Správa síťových prvků v kontextu nástrojů Flowmon

Filip Dostálík

Bakalářská práce
2021

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav automatizace a řídicí techniky

Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Filip Dostálík**
Osobní číslo:	**A18205**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Forma studia:	**Kombinovaná**
Téma práce:	**Správa síťových prvků v kontextu nástrojů Flowmon**
Téma práce anglicky:	**Asset Management in the Context of Flowmon Utilities**

Zásady pro vypracování

1. Prostudujte a popište aktuální architekturu systému Flowmon a jeho nástrojů.
2. Popište požadavky uživatelů na správu síťových prvků Flowmon.
3. Navrhněte architekturu nového modulu správy síťových prvků.
4. Implementujte nový modul pro správu síťových prvků, včetně zajištění komunikace s okolními moduly a vytvoření databáze.
5. Implementujte unit testy pro zajištění kontinuálního testování funkčnosti nového modulu při dalším vývoji.

Forma zpracování bakalářské práce: **Tištěná/elektronická**

Seznam doporučené literatury:

1. ČELEDA, Pavel, Milan KOVÁČIK, Tomáš KONÍŘ, Vojtěch KRMÍČEK, Petr ŠPRINGL a Martin ŽÁDNÍK. FlowMon Probe. B.m.: PB tisk, s.r.o., 2007. ISBN 978-80-239-9285-4.
2. ZHANG, T., L. LINGUAGLOSSA, M. GALLO, P. GIACCONE a D. ROSSI. FlowMon-DPDK: Parsimonious Per-Flow Software Monitoring at Line Rate. In: 2018 Network Traffic Measurement and Analysis Conference (TMA): 2018 Network Traffic Measurement and Analysis Conference (TMA). 2018, s. 1–8. ISBN 978-1-5386-7152-8.
3. KAMISIŃSKI, Andrzej a Carol FUNG. FlowMon: Detecting Malicious Switches in Software-Defined Networks. 2015. DOI:10.1145/2809826.2809833
4. REHÁK, M., K. BARTOŠ, M. GRILL, J. STIBOREK a M. SVOBODA. Monitoring sítí pomocí NetFlow dat – od paketů ke strategiím. 2009. Plzeň: Vydavatelský servis, 2009. s. 75-81. ISBN 978-80-86583-17-4.
5. ELICH, Martin, Matěj GRÉGR a Pavel ČELEDA. Monitoring of Tunneled IPv6 Traffic Using Packet Decapsulation and IPFIX (Short Paper). In: Jordi DOMINGO-PASCUAL, Yuval SHAVITT a Steve UHLIG, ed. Traffic Monitoring and Analysis [online]. Berlin, Heidelberg: Springer, 2011, s. 64–71. Lecture Notes in Computer Science. ISBN 978-3-642-20305-3.

Vedoucí bakalářské práce: **Ing. Tomáš Dulík, Ph.D.**
Ústav informatiky a umělé inteligence

Konzultant bakalářské práce: **Mgr. Jaromír Viteker**
Ústav automatizace a řídicí techniky

Datum zadání bakalářské práce: **15. ledna 2021**
Termín odevzdání bakalářské práce: **17. května 2021**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
DĚKAN
L.S.

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

prof. Ing. Vladimír Vašek, CSc. v.r.
ředitel ústavu

Ve Zlíně dne 15. ledna 2021

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářské práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky. Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně Filip Dostálík v.r.

podpis autora

ABSTRAKT

Tato práce analyzuje přístup k síťovým prvkům v kontextů monitorovacích nástrojů Flowmon. V první části práce se zabývám teoretickými postupy, které jsou použity při vývoji. V druhé části práce analýzou a návrhem databáze pro nový modul správy síťových prvků a v poslední části práce popisuji implementaci modulu.

Klíčová slova: Flowmon, Netflow, PHP, SQL, síť, multitenance

ABSTRACT

This thesis analyzes asset management in the contexts of Flowmon monitoring tools. The first part of this thesis is theoretical procedures that are used in development. In the second part of the work I analyze and design the database model for the new asset management module. The last part contains implementation details of the module.

Keywords: Flowmon, Netflow, PHP, SQL, network, multitenancy

PODĚKOVÁNÍ

Rád bych poděkoval doktoru Dulíkovi za vedení této bakalářské práce. Dále bych chtěl poděkovat Mgr. Jaromíru Vitekerovi nejen za odborné konzultace, ale i za všechny roky společné práce ve firmě Flowmon. S tím souvisí i poděkování firmě Flowmon Networks za umožnění práce na tomto tématu. Také děkuji Ing. Petru Hromádkovi za odborné konzultace. Nakonec bych rád poděkoval celé moji rodině, za důvěru a podporu v průběhu celého studia.

OBSAH

ÚVOD

Rychlý vývoj informačních technologií a přenos služeb do cloudových prostředí vyvíjí tlak na zprostředkovatele služeb monitorování a bezpečnosti počítačových sítí. Cílem těchto služeb je rychle odhalit anomálie, analyzovat zatížení sítě, identifikovat kybernetické útoky a předejít ztrátě dat či omezení provozu sítě.

Nástroje, které pomáhají s monitorováním a bezpečností sítě, nemusí být vždy intuitivní a snadné na používání. Uživatel potřebuje dobře znát síť, kterou monitoruje. Také musí dobře znát síťové protokoly a musí umět využít jejich vlastnosti k získání požadovaných výstupních dat z monitoringu a analýzy sítě. Tyto znalosti musí mít také k tomu, aby dokázal monitorovací software správně nastavit.

Řešení Flowmon [1] disponuje možností monitorování sítě, detekcí anomálií a ochranou před kybernetickými útoky. Flowmon můžeme rozdělit na sondu, která sbírá a exportuje data ve formátu Netflow/IPFIX [2], a kolektor, který data ukládá ke zpětné analýze. Flowmon již v dnešní době poskytuje mnoho automatických nástrojů jak monitorování a analýzu síťového provozu zjednodušit. Nevýhodou je, že k využití plného potenciálu produktu Flowmon, je nutné mít alespoň základní znalost v oblasti síťových technologií.

Pro snížení nutnosti mít technické zázemí vznikl cíl modernizovat existující prvky produktu Flowmon za účelem přiblížení správy sítě i méně technicky orientovaným uživatelům. Čím více uživatelů bude schopných pracovat s tímto nástrojem, tím více se zvýší šance na úspěšné zjištění anomálií a poruch v síti.

Z tohoto důvodu vznikla tato práce, jejímž cílem je navrhnout flexibilnější a lépe nastavitelný prostředek pro správu síťových prvků, než jakým nyní nástroje Flowmon disponují. Cílem práce je vytvořit nový model reprezentující síť, kterým v systému Flowmon výhledově nahradíme profilový model. Tento nový model jsme pojmenovali „Asset model“, kde pojem „Asset“ představuje abstrakci některého se síťových prvků v monitorované topologii. Podrobnější popis tohoto pojmu je možné nalézt v analytické části práce, viz kapitola 8.1.

V teoretické části práce se zabývám vysvětlením pojmů, které jsou později v práci použity. Dále se zaměřuji na princip komunikace komponent systému Flowmon pomocí protokolu AMQP a na metodiku testování, kterou jsem využil při vývoji. Poslední bod teoretické části je popis, jakými prvky mohou zákazníci pracovat při analýze dat v produktu Flowmon. Architektura produktu Flowmon je velmi rozsáhlá a zahrnuje komponenty a technologie, které nejsou součástí této bakalářské práce. Při popisu architektury jsem se zaměřil pouze na ty části, které jsem sám implementoval nebo které jsou pro tuto práci důležité, což je analýza dat a monitorování síťového provozu.

Na začátku analytické části práce popisuji uživatelské požadavky na správu síťových prvků. Definuji typové uživatele, kteří s produktem pracují a popisuji způsob práce s původním řeše-

ním využívající profilového modelu. Z tohoto poté plynule přecházím do možných vylepšení, které „Asset model“ přinese.

Jádrem analytické části je návrh nové architektury. Obsahuje konkrétní návrhy, jak řešit problémy s viditelností sítě, jako jsou vyhledávání, vlastní atributy a obecně zjednodušení práce s produktem. Tato část splývá s předchozím bodem zadání. Analýza architektury obsahuje také návrh tříd a komunikaci mezi nimi. Tento návrh je důležitý pro implementaci.

V poslední části práce se zabývám popisem implementace prototypu a jeho testování pomocí technologií Docker a PHPUnit. Je zde popsána také struktura projektu a jeho spuštění.

I. TEORETICKÁ ČÁST

1 Vysvětlení pojmů

Některé pojmy použité v této práci nemají vhodný překlad nebo se běžně nepřekládají a je nutné je vysvětlit.

1.1 Asset

Popis pojmu „Asset“ je důležitý pro základní pochopení kontextu této bakalářské práce a nástrojů Flowmon. Detailnější popis toho, jak Asset ve Flowmon chápeme, je možné najít v analytické části této práce, a to v kapitole 8.1.

Asset představuje abstrakci logické části sítě, která zapouzdří data a události. Každý asset má svůj daný typ v závislosti na tom, jakou část sítě představuje. Asset může představovat celou řadu síťových prvků. Na nejvyšší úrovni to může být celá síť, na nižších úrovních může představovat podsítě, servery nebo síťové aplikace. Tyto prvky jsou obrazem skutečné monitorované topologie.

1.2 Tenant a multitenance

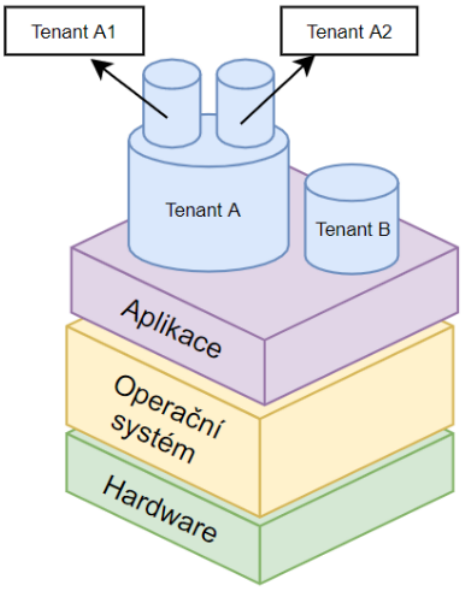
Jelikož jádro nástroje Flowmon umožňuje tzv. „multitenanci“, je důležité s tím počítat v analýze a zároveň tento pojem i vysvětlit. Multitenance je způsob implementace softwaru, kdy více nezávislých jednotek jedné aplikace operuje ve sdíleném prostředí. Taková jednotka se nazývá „Tenant“. Tenant-y jsou odděleny logikou, ale zároveň jsou spojeny fyzicky. Oddělení logikou musí být striktní, ale v rámci fyzického oddělení nemusí být přesně nastavené hranice. Hlavní výhodou multitenance je zvýšení účinnosti aplikace nad stejným hardware. Můžeme například pronajímat jednu aplikaci více zákazníkům odděleně, bez toho aniž by o sobě navzájem věděli. [3]

1.3 Widget

„Widget“ je část aplikace fungující na straně klienta, která umožňuje kompaktní vizualizaci dat a dává tím pohled do systému na nejabstraktnější úrovni. Widget může mít mnoho podob, nejčastější v kontextu vizualizace síťových toků zobrazuje widget průběhový graf, tabulku nebo jejich kombinaci. Předpokládaný pracovní postup je takový, že uživatel nejdříve vidí základní pohled ve widgetu a až poté ho analyzuje detailněji v specializovaných nástrojích (ve Flowmonu je to například nástroj DDoS Defender).

1.4 Report

Překlad pojmu report může být v kontextu této bakalářské práce zavádějící. Report se překládá do češtiny jako zpráva, ale protože v této práci je použit pojem zpráva v kontextu předávání



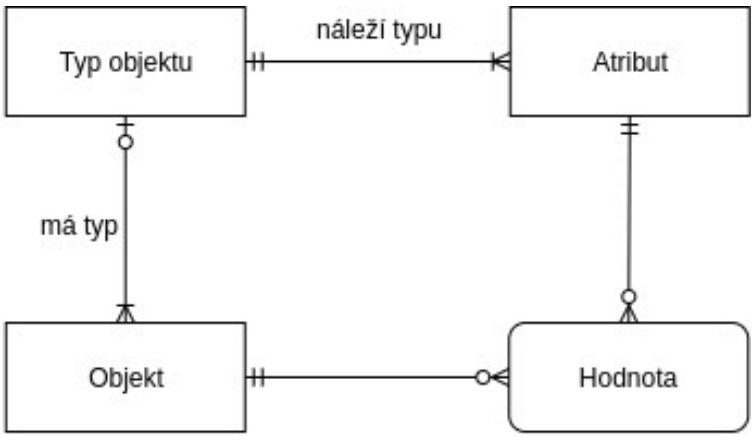
Obr. 1.1 Vizualizace multitenance v systému

událostí mezi moduly systému Flowmon, mohou se tyto dva pojmy snadno zaměnit. Z tohoto důvodu budu v textu dále používat pojem report v následujícím významu.

Report je v systému Flowmon záznam dat za určitý časový úsek, který má za cíl předat zákazníkovi náhled na monitorovanou síť. Report není určen k hlubší analýze dat, ale pouze k základní identifikaci a případně prezentaci problému v síti.

2 Princip abstrakce atributů

Při vytváření databázového modelu vznikl požadavek na zadávání vlastních atributů[1)] uživatelem, který je důležitým základem celkového modelu vytvořeného v analytické části této práce. Jedná se o abstrakci atributů. Dle tohoto principu má každý objekt svůj typ, každý typ objektu vlastní množinu atributů a hodnoty těchto atributů jsou tvořeny vazbou mezi objektem a atributem. [4]



Obr. 2.1 Princip abstrakce atributů

[1)] definovaná vlastnost objektu, elementu nebo souboru

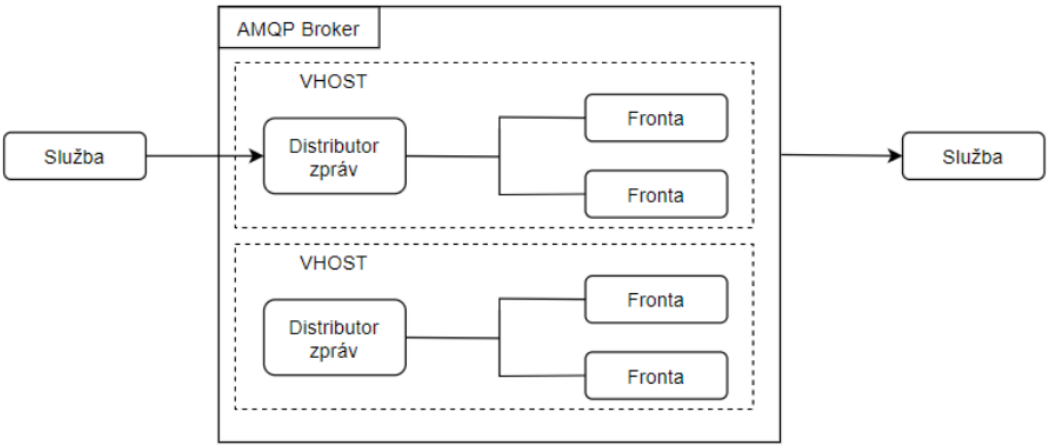
3 Systém zasílání zpráv

Systém zasílání zpráv umožňuje propojení mezi softwarovými komponentami nebo aplikacemi. Aplikace mohou být mezi sebou propojeny jako komponenty většího systému. V systému Flowmon využíváme asynchronní zasílání zpráv pomocí protokolu AMQP a jeho open-source implementaci RabbitMQ. [5]

AMQP (Advanced Message Queuing Protocol) je protokol aplikační vrstvy, který se soustředí na komunikaci mezi procesy[1)] přes síť. Umožňuje spolu komunikovat různým serverům nezávisle na použité technologii. AMQP obsahuje soubor standardů, které řídí proces zasílání zpráv přes tzv. AMQP broker (například RabbitMQ). Obsahuje také pravidla určující jak budou zprávy přenášeny a uloženy. [6]

3.1 AMQP model

Na obrázku 3.1 lze vidět princip zasílání zpráv. Klient odešle zprávu distributorovi zpráv. AMQP Broker zapouzdřuje jednotlivé distributory zpráv, kteří mají za úkol zkopírovat zprávu do jednotlivých front. Řídí se přitom pravidly, které definuje daná zpráva. Nakonec je zpráva z fronty zpracována cílovou službou (konzumentem).



Obr. 3.1 AMQP model

3.2 Komponenty AMQP

3.2.1 Fronta zpráv

Slouží pro uchovávání zpráv, které budou následně zpracovány. Při vytvoření fronty můžeme specifikovat její chování jako například automatické mazání a exkluzivitu. Exkluzivita zna-

[1)] process-to-process

mená, že frontu může používat výhradně jedno připojení a tato fronta bude smazána jakmile toto připojení skončí.

3.2.2 Distributor zpráv

Kanál přesměrovává zprávy do front podle toho, jaký mají typ a vazbu mezi distributorem zpráv a frontou. Každá fronta musí mít alespoň jednoho distributora zpráv. Při vytvoření může mít distributor zpráv deklarované atributy, které mohou zajistit například odolnost (nesmazatelnost) nebo automatické smazání pokud zaniknou všechny jeho fronty.

3.2.3 Zpráva a její obsah

Každá zpráva musí mít definovanou hlavičku. Hlavička zprávy obsahuje informaci o délce života zprávy a prioritě. Každá zpráva musí obsahovat také jméno.

3.2.4 Kanál

Kanál je virtuální spojení mezi dvěma koncovými službami. Jedno připojení může operovat ve více kanálech.

3.2.5 VHOST

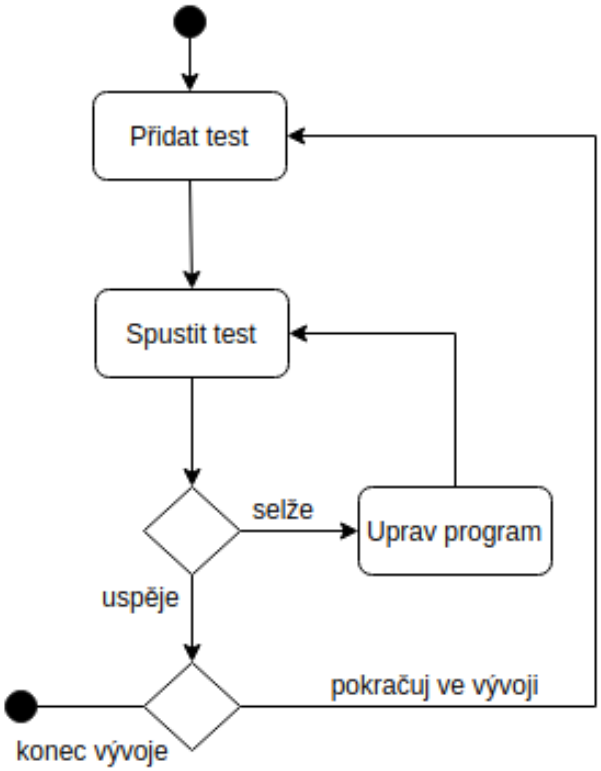
VHOST (Virtual host) umožňuje vytvářet oddělené prostředí uvnitř AMQP brokeru. Uživatelům mohou být nastavena práva k přístupu k jednotlivým instancím VHOST. Fronty a distributoři zpráv jsou tvořeni uvnitř jednotlivých VHOST.

4 Vývoj řízený testy

Kromě funkční aplikace je cílem vývoje softwaru, aby kód, který programátor píše, byl dlouhodobě udržovatelný a testovatelný. K tomuto slouží různé metody vývoje, které programátorům pomáhají organizovat práci.

Vývoj řízený testy (test-driven development, dále jen TDD) je metoda vývoje softwaru, při které se vytvoří soubor požadavků na daný software a ten je následně převeden na testovací scénáře[1]. Toto se děje předtím, než je software dokončen. [7]

Mezi výhody TDD patří např. podpora vývoje kvalitního kódu, snadnější implementace uživatelských požadavků, snížení počtu chyb v softwaru nebo vytváření důkazů, že software funguje. Nevýhodou je, že ne vždy je možné určit, zda testy dostatečně pokrývají požadovanou část aplikace. [8]



Obr. 4.1 Postup při vývoji

Programátor nejprve přidá nový testovací scénář. Poté spustí všechny testy. Nový test selže, protože ta část kódu, kterou testuje ještě nebyla implementovaná. Po selhání programátor naprogramuje logiku aplikace, která vyhovuje novému testovacímu scénáři, který selhal. Poté spustí všechny testy. Jakmile všechny testy uspějí, může se programátor vrátit k přidání nového testovacího scénáře. [7]

[1] test cases

5 Technologie Docker

Docker je nástroj pro virtualizaci, který se vyznačuje tvorbou kontejnerů. Jde o zjednodušený virtuální stroj. Kromě funkce, která umožňuje kontejnery vytvářet, nabízí pomoc tvorby aplikací uvnitř kontejnerů. [9]

Při vývoji je důležité, aby byl vývojář schopný simulovat prostředí, které nejvíce odpovídá produkčnímu prostředí. Tím lze předejít mnoha chybám ještě před vydáním produktu.

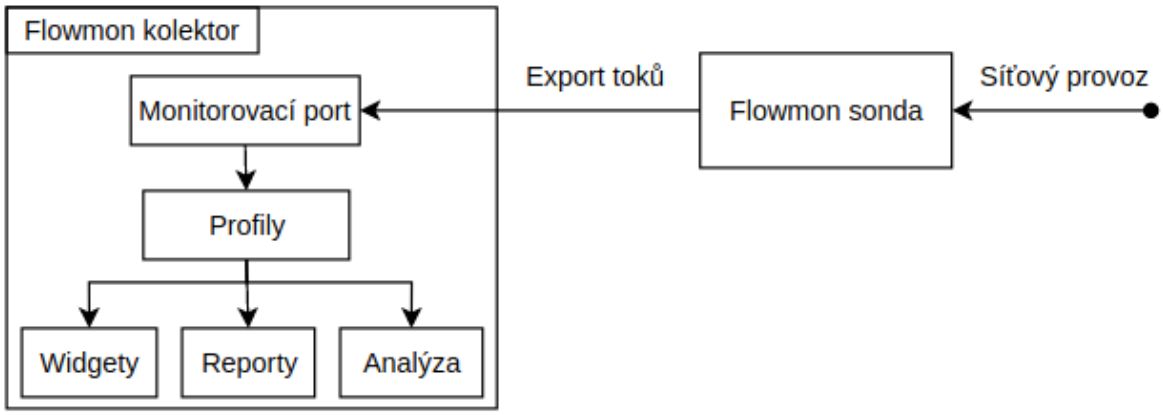
Na rozdíl od virtuálního stroje, nemají kontejnery velkou paměťovou náročnost, protože neobsahují celou kopii operačního systému. Další výhodou kontejnerů je možnost aktualizovat jednotlivé verze aplikací, které obsahují, nezávisle na již vytvořených kontejnerech. Cílem je používat více kontejnerů, které obsahují pouze potřebnou část z operačního systému a tím ušetřit paměťové a výpočetní prostředky.

Jednotlivé kontejnery je možné přesouvat mezi počítači s různými operačními systémy. [9]

Pro vytvoření kontejneru je nutné spustit program Docker ve složce, která obsahuje konfigurační soubor. Tento konfigurační soubor se jmenuje Dockerfile a obsahuje veškeré informace nutné k vytvoření kontejneru. Nejdůležitější informací v konfiguračním souboru je definice virtuálního obrazu, ze kterého kontejner vychází. Obraz může být celý operační systém (Ubuntu, CentOS) nebo pouze jedna aplikace (PostgreSQL, PHP interpret). Dockerfile může obsahovat další příkazy, které mohou měnit nastavení a instalovat aplikace uvnitř kontejneru.

6 Technologie Flowmon

Technologie Flowmon slouží k zachytávání, zpracování, ukládání a k analýze síťového provozu. Flowmon obsahuje řadu nástrojů jako jsou ochrana proti DDOS útokům[1)], detekce anomálií v síti[2)], monitorování výkonu služeb[3)] a další. Jednotlivé nástroje jsou dostupné formou rozšíření, které je možné po zakoupení příslušné licence nainstalovat na jádro Flowmon.



Obr. 6.1 Architektura Flowmon

Produkt Flowmon je server založený na systému CentOS. Produkt můžeme rozdělit na dvě části, sondu a kolektor. Flowmon sonda zajišťuje zpracování síťového provozu do formy toků ve formátu Netflow/IPFIX a jejich exportování na Flowmon kolektor. Flowmon kolektor tyto toky přijímá a ukládá pro pozdější analýzu. Na jeden kolektor můžeme připojit více sond. Kolektor musí mít velkou kapacitu disků a doporučuje se zapojení do RAID[4)], kvůli zabezpečení proti selhání pevného disku.

Předmětem této bakalářské práce je především Flowmon kolektor a konkrétně jeho část, která se zabývá monitorováním a analýzou. Tato část se nazývá Flowmon Monitoring Center a slouží k ukládání a analýze toků.

6.1 Profilový model

Ve stávajícím modelu správy systému Flowmon, který nazýváme „profilový model“, se uložená data sbírají do entit zvaných profily. [10]

Tyto entity zapouzdřují data a mají na sebe navázaný filtr. Profily se uskupují do stromové struktury. Každé zanoření může zprostředkovávat detailnější pohled na síť. Profily lze vytvořit v různých podobách, které specifikují styl ukládání dat:

[1)] Flowmon DDoS Defender
[2)] Flowmon ADS
[3)] Flowmon APM
[4)] https://cs.wikipedia.org/wiki/RAID

- stínový profil zaznamenává pouze metadata nutné pro vykreslení grafů,
- reálný profil zaznamenává veškerá data,
- ukončený profil již nesbírá data, ale historická data obsahujem
- neukončený profil periodicky sbírá data.

6.1.1 Kanály

Profil se skládá z kanálů do kterých přicházejí data. Každý profil obsahuje minimálně jeden kanál. Kanály jsou odvozeny od jednoho nebo více kanálů v rodičovském profilu. Kanál je definován filtrem, barvou a pořadím, v jakém se má vykreslovat v grafu. Kanály mohou obsahovat grafy znázorňující síťový provoz. [11]

6.1.2 Filtry

Na uložená data může uživatel aplikovat filtry, které umožní zobrazit detail komunikace v závislosti na zdrojovém nebo cílovém hostiteli, protokolu, počátku komunikace a mnoha dalších parametrech, které umožňuje technologie síťových toků. Syntaxe filtru je podobná knihovně pcap využívané programem tcpdump. [12]

```
((proto tcp) or (proto udp)) and (port "smtp")
```

Obr. 6.2 Ukázka filtru

6.1.3 Přehled

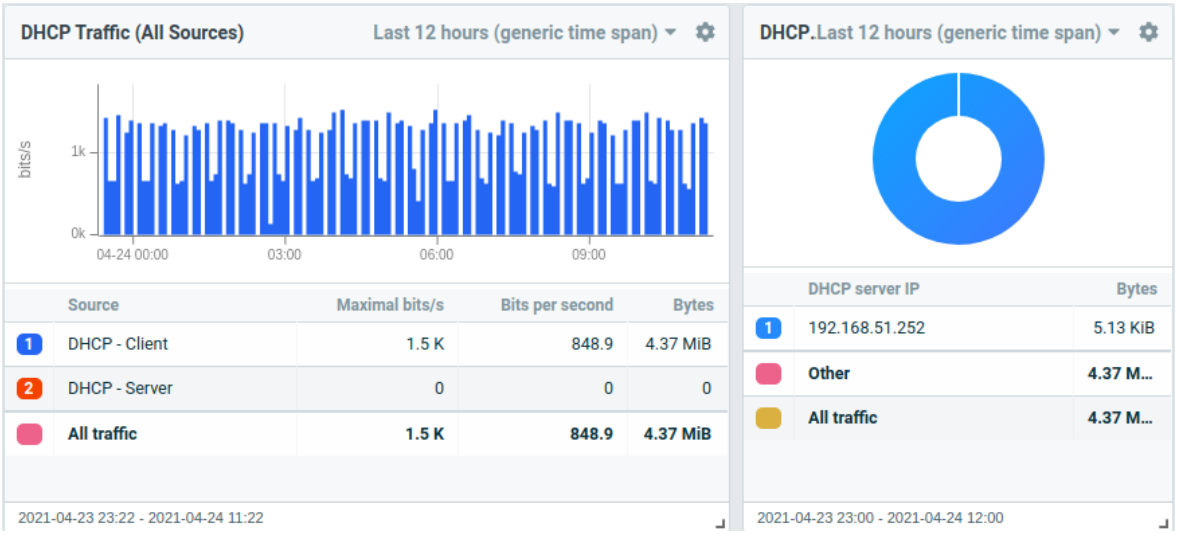
Aby uživatel mohl analyzovat data z profilů, potřebuje mít prostředek, který je zobrazí. K tomu slouží časový graf, ve kterém je možné označit zkoumanou oblast a provést detailní analýzu.

Nejenže uživatel může manuálně analyzovat síťový provoz z grafů a filtrovaných statistik, ale je možné některé činnosti analýzy a monitoringu automatizovat. Mezi automatizované nástroje patří zasílání reportů a upozornění, které mohou automaticky reagovat na podezřelý provoz nebo výskyt anomálií.

6.2 Manažerské nástroje

Další důležitou skupinou nástrojů je Flowmon Dashboard. Tak jako analýza, slouží Flowmon Dashboard jako náhled na data. Na rozdíl od analýzy se jedná o abstraktnější pohled určený pro méně technicky založené pracovníky. Flowmon Dashboard nabízí dvě hlavní funkcionality, widgety a reporty.

Widget slouží k náhledu na kratší časový úsek a obsahuje možnost přesměrování na detailnější analýzu. Skládá se převážně z grafů a statistik, které si uživatel může vytvořit podle potřeby.



Obr. 6.3 Flowmon Dashboard obsahující widgety

Report je entita, která může být převedena do formátu PDF nebo CSV. Do těchto formátů se report generuje při několika příležitostech. Jednou z nich je generování pomocí uživatelského požadavku z grafického rozhraní. Druhou metodou je nastavení automatického zasílání naplánovaného reportu na email.

Každý report obsahuje skupinu widgetů, které zobrazují data. Jsou to stejné widgety, jaké je možné nalézt na Flowmon dashboardu. Vygenerovaný report obsahuje widgety, které zobrazují data v předem nastaveném časovém intervalu.

Pro generování reportů z grafického rozhraní stačí, když uživatel nastaví časový rozsah a poté vygenerované reporty stáhne. Při automatickém generování, je nutné vytvořit rozvrh, který specifikuje příjemce, rozsah dat, intervaly opakování a formáty, ve kterých se mají reporty odeslat.

6.3 Cíl nového modelu

Pro použití profilového modelu musí mít uživatel dostatečnou technickou znalost síťových technologií. Cílem této práce je proto vytvořit jiný, nový model, který bude pochopitelný nejen pro síťové analytiky, ale i pro uživatele, kteří nemají detailní znalosti síťových technologií. Může se tím rozšířit počet uživatelů, kteří jsou schopni s produktem Flowmon pracovat a tím se i může rozšířit počet potenciálních zákazníků.

II. ANALYTICKÁ ČÁST

7 Požadavky uživatelů na správu síťových prvků

Než bude možné navrhnout finální model správy síťových prvků, je nutné nejprve prozkoumat požadavky uživatelů. Je potřeba zjistit, jak uživatelé s produktem pracují nyní a jaké jsou možnosti pro zjednodušení a vylepšení jejich práce. Konkrétně je důležité zjistit, jaké jsou možnosti vylepšení a zjednodušení v oblasti správy dat a monitoringu síťového provozu.

V předchozí kapitole je popsána nynější implementace funkcí uživatelského rozhraní, které pomáhají uživatelům monitorovat jejich síť. Tuto stávající implementaci nazýváme „profilový model“. Reprezentace sítě pomocí profilů není ideální, jelikož profily představují hodně technický pohled na data, především z perspektivy filtrování a tvorby monitorovaných entit. Jednotlivé profily nepředstavují konkrétní prvky sítě, jsou příliš anonymní na to, aby uživatel mohl jednoduše identifikovat funkčnost a účel, za kterým byl daný profil vytvořen. Běžný uživatel, který se dívá na svoji reálnou monitorovanou síť, vidí fyzické propojení jednotlivých prvků sítě, které má určitou topologii. Zobrazení této topologie mu ale profilové uspořádání nenabízí. Cílem nového modelu proto je přidat do systému Flowmon pohled reálné monitorované topologie sítě a přenést jej do zobrazení, správy dat a správy síťových prvků.

Pro získání požadavků, které mají uživatelé na nový způsob správy síťových prvků, který jsme nazvali „assetový model“, je dobré popsat pracovní postup, který uživatelé využívají při monitorování sítě.

Základním požadavkem uživatelů je umět rozdělit data mezi jednotlivé tenant-y. Případ užití je takový, že zákazník vlastní firmu, která má více různých poboček. Každá pobočka má přiřazena data síťového provozu bez toho, aniž by viděla síťový provoz ostatních poboček. Tento přístup usnadní práci hlavnímu správci sítě, který dokáže rozložit prostředky mezi jednotlivé pobočky. Tato rozdělená data poté mohou spravovat správci sítě na jednotlivých pobočkách. Přidělená data jednotlivým správcům sítě na pobočkách nebudou stačit, jelikož se jedná o čistý, téměř nefiltrovaný provoz. Úkolem správce je roztřídit a filtrovat tato data na menší díly, které se budou lépe analyzovat.

V současném řešení správce sítě musí vytvořit profily, které zapouzdří data, která na pobočku přichází. Musel při tom dbát na to, aby použil správné filtry, které mu vytvoří logické oddělení od ostatních profilů. Je tedy důležité, aby znal všechny síťové protokoly a filtrační pravidla, které jsou cílem jeho monitorovacího procesu. Pokud by všichni uživatelé produktu měli tyto znalosti, tak by nebylo potřeba oprošťovat se od aktuálního profilového modelu.

Správci sítě nejsou jediní, kteří mají zájem podílet se na fungování síťové infrastruktury firmy. Firma může mít zaměstnance na pozicích analytik, kteří mají za úkol hlídat síťový provoz, analyzovat a upozornit na anomálie. Dalším typem lidí, kteří by měli mít zájem na fungování sítě jsou lidé na manažerských pozicích, kteří nepotřebují detailní informace o fungování sítě, ale stačí jim pouze report nebo widget, které jim zprostředkuje správce nebo

analytik. Reporty a widgety jsou svoji podstatou velmi podobné assetovému přístupu k datům, ale stále je nutné manuálně tvořit profily, ze kterých reporty a widgety získávají data.

Profilový model se hůře udržuje, kvůli jeho složitosti. Problém nastává v situaci, kdy zkušený správce sítě má kolegy, kteří nemají tak velký přehled o spravované síti jako on. V takovém případě je nutné, aby nezkušené kolegy zaškolil nebo jim dal na starost pouze malou část sítě. Další problém nastává v situaci, kdy je správce sítě nemocný nebo změní zaměstnání. Noví zaměstnanci, kteří nastoupí na jeho pozici, stráví velmi mnoho času s průzkumem profilového modelu.

Tento způsob práce je pro uživatele velmi náročný a potenciálně může odradit nové zákazníky. Z tohoto důvodu vznikla myšlenka uživatelům práci usnadnit pomocí nového modelu, jehož správa nebude navržena pouze pro technicky založené uživatele. Vytvořený model bude reprezentovat síťovou topologii a konkrétní typy prvků v síti, bez nutnosti složité tvorby.

7.1 Typy uživatelů

V předchozím odstavci již bylo naznačeno, jaké typy uživatelů v oblasti monitorování a správy sítě pracují. Nyní je vhodné přiřadit jednotlivým uživatelům funkce, které specifikují jejich oblast zájmu v monitorování sítě.

Základními typy uživatelů, kteří pracují s produktem jsou manažer, analytik a správce sítě. Správce sítě je definován podrobnou znalostí nad síťovým provozem a síťovou topologií. Správce je schopný přiřadit jednotlivé sítě tenantům a rozdělit síť na podsítě a služby pomocí filtrů. Analytika lze definovat jako méně znalého síťových principů, ale dostatečně znalého pro práci s monitorovacími nástroji a analýzou dat. Jako manažera můžeme označit uživatele, který potřebuje získávat informace pouze na nejvyšší úrovni v nejjednodušší formě, kterou docílí použitím reportů a widgetů.

7.2 Role uživatelů při monitorování sítě

Zpracování a nastavení monitorované topologie začíná u správce sítě. Ten má za úkol rozdělit síťový provoz do jednotlivých tenantů. V jednotlivých tenantech chce rozdělit monitorovanou síť tak, aby se v ní analytik dobře vyznal. Docílí toho tím, že vybere ze skutečné topologie prvky, které chce monitorovat a pro každý prvek vytvoří asset. K tomu bude potřebovat několik základních stavebních prvků. Síťové prvky v topologii jsou různých typů a toto se projeví i v abstraktním modelu. Na nejvyšší úrovni je celá síť, kterých může být v jednom tenantu více. V tenantu tedy správce vytvoří pro každou síť jeden asset, který je typu síť. Sítě může poté dále rozdělit na další díly, které představují další prvky monitorované topologie. V jedné ze sítí může vzniknout požadavek na monitorování zaměstnanců, kteří vlastní počítače umístěné v podsíti již vytvořeného assetu. Správce tedy vytvoří asset typu podsíť, který bude vycházet z assetu typu síť a bude zahrnovat pouze data zaměstnanců v podsíti. Toto zanoření nemusí

být finální a správce může identifikovat v podsíti konkrétní server, který bude chtít sledovat zvlášť, pak pro takový prvek sítě vytvoří nový asset a tím pádem i další zanoření.

Jakmile má správce vytvořen abstraktní model skutečné topologie se zaměřením na jednotlivé síťové prvky, o které se zajímá v kontextu monitorování sítě, může začít analyzovat síť a nebo předat vytvořený model svým kolegům analytikům. Analytik poté nepotřebuje znát přesné zanoření vůči reálné síti, ale může dostat přidělený pouze konkrétní prvek sítě. Tím pádem nemusí mít analytik tak moc velké znalosti síťových technologií a stačí mu znalosti pro práci s konkrétním assetem. Základní práce analytika může být analýza přijatých dat a prověřování anomálií. Jakmile analytik nalezne anomálii nebo jiný problém, tak může pro jeho řešení kontaktovat správce sítě a vytvořit report s nalezeným problémem. Správce by měl problém vyřešit a vedení firmy poté získat ty nejdůležitější informace z připraveného reportu.

Uživatel v manažerské pozici přijímá informace od svých podřízených, ale sám může mít vytvořený widget a report, který mu zprostředkuje alespoň okrajový pohled na problémy v monitorované síti. Tím, že má vytvořen svůj obecný přehled, tak nemusí příliš zasahovat do práce svých zaměstnanců, kteří ho v ideálním případě kontaktují, až jakmile nastane v síti nějaký problém. Tento princip pomůže s eliminací mikromanagementu[1)].

7.3 Jednotlivé požadavky uživatelů

Z předchozí podkapitoly vyplývá, že správce sítě bude vytvářet jednotlivé prvky v asset modelu a připravovat strukturu představující monitorovanou topologii, která se bude do sebe zanořovat. Jednotlivé entity představují prvky sítě a jejich společné vlastnosti. Tyto entity se dají rozdělit podle typů síťového prvku. V základní konfiguraci je výhodné nabídnout uživateli seznam předdefinovaných typů. Tyto předdefinované typy se vyznačují tím, že se budou u více zákazníků opakovat a jejich před-vytvoření ušetří zákazníkům čas. Tyto již existující typy mohou sloužit pro zákazníka jako vzor pro vlastní typy. Monitorovací topologie každého zákazníka se může v detailech lišit, a každý zákazník může mít různé typy síťových prvků. Z tohoto důvodu by měl být správce sítě schopný vytvořit vlastní typy síťových prvků. Asset a jeho typy jsou popsány v kapitole 8.1.

Asset bude při zanoření tvořit dědičnou stromovou strukturu. Jednotlivé assety, které budou zanořeny, budou mezi sebou vykazovat jisté podobnosti na logické úrovni. Pokud uživatel vytvoří asset typu síť a poté ji rozdělí na podsítě, tak bude chtít danou síť specifikovat popiskem, IP adresou nebo nějakou jinou vlastností. Tyto vlastnosti mohou být společné i pro jednotlivé podsítě, které vznikly rozdělením síťového assetu. Vlastnosti zanořených prvků sítě se začnou překrývat. Při přidělování atributů je možné využít zanoření. Uživateli se tím ulehčí práce přenosem těchto vlastností na nižší úrovně zanoření. Tento princip je z hlediska

[1)] https://cs.wikipedia.org/wiki/Mikromanagement

návrhu popsán v kapitole 8.2.

Uživatel nemusí zůstat pouze u zanoření do dvou až tří úrovní. Cílem může být monitorování komunikace konkrétních síťových aplikací. Asset reprezentující tyto aplikace může být zanořen i více než čtyři úrovně od vrcholu stromu. V jednom tenantu je také možné mít více sítí, které mohou být zanořené do libovolné hloubky. S větším počtem síťových prvků může nastat problém s přehledností. Tento problém řeší způsob vyhledávání, který je popsán v kapitole 8.3.

„Asset model“ vytvořený podle existující topologie je pouze kostru obsahující zapouzdřené entity. Uživatel musí mít možnost vidět a analyzovat data. V profilovém modelu jsou k dispozici grafy, na které jsou uživatelé zvyklí. V novém modelu se použije stejný princip zobrazení dat. Více o zobrazování dat je možné najít v kapitole 8.1.

Při dlouhodobé práci s assety se může stát, že jeden z uživatelů provede s assetem takovou úpravu, která asset poškodí. Může jít o změnu filtru nebo odebrání některého atributu. V takovém případě může dojít i k úplnému znehodnocení assetu. Takovou úpravu uživatel nemusel udělat úmyslně. Řešením tohoto problému může být zaznamenávání aktivity. Každá změna provedená v assetu se projeví zaznamenáním informací, které pomohou ke zpětnému dohledání změn. Jestliže nějaký uživatel poškodí asset, tak správce sítě nejprve prověří záznamník akcí a až poté se může přesunout k prozkoumání problémů v síti. Tímto postupem se dá snížit zatížení zákaznické podpory produktu firmy Flowmon.

Toto jsou pouze základní požadavky, které mohou uživatelé klást na model správy síťových prvků, které budou stačit pro vytvoření základního prototypu. Další požadavky se dají získat zpětnou vazbou od konkrétních zákazníků, kteří si nový modul vyzkouší po vydání první verze.

8 Základní požadavky na „Asset model“

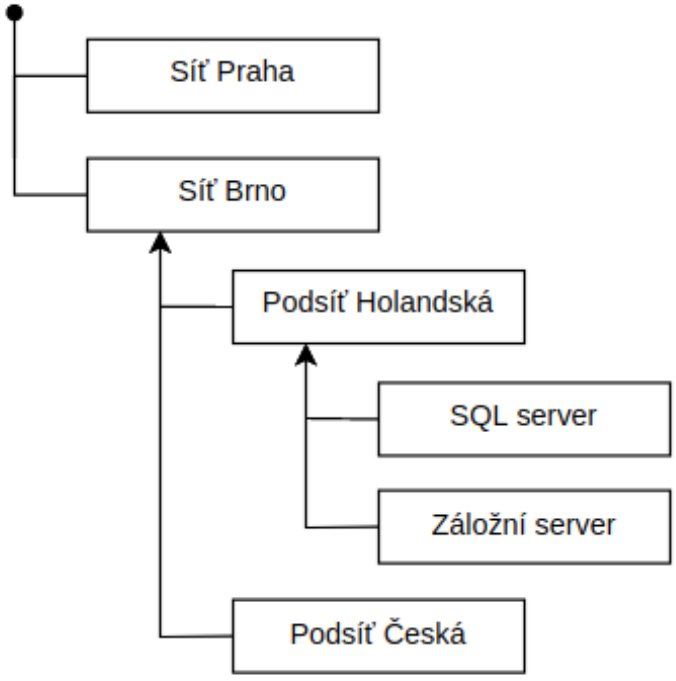
Nyní je potřeba vytvořit relace a navrhnout model, podle požadavků od uživatelů.

Uživatel má za cíl vytvořit asset, který bude reprezentovat monitorovanou část sítě. Asset při správné konfiguraci značně zjednoduší přehled nad monitorovanou sítí.

8.1 Základ modelu

Assety tvoří stromovou strukturu. Na nejvyšší úrovni jsou assety představující celou síť. Nové assety mohou být potomky již existujících assetů nebo mohou být na nejvyšší úrovni a představovat novou síť. Tvorbou potomků vzniká stromová struktura, jejímž cílem je získat detailnější pohled na zkoumanou část sítě.

Data se přidělují na nejvyšší úrovni stromové struktury. Na této úrovni se nachází kořenový „root asset“. Root asset je charakteristický tím, že nemá rodičovský asset.

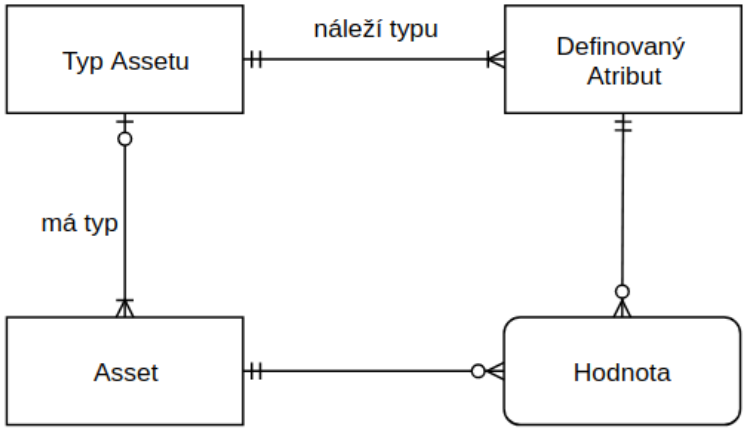


Obr. 8.1 Stromová struktura assetů

Asset můžeme rozdělovat podle typů. Typ assetu by měl definovat specifickou část naší sítě. Základními typy, které jsou již předdefinované, jsou například síť, podsíť a služba.

Předdefinované typy nemusí stačit a uživatel musí mít možnost vytvořit i vlastní typy. Typ assetu definuje konkrétní asset a jelikož každý asset bude jiný v závislosti na tom, jaký prvek sítě znázorňuje, bude také jiný každý typ assetu. Jednotlivé typy assetů mohou mít různé atributy.

Pokud má uživatel možnost vytvořit si vlastní typ, musí mít také možnost nastavit pro asset i vlastní atributy. Typ assetu se tedy stává nosičem atributů, ale samotné hodnoty těmto atributům nastavuje až konkretní asset. V tomto místě je možné využít princip abstrakce atributů. Tento princip je v abstraktní formě uveden v kapitole 2. Na obrázku 8.2 je možné vidět konkrétní použití principu abstrakce atributů.



Obr. 8.2 Relace mezi atributy a assety

8.1.1 Zobrazení dat

V aktuálním přístupu k datům se toky zobrazují v časových grafech, ze kterých lze analyzovat daný provoz. Asset bude tyto grafy uchovávat v entitách zvaných „pohledy“, jelikož jsou to pohledy na data. Pohled na skupinu dat není omezen pouze grafem, ale dají se zde definovat i „top n“ statistiky[1)].

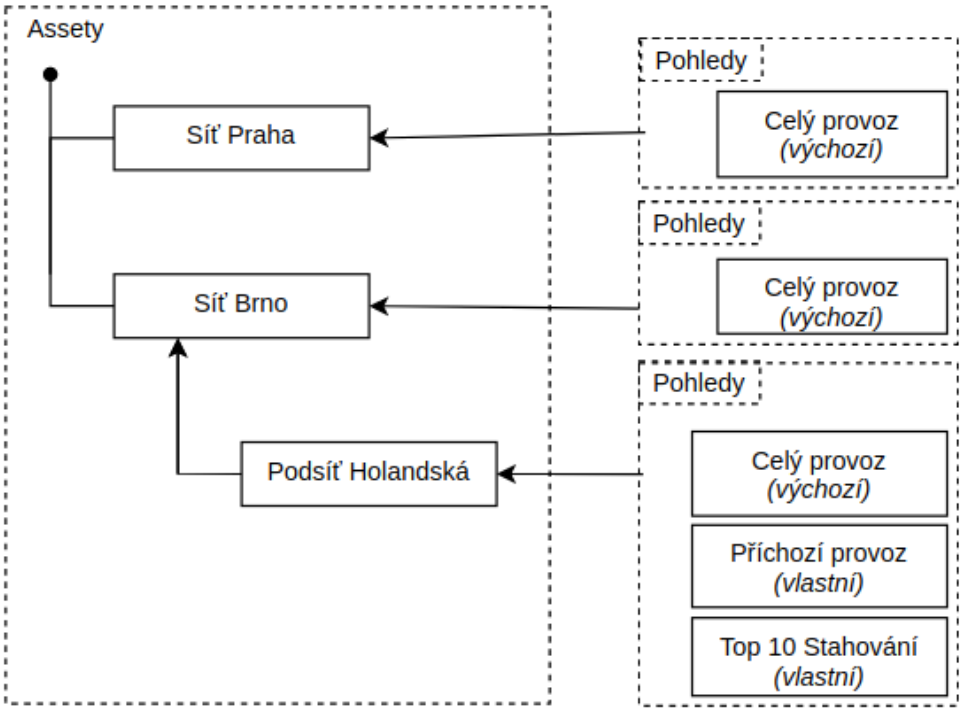
Každý asset má po vytvoření automaticky vytvořen pohled, který zahrnuje veškerý provoz tohoto assetu. Na tento základní pohled je aplikován pouze hlavní filtr, který má asset nastaven.

Uživatel si může vytvořit další filtry, které jsou omezeny hlavním filtrem, který asset obsahuje a filtrem, který si uživatel může přidat pomocí atributů[2)]. Pokud chce uživatel použít více filtrů v jednom pohledu, vybere filtry, které chce použít, a ty se spojí do jednoho pomocí operandu AND.

8.1.2 Zdroje dat a jejich filtrace

„Asset model“ má uživatelům pomáhat se správou dat a proto je potřeba vytvořit pravidla, jakými se budou data rozdělovat mezi jednotlivé assety.

[1)] seřazený seznam, který vznikl dle zadané metriky (např. „TOP 10 Hostnames in asset“)
[2)] více v následující kapitole „Zdroje dat a jejich filtrace“



Obr. 8.3 Ukázka pohledů navázaných na assety

Po instalaci nového modulu se automaticky vytvoří asset reprezentující všechny zdroje dat. Tento asset, stejně jako všechny ostatní assety, které jsou vytvořeny uživatelem, obsahuje pohled zobrazující veškerý provoz. Tento asset slouží jako základní pohled na data a nelze jej smazat. Práva na tento asset má pouze administrátor v nejvyšším tenantu. Uživatel nemůže dělat v tomto assetu jakékoliv úpravy. Pokud dojde ke smazání zdroje toků, tak asset zanikne.

Po připojení nového zdroje toků vyšle Flowmon zprávu, kterou přijme „Asset model“. Tato zpráva obsahuje informace potřebné k vytvoření nového pohledu reprezentujícího zdroj dat. Zdroj dat může mít přiřazen pouze „root asset“, další assety ve stromu už jen dědí tento zdroj dat a upraví jej filtrem.

8.1.3 Chování v multitenanci

Jelikož jsou zdroje toků již součástí multitenance a zároveň jsou zdrojem dat pro „Asset model“, musí být asset také součástí multitenance.

Nově vytvořený asset je přiřazen právě aktivnímu tenantu. Tuto akci zajistí jádro Flowmonu zprávou, která zajišťuje vytvoření assetu. „Asset model“ nepotřebuje mít informaci o tom, který tenant je aktivní. Administrátor tenantu poté může asset přiřadit některému ze subtenantů. Z toho vyplývá, že je nutné, aby asset nesl informaci o tenantu, který jej vlastní, a o tenantech, do kterých byl přiřazen.

Dědičnost assetů poté závisí na tom, zda je budoucí rodič dostupný v aktuálním tenantu.

To znamená, že pokud budeme chtít vytvořit nový asset, jeho rodič musí být vlastněn stejným tenantem nebo musí být přiřazen do tenantu, v kterém je vytvořen nový asset.

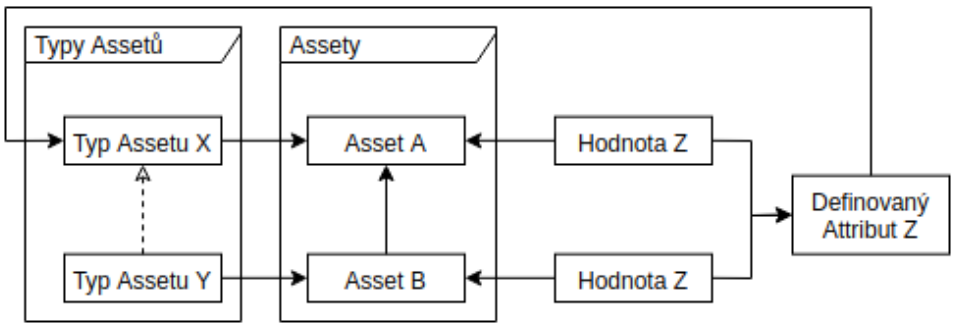
8.2 Možnost přiřazení vlastních atributů

Uživatel má možnost vytvořit si vlastní atribut, který poté může přidělit některému z typů síťových prvků. Atributy mají vlastní validaci a datový typ.

8.2.1 Dědičnost atributů

Stromovou strukturu je možné využít k vytvoření dědičnosti atributů. Tím je možné definovaný atribut v rodičovském assetu zobrazit i v jeho potomcích. Možnost dědit atributy je volitelná a je nastavena při přiřazení atributu k typu assetu[3)].

Při tvorbě assetu uživatel vybírá rodičovský asset. Pokud uživatel zvolí rodičovský asset, jehož typ má přiřazené dědičné atributy, pak jsou tyto dědičné atributy zkopírovány a nabídnuty právě vytvářenému assetu. Jelikož však existuje vazba na typ a na hodnotu, může více hodnot ukazovat na společný definovaný atribut. Tento princip je vidět na obrázku 8.4. *Typ Assetu X* nabízí *Definovaný Atribut Z*, který je dědičný a po vytvoření *Assetu B*, který je potomkem *Assetu A*, je tento atribut nabídnut k doplnění vlastní hodnoty *Assetem B*. To znamená, že každý asset na obrázku má definovanou *Hodnotu Z* pro stejný definovaný atribut.



Obr. 8.4 Znázornění dědičnosti v kontextu definovaných atributů

Při odstranění definovaného atributu musí být odstraněny všechny jeho hodnoty i ve zděděných assetech. Pokud chce uživatel odstranit příznak dědičnosti atributu, musí být informován o tom, že tato změna může ovlivnit již vytvořené assety, v tomto případě smazáním hodnot a atributů v potomcích.

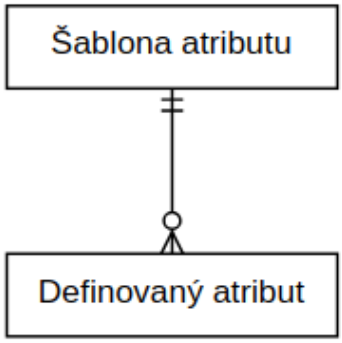
8.2.2 Definovaný atribut a jeho šablona

Pokud je atribut přiřazen nějakému typu a z tohoto typu je poté vytvořen nový asset, může nastat situace, že tento atribut bude potřeba editovat. Vzhledem k tomu, že atribut je již přiřa-

[3)] což je tvorba definovaného atributu

zen a jeho hodnota je vyplněna, může úprava atributu narušit tuto relaci. Řešením je rozdělení tvorby atributů do dvou oddělených entit. První entitou je definovaný atribut a druhou entitou je šablona atributu.

Uživatel tedy nejprve vytvoří šablonu atributu. Vyplní její jméno, zda má být její hodnota povinně zadána, příznak dědičnosti a datový typ. Po uložení se vytvoří šablona atributu.



Obr. 8.5 Relace šablony a definovaného atributu

Pokud je šablona atributu přiřazena nějakému typu, stává se z šablony definovaný atribut. Po přiřazení a uložení typu se vytvoří nová instance definovaného atributu, která je kopií šablony.

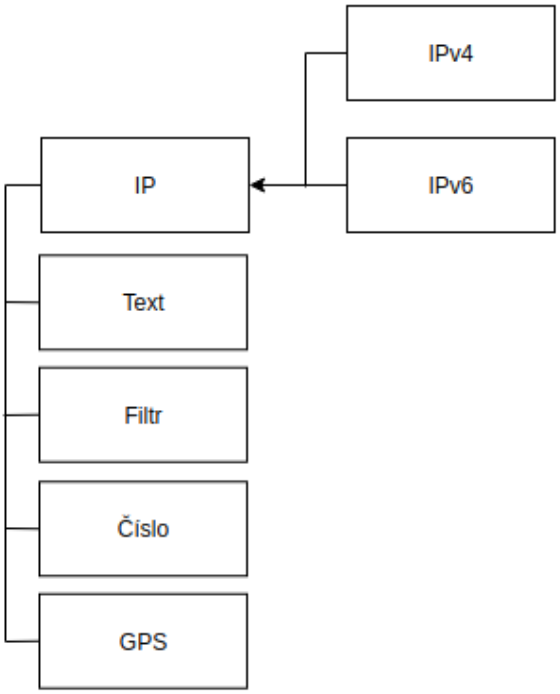
Tento princip má velkou výhodu v tom, že když v budoucnu některý z uživatelů bude chtít některou z šablon upravit, tak již přiřazené šablony[4)] se tímto zásahem neovlivní.

8.2.3 Kontrola datových typů

Šablona atributu, případně definovaného atributu, obsahuje informaci o datovém typu. Každý datový typ musí obsahovat validaci a případně nastavení, které předá informaci uživatelskému rozhraní, jaký je doporučený HTML element pro vstupní hodnoty.

Rozlišujeme základní datové typy jako jsou integer[5)], float[6)] a text. Složitější datové typy vychází ze základních. Mezi složitější typy patří IP adresa a GPS souřadnice. Každý datový typ má schopnost validace. Uživatel se nemusí omezovat předepsanou validací a jednotlivé validace může parametrizovat. Nastavení parametrů validace zadává uživatel během přidělení atributu k typu assetu[7)]. Parametry validace není možné v budoucnu měnit z důvodu ohrožení již uložených hodnot. Hrozí, že již uložené hodnoty by nemusely nové limity validace

[4)] definované atributy
[5)] celé číslo
[6)] desetinné číslo
[7)] během tvorby nebo editace typu



Obr. 8.6 Diagram tříd reprezentující datové typy

splňovat.

8.3 Vyhledávání

Asset je možné skládat do stromové struktury. Výsledná množina může být velká a uživatel může ztratit přehled o pozici konkrétního prvku ve stromové struktuře. Jedním ze základních požadavků je tedy možnost vyhledávání. Standardní je vyhledávání podle jména síťového prvku. Tento způsob se dá rozšířit o vyhledávání a filtrování pomocí štítků[8)].

Asset je možné označit štítkem. Podobu štítku můžeme přirovnat k štítkům v aplikaci Gmail, ve které si můžeme označit své emaily a pomocí štítků je například třídit do složek.

Je nutné se zamyslet i nad tím, jak při dědičnosti assetů přistupovat k těmto štítkům. Z uživatelského pohledu dává smysl, aby byl při vyhledávání nalezen celý jeho podstrom. To znamená, že štítky musí být dědičné. Štítky se tedy zobrazí v podstromu jako implicitní, bez možnosti je z assetu odstranit. Pro odstranění štítku musí uživatel udělat změnu v rodičovském assetu, kterému byl štítek přiřazen.

Důležité je také vymezit rozsah použití štítků v rámci multitenance. Jelikož každý tenant může být provozován jinou skupinou uživatelů, musí být možné, aby uživatelé tyto štítky přiřazovali pouze v rámci svého tenantu. Při dotazu na získání stromu musí jít rozpoznat do kterého tenantu jednotlivé štítky patří. Proto i jednotlivé štítky obsahují identifikátor tenantu,

[8)] label/tag

v kterém byly vytvořeny.

8.4 Zpětná revize změn

Některé změny v provedené uživatelem mohou být tak závažné, že ovlivní chování natolik, že asset z hlediska monitorování sítě ztratí význam. Z toho důvodu je důležité mít nástroj, který dokáže zaznamenávat změny, tak aby se k nim mohl uživatel kdykoliv vrátit a případně je vrátit zpět.

Uživatel definuje šablonu záznamu, který se má vykonat při manipulací se síťovým prvkem. Důvod tvorby jednotlivých šablon pro záznamy je ten, že uživatel poté může rozlišit různě závažné zásahy do stromu. Po editaci se vytvoří konkrétní záznam, v kterém lze vidět jaký uživatel provedl změnu a o jakou akci se jedná. Případně další informace z šablony, jako je závažnost a dopad na funkčnost v podobě uživatelsky definované zprávy.

8.5 Případy užití

Při tvorbě analýzy je potřeba brát v potaz pohled uživatele a vytvořit z něj případy užití. Dává smysl určit složitější kombinace tvorby sítě. Výstup z této části má za cíl vizualizovat možnosti uživatelů a tím i ulehčit budoucí tvorbu uživatelského rozhraní, které může navázat na tuto práci.

Příkaz pro tvorbu entit v modelu může přijít pouze z jádra Flowmon. Tento zdroj pouze přeposílá příkazy z dvou hlavních zdrojů. Prvním zdrojem je import zálohy konfigurace a druhým zdrojem je uživatelský vstup v grafickém uživatelském rozhraní na kolektoru.

8.5.1 Tvorba assetu z předdefinovaného typu

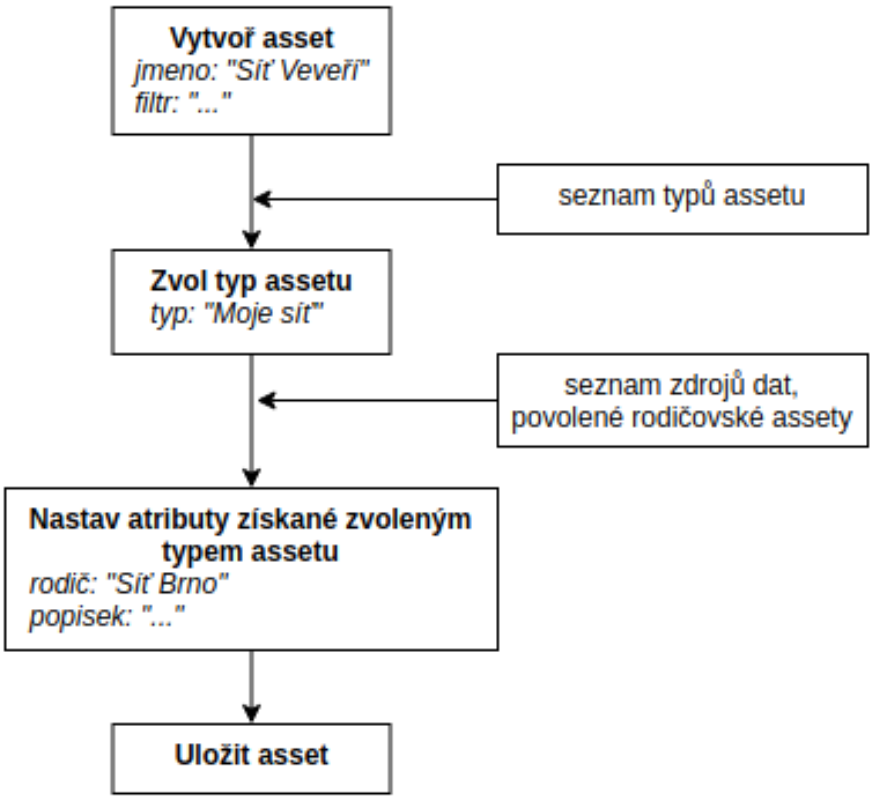
Pokud bude tvorba probíhat z importu zálohy konfigurace, budou již všechny položky assetu vyplněny a provede se jednotná validace nad celkovou entitou.

Pokud však přijde požadavek z uživatelského rozhraní, bude potřeba rozdělit tvorbu na více částí. Na obrázku 8.7 je tento postup znázorněn.

Po vstupu do formuláře zadá uživatel typ assetu. Protože každý typ má seznam povolených rodičovských typů, tak až v tuto chvíli může formulář zažádat o seznam rodičů, kteří jsou validní možností pro aktuálně tvořený asset.

8.5.2 Vytvoření typu assetu

Tak jako při tvorbě assetu se tvorba typu provede ve dvou krocích. V prvním kroku se nastaví typy, které mohou být rodiči nového typu. V dalším kroku se získá seznam atributů a to konkrétně těch, které jsou obsaženy v rodičovských typech nastavených v předchozím kroku. Tento seznam atributů dokáže uživateli říci, které atributy asset zdědí.



Obr. 8.7 Tvorba assetu

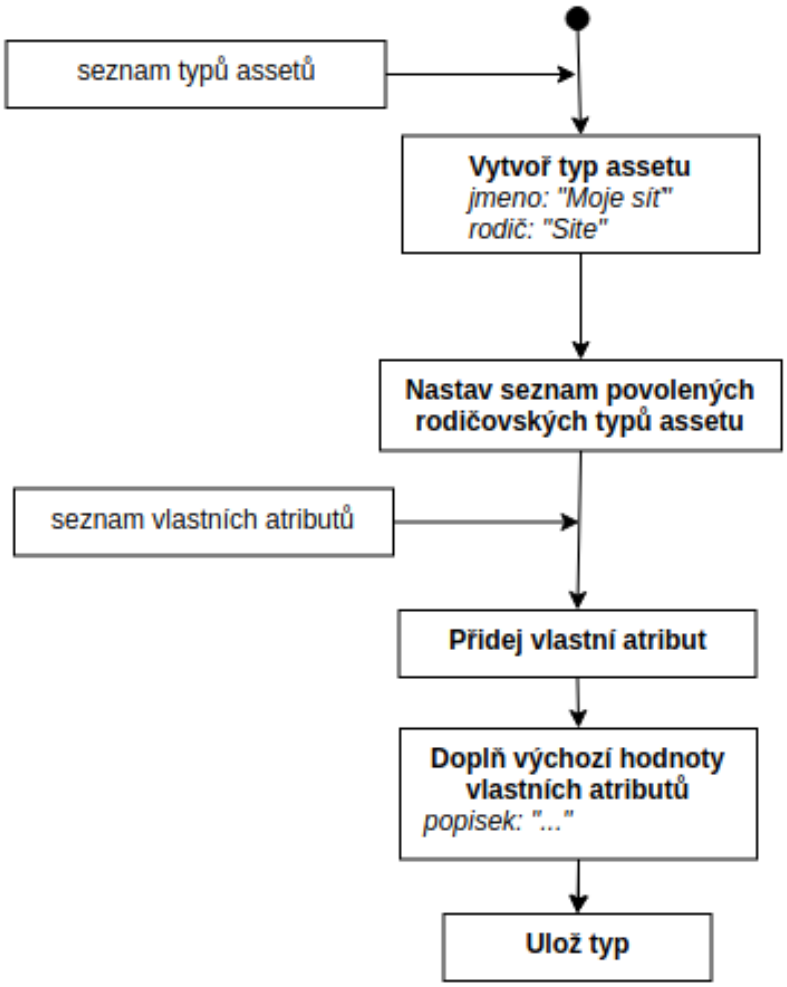
8.6 Výsledný databázový model

Na obrázku 8.9 je možné vidět finální model databáze. V následujícím textu budou popsány významné relace.

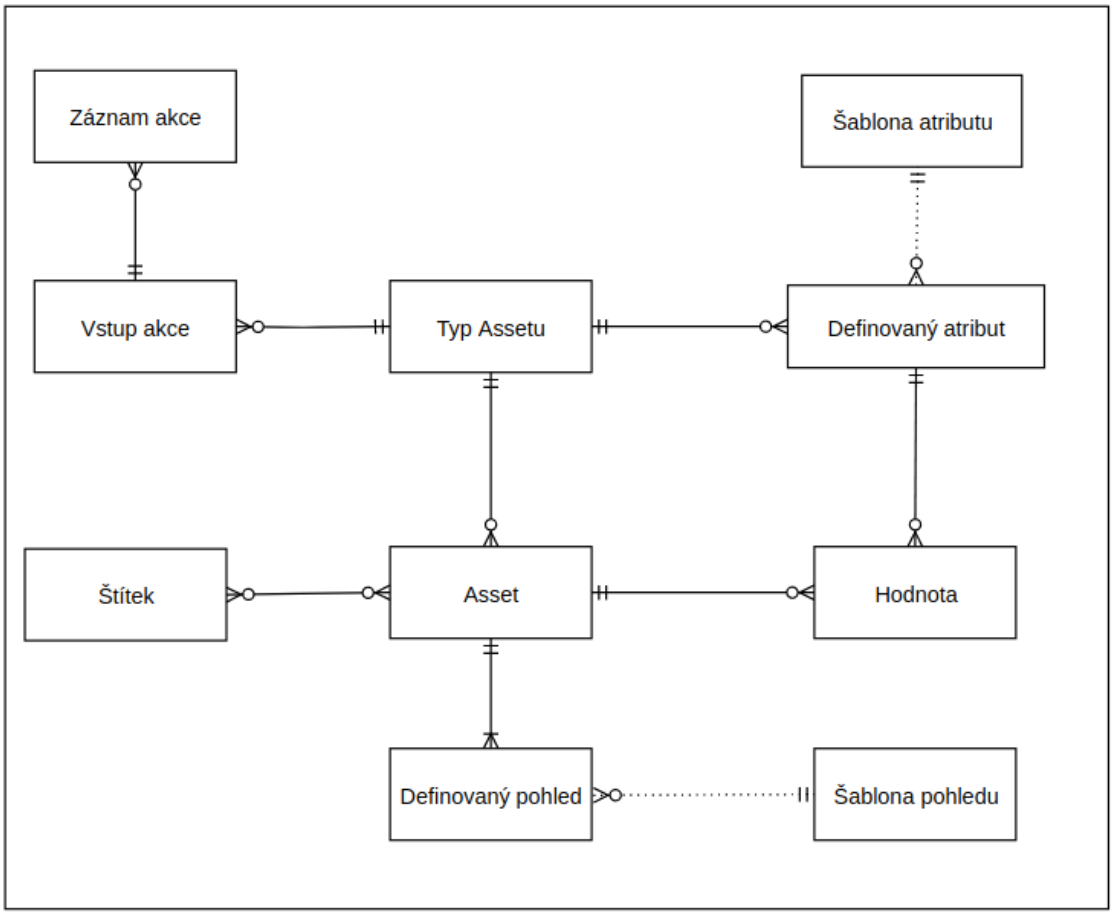
Jádrem modelu je typ assetu a asset, na které se vážou ostatní entity.

Štítky jsou navázaný na asset. Štítky je možné vytvořit zvlášť a přidělovat je síťovým prvkům později, proto zde vzniká relace, kdy štítek nemusí patřit žádnému ze síťových prvků. Asset může mít zároveň více štítků. Mezi těmito entitami tedy musí vzniknout vazební tabulka.

Na obrázku lze vidět vazby, které jsou značené přerušovanou čarou. Tyto vazby představují pouze logickou vazbu mezi entitami a v databázi nejsou vytvořeny pomocí cizích klíčů. Jednou z těchto vazeb je vazba mezi šablonou atributu a definovaným atributem. Tyto dvě entity mají každá svou tabulku. Uživatel si vytvoří šablonu atributu a poté při tvorbě typu se šablona zkopíruje do tabulky definovaného atributu. Tento princip platí i v případě tabulek pohledů.



Obr. 8.8 Tvorba typu assetu



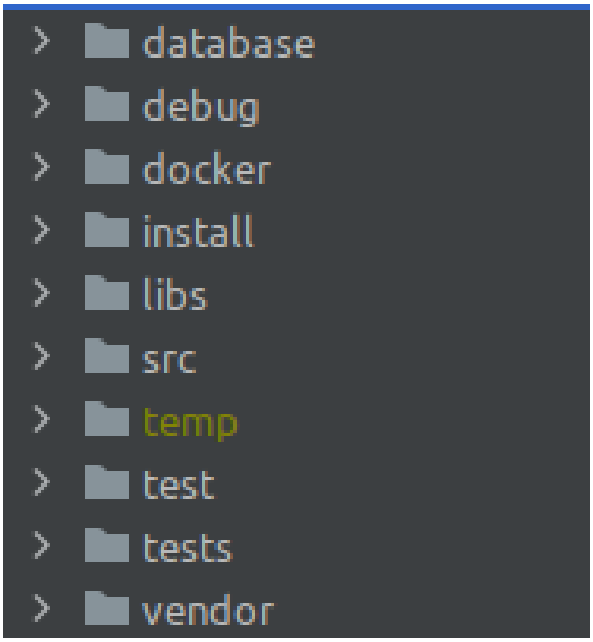
Obr. 8.9 Entity-relationship diagram

III. PROJEKTOVÁ ČÁST

V této části práce, se zabývám konkrétními implementačními koncepty, které bylo nutné vyřešit pro vytvoření základního modelu. Popsány jsou komunikace a vztahy mezi konkrétními třídami, které přímo nesouvisí s relacemi modelu, ale jsou součástí manažerů a vstupně/výstupních tříd, které zapouzdří model popsaný v analytické části práce.

9 Struktura projektu

Projekt budu popisovat po jeho jednotlivých částech. Na obrázku 9.1 můžeme vidět jednotlivé složky, které obsahují zdrojové kódy a instalační informace. Složka database obsahuje SQL soubory, které se využívají při instalaci modulu a mají zajistit inicializaci databáze. Další důležitou složkou je složka docker, v které jsou všechny potřebné informace pro spuštění docker kontejnerů. Nejdůležitějšími složkami jsou src, která obsahuje zdrojové soubory projektu a tests, v které jsou unit testy. Složka vendor obsahuje balíčky nainstalované pomocí programu composer. Poslední složkou, kterou je potřeba zmínit, je složka libs, která obsahuje knihovny, které nejsou součástí projektu.



Obr. 9.1 Struktura projektu

10 Prostředí pro spuštění

Nezávislost modulu na okolí je důležitým prvkem projektu. Umožní se tím spouštění unit testů na vlastním počítači a není nutné nahrávat projekt na Flowmon kolektoru a až tam spouštět unit testy. V dnešní době existují technologie, které nám s tímto problémem pomohou. V této práci je použita technologie Docker.

10.1 Testování v lokálním prostředí

K testování projektu slouží PHPUnit 9[1)]. Spuštění testů není v lokálním prostředí problém, pokud máme prostředí s nainstalovaným PHP příslušné verze a databázi s připravenými tabulkami modelu. Chování projektu může poté ovlivnit opakované spouštění, nižší kontrola nad databází a operační systém, na kterém jsou testy spouštěny. Tento problém pomůže vyřešit použití technologie Docker.

10.2 Docker

Pro projekt jsou potřeba dva kontejnery. První nám zajistí spuštění PHP a druhý slouží k zajištění databáze. Každý kontejner potřebuje vlastní Dockerfile. Dockerfile je textový soubor, který obsahuje všechny potřebné příkazy nutné k tomu, aby byl kontejner schopný funkčnosti.

Pro vytvoření databáze nám bude stačit typ zdrojového image. Což je v našem případě postgres :12. Tento typ databáze by měl zaručit, že SQL soubory budou fungovat i na serveru s Flowmonem.

```
FROM php:7.4-cli-buster
ARG DEBIAN_FRONTEND=noninteractive
RUN ln -fs /usr/share/zoneinfo/Europe/Prague /etc/localtime

RUN apt-get update

RUN apt-get install -y libpq-dev \
    && docker-php-ext-configure pgsql -with-pgsql=/usr/local/pgsql \
    && docker-php-ext-install pdo pdo_pgsql pgsql

RUN docker-php-ext-install sockets
```

Obr. 10.1 Náhled na Dockerfile zajišťující spuštění PHP

Druhý Dockerfile je vytvořený z php:7−4−cli−buster. Zde se ještě musí doinstalovat rozšíření pro PHP postgresql, které tento image v základní konfiguraci neobsahuje. Jelikož projekt využívá knihovnu, která zajišťuje komunikaci přes RabbitMQ, je nutné nainstalovat PHP rozšíření sockets .

[1)] https://phpunit.de/index.html

10.3 Spuštění

Po instalaci assetového modulu zajistí linuxové jádro na Flowmon kolektoru spuštění služby, která běží na pozadí. Poté je modul připraven přijímat a odesílat zprávy prostřednictvím RabbitMQ.

Pro spuštění testů nemusí být modul nainstalovaný na kolektoru, ale stačí pouze spustit script **runTests.sh**, který je součástí projektu. Tento script inicializuje a spustí Docker, který následně zajistí spuštění testů. Lokální spuštění je možné, pouze za předpokladu, že má vývojář k dispozici interní knihovny produktu Flowmon.

Jakmile se vytvoří kontejnery, spustí se script, který načte všechny SQL soubory a aplikuje je do databáze. Součásti těchto SQL souborů jsou i výchozí data. Mezi tato data patří především výchozí typy assetů. Poté se spustí postupně všechny unit testy. Každý test může mít jeden ze tří výstupů.

- úspěšný test reprezentovaný symbolem tečky,
- neúspěšný test reprezentovaný symbolem F (fail),
- varování, pokud při testu nedošlo k žádné kontrole.

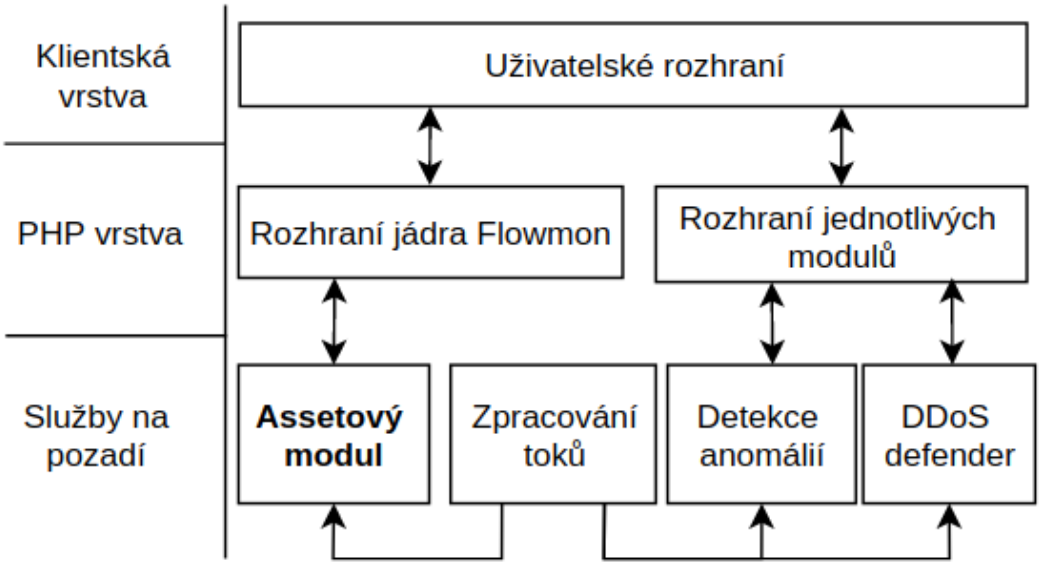
```
Running PHP tests...
PHPUnit 9.5.4 by Sebastian Bergmann and contributors.

.......................                                       23 / 23 (100%)

Time: 00:01.557, Memory: 22.00 MB

OK (23 tests, 114 assertions)
```

Obr. 10.2 Úspěšný výstup po lokálním spuštění testovacího prostředí

Pokud jsou všechny testy úspěšné, můžeme prohlásit, že modul splňuje navržené testovací metriky.

11 Zasazení modulu do kontextu produktu

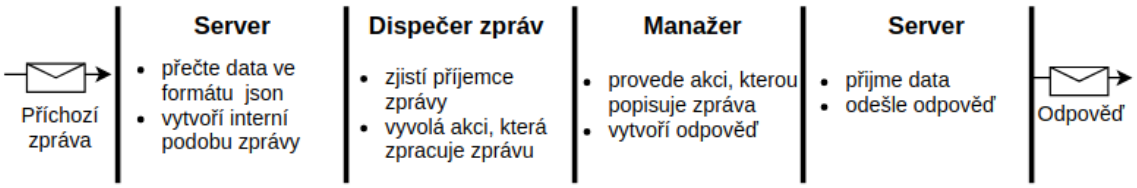
Na obrázku 11.1 můžeme vidět zasazení assetového modulu do kontextu ostatních modulů produktu Flowmon. Z obrázku je patrné, že assetový modul se nachází až za webovým rozhraním, se kterým komunikuje GUI[1)]. Toto má velkou výhodu, jelikož se zjednoduší komunikace mezi jednotlivými moduly. Rozhraní jádra Flowmon má již vyřešenou komunikaci. Assetový modul pak nemusí řešit komunikaci s klientským webovým rozhraním[2)] a řeší pouze komunikaci přes RabbitMQ.



Obr. 11.1 Zasazení assetového modulu do kontextu ostatních modulů

Z uživatelského rozhraní se posílají HTTP požadavky na rozhraní jádra Flowmon, ten je zpracuje na RabbitMQ zprávy a „Asset model“ tyto zprávy poté vykoná.

11.1 Zpracování zpráv a vykonávání akcí modulem



Obr. 11.2 Pracovní postup modulu

Modul má přesně definovaný postup, jak vykonávat akce. Tento postup nelze „z venku“ nijak obejít. Aby mohl modul fungovat, musí do něj nejprve někdo z venčí odeslat zprávu.

[1)] graphic user interface
[2)] webový frontend, javascript

Jakmile modul přijme zprávu, rozbalí ji, a pomocí třídy MessageFactory sestaví její vnitřní podobu. Pokud tato třída zprávu nerozpozná, ihned se vyšle odpověď informující o neúspěšném vytvoření zprávy. Pokud zprávu rozpozná, předá ji třídě MessageDispatcher, která má za úkol tuto zprávu předat příslušnému manažeru k obsloužení.

Manažeři přijímají následující typy zpráv:

- **create** - vytvoří novou entitu,
- **update** - upraví již existující entitu,
- **delete** - smaže entitu,
- **get** - jako odpověď vrátí danou entitu,
- **getAll** - jako odpověď vrátí všechny entity,
- **validate** - jako odpověď vrátí, zda je daná entita validní pro uložení.

Jakmile manažer zprávu vykoná, vytvoří pomocí třídy MessageFactory odpověď a vrátí ji třídě RabbitServer k odeslání tomu, kdo vyslal požadavek. Odpověď může mít tři základní typy. V případě úspěšného vykonání akce se odešle SuccessResponse, v případě validační chyby při ukládání se vytvoří ValidationResponse. Pokud se stane, že selže ukládání do databáze, dojde místo na disku nebo se vyskytne nějaká chyba při vykonávání kódu, který není pokrytý testem, vytvoří se chybová zpráva typu FailResponse.

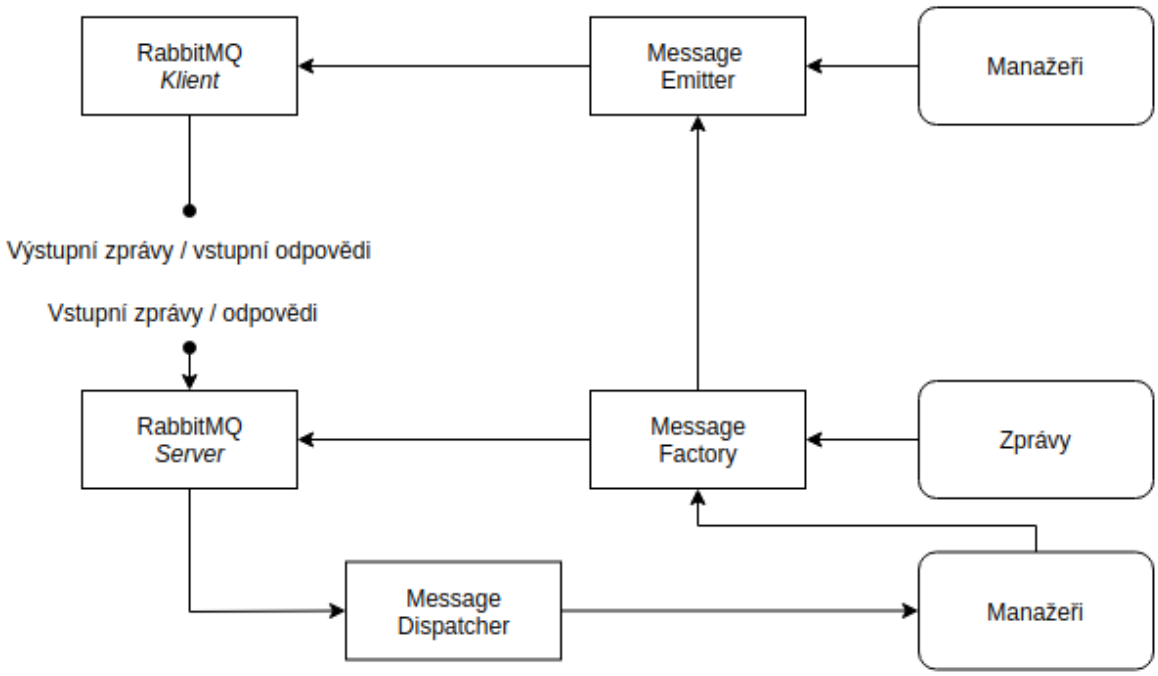
Příchozí zprávy se dají rozdělit na dvě kategorie, ty které žádají o data a ty, které chtějí data uložit.

Jednotlivé entity jsou implementovány pomocí principu ORM[3)]. Jednotlivé entity jsou PHP třídy, které reprezentují právě jednu tabulku v databázi. Nad každou takovou entitou můžeme provádět CRUD[4)] operace. Každá entita, s kterou chceme nějak pracovat z venčí, má vlastní manažer. Pro entitu Asset vzniká AssetManager, pro AssetType vzniká AssetTypeManager a obdobně i pro ostatní entity.

Většinu entit je potřeba validovat, zda neobsahují neplatná data, která by mohla vadit při ukládání. Proto obsahují všechny entity metodu validate , která vrací binární hodnotu, zda entita splňuje podmínky uložení a v případě neúspěchu i pole validačních zpráv.

Pouhá validace nemusí stačit. Omezit možnost ukládání entit musí být docíleno ještě jednou vrstvou. Některé specifické entity mohou být z pohledu obsahu dat validní, ale mohou být chráněny proti smazání či editaci. Takovou entitou je například předdefinovaný typ assetu, který nelze smazat ani editovat, ale mohou nastat případy, kdy jeho editaci chceme povolit. Pro tento účel existuje rozhraní IAccessLimiter, které definuje metody canEdit, canDelete a

[3)] https://en.wikipedia.org/wiki/Object-relational_mapping
[4)] create, update, delete



Obr. 11.3 Propojení manažerů a RabbitMQ klient a server

canCreate. Manažer, který potřebuje tyto metody pouze rozšíří implementaci o toto rozhraní. Toto můžeme vidět na obrázku 11.4.

Jednotlivé entity používají pro identifikaci atribut id, který je textový řetězec a jedná se o UUID[5)]. Toto použití je především z důvodu exportu konfigurace kolektoru a následného importu na jiný kolektor. Pokud by identifikátor nebyl tohoto typu, mohlo by lehce docházet ke konfliktům v databázi.

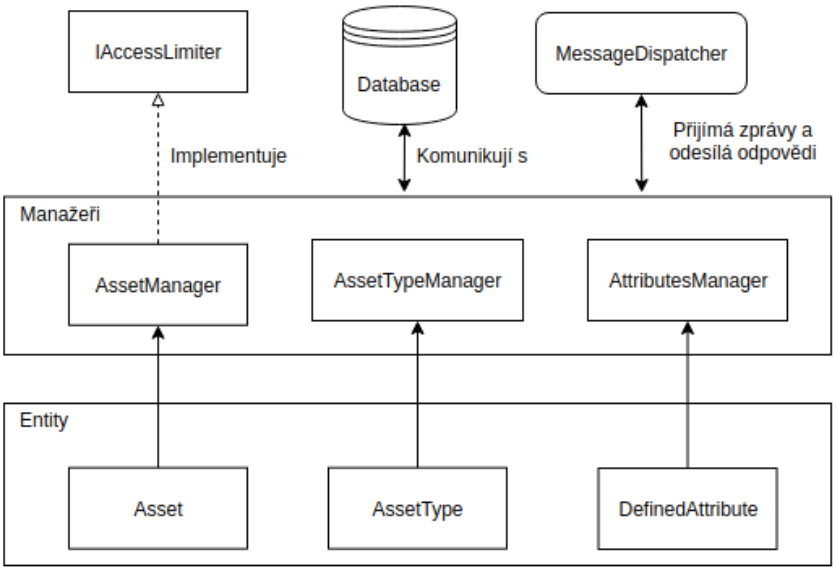
11.2 Testy

Při vývoji je využita metodika vývoje řízeného testy (TDD). Popis této metodiky a její algoritmus je možné nalézt v kapitole 4.

Základním prvkem je testovací třída, která obsahuje testovací metody. Každá testovací třída reprezentuje některou z implementovaných entit nebo manažerů. Metody uvnitř každé testovací třídy jsou pojmenovány jako pravidla a podmínky vystihující očekávané chování a případně i cíl testu. Hlavní výhodou výstižného pojmenování testovacích metod, je zpřehlednění a ulehčení práce při budoucím vývoji.

Testy jsou rozděleny na dva typy, validační a modelové. Validační testy testují validace jednotlivých entit a modelové testují, zda manažer pracující s entitami vrací správná data při různých vstupech.

[5)] https://en.wikipedia.org/wiki/Universally_unique_identifier



Obr. 11.4 Relace mezi manažery a entitami

ZÁVĚR

V této práci se podařilo analyzovat a navrhnout model sloužící pro správu síťových prvků v produktu Flowmon. Nový „Asset model“ vylepšuje starý profilový model. Hlavním přínosem nového modulu je možnost přiblížit správu síťových prvků uživatelům, kteří nemají detailní znalosti síťových technologií. Může se tím rozšířit počet uživatelů, kteří jsou schopni s produktem pracovat a tím se může zvýšit i počet potenciálních zákazníků.

Kromě návrhu vznikl také prototyp modulu, který zajišťuje základní spuštění a testování modulu. Modul lze rozdělit na několik částí. První z nich je databáze, druhou je objektově relační model v jazyku PHP a třetí částí je práce s knihovnu RabbitMQ, pro komunikaci s ostatními moduly, která je také napsána v jazyku PHP.

„Asset model“ je dědičná stromová struktura. Každý asset má svůj vlastní typ. Uživatel může assetovému typu přiřadit množinu vlastních atributů. Tyto atributy poté mohou být dědičné v rámci stromové struktury. Assety si může uživatel označit štítkem pro jednoduší vyhledávání. Na jednotlivé assety jsou navázané pohledy, které pomáhají s vizualizací dat.

Modul je možné rozšířit o další entity, které je možné navázat na asset nebo typ assetu. Jako možné rozšíření se nabízí přímá vazba na report nebo na widget. Tato vazba umožní uživateli jednoduší správu widgetů nebo reportů bez nutnosti detailní znalosti sítě.

Testovací proces zajišťují technologie Docker a PHPUnit. Tato kombinace umožňuje nezávislost spuštění na vývojovém prostředí. Tím je možné se vyvarovat problémům se spuštěním na různých operačních systémech. V budoucnu to také ušetří čas vývojářům, kteří by chtěli tento projekt rozšířit, a to hlavně z důvodu, že nebudou muset instalovat modul na server Flowmon, kvůli základnímu spuštění při vývoji. Další výhodou kombinace technologií Docker a PHPUnit, je možnost jednoduše přidat automatické testování do moderních pracovních postupů, jakými jsou například Gitlab CI/CD nebo Jenkins.

Na tuto práci je možné navázat integrací do systému Flowmon. Integrace bude vyžadovat zajištění komunikace s ostatními moduly. Při integraci se také musí zajistit automatické spuštění RabbitMQ serveru.

SEZNAM POUŽITÉ LITERATURY

[1] ČELEDA, Pavel, KOVÁČIK, Milan, KONÍŘ, Tomáš, KRMÍČEK, Vojtěch, ŠPRINGL, Petr, ŽÁDNÍK, Martin, *FlowMon Probe*. PB tisk, s.r.o. [Online]. Available: https://is.muni.cz/publication/746159/en/FlowMon-Probe/Celeda-Kovacik-Konir-Krmicek

[2] ELICH, Martin,GRÉGR, Matěj,ČELEDA, Pavel, "Monitoring of tunneled IPv6 traffic using packet decapsulation and IPFIX (short paper)," in *Traffic Monitoring and Analysis*, ser. Lecture Notes in Computer Science, J. Domingo-Pascual, Y. Shavitt, and S. Uhlig, Eds. Springer, pp. 64–71.

[3] Gartner, Inc. (2021) Multitenancy. [online]. Dostupné z: https://www.gartner.com/en/information-technology/glossary/multitenancy. Navštíveno: 2.5.2021.

[4] VITEKER, Jaromír. (2009) Nástroj pro modelování redakčních systémů. Diplomová práce, Masarykova univerzita.

[5] VMware, Inc. What can rabbitmq do for you? [online]. Dostupné z: https://www.rabbitmq.com/features.html. Navštíveno: 2.5.2021.

[6] CloudAMQP. (2019) What is amqp and why is it used in rabbitmq? [online]. Dostupné z: https://www.cloudamqp.com/blog/what-is-amqp-and-why-is-it-used-in-rabbitmq.html. Navštíveno: 2.5.2021.

[7] BAGUS,Ida., "Combination of test-driven development and behavior-driven development for improving backend testing performance," *Procedia Computer Science*, vol. 157, pp. 79–86, 2019, the 4th International Conference on Computer Science and Computational Intelligence (ICCSCI 2019) : Enabling Collaboration to Escalate Impact of Research Results for Society. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050919310622

[8] MOE, M., "Unit test using test-driven development approach to support reusability," *International Journal of Trend in Scientific Research and Development*, vol. 3, pp. 194–196, 2019. [Online]. Available: http://www.ijtsrd.com/papers/ijtsrd21731.pdf

[9] ANDERSON, Charles, "Docker [software engineering]," *IEEE Software*, vol. 32, no. 3, pp. 102–c3, 2015.

[10] Technology, *FlowMon*. Anim Publishing. [Online]. Available: https://www.morebooks.de/store/gb/book/flowmon/isbn/978-620-0-30796-5

[11] F. N. a.s, *Uživatelská příručka Flowmon 11*, Brno, 2020.

[12] KAMISIŃSKI,Andrzej,FUNG,Carol, "FlowMon: Detecting malicious switches in software-defined networks."

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

HTML	Značkovací jazyk využívaný ve vývoji webových stránek
IPFIX	Protokol rozšiřující technologii Netflow
JSON	Strukturovaný způsob zápisu dat
ORM	Technika, která konvertuje data mezi databází a programovacím jazykem
PHP	Skriptovací jazyk používaný k vývoji webů
RAID	Způsob ukládaní a ochrany dat
SQL	Dotazovací jazyk pro práci s daty v relačních databázích

SEZNAM OBRÁZKŮ