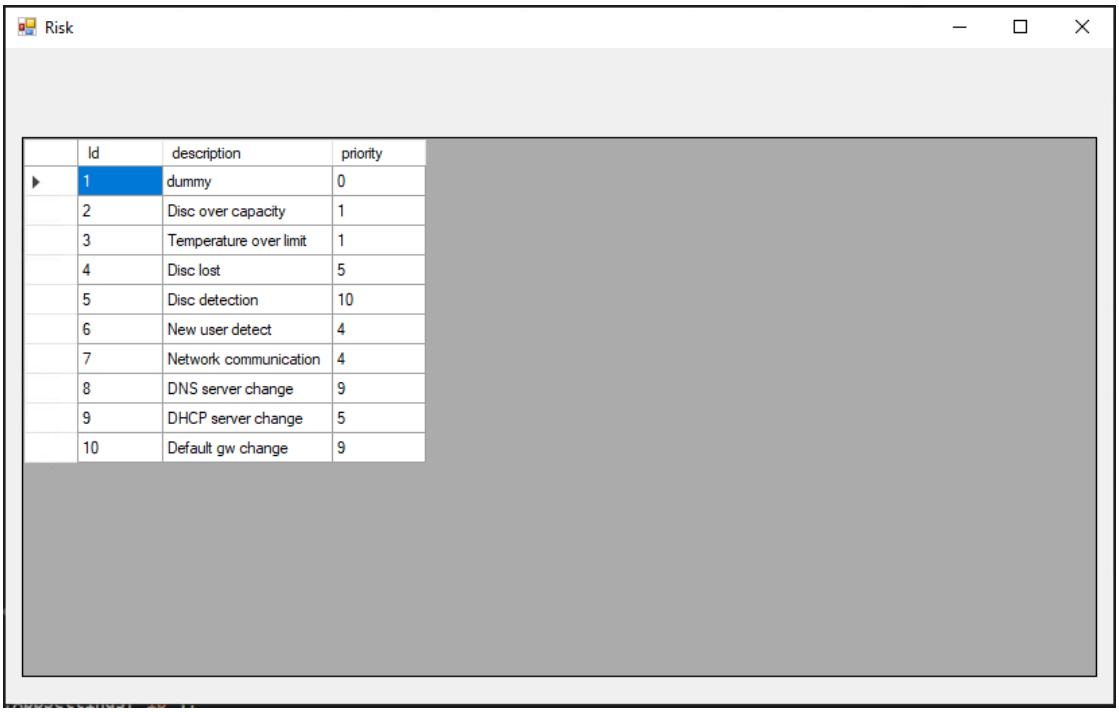
Výpis kódu 7 Průběh úvodní komunikace

Příloha D

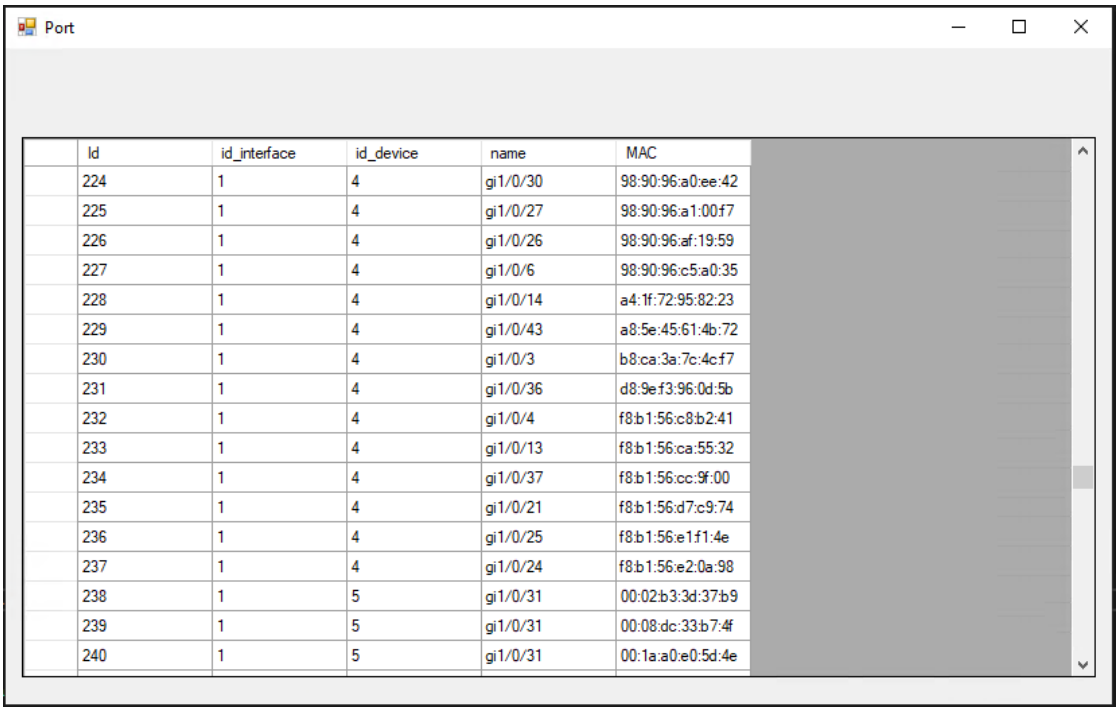
Snímky výsledné aplikace



Obrázek D.1 Základní výpis informací o počítači



Obrázek D.2 Výpis všech sledovaných rizik



Obrázek D.3 Výpis portů s připojenými MAC adresami

Bibliografie

1. ČESKO. *Zákon č. 563/1991 Sb., o účetnictví* [online]. Zákony pro lidi.cz, 2022 [cit. 2023-04-27]. Dostupné z: `https://www.zakonyprolidi.cz/cs/1991-563`.
2. BEZPALEC, Pavel. *Nové trendy v elektronických komunikacích: Bezpečnost sítí* [online]. [B.r.]. [cit. 2023-04-22]. Dostupné z: `https://publi.cz/books/223/Cover.html`.
3. ZLATANOV, Nikola. Computer Security and Mobile Security Challenges. In: 2015, s. 5.
4. KEARY, Tim. *Best computer inventory management software of 2023* [online]. 2023. [cit. 2023-05-01]. Dostupné z: `https://www.comparitech.com/net-admin/computer-inventory-management-software/`.
5. LTD., FinalWire. *About FinalWire* [online]. [B.r.]. [cit. 2023-05-03]. Dostupné z: `https://www.aida64.com/about-finalwire`.
6. LTD., FinalWire. *AIDA64 User Manual* [online]. [B.r.]. [cit. 2023-05-07]. Dostupné z: `https://aida64.co.uk/user-manual/aida64-user-manual`.
7. *OWASP Top 10* [online]. 2021. [cit. 2023-04-25]. Dostupné z: `https://owasp.org/Top10/`.
8. PAVANI, K.; SRIRAMYA, P. Enhancing Public Key Cryptography using RSA, RSA-CRT and N-Prime RSA with Multiple Keys. In: *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*. 2021, s. 1–6. Dostupné z DOI: `10.1109/ICICV50876.2021.9388621`.
9. RIETTA, Frank S. Application Layer Intrusion Detection for SQL Injection. In: *Proceedings of the 44th Annual Southeast Regional Conference*. Melbourne, Florida: Association for Computing Machinery, 2006, s. 531–536. ACM-SE 44. ISBN 1595933158. Dostupné z DOI: `10.1145/1185448.1185564`.

