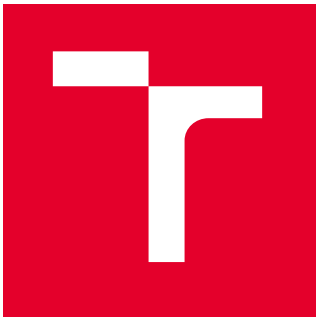


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

MULTIPLATFORMNÍ APLIKACE PRO HROMADNOU SPRÁVU ZAŘÍZENÍ MIKROTIK ROUTEROS

MULTI-PLATFORM APPLICATION FOR BULK MANAGEMENT OF MIKROTIK ROUTEROS DEVICES

SEMESTRÁLNÍ PRÁCE

SEMESTRAL THESIS

AUTOR PRÁCE — **Jiří Kozárek**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Ondřej Krajsa, Ph.D.**

SUPERVISOR

BRNO 2025

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Semestrální práce

bakalářský studijní program **Telekomunikační a informační systémy**

Ústav telekomunikací

Student: Jiří Kozárek ***ID:*** 256766

Ročník: 3 ***Akademický rok:*** 2025/26

NÁZEV TÉMATU:

Multiplatformní aplikace pro hromadnou správu zařízení MikroTik RouterOS

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte a implementujte multiplatformní aplikaci (Windows, Linux) v Pythonu pro správu zařízení s RouterOS prostřednictvím oficiálního API (šifrované připojení TLS na portu 8729). Aplikace musí umožnit kompletní správu jednotlivých zařízení, bezpečné uložení přístupových údajů, jednotlivé i hromadné provádění konfiguračních změn (např. přes šablony),zálohu a obnovu konfigurace a jednoduchou správu logů. Výstupem práce bude plně funkční aplikace, zdrojové kódy a uživatelská a instalační příručka.
V rámci semestrální práce bude navržena architektura, přidávání zařízení, zabezpečení přístupových údajů a otestována funkčnost API.

DOPORUČENÁ LITERATURA:

Podle pokynů vedoucího práce.

Termín zadání: 24.9.2025 ***Termín odevzdání:*** 10.12.2025

Vedoucí práce: Ing. Ondřej Krajsa, Ph.D.

prof. Ing. Jiří Mišurec, CSc.
předseda rady studijního programu

ABSTRAKT

Semestrální práce má za cíl vytvořit architekturu budoucí aplikace, otestovat zabezpečené spojení se zařízením s běžícím operačním systémem MikroTik RouterOS pomocí api na portu 8729 a navrhnout a implementovat bezpečné ukládání hesel nutných pro přístup k zařízením. Pro účely této práce byla vytvořena jednoduchá aplikace s grafickým uživatelským rozhraním, jež je veřejně dostupná v repozitáři na platformě GitHub.

KLÍČOVÁ SLOVA

Python, MikroTik, RouterOS, konfigurace

ABSTRACT

Tho goal of this semestral thesis is design architecture of future app, test secure api connection to device that runs MikroTik RouterOS at port 8729 and design and implement secure storage of passwords required for device access. For the purpose of this thesis simple grafical application was created and is available publicly on GitHub.

KEYWORDS

Python, MikroTik, RouterOS, configuration

KOZÁREK, Jiří. *Multiplatformní aplikace pro hromadnou správu zařízení MikroTik RouterOS*. Semestrální práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2025. Vedoucí práce: Ing. Ondřej Krajsa, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Jiří Kozárek
VUT ID autora:	256766
Typ práce:	Semestrální práce
Akademický rok:	2025/26
Téma závěrečné práce:	Multiplatformní aplikace pro hromadnou správu zařízení MikroTik RouterOS

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce. V souvislosti s vytvořením závěrečné práce jsem neporušil zásady a doporučení VUT k využívání generativní umělé inteligence.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Ondřeji Krajsovi, Ph.D. za vedení, konzultace a podnětné návrhy k práci. Dále bych také chtěl poděkovat panu Bc. Jakubu Raimrovi za rady při psaní této práce a panu Martinu Smejkalovi za věcné rady v oblasti bezpečnosti a zabezpečení dat.

Obsah

Seznam obrázků

Úvod

Cílem semestralní práce je vytvořit prototyp funkční multiplatformní aplikace pro správu zařízení Mikrotik s využitím šifrované verze api systému RouterOS. Aplikace v sobě zahrnuje implementaci nakládání s přístupovými údaji včetně šifrování a test spojení s vybraným zařízením pomocí šifrované api komunikace na portu 8729.
Pro komunikaci je využit upravený ukázkový skript [12]. Funkčnost spojení s api je zřejmá při přidávání zařízení, kdy při ukládání dochází k navázání komunikace a uložení systémové identity zařízení za předpokladu, že je zařízení dostupné a má povolený přístup přes api-ssl. Zabezpečení hesel je realizováno pomocí tzv. „master password“, který slouží jako klíč pro zašifrování hesel pomocí AES-256-CBC. Heslo v šifrované formě je uloženo do databáze společně s inicializačním vektorem. Je tedy zaručeno, že heslo dokáže získat pouze uživatel, který „master password“ zná.
V první kapitole (1) jsou probrány úplné základy vybraných partií počítačových sítí jakožto úvod do řešené problematiky.
Kapitola druhá (2) pojednává převážně o způsobech správy zařízení využívajících MikroTik RouterOS.
O úvod do databázových systémů je postaráno kapitolou třetí (3). Čtvrtá kapitola (4) zase pojednává o vybraném grafickém rozhraní.
Pátá kapitola (5) popisuje samotné implementace hlavních řešených problémů.
Kapitola šestá (6) je věnována návrhu architektury aplikace budoucí s ukázkami implementací ve vývojové verzi aplikace.

1 Síťová zařízení

Dnešní svět, silně propojený celosvětovou počítačovou sítí Internet, si bez její existence nedokážeme představit. Díky této technologii lze docílit bezproblémové realtime komunikace, a to ze kteréhokoli místa na světě na kterékoli jiné. Díky standardizaci celosvětového rozsahu je možné takové sítě stavět a provozovat [1]. Jakkoliv malá část této sítě potřebuje někoho, kdo se o její chod bude starat. Lidem, kteří se o chod starají, říkáme správci sítě či zkráceně jen správci. Ti se starají o jednotlivé prvky sítě.

1.1 Prvky sítě

Jak již bylo řečeno, sítě se skládají z jednotlivých prvků, o které je třeba se start a udržovat je v chodu. Základní dělení síťových prvků je na pasivní pravky a aktivní.

1.1.1 Pasivní prvky

Pasivní síťové prvky jsou prvky, které nezasahují do přenášených dat, nepracují s jejich částmi a nedokáží reprezentovat význam přenášených dat. Takovými prvky jsou například kabely přenášející signál.

1.1.2 Aktivní prvky

Aktivní prvky jsou opakem prvků pasivních. Ze své podstaty pracují s přijatým signálem, dále ho reprezentují, popřípadě s daty manipulují. Tyto prvky z pravidla obsahují implementace protokolů a standardů. Jejich konfigurace není automatická, je třeba, aby byly nastaveny dle potřeb a požadavků na jejich chování.

1.2 MikroTik

Na světě existuje nespočet firem, malých či velkých, které se zabývají vývojem či výrobou speciálního softwaru či hardwaru pro síťové aplikace. Namátkou lze vybrat firmy známé, jejichž jména se stala v oboru ikonickými, jako jsou společnosti CISCO Systems [2], HP (Hewlett-Packard) [3] a její odnoš Enterprise [4], Arista [5]. Z řady menších, ale stále dosti významných firem, stojí za zmínku firma Ubiquiti [6], která se snaží o zjednodušení konfigurace a správy aktivních síťových prvků.
Zaměřením této práce je však litevská firma SIA Mikrotīkls, známá jako MikroTik [7]. Jedná se o výrobce kompletních síťových řešení v oblasti aktivních prvků

ve smyslu výroby vlastního síťového hardwaru [8] a především síťového softwaru RouterOS [9].

Obr. 1.1: Ukázka routeru společnosti MikroTik

2 RouterOS

Síťový oprační systém ROS společnosti MikroTik je komunitou vysoce oblíbený a rozšířený napříč spektrem síťových zařízení. Jeho distrubuce probíhá společně s hardwarovými produkty. Díky oblibě komunity se také stal samostatným produktem společnosti. Lze jej pořídit samostatně a provozovat přímo na strojích s architekturou x86 či využít verzi CHR a provozovat tento sytém jako virtuální stroj. Tímto operačním systémem je nabízena konfigurace nepřeberného množství protokolů převážně na prvních třech vrstvách ISO/OSI modelu [11].

2.1 Konfigurace

Sytémem ROS je nabízeno velké množství způsobů konfigurace. K navigaci je použita stromová struktura příkazů, ve které je třeba se pro podrobnější správu vydat po dané větvi dál od konfiguračního kmene až k samotným možným argumentům příkazů. Na následujícím obrázku je zobrazen seznam možností přístupů konfigurace a port, na kterém služba aktuálně naslouchá. Stojí za zmínku, že všechny zobrazené služby běží na standardních (výchozích) portech.

Name	Port
api	8728
api-ssl	8729
ftp	21
ssh	22
telnet	23
winbox	8291
www	80
www-ssl	443

Obr. 2.1: Některé z možností konfigurace ROS

2.1.1 api

Jedná se o přímé napojení využívající komunikaci na úrovni bytů, kterou je třeba dekódovat. Samotná užitečná komunikace pak probíhá ve formátu příkazového slova společně s jeho argumenty, následuje odpověď daného zařízení ve formě výčtu ve formátu parametr – hodnota.

Ukázka komunikace pomocí api s využitím ukázkového skriptu [12], zkráceno

```
<<< /ip/address/print
<<<
>>> !re
```

```
>>> =.id=*2
>>> =address=10.255.255.255/32
>>> =network=10.255.255.255
>>> =interface=lo
>>> =actual-interface=lo
>>> =invalid=false
>>> =dynamic=false
>>> =slave=false
>>> =disabled=false
```

Ukázka nevyužívá všechna možná slova a parametry, které lze použít při komunikaci. Pro ukázku je tento jednoduchý příkaz dostatečný.

2.1.2 api-ssl

Jedná se o šifrovanou variatu api . Systém ROS podporuje rozsáhlou práci s certifikáty (vytváření vlastních CA, podepisování či vytváření self-signed certifikátů, atd). Právě tento způsob komunikace bude využit pro interakci se zařízeními společně se self-signed certifikáty vygenerovanými přímo na zařízení.

2.1.3 ftp

FTP je primitivní protokol pro přenos souborů. Pro účel této práce je bezvýznamný.

2.1.4 console, ssh, telnet

Konzole slouží pro správu zařízení pomocí příkazů v graficky založených (www, www-ssl, winbox) konfiguračních přístupech. K dispozici je také fyzické rozhraní konzole (USB, RJ-45) pro přímé připojení k zařízení.

Výpis 2.1: Ukázka konzolového příkazu a jeho výstupu

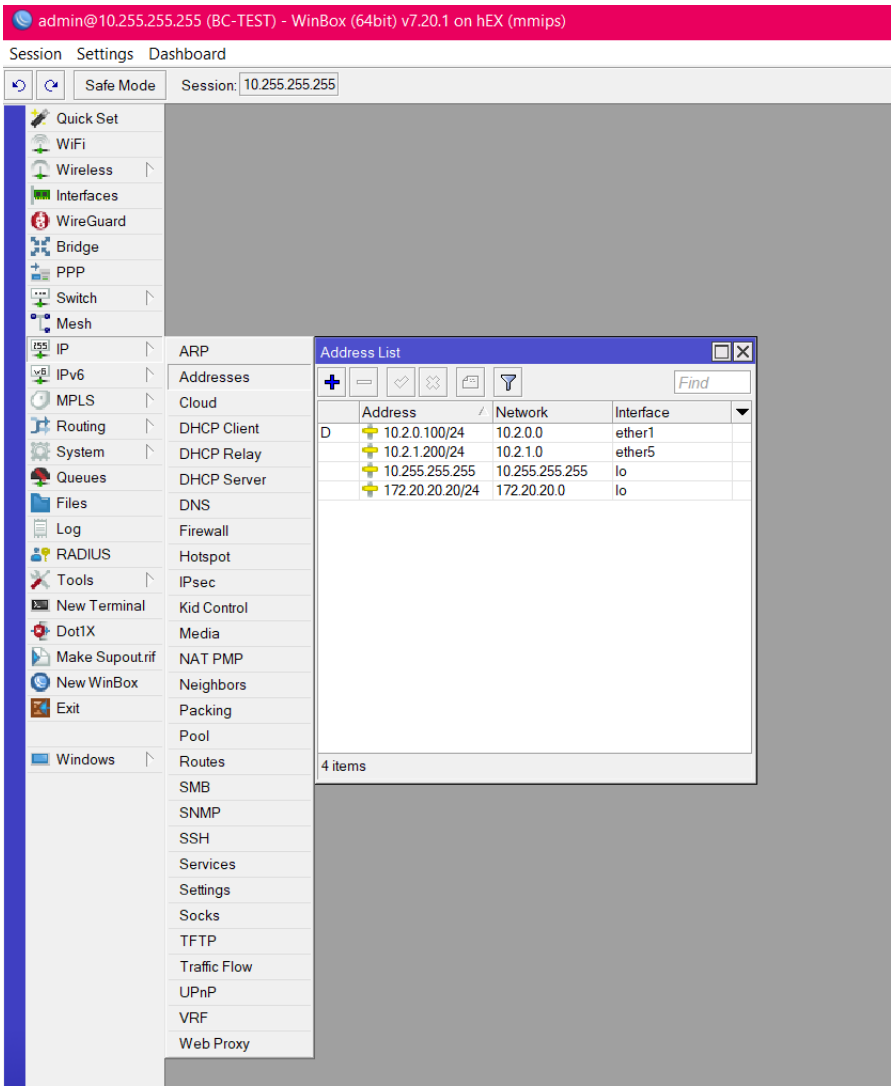
```
[admin@BC-TEST] > /ip/address/print
Flags: D - DYNAMIC
Columns: ADDRESS, NETWORK, INTERFACE
#    ADDRESS             NETWORK          INTERFACE
0    10.255.255.255/32   10.255.255.255   lo
1    10.2.1.200/24       10.2.1.0         ether5
2 D  10.2.0.100/24       10.2.0.0         ether1
3    172.20.20.20/24     172.20.20.0      lo
```

Telnet je protokol sloužící ke vzdálenému přístupu právě přes zmíněnou konzoli. Pro práci není dále důležitý.

SSH je protokol sloužící k šifrovanému vzdálenému přístupu právě přes zmíněnou konzoli. Pro práci není dále důležitý.
Správa pomocí Telnetu a SSH je tedy pro uživatele obdobná, jako přímé připojení přes konzoli a vypadá obdobně, jako uvádí výpis 2.1.

2.1.5 winbox

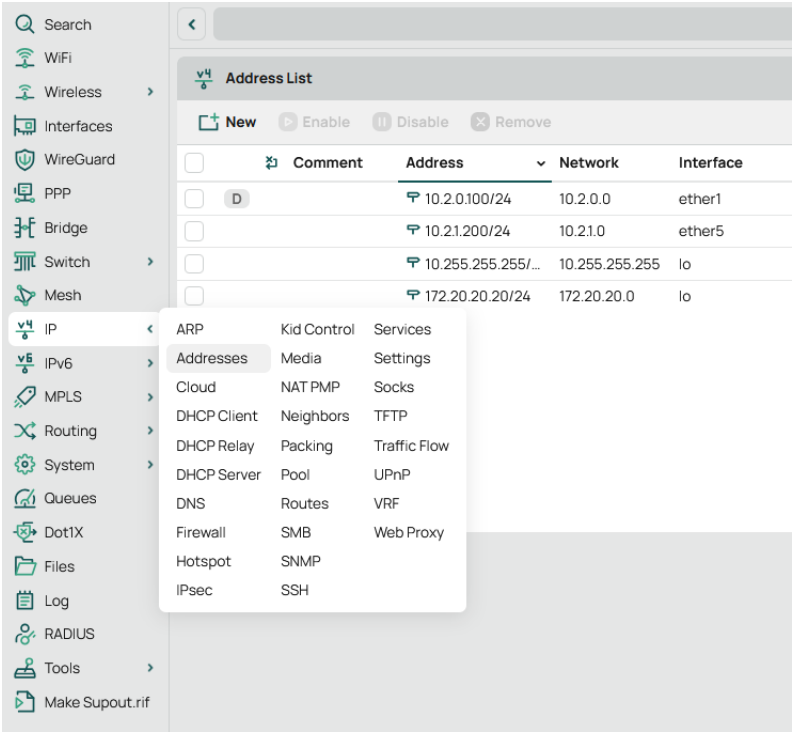
WinBox je grafický proprietární konfigurační software pro zařízení se systémem Mikrotik RouterOS. Je založen na systému oken s konfiguračními podokny, pomocí kterých se správa provádí. Lze konstatovat, že se jedná o grafické zobrazení stromové hierarchie příkazů dostupných v konzoli.



Obr. 2.2: Ukázka grafického rozhraní WinBox

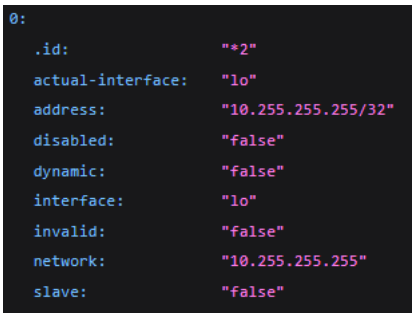
2.1.6 www, www-ssl, REST

WebFig je webový konfigurační nástroj připomínající vizuálem a funkčností WinBox. Na rozdíl od WinBoxu však nedokáže pracovat s vícero okny zároveň ale pouze s jedním v daný okamžik. Stejně jako api má i svou zabezpečenou variantu využívající certifikáty.



Obr. 2.3: Ukázka webového rozhraní WebFig

V rámci webové služby je dostupná i možnost přístupu pomocí dnes již standardního prostředí REST api. Tato funkcionalita vyžaduje funkční zabezpečené přípojení.



Obr. 2.4: Ukázka odpovědi rozhraní REST api

3 Databáze

Jedním ze základních úkolů této práce pro autora byl návrh databáze a systému ukládání hesel. Struktura databáze má dalekosáhlé důsledky pro návrh celé architektury aplikace a jejím budoucím možnostem.

3.1 Možné realizace databáze

Autor vybíral ze základních dvou přístupů pro práci s databází. Jedním z nich je přístup pomocí serveru. V tomto modelu je třeba serverová část, spuštěná lokálně či na vzdáleném serveru, ke které se uživatel připojí a využívá její služby pomocí síťového stacku operačního systému. Má však velké úskalí, díky kterému je dle autora nevhodná pro tuto aplikaci.
Při lokálně spuštěné serverové instanci je nejprve nutno tuto instanci na daném počítači nainstalovat a spustit. Takové řešení sice je možné, ale autor má za to, že takové řešení přidává zbytečnou komplexitu projektu.
Při použití vzdáleného serveru je předpokládána funkčnost síťového spojení na daný server a jeho funkčnost. Z logiky vyplývá, že by zde mohl nastat zásadní problém, a to v momentu výpadku. Nastala by situace, kdy by síťové prvky, nutné pro spojení s databázovým serverem, nemohly být nakonfigurovány, jelikož přístupové údaje k těmto prvkům by byly uloženy na v tu chvíli nedostupném serveru. Z výše uvedeného vyplývají zásadní možné chyby a problémy dané realizace databáze, proto autor vybral alternativní metodu uvedenu níže.

3.1.1 SQLite

Díky předešlým úvahám a vlastnostem uvedených systémů autor využil v aplikaci SQLite [13], konkrétně implementaci tohoto systému knihovnou sqlite3 [14]. Jedná se o lightweight knihovnu obsaženou přímo v pythonu, není tedy třeba její doinstalace. Tento systém nevyužívá spojení na databázový server, nýbrž lokální databázový soubor. Aplikace tedy bude fungovat i v případě komplentního odpojení od sítě či jejím výpadku. Je třeba zmínit hlavní nevýhodu tohoto přístupu, a toum je nutnost databázové soubory distribuovat či synchronizovat. Dále také fakt, že nemůže být přistupováno k databázovému souboru vícero uživateli najednou. Při využívání databázového souboru je soubor zamčen pro čtení i zápis ostatním uživatelům. Potenciální problémy s přístupem k databázovému souboru však nejsou pro tuto aplikaci relevantní, jelikož aplikaci může využívat v daném okamžiku na jednom počítači pouze jeden uživatel. Otázka distribuce a synchronizace bude popsána dále.

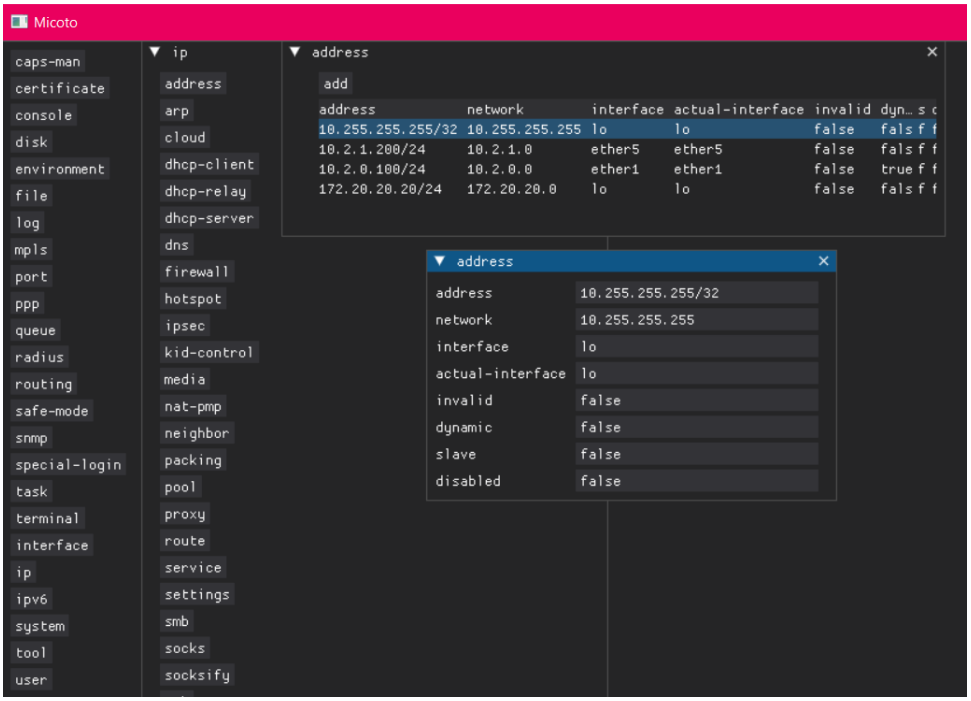
Proto má autor za to, že řešení pomocí systému SQLite, používající lokální databázové soubory, bude nejvhodnějším řešením.

3.2 Ukládání hesel

Ačkoli je autorovi známo, že ukládání hesel jako takových je naprosto nevhodné a silně nedoporučené, v tomto případě není zbytí. Systém ROS nepodporuje pro přístup pomocí api jinou autentizační metodu nežli přihlášení uživatelským jménem a heslem. Jelikož je jeden pohled aplikace zamýšlen spíše uživatelsky, viz dále, je nutné vytvořit jednoduchý systém, kdy po zadání jednoho hesla budou zpřístupněna všechna zařízení. Protože nastavení stejného hesla do vícero zařízení není z bezpečnostního hlediska lepším řešením, zvolil autor řešení využívající databázi se zašifrovanými přístupovými hesly.
Prvně bylo autorem zamýšleno využít systém symetrického šifrování s využitím knihovny cryptography, a to konkrétně již připraveným systémem Fernet [15]. Tento systém je jednoduchý na implementaci, avšek po konzultaci s vedoucím práce bylo rozhodnuto pro jeho nevyužití, jelikož pro šifrování je využíván AES-128-CBC. Ačkoli je AES-128 dnes stále považován za bezpečný a splňuje minimální požadavky NÚKIBu [16], bylo rozhodnuto o nevyužití tohoto systému a implementaci řešení využívající AES-256-CBC, na který je aktuálně nahlíženo jako i do budoucna bezpečný algoritmus.

4 Grafické rozhraní

Práce s aplikací má probíhat pomocí GUI. Programovací jazyk python nabízí nemalé množství knihoven, které implementaci grafických rozhraní ulehčují [17]. Jelikož je předpoklad funkční podobnosti aplikace s aplikací WinBox, bylo rozhodnuto využít knihovnu DearPyGui [18] [19]. Jedná se o lightweight grafickou knihovnu s hardwarovou akcelerací. Hlavním důvodem výběru této knihovny je možnost jednoduché práce s okny v okně. Předpoklad této funkce je stěžejní pro celou koncepci programu.



Obr. 4.1: Grafické rozhraní konfiguračního okna (vývojová verze)

5 Implementace

V předešlých částech byla probrána základní teorie nutná pro pochopení problémů, se kterými se řešitel musel vypořádat. Tato kapitola je věnována implementaci daných částí kódu s textovým popisem pro jednodušší pochopení a s doplňujícími informacemi. Aplikace byla pojmenována **micoto** (Mikrotik configuration tool) a její aktuální verze je dostupná na `https://github.com/jir14/micoto`, branch ***main***. Pro zájemce je zde také dostupná branch ***dev***, ve které se nachází vývojová verze aplikace.

5.1 Komunikace api

Pro zpracování komunikace se samotnými zařízeními byl využit a autorem upraven ukázkový skript [12]. Ve skriptu bylo třeba upravit nefunkční navázání zabezpečeného spojení.

Výpis 5.1: Ukázka upraveného skriptu pro komunikaci s api

```
def open_socket(dst, port, secure=False):
  s = None
  res = socket.getaddrinfo(dst, port, socket.AF_UNSPEC, socket.
   SOCK_STREAM)
  af, socktype, proto, canonname, sockaddr = res[0]
  skt = socket.socket(af, socktype, proto)
  if secure:
    context = ssl.create_default_context()  # vytvoření contextu
    context.check_hostname = False      # vypnutí kontroly hostname
    context.verify_mode = ssl.CERT_NONE   # umožní fungování i se
   self-signet certifikáty
    s = context.wrap_socket(skt, server_hostname=dst) # zapouzdření
    socketu do ssl/tls
  else:
    s = skt
  s.connect(sockaddr)
  return s
```

Dále byly odstraněny veškeré elementy sloužící ke komunikaci s uživatelem prostřednictvím příkazové řádky. Výstupy takto upraveného skriptu jsou využívány jako vstupy do parserů. Parsery mají za úkol zpracovat přijatou zprávu, provést s ní dané operace a vrátit její výstup ve tvaru slovníku, jehož položky lze jednoduše programově procházet.

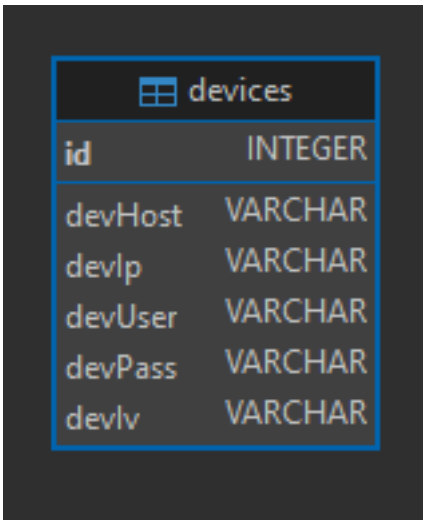
Výpis 5.2: Ukázka části kódu parseru

```
def printDir(self, dirID, id=None, bID=None):
  ...
  for re in self.api.talk(sentence):  # procházení odpovědi na
   dotaz
    if re[0]=="!re":  # kontrola, zda došla správná odpověď
      if first: # kontrola, zda se jedná o první blok odpovědi
        for k in re[1].keys():  # přidání klíčů
          k = k.replace("=","")
          if k == ".id":
            continue
          keys.append(k)
        first = False
      vals = []
      for rec in re[1].values():  # přiřazení hodnot ke klíčům
        if "*" in rec:
          ids.append(rec.replace("*",""))
          continue
        vals.append(rec)
      values.append(vals)
  ...
```

Komunikace se zařízením byla vyzkoušena do výrazně větší míry, než je použita v ukázkové aplikaci. Jelikož pro funkčnost aplikace v aktuálním stavu není rozšířená funkčnost podstatná, její přesný popis v aktuální chvíli není potřebný.

5.2 Ukládání zařízení

S přihlédnutím k výše uvedenému a zkušenostem autora bylo rozhodnuto vytvořit samostatný databázový soubor obsahující tabulku se všemi dostupnými zařízeními. Hesla jsou zde zabezpečena šifrováním AES-256-CBC. Bez znalosti klíče, v této aplikaci nazývaného jako „master password“, a přiloženého inicializačního vektoru jsou pro potenciálního útočníka takto uložená hesla nepoužitelná.
Na obrázku 5.1 můžete vidět již naimplementovanou tabulku *devices* v databázi. Hodnoty polí v tabulce jsou dále popsány v seznamu níže společně s ukázkou zašifrovaného hesla (5.2).



Obr. 5.1: Diagram tabulky s hesly

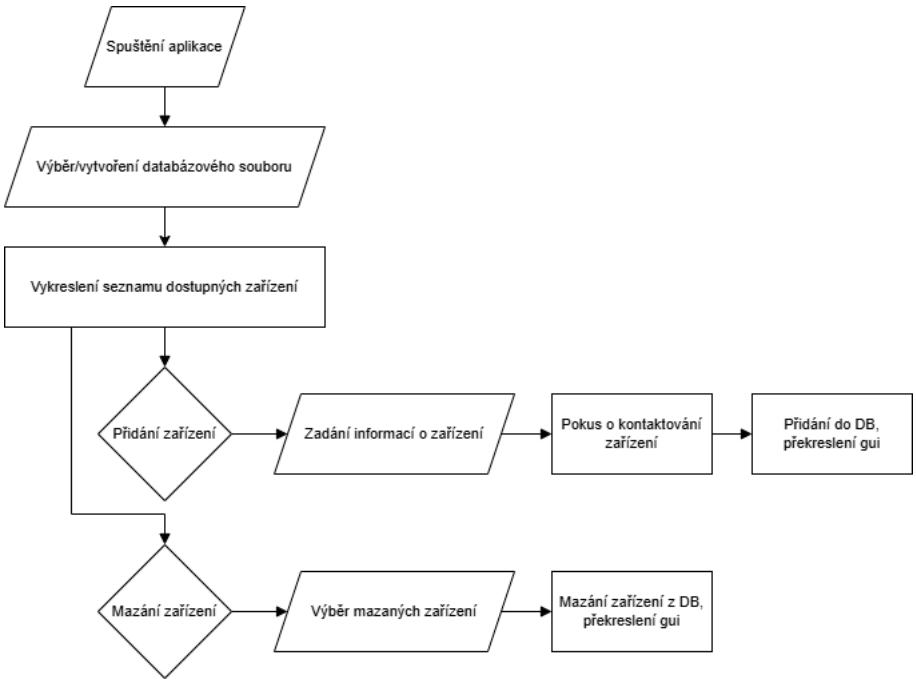
id - automaticky generovaný jednoznačný identifikátor záznamu
devHost - automaticky doplněný hostname (v ROS položka */system/identity*)
devIp - IP adresa zařízení
devUser - uživatelské jméno pro přihlášení
devPass - zašifrované heslo
devIv - inicializační vektor šifrátoru

```
e6 01 eb 17 cd d3 6b 9d fa 3e 81 84 fa d9 21 f2
f6 7e 8f ba f0 52 ba 47 f5 fc 73 ce f0 17 7b d1
```

Obr. 5.2: Ukázka zašifrovaného hesla v databázi

6 Architektura

Součástí zadání je také návrh architektury budoucí aplikace. Ta byla navržena dle požadavků práce a vedoucího s ohledem na jednoduchost a funkčnost.
Na obrázku 6.1 můžete vidět aktuální digram fungování aplikace.

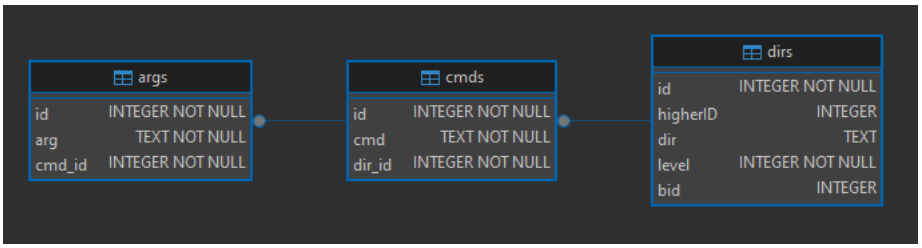


Obr. 6.1: Diagram aplikace

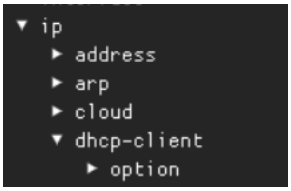
Celkový vývoj aplikace je v pokročilejším stádiu než uvedená aplikace, a tak bude architektura aplikace předvedena na příkladech již funkčních částí kódu finální aplikace.

6.1 Databáze

Kromě databáze zařízení (5.1) aplikace využívá také databázi dostupných příkazů. Příkazy jsou uloženy v samostatné databázi, která není určena pro uživatelskou interakci. Zdůvodnění výběru této architektury je uvedeno; dále, viz 6.1.1. Grafická reprezentace stromové struktury příkazů je zobrazena na obrázku 6.3.



Obr. 6.2: Diagram navrhované databáze pro ukládání příkazů



Obr. 6.3: Grafická reprezentace stromové struktury příkazů systému ROS

6.1.1 Pohledy

Spousta systémů síťových prvků nabízí možnost využít tzv. „views“ či „role-based“ přístup [20]. Tato funkcionalita umožňuje definovat, k jakým příkazům a jejich parametrům bude mít daný uživatel systému přístup. Právě tato funkcionalita v systému ROS chybí. Díky výše zmíněné volbě architektury (6.1) využívající lokální databázi dostupných příkazů lze vytvořit pro daného uživatele speciální databázový soubor obsahující pouze příkazy a parametry, které potřebuje ke své práci, a to za účelem zjednodušení či jako „bezpečnostní“ faktor, kdy není uživatel schopen jednoduše procházet či měnit části konfigurace z aplikace, s jejichž fungováním není plně seznámen a/nebo k nim nemá mít přístup.

6.2 Konfigurační grafické rozhraní

Grafické konfigurační rozhraní je silně inspirováno aplikací WinBox. Vyvíjená aplikace implementuje vše dobré, co nabízí WinBox a má za cíl přidat další funkcionality, jako výše zmíněný „role-based“ přístup či hromadné konfigurace pomocí šablon. Konkrétní implementace těchto funkcí je zatím v plenkách, idea je však taková, že pro uživatelem vybraná zařízení bude možno otevřít jedno konfigurační rozhraní. Veškeré změny provedené v tomto rozhraní budou aplikovány na všechna dříve zvolená zařízení zaráz.
Ukázka grafického rozhraní z vývojové verze je zobrazeno na obrázku 4.1.

6.3 Administratorská aplikace

Pro správu dostupných příkazů, seznamu dostupných zařízení a dalších prvků bude vyvinuta speciální aplikace, která bude nabízet grafické rozhraní pro výběr dostupných příkazů s využitím „tree view“ (6.3). Dále bude administrátor přidávat a odebírat zařízení dostupná na konfiguraci. Databáze konfiguračního stromu bude distribuována společně s aplikací. Databáze zažízení může být vybrána uživatelem aplikace, jde tedy využívat různé databáze, které lze jednoduše synchronizovat mezi počítači. Administrátor aplikace pak pouze upraví verzi na své straně a uživatelům bude doručena nová databáze dostupných zařízení. Díky této architektuře lze každému uživateli vytvořit konfigurační strom a seznam zařízení na míru. Lze také distribuovat vícero databází dostupných zařízení. Využívání databází určených pro danou skupinu uživatelů lze jednoduše vynutit sdělením „master passwordu“ pouze k dané databázi. Ostatní databáze budou pro daného uživatele nepoužitelné. Ukázka vývojové verze aplikace se nachází na obr. 4.1.

Závěr

Cílem semestrální práce bylo vymyslet architekturu aplikace, systém ukládání hesel a ověření funkčnosti komunikace pomocí api.
Architektura byla vymyšlena s ohledem an budoucí rozšíření aplikace v rámci bakalářské práce. Architekura je tvořena pružně spojením samostatně fungujících celků. Dás se konstatovat, že aplikace je vyvíjena velice dynamicky, takže je vhodná i pro budoucí využití. Je však nutno podotknout, že aplikace je vyvíjena pro systém ROS verze 7 a vyšší!
Systém ukládání hesel využívá AES-256-CBC, která je považována za bezpečnou dnes i do budoucna. Stojí také za zmínku, že aplikace pracuje s hesli v otevřené podobě pouze interně, tudíž hesla nejsou nikde v uživatelském rozhraní dostupná.
Pro připojení je využit modifikovaný ukázkový skript, jehož funkčnost byla ověřena v rámci testování systému komunikace, a to především při implementace funkcí dostupných již nyní v *dev* branch.

Literatura

[1] IEEE. *The world's largest technical professional organization dedicated to advancing technology for the benefit of humanity.* Online. https://www.ieee.org. [cit. 2025-10-30].

[2] Cisco Systems *AI Infrastructure, Secure Networking, and Software Solutions* Online. https://www.cisco.com/site/us/en/index.html. [cit. 2025-10-30].

[3] HP *Laptop Computers, Desktops, Printers, Ink amp; Toner | HP® Official Site* Online. https://www.hp.com/us-en/home.html. [cit. 2025-10-30].

[4] HP Enterprise *Hewlett Packard Enterprise (HPE) | Czechia* Online. https://www.hpe.com/cz/en/home.html. [cit. 2025-10-30].

[5] Arista Networks *Data-Driven Cloud Networking - Arista* Online. https://www.arista.com/en/. [cit. 2025-10-30].

[6] Ubiquiti Networks *Ubiquiti - Rethinking IT - Ubiquiti* Online. https://ui.com/. [cit. 2025-10-30].

[7] SIA Mikrotīkls *MikroTik Routers and Wireless* Online. https://mikrotik.com/. [cit. 2025-10-30].

[8] MikroTik Routers and Wireless *Products* Online. https://mikrotik.com/products. [cit. 2025-10-30].

[9] MikroTik Routers and Wireless *Software* Online. https://mikrotik.com/download. [cit. 2025-10-30].

[10] MikroTik RB750Gr3 *Picture* Online. Dostupné z: https://cdn.mikrotik.com/web-assets/rb_images/1406_lg.webp. [cit. 2025-11-27].

[11] X.200 : Information technology - Open Systems Interconnection *Basic Reference Model: The basic model* Online. https://www.itu.int/rec/T-REC-X.200/en/. [cit. 2025-11-08].

[12] RouterOS - MikroTik Documentation *Python3 Example* Online. https://help.mikrotik.com/docs/spaces/ROS/pages/47579209/Python3+Example. [cit. 2025-11-08].

[13] SQLite *Home Page* Online. https://sqlite.org/. [cit. 2025-11-13].

[14] sqlite3 — DB-API 2.0 interface for SQLite databases *Python 3.14.0 documentation* Online. https://docs.python.org/3/library/sqlite3.html. [cit. 2025-11-13].

[15] Fernet (symmetric encryption) *Cryptography 47.0.0.dev1 documentation* Online. https://cryptography.io/en/latest/fernet/. [cit. 2025-11-13].

[16] Národní úřad pro kybernetickou a informační bezpečnost *Doporučení v oblasti kryptografických prostředků* Online. https://nukib.gov.cz/download/uredni_deska/Minimalni_pozadavky_v4_FINAL.pdf. [cit. 2025-11-13].

[17] GuiProgramming - Python Wiki *GUI Programming in Python* Online. https://wiki.python.org/moin/GuiProgramming. [cit. 2025-11-13].

[18] GitHub - hoffstadt/DearPyGui *Dear PyGui: A fast and powerful Graphical User Interface Toolkit for Python with minimal dependencies* Online. https://github.com/hoffstadt/DearPyGui. [cit. 2025-11-13].

[19] Dear PyGui's Documentation *Dear PyGui documentation* Online. https://dearpygui.readthedocs.io/en/latest/index.html. [cit. 2025-11-13].

[20] Cisco Systems *Role-Based CLI Access* Online. https://www.cisco.com/en/US/docs/ios/12_3t/12_3t7/feature/guide/gtclivws.html. [cit. 2025-11-14].

Seznam symbolů a zkratek

CHR Cloud Hosted Router (13)
ROS RouterOS (13)
CA Certification authority (Cerifikační autorita) (14)
GUI Graphical user interface (Grafické uživatelské rozhraní) (19)

Seznam příloh