Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

Middleware pro agregaci dat internetu věcí

Bc. Lukáš Čížek

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Lukáš Čížek

Informatika

Název práce

Middleware pro agregaci dat internetu věcí

Název anglicky

Middleware for Aggregation of Internet of Things Data

Cíle práce

Diplomová práce je tématicky zaměřena na zhodnocení middleware sloužící k agregaci, transformaci a distribuci dat získaných prostřednictvím zařízení internetu věcí.

Hlavním cílem práce je zhodnotit vybrané middleware služby internetu věcí.

Dílčí cíle práce jsou:
- charakterizovat middleware služby pro internet věcí dostupné online
- analyzovat klíčové požadavky na agregaci, transformaci a distribuci dat internetu věcí
- zhodnotit vybrané middleware služby internetu věcí

Metodika

Teoretická část diplomové práce se bude zakládat na analýze a rešerši odborných zdrojů.

V praktické části práce budou na základě poznatků zjištěných z teoretické části vybrány klíčové požadavky na middleware služby. Následně budou zhodnoceny vybrané služby pomocí metod vícekriteriální analýza variant.

Na základě syntézy teoretických a praktických poznatků budou zpracovány závěry diplomové práce.

Doporučený rozsah práce

65 stran

Klíčová slova

IoT, Middleware, agregace, transformace, datové sklady

Doporučené zdroje informací

Adrian McEven & Hakim Cassimally. Designing Internet of things. John Wiley and Sons, Ltd., 2014. ISBN 978-1-118-43062-0.

BURIAN, P. Internet inteligentních aktivit. Praha: Grada, 2014. ISBN 978-80-247-5137-5.

HASSAN QUSAY F. Internet of Things A to Z: Technologies and Applications.Wiley-IEEE Press, 2018. ISBN: 978-1-119-45674-2

MILLER LAWRENCE. Internet of Things for Dummies. John Wiley and Sons, Ltd., 2017 ISBN 978-1-119-34992-1

Předběžný termín obhajoby

2018/19 LS – PEF

Vedoucí práce

Ing. Michal Stočes, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 15. 10. 2018

Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 19. 10. 2018

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 17. 02. 2019

Čestné prohlášení

Prohlašuji, že svou diplomovou práci " Middleware pro agregaci dat internetu věcí" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 29.3.2019 _____

Poděkování

Rád bych touto cestou poděkoval panu Ing. Michalovi Stočesovi, Ph.D. za jeho ochotu, rady, trpělivost při vedení mé práce a především za jeho včasné a věcné odpovědi. Dále velmi děkuji celé rodině a přátelům za jejich trpělivost a rady nejen při psaní této práce, ale také během celého studia.

Middleware pro agregaci dat internetu věcí

Abstrakt

Diplomová práce je zaměřena na analýzu a zhodnocení middleware služeb sloužící k agregaci, transformaci a distribuci dat získaných prostřednictvím zařízení internetu věcí. V teoretické části jsou popsána teoretická východiska pro internet věcí, skladování, typy a dostupnost dat a middleware služby analýzou odborných literárních zdrojů.

V návaznosti na teoretickou část jsou v praktické části analyzovány klíčové požadavky na agregaci, transformaci a distribuci dat pomocí middleware služeb, ze kterých vyplynuly kritéria pro pozdější analýzu.

Byly charakterizovány dostupné open source middleware služby internetu věcí, které byly následně hodnoceny podle vhodně zvolených vah pomocí vícekriteriální analýzy variant.

Výsledkem VAV byla kompromisní varianta ThingsBoard, která vhodná pro zvolený případ užití, další možnou variantou bylo implementace vlastního řešení pomocí knihovny DeviceHive. Nakonec proběhlo zhodnocení výsledků práce a aplikace na další případy užití.

Klíčová slova: IoT, internet věcí, Middleware, agregace, transformace, distribuce, datové formáty, datové sklady, velká data, otevřená data, vícekriteriální analýza variant

Middleware for Aggregation of Internet of Things Data

Abstract

The diploma thesis is focused on analysis and evaluation of middleware services used for aggregation, transformation and distribution of Internet of Things data.

The theoretical part described Internet of Things, storage, types and availability of data and middleware services by analysing technical sources.

The practical part was based on the previous theory defined crucial requirements for aggregation, transformation and distribution using middleware services that were used for selecting criteria for analysis.

Available open source middleware services were described and evaluated with appropriate weights by multi-criteria analysis of variants.

As a result of multi-criteria analysis was middleware service ThingsBoard that was suitable for selected use case. As an alternative, DeviceHive could be used for implementing own solution. At the end was evaluation of results and application to other use cases.

Keywords: IoT, Internet of Things, Middleware, aggregation, transformation, distribution, data format, data warehouse, big data, open data, multi-criteria analysis of variants

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Již dlouhou dobu používáme digitální zařízení na měření, kontroly a počítání avšak tato zařízení s příchodem všudypřítomného připojení mohou data sdílet a vytvářet sofistikované sítě. Těmto zařízením v sítí se říká obecně internet věcí a nejsou již doménou vědeckých pracovišť, ale našla svoje kvalitní využití v zemědělství. Množství těchto různorodých dat je potřeba zpracovávat předtím než jsou vhodná pro zobrazení a pro tento účel nám slouží software zvaný middleware. Aktuálně na internetu existuje mnoho dostupných řešení pro middleware služby a je složité se mezi nimi orientovat. Každé řešení se hodí na jiný případ užití a analýza užití bývá často velmi nákladná.

V první části bude popsán obecně internet věcí, jaká je situace na dnešním tuzemském trhu, co všechno internet věcí obsahuje a možnosti jeho využití. Internet věcí není jen o jednom zařízení, ale o celých skupinách a sítích pracujících a komunikujících dohromady. Při popisování internetu věcí je také třeba nastínit síťové technologie, které dnes s internetem věcí úzce souvisí.

Dále budou uvedena otevřená data, která s internetem věcí přímo souvisí, jak jejich dostupností, tak kvalitou. Data z internetu věcí mohou být považována za velká data a tím přichází potřeba na specializované databáze pro tento typ dat, které budou v této kapitole popsány. Tato kapitola bude také sloužit jako nastínění problematiky a úvod do zpracování dat internetu věcí pomocí vhodného nástroje nebo softwaru.

V další části bude popsána agregace, transformace a distribuce dat přicházejících z internetu věcí. Budou představeny, charakterizovány a popsány middleware služby dostupné online. Dostupných řešení a nástrojů online pro využití jako middleware je mnoho a je složité si vybrat ten vhodný nástroj pro konkrétní daný případ užití. Je nutné tedy analyzovat klíčové požadavky, kritéria a faktory při výběru middleware řešení, na které naváže praktická část. V praktické části budou vybrány nejvhodnější kritéria podle kterých bude vybráno několik middleware řešení. Ty budou představeny a zhodnoceny podle vícekriteriální analýzy variant. Na základě výsledků budou zvoleny vhodné middleware pro zvolené případy užití v internetu věcí.

2 Cíl práce a metodika

2.1 Cíl práce

Diplomová práce je tematicky zaměřena na zhodnocení middleware sloužící k agregaci, transformaci a distribuci dat získaných prostřednictvím zařízení internetu věcí. Hlavním cílem práce je zhodnotit vybrané middleware služby internetu věcí.

Dílčí cíle práce jsou:
- charakterizovat middleware služby pro internet věcí dostupné online,
- analyzovat klíčové požadavky na agregaci, transformaci a distribuci dat internetu věcí,
- zhodnotit vybrané middleware služby internetu věcí.

2.2 Metodika

Teoretická část diplomové práce se bude zakládat na analýze a rešerši odborných zdrojů.

V praktické části práce budou, na základě poznatků zjištěných z teoretické části, vybrány klíčové požadavky na middleware služby. Následně budou zhodnoceny vybrané služby pomocí metod vícekriteriální analýza variant.

Na základě syntézy teoretických a praktických poznatků, budou zpracovány závěry diplomové práce.

3 Teoretická východiska

Literární rešerše se zabývá nejprve internetem věcí jako takovým, kde jde o vymezení a popsání problematiky internetu věcí, analyzuje aktuální situace na trhu, jaké zařízení jsou dostupné na trhu, jejich cena a zájem veřejnosti v České republice o problematiku internetu věcí. Kapitola internet věcí, jakožto úvod do problematiky, přechází do specifičtějšího zbytku práce, kde se bude jednat převážně o práci s informací a daty.
Na předchozí kapitolu naváže kapitola 1.2, která je zaměřena na data a s tím spojené získávání z volně dostupných zdrojů jako otevřená data, datové sklady do kterých je možné data z internetu věcí posílat. Dojde k vymezení pojmu velkých dat, způsoby zpracování a ukládaní obecně.

Po vymezení internetu věcí a popsání dat se kapitola 1.3 zabývá middleware řešeními. Účely pro která jsou middleware služby využívány a způsoby přijmu dat. V další části vývoj middleware služeb jako takových, očekávaná funkcionalita pro správu dat, zařízení a uživatelů a další způsoby a možnosti distribuce dat.

3.1 Internet věcí

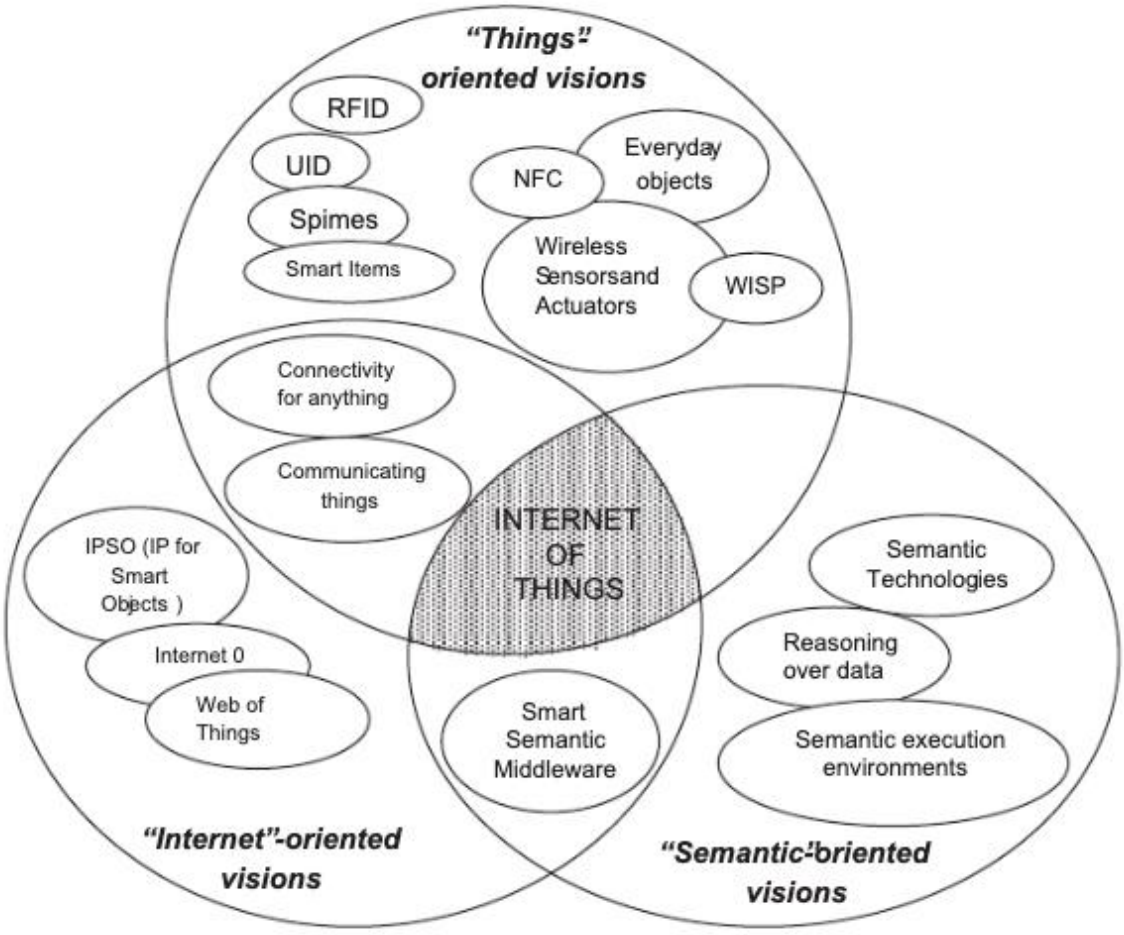
V době mobilního internetového připojení a v době informací se rozmohla nová evoluce internetu tak, jak ho známe. Předchozí komunikace probíhala přes protokoly jako HTTP pro webové stránky a SMTP pro mailovou komunikaci. Novým mezníkem je komunikace samotných zařízení mezi sebou a to nejen pro výměnu jednoduché informace, ale také pro orchestraci systémů a rozhodování v dalším chování. Tato nová komunikace strojů a zařízení samotných se nazývá M2M (machine to machine, komunikace stroj-stroj). Internet věcí, jeho rozsah a účel, se tedy dá definovat jako: *"nově vznikajícíٔ globálníٔ síťová architektura založená na internetu, která usnadňuje výměnu zboží a služeb v rámci globálních dodavatelských sítí a má vliv na bezpečnost a soukromí všech zúčastněných stran."* [1] [2]

Historicky zmínka o propojování zařízení je z roku 1991, kdy Mark Weiser popsal vizi a budoucnost internetu jako "všudypřítomný internet". Vize byla zaměřena na propojení mobilních zařízení v chytrém a žijícím prostředí pro vytvoření multimediálního systému. V roce 1999 přišla MIT a laboratoře Auto-ID s nápadem vytvoření produktového kódu, který by pomocí RFID identifikoval věcí v síti. Ačkoliv tato myšlenka je velmi podobná definici

internetu věcí, první zmínka o internetu věcí jako takovém se začíná objevovat až v roce 2003-2004 v knihách. Začínají se také objevovat další názvy jako Internet0, ten s internetem věcí souvisí jakožto pomalé, energeticky a cenově nenáročné připojení zařízení mezi sebou bez potřeby serveru.

Hlavní počátek trendu internetu věcí přišel v roce 2008, kdy skupina 50 společností, jako jsou například Cisco, Intel, SAP, které vytvořily IPSO alianci (IP for Smart Object, tedy připojení pro chytrá zařízení) a podpořily využití internetu pro koncept internetu věcí. V roce 2009 pak Cisco Internet Business Solutions Group (IBSG) definuje internet věcí. Internet věcí je definován jako množina chytrých zařízení/objektů jako jsou domácí zařízení, mobily nebo laptopy spojeny pomocí unikátního spojení a připojeny do Internetu přes jednotný framework. [3]

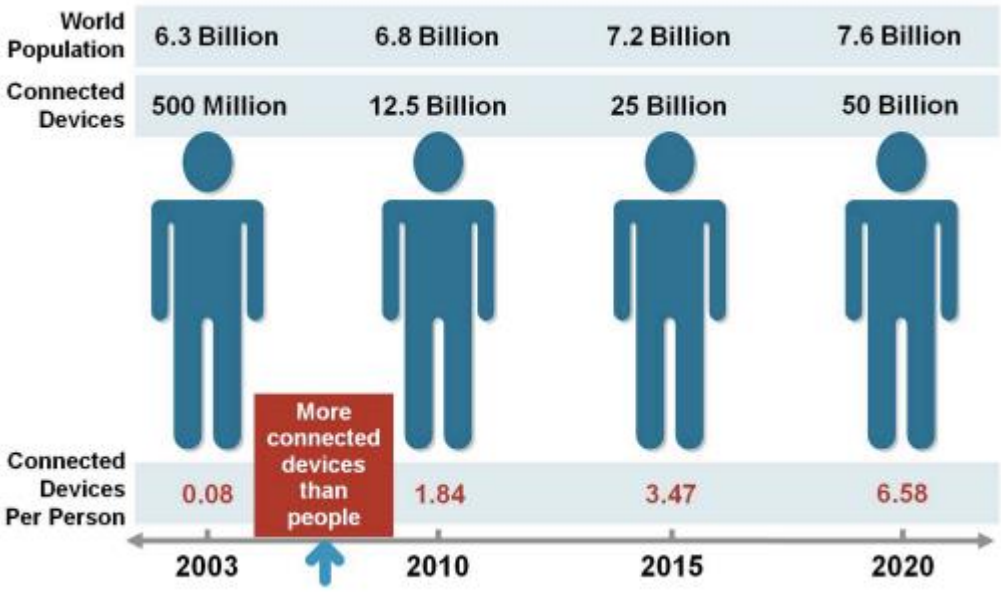
Atzori rozdělil paradigma internetu věcí ve své práci do tří skupin. První skupina jako internetově orientované vize, do které patří web věcí, internet 0 a IP pro chytrá zařízení, ta zastává právě připojení zařízení do sítě. Druhou skupinou je sémanticky orientovaná vize, která by se dala shrnout jako znalost práce s informací. Do této skupiny patří analýza dat a tedy i middleware, o kterém pojednává tato práce. Průnik těchto dvou skupin je pak chytrý připojený middleware. Poslední třetí skupinou je vize orientovaná na zařízení, do které patří samotná zařízení. Průnik všech tří skupin je pak právě internet věcí. Celé schéma je zobrazeno níže (Obrázek 1). [4]



Obrázek 1 - Internet věcí podle Atzoriho. Zdroj: [4]

3.1.1 Situace na trhu

Aktuální situace na světovém trhu a explozivní nárůst chytrých zařízení za poslední roky zvedl počet zařízení připojených na osobu ve světě okolo 3. Přesný počet zařízení se liší zdroj od zdroje a mnoho předpovědí je založených na článku od Cisco z roku 2011, kde Cisco uvádí 12.5 miliardy připojených zařízení v roce 2010, tedy rok před vznikem reportu. Cisco použil pro tuto předpověď studii z Číny, která sledovala internetová routovací data za každý půlrok od roku 2001 do roku 2006, ze kterých zjistili, že počet zařízení má podobné vlastnosti jako Morův zákon. Jejich zjištění bylo, že počet zařízení se zdvojnásobí každých 5 let (přesně 5,32 let). Tuto formuli použili na odhad pro rok 2020, kdy by měl počet zařízení dosáhnout 50 miliard. Všechny zmíněné odhady a měření s poměrem k počtu obyvatel země je na schématu od společnosti Cisco níže (Obrázek 2). [5]

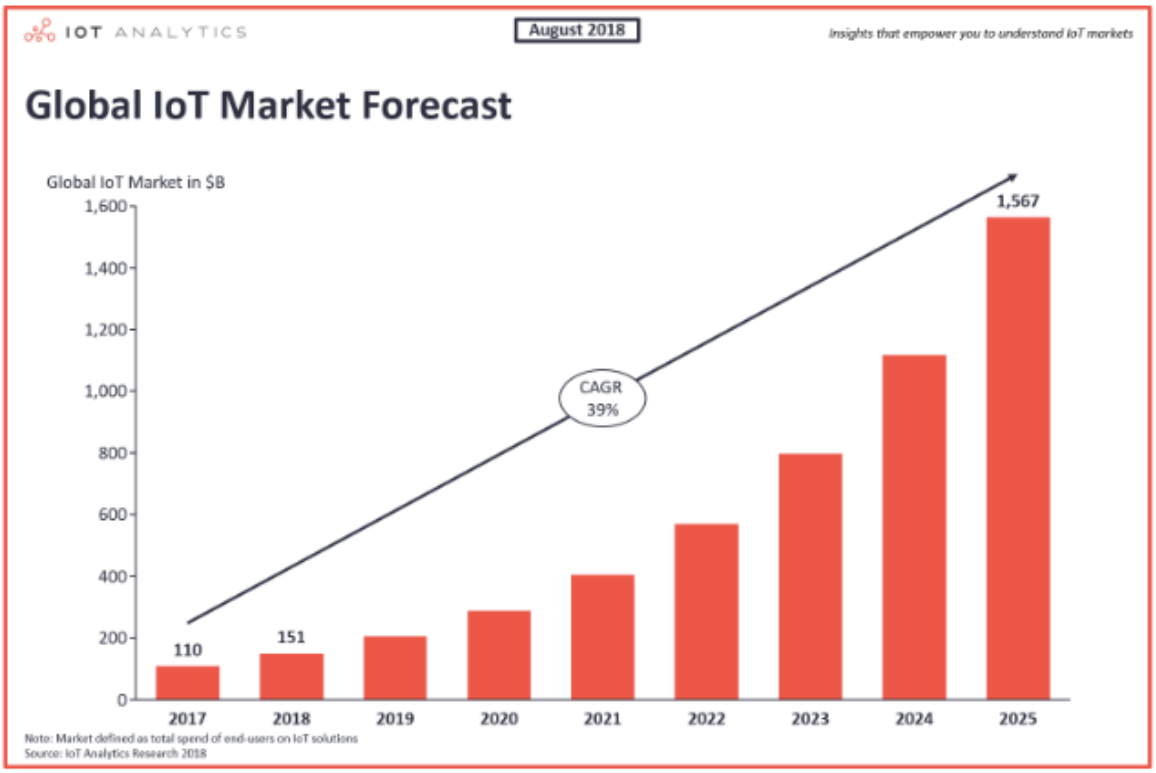


Obrázek 2 - Počet připojených zařízení. Zdroj: [5]

Data, která jsou novější, pochází ze zdroje na iot-analytics, která mluví o 17.8 miliard zařízení v roce 2018 a odhadují, že v roce 2025 bude 34,2 miliard aktivních zařízení, což se zdá být velmi pesimistické proti předpovědí Cisco.

Důležitým faktem pro situaci na trhu je také velikost trhu samotného, jehož růst podle grafu iot-analytics (Obrázek 3) je 39% a v roce 2018 dosáhl 151 miliard. Tento masivní růst je očekáván hlavně z důvodu přicházejícího mobilního připojení 5. generace (5G) a LPWA

technologie, která je přesně tvořena pro zařízení internetu věcí se svojí velmi nízkou energetickou náročností. [6]



Obrázek 3 - Odhad velikost trhu internetu věcí. Zdroj: [6]

Další podstatným faktem situace na trhu je cena samotných zařízení. Největší tuzemský prodejce elektroniky Alza.cz má speciální sekci vyhrazenou pro zařízení internetu věcí od zařízení pro chytrou domácnost - žárovky, kamery, termostaty, hlasoví asistenti po programovatelné stavebnice Arduino, Raspberry Pi. Vše se pohybuje v cenové relaci od 300 korun za nejmenší kousky stavebnic po 2000 korun, za které je možné pořídit většinu chytrých zařízení. Na stav tuzemského trhu má také vliv český výrobce IQRF technology. Ten nabízí zařízení internetu věcí spíše pro profesionálnější užití s důrazem na konektivitu zařízení. Okolo zařízení IQRF se také vytvořila IQRF aliance, která sdružuje další výrobce zařízení a stojí za mnohými konferencemi několikrát ročně. Tyto konference napomáhají konkurenceschopnosti českého trhu.

3.1.2 Využití internetu věcí

Využití internetu věcí jsou množství a je velmi rozličné ve všech oblastech běžného života, průmyslu, zemědělství, bezpečnosti a dalších. Hlavní skupiny využití by se daly rozdělit podle Porkodi do 3 oblastí (Tabulka 1). Jde o společnost, prostředí a průmysl do nich patří

například chytrá města, chytré zemědělství a voda, maloobchod a logistika, zdravotnictví a bezpečnost. Obecně jde o sběr dat pro zkvalitnění života a zefektivnění zdrojů.

Tabulka 1 - Rozdělení internetu věcí. Zdroj: [7]

Oblast	Popis	Využití
Společnost	Činnosti spojené se zlepšováním a vývojem společnosti, měst a lidí	Chytré města, chytré zemědělství, chytrá domácnost, zdravotní péče, telekomunikace, energie, obrana, zdravotnické technologie a nezávislé bydlení
Prostředí	Činnosti spojené s ochranou, monitorováním a rozvojem všech přírodních zdrojů	Chytré prostředí, chytré měření, chytré recyklování vody a ochrana před pohromami
Průmysl	Činnosti spojené s financemi, obchodními vztahy mezi společnostmi, organizacemi a dalšími subjekty	Maloobchod, logistika, řízení dodavatelského řetězce, automobilový průmysl, průmyslová kontrola, letecký průmysl a letectví obecně

V chytrých městech jde o data o úrovních dopravy, pohybu chodců, inteligentní osvětlení ulic, které se přizpůsobuje počasí. Odklonění dopravy, chytré informační tabule, kontrolovaná parkovací místa, jako je třeba projekt spinpark.cz, lepší sběr odpadů, inteligentní energetické sítě, optimalizace hromadné dopravy. Jde nejen o řízení a přímé ovládání nějakého systému, může jít jen o sběr dat, která slouží k dlouhodobé analýze, jako mapy hlučnosti dopravy, čistota ovzduší a další. S chytrými městy souvisí i chytré budovy, které mohou zařízení a analýzu dat internetu věcí použít k chytrým termostatům, kontrolu přístupu přes chytré zámky, řízení a vypínání domácí elektroniky a spotřebičů, dálková kontrola o pacienty, alarmy a čidla pro bezpečnost, management spotřeby a úspory energie a například senzory pro navádění bezpečnostních složek uvnitř budovy. Zařízení v chytrých městech mohou využívat RFID a bezdrátové senzory na malé až velké vzdálenosti. [7] [8]

Zemědělství využívá internet věcí k měření vlhkosti půdy, měření hodnoty počasí, jako vítr, vlhkost vzduchu, síla větru, úroveň vitamínu v zemině, šířka kmene vinné révy pro odhad potřeby zavlažování a živin. Za zmínku stojí projekt UTIPA na České Zemědělské Univerzitě, který se tohoto tématu dotýká. Využití internetu věcí je prakticky neomezené, důležité je získaná data analyzovat a využít správným způsobem. [7] [8]

3.1.3 Síťové a přenosové technologie

Data, která zařízení posílají, bývají malé v řádu bytů. To podporuje nové technologie, jako LPWA sítě, které právě na tyto velmi malé přenosy cílí, více o datech a jejich velikosti v kapitole 1.2. Hlavní myšlenkou je vytváření sítí ze zařízení internetu věcí, které získaná data agregují. Kapitola pojednává o různých síťových možnostech v oblasti internetu věcí.

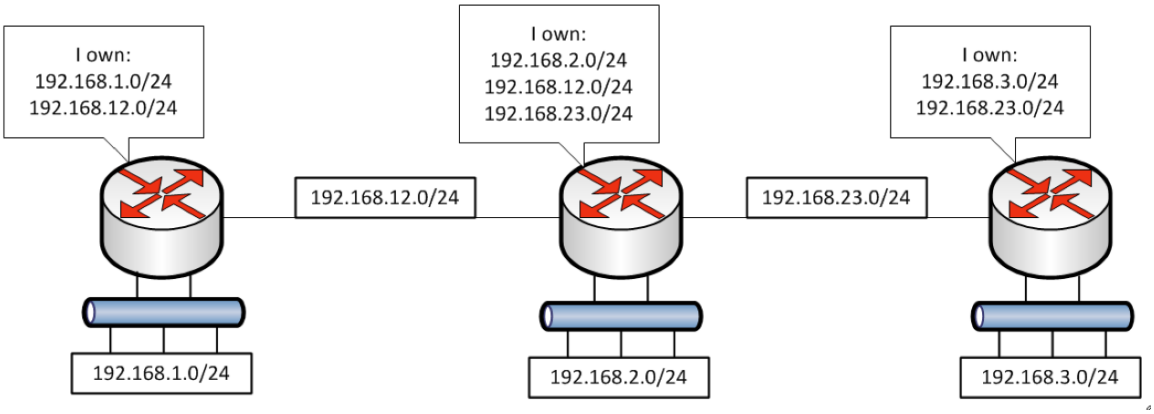
Velkým faktorem pro zařízení internetu věcí je energetická náročnost vzhledem k povaze a velikost zařízení, které se snaží být co nejmenší, mnohdy jen s AA baterií, jako například MiniUNI od společnosti SigFox, která dokáže měřit tlak, teplotu, vlhkost, světlo a má akcelerometr. Výrobce uvádí, že na dvě AA baterie vydrží 2000 dní. [9]

Kapitola popisuje veškerou klíčovou problematiku cesty dat od zařízení k middlewaru. Nejprve půjde o základní internetový protokol a jeho aktuální vývoj následně o vybrané síťové technologie, které internet věcí využívá, nebo byly zmíněny v předchozích kapitolách.

3.1.3.1 Internet protocol

Všechna zařízení v síti mají svoji jedinečnou adresu. Název této adresy je zkratkou celého protokolu tedy - IP. Tato adresa funguje jako identifikátor kam poslat příslušná data. Toto směrování nefunguje od konce ke konci, ale každý prvek na cestě ví, jak se dostat ke svým sousedům a tedy i jeho sousedi, kteří nevědí nic o jeho tabulce sousedů - routovací tabulce, vědí jak se dostat právě k němu. Tato adresa tedy není přímo adresou zařízení, ale spíše adresou cesty k němu. Díky tomuto charakteru je tedy možné schovat do vnitřní sítě prakticky neomezeně zařízení. Na praktické ukázce (Obrázek 4) je zobrazen princip fungování routingu. Router v pravé části, má ve své routovací tabulce informaci o tom, pokud chce kontaktovat adresy, které distribuuje router uprostřed, tedy ty, které vlastní a souseda - router na levé straně, musí poslat data po příslušné cestě. S touto maskou podsítě (/24) se mohou v každé sítí

schovávat další zařízení, nebo dokonce i další podsítě, které propagují dál společnou adresu pro celou síť. [10]



Obrázek 4 - Princip fungování routingu. Zdroj: Unadulteratednerdery.com - The basics of ANW2 Routing

Schopnost chytrých zařízení komunikovat s dalšími a s jejich okolím je naprosté jádro celého internetu věcí. V předchozím odstavci byl vysvětlen internet protokol a jeho unikátní adresy. Problém s internet protokolem 4. generace, IPv4 je ten, že nabízí pouze omezený počet adres a to přesně 2 na 32. Tento počet adres bude plně využit na konci roku 2019 až konec roku 2020 v závislosti na kontinentu a i přes schovávání zařízení do vnitřní sítě bude potřebovat nahrazení. [11]

Přicházející a již s fungujícím řešením je internet protokol 6. generace - IPv6, který nabízí adres 2 na 128. Velký adresový prostor není jedinou výhodou, jde také o lepší směrovací možnosti, bezpečnost a další. IPv6 byl poprvé popsán ve specifikace RFC 2460 roku 1998[12]. IPv6 je ve spojení s internetem věcí velmi důležitý, protože podporuje nezadržitelný rozvoj počtu zařízení, na který by adresový prostor starší verze protokolu nestačil. [8][13]

3.1.3.2 Wi-Fi

Wi-Fi je bezdrátová lokální síť, která je založena na standardu IEEE 802.11, označení pro jednotlivé generace se určuje podle písmen (b, a, g, n, ac, ad) za tímto standardem. Rychlost se podle generace pohybuje od 11 Mbps ve verzi b, která je nejstarší do nejnovější verze ad s rychlostmi až 7 Gbps. Pásmo Wi-Fi je 2.4 GHz, 3.6 Ghz nebo 4.9 - 5 GHz v závislosti na generaci. Dosahem se Wi-Fi pohybuje do 100 metrů od zdroje signálu, tento dosah tedy

vyžaduje stanici pro sběr dat na místě výskytu koncových zařízení v případě využití Wi-Fi a obecně není svojí energetickou náročností vhodná pro koncová čidla. [13][14]

3.1.3.3 SigFox

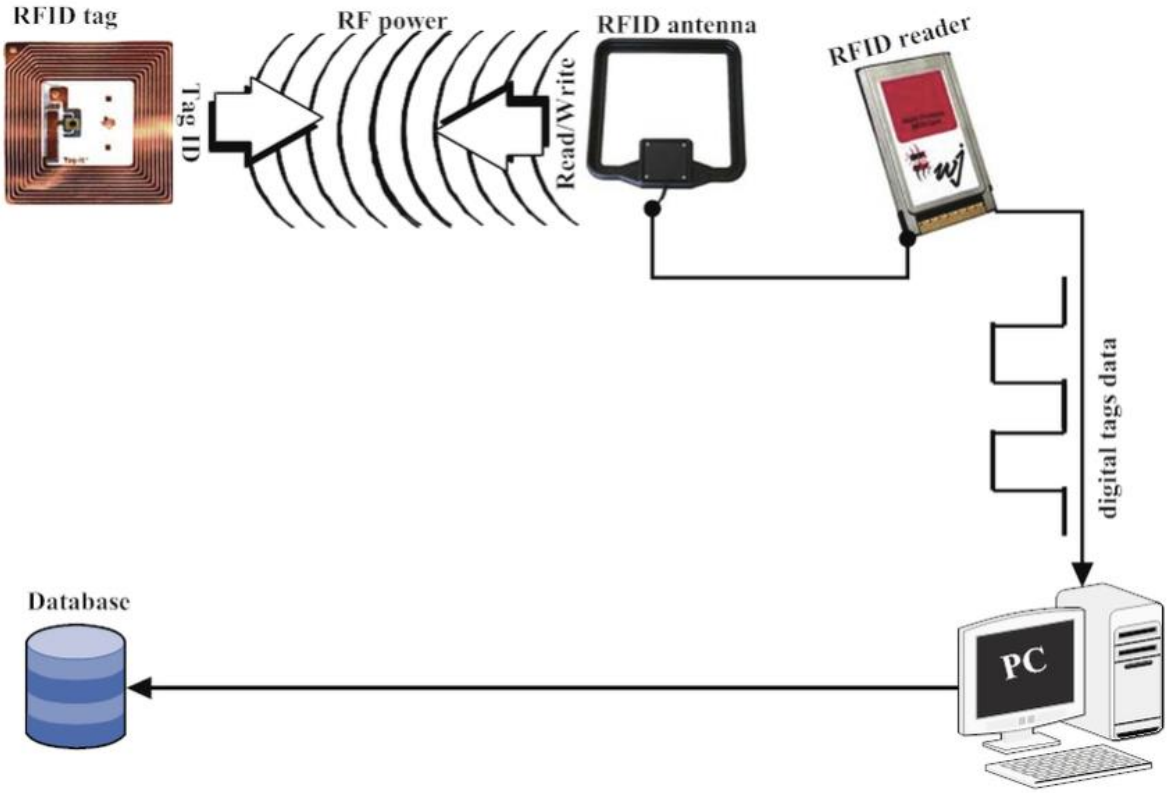
Alternativou pro delší vzdálenosti, je v předchozích odstavcích zmíněný SigFox, který se svojí spotřebou 50 microwatt je ideálním pro koncová zařízení, v poměru s telefonním přenosem, který vyžaduje okolo 5000 microwatt. To znamená, že jedna baterie s kapacitou 2,5Ah vydrží 20 let využíváním SigFox a přibližně 2 měsíce s pomocí telefonního přenosu. Dosah je 30 až 50 km mimo obydlenou oblast a 3 až 10 km v obydlené oblasti, toto je způsobeno pravděpodobně kvůli použité technologii (kmitočtu) pro úsporu energie, které jsou okolo 900 MHz. Přenosové rychlosti jsou do 1000 bps, které jsou dostačující pro přenosy dat internetu věcí.

SigFox také nabízí Cloudové řešení pro SigFox zařízení, do kterého se posílají data z jednotlivých zařízení. [13][14]

3.1.3.4 RFID

RFID je technologie pro přenos na malé (10 cm) až po delší (100m) vzdálenosti. RFID systém se skládá z tagu, který může být, jak aktivní v případě lokalizace zboží ve skladě, kdy tag bude moci být lokalizován na dálku, tak pasivní v podobě přístupové karty, dále pak anténa, která informace o tagu posílá do čtečky, která data zpracovává. Pasivní tag je napájen pomocí elektromagnetického pole čtečky. RFID se obecně rozdělují do tří skupin na nízké frekvence (125 - 134 kHz, s dosahem do 10 cm), vysoké frekvence (13,36 MHz, s dosahem do 30 cm) a poslední ultra vysoké frekvence od 856 MHz do 960 MHz s dosahem do 100 m, vhodné právě pro zmíněné sklady. Ukázku schématu fungování RFID vidíme na příkladu níže (Obrázek 5)

Na rozdíl od NFC, které je podmnožinou RFID rodiny, nemůže být čtečka zároveň i tagem a neprobíhá tedy peer-to-peer komunikace.[13][14]



Obrázek 5 - Schéma fungování RFID. Zdroj: https://www.researchgate.net/figure/RFID-work-scheme_fig1_271465664

3.2 Skladování, typy a dostupnost dat

Data, která jsou získaná ze zařízení internetu věcí mohou být různého charakteru, který vyžaduje jiný přístup k analýze. Kapitola pojednává o datech jako takových, definuje různé druhy dat a popis dat, který je neméně důležitou součástí sdílení a posílání dat. Získávání dat z vlastních zařízení není jediným způsobem přístupu k datům tohoto charakteru, online je k dispozici mnoho zdrojů, které obsahují data od katastrálních záznamů po informace o počasí. V poslední části se podíváme na velká data, jak se s nimi nakládá, jak probíhá analýza nad velkými daty a jaké jsou způsoby ukládání velkých dat. [15]

3.2.1 Otevřená data

Termín Otevřená data je velmi specifický a zahrnuje dva aspekty otevřenosti: Data jsou otevřena po legální stránce věci, což obecně znamená, že data jsou publikována pod otevřenou licencí a jsou specifikovány podmínky pro využívání. Druhým aspektem je technická otevřenost dat, kde data jsou strojově čitelná a nechráněná jak to je jen možné. Prakticky to znamená, že data jsou veřejně dostupná, použitý formát, ve kterém je soubor a jeho obsah nejsou limitovány na software, který není open source, Otevřená data mohou být volně využívána, upravena a sdílena. Tyto vlastnosti a jejich rozsah jsou popsány v licenci. To, že je možné využít otevřená data ne vždy to znamená, že jsou data dostupná naprosto zdarma a bez nějakého poplatku, ačkoliv volně a zdarma dostupné na internetu je preferováno. [16]

Příkladem otevřených dat u nás může být portál eagri.cz na adrese http://eagri.cz/public/web/mze/farmar/LPIS/, na kterém je možné najít data týkajících se českého zemědělství, jako jsou informace o půdě, vinicích, zvířatech a mnoho dalšího. Podobným portálem evropského typu, kde jsou data jak ve formátech strojově čitelných (txt, json, xml a dalších), tak ostatních (doc a dalších), jde o data z většiny států evropy nad kterými dělá portál různé analýzy použitelnosti ve snaze jít směrem strojové čitelnosti, data jsou na adrese https://www.europeandataportal.eu/. Data na tomto portálu jsou sesbíraná z různých státních datových fondů pro jednotlivé státy. Dalším evropským portálem, který data nesbírá, ale uživatelé je nahrávají je EU Open data portal dostupný na adrese http://data.europa.eu/euodp/en/home. Obdobným státních datovým fondem je americký portál

https://www.data.gov/, který sbírá data v USA, převážně zemědělského charakteru a s ním spojeným.