10. MAVROMOUSTAKOS, Stephanos; PATEL, Aakash; CHAUDHARY, Kinjal; CHOKSHI, Parth; PATEL, Shaili. Causes and Prevention of SQL Injection Attacks in Web Applications. In: *Proceedings of the 4th International Conference on Information and Network Security*. Kuala Lumpur, Malaysia: Association for Computing Machinery, 2016, s. 55–59. ICINS '16. ISBN 9781450347969. Dostupné z DOI: 10.1145/3026724.3026742.

11. DIZDAR, Admir. *Best computer inventory management software of 2023* [online]. 2022. [cit. 2023-05-08]. Dostupné z: https://brightsec.com/blog/sql-injection-attack/.

12. ČESKO. *Vyhláška č. 82/2018 Sb., o bezpečnostních opatřeních, kybernetických bezpečnostních incidentech, reaktivních opatřeních, náležitostech podání v oblasti kybernetické bezpečnosti a likvidaci dat (vyhláška o kybernetické bezpečnosti)* [online]. 2018. [cit. 2023-05-05]. Dostupné z: https://www.govcert.cz/download/kii-vis/NovaVKB/VKB_82-2018sb.pdf.

13. OVERFLOW, Stack. *Questions tagged [dapper]* [online]. [B.r.]. [cit. 2023-05-05]. Dostupné z: https://stackoverflow.com/questions/tagged/dapper.

14. PROJECTS, ZZZ. *Learn Dapper* [online]. 2023. [cit. 2023-05-05]. Dostupné z: https://www.learndapper.com/.

15. *Dapper* [online]. 2023. Ver. 2.0.123 [cit. 2023-04-20]. Dostupné z: https://github.com/DapperLib/Dapper.

16. MICROSOFT. *Computer System Hardware Classes* [online]. 2021. [cit. 2023-05-10]. Dostupné z: https://learn.microsoft.com/en-us/windows/win32/cimwin32prov/computer-system-hardware-classes.

17. ČERMÁK, Miroslav. *Vícevrstvá architektura – tenký, tlustý a chytrý klient* [online]. 2012. [cit. 2023-05-04]. ISSN 2694-9830. Dostupné z: https://www.cleverandsmart.cz/vicevrstva-architektura-tenky-tlusty-a-chytry-klient/.

18. MICROSOFT. *RSAEncryptionPadding.OaepSHA1 Class* [online]. 2021. [cit. 2023-05-06]. Dostupné z: https://learn.microsoft.com/en-us/dotnet/api/system.security.cryptography.rsaencryptionpadding.oaepsha1?view=net-8.0.

19. *IT Asset Management (ITAM) Software* [online]. [B.r.]. [cit. 2023-05-09]. Dostupné z: https://www.manageengine.com/products/desktop-central/it-asset-management.html.

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu média
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF