VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2018 Bc. Adrián Bárdossy

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 SPRÁVU SÍŤOVÝCH PRVKŮ MIKROTIK

MULTIPLATFORM APPLICATION FOR MIKROTIK NETWORK DEVICES MANAGEMENT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Adrián Bárdossy**
AUTHOR

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

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Adrián Bárdossy ***ID:*** 154674
Ročník: 2 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Multiplatformní aplikace pro správu síťových prvků Mikrotik

POKYNY PRO VYPRACOVÁNÍ:

Vytvořte interaktivní aplikaci pro hromadnou správu sítě založené na aktivních prvcích Mikrotik. Aplikace bude využívat Mikrotik API-SSL, uživatelské rozhraní bude realizováno v jazyce Python a přenositelné mezi různými operačními systémy.

DOPORUČENÁ LITERATURA:

[1] BURGESS, Dennis. Learn RouterOS. Lexington]: Dennis Burgess, 2009, 391 s. : il. ISBN 978-0-557-09271-0.

[2] ROMANO, Fabrizio, Dusty PHILLIPS a Rick van HATTEM. Python. Birmingham: Packt Publishing, 2016.

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 21.5.2018

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

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

KĽÚČOVÉ SLOVÁ

ABSTRACT

KEYWORDS

BÁRDOSSY, Adrián *Multiplatformní aplikace pro správu síťových prvků Mikrotik*: diplomová práca. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2018. 45 s. Vedúci práce bol Ing. Ondřej Krajsa, Ph.D

PREHLÁSENIE

Prehlasujem, že som svoju diplomovoú prácu na tému „Multiplatformní aplikace pro správu síťových prvků Mikrotik“ vypracoval(a) samostatne pod vedením vedúceho diplomovej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor(ka) uvedenej diplomovej práce ďalej prehlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil(a) autorské práva tretích osôb, najmä som nezasiahol(-la) nedovoleným spôsobom do cudzích autorských práv osobnostných a/nebo majetkových a som si plne vedomý(-á) následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona č. 121/2000 Sb., o právu autorském, o právoch súvisejúcich s právom autorským a o zmeně niektorých zákonov (autorský zákon), vo znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhé, hlavy VI. diel 4 Trestného zákoníka č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

POĎAKOVANIE

Rád by som poďakoval vedúcemu bakalárskej práce pánovi Ing. Ondřejovi Krajsovi, Ph.D. za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci.

Brno ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic
http://www.six.feec.vutbr.cz

POĎAKOVANIE

Výzkum popsaný v tejto diplomovej práci bol realizovaný v laboratóriách podporených projektom SIX; registračné číslo CZ.1.05/2.1.00/03.0072, operačný program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .
podpis autora(-ky)

OBSAH

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

1 ÚVOD DO DIPLOMOVEJ PRÁCE

Diplomová práca na tému "Multiplatformní aplikace pro správu síťových prvku Mikrotik"sa bude primárne zaoberať samostatným mikrotikom. Primárne pomocou application programable interface (API) vytvorenie jej konzolovej časti (backendu) a grafickej časti (frontendu). Tieto dve časti dajú celkovú applikáciu dokopy ako celok.

V prvej časti práci bude definovanie Mikrotik API a jeho možností, porovnanie podobnosti s operačným systémom unix. Ďalej budú popísané možnosti zabezpečenia API pomocou secure socket layer (SSL). Budú tu tiež spomenuté použité porty,a ďalšie možnosti.

V druhej časti práce bude popis API a spôsoboch softvérového riešenia aplikácie pre správu Mikrotikov. Táto časť bude tiež obsahovať niečo ohľadom technológie git, popise, čo je git, princíp tzv. commitu a pushu. Rozdiely medzi vetvami, prepínanie medzi vetvami a pridávania zmien. Taktiež tu bude spomenutý aj úvod do certifikátov a to konkrétne Single Sign-on metódy.

V ďalšej časti bude návrh riešenia softvérovej implementácie aplikácie. Bude obsahovať popis, princípy, diagramy, hlavne Unified modeling language (UML), popisy knižníc, jednotlivých tried a modulov. Každý modul bude popísaný svojou funkcionalitou, parametrami a výstupom s praktickými ukážkami.

V ďalšej časti bude použitá implementácia softvérového návrhu riešenia. Bude tu riešenie ako v konzolovej časti, jeho ukážky, test a výsledky.

V poslednej časti práce bude ukážka grafického spracovania konzolovej časti aplikácie a ich prepojenia do jednej aplikácie, spoločne s ukážkami kódov, testu a výsledkov.

2 MIKROTIK A ROUTEROS (SWITCHOS)

V dnešných malých a stredne veľkých firmách sa na správu siete používajú prevažne routre a switche typu Mikrotik. Mikrotik je firma vyvíjajúca routre a switche,prístupové body a ďalšie sieťové prvky v Litve.
Mikrotik zariadenia používajú operačný systém routerOS, prípadne switchOS. Rozdiel medzi nimi je na základe použitého zariadenia. Čo sa týka routrov, používa operačný systém routerOS, switch používa switchOS, v prípade prístupových bodov (AP) je to routerOS.

2.1 Mikrotik API

Za pomoci Mikrotik API môžeme programovať užívateľské programy a prostredia na riadenie a konfiguráciu Mikrotik zariadení. V dnešnej dobe existuje softvér na konfiguráciu mikrotik zariadení a to pod názvom **Winbox**. Winbox v dnešnej dobe existuje len na operačný systém Windows a Macintosh (MAC). Bohužiaľ na operačný systém Linux winbox samostatne neexistuje a musí sa simulovať pomocou emulátoru Windows aplikácií za pomoci programu Wine. Toto spôsobuje komplikácie pri použití niektorých funkcií winboxu ale aj iných programov operačného systému Windows. Výstupom práce bude práve Graphical User Interface (GUI).

2.1.1 Požiadavky na použitie API

- Verzia routerOS verzie *3.0.X* a vyššie [1]

2.1.2 Porty

Základné porty na použitie Mikrotik API [1] sú:

- **API port**: 8728
- **Application programable interface Secure Socket Layer (API-SSL) port**: 8729

2.1.3 Základný port 8728

Na základné pripojenie k API aplikácii na prvku Mikrotik musí byť povolený port 8728, ktorý tiež nájdeme v IP-> Services spoločne s API-SSL.
Na základné pripojenie nie je potreba žiadneho transport layer security (TLS) certifikátu. Stačí jednoducho napísať kód a skompilovať ho.

2.1.4 SSL port 8729

Pre použitie portu 8729 tiež známeho ako API-SSL portu je potreba zabezpečenej komunikácie pomocou SSL protokolu.
Primárne muisú byť natavený port, základný port 8729 v IP -> Services. Môžeme ale definovať aj užívateľsky definovaný port.
Možnosti nastavenia API-SSL:

- prístup bez certifikátu TLS
- prístup pomocou certifikátu TLS

Prístup pomocou certifikátu TLS

Pre použitie certifikátu TLS je potrebné vygenerovať certifikát TLS, a to na certifikačnej autorite alebo na ľubovoľnej linux stanici ideálne, ale tiež to dokážeme spraviť aj na WIndows stanici či MAC. Spôsoby vygenerovania certifikátov:

- openssl
- easy-rsa
- Windows Server Certificate Services

Openssl

Openssl [4] je softvér na generovanie certifikátov pre komunikáciu v počítačovej sieti. Koreňovo sa používa na prístup na web skrz protokol Hyper Trasfer Transport Protocol Secure (HTTPS). Pre vygenerovanie certifikátov sa musí vygenerovať:

- certifikát **.crt*
- certifikačný požiadavok **.csr*
- kľúč k certifikátu **.key*

Easy-rsa

Softvér easy-rsa [2] sa používa na vytvorenie open-source certifikačnej autority a užívateľých certifikátov napr. pre potreby HTTPS spojenia.
Po nainštalovaní easy-rsa napr. na Ubuntu príkazom *sudo apt install easy-rsa* sa musí spraviť nasledovné:

- Nakopírovanie konfiguračných súborov do zložky autority
- Vytvorenie šablóny na vygenerovanie certifikačnej autority
- Vytvorenie užívateľksých certifikátov

Active Directory Certificate Services

Windows riešenie [18] pre generovanie certifikačnej autority je inštalácia roly servera Active DIrectory Certificate Services.

Pre použitie certifikačnej autority na Windows servery je potreba:

- Inštalácia role serveru
- Nadefinovanie certifkačnej autority
- Generovanie certifikátov

2.2 API slová

API slová [1] sú základnou časťou API "vety". API "veta"predstavuje príkaz v použití príkazu napr. */ip/address/print*, */ip/address/add address="10.1.1.1/24ïnterface=ëther1"*.
Parametre na slová:

- každé slovo má svoju zakódovanú dĺžku t.j.
  - 0 - 127 bitov zaberá 1 Byte
  - 128 - 1023 bitov zaberá 2 Byty
  - 1024 bitov - 2097 kib zaberá 3 Byty
  - viac ako 2098 kib zaberá 4 Byty
- jednotlivé slová súzaradené do viet
- maximum bztov na slovo sú 4 Byty
- kontrolné byty sa nepoužívajú

2.3 Príkazové slová API

Slová Mikrotik API sa zaraďujú do API viet použitím API slov, na ktoré platia požiadavky, ktoré sú spomenuté v kapitole 2.2.Na použitie API viet je potreba začínať znakom /. Napr. miesto *ip address print* sa použije */ip/address/print*.
Pre úplnosť API viet musí platiť [1]

- zakódovaná dĺžka slova
- slovo musí začínať znakom /
- musí byť použitá správna syntax

2.4 Použitie atribútov v príkaze a filtrovanie

V prípade konfigurácie mikrotik zariadení sa pre nastavenie jednotlivých prvkov používajú tzv. atribúty [1] napr. ip adresa, číslo pravidla, meno rozhrania, nastavenie virtuálnej lokálnej sieti (VLAN).
Použitie atribútov má špeciálnu syntax pre konfiguráciu prípadne zmenu prvku na mikrotiku, prípadne pridanie a zmazanie prvku. Na použitie atribútov sa použije špeciálny znak =. Napr. */ip/address/add =address=10.1.1.1/24 =interface=erher1*.
Pre filtrovanie prvkov v rámci mikrotik API syntaxe sa používa špeciálny atribút

parameter so znakom *?*. Napr. /ip/address/print =?type=ether1 vyfiltruje len rozhranie ether1.

2.5 Špeciálne slová API

Miktotik API má možnosť tzv. špeciálnych slov [1]. Špeciálne slová sú slová, ktoré sú rezervované a nesmú sa použiť pre iné použitie ako napríklad meno premennej, metódy, triedy, a iné. Medzi špeciálne slová patria:

- prihlásenie sa na zariadenie /login
- ukončenie spojenia na zariadenie /cancel
- odhlásenie sa zo zariadenie /logout
- získanie všetkých parametrov /getall

3 PRIPOJENIE NA MIKROTIK

3.1 Možnosti pripojenia

Pripojenie na mikrotik je realiyované pomocou niekoľkých typov softvéru:

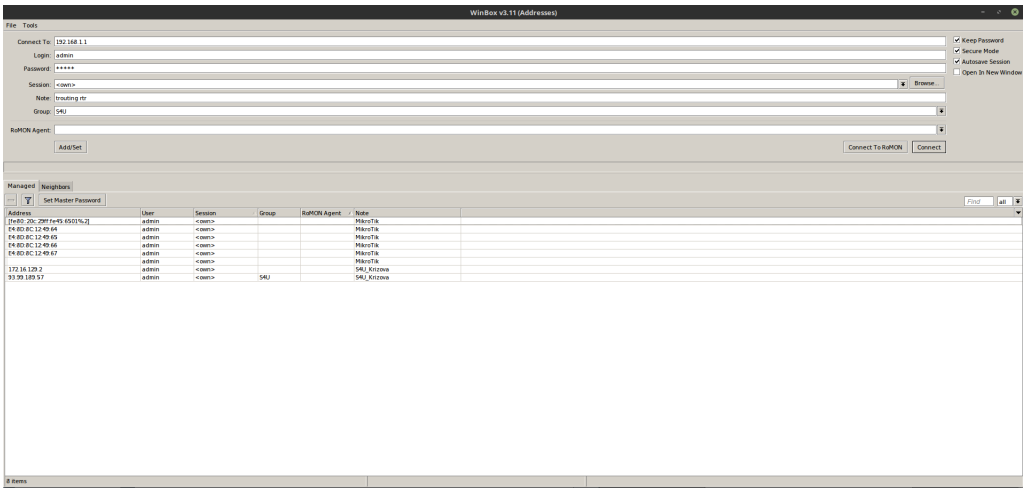
- **Winbox** - základný softvér na konfiguráciu mikrotiku
- **Webfig** - konfigurácia mikrotiku pomocou webového rozhrania štandardne na portoch 80 a 443
- Riadenie mikrotiku pripojením na mac adresu - **mactelnet**
- Pripojenie pomocou protokolu **SSH** - zabezpečené a šifrované spojenie
- Pripojenie pomocou protokolu **telnet** - nebezpečné v dnešnej dobe

3.2 Pripojenie pomocou winboxu

Winbox[17] je nástroj na administráciu mikrotiku. Medzi jeho vlastnosti patrí:

- GUI nástroj (klikátko)
- rýchlosť
- spoľahlivosť

Winbox je prepis konzolovej aplikácie do grafickej. Obsahuje tiež nástroje ktoré sa v konzole nedajú odsimulovaťnapr. graphs, torch, netmon, scheduler,...
Niektoré funkcie nevieme meniť pomocou winboxu napr. Media Access Control (MAC) adresu rozhrania.



Obr. 3.1: Winbox základné prihlasovacie rozhranie

Režimy winboxu:

- jednoduchý režim - obsahuje na pripojenie len užívateľské meno, heslo a adresu mikrotiku

- pokročilý režim - možnosť pridania skupiny mikrotikov, popisky a názov spojenia

3.3 Pripojenie pomocou webfigu

Webfig[14] je webová aplikácia RouterOS a umožňuje konfiguráciu, minitoring a údržbu prvkov RouterOS. Medzi hlavné tasky webfigu patrí:

- konfigurácia mikrotiku
- mnotring mikrotiku
- riešenie problémov na mikrotiku za pomoci webového rozhrania

Obr. 3.2: Webfig základné prihlasovacie rozhranie

3.4 Mactelnet

Mactelnet[3] predstavuje aplikačný protokol riadený na druhej vsrtve referenčného modelu. Tiež predtavuje kombináciu winboxu a telnetu v jednom protokole. Riadi prístup na napr. nový mikrotik, ktorý ešte neobsahuje žiadnu konfiguráciu. Pracuje absolútne rovnakým spôsobom ako telnet. Je možné sa pripojiť len na fyzicky pripojený mikrotik pomocou mactelnet, vzdialený prístup pomocou mactelnet nie je možný.

```
adrian@adrian-Lenovo-Z50-75 ~/Desktop/diplomka2016/bakalaris $ mactelnet  -l
Searching for MikroTik routers... Abort with CTRL+C.

IP              MAC-Address       Identity (platform version hardware) uptime
192.168.1.1     64:d1:54:53:59:72 MikroTik (MikroTik 6.41.2 (stable) RB750r2)  up 10 days 23 hours  GMB8-QCU8 ether2
```

Obr. 3.3: Výstup príkazu mactelnet

Po pripojení na mikrotik pomocou mactelnet sa nastaví základná konektivita a pripájame sa potom na základe Internet Protocol (IP) adresy.

3.5 Pripojenie pomocou telnet a SSH

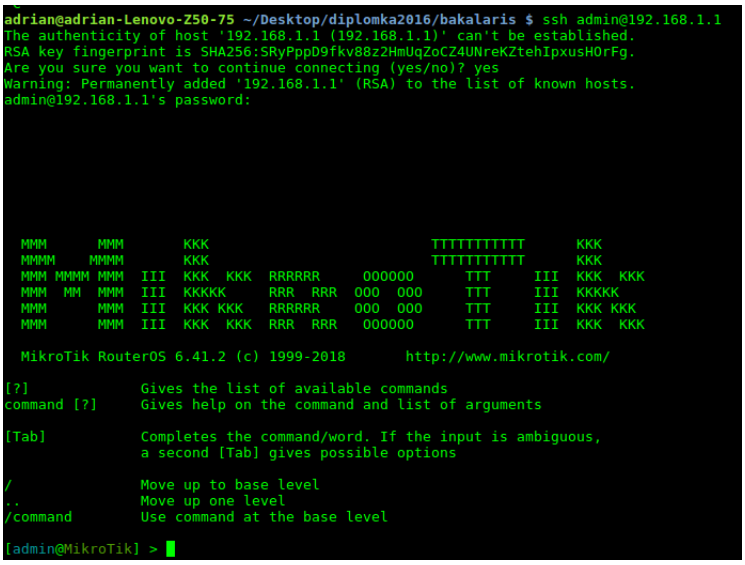
Ďalšou možnosťou pripojenia na mikrotik je prihlásenie sa pomocou telnetu[12] prípadne SSH[11] na konzolu mikrotiku. Napríklad na nastavenie fronty,firewallu,... .

3.5.1 Pripojenie cet telnet

Telnet predstavuje protokol, ktorý umožňuje pripojenie na vzdialené servery. Jeho štandardným portom je port 23.
Na povolenie pripojenia pomocou telnetu je potrebné povoliť službu telnet na mikrotiku v IP -> Services. Pre bezpečnostné účely by sa telnet nemal používať, je terčom útokov nakoľko je nešifrovaný. Pokiaľ chceme povoliť telnet na pripojenie na mikrotik, by sa mal minimálne zmeniť štandardný port z 23 na užívateľsky definovaný port.
Príklad príkazu na pripojenie na zariadenie pomocou telnetu: *telnet <IP adresa> <port>*

3.5.2 Pripojenie pomocou ssh

SSH predstavuje protokol, ktorý umožňuje vzdialené pripojenie pomocou tohoto protokolu. Používa štandardný port 22. Tak isto ako u telnetu, pre SSH platí to isté, je potrebné ho povoliť v IP -> Services. SSH na rozdiel od telnetu je ale šifrovaný a zabezpečený protokol. SSH predstavuje bezpečnú verziu telnetu. Je možné si zabezpečiť SSH prístup na bezpečnejší, a to tak, že sa budú porovnávať verejný a súkromný kľúč certifikátu TKIP. V7stupom pripojenia SSH na mikrotik je na obrázku 3.4.



Obr. 3.4: Prihlásenie na mikrotik pomocou príkazu SSH

4 PROGRAMOVACÍ JAZYK PYTHON

Python je interpretovaný, interaktívny, objektovo-orientovaný a vysoko-úrovňový programovací jazyk.Jazyk Python bol vytvorený pánom Guido van Rossum v Wiskundskom centre informatiky v 80-tych rokoch.
Medzi jeho vlastnosti patrí:

- dynamické typovanie
- konzolové aplikácie
- objektové aplikácie
- všetko v pythone je objekt
- jednoduchá syntax
- biele znaky sú súčasťou jazyka
- dynamické typy premenných
- široká škála knižníc
- dokumentácia na vysokej úrovni
- používaný na webové aplikácie, strojové učenie, teórie zložitosti,...

Verzie jazyku Python:

- Python verzia 2
- Python verzia 3

4.1 Python 2

Vlastnosti jazyku Python 2[10]

- automatická spáva pamäti (garbage collector)
- podporuje viac vstupných paradigiem
- Volanie niektorých príkazov je odlišné od Python 3
- referenčný interpret sa nazýva CPython a spravuje ho organizácia Python Software Foundation
- Súčasne sa používa Python vo verzii 2.7.2

4.2 Python 3

Vlastnosti jazyku Python 3[15]

- V niekorých častiach syntaxe v porovnaní s jazykom Python 2je trošku odlišná (napr. príkaz print,...)
- Od verzie 3.6 má premenná typu slovník interné zachovávané poradie vkladaných prvkov
- Pridanie anotácií cez metatriedy

- deklarácia nelokálnej premennej vonku z funkcie
- Slová typu True, False a None sú rezervované slová
- Mnoho vlastností ma rovnakých ako Python 2
- Miesto <> sa voverzii 3 používa relačný operátor !=
- od Júla 2018 by mala výjsť verzia Python 3.7 s ďalšími novinkami

4.3 Prostredia na programovanie v jazyku Python

Na realizovanie python programu je nutnosť mať nainštalovaný softvér na kompiláciu softvéru napísaného v jazyku Python. Na tieto účely slúži tzv. intergrated developement envinroment (IDE). Ecistuje niekoľko ciet aj mimo IDE ako spustiť kód napísaný v jazyku python.

- Napísanie kódu napr. v textovom editore typu nano, vim, gedit ale aj windows riešenie ako napr. notepad
- nainštalovaný python kompilátor
- spustenie programu príkazom python <názov.py>

4.4 Pycharm

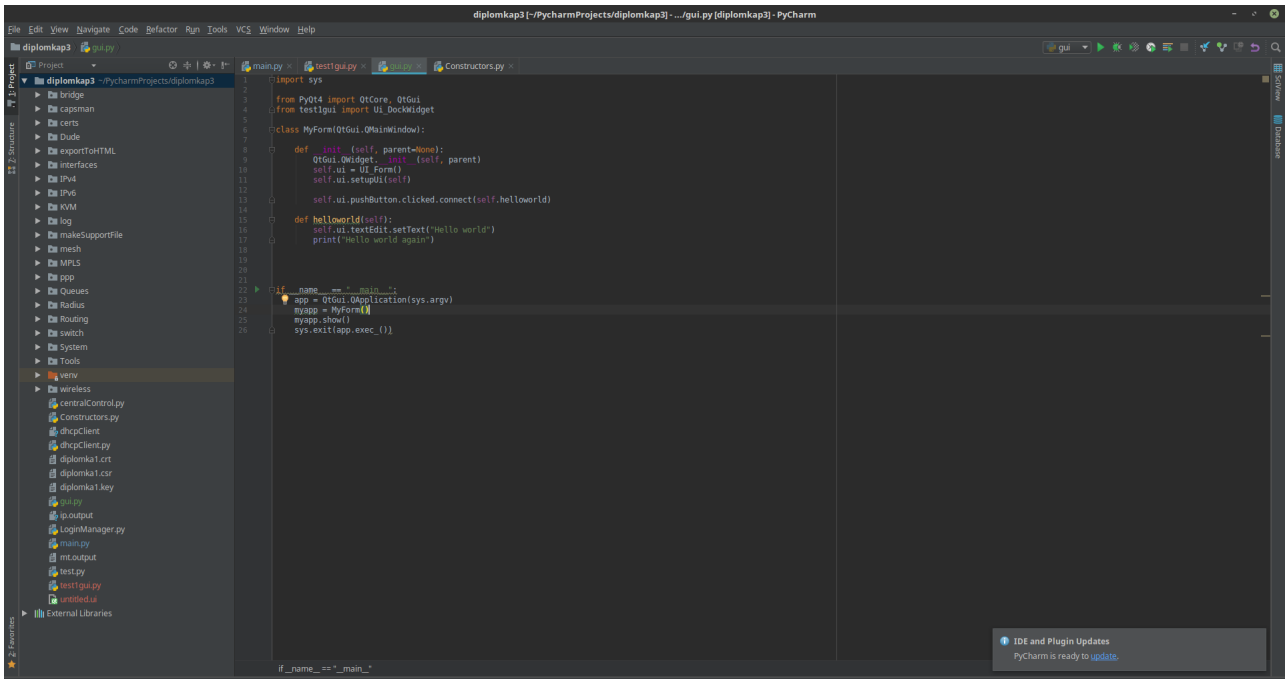
Pycharm predsatvuje IDE na pokročilé aplikácie napísané v jazyku Python. Exustuje v dcoch verziách:

- Pycharm Community Edition - voľne dostupné, nelicencované, neobsahuje niektoré doplnky professional verzie
- Pycharm Professional Edition - licencované, voľne dostupné na 30 dní, licencované, plný prístup ku všetkým doplnkom

Na nainštalovanie pycharm ľubovoľnej verzie je potreba:

- Stiahnutie bin respektíve exe súboru inštalátoru
- Napojenie na pracovný adresár projektu
- Napojenie na tzv. envinroment, to je použitie cesty k volaniu príkazu python respektíve python3

Po spustení programu Pycharm sa vytvorí projekt, kde po jeho inicializácii nájdeme podobný výstup.



Obr. 4.1: Rozhranie IDE Pycharm Professional Edition

Jednou z najväčších výhod je generovanie UML diagramov z kódu.

5 POUŽITÉ KNIŽNICE V DIPLOMOVEJ PRÁCI

5.1 OS.SYSTEM

Modul operačný systém (OS) [5] je zahrnutý v rámci štandardných knižnícv jazyku python. Jeho hlavnou výhodou je použitie príkazov operačného systému na ktorom beží python.Najčastejšie sa volá príkazom os.system('príkaz'). Použitie modulu aplikujem v rámci diplomovej práce. Používa sa hlavne pri vyhľadávaní a pripájaní sa na mikrotiky pomocou protokolu mactelnet. Jeho výstup môžeme aplikovať na:

- Štandardný výstup do konzoly
- Výstup do osobitného súboru, ktorý sa potom ďalej spracuje

V rámci práce som tento modul použil v rámci knižnice loginManager, kde v metódach pre mactelnet vyhľadáva mikrotiky za pomoci protokolu mactelnet a ukladá to do textového súboru mt.output.
Nie je potreba inštalácie modulu OS, pretože je zahrnutý v rámci štandardných knižníc jazyku python. V kóde vidíme ukážku metódy,ktorá vylistuje zoznam mikrotik zariadení za pomoci funkcie mactelnet a jej návratovou hodnotou je list zoznamu mikrotik zariadení

Listing 1: Použitie knižnice os.system

```
def listMikrotikDevices(self):
 deviceList = []
 loadMacAddress = False
 os.system("mactelnet -l -t 20 2>&1 > mt.output")
 with open( "mt.output", "r" ) as file:
 for line in file:
  if loadMacAddress:
   macAddress = line.split( )[1]
   deviceList.append( macAddress )
  else:
   header = line.split( )
    if len( header ) > 1:
     if "IP" in header[0] and "MAC-Address"in header[1]:
      loadMacAddress = True
 return deviceList
```

5.2 Telnetlib

Telnetlib [13] obdobne ako OS knižnica je vstavaná knižnicaprogramovacieho jazyku python. Knižnice *telnetlib* implementuje protokol telnet do pythonu, definovaného referenčným modelom RFC 854. V rámci definície modulu telnetlib sa používa hlavičkový súbor *telnet.h* s odstráneným záhlavým obsahujúcim *TELOPT_*.
Modul telnetlib predstavuje jednoduchého telnet klienta pripájajúceho sa na telnet server. Na vytvorenie spojenia je potreba nasledujúcich krokov:

- vytvorenie objektu telnetlib s parametrami *host*,čo predstavuje IP adresu telnet serveru, *port*, štandardne 23, nepovinným parametrom je *timeout*.
- Je nutné otvoriť spojenie metódou *open()*
- ďalej metódami readuntil() a write() vyžadujeme očakávaný výstup a odoslanie dát na server (príkazov)
- potom ukončíme spojenie metódou close()

Medzi najpoužívanejšie metódy v rámci diplomovej práce sú použité:

- *Telnet()*
- *readuntil()* -očakávanie výstupu serveru
- *write()* - zápis príkazov
- *sleep()* - doba trvania odoslania príkazu v sekundách
- *close()* - ukončenie spojenia

V rámci výstupu kódu je ukážka metódy na pripojenie na telnet server pomocou knižnice telnetlib.

Listing 2: Použitie knižnice telnetlib

```
def loginTelnet(self, password, login="admin"):
 import telnetlib
 central = centralControl(login, password)
 server_list = central.listMikrotikDevices()
 print(server_list)
 for server in server_list:
 try:
  telnetcon = telnetlib.Telnet( host=server, port=23 )
  telnetcon.read_until( b"Login: " )
  telnetcon.write( login.encode( ) + "\n" )
  telnetcon.read_until( b"Password: " )
  telnetcon.write( password.encode( ) + b"\n" )
  time.sleep( 10 )
  telnetcon.close( )
 except:
  print( "Cannot connect to router via telnet" )
```

5.3 Pxssh a pxexpect

Moduly pexpect[7] a jeho submodul pxssh[9] sú knižnice, ktoré slúžia na vyžadovanie určitého výstupu zariadenia, na to slúži knižnica *pexpect* a na pripojenie sa na server z pythonupomocou protokolu SSH slúži knižnica *pxssh*
Pexpect je čistá python modulna kontrolu a riadenie aplikácií. Pozostáva z dvoch krokov:

- Vyžadovanie výstupu
- Odoslanie požadovaného výstupu

Pexpect môže byť použitý na interaktívne aplikácie, ktoré používajú protokoly SSH, File transfer protocol (FTP), telnet,atď. Pre implementáciu pexpect nie je potreba importovania knižníc z jazyka C na skompilovanie do jadra. Pracujú na všetkých platformách, a to v podobe štandardného vstupu a výstupu v príkazovom riadku operačného systému, či už serveru ale aj klienta. Pexpect je jednoduchýna implementáciu.
Pxssh predtavuje submodul modulu pexpect. Na zavolanie submodulu pxssh je nutné prednostne zavolať metódu *spawn()*. Po vytvorení spojenia metódou *spawn()* je nutné použiť metódy *login()*, *spawn()*a *logout()*.

5.3.1 Inštalácia pexpect

Pexpect je súčasťou sady nástrojov Pypi. Aktuálna verzia modulu pexpect je verzia 4.4. Požiadavky na softvér:

- Python vo verzii 2.7 alebo 3.3 a vyššie
- pre windows je potreba inštalácie modulu POSIX pre jeho funkčnosť

Na nainštalovanie pexpect [8] na linuxe,sa v príkazovom riadku zadá:

Listing 3: Inštalácia Pexpect

```
pip install pexpect
pip3 install pexpect
```

Pre inštaláciu na operačnom systéme Windows je potreba maťnainštalovaný program python prípadne python3 externe, pretože nie je štandardne zahrnutý v balíčkoch operačného systému.
Modul pexpect zahŕňa modul pxssh a nie je potreba ho potom extra inštalovať.
Nižšie vidíme ukážku použiia kombinácie modulov pexpect a pxssh na pripojenie na mikrotik.

Listing 4: Inštalácia Pexpect

```
def loginSSH(self, server,login, password):
 from pexpect import pxssh, spawn, expect
 import getpass
 try:
  connect = pxssh.pxssh( )
  server = '172.16.49.2'
  login = 'admin'
  password = 'admin'
  port = 22
  connect.login( server, login, password )
  commands = pxssh.spawn( )
  time.sleep( 10 )
 except pxssh.ExceptionPxssh as e:
  print( "Error" )
  print( str( e ) )
```

5.4 TikApy

Na správu mikrotik smerovačov je potreba implentovať do pythonu modul tikapy[16]. Modul tikapy funguje voverzii pythonu 3 a vyššie. Podobne ako bolo spomenuté v kapitole 2.2, API pracuje na základe tzv. "slov". Slová predstavujú jednotlivé príkazy na mikrotiku. Tieto príkazy budú popísané v ďalších kapitolách diplomovej práce. Modul tikapy ako celkovo mikrotik API komunikuje nezabezpečene na porte 8728 a pomocou API-SSL na porte 8729. Na pripojenie sa na mikrotik pomocou modulu tikapy, ktorýmá v sebe zahrnutých mnoho knižníc na komunikáciu s mikrotikom. Medzi najčastejšie patria:

- vytvorenie objektu TikapyClient prípadne TikapySSLClient
- *login()* - prihlásenie sa na mikrotik pomocou API
- *talk()* - odosielanie príkazov na mikrotik
- *close()* - ukončenie spojenia

Na nainštalovanie tikapy do pythonu je potreba to nainštalovať nasledovne, v príkazovom riadku zadíme príkaz:

Listing 5: Inštalácia tikapy

```
pip install tikapy
```

V rámci vytvorenia objektu sú dve možnosti:

- vytvorenie klienta a to buď SSL klienta prípadne štandardného klienta TikapyCLient(adresa,port), pre SSL klienta to je port 8729, pre štandardného klienta port 8728
- vytvorenie metódy login(užívateľ,heslo)
- odoslanie príkazu metódou *talk(['príkaz'])*

Nižsie vidíme príklad metódy spoločne s konštruktorom triedy na pridanie IP adresy s príkladom použitia API slov.

Listing 6: Príklad použitia tikapy

```
from tikapy import TikapyClient
from tikapy import TikapySslClient
class Addresses:
    def __init__(self,address,username,password):
        self.client = TikapySSLClient( address, 8729 )
        self.client.login( username,password)
    def addAddress(self,address,interface):
     """
␣␣␣␣␣Method␣will␣add␣address
␣␣␣␣␣:param␣address:
␣␣␣␣␣:param␣interface:
␣␣␣␣␣:return:
␣␣␣␣␣"""
     ipv4 = self.client.talk(['/ip/address/add',
     '=address='+address,'=interface='+interface])
     return ipv4
```

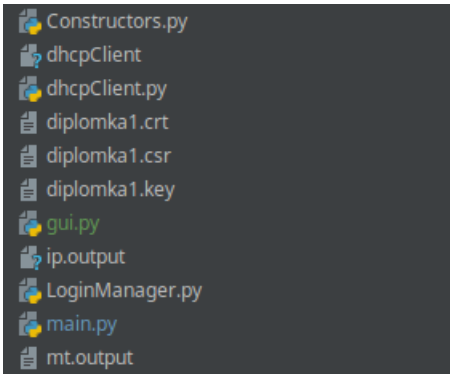
6 KONZOLOVÁ ČASŤ APLIKÁCIE NA SPRÁVU MIKROTIKOV

V tejto kapitole si popíšeme fungovanie naprogramovanej aplikácie. Celkovo je konzolová časť aplikácie napísaná za pomoci knižnice tikapy popísanej v kapitole 5.4. Kapitola bude rozdelená do niekoľkých častí:

- časť 1: popis naprogramovanej časti pre vyhľadávanie mikrotikov, pripojenie sa na mikrotik cez python pomocou protokolov telnet, SSH, mactelnet a napojenie na metódy
- časť 2: Popis infraštruktúry backendu - zložky, ich vysvetlenie, zoznam súborov na konfiguráciu mikrotku, vysvetenie rozdelenia, vysvetlenie tried, metód daných tried a volanie funkcií
- časť 3 - prtidanie tabuliek jednotlivých tried a ich metód v každej zložke, krátka sumarizácia, ich niektoré vybrané UML diagramy, ostatné budú zahrnuté v prílohe

6.1 Popis naprogramovanej časti prihlasovania na mikrotik

V tejto časti si zobrazíme rozbor časti prihlasovania na mikrotik a základné funnkcie. Toto je riadené v rámci projektu nazvaného *diplomkap3* v ktorom je súbor *loginManager.py*. V rámci login managera tu nachádzame ďalšie súbory, ktoré sú zobrazené na obrázku 6.1.



Obr. 6.1: Zoznam základných konfiguračných súborov

6.1.1 Súbor centralControl

V súbore centrolControl sa popisuje spôsob hromadnej obsluhy mikrotikov na základe protokolu mactelnet. Pozostáva z metód:

- *konštruktor* - pozostáva zužívateľských mien a hesiel, heslá v premennej credentials sú uložené ako slovník v podobe IP adresa: heslo
- *listMikrotikDevices()* - metóda vráti zoznam MAC a IP adries nájdených mikrotikov, uloží ich do súboru, a finálny výstup predstavuje list MAC adries
- *addCredentials()* - metóda pridíva heslo k užívateľskému účtu do slovníku, štandardnéužívateľské meno sa používa admin, ale tiež sa môže použiť aj iné užívateľské meno pri volaní metódy
- *loginSSH()*- metóda je použitá na hromadné prihlásenie pomoocu protokolu SSH na mikrotiky, používa sa tu pritom knižnica pexpect a jej subkižnica pxssh, jej vstupné parametre sp IP adresa serveru, užívateľské meno a heslo

Listing 7: Konštruktor súboru

```
def __init__(self, login):
        self.username = login
        self.credentials = {
            "192.168.1.1": "admin",
            "192.168.2.1": ""
        }
```

Listing 8: Meóda zobrazenia mikrotikov

```
 def listMikrotikDevices(self):
   deviceList = []
   loadAddress = False
   os.system("mactelnet -l -t 20 2>&1 > mt.output")
    with open( "mt.output", "r" ) as file:
     for line in file:
      if loadAddress:
         address = line.split( )[0]
         deviceList.append( address )
      else:
         header = line.split( )
         if len( header ) > 1:
            if "IP" in header[0]:
              loadAddress = True
         for i in deviceList:
           print(i)
  return deviceList
```

Listing 9: Metóda pridania užívateľských mien a hesiel

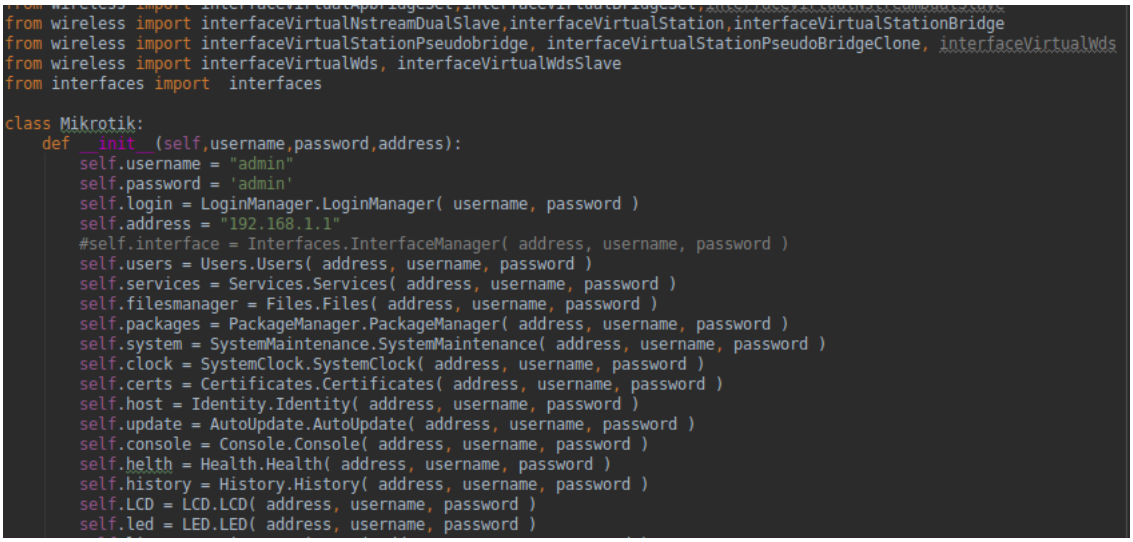
```
 def addCredentials(self, login="admin"):
   server_list = self.listMikrotikDevices()
   print( server_list )
   for server in server_list:
   try:
    password = self.credentials[server]
    except KeyError:
    password = input( "Please eneter the
    password for " + server + ":" )
    self.credentials[server] = password
    return server_list
```

Listing 10: Metóda hromadného prihlásenia pomocou protokolu SSH

```
def loginSSH(self, server,login, password):
 from pexpect import pxssh, spawn, expect
 import getpass
 for server in self.credentials:
 try:
  connect = pxssh.pxssh( )
  server = self.credentials
  login = 'admin'
  password = self.credentials[server]
  port = 22
  connect.login( server, login, password )
  commands = pxssh.spawn( )
  time.sleep( 10 )
 except pxssh.ExceptionPxssh as e:
  print( "Error" )
  print( str( e ) )
```

6.1.2 Súbor Constructors

Súbor predstavuje zoznam konštruktorov pre konkrétne naprogramované API moduly pomocou knižnice tikapy. V úvode konštruktoru sú popísané importy jednotlivých modulov a submodulov pre konfiguráciu mikrotiku za pomoci API.
Následne je vytvorená trieda Mikrotik, ktorá zahŕňa všetky konštruktory spoločne s ich vstupnými parametrami, ktoré sú adresa,užívateľské meno a heslo.



Obr. 6.2: Ukážka konštruktorov projektu

6.1.3 Súbor dhcpClient

V tomto súbore sa nachádza základná konfigurácia mikrotiku po pripojení naň. Obsahuje triedu basicConfig, ktorá pozostáva z dvoch metód.
V konštruktore sa nastaví rozhranie, na ktorom sa má adresa nastaviť, IP adresa-/subnet a MAC adresa na pripojenie na mikrotik pomocu protokolu mactelnet.

Listing 11: Trieda basicConfig

```
class basicCOnfig:
    def __init__(self,interface,mac,ip):
        self.interface = interface
        self.mac = mac
        self.ip = ip
```

Prvá metóda *dhcp()*, ktorej vstupné parametre sú užívateľské meno a heslo. Pozostáva z prihlásenia na mikrotik, a nastavenia Dynamic Host Client Protocol (DHCP) klienta na rozhraní, ktoré sa definuje pri volaní objektu v rámci konštruktoru.

Listing 12: Metóda dhcp

```
 def dhcp(self,username,password):
        child =pexpect.spawn('mactelnet␣'+self.mac)
        child.expect('Username:')
        child.sendline(username)
        child.expect('Password:')
        child.sendline(password)
        child.sendline('\r')
        try:
            child.expect('>␣')
            child.sendline('ip␣dhcp-client␣add
␣␣␣␣␣␣␣␣␣␣␣␣interface='+self.interface+"\r")
            child.expect('>␣')
            child.close()
        except:
            print("error")
            child.close()
        time.sleep(1)
```

Druhá metóda *setAddress()*, ktorá bere ako vstupné parametre užívateľské meno a heslo nastaví statickú IP adresu na rozhraní definovanom v rámci konštruktoru.

Listing 13: Metóda setAddress

```
def setAddress(self,username,password):
    child = pexpect.spawn( 'mactelnet␣'+
    self.mac )
    child.expect( 'Username:' )
    child.sendline( username )
    child.expect( 'Password:' )
    child.sendline( password )
    child.sendline( '\r' )
    try:
        child.expect( '>␣' )
        child.sendline( 'ip␣address␣add␣address='
        +self.ip + "␣interface="
        +self.interface+"\r" )
        child.expect( '>␣' )
        child.close()
    except:
        print( "error" )
        child.close()
    time.sleep( 1 )
```

6.1.4 Súbor LoginManager

Súbor LoginManager pozostáva z niekoľkých metód, tieto metódy majú podobnú štruktúru ako súbor centralConrol popisujúci v kapitole6.1.1.
Ako prvá popísaná časť je konštruktor, ktorý prijíma vstupné parametre užívateľské meno a heslo.

Listing 14: Konštruktor súboru

```
def __init__(self, login,password):
    self.username = login
    self.pwd = password
```

Druhá metóda je metóda *loginTelnet()*,v rámci tejto metódy sa rieši prihlásenie na mikrotik pomocou protokolu telnet za použitia knižnice telnetlib. Vo vstupe metódy sa definuje premenná *serverꝉist.TátopremennájenaplnenáIPadresamimikrotikovvrámcisúb*

Ďalej sa tu nachádza metóda loginSSH(), táto metóda pracujúca podobne ako metoda loginTelnet() pracuje na základe protokolu SSH, na vstupe má server IP adresu, užívateľské meno a heslo.

Listing 16: Metóda loginSSH

```
def loginSSH(self, server,login, password):
    from pexpect import pxssh, spawn, expect
    import getpass
    try:
        connect = pxssh.pxssh( )
        server = '172.16.49.2'
        login = 'admin'
        password = 'admin'
        port = 22
        connect.login( server, login,
        password )
        commands = pxssh.spawn( )
        time.sleep( 10 )
    except pxssh.ExceptionPxssh as e:
        print( "Error" )
        print( str( e ) )
```

Ďalšou metódou je metóda na vylistovanie všetkýchmikrotikov, táto metóda je bez vstupného parametru. Ako výstup je súbor mikrtik.output naplnený MAC adresami mikrotikov.

Listing 17: Metóda listMikrotikDevices

```
    def listMikrotikDevices(self):
        deviceList = []
        loadMacAddress = False
        os.system("mactelnet -l -t 20
        2>&1 > mt.output")
        with open( "mt.output", "r" )
        as file:
            for line in file:
                if loadMacAddress:
                   macAddress = line.split( )[1]
                   deviceList.append( macAddress )
                else:
                    header = line.split( )
                    if len( header ) > 1:
                        if "IP" in header[0]
                        and "MAC-Address"
                        in header[1]:
                            loadMacAddress = True
        return deviceList
```

Poslednou metódou je metóda *mactelnetLoginToSingleDevice()*, vďaka ktorej sa pripája pomocou protokolu mactelnet na jedno mikrotik zariadenie pomocou macadresy získanej z výstupu metódy *listMikrotikDevices() mikrotik.output*.

Listing 18: Metóda mactelnetLoginToSingleDevice

```
def mactelnetLoginToSingleDevice(self, username,
password, address=None):
    deviceList = self.listMikrotikDevices()
    print( deviceList )
    if address:
        print('mactelnet␣{}␣-u␣{}␣-p␣{}'.format
        ( address, username, password ))
        os.system( 'mactelnet␣{}␣-u␣{}␣-p␣{}'.format
        ( address, username, password ) )
    elif deviceList:
        print( 'mactelnet␣{}␣-u␣{}␣-p␣{}'.format
        ( deviceList[0], username, password ) )
        os.system( 'mactelnet␣{}␣-u␣{}␣-p␣{}'.format
        ( deviceList[0], username, password ) )
    else:
        print("No␣device␣was␣found")
```

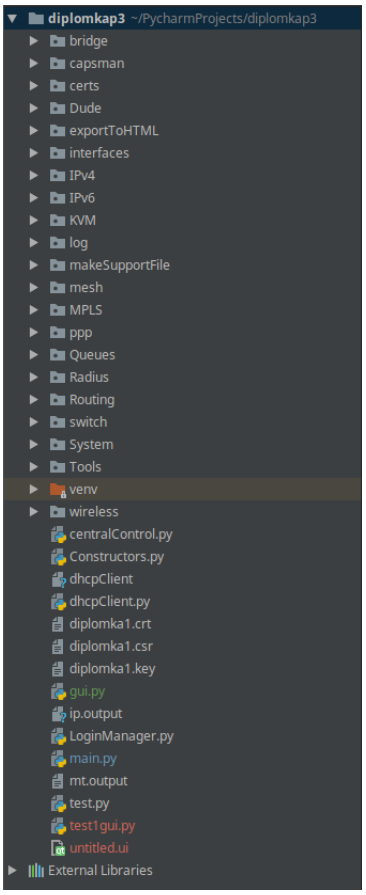
6.2 Rozbor hlavnej časti backendu

V rámci hlavnej konfiguračnej časti diplomovej práce, pre konfiguráciu backendu mikrotiku za pomoci porgramovacieho jazyka python som projekt rozdelil do niekoľkých častí:

- **bridge** - táto časť obsahuje prvky konfiguácie, pridania, odstránenia, zapnutia, vypnutia možnosti bridgu na mikrotiku, konfigurácia existujúceho bridgu, zobrazenie zoznamu bridgov
- **capsman** - táto časť obsahuje konfiguráciu hromadnej obsluhy mikrotik úrístupových bodov a WiFi, profily, bezpečnosť, konfiguácie, povolené rýchlosti, zobrazenie zoznamu pripojených prvkov a ďalšie funkcie
- **certs** - obsahuje certifikáty na pripojenie sa na mikrotik pomocou protokolu api-ssl
- **Dude** - obsahuje popis konfigurácie ako nastaviť nástroj Dude klienta, ako nakonfigurovať Dude na vzdialený monitoring na Dude serveri, taktiež Dude server, a ďalšie možnosti
- **exportToHtml** - časť predstavuje generovanie súboru na analýzu v podobe webovej stránky

- **interfaces** - časť predstavuje konfiguráciu rozhraní na mikrotiku, tieto časti sú tiež popísané aj v iných zložkách ako napr. bridge. 4asť popisuje pridanie, odstránenie, zapnutie, vypnutie, konfiguráciu existujúcich rozhraní.
- **IPv4** - rozsiahla časť, obsahuje konfiguráciu IP adries, firewallu, monitoringu, smerovania a ďalších nástrojov spadajúcich pod IP zložku na mikrotiku.
- **IPv6** - pre zložku IPv6 platí to isté čo pre zložku IPv4, ale platí pre konfiguráciu na základe IPv6 adresného rozsahu
- **KVM** - sekcia bude popisovať možnosti virtualizácie mikrotiku.
- **log** - sekcia bude popisovať analýzu a konfiguráciu logu zariadenia
- **makeSupportFile** - seckia bude popisovať vytvorenie súboru potrebného pre analýzu na mikrotik podpore
- **mesh** - sekcia popisuje konfiguráciu tzv. mesh technológie, technológii podobne ako v rámci časti bridge
- **MPLS** - sekcia bude popisovať možnosti konfiurácie Multi Protocol Label Switching (MPLS), jej pridanie, odstránenie ,zapnutie, vypnutie, modifikácie a ďalšie funkcie.
- **PPP** - sekcia bude popisovať konfiguráciu Point to Point Protocol (PPP) a ďalších možností Virtual Private Network (VPN) konfigurácie.
- **Queues** - sekcia budep popisovať konfiguráciu sieťových front, možnosti front, typy front a ďalšie funkcie
- **Radius** - sekcia bude popisovať nastavenie funkcie Radius - autentizačnej služby užívateľov , jeho modifikáciu, konfiguráciu a ďalšie funkcie.
- **Routing** - sekcia bude popisovať možnosti dynamického smerovania, statické smerovanie bude popísané v rámci časti IPv4, dynamické smerovacie protokoly, ich konfigurácie, a ďalšie možnosti.
- **Switch** - sekcia bude popisovať konfiguráciu prepínača, niektoré mikrotiky sú typu SwitchOS a sú štandardne prepínač. Kofiguuráciu portov, trunkov, a ďalších funkcií.
- **System** - sekcia bude popisovať časť konfigurácie systémových nástrojov, ich funkcií a konfigurácie, a ďalších funkcií.
- **Tools** - sekcia bude popisovať konfiguráciu mikrotik nástroj, a však nie všetky bolo možné odsimulovať v rámci konzolvej časti aplikácie, ich konfiguráciu, spustenie, riadenie a ďalšie funkcie.
- **Wireless** - sekcia bude obsahovať konfiguráciu bezdrátového rozhrania, moduly, módy, konfiguráciu, nastavenie, a ďalšie funkcie
- **konfiguračné súbory mimo zložiek** - sekcia popísaná v kapitole 6.1, popisuje súbory na základnú konfiuráciu mikrotiku, nastavenie základnej konfigurácie.

Ukážka súborovej štruktúry je zobrazená na obrázku 6.3:



Obr. 6.3: Štruktúra projektu konzolovej časti projektu

7 HLAVNÁ ČASŤ BACKENDU

Cieľom kapitoly je detailný popis backend časti aplikácie na správu mikrotikov. V jendotlivých podkapitolách bude popísaná každá zložka projektu diplomkap3.

7.1 Zložka bridge

LITERATÚRA

[1] *Manual:API* [online]. 2014, [cit. 24. 03. 2018]. Dostupné z URL: <https://wiki.mikrotik.com/wiki/Manual:API>

[2] *How to Install Configure Easy-RSA* [online]. 2013, [cit. 24. 03. 2018]. Dostupné z URL: <https://docs.bigchaindb.com/projects/server/en/latest/production-deployment-template/easy-rsa.html>

[3] *MAC Level Access (Telnet and Winbox)* [online]. 2007, [cit. 26. 03. 2018]. Dostupné z URL: <https://mikrotik.com/testdocs/ros/2.9/tools/mactelnet.php>

[4] Mitchell Anicas *OpenSSL Essentials: Working with SSL Certificates, Private Keys and CSRs* [online]. 2012, [cit. 24. 03. 2018]. Dostupné z URL: <https://www.digitalocean.com/community/tutorials/openssl-essentials-working-with-ssl-certificates-private-keys-and-csrs>

[5] *os — Miscellaneous operating system interfaces* [online]. 2012, [cit. 04. 04. 2018]. Dostupné z URL: <https://docs.python.org/2/library/os.html>

[6] *CISCO: Open Shortest Path First (OSPF)* [online]. 2009, [cit. 09. 11. 2014]. Dostupné z URL: <http://www.cisco.com/c/en/us/products/ios-nx-os-software/open-shortest-path-first-ospf/index.html>.

[7] *Pexpect version 4.4* [online]. 2018, [cit. 02. 04. 2018]. Dostupné z URL: <https://pexpect.readthedocs.io/en/stable/>.

[8] *Installation* [online]. 2018, [cit. 02. 04. 2018]. Dostupné z URL: <https://pexpect.readthedocs.io/en/stable/install.html>.

[9] *pxssh (version 2.3)* [online]. 2018, [cit. 02. 04. 2018]. Dostupné z URL: <http://pexpect.sourceforge.net/pxssh.html>.

[10] *Python 2.7.2 Release* [online]. 2018, [cit. 02. 04. 2018]. Dostupné z URL: <https://www.python.org/download/releases/2.7.2/>.

[11] Tatu Ylonen *SSH PROTOCOL)* [online]. 2017, [cit. 26. 03. 2018]. Dostupné z URL: <https://www.ssh.com/ssh/protocol/>

[12] *What is telnet?)* [online]. 2018, [cit. 26. 03. 2018]. Dostupné z URL: <https://kb.iu.edu/d/aayd>

[13] *telnetlib — Telnet client)* [online]. 2018, [cit. 04. 04. 2018]. Dostupné z URL: <https://docs.python.org/3.1/library/telnetlib.html>

[14] *Manual: Webfig* [online]. 2018, [cit. 26. 03. 2018]. Dostupné z URL: <https://wiki.mikrotik.com/wiki/Manual:Webfig>

[15] *What's New In Python 3.0* [online]. 2018, [cit. 02. 04. 2018]. Dostupné z URL: <https://docs.python.org/3.0/whatsnew/3.0.html>.

[16] *tikapy* [online]. 2018, [cit. 02. 04. 2018]. Dostupné z URL: <https://github.com/vshn/tikapy/blob/master/README.rst>.

[17] *Manual: Winbox* [online]. 2018, [cit. 26. 03. 2018]. Dostupné z URL: <https://wiki.mikrotik.com/wiki/Manual:Winbox>

[18] *Install the Certification Authority* [online]. 2017, [cit. 24. 03. 2018]. Dostupné z URL: <https://docs.microsoft.com/en-us/windows-server/networking/core-network-guide/cncg/server-certs/install-the-certification-authority>

ZOZNAM SYMBOLOV, VELIČÍN A SKRATIEK

AP Prístupový bod

API Application programable interface

API-SSL Application programable interface Secure Socket Layer

FTP File Transfer Protocol

GUI Graphical User Interface

IDE Integrated Developement Envinroment

IP Internet Protocol

IPSEC Internet Protocol Security

MAC macintosh

MAC Media Access Control

MPLS Multi Protocol Label Swiching

OS Operačný systém

PPP Point to Point Protocol

SSL Secure Socket Layer

TLS Transport Layer Security

UML Unified Modeling Language

VPN Virtual Private Network

ZOZNAM PRÍLOH