FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Nástroj pro hromadnou správu síťových zařízení
Student:	Lukáš Merta
Vedoucí:	Ing. Jiří Mlejnek
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce zimního semestru 2018/19

Pokyny pro vypracování

Seznamte se s existujícími nástroji pro hromadnou správu síťových zařízení využitelných poskytovatelem připojení.
Analyzujte požadavky na aplikaci, která umožní přehledně prohlížet a filtrovat jednotlivá nastavení z více zařízení současně a hromadně tato nastavení měnit. Zaměřte se především na oblast správy uživatelů a konfiguraci síťového nastavení.
Na základě analýzy proveďte návrh a implementaci této aplikace. Rozsah realizovaných požadavků konzultujte s vedoucím práce.
Aplikaci nasaďte a otestujte její připravenost pro produkční nasazení.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 27. září 2016

Poděkování

Rád bych poděkoval vedoucímu své práce Ing. Jiřímu Mlejnkovi za jeho čas a rady a své rodině za podporu během celého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické 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. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“) a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Brně dne 8.1.2019

Odkaz na tuto práci

Merta, Lukáš. *Nástroj pro hromadnou správu síťových zařízení.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Tato práce se zabývá možností konfigurace více síťových prvků s využitím responzivního webového uživatelského rozhraní, které je použitelné na všech zařízeních. Zaměřuje se na správu uživatelských účtů a konfiguraci síťového rozhraní na prvcích se systémem RouterOS, k nimž se aplikace připojuje pomocí API nebo protokolu SSH.

Klíčová slova

webová aplikace, NMS, hromadná správa zařízení, konfigurace síťových prvků, bezpečnost, SNMP, SSH, API

Abstract

This thesis deals with configuration of multiple network devices and includes responsive web interface to target many devices. Its main focus is on user and interface management of RouterOS powered systems. To each device connects via API or SSH protocol.

Keywords

web application, NMS, bulk device management, network device configuration, security, SNMP, SSH, API

Obsah

Úvod

Práce se zabývá možností konfigurace jednotlivých zařízení se zaměřením na správu více zařízení najednou. Zaměřuje se na správu uživatelů, kteří mohou k zařízení přistupovat a nastavení síťových rozhraní.

Pro správu jediného síťového zařízení nejsou potřeba žádné speciální nástroje, každé zařízení má vlastní podporu konfigurace. Většina běžných routerů (směrovačů), tiskáren, telefonů a podobných zařízení s možností připojení do sítě určených pro použití v domácnostech a menších firmách, se konfiguruje přes webové rozhraní, uživatel si tak vystačí s webovým prohlížečem. Pouze některá z těchto zařízení mají možnost připojení přes příkazový řádek nebo jiné rozhraní pro propojení s jinými systémy (tzv. API). U profesionálních zařízení naopak webové rozhraní zcela chybí nebo je chápáno jako doplněk a primárním způsobem připojení k nim je příkazový řádek. Práce se dále nebude zabývat možností konfigurace přes webové rozhraní, protože není primárně určeno pro napojení jiných systémů.

Pokud se ovšem množina spravovaných zařízení rozrůstá, je stále obtížnější udržet si přehled o nastavení jednotlivých zařízení a každá změna týkající se více z nich je také časově náročná na provedení. Znamená totiž manuální připojení ke všem zařízením a provedení změny konfigurace. Velmi podobná situace je se sledováním nejrůznějších statistik mnoha zařízení a hledání vzájemných souvislostí.

Práce se zaměřuje na nástroje pro hromadnou správu (označovány zkratkou NMS z anglického spojení „network management system“), které usnadňují a zpříjemňují práci všem, kteří mají na starosti více než jedno zařízení. Umožňují jednoduše prohlížet nastavení a statistiky a spravovat více zařízení najednou bez nutnosti se přihlašovat ke každému zvlášť. Kromě toho také mnohdy zvládají monitorování hodnot, zobrazení historie (graf) a zasílání upozornění na definované události (například překročení limitní hodnoty nebo výpadek). Práce se soustředí na bezplatné nástroje (nejlépe s otevřeným kódem pro možnost vlastních úprav).

1 Aktuální stav

V současnosti musí správci více zařízení (nepoužívající žádný nástroj pro hromadnou správu) přistupovat k jednotlivým zařízením zvlášť. Pokud chtějí například přidat uživatelský účet s přístupovými právy k zařízení, musí se postupně připojit ke všem zařízením a u všech tento nový účet vytvořit. Tento postup je nejen zdlouhavý a nudný, ale také náchylný k chybám, může se lehce stát, že do některého zařízení se správce nepřipojí vůbec (buďto jen zapomene, nebo v daný okamžik není zařízení dostupné a musí se konfigurace nahrát později, až přístupné bude) nebo nastaví hodnotu odlišně.

V referenční síti autora práce je aktuálně spravována necelá tisícovka síťových prvků využívajících převážně systém RouterOS (1) od firmy MIKROTIKLS SIA (dále jen „Mikrotik“ – velmi často se chybně používá i pro označení systému, pod značkou Mikrotik firma prodává své produkty – síťové prvky; v textu bude výraz Mikrotik používán pro označení výrobce i značky). Současný systém jednotlivá zařízení monitoruje a do přístupových bodů (nikoliv však do koncových zařízení uživatelů sloužících pro jejich přístup k síti) umožňuje nahrát připravený skript, ale už neřeší, zda proběhlo spuštění skriptu správně (pokud navíc skončí provádění skriptu chybou, nelze do zařízení nahrát novější skript, systém se vždy pokouší nejprve spustit původní chybný skript – v takovém případě je potřeba ze všech zařízení daný chybný skript ručně smazat). Protože chybí návratová hodnota skriptu, nelze tímto způsobem ani zjistit hodnoty, které systém nesleduje prostřednictvím SNMP (zkratka vysvětlena dále v textu v samotné sekci SNMP na straně 10) – např. uživatelské účty vytvořené v zařízení tímto způsobem ani sledovat nelze.

Z předchozího odstavce je zřejmý důvod, proč bylo vybráno právě dané téma práce. Cílem je najít nástroj, který pomůže s otázkami typu:

- Jaká je nejstarší verze systému a na kterém zařízení?
- Jaké je zastoupení jednotlivých verzí?
- Které balíčky jsou povoleny na kterých zařízeních?
- Kteří uživatelé mohou k jakému zařízení přistupovat?
- Má určitý uživatel právo přistupovat ke všem zařízením?
- Nezůstal na některém zařízení výchozí účet pro správu bez hesla?

Z těchto otázek dále vyplynou některé požadavky na systém.

2 Analýza problému, definice požadavků

Obecně se jedná o aplikaci pro podporu správy sítě. Pro správnou a spolehlivou funkčnosti počítačové sítě je potřeba zajistit její evidenci (jaké zařízení se kde nachází), správu (jaká je funkce zařízení, které služby na ní běží, jejich závislosti a vzájemná propojení, jakou má aktuální konfiguraci), monitoring (sběr a analýza dat – využití jednotlivých částí sítě, datové toky, využití procesoru a spousta dalších metrik), odstraňování a řešení potíží a alerting (včasné upozornění na blížící se nebo nastalý problém – komu a jakým způsobem bude notifikace zaslána, případně jaká bude automatická reakce), a to vše s ohledem na bezpečnost (zabezpečení jednotlivých zařízení i částí sítě).

Hlavním cílem aplikace je zajištění přehledu o všech spravovaných zařízeních a možnost nejen jednoduše zobrazit požadovaný parametr napříč těmito zařízeními, ale také daný parametr změnit. Kromě tohoto ústředního cíle jsou od aplikace vyžadovány další funkčnosti.

2.1 Obecné požadavky

Na aplikaci jsou kladeny požadavky s ohledem na interní použití správci sítě (techniky), kteří jsou nejen v kanceláři, ale také se často pohybují v terénu. Proto musí být dostupná odkudkoliv pro jakékoliv zařízení.

1) bezplatná licence k užívání, nejlépe i otevřený kód (open source) pro možnost případných úprav
2) dostupná z vnější sítě (internetu)
3) webové rozhraní pro dostupnost na široké škále zařízení
4) přizpůsobené zobrazení i na menších obrazovkách mobilních telefonů
5) multiplatformní serverová část (tj. umožňující běh pod různými operačními systémy)
6) programové rozhraní (API (2)) pro propojení s jinými aplikacemi
7) zobrazování hodnot v reálném čase (pouze s minimální prodlevou nutnou na zpracování)
8) podpora zařízení se systémem RouterOS (konfigurace přes příkazovou řádku nebo API)
9) zabezpečení uložených přihlašovacích údajů k zařízením

Nad rámec těchto povinných požadavků (náměty na případná rozšíření)

10) unifikované rozhraní pro konfiguraci různých zařízení
11) podpora dalších systémů jiných výrobců
12) role jednotlivých uživatelů (aktuálně budou k aplikaci přistupovat pouze správci, kteří mají úplnou kontrolu nad celou sítí), přístupy k zařízením na základě uživatelů a (nebo) rolí
13) záznam komunikace se zařízením (např. prováděné příkazy)

2.2 Funkční požadavky

Aplikace umožní uživatelům připojení k zařízením a jejich konfiguraci. Na začátku je potřeba zadat jednotlivá zařízení včetně přístupových údajů k nim (aby se aplikace mohla připojit), poté zjistit a zobrazit aktuální konfiguraci a nastavovat novou. Vedle toho je ještě potřeba pracovat s uživateli, kteří k aplikaci přistupují. Jednotlivé požadavky jsou dále v kapitole podrobně popsány.

2.2.1 Správa přístupových údajů k zařízením

Pro komunikaci s jednotlivým zařízením je potřeba použít přístupové údaje (uživatelské jméno a heslo). Obecně mohou být využity i jiné možnosti – certifikát, identifikace na základě IP adresy a další, kterými se práce nezabývá, protože nejsou v referenční síti používány a zařízení nejsou pro jiné možnosti nakonfigurována. Jsou ale doplněny jako náměty pro další rozšíření dále v kapitole.

Protože přihlašovací údaje lze využít stejné pro připojení k více zařízením, je v aplikaci možné evidovat přístupové údaje hromadně a při přidávání zařízení může uživatel vybrat z předvyplněných nebo zadat nové. Aplikace nabídne rozhraní se seznamem těchto údajů s možností jejich vytvoření, úpravy nebo smazání (pouze v případě, že není údaj použit u některého zařízení).

Protože se jedná o citlivé údaje (hesla), tyto údaje je potřeba zabezpečit (možnostmi se práce dále zabývá při implementaci).

Přístupové údaje se skládají z položek

- uživatelské jméno
- heslo
- poznámka (pro případné rozeznání stejných uživatelských jmen)

2.2.2 Správa zařízení

Každý uživatel může přidat nové zařízení po vyplnění povinných údajů (IP adresa nebo název hostitele a port pro připojení, přihlašovací údaje – možnost vybrat ze seznamu předvyplněných), které jsou potřeba, aby se aplikace k danému zařízení mohla připojit a načíst z něj údaje. Po přidání nového zařízení se k němu aplikace pokusí připojit a načíst z něj údaje.

Název hostitele je ve výsledku vždy přeložen na IP adresu. O tuto část se již nestará aplikace, ale některá z nižších vrstev, nad níž běží – framework nebo operační systém).

Některé údaje lze ze zařízení pouze načíst (např. unikátní identifikátor zařízení, verze systému nebo aktuální doba běhu) a nelze je přímo měnit. Unikátní identifikátor zařízení je pevně spojen s daným zařízením (obdoba sériového čísla) a nelze jej nijak programově změnit (běžnými prostředky), stejně tak verzi systému nebo aktuální dobu běhu není možné změnit jednoduše tak, že se do zařízení zapíše jiná hodnota. Změna je možná pouze jiným způsobem, nepřímo – instalací novější verze systému, restart zařízení apod. Z pohledu aplikace tyto údaje nelze měnit.

Jiné údaje lze ze zařízení načíst, ale zároveň do něj zapsat údaj nový (např. IP adresu lze nejen načíst, ale i přímo zapsat).

Pro potřeby aplikace (do zařízení se nijak nezapisují ani z něj nenačítají) jsou evidovány

- uživatelský název (popis, komentář pro označení uživatelem a snazší orientaci)
- IP adresa (nebo název hostitele) a port pro přístup (port pouze v případě, že není standardní pro dané zařízení)
- přihlašovací údaje (uživatelské jméno a heslo)
- kategorie (dle použití daného zařízení – klientské, přístupové apod.) – slouží zejména pro hromadnou konfiguraci (např. servisní technik má vytvořen účet ve všech klientských zařízeních)

Ze zařízení lze pouze načíst a nelze přímo měnit

- unikátní identifikátor zařízení (pokud existuje, může jím být např. sériové číslo, slouží k odhalení případných duplicit v seznamu zařízení – pod 2 různými IP adresami/názvy hostitele se může nacházet totéž zařízení)
- název (identifikace) zařízení
- operační systém (nebo firmware) – název a verze
- aktuální doba běhu (od posledního startu) – ze zařízení se získá údaj o době běhu, ale do systému se uloží údaj posledního startu a doba běhu bude vypočítávána k aktuálnímu času

Údaje, které se ze zařízení načítají a lze je v aplikaci měnit (nebo nastavovat jejich část)

- aktuální datum a čas
- IP adresy
- rozhraní (podrobněji rozepsáno dále samostatně)
- balíčky (lze zakázat nebo povolit, např. balíček DHCP obsahuje DHCP klienta a server)
- uživatelé a jejich role
- role uživatelů (seznam názvů)

Rozhraní

K síťovému zařízení jsou evidována jeho jednotlivá rozhraní (automaticky z něj načtena, není možné je přidávat nebo mazat, lze jim pouze konfigurovat síťová nastavení – např. přidělit, upravit nebo odebrat IP adresu). O každém z nich se ukládá

- název
- typ (druh) rozhraní
    - fyzické nebo virtuální
    - kabelové (optické, metalické), bezdrátové (licencované nebo nelicencované pásmo)
- přiřazené IP adresy (vč. masky)

- stav spojení (připojeno nebo odpojeno)

2.2.3 Správa uživatelů zařízení

Uživatelé (včetně role) jsou načítání ze zařízení, v aplikaci lze uživatele přidávat, editovat a mazat.

Údaje o uživatelích zařízení

- uživatelské jméno
- heslo (ze zařízení nelze načíst, uloženo v zašifrované podobě pouze v případě, že bylo do zařízení zapsáno aplikací)
- role
- aktivní (udává, zda má uživatel povolen nebo zakázán přístup)
- poznámka

2.2.4 Správa uživatelů aplikace

K aplikaci se uživatelé přihlašují pomocí svého uživatelského jména a hesla. Pro případné zasílání notifikací nebo odkazu pro reset zapomenutého hesla je také evidován email. Každý uživatel může v aplikaci vytvářet nové uživatelské účty, zobrazit si jejich seznam, editovat současné a také jim ukončit platnost (mazání aplikace nepodporuje z důvodu zachování historie) k aktuálnímu nebo budoucímu datu.

Všichni uživatelé v aplikaci mají stejná práva, neřeší se role nebo přístupová práva k jednotlivým entitám nebo částem aplikace.

Údaje o uživatelích aplikace

- jméno
- příjmení
- uživatelské jméno (unikátní)
- email (unikátní)
- heslo (hashovaná podoba)
- konec platnosti účtu – kontrola pouze při přihlašování, ne při provádění akcí

2.2.5 Skupiny úloh, úlohy – hromadné akce nad skupinou zařízení

Po výběru skupiny zařízení s ní uživatel může provádět obdobné operace jako s jednotlivým zařízením. Systém uživateli nabídne možnost vytvoření tzv. skupiny úloh – souboru operací nad skupinou zařízení.

Typy úloh (jednotlivé akce se zařízením)

- Otestování dostupnosti zařízení (pomocí přístupu na TCP nebo UDP port, na němž protokol běží, nebo odezvy na ICMP echo (3))
- Otestování vyplněných přístupových údajů k zařízení
- Načtení (aktualizace) údajů (všech evidovaných – načítaných) ze zařízení

- Vytvoření, smazání uživatele v zařízeních
- Přidání, odebrání, změna IP adresy k rozhraní
- Spuštění uživatelem napsaného příkazu (pro příkazovou řádku)

Úloha pro jednotlivé zařízení

Uživatel může spustit (vytvořit) novou úlohu, zastavit ji nebo změnit interval opakování. U dokončené úlohy si může prohlédnout její výsledek (zda proběhla úspěšně nebo s nějakou chybou).

Údaje o úloze

- Uživatel, který ji spustil
- Datum a čas spuštění
- Datum a čas dokončení
- Celkový výsledek (úspěch, neúspěch – důvod)
- Maximální a aktuální počet opakování a interval mezi jednotlivými následnými pokusy
- Stav (připravená, v běhu, dokončená, přerušená)

Příkaz

Jednotlivé provedení úlohy znamená vykonání příkazu na zařízení. Příkaz může být proveden přes příkazovou řádku, přes API nebo jiným způsobem, který zařízení podporuje. Vykonávání příkazu je jednorázové a výsledkem může být úspěšné vykonání nebo neúspěšné (chyba při provádění v zařízení nebo chyba při připojování k zařízení – nedostupnost nebo vypršení časového limitu).

Údaje evidované u příkazu

- Datum a čas spuštění
- Datum a čas dokončení
- Výsledek (úspěch, neúspěch)
- Druh komunikace

Soubor úloh

Pokud v jeden okamžik spouští uživatel stejnou úlohu pro více zařízení, jedná se o jejich soubor. Jednotlivé úlohy pak probíhají samostatné, v souboru úloh lze sledovat stav dílčích úloh.

Údaje o souboru úloh

- Uživatel, který ho spustil
- Datum a čas spuštění
- Datum a čas dokončení
- Celkový výsledek (udává, zda všechny úlohy proběhly úspěšně nebo neúspěšně, případně jejich část)
- Stav úloh (připravené, v běhu, dokončené, přerušené)

2.2.6 Vyhledávání, filtrování, řazení, seskupování

V aplikaci bude možné využít pokročilého vyhledávání, které umožní hledat nejen v hodnotách, ale také v názvech evidovaných parametrů. Číselné hodnoty bude možné filtrovat standardními matematickými operátory (tj. rovnost, větší než, menší než, vetší nebo rovno, menší nebo rovno), textové hodnoty pomocí výběru – jedna nebo více možností (z hodnot, kterých může vlastnost nabývat) nebo zadáním textu, s nestandardními hodnotami (podrobněji popsány dále v kapitole) bude pracovat „speciálně“.

Vyhledávání v názvech parametrů bude probíhat fulltextově – vyfiltrují se všechna zařízení, která mají evidovanou vlastnost, která odpovídá zadanému textu. Tuto možnost bude uživatel moct změnit (tj. určit, zda vyhledávání probíhá v hodnotách nebo i v názvech parametrů).

2.2.6.1 Filtrování a řazení verze systému

Verze systému s formátem verzování (4) „major.minor.patch“ (např. verze 6.31.8), u kterého by klasické abecední řazení nefungovalo dle očekávání. Bude možné vytvořit filtr, který umožní filtrovat všechny verze větší než zadané (např. vyšší verzi než 6.31 – z pohledu filtrování bude pokládáno za 6.31.0 – bude vyhovovat nejen 6.32, ale také 6.31.1 nebo 6.31.5, naopak nebude vyhovovat 6.30, 6.30.12 ani 6.31). Problém může nastat s „nestandardními“ verzemi, např. 6.31rc5 nebo 6.42beta8 (jedná se o testovací beta verzi). V tomto případě bude aplikace verzi pokládat za „major.minor.0.beta“, což umožní standardní filtrování a řazení.

2.2.6.2 Filtrování a řazení IP adres a sítí (podle masky)

Obdobně jako verze systému může být filtrování IP adres nejen pomocí zadání konkrétní IP adresy (nebo seznamu), ale i zadáním rozsahu nebo masky sítě, které odpovídá více IP adres. Bez zadání masky u IP adresy je filtrováno na konkrétní danou adresu, při zadání masky se filtruje na celou síť. U IP adres díky snadnému převodu na číselnou hodnotu se interně s IP adresami může pracovat jako s čísly a se sítěmi jako číselnými intervaly. Díky tomu je zaručeno standardní (a očekávané) řazení a filtrování.

2.2.6.3 Seskupování

Jednotlivé výsledky bude možné seskupovat, tj. zobrazit výsledky po skupinách. Např. lze seskupit podle verze systému – všechna zařízení, která mají stejnou verzi se budou zobrazovat dohromady ve skupině, s jinou verzí budou ve skupině jiné. Aplikace bude pro lepší přehled zobrazovat i počet zařízení v jednotlivých skupinách.

Obdobně jako u filtrování a řazení je možné „speciální“ seskupování pro určité vlastnosti. Např. verzi systému může uživatel seskupit podle „major“ verze (první číslo před tečkou, např. pro verzi „6.43.8“ se jedná o „6“), takže všechna zařízení se stejnou „major“ verzí, ale s rozdílnou „minor“ verzí (druhé číslo, za první tečkou) se zobrazí v jedné skupině. Podobně IP adresy bude možné seskupovat podle celých podsítí, tj. zobrazit všechna zařízení z jedné podsítě ve stejné skupině.

2.2.6.4 Pohledy (zobrazení)

Uživatel si může definovat, které vlastnosti zařízení jsou zobrazovány a které naopak skryty. Díky tomu vidí přesně informace, které v danou chvíli potřebuje. Např. při práci s uživatelskými účty si nemusí zobrazovat (v danou chvíli nepodstatné) údaje o jednotlivých rozhraních a naopak.

2.2.6.5 Ukládání a načítání pohledů (zobrazení) a filtrů

Jednotlivé pohledy (zobrazení) a filtry bude možné ukládat pro pozdější načtení. Zejména pokud si uživatel sestaví složitější filtr (např. všechna zařízení s určitou verzí systému, typem rozhraní a uživatelským účtem), může si jej uložit a pojmenovat a kdykoliv později se k němu vrátit (např. pro pozdější kontrolu, zda daná zařízení byla aktualizována na novější verzi nebo zda má daný uživatel stále k zařízením přístup).

Filtry bude mít každý uživatel své, nepočítá se se sdílenými filtry (stejný filtr bude moct použít více uživatelů).

2.2.7 Komunikace se zařízeními

S každým zařízením je navázáno spojení nezávislé na jiných zařízeních a na jiných úlohách, můžou tak běžet současně různé požadavky na jedno zařízení. Každý příkaz se na zařízení buď úspěšně provede, nebo skončí s chybou (např. zařízení daný příkaz nezpracuje korektně, není navázáno spojení, přístupové údaje jsou chybné). Výsledek komunikace a úspěšnost provedení příkazu je vrácen aplikaci k dalšímu zpracování (typicky zobrazení uživateli s podrobnostmi o chybě). Všechny příkazy k zařízení jsou ukládány včetně výsledků, je tak možné v jejich historii zjistit, jaké operace a kdy byly s kterým zařízením prováděny.

2.2.8 Další náměty na případná rozšíření

1) nahrání nového nastavení pro odpojené zařízení automaticky po jeho opětovném připojení (ihned poté, co bude zjištěna jeho dostupnost se pokusit o nahrání konfigurace)
2) zobrazení grafů historických hodnot s možností vlastního měřítka a období
3) využití jiných možností přihlašování (než kombinace uživatelského jména a hesla) systému k zařízením (např. pomocí certifikátu nebo IP adresy)
4) sdílení filtrů mezi uživateli (popř. možnost „importu“ filtru od jiného uživatele)
5) pokud se připojení k přidávanému zařízení nepodaří, naplánuje se za daný interval (dle nastavení aplikace) nový pokus o načtení údajů
6) možnost přidávání rozhraní (virtuálních) k zařízení
7) běžící služby na zařízení (DNS server) nebo na rozhraní (DHCP server, klient)
8) kromě okamžitého spuštění úlohy mít možnost naplánování do budoucna nebo spuštění na základě podmínky nebo akce (trigger) – např. v zařízení se objeví účet admin a na základě akce se provede jeho odstranění nebo nastavení hesla

2.3 Obecné možnosti konfigurace jednotlivých zařízení

Pro čtení údajů a konfiguraci jednotlivého zařízení lze využít několik způsobů, které si podrobněji projdeme. Standardem v tomto ohledu je protokol SNMP, který byl přesně pro tyto účely navržen. Dalším hojně využívaným přístupem je využití příkazové řádky (taktéž označován, i když je význam trochu jiný, jako konzola nebo terminál) – k tomu se využívají různé protokoly (nejběžnější použití je SSH nebo telnet), dále některá zařízení nabízejí vlastní rozhraní (tzv. API). Naopak u nejlevnějších zařízení pro domácí použití je nejběžnější webové rozhraní.

2.3.1 SNMP

Jak již význam zkratky napovídá (z anglického „Simple network management protocol“, neboli protokol pro správu sítě), jedná se o standard pro správu síťových prvků (5), proto také většina z nich tento protokol na různé úrovni možností podporuje. Jedná se o jeden z nejčastěji využívaných protokolů pro čtení stavových hodnot ze zařízení jako např. teplota, vytížení procesoru, využití paměti apod. Lze jej ale použít i pro konfiguraci.

Ze stránky s manuálem k SNMP pro RouterOS (6) lze stáhnout sadu objektů pro správu (v části „Management information base (MIB)“), v ní se ale nenachází informace o uživatelských účtech. Příkaz *print* v některých částech menu v konzoli (např. *interface*) obsahuje parametr *oid*, který zobrazí identifikátory objektů spravovaných pomocí SNMP. V části menu *user* (která slouží pro správu uživatelů) příkaz *print* parametr *oid* neobsahuje. Z toho vyplývá, že tento protokol v RouterOS nelze použít pro správu uživatelských účtů, a proto se jím práce dále zabývat nebude.

2.3.2 Webové rozhraní

Jedná se o uživatelsky nejpřívětivější možnost konfigurace, kdy pro připojení je potřeba pouze webový prohlížeč. V něm se pak zobrazí jednotlivé stránky s nastavením a statistikami. Toto ovládání je velmi intuitivní, uživatelé jsou na něj běžně zvyklí z internetových stránek.

Pro automatizaci není tato možnost příliš vhodná z několika důvodů. Předně většinou neexistuje kompletní popis celého rozhraní, k dispozici jsou pouze jednoduché manuály, které jsou psány spíš jako průvodci. Dále získání hodnot znamená čtení a zpracování jazyka HTML, který slouží k zobrazování dat, opačný postup (získání dat) je proto mnohem náročnější (navíc náchylný na změny v rozhraní – z toho vyplývá nutnost zkoumat změny při nefunkčnosti a patřičně upravit aplikaci). Další překážkou u některých zařízení může být fakt, že jsou stránky mnohdy generované až na straně webového prohlížeče, nestačí tak poslat požadavek na danou stránku a pak ji jednoduše zpracovat, ale je nezbytné mít zprovozněn virtuální webový prohlížeč a z něj po zpracování data získávat. Z uvedených důvodů se touto možností nebudu dále zabývat

2.3.3 Příkazová řádka – telnet, SSH

Většina profesionálních zařízení (ne-li všechna) disponuje možností zadávání příkazů přes místní nebo vzdálený terminál (též označován jako konzola). Uživateli se zobrazí textová příkazová řádka, kam může psát příkazy a zařízení na jejich základě vypisuje požadované informace nebo provádí nastavení.

Pro přístup k příkazové řádce se stále ještě používá (dnes již zastaralý a nezabezpečený) protokol telnet (7), který je ale nahrazen novějším a zabezpečeným protokolem SSH (8). Tyto protokoly nejsou jediné, ale jsou zdaleka nejrozšířenější. Zkratka SSH vznikla z anglického „secure shell“ a označuje zabezpečené rozhraní pro komunikaci. Z uživatelského hlediska je přístup přes telnet totožný s přístupem přes SSH. Oba protokoly jsou standardizovány a díky tomu implementovány v mnoha zařízeních.

Výhodou tohoto přístupu je dobrá dokumentace jednotlivých příkazů a jejich parametrů (9). Určitou nevýhodou je mnohdy textově orientovaný formátovaný výstup, který je potřeba dále zpracovávat pro strojové načtení a následné zobrazení. Některá zařízení ale disponují příkazy (nebo jejich parametry), které výstup formátují právě pro strojové čtení. Jako obecný příklad poslouží formát CSV (10) („comma separated values” – hodnoty oddělené čárkou; u nás je používanějším oddělovačem středník, protože čárka slouží k oddělení desetinných míst u čísel, k oddělení ale může být použit i jiný znak, např. tabulátor). Na prvním řádku (hlavička) mohou být volitelně vypsány názvy oddělené čárkou a na dalších řádcích jsou hodnoty vlastností s názvem z prvního řádku, oddělené taktéž čárkou. Pro takový formát existují knihovny (11) (12) pro načtení a zpracování – vývojář tak má přístup přímo ke zpracovaným údajům.

Konkrétně příkazová řádka systému RouterOS nabízí několik parametrů příkazu *print* (13) pro strojové zpracování nebo přehlednější zobrazení (zkratky použité v ukázkách níže: *id* – identifikátor záznamu, *z* – záznam, *v* – vlastnost záznamu). Problém při strojovém zpracování může způsobit hodnota, která obsahuje středník nebo symbol rovnosti (tyto symboly mohou být obsaženy např. v komentářích).

Detailnější popis jednotlivých parametrů

- *as-value* – všechny záznamy na jednom řádku jako pole parametrů a hodnot oddělených středníkem

  Obecný formát (ukázka)
  ```
  id=*1;z1v1=h1;z1v2=h2;id=*2;z2v1=h3;z2v2=h4
  ```
  Konkrétní příklad
  ```
  .id=*1;address=192.168.1.1/24;comment=defconf;interface=bridge;
  network=192.168.1.0;.id=*2;address=192.168.100.11/24;comment=;i
  nterface=ether1;network=192.168.100.0
  ```

- *terse* – každý záznam na jednom číslovaném řádku (za číslem mohou být další symboly značící určité stavy nebo hodnoty), vlastnosti s hodnotou oddělené mezerou

Obecný formát (ukázka)

```
0 I z1v1=h1 z1v2=h2
1   z2v1=h3 z2v2=h4
```

Konkrétní příklad

```
 0   comment=defconf address=192.168.126.1/24
network=192.168.126.0 interface=bridge actual-interface=bridge
 1   address=192.168.100.11/24 network=192.168.100.0
interface=ether1 actual-interface=ether1
```

- *value-list* – každý záznam jako jeden sloupec, na řádku je vždy jedna vlastnost (název s dvojtečkou na začátku řádku, hodnoty oddělené mezerami – zarovnáno do sloupců)

  Obecný formát (ukázka)

  ```
  v1: z1h1 z2h2
  v2: z1h3 z2h4
  ```

  Konkrétní příklad

  ```
               address: 192.168.126.1/24 192.168.100.11/24
               network: 192.168.126.0    192.168.100.0
             interface: bridge           ether1
    actual-interface: bridge           ether1
  ```

2.3.4 API

Zkratka vychází z anglického „application programming interface“ (2) a označuje rozhraní vytvořené speciálně pro komunikaci mezi zařízeními nebo aplikacemi (není primárně určeno pro komunikaci s uživatelem). Obecně API umožňuje propojení různých systémů, dobře se z něj načítají data pro další zpracování a do něj zapisují. Dokumentace k API bývá podrobná, příkladem může být API pro systém RouterOS (14), kterým se práce zabývá, nebo mnohá další (15). Dále v této části je RouterOS API stručně popsáno.

Díky podpoře API mohly vzniknout služby, které využívají různá API mnoha aplikací a uživatelům nabízejí jejich propojení. Jako příklad mohou posloužit Zapier (16) nebo Integromat (17), které jsou detailněji (včetně mnoha příkladů použití) popsány na svých stránkách. Lze tak velmi jednoduše na změny v jedné aplikaci (např. příchozí email) reagovat v aplikaci jiné (např. založení požadavku v systému řízení změn). Bez podpory API by žádná obdobná služba vzniknout nemohla.

RouterOS API

V RouterOS probíhá komunikace přes API v definovaném formátu (14). Při volání API se odesílá „věta“ (v dokumentaci označována jako „sentence“), která se skládá ze „slov“ (označení „word“). Slovem může být příkaz (označení „command word“), atribut („attribute word“), API atribut („API attribute

word“) nebo dotaz („query word“). Každé slovo na začátku obsahuje zakódovanou svou délku (dle specifikace v dokumentaci) a poté vlastní obsah. Za posledním slovem pro ukončení věty je odesláno slovo s nulovou délkou (tj. pošle se nula, za níž dále komunikace neprobíhá nebo začíná nová věta a RouterOS v tomto okamžiku příkaz zpracuje a odešle zpět odpověď).

Jednotlivé příkazy („command word“) odpovídají příkazům dostupným přes příkazovou řádku s drobnými odlišnostmi (18), např. příkazu `/system clock print` odpovídá v API `/system/clock/print`. Místo mezerami jsou jednotlivé příkazy odděleny lomítky. Za příkazem mohou následovat další „slova“. Atributy mají formát `=nazev-atributu=hodnota-atributu`, API atributy `.nazev-API-atributu=hodnota-API-atributu` (obsahují před názvem tečku, aktuálně je pouze jediný - `tag`), dotazy (sloužící k filtrování objektů) mají více možných formátů

`?nazev-vlastnosti` – objekt má vlastnost daného názvu,
`?-nazev-vlastnosti` – objekt nemá vlastnost daného názvu,
`?nazev-vlastnosti=hodnota` nebo `?=nazev=hodnota` – vlastnost objektu je roven zadané hodnotě,
`?<nazev-vlastnosti=hodnota` – vlastnost nabývá hodnoty menší než zadaná,
`?>nazev-vlastnosti=hodnota` – vlastnost nabývá hodnoty větší než zadaná,
`?#operace` – slouží k operacím nad předchozími zadanými dotazy, operací může být například logická spojka „nebo“ (značka „`|`“).
Pořadí jednotlivých atributů není podstatné, naopak pořadí dotazů je (při přehození může dojít ke změně významu a tím i výsledku).

Po odeslání celé „věty“ (ukončené „slovem“ s nulovou délkou) je výsledek příkazu vrácen jako „věta“, v níž první „slovo“ začíná vykřičníkem, pak následují „slova“ s vlastní odpovědí ukončené „slovem“ s významem konce odpovědi – „!done“.

Například dotaz `/interface/print` vrátí informace o všech rozhraních, přidáním dotazu `?name=ether1` se vyfiltruje pouze rozhraní s názvem „ether1“, přidáním dalšího dotazu `?disabled=yes` se vyfiltruje dané rozhraní jen v případě, že je zakázáno (v opačném případě v odpovědi nebude žádné rozhraní). Jednotlivá „slova“ odpovědi mohou být
`!re` – příkaz proběhl správně, odpověď obsahuje požadovaná data
`!trap` – příkaz skončil chybou nebo výjimkou, která je obsažena v odpovědi
`!fatal` – pokud je spojení po chybě ukončeno

Po odeslání příkazu `/user/print` je vráceno

```
!re
=.id=*2
=name=admin2
```

```
=group=full
=last-logged-in=dec/11/2016 10:12:34
=disabled=false
!done
```

Pro snadné použití v aplikaci jsou k dispozici připravené knihovny pro různé programovací jazyky (jejich seznam je uveden na konci dokumentace (14)), které jsou postavené nad touto specifikací a poskytují vývojáři přístup k API na vyšší úrovni (nemusí se např. starat o kódování délky „slov“ nebo načítání jednotlivých částí odpovědi). Některá poskytují i připravené objekty (např. „tik4net“ (19)), nad nimiž lze provádět pouze relevantní množinu operací (např. vlastnosti pouze pro čtení nelze nastavit hodnotu).

2.4 Současné nástroje

Přímo tvůrce RouterOS, firma Mikrotik, na svých stránkách nabízí ke stažení několik nástrojů pro konfiguraci. Jedním z nich je The Dude (20), který slouží pro správu jakékoliv sítě (s množstvím pokročilých funkcí jakými jsou automatické prohledávání sítě, sledování statistik a vytváření grafů, zasílání upozornění při výpadku). Samozřejmostí je podpora pro zařízení se systémem RouterOS, pro jejichž správu je nástroj určen, proto se jím práce bude zabývat nejvíce (prozkoumá ale i další volně šiřitelné nástroje). Ostatní konfigurační nástroje (Winbox, Webfig) od firmy Mikrotik slouží pro nastavení jednoho zařízení. Práce vynechává takové nástroje, které nedisponují webovým rozhraním a jsou vázány na použití pouze na běžném počítači (nebo jen na některém z podporovaných operačních systémů). Mezi open source aplikacemi existuje spousta, které se zabývají pouze monitoringem, ale žádná s podporou RouterOS pro podporu hromadné konfigurace.

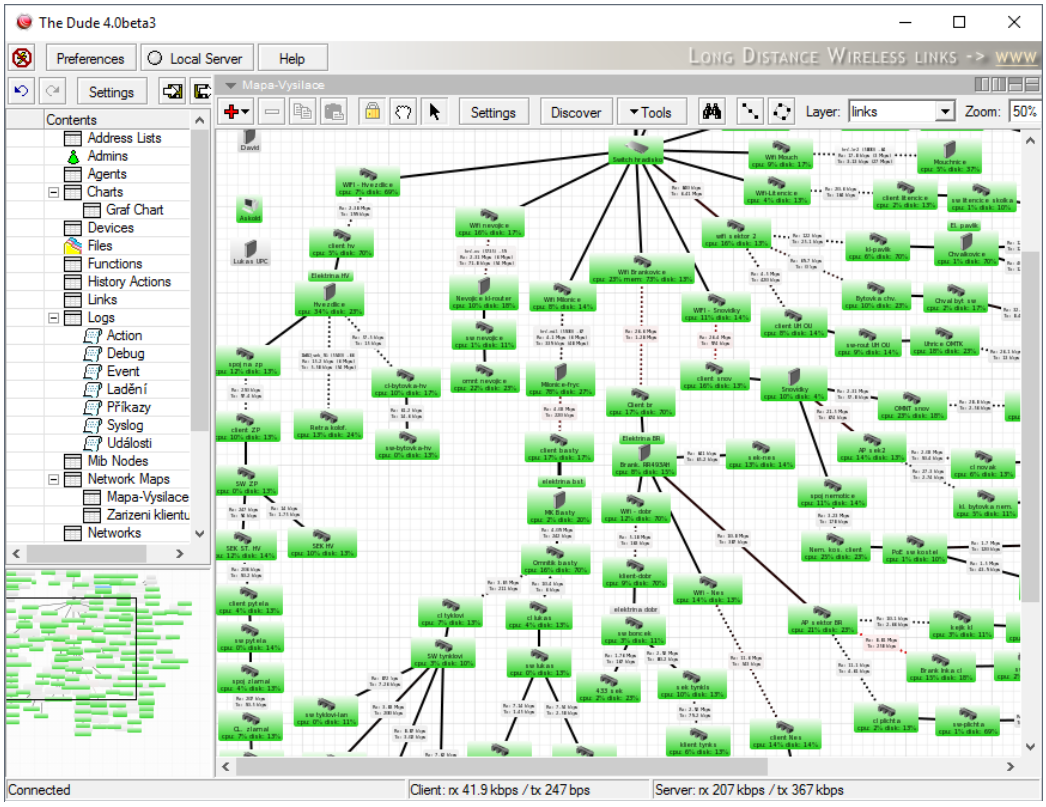
2.4.1 The Dude

Pro správu sítě je používán tento nástroj v referenční síti již dlouhou dobu, obsahuje aplikaci pro Windows (s možností běhu pod Linuxem nebo MacOS prostřednictvím Wine, resp. Darwine) a k tomu (s mnohem omezenější funkčností) webové rozhraní. V používané starší verzi (4.0beta3) je ještě možné v klientské části vytvořit tzv. lokální server, který může běžet i jako služba na pozadí, což je aktuálně v reálném provozu využíváno. Nástroj je dostupný volně k užívání, má ale uzavřený kód.

Správa zařízení

Nástroj poskytuje přehled nad všemi sledovanými zařízeními zobrazenými na mapách (Obrázek 1) nebo vypsanými v seznamu s indikací stavu (Obrázek 2) a umožňuje snadné spuštění konfiguračních (Winbox, telnet, SSH, …) a diagnostických (ping, traceroute, snmpwalk, …) nástrojů.

Pokud se jedná o zařízení se systémem RouterOS, lze v seznamu zjistit aktuální verzi systému, distribuovat novější a zařízení tak hromadně aktualizovat. Nejdříve je nutné danou verzi do systému uložit, ten pak při aktualizaci hlídá, zda se jedná o správnou platformu a není-li již v zařízení novější verze. Bohužel ale už nekontroluje závislosti mezi zařízeními, může se tak při nevhodném výběru zařízení stát, že zatímco se novější verze nahrála do jednoho zařízení, které se právě restartuje, aby se zaktualizovalo, nahrávání do jiného (závislého) se přeruší (a již nenaváže). Je proto doporučeno vytvořit si skupiny, které obsahují pouze vzájemně nezávislá zařízení. Bohužel při změnách topologie sítě je potřeba myslet



OBRÁZEK 1

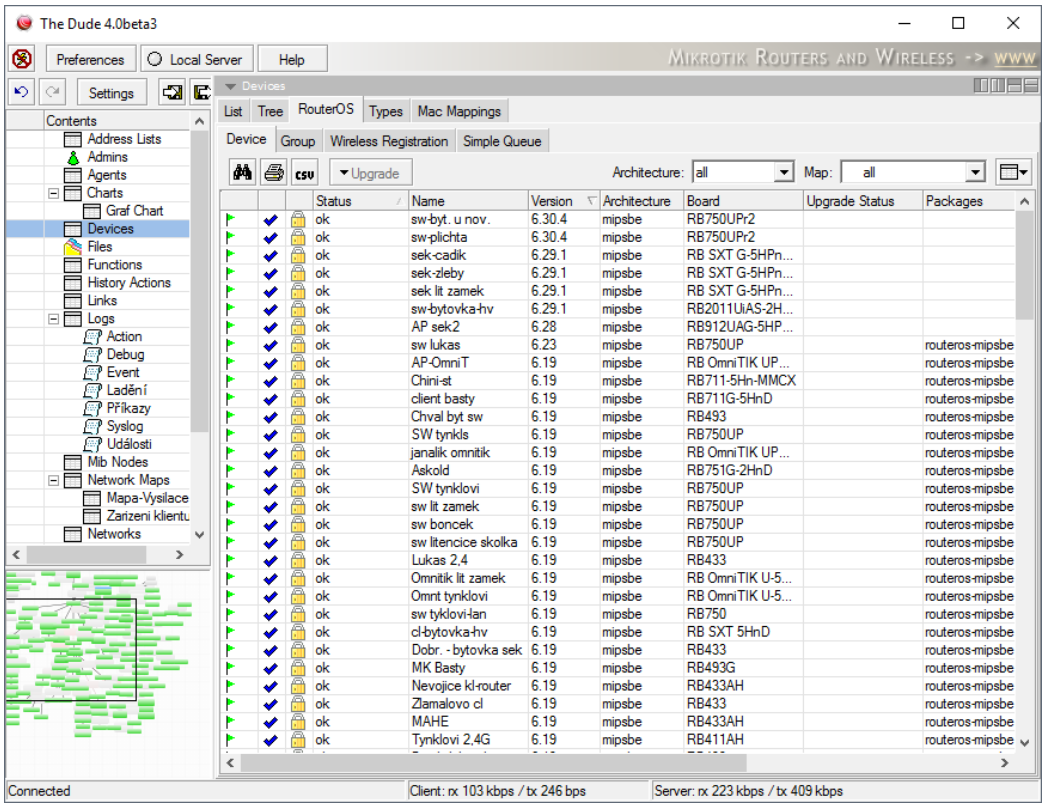
také na úpravu jednotlivých skupin.

V případě, že některé zařízení není přímo dostupné, lze využít v síti více agentů (tím se rozumí běžící instance systému), kteří spolu komunikují. Lze tak zvenčí zpřístupnit pouze tohoto agenta bez nutnosti

povolování přístupu přímo k zařízením. Správce pak vidí v přehledu všechna zařízení (může si je členit do map nebo skupin) ze všech agentů.

Monitoring služeb běžících na sledovaném zařízení

Pro každé zařízení lze definovat služby, které se mají sledovat, ke každé je přiřazena odpovídající sonda (21) (což je definice, jak se má zjistit, zda služba běží nebo ne – jaký protokol použít, na jaký port přistoupit, jaká data odeslat a jaká je očekávaná odpověď). Systém pak u služeb sleduje dostupnost



OBRÁZEK 2

(s nastavením jak často se má zjišťovat a jak dlouho čekat na odpověď), zaznamenává výpadky (je možné nastavit, kolik po sobě jdoucích neúspěšných pokusů se vyhodnotí jako výpadek) a umožňuje nastavit různé formy upozornění (od zapípání nebo zablikání přes zaznamenání do místního nebo vzdáleného logu až po odeslání emailu). Lze tak velmi granulárně rozlišovat důležitost služby a v případě potřeby informovat zodpovědnou osobu. Kromě notifikací se také výpadek služby projeví na mapě různou změnou barvy při výpadku části sledovaných služeb nebo (další změnou barvy) při výpadku všech. Pokud se někdo daným výpadkem již zabývá, může ho potvrdit, což se opět projeví změnou barvy a ostatní tak jsou o této skutečnosti informováni. Všechny barvy lze měnit, nicméně výchozí nastavení (zelená – vše v pořádku, oranžová – částečný výpadek, červená – úplný výpadek všech služeb, modrá – potvrzení řešení výpadku) je zvoleno vhodně.

Součástí monitoringu jsou i grafy sledovaných služeb (např. doba odezvy), u nichž lze nastavit, jak dlouho se mají a s jakou granularitou ukládat (data pro starší období se stále více agregují – např. pro poslední 2 hodiny se uchovávají všechny hodnoty, za poslední den se vytváří průměr za 5 minut, za

poslední týden se průměruje každá hodina atd.). Mezi grafy najdeme i datové toky sledovaných spojů mezi zařízeními. Obecně lze vytvářet graf s jakoukoliv přesností z jakéhokoliv zdroje (které lze libovolně uživatelsky definovat – např. dělat průměr odezvy různých zařízení, sčítat toky v různých částech sítě).

Webové rozhraní

Webové rozhraní na rozdíl od desktopového má velmi omezené možnosti. V podstatě v něm lze pouze pracovat se seznamy (prohlížet a přidávat položky, např. zařízení, sledované služby) a prohlížet mapu (která se obnovuje místo po sekundě každých 30 s, nelze kliknutím na zařízení zobrazit jeho podrobnosti nebo měnit jeho pozici). Je možné také zobrazovat grafy sledovaných hodnot pro jednotlivá zařízení (např. odezva) nebo spoje mezi nimi (příchozí/odchozí provoz).

Aktuální verze

Novější verzi (serverovou část) lze instalovat pouze jako balíček do systému RouterOS (obsahuje balíčky např. pro bezdrátové sítě, směrování, zabezpečení). V testovaném staženém obrazu již je balíček dude obsažen, stačí jej pouze dle návodu (22) příkazem `/dude set enabled=yes` povolit a poté router restartovat. Bohužel poslední verze už neobsahuje webové rozhraní, které bylo odstraněno.

2.4.2 Network Configuration Manager (ManageEngine)

Tento produkt je zaměřený na správu konfigurace pomocí konfiguračních souborů, které si sám periodicky stahuje (nastavitelné) a porovnává je s uloženými. Ke stažení může dojít na základě události v syslogu (23) (24) – ten je aplikací monitorován a pokud zjistí, že došlo k odhlášení uživatele, zkontroluje, jestli byly provedeny nějaké změny konfigurace. Na změnu může automaticky zareagovat (např. zasláním upozornění nebo automatickým nahráním původní konfigurace). Uživatel má možnost při detekci změny konfigurace ji potvrdit a tím se stává novou aktuální (ke které se lze případně vrátit a vůči níž se porovnávají změny) nebo změny vrátit zpět. Aktuální konfigurace slouží také pro případ poruchy zařízení a nutnost jeho fyzické výměny. Samozřejmostí je uchovávání historie konfigurací a možnost vrácení se ke kterékoliv předchozí konfiguraci, stejně tak vzájemné porovnání kterýchkoliv starších verzí na jednom zařízení nebo i mezi různými zařízeními.

Konfigurace zařízení může probíhat pomocí tzv. „configletů“ – parametrizovatelnými skripty, které lze spouštět na více zařízeních – jednorázové, naplánovat spuštění na určitý čas nebo je spouštět pravidelně. Největší výhoda spočívá právě v možnosti parametrizace. Například aktualizace firmwaru zařízení probíhá nahráním souboru přes FTP (adresa serveru je předána parametrem) a spuštěním příkazu pro aktualizaci nebo restartováním zařízení. Se zařízením probíhá spouštění příkazů přes telnet nebo SSH. Podporována jsou zařízení mnoha výrobců (Cisco, HP, Mikrotik a mnoho dalších). Aplikace si umí oskenovat okolní síť (dle nastavení uživatelem) a nalezená zařízení přidat do své správy, případně vytvořit pro ně skupiny, se kterými lze při hromadné správě dále pracovat. Jsou podporovány i role uživatelů.

Zatímco administrátor má neomezená práva, změny konfigurace operátora musí být potvrzeny administrátorem.

Z pohledu požadavků systém dokáže zobrazit verzi systému a aktualizovat ji, vést informace o zařízeních a přístupové údaje k nim. Dále je možné spravovat uživatele aplikace (navíc s podporou pro role), řadit a filtrovat zařízení, chybí možnost seskupování. Spouštění úloh je možné pro jednotlivé zařízení i pro definovatelnou skupinu. Tím, že veškeré nastavení je pomocí konfiguračních souboru, lze z nich zjistit v podstatě veškeré potřebné informace, ovšem bez podpory zobrazení jednotlivých hodnot v uživatelském rozhraní. Například není možné si zobrazit seznam uživatelů nebo IP adres na jednotlivém rozhraní a tato data dál filtrovat. Aplikace poskytuje REST API pro propojení s dalšími systémy. Licence pro použití pro 2 zařízení je zdarma, pro více je potřeba komerční licence (s možností bezplatného vyzkoušení na 30 dní).

2.4.3 Unimus

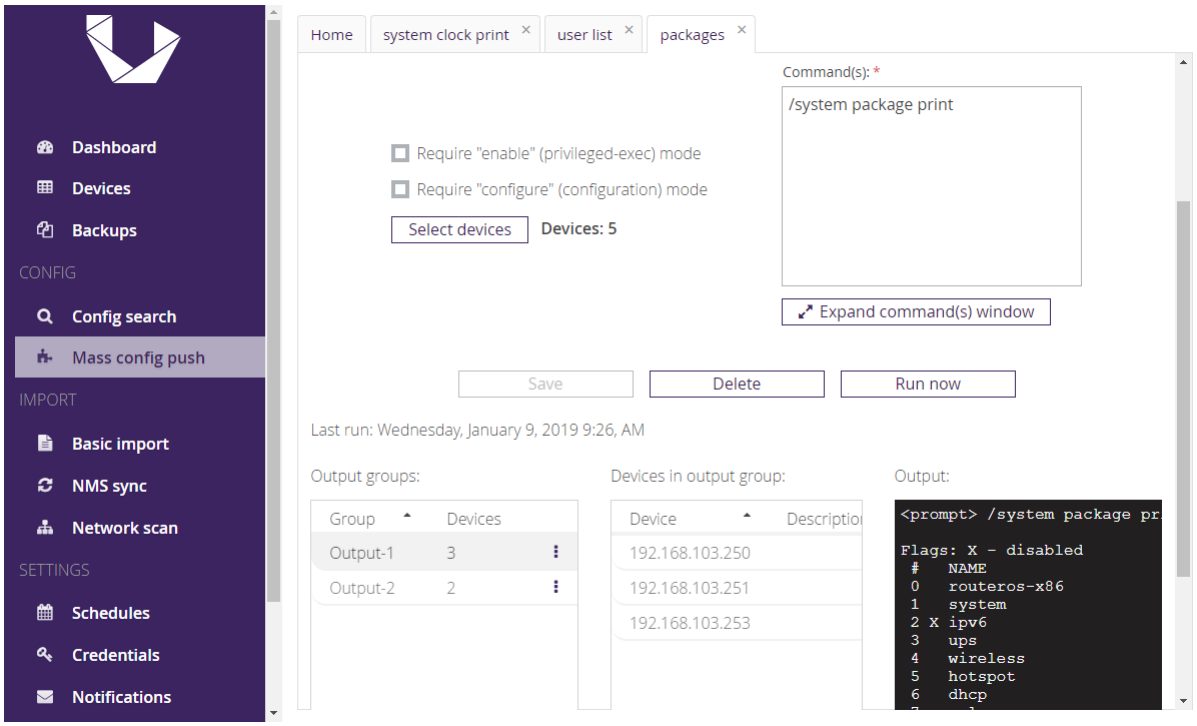
Jednoduchý nástroj (25) s hlavním účelem – hromadné spouštění skriptů. Kromě toho umí zálohovat aktuální konfiguraci, udržovat jejich historii, porovnávat jednotlivé verze (stejného zařízení nebo vzájemně mezi jednotlivými) a vyhledávat v nich (včetně pokročilého s použitím regulárních výrazů).

Kromě klasického instalačního balíčku lze stáhnout i přenositelnou verzi bez nutnosti instalace, vše je v jediném spustitelném souboru. Napoprvé se zobrazí krátký průvodce úvodním nastavením. V jeho průběhu se vyplní uživatelské jméno a heslo nového uživatele pro přihlášení, šifrovací klíč pro zabezpečení přihlašovacích údajů k zařízením uložených v databázi, připojení k databázi (obsahuje i možnost použití lokální integrované databáze), přihlašovací údaje k zařízením a vloží se licenční klíč (který lze získat po registraci přímo ze stránek aplikace). Poté již stačí naimportovat jednotlivá zařízení (ze souboru nebo přímo vypsat v zadaném formátu) nebo nadefinovat sítě a spustit jejich skenování, které lze rovnou i naplánovat pravidelně. Zařízení mohou být importována také synchronizací s externím systémem pro monitoring (podporovány jsou NetXMS, Zabbix a Powercode).

Ke každému zařízení systém automaticky zkouší přistoupit pomocí všech vyplněných přístupových údajů, dokud se mu nepodaří se přihlásit. Ke každému zařízením lze místo tohoto výchozího chování napevno napárovat konkrétní přihlašovací údaje. Pokud je v seznamu údajů i uživatel admin (s jakýmkoliv heslem), pak lze zjistit i zařízení, která mají ponechán výchozí účet admin bez hesla. Připojení přes SSH k zařízení s výchozím účtem se vždy podaří (po pokusu o přihlášení s uživatelským účtem bez hesla je přihlášení vždy úspěšné).

Pro uživatele aplikace jsou podporovány role Administrator (s plným přístupem), Operator (bez přístupu k účtům aplikace) a Read-Only (pouze pro čtení, na rozdíl od Operátora nemůže ani provádět žádné

změny). Přístup uživatelů k jednotlivým zařízením je řešen pomocí štítků, které lze přiřadit jak k zařízením, tak k uživatelům a tím jim zpřístupnit pouze danou skupinu zařízení.



OBRÁZEK 3

Po spuštění skriptu nad skupinou zařízení jsou výsledky seskupeny podle odpovědi zařízení, lze tak jednoduše zjistit, která zařízení vůbec neodpověděla nebo která měla stejnou odpověď. Se zařízeními, která měla stejný výsledek je možné dále pracovat (použít pro další dotaz nebo znovu spustit stejný). Tímto způsobem lze zjistit, která zařízení mají stejná nastavení (např. verzi systému, stejné balíčky apod.). V seznamu zařízení je možné řadit a vyhledávat. Kromě základních údajů (IP adresa, uživatelský popis, výrobce, typ a model) se zde nezobrazují žádné další informace. Vyhledávání a řazení je tak omezeno pouze na tyto údaje – nezobrazuje se např. verze systému na rozdíl od předchozí aplikace „Network Configuration Manager (ManageEngine)“. Stejně tak při srovnání s tímto produktem není možné parametrizovat skripty ani je pravidelně automaticky spouštět. Celkově se jedná v porovnání o velmi zjednodušenou variantu, díky tomu ale i mnohem jednodušší na ovládání. Licence pro 5 zařízení je zdarma, pro více je placená.

2.4.4 Další nástroje

Při důkladnějším zkoumání dalších nástrojů pro správu sítí bylo vždy zjištěno, že se jedná pouze o podporu monitorování sítě (byť mnohdy na vysoké úrovni) s žádnou nebo minimální podporou pro nastavení (většinou pomocí protokolu SNMP) nebo dostupnou jako placený doplněk. Nemalé množství aplikací bylo úplně přeskočeno bez podrobnějšího zkoumání, protože disponovaly pouze desktopovým rozhraním. Jejich rozsáhlý (samozřejmě ne úplný) seznam lze, kromě využití internetových vyhledávačů, také najít (včetně možností filtrování podle funkcí a vlastností) na GetApp (26).

OpenNMS

Jedním z hojně používaných nástrojů splňujícím kritéria na bezplatnou licenci a otevřený kód je OpenNMS (27). Po bližším prozkoumání bylo bohužel zjištěno, že je zaměřen na monitorování. Umožňuje sledovat různé systémy a služby pomocí široké škály podporovaných protokolů a údaje poté vykreslovat do nastavitelných grafů. Podporuje události a na jejich základě generuje notifikace nebo po propojení se systémy pro správu úkolů může dle nastavených pravidel vytvořit nový s popisem chyby.

2.5 Shrnutí

Nejdříve byly stanoveny požadavky a očekávání od nástroje, který by pomohl se správou síťových zařízení. U současných nástrojů byla zkoumána zejména správa uživatelů a konfigurace síťového nastavení. Protože referenční síť je z největší části provozována na nařízeních se systémem RouterOS, byla hlavní podmínkou spolupráce s tímto systémem. Nejpokročilejší možnosti nabízí aplikace The Dude od samotného výrobce RouterOS (bohužel nebyla dlouho aktivně vyvíjena), ale v současnosti nepodporuje správu uživatelů. Kromě hromadné aktualizace systému nemá další pokročilé možnosti hromadné správy více zařízení, umožňuje pouze pohodlně z jednoho prostředí měnit nastavení konkrétního zařízení. Ostatní zkoumané systémy se zaměřují hlavně na monitoring a pokud umožňují nastavovat některé vlastnosti, děje se tak pomocí protokolu SNMP (který v RouterOS nelze použít pro správu uživatelů) nebo se jedná o placené aplikace určené pro práci s konfiguračními skripty (případně ručním zadáváním příkazů). Pro přehlednost je uvedena tabulka se souhrnem významných vlastností.

TABULKA 1

	The Dude	Network Configuration Manager	Unimus
Podporovaná platforma	Windows	Linux, Windows, iOS klient	Linux, Windows, MacOS
Webové rozhraní	ano, omezené	ano	ano
Open source (otevřený kód)	ne	ne	ne
Licence	bezplatná	bezplatná (max. 2 zařízení), placená (nad 2 zařízení)	bezplatná (max. 5 zařízení), placená (nad 5 zařízení)
Programové rozhraní API	ne, v nové verzi možnost SSH	ano	ano
Zobrazení verze systému v seznamu zařízení	ano	ano	ne (lze spuštěním skriptu)
Správa přístupových údajů k zařízením	ne, pouze k jednotlivému zařízení	ano	ano
Zobrazení uživatelů zařízení	ne	ano, spuštěním skriptu	ano, spuštěním skriptu
Hromadné přidání uživatelů zařízení	ne	ano, spuštěním skriptu	ano, spuštěním skriptu

Zobrazení ip adres rozhraní	pro jednotlivé zařízení	ano, spuštěním skriptu	ano, spuštěním skriptu
Hromadné přidání ip adres	ne	ano, spuštěním skriptu	ano, omezené
Vyhledávání, řazení, filtrování	pouze základní vyhledávání	ano, základní	ano, základní
Hromadné akce se zařízeními	ano, omezené	ano, pokročilé, parametrizovatelné skripty	ano, základní, skripty
Správa uživatelů aplikace, oprávnění	ano, základní, vč. rolí	ano, pokročilé, vč. rolí	ano, základní, vč. rolí

3 Testovací prostředí

Pro testování funkčnosti různých aplikací není vhodné používat reálnou síť s uživateli, a to hned z několika důvodů. Předně uživatel nezná kvalitu testované aplikace, případné chyby mohou způsobit chybné provedení příkazů. Dále může obsahovat automatické procesy, které probíhají na pozadí – např. sama prozkoumává síťové okolí a snaží se zjistit běžící služby nebo se k některým připojit a získat z nich údaje. Podobné obtíže může způsobit neznalost uživatele chování dané aplikace (např. jestli je potřeba změny uložit nebo se provádějí automaticky). Rozhodně není žádoucí stav, kdy z nějakého obdobného důvodu testování aplikace dojde k výpadku služby koncovému uživateli. Pro vytvoření testovacího prostředí byl zvolen nástroj GNS3 (28), který umožňuje vytvářet virtuální zařízení (ať už routery, switche, huby nebo také počítače) a různě je mezi sebou (nebo i s hostitelským počítačem) propojovat. Díky tomu, že tento program splňoval nároky pro testování, nebyly zkoumány jiné alternativy.

3.1 Instalace GNS3 VM, zprovoznění, vytvoření prostředí

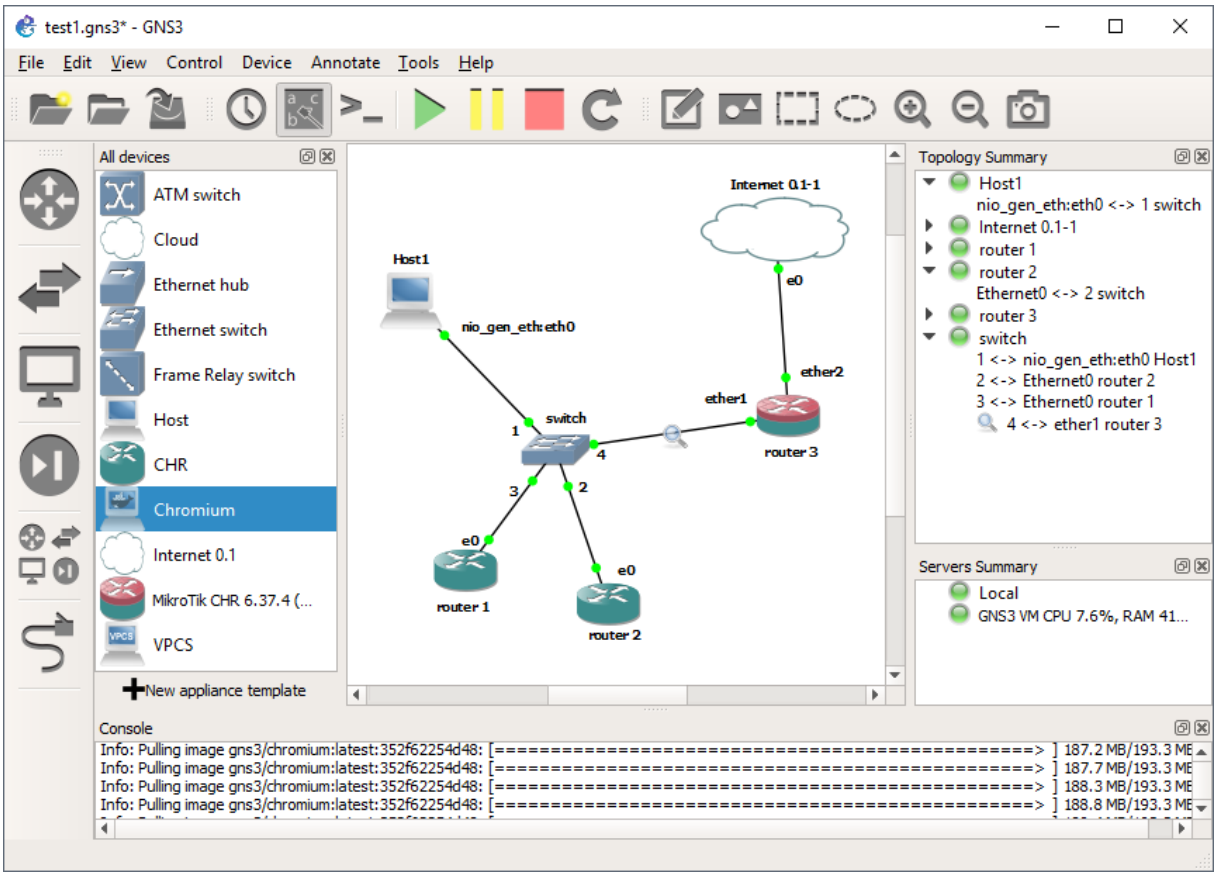
První překážkou při zprovozňování testovacího prostředí byla nutnost podpory virtualizační technologie procesorem (u procesorů Intel označována jako VT (29)). Přestože ji procesor podporuje, virtualizační (ale i identifikační) nástroje mohou zahlásit, že podporována není. Při vyhledávání problému na internetu jsou k dispozici zejména návody, jak povolit virtualizaci v BIOSu (několikanásobný vstup do něj, ověření nastavení a změna kombinace podporovaných technologií nemusí vést k úspěchu). Nakonec problém může vyřešit zkoumání chyby při spouštění některého dalšího virtualizačního nástroje. Odkaz na vyřešení opět pomocí restartu se vstupem do BIOSu již vyzkoušen byl, ale součástí návodu bylo také odinstalování Hyper-V, což se nakonec ukázalo jako průvodce celého problému.

Samotná instalace GNS3 je vcelku snadná. Nejdříve je nutné nainstalovat některý z virtualizačních nástrojů. Dle doporučení na stránkách GNS3 (rychlejší běh) byl nainstalován VMware Workstation Player (30), který je k nekomerčním účelům k dispozici zdarma. Pro něj lze pak ze stránek GNS3 stáhnout virtuální obraz (31). Po instalaci a spuštění GNS3 se zobrazí průvodce, ve kterém stačí vybrat daný virtualizační nástroj a obraz. Všechna testovací zařízení se pak spouští uvnitř virtuálního počítače z tohoto obrazu. Obrázek 4 ukazuje spuštěný program s několik běžícími a propojenými routery.

Spolu s programem lze volitelně nainstalovat další užitečné nástroje. Za zmínku stojí především program Wireshark (32), který umožňuje sledování síťového provozu. Stačí kliknout pravým tlačítkem na propojení dvou zařízení a vybrat možnost „Start capture“ (spustit zachytávání provozu). Zatím není možné sledovat provoz mezi dvěma routery, pouze na portech rozbočovačů. Stačí ale mezi routery umístit rozbočovač a zachytávat síťový provoz na něm. K čemu je tato schopnost dobrá, není potřeba příliš vysvětlovat – hodí se vždy pro kontrolu nebo ověření, jaká data mezi zařízeními putují a odladění vlastního programu nebo prozkoumání určitého protokolu.

Velkou výhodou GNS3 je možnost připojení k lokální konzoli každého zařízení. Není tak nutné mít zařízení vůbec propojená, nemusí mít ani nakonfigurovány síťová rozhraní, přesto je lze nastavovat a až poté propojit s připravenou konfigurací. Pokud nějaké zařízení přestane odpovídat po změně nastavení, hodí se tato možnost pro zjišťování příčin.

Součástí GNS3 je i Marketplace (33), v němž se nachází mnoho připravených zařízení různých výrobců, která nejsou standardní součástí programu. Lze v něm např. najít i „MikroTik Cloud Hosted Router“, který lze stáhnout a přidat do seznamu zařízení v GNS3.



OBRÁZEK 4

3.2 Licencování RouterOS pro testování

Pro testování RouterOS se nabízí několik možností licencí. První možností je si zakoupit licenci pro každý testovací router, což vzhledem k účelu není vhodná varianta. Mikrotik na svých stránkách nabízí ke stažení obrazy k instalaci systému, který běží bez omezení 24 hodin a poté je potřeba licenci zakoupit (nebo router smazat a vytvořit nový), do tohoto časového omezení se počítá pouze doba běhu, nikoliv čas, kdy byl vypnutý. V případě potřeby delšího testování je možné po zaregistrování na stránkách získat časově neomezenou verzi, ale s omezenou funkcionalitou.

Nejlepší možností pro daný účel (testování konfigurování zařízení) je tzv. verze Cloud Hosted Router, neboli varianta přímo určená pro běh ve virtualizovaném prostředí (předešlé varianty jsou určené pro

běh fyzických zařízeních, přestože je virtualizace možná). Opět je možné využít několik možností. Jednou z nich je časově omezená 60denní plně funkční zkušební verze (licenci lze získat pro zaregistrované uživatele). Během této doby lze produkt zakoupit (stálá licence včetně aktualizací) nebo po jejím uplynutí zůstane systém plně funkční, ale nepůjde jej aktualizovat. Poslední možností je bez časového omezení a bez nutnosti registrace (a je proto vhodná pro dlouhodobé testování, aniž by se musely routery mazat a znovu vytvářet), zato s omezením propustnosti, která je snížená na 1 Mb/s pro jednotlivé rozhraní, což vzhledem k zamýšlenému užívání není omezující. Z tohoto důvodu je využívána poslední možnost (lze neustále testovat aktualizace, aniž by bylo nutné do testovacího prostředí zasahovat).

3.3 Vytvoření zařízení/routerů

Do takto připraveného prostředí bylo vytvořeno několik routerů, které byly přes virtuální switch (přepínač) zapojeny k hostitelskému (virtuálnímu) rozhraní, které si vytvořil virtualizační nástroj. Na tomto rozhraní běží DHCP server, naproti tomu ve výchozím stavu na routerech běží na každém rozhraní DHCP klient, takže všechny routery získají IP adresu. Samozřejmě je možné toto chování změnit, nicméně pro rychlé počáteční testování nástrojů je takové chování žádoucí. Pro dlouhodobější testování je lepší varianta s ručním nastavením, u níž se nemohou měnit adresy jednotlivých routerů. Server DHCP může při každé žádosti o adresu přiřadit jinou, což by komplikovalo testování a vývoj aplikace.

4 Vývojové prostředí a nástroje

Celá aplikaci je rozčleněna na dvě logické části. První (nazvěme ji serverová neboli backend) je zodpovědná za komunikaci s jednotlivými zařízeními a uchovávání dat (vkládaných uživatelem nebo načtených ze zařízení). Obsahuje také rozhraní (API) pro komunikaci s jinými systémy (nebo napojení uživatelského rozhraní). Druhá (klientská neboli frontend) slouží k zobrazení dat a obecně interakci s uživatelem. Toto rozdělení nutně neznamená dvě různé samostatné aplikace, přestože k rozdělení nakonec došlo. Naproti tomu celá aplikace (obě části) může běžet jako jeden program na zařízení, z něhož se používá (stará se jak o přístup k síti a ukládání dat, tak o vykreslení uživatelského rozhraní).

4.1 Klientská aplikace

Aby byla klientská část aplikace dostupná z co nejširší škály zařízení (což z dnešního pohledu znamená stolní počítače s operačními systémy Windows, macOS a Linux, mobilní platformy se systémy Android, iOS a Windows Mobile – další platformy, které mají nízké zastoupení nebudou uvažovány), je možné využít několika možností vývoje.

- Vytvoření nativních aplikací pro každé prostředí, tato možnost je díky různorodosti nejnáročnější, protože vyžaduje vytvoření mnoha aplikací, na druhou stranu přináší uživatelům nejvyšší komfort; kvůli své náročnosti jak pro počáteční vývoj, tak pro další údržbu nebude o této variantě vůbec uvažováno.
- Další možností je využít nástrojů pro multiplatformní vývoj, například Xamarin pro .NET (s podporou jazyků C# a F#) umožňuje vytvoření jedné aplikace (včetně uživatelského rozhraní), která běží na Windows (od verze 8.1), Windows Mobile, Android (od verze 4.0.3) a iOS a s možností sdílet kód kromě uživatelského rozhraní také pro macOS (uživatelské rozhraní musí být vytvořené zvlášť).
- Poslední z variant je vytvoření webového rozhraní, které je přístupné ze všech platforem díky tomu, že v dnešní době každý běžný klientský počítač a mobil obsahuje webový prohlížeč.

Pro první dvě možnosti se jedná o samostatnou aplikaci, která se připojuje k API serverové části, třetí může být její součástí. Díky největšímu zastoupení webových prohlížečů a nejsnadnějšímu vývoji byla zvolena třetí variantu, pro níž lze rozhraní vytvořit dvěma rozdílnými způsoby.

- Webový prohlížeč slouží pouze pro zobrazení rozhraní, ale stránky se generují na straně serveru, s každou interakcí se posílá na klientský prohlížeč nová stránka (nebo její část). Takto funguje dnes většina webů a součástí .NET platformy je přesně pro tyto účely vytvořené ASP.NET (34) (standardní klasické nebo nové ASP.NET Core).

- Celá aplikace běží ve webovém prohlížeči, tento způsob se označuje jako „Single page application“ (často zkracováno jako „SPA“ (35)). Ze serveru se načte při přístupu na stránku celá aplikace, která pak interaguje se serverovou části přes API obdobně jako klasická desktopová aplikace. Serverová část tak slouží pouze jako úložiště načítaného programu a dat.

Volba padla na druhou možnost, protože poskytuje uživateli mnohem lepší zážitek z používání (rychlejší odezva), navíc disponuje dalšími pokročilými schopnostmi. Celou aplikaci lze např. používat i offline (po patřičné úpravě), může si lokálně ukládat data nebo z ní lze vytvořit desktopovou verzi.

4.1.1 Jednostránkové aplikace (SPA)

Existuje celá řada různých knihoven a frameworků pro vytváření SPA (35). Mezi hojně rozšířené patří jQuery (36), Angular (37), Ember (38) a React (39), existuje ale mnoho dalších. Zatímco jQuery je knihovna pro snadnou práci s DOM (vysvětleno dále v textu v následujícím odstavci), událostmi a AJAX (40), Angular a Ember jsou MVC (41) nebo MVVM (42) (dle použití (43)) frameworky, které poskytují kompletní řešení pro celou aplikaci (od práce s daty, modelem až po použití šablon pro vykreslení uživatelského rozhraní), React naproti tomu je knihovna pro tvorbu uživatelských rozhraní.

DOM (44) neboli „document object model“ je rozhraní pro reprezentaci HTML ve webovém prohlížeči. Každý prvek stránky (od zadávacího pole přes tlačítka až po textové odstavce) je reprezentován objektem, jednotlivé objekty se vzájemně zanořují (dle vlastních pravidel) a vytváří tak stromovou strukturu. Na začátku při načtení stránky prohlížeč z HTML vytvoří DOM, který následně vykreslí. Pro manipulaci s DOM objekty lze využít javascript (změna jednotlivých uzlů, přidávání, mazání). Tímto způsobem lze např. na základě výběru možnosti ve formuláři patřičně zobrazit další zadávací pole nebo měnit barvy atd. Pro usnadnění těchto operací (a zjednodušení pro různé verze různých prohlížečů) vznikla knihovna jQuery, kterou je vhodné použít hlavně pro oživení statických webových stránek interakcemi a animacemi (a pomoc s komunikací se serverem), nehodí se příliš pro komplexní webové aplikace. Pracuje přímo s prvky webové stránky (DOM), díky čemuž je mnohem pomalejší než moderní frameworky. Ve své době ale znamenala obrovský pokrok (vývojář nemusel řešit rozdílnosti jednotlivých prohlížečů) a dodnes se jedná o nejpoužívanější knihovnu na webových stránkách.

React popularizoval myšlenku tzv. „virtualdomu“. Zatímco prvky DOMu obsahují velké množství vlastností a navázaných událostí, prvky virtualdomu jsou jejich mnohem odlehčenou variantou (jedná se o objekty Javascriptu). Aplikace místo náročných a pomalých úprav DOMu celou stránku rychle vykreslí do virtualdomu, poté provede rozdíl původního virtualdomu (který je nyní převeden na DOM a zobrazován v prohlížeči) s novým (tato operace je taktéž rychlá díky použití vhodných algoritmů) a do pomalého DOMu provede pouze nezbytně nutné úpravy tak, aby odpovídal aktuálnímu virtualdomu. Tato technika umožňuje celou stránku vždy znovu rychle překreslit, autor se tak nemusí starat o složité inter-

akce a aktualizace částí webu (např. pro chatovací aplikaci, která kromě chatu zobrazuje i počet nepřečtených zpráv není nutné řešit, kde všude se počet nepřečtených zpráv zobrazuje, jednoduše se překreslí celá).

Nové myšlenky mezi frameworky a knihovny pro tvorbu SPA přináší práce „Elm: Concurrent FRP for Functional GUIs“ (45), jejíž součástí je i jazyk Elm (který se od doby práce dále rozvíjí), ale hlavně Elm architektura. Ta znamená další zjednodušení – model celé aplikace je uložen na jednom místě, pro zobrazení se použije funkce, která převede model na HTML (stejný model se vždy převede na stejné HTML), odpadá tak problém se synchronizací. Změny v modelu jsou vyvolány zprávami, které přijímá aktualizační funkce spolu s původním modelem a vytváří model úplně nový, který opět vstupuje do funkce pro zobrazení HTML. Kromě virtualdomu využívá Elm i další techniky (např. requestAnimationFrame (46)) pro efektivní a velmi rychlé renderování HTML.

Protože pro serverovou část je použit jazyk F#, nabízí se možnost ho využít i pro vývoj webového rozhraní SPA. Toto je možné díky projektu Fable (47), který převádí F# do Javascriptu. Pro tento projekt je navíc dostupná knihovna Elmish (48), která je postavená na základech Elm architektury a umožňuje využít libovolnou knihovnu nejen pro renderování HTML, ale i nativních aplikací (propojením s React Native (49)). Díky ohromné popularitě Reactu pro něj existuje velké množství užitečných nástrojů a knihoven, podporuje ho i Elmish pro renderování.

Pro vizuální stránku tvorby uživatelského rozhraní slouží další knihovny obsahující množství komponent pro zobrazení různých prvků na stránce (např. tlačítka, formulářová zadávací pole nebo navigaci) včetně designu (rozvržení) stránky samotné. Jednou z takových je Bootstrap (50), který je v projektu využíván. Je navržen od základů pro tvorbu responzivního (51) designu (stránka a jednotlivé prvky se přizpůsobují podle velikosti obrazovky).

Díky rozšířenosti Reactu jsou pro něj k dispozici i pokročilé ladící nástroje přímo v prohlížeči (např. „React Developer Tools“ (52), které zobrazí přímo v prohlížeči komponenty Reactu, přestože jsou ve výsledku renderování jako běžné DOM objekty). Dále je pro vývoj používán Webpack (53) (nejpoužívanější v komunitě Reactu), který obsahuje server pro vývoj, jež umožňuje tzv. „hot reload“ (při změně v kódu se automaticky synchronizuje i zobrazovaná stránka), což umožňuje rychlejší a snadnější vývoj. Knihovna Elmish navíc obsahuje podporu pro vývojářské nástroje, které jednotlivé zprávy zachytává a umožňuje se vracet ke kterýmkoliv dřívějším modelům (před zasláním zprávy), lze se tak vrátit k jakémukoliv stavu v aplikaci nebo naopak jednotlivé zprávy znovu „přehrát“.

4.2 Serverová část

Aby mohla aplikace (serverová část) komunikovat se zařízeními v síti, musí je mít přístupná. Toho lze dosáhnout dvěma způsoby. Buďto aplikace běží přímo uvnitř dané sítě nebo je k ní propojena pomocí VPN („virtual private network“ – slouží k připojení ke vzdálené síti odkudkoliv – koncové zařízení se

připojí k bodu, který má přístupnou požadovanou síť a tváří se, jako by bylo součástí vzdálené sítě). Z pohledu aplikace jsou obě možnosti rovnocenné (VPN se pro ni tváří transparentně).

Předpokladem je běh na některém z běžně používaných serverových systémů (Windows nebo Linux), resp. možnost běhu na kterémkoliv z nich. Existuje množství prostředí, která lze pro vytvoření této části využít. Z programovacích jazyků a prostředí bylo na výběr z několika možností:

- C++ – nativní aplikace pro obě platformy – standardně se zdrojové kódy programu kompilují na každé platformě zvlášť, ale je možné je zkompilovat i pouze na jedné z nich pro obě,
- Java – aplikace běží ve virtuálním stroji (JVM), který je dostupný pro obě platformy a velmi rozšířený, kompiluje se pouze jednou pro obě platformy,
- C# nebo F# – součástí .NET platformy – velmi podobné z obecného pohledu Javě – aplikace taktéž běží ve virtuálním stroji, který je součástí téměř každé instalace Windows, naopak na Linuxu není tak rozšířený jako JVM,
- Javascript – původně jazyk pro webové stránky s podporou ve webových prohlížečích, dnes (zejména díky Node.js) lze psát i serverovou část (dříve bylo taktéž možné, ale ne zdaleka tak rozšířené), díky čemuž velmi rychle roste na popularitě, umožňuje sdílet kód mezi serverovou a klientskou částí; díky rozšířenému frameworku Electron (54) je vhodný i pro vytváření multiplatformních desktopových aplikací, které vypadají na všech platformách totožně. Vývojáři používají stejný jazyk se stejnými nástroji pro vše – tvorbu webových, desktopových aplikací i serverové části

Vzhledem k dlouhé zkušenosti s prostředím .NET padla volba na něj a na jazyk F#.

4.2.1 Webový server

Kromě komunikace se síťovými zařízeními se serverová část musí postarat také o komunikaci s klientskou částí a vzhledem k tomu, že se jedná o webovou aplikaci, musí se také odněkud načítat. Ke splnění obou požadavků slouží právě webový server. Webový prohlížeč se k němu připojí, načte a zobrazí patřičnou aplikaci. V případě SPA potom dále se serverem komunikuje – posílá na něj a načítá z něj potřebná data (např. údaje o zařízení), která poté zobrazí uživateli.

4.2.1.1 Protokol HTTP

Komunikace s webovým serverem probíhá pomocí protokolu HTTP (55) (56) nebo jeho šifrované varianty HTTPS. Prohlížeč pošle požadavek (na dokument) na server a ten pošle zpět dokument (spolu se stavovým kódem o úspěchu). Pokud není z nějakého důvodu dokument vrácen, posílají se jiné stavové kódy – např. s informací, že byl dokument přemístěn (a kam) nebo že neexistuje. Stavové kódy pro různé situace jsou přesně definovány v RFC 7231 (57). Přes HTTP protokol probíhá standardně přenos všech součástí dokumentu (texty, obrázky, videa apod.).

4.2.1.2 Komunikace webového serveru s klientem

Protože HTTP požadavek iniciuje vždy klient a server na něj pouze odpovídá, nemůže server při změně právě zobrazovaných dat (např. obdržení odpovědi od spravovaného zařízení) poslat tuto zprávu klientovi. Pokud tedy probíhá nějaká úloha na serveru na pozadí a chceme její dílčí výsledky zobrazovat v prohlížeči, musí se prohlížeč neustále dotazovat na aktuální stav v určitém intervalu (např. 1 s), uživateli se tak zobrazují aktuální data a nemusí sám neustále obnovovat stránku. Tato možnost (označována jako „polling“) ovšem znamená zbytečné plýtvání prostředky (klient se neustále dotazuje na totéž a mnohdy dostává neustále stejnou odpověď).

Existují však techniky, jak dosáhnout možnosti zaslání dat ze serveru, aniž by se klient neustále dotazoval. Protokol HTTP nebyl navržen pro obousměrnou komunikaci, i proto starší metody „HTTP long polling“ a „HTTP streaming“ (58) s sebou přinášely mnohá omezení a neefektivní využívání prostředků. HTTP long polling funguje tak, že klient pošle požadavek, ale server spojení neuzavírá a čeká, až bude mít data k dispozici, poté je pošle a spojení uzavírá. Klient stejným způsobem opět navazuje nové spojení (buďto okamžitě nebo s krátkou prodlevou) a celý proces se s každými daty opakuje. HTTP streaming navazuje jedno spojení, přes které server posílá po celou dobu potřebná data.

Později vznikly další novější možnosti komunikace serveru s klientem. Na základech HTTP streamingu je založena specifikace Server-sent events (59) (klient naváže spojení a server mu může kdykoliv poslat zprávu, která se u klienta projeví ve formě události s daty). Bohužel tuto metodu nepodporují stále nezanedbatelně zastoupené prohlížeče Microsoftu (Internet Explorer a Edge) (60).

Relativně novou možností je protokol Websockets (61) (62), který umožňuje obousměrné zasílání zpráv. V případě, že se na serveru nějaká data změní, může být klientská aplikace ihned informována a uživateli tak zobrazit aktuální data s minimálním zpožděním, navíc nedochází k plýtvání prostředků – klientská aplikace

4.2.1.3 Autentizace

Jedná se o proces ověření identity uživatele, tj. zda je opravdu tím, za koho se vydává. Samotný protokol HTTP přímo podporuje a popisuje možnosti autentizace (63).

„Basic“ autentizace (64) spočívá v zasílání přihlašovacích údajů (uživatelské jméno a heslo) s každým požadavkem. Samotný prohlížeč vyzve uživatele k jejich zadání, běžně si údaje zapamatuje, aby je nemusel uživatel s každým načtením stránky znovu zadávat, a pak je při další komunikaci se serverem (při vstupu na určité stránky) automaticky posílá. Nevýhodou tohoto přístupu je, že se uživatel prakticky nemůže odhlásit a přihlašovací údaje se posílají s každým požadavkem (o tom, jak dlouho se údaje uchovají, rozhoduje prohlížeč – typicky se údaje mažou po zavření okna). Nelze tak např. vynutit automatické odhlášení po delší době nečinnosti. Z pohledu uživatelského prostředí aplikace není možné mít vlastní vzhled obrazovky pro přihlášení, opět se o vše stará prohlížeč (což na druhou stranu přináší

výhodu pro vývojáře, že nemusí formulář pro přihlášení vytvářet). Bezpečnější varianta autentizace je pomocí „HTTP Digest Access Authentication“ (65).

Další hojně využívanou možností autentizace je pomocí standardního HTML formuláře (66). Pro přihlášení je speciálně vytvořená stránka (nebo část) s formulářem pro zadání přihlašovacích údajů. Pokud uživatel přistupuje ke chráněné části webu, server jej přesměruje na přihlašovací stránku, uživatel zadá přihlašovací údaje a po zpracování odeslaných údajů (úspěšné autentizaci a autorizaci) získá k této části přístup. Vlastní řešení bývá často pomocí cookies (67) nebo předáváním vygenerovaného identifikátoru v každé adrese jako její parametr (všechny odkazy v rámci webu jej obsahují). Tento přístup ovšem přináší nevýhodu, pokud aplikace (webové stránky) přistupuje ke chráněnému zdroji z jiné domény (běží na jiné adrese, než odkud se načítají některá data). Je tak potřeba řešit bezpečnost z pohledu CORS (68) a CSRF (69). Nepraktické může být také připojení k takovému zdroji odjinud než z webového prohlížeče (např. mobilní nebo desktopová aplikace – vyžaduje práci s cookies kontejnery (70) pro uchování a odesílání cookies).

V aplikaci je pro své výhody používán *JSON web token* (71) se schématem „Bearer“ (72). Po odeslání přihlašovacích údajů (uživatelské jméno a heslo) je na serveru vytvořen token, který je vrácen do klientské aplikace, ta jej s dalšími požadavky přidává do HTTP hlavičky („Authorization: Bearer *token*“) a na serveru proběhne jeho ověření. Namísto přidání do hlavičky může být předán i jiným způsobem, např. přímo mezi daty požadavku. Token může také mj. obsahovat údaje o délce své platnosti. Pomocí tokenu lze přistupovat k jakémukoliv zdroji na serveru nebo i jiných serverech, bez ohledu na původní adresu, odkud se požadavek posílá, případně se pomocí něj přihlašovat i do jiných služeb. Protože není potřeba řešit, kde běží aplikace a k jakému zdroji se připojuje, lze snadno použít CDN (73) (síť serverů po celém světě s cílem „být co nejblíže uživateli“) pro rychlé načítání aplikace a poté se připojí k datům z jiného zdroje. Při použití tokenu na rozdíl od formulářů není potřeba pro relaci ukládat údaje na serveru (a s každým požadavkem je z úložiště načítat), token v sobě obsahuje všechny potřebné údaje a ověření probíhá pouze kontrolou podpisu.

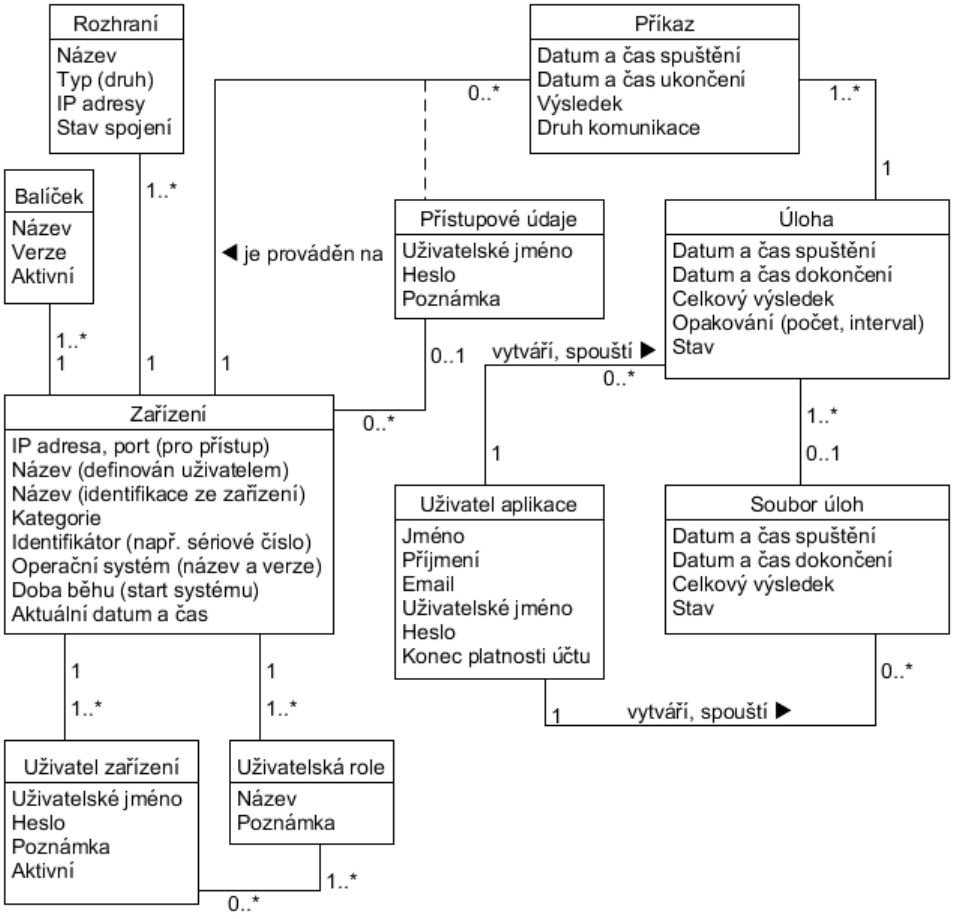
Pokud se používá standardní nešifrovaný protokol HTTP, putují i údaje pro autentizaci nešifrovaně a jsou náchylné na různé druhy útoků, proto je vhodné komunikaci zabezpečit (např. použitím HTTPS).

5 Analýza a návrh aplikace

Základem celé aplikace jsou dvě části, serverová a klientská. Serverová běží jako webový server, který poskytuje rozhraní klientské části a zároveň obstarává komunikaci se zařízeními a ukládání dat. Klientská část je vytvořena webovou aplikací, která se načítá z webového serveru a dále s ním komunikuje přes definované rozhraní.

5.1 Doménový model

Na obrázku jsou zachyceny entity a jejich vzájemné vazby. Obsaženy jsou pouze důležité entity bez pomocných vzniklých jako implementační detail.



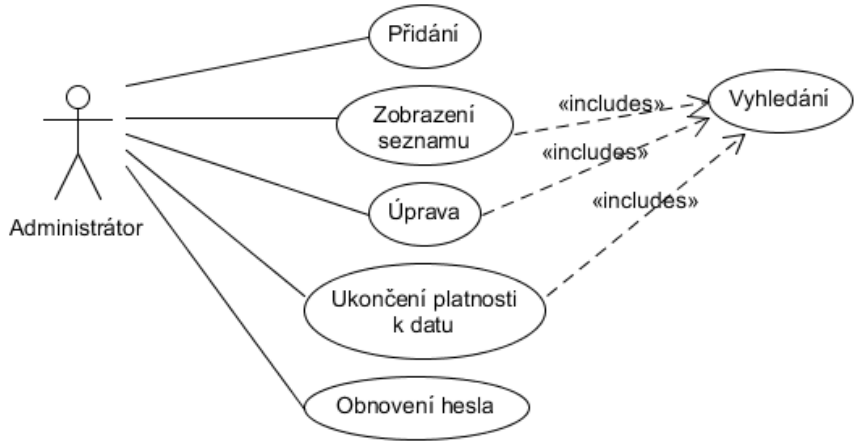
OBRÁZEK 5

5.2 Diagram případů užití

V diagramech jsou zachyceny některé zajímavé případy užití aplikace uživatelem. Rozděleny jsou dle logických celků podle entit, s nimiž uživatel interaguje.

Uživatel aplikace

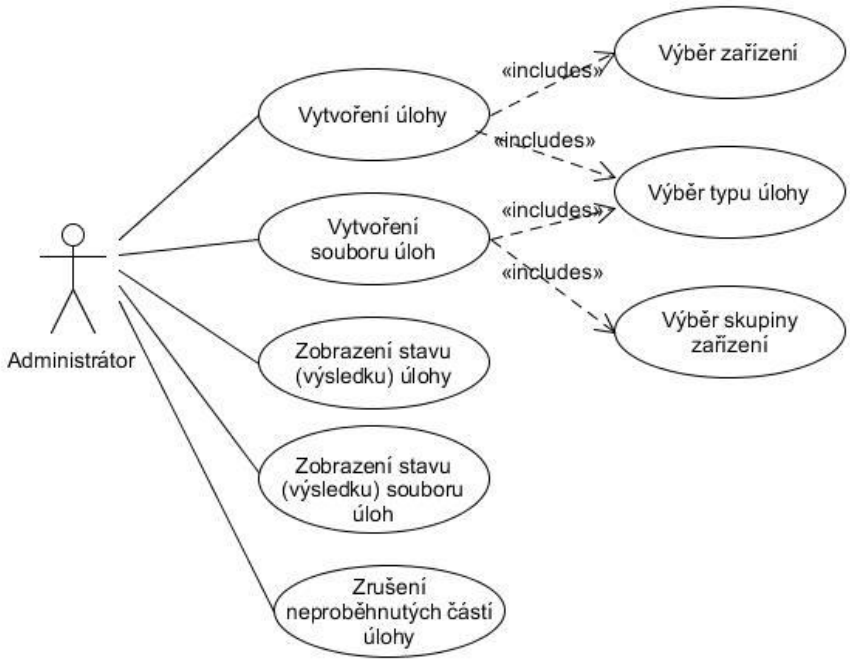
Na obrázku jsou znázorněny případy užití ve vztahu k uživateli aplikace. Kromě standardních operací (zobrazení, přidání, úprava) může být uživateli ukončena platnost jeho účtu a může si nechat obnovit zapomenuté heslo.



OBRÁZEK 6

Úloha

Uživatel při vytváření úlohy musí nejprve vybrat zařízení, pro které se bude úloha vykonávat, poté vybere typ úlohy a může ji vytvořit (spustit). V průběhu provádění úlohy nebo po jejím doknčení si může zobrazit aktuální stav. Pro skupinu úloh vybírá seznam zařízeních a typ úlohy (stejný typ je pro všechna zařízení).



OBRÁZEK 7

5.3 Ukládání dat

Aplikace potřebuje ke svému běhu ukládat různá data. Pokud mají mít uživatelé možnost se přihlašovat, musí vést jejich seznam včetně (hashovaných) hesel. Stejně tak informace o spravovaných zařízeních musí být uložena pro další načítání, zobrazení a práci s nimi.

Protože možností, jak ukládat data je mnoho (strukturovaný soubor, databáze SQL, NoSQL, grafová, objektová), a ještě více konkrétních implementací, poskytuje aplikace obecné rozhraní pro přístup k datům a přes něj se může napojit jakýkoliv zdroj dat (klidně se může jednat pouze o data uložená v operační paměti nebo souboru ve formátu JSON, XML, nebo jakémkoliv jiném). Samozřejmě je nutné dané napojení naprogramovat a namapovat konkrétní zdroj dat do aplikace.

5.4 Instalace, spuštění

Výslednou aplikaci lze přímo spustit a začít používat, jedná se o samostatný program, je vyžadována minimální konfigurace. Ve výchozím stavu webový server naslouchá na standardním portu 80 (který je možné konfigurovat) a data ukládá do SQL databáze. Zvolený ovladač SQLProvider (74) je univerzální a podporuje velké množství různých databází – např. MySQL, PostgreSQL, Oracle, SQLite. Testováno bylo s MS SQL Server (75). Databázi je potřeba před prvním použitím iniciovat pomocí skriptu (součást zdrojových kódů) pro vytvoření databázové struktury a naplnění počátečních dat. Aby mohla aplikace s databází komunikovat, je potřeba nastavit před spuštěním řetězec pro připojení k databázi.

5.4.1 Vývojářská verze

Pro sestavení aplikace ze zdrojových kódů stačí spustit skript „build“ s patřičným parametrem (popsány ve skriptu), podle něj dojde automaticky k sestavení a spuštění serverové a (nebo) klientské části v režimu naslouchání na změny ve zdrojových kódech s automatickým novým sestavením. Po něm je aplikace restartována, aby reflektovala provedené změny, případně zobrazovaná stránka obnovena. Všechny požadované součásti (knihovny) jsou automaticky staženy z internetu, především se jedná o balíčky z repozitáře Nuget (76).

5.4.2 Produkční verze

Na rozdíl od vývojářské verze obsahuje již všechny potřebné součásti pro běh ve zkompilované podobě, tzn. není potřeba žádných nástrojů na sestavení. Tato verze neobsahuje žádné ladící informace, je optimalizována pro rychlý běh a je určena k použití v reálném prostředí.

Budoucí rozšíření

Jednou z oblastí, kterou lze dále rozšiřovat je zobrazování různých druhů grafů. Kromě standardních sloupcových nebo spojnicových lze použít další kombinace (koláčový, paprskový) nebo vytvořit vlastní, stejně tak lze prozkoumat možnosti použití jiných měřítek, např. logaritmického. Zobrazování grafů samotné nabízí mnoho dalších příležitostí k vylepšování, které by vydaly na samostatnou práci, příkladem může být pokročilé ovládání myší (posunování po osách přetahováním, přiblížení nebo oddálení

kolečkem, zobrazení podrobností po najetí kurzorem nad hodnotu) nebo skládání a porovnávání různých hodnot různých zařízení do jediného grafu.

Rozšíření je také možné v oblasti notifikací na různé události – překročení hodnoty, dlouhodobé vytížení sítě apod.

Vzhledem k možnosti propojení na jiné systémy se taktéž nabízí možnost integrace nebo vytvoření dalších klientů kromě webového. Jako nejužitečnější (pro nejvíce potencionálních uživatelů) se jeví klasická desktopová nebo mobilní aplikace.

Závěr

Práce se zabývá možnostmi správy sítě s mnoha zařízeními. Protože referenční síť sestává z většiny ze zařízení s operačním systémem RouterOS, zaměřuje se právě na tento systém. Nejdříve prozkoumává současné nástroje se zjištěním, že většina z nich se zaměřuje na monitoring a nastavení umožňují pomocí SNMP, které je bohužel vzhledem k požadavkům nevhodné, protože neposkytuje v zařízeních se systémem RouterOS veškeré možnosti nastavení. Pro připojení k jednotlivým zařízením byly využity protokoly telnet a SSH.

Uživatelské rozhraní je vytvořeno s cílem maximální jednoduchosti a intuitivního ovládání tak, aby nebyl potřeba žádný manuál k použití.

Reference

1. **MIKROTIK SIA.** MikroTik Routers and Wireless: Software. [Online] [Citace: 3. leden 2017.] https://mikrotik.com/software.

2. **Mozilla Corporation.** API | MDN. [Online] [Cited: prosinec 20, 2018.] https://developer.mozilla.org/en-US/docs/Glossary/API.

3. **Postel, J.** RFC 792 - Internet Control Message Protocol. [Online] [Citace: 22. prosinec 2017.] https://tools.ietf.org/html/rfc792.

4. **Preston-Werner, Tom.** *Semantic Versioning 2.0.0 | Semantic Versioning.* [Online] [Cited: prosince 20, 2018.] https://semver.org/.

5. **aj., Jeffrey Case.** RFC 1157 - Simple Network Management Protocol (SNMP). [Online] [Citace: 28. leden 2018.] https://tools.ietf.org/html/rfc1157.

6. **MIKROTIKLS SIA.** Manual:SNMP - MikroTik Wiki. [Online] [Citace: 28. leden 2018.]

7. **Postel, J. a Reynolds, J.** RFC 854 - Telnet Protocol Specification. [Online] [Citace: 3. únor 2017.] https://tools.ietf.org/html/rfc854.

8. **Ylonen, T. a Lonvick, C.** The Secure Shell (SSH) Protocol Architecture. [Online] [Citace: 12. leden 2017.] https://www.ietf.org/rfc/rfc4251.txt.

9. **MIKROTIKLS SIA.** Manual:TOC - MikroTik Wiki. [Online] [Cited: listopad 5, 2017.] https://wiki.mikrotik.com/wiki/Manual:TOC.

10. **Shafranovich, Yakov.** RFC 4180 - Common Format and MIME Type for Comma-Separated Values (CSV) Files. [Online] [Cited: 11 5, 2017.] https://tools.ietf.org/html/rfc4180.

11. **Close, Josh.** CsvHelper. [Online] [Cited: 11 05, 2017.] http://joshclose.github.io/CsvHelper/.

12. **Petricek, Tomas and Guerra, Gustavo.** F# Data: CSV Parser. [Online] [Cited: 11 05, 2017.] http://fsharp.github.io/FSharp.Data/library/CsvFile.html.

13. **MIKROTIKLS SIA.** Manual:Scripting - MikroTik Wiki. [Online] [Cited: listopad 5, 2017.] https://wiki.mikrotik.com/wiki/Manual:Scripting.

14. —. Manual:API - MikroTik Wiki. [Online] [Cited: leden 28, 2018.] https://wiki.mikrotik.com/wiki/Manual:API.

15. **Javorek, Honza.** GitHub - honzajavorek/cs-apis: List of Czech & Slovak public APIs. [Online] [Cited: prosinec 20, 2018.] https://github.com/honzajavorek/cs-apis.

16. **Zapier Inc.** Zapier | The easiest way to automate your work. [Online] [Cited: prosinec 18, 2018.] https://zapier.com/.

17. **Integromat s.r.o.** Integromat - Automatizujeme Vaši práci. [Online] [Cited: prosinec 20, 2018.] https://www.integromat.com/cs.

18. **MIKROTIKLS SIA.** API command notes - MikroTik Wiki. [Online] [Cited: prosinec 18, 2018.] https://wiki.mikrotik.com/wiki/API_command_notes.

19. **Frantik, Daniel.** High level API with O R mapper · danikf/tik4net Wiki · GitHub. [Online] [Cited: prosinec 20, 2018.] https://github.com/danikf/tik4net/wiki/High-level-API-with-O-R-mapper.

20. **MIKROTIKLS SIA.** The Dude. [Online] [Citace: 3. leden 2017.] https://mikrotik.com/thedude.

21. —. Manual:The Dude v6/Probes - MikroTik Wiki. [Online] [Citace: 3. ledna 2017.] https://wiki.mikrotik.com/wiki/Manual:The_Dude_v6/Probes.

22. —. Manual:The Dude v6/Installation - MikroTik Wiki. [Online] [Citace: 3. leden 2017.] http://wiki.mikrotik.com/wiki/Manual:The_Dude_v6/Installation.

23. **Lonvick, Chris.** RFC 3164 - The BSD Syslog Protocol. [Online] [Cited: prosinec 20, 2018.] https://tools.ietf.org/html/rfc3164.

24. **Gerhards, Rainer.** RFC 5424 - The Syslog Protocol. [Online] [Cited: prosinec 20, 2018.] https://tools.ietf.org/html/rfc5424.

25. **NetCore j.s.a.** *Unimus.* [Online] [Cited: leden 03, 2019.] https://unimus.net/.

26. **Nubera eBusiness S.L.** IT, Server & Network Monitoring Software 2019 - Best Application Comparison | GetApp®. [Online] [Cited: prosinec 20, 2018.] https://www.getapp.com/it-management-software/it-server-network-monitoring/.

27. **OpenNMS Group, Inc.** *Enterprise Grade Network Management.* [Online] OpenNMS Group, Inc. [Citace: 22. květen 2016.] http://www.opennms.org/.

28. **GNS3 Technologies Inc.** GNS3 | The software that empowers network professionals. [Online] [Citace: 11. prosinec 2016.] https://gns3.com/.

29. **Burger, Thomas Wolfgang.** Intel® Virtualization Technology for Directed I/O (VT-d): Enhancing Intel platforms for efficient virtualization of I/O devices | Intel® Software. [Online] [Cited: listopad 26, 2017.] https://software.intel.com/en-us/articles/intel-virtualization-technology-for-directed-io-vt-d-enhancing-intel-platforms-for-efficient-virtualization-of-io-devices.

30. **VMware, Inc.** VMware Workstation Player - VMware Products. [Online] [Citace: 11. leden 2017.] http://www.vmware.com/products/player/playerpro-evaluation.html.

31. **GNS3 Technologies Inc.** Software - VM Download - GNS3. [Online] [Citace: 8. únor 2017.] https://gns3.com/software/download-vm.

32. **Wireshark Foundation.** Wireshark · Go Deep. [Online] [Citace: 11. leden 2017.] https://www.wireshark.org/.

33. **GNS3 Technologies Inc.** Marketplace - Appliances - GNS3. [Online] [Citace: 11. leden 2017.] https://gns3.com/marketplace/appliances.

34. **Microsoft Corporation.** *ASP.NET | The ASP.NET Site.* [Online] [Citace: 3. únor 2017.] https://www.asp.net/.

35. **Code School LLC.** Single-page Applications | Code School. [Online] [Citace: 10. prosinec 2017.] https://www.codeschool.com/beginners-guide-to-web-development/single-page-applications.

36. **The jQuery Foundation.** *jQuery.* [Online] [Citace: 3. únor 2017.] http://jquery.com/.

37. **Google Inc.** *One framework. - Angular.* [Online] [Citace: 3. únor 2017.] https://angular.io/.

38. **TILDE INC.** *Ember.js - A framework for creating ambitious web applications.* [Online] [Citace: 3. únor 2017.] http://emberjs.com/.

39. **Facebook Inc.** *A JavaScript library for building user interfaces - React.* [Online] [Citace: 3. únor 2017.] https://facebook.github.io/react/.

40. **Dria.** Ajax - Web developer guides | MDN. [Online] [Cited: prosinec 9, 2017.] https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX.

41. **Mills, Chris.** MVC architecture - App Center | MDN. [Online] [Cited: prosinec 9, 2017.] https://developer.mozilla.org/en-US/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture.

42. **Microsoft Corporation.** The MVVM Pattern. [Online] [Citace: 9. prosinec 2017.] https://msdn.microsoft.com/en-us/library/hh848246.aspx#sec4.

43. **Google Inc.** Angular Docs. [Online] [Cited: prosinec 9, 2017.] https://angular.io/guide/template-syntax#template-syntax.

44. **Mozilla Corporation.** Introduction to the DOM - Web APIs | MDN. [Online] [Citace: 9. prosinec 2017.] https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction.

45. **Czaplicki, Evan.** Elm: Concurrent FRP for Functional GUIs. 2012.

46. **Shepherd, Eric.** window.requestAnimationFrame() - Web APIs | MDN. [Online] [Cited: prosinec 9, 2017.] https://developer.mozilla.org/en-US/docs/Web/API/window/requestAnimationFrame.

47. **Garcia-Caro, Alfonso.** Fable: JavaScript you can be proud of! [Online] [Citace: 10. prosinec 2017.] http://fable.io/.

48. **Tolmachev, Eugene.** Elmish. [Online] [Citace: 11. prosinec 2017.] https://fable-elmish.github.io/elmish/.

49. **Facebook Inc.** React Native · A framework for building native apps using React. [Online] [Citace: 10. prosinec 2017.] https://facebook.github.io/react-native/.

50. **Twitter, Inc.** [Online] [Citace: 10. prosinec 2017.] http://getbootstrap.com/.

51. **Michálek, Martin.** *Vzhůru do (responzivního) webdesignu.* s.l. : Michálek Martin - Vzhůru dolů, 2017. 978-80-88253-00-6.

52. **Facebook Inc.** GitHub - facebook/react-devtools: An extension that allows inspection of React component hierarchy in the Chrome and Firefox Developer Tools. [Online] [Citace: 10. prosinec 2017.] https://github.com/facebook/react-devtools.

53. **Koppers, Tobias.** webpack. [Online] [Citace: 10. prosinec 2017.] https://webpack.js.org/.

54. **GitHub, Inc.** Electron - Build cross platform desktop apps with JavaScript, HTML, and CSS. [Online] [Citace: 3. únor 2017.] http://electron.atom.io/.

55. **Roy Fielding, Julian Reschke.** Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. [Online] [Citace: 13. prosinec 2017.] https://tools.ietf.org/html/rfc7230.

56. **další, Mike Belshe a.** RFC 7540 - Hypertext Transfer Protocol Version 2 (HTTP/2). [Online] [Citace: 17. prosinec 2017.] https://tools.ietf.org/html/rfc7540.

57. **Fielding, Roy a Reschke, Julian.** RFC 7231 - Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. [Online] [Citace: 16. prosinec 2017.] https://tools.ietf.org/html/rfc7231#section-6.

58. **Loreto, Salvatore, a další.** RFC 6202 - Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP. [Online] [Citace: 17. prosinec 2017.] https://tools.ietf.org/html/rfc6202.

59. **WHATWG.** HTML Standard. [Online] [Citace: 17. prosinec 2017.] https://html.spec.whatwg.org/multipage/server-sent-events.html#server-sent-events.

60. **Deveria, Alexis.** Can I use... Support tables for HTML5, CSS3, etc. [Online] [Citace: 17. prosinec 2017.] https://caniuse.com/#search=Server-sent%20events.

61. **Fette, Ian a Melnikov, Alexey.** RFC 6455 - The WebSocket Protocol. [Online] [Citace: 17. prosinec 2017.] https://tools.ietf.org/html/rfc6455.

62. **WHATWG.** HTML Standard. [Online] [Citace: 17. prosinec 2017.] https://html.spec.whatwg.org/multipage/web-sockets.html.

63. **Fielding, Roy a Reschke, Julian.** RFC 7235 - Hypertext Transfer Protocol (HTTP/1.1): Authentication. [Online] [Citace: 22. prosinec 2017.] https://tools.ietf.org/html/rfc7235.

64. **Reschke, Julian.** RFC 7617 - The 'Basic' HTTP Authentication Scheme. [Online] [Citace: 17. prosinec 2017.] https://tools.ietf.org/html/rfc7617.

65. **Shekh-Yusef, Rifaat, Ahrens, David a Bremer, Sophie.** RFC 7616 - HTTP Digest Access Authentication. [Online] [Citace: 22. prosinec 2017.] https://tools.ietf.org/html/rfc7616.

66. **aj., Yutaka Oiwa.** RFC 8053 - HTTP Authentication Extensions for Interactive Clients. [Online] [Citace: 22. prosinec 2017.] https://tools.ietf.org/html/rfc8053.

67. **Barth, Adam.** RFC 6265 - HTTP State Management Mechanism. [Online] [Citace: 22. prosinec 2017.] https://tools.ietf.org/html/rfc6265.

68. **Kesteren, Anne van.** Cross-Origin Resource Sharing. [Online] [Citace: 29. leden 2018.] https://www.w3.org/TR/cors/.

69. **The Open Web Application Security Project.** Cross-Site Request Forgery (CSRF) - OWASP. [Online] [Citace: 29. leden 2018.] https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF).

70. **Microsoft.** CookieContainer Class (System.Net). [Online] [Citace: 31. leden 2018.] https://msdn.microsoft.com/en-us/library/system.net.cookiecontainer%28v=vs.110%29.aspx?f=255&MSPPError=-2147217396.

71. **Jones, Michael, Bradley, John a Sakimura, Nat.** RFC 7519 - JSON Web Token (JWT). [Online] [Citace: 22. prosinec 2017.] https://tools.ietf.org/html/rfc7519.

72. **Jones, Michael a Hardt, Dick.** RFC 6750 - The OAuth 2.0 Authorization Framework: Bearer Token Usage. [Online] [Citace: 22. prosinec 2017.] https://tools.ietf.org/html/rfc6750.

73. **Jahoda, Bohumil.** K čemu slouží CDN? [Online] [Citace: 31. leden 2018.] http://jecas.cz/cdn.

74. **McKinlay, Ross.** SQLProvider. [Online] [Citace: 24. leden 2018.] http://fsprojects.github.io/SQLProvider/index.html.

75. **Microsoft.** SQL Server 2016. [Online] [Citace: 24. leden 2018.] https://www.microsoft.com/cs-cz/sql-server/sql-server-2016.

76. —. NuGet Gallery | Home. [Online] [Citace: 22. prosinec 2017.] https://www.nuget.org/.