Velmi důležitou vlastností otevřených dat je jejich popis. Například zmíněný EDP - evropský datový portál neukládá otevřená data samotná, ale jen jejich popis, tedy jejich metadata. Metadata jsou strukturovaná informace, která poskytují kontext a informaci o popisovaném objektu. Tyto data umožňují znuvupoužitelnost, konzistentnost a celkovou kvalitu dat. Meta informace mohou být popisovány nejen data v dokumentech, kde budou uvádět, jestli jde o text, číslo či popis dat samotný. Velmi běžným využitím metadat je také popis HTML stránek pomocí meta tagů. Příklad meta tagů je vidět na příkladu níže (Obrázek 6) [17]

```
<meta name="description" content="KIT, PEF, ČZU, katedra, it informatika,
spolupráce">
<meta property="og:url" content="http://kit.pef.czu.cz/">
<meta property="og:type" content="article">
<meta property="og:title" content="Katedra informačních technologií">
```

Obrázek 6 - Příklad využití metadat. Zdroj: kit.pef.czu.cz

3.2.2 Velká data

Podle studie Cisco bylo v roce 2018 vygenerováno 400 ZB (zetta bytů) dat internetem věcí, toto číslo by mělo narůst na 850 ZB v roce 2021. Takto ohromná data přináší mnohá úskalí. Velká data jsou data, jejich rozměr, distribuovanost nebo rozdílnost přináší nutnost využití nových architektur, softwarových řešení analytických nástrojů. Velká data by mohla být definována třemi charakteristikami a to velikost, různorodost a proměnlivost. Velikost znamená jejich rozměr, který se měří v stovkách TB (tera bytů) až několika PB (peta bytů), různorodostí je myšlen nejen jejich charakter, ale také datový formát a přístup k analýze jednotlivých dat. Proměnlivost znamená, rychlost jakou se data mění a obecně jak rychle jsou vytvářena. [18]

3.2.3 Uložení velkých dat

Jedna z prvních věcí, které organizace musí vyřešit ohledně velkých dat, je otázka, kde a jak budou data uložena po jejich získání. Do tradiční metody strukturovaného ukládání a získávání dat patří relační databáze, datová tržiště a datové sklady. Získaná data z internetu věcí jsou nahrány do datového skladu pomocí Extract, Transform, Load (ETL), nebo Extract, Load, Transform (ELT) metodik. Tyto metodiky získají data ze zařízení, data transformují, na

požadovanou velikost a strukturu a uloží data do databáze nebo datového skladu. To znamená, že data jsou očištěna o data chybná, transformována a indexována než jsou dány k dispozici pro data mining, online analýzu a další zpracování.

Pro ukládání a správu nestrukturovaných nebo nerelačních dat existuje jiné řešení tvz. nerelační databáze, mezi nerelační databáze se většinou řadí například NoSQL jako je MongoDB, Hadoop a další. Databáze NoSQL se zaměřují na masivní škálování, flexibilitu datového modelu, zjednodušený vývoj a nasazení aplikací, které je využijí. Na rozdíl od relačních databází oddělují NoSQL databáze správu dat a ukládání dat. Tyto databáze se zaměřují na vysoce výkonné ukládání dat a umožňují příkazy pro manipulaci s daty, aby byly zapisovány do aplikační vrstvy, než aby byly přímo v jazyku specifickém pro konkrétní databázi lokálně.

Alternativou k NoSQL databázi je framework Hadoop. Hadoop se zaměřuje na provádění analýz velkých dat se svou spolehlivostí, škálovatelností, spravovatelností. Těchto vlastností dosahuje použitím implementace paradigmatu MapReduce. MapReduce a Hadoop samotný je popsán v následující části. [19]

3.2.3.1 Hadoop

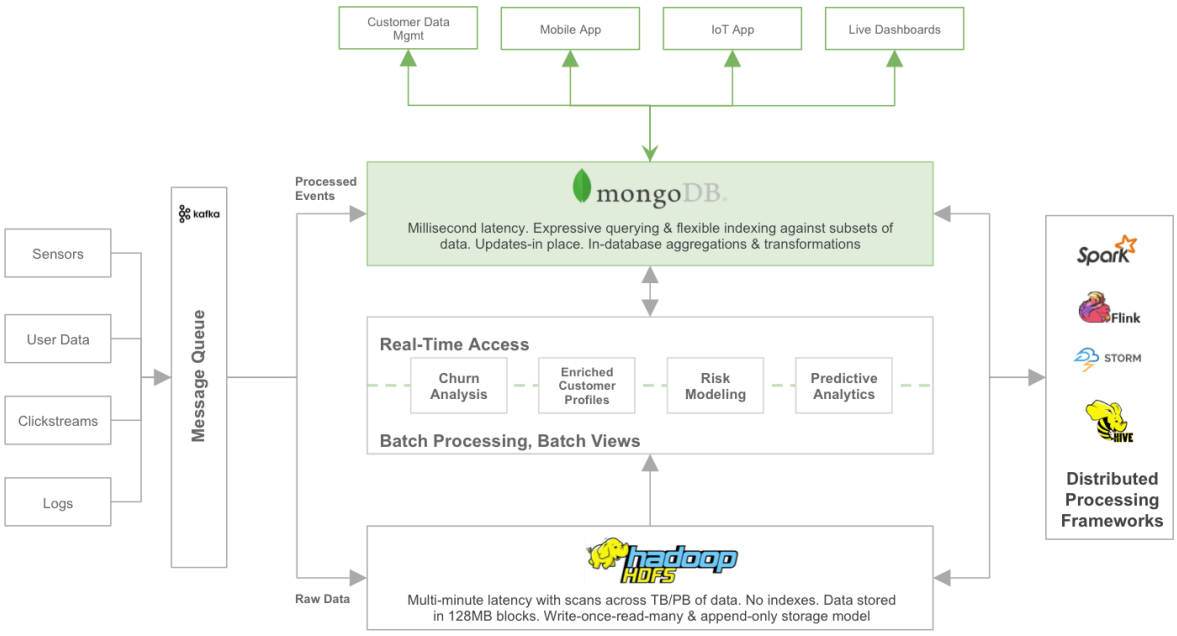
Framework Hadoop je standardem v oblasti ukládání velkých dat od společnosti Apache Software Foundation napsaný v programovacím jazyku Java. Hadoop využívá distribuovaný souborový systém na který přistupuje pomocí MapReduce paradigmatu, které je tímto způsobem použito poprvé pro mnohem širší soubor úloh.
Velkým rozdílem proč je Hadoop tak jednoduše škálovatelný je celé nastavení systému, že díky MapReduce může být prakticky neomezeně distribuován včetně datové redundance, ale také, že data samotná nejsou posílána, čímž se liší od standardních databází. Hadoop posílá jednotlivé funkce k datům, kde se vykonávají.

Do Hadoopu se data posílají jako nepřetržitý sériový stream. Data jsou určena k analýze jako celek, jednotlivá data jsou bezcenná. Hadoop je databáze určená pouze pro velká data je nevhodný z důvodu své struktury. Pro business data, jako jsou záznamy o uživatelích, pracovnících a další strukturovaná data je potřeba využít jinou databázi, která je pro toto

standardní užití určena. Ve zdrojích byly uvedeny příklady pro business databázi MongoDb nebo PostgreSQL.

Praktický příklad je vidět na příkladu dole (Obrázek 7), na kterém jsou vidět zleva přicházející data ze senzorů, zařízení, logů a dalších přes frontu, které tato data rozděluje a zajišťuje posílání, jako jsou například RabbitMQ a nebo Kafka. Zpracovaná data dále putují do zmíněné NoSQL databáze MongoDB a všechna ostatní do datového skladu Hadoop.

Napravo a nalevo je vidět analýza middleware řešením pomocí nástrojů jako jsou Spark pro distribuovanou datovou analýzu pro dávkové zpracování s tolerancí chyb. Využívá se pro zrychlení MapReduce operací a Storm, který zpracovává příchozí stream dat. Celá analýze je poté uložena do MongoDB a poskytnuta koncovému zobrazení. [20]



Obrázek 7 - Navrhnutá architektura pro analýzu dat. Zdroj: [20]

3.2.4 Analýza velkých dat

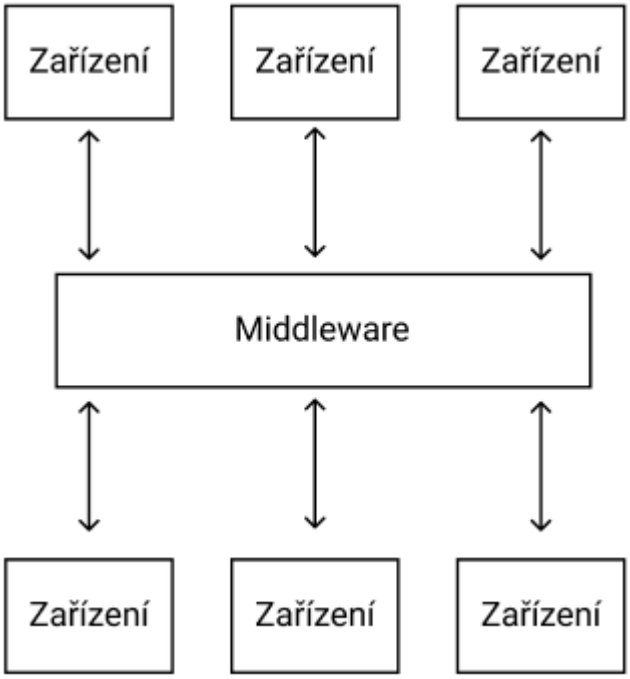
Po uložení velkých dat přichází analýza. Podle [21] existují čtyři kritické požadavky na analýzu a zpracování velkých dat. Prvním požadavkem je rychlé načítání dat. Protože disk a síťové řešení narušuje provádění dotazů během načítání dat, je nutné minimalizovat dobu načítání. Druhým požadavkem je rychlé zpracování dotazu samotných. Aby byly splněny požadavky na analýzu velkých dat v reálném čase, je mnoho dotazů kritických na dobu

odezvy. Struktura umístění dat tedy musí být schopna zachovat vysokou výkonnost analytických dotazů i při zvyšování jejich počtu. Třetím požadavkem na analýzu dat je efektivnost využití úložného prostoru. Vzhledem k rychlému růstu objemu dat z internetu věcí musí uložiště umožňovat škálovatelnou úložnou kapacitu a výpočetní výkon. Omezené místo na disku vyžaduje, aby byla analýza dat dobře optimalizována pro velikost výstupních dat a zbytečná data nebyla nijak uložena, aby se tím maximalizovalo využití prostoru. Posledním, čtvrtým požadavkem je možnost budoucích změn v analýze bez nutnosti změnit architekturu. Vzhledem k tomu, že soubory velkých dat jsou analyzovány různými aplikacemi a uživateli pro různé účely a různými způsoby, základní systém by měl být vysoce adaptivní na neočekávanou dynamickou změnu v analýze dat a neměl by být svázaný na konkrétní aktuální implementaci analýzy. [22]

3.3 Middleware

Kapitola definuje middleware, pojednává o důvodech a způsobech jeho užití v praxi. Middleware je vrstva nebo sada dílčích vrstev softwaru mezi různými úrovněmi aplikace. Cílem middleware je skrýt detaily různých technologií, protokolů, síťového prostředí, duplicitních dat, jako vyšší cíl je odstínit programátora od problémů, které nejsou přímo relevantní pro jeho rozsah a poslání vývoje pro konkrétní aplikaci. Middleware navíc maskuje heterogenitu počítačových architektur, operačních systémů, programovacích jazyků a síťových technologií pro usnadnění programování a správu aplikací.

Důvodů pro použití middleware v IoT je mnoho od bezpečnosti o které pojednává pozdější kapitola, tak také jeho abstrakci od zařízení samotných, datový specialista analyzující data přicházející z internetu věcí se tak nemusí starat o sběr dat, možné chyby a duplikaci dat. Middleware, o kterém pojednává tato práce není abstrakcí zařízení samotných, jejich vstupně-výstupních systému, ani abstrakce networkingu, jde o middleware, který zpracovává příchozí data ze zařízení. Příkladem takové architektury může být příklad (Obrázek 8), na kterém je znázorněna funkce middleware, který je mezi pomyslným datovým tokem mezi zařízením dole a koncovou aplikací, webovou stránkou či dashboardem nahoře. [23]



Obrázek 8 - Schéma architektury middleware v internetu věcí. Zdroj: vlastní práce

3.3.1 Architektonické vzory middleware

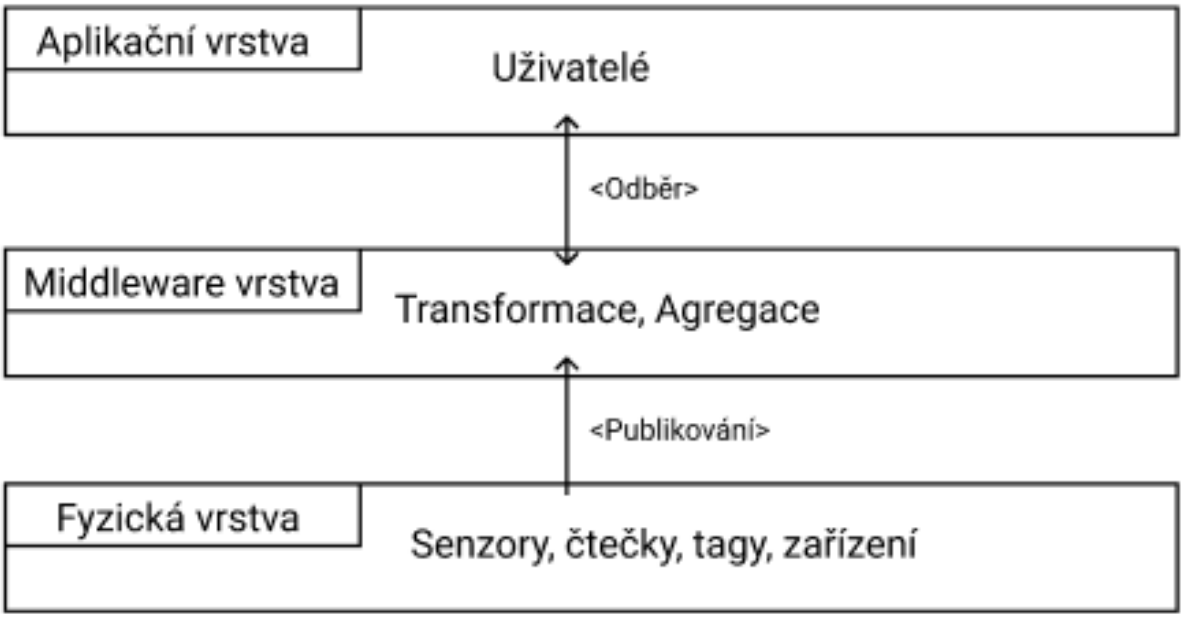
Middleware se dají rozdělit do několika skupin podle jejich návrh [24] do sedmi skupin: řízené událostmi, orientované na služby, řízené virtuální stroji, řízený agenty, podle paradigmatu sdílené paměti (tuple space), orientované na databázi a specifickou pro aplikaci. Některé middleware jsou kombinací více skupin, které takto profitují z výhod zvolených skupin. Popsány budou tři nejpoužívanější a základní, ze kterých se dají odvodit i ostatní. Middleware založené na událostech kde middleware hraje aktivní roli v toku dat a posílá data sám dal do aplikační vrstvy, middleware orientovaný na služby kde je pasivní a data poskytuje a vystavuje API rozhraní. Poslední popsaným vzorem bude middleware řízené

virtuálním strojem, ve kterém je middleware pasivní z hlediska aplikační vrstvy a data jen poskytuje, ale aktivní vůči fyzické vrstvě do které nasazuje virtuální stroje, na podobném principu vlastní propagace pak funguje třeba architektonický vzor middleware řízený agenty.

3.3.1.1 Založené na událostech

První skupinou middleware, jsou middleware založené na událostech. Všechna interakce a komunikace v komponentách, aplikacích a všech ostatních účastnících je založena na přijaté události. Tato skupina middleware využívá návrhového vzoru Pozorovatele, kde koncový uživatel, tedy dashboard, aplikace webová stránka naslouchá změnám dat a v případě změny se updatuje. Události se šíří z odesílajících aplikačních komponent (poskytovatelů) do přijímajících aplikačních komponent (spotřebitelů). Systém middleware založeného na událostech se může sestavovat z potenciálně velkého počtu aplikačních komponent (entit), které poskytují a spotřebují události. Podmnožinou middleware založených na událostech jsou middleware založené na zprávách (Message-oriented middleware (MOM)). V tomto modelu se komunikace zakládá na zprávách, které zahrnují extra-metadata ve srovnání s událostmi. Obecně platí, že zprávy nesou adresy odesílatele a přijímatele a jsou doručovány konkrétními podmnožinami účastníků, zatímco události jsou vysílány všem účastníkům.

Příklad schématu middleware založeného na událostech je vidět na příkladu níže (Obrázek 9), na kterém je v horní části aplikační vrstva reprezentující dashboardy, aplikace a webové stránky, uprostřed je vrstva middleware, která transformuje data, o ukládání se může v tomto případě starat jak aplikační vrstva, tak middleware vrstva a nejnižší - fyzická vrstva reprezentuje koncové zařízení a senzory internetu věcí. Tato architektura je vhodná pro menší řešení kvůli jednoduchosti middleware. [24][25]

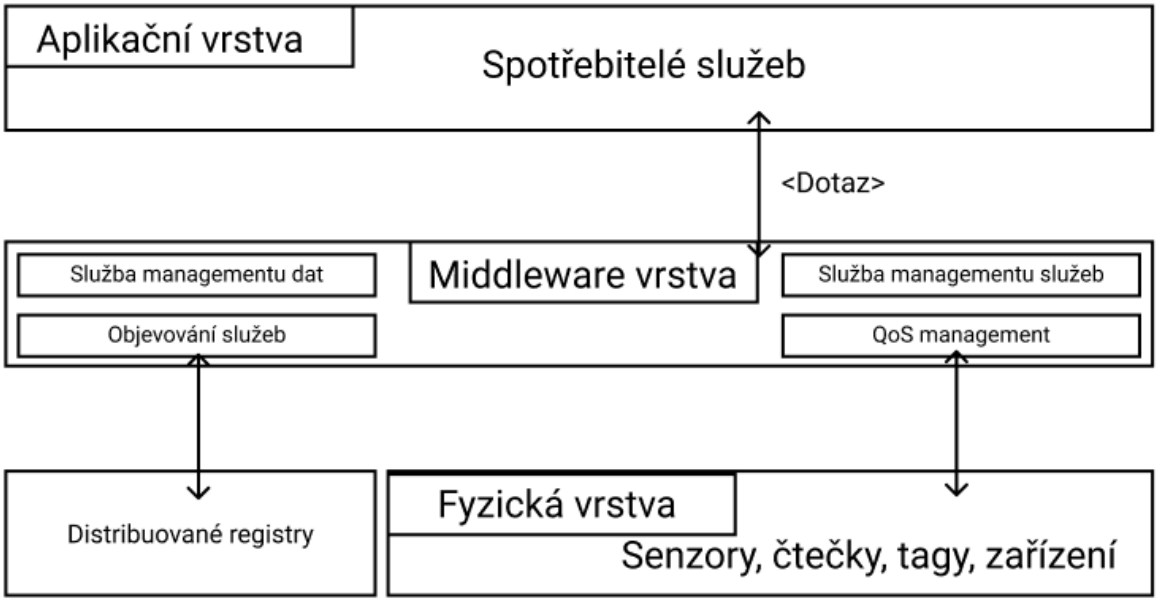


Obrázek 9 - Schéma middleware založeného na událostech. Zdroj: vlastní práce

3.3.1.2 Orientované na služby

Paradigma designu orientovaného na služby vytváří software nebo aplikace jako služby. Service-oriented computing (SOC) do které tato skupina middleware spadá je založen na přístupech architektury orientované na služby (SOA - software-oriented architecture) a byl tradičně používán v podnikových IT systémech. Charakteristiky SOC, jako je technologická nezávislost, nezavazování a nezávislost systémů mezi sebou, znovupoužitelnost služeb, schopnost složení služeb z microservice a zjistitelnost služeb v síti, jsou charakteristiky potenciálně přínosné pro aplikace IoT. V internetu věcí, kde jsou síťová spojení na velmi velkých rozměrech, zařízení jsou energeticky omezena a jejich mobilní potenciál dělá kompozici služeb a zjistitelnost služeb velmi obtížnou Middleware orientované na služby zlepšují tyto překážky díky svým distribuovaným registrům, která umožňuje adaptivní kompozici služeb, když služby nejsou k dispozici. Middleware je tedy rozdělen na dvě části,

jedna co spravuje zařízení, jejich zprávy a spravuje případně odchylky a chyby v komunikace a druhá, která využívá služeb dalších middleware a sdílí s nimi data, schéma middleware orientovaného na služby je vidět na příkladu níže (Obrázek 10). Tento typ middleware je velmi rozšířený díky svým výhodám a možnosti aplikační vrstvy být pouze zobrazovací médium. [24][25]

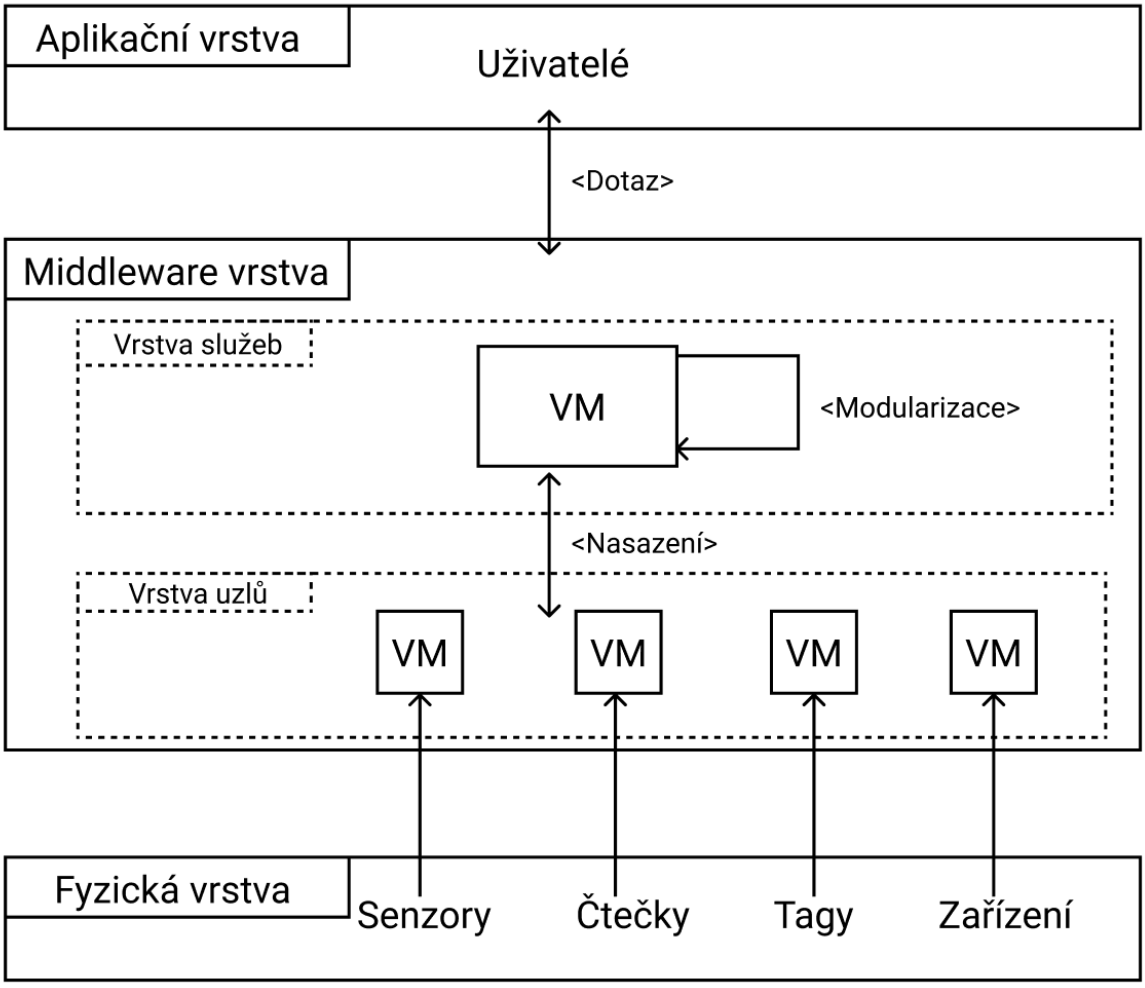


Obrázek 10 - Schéma middleware orientovaného na služby. Zdroj: vlastní práce

3.3.1.3 Řízené virtuálními stroji

Vzor middleware orientovaný na virtuální stroje poskytuje podporu pro bezpečné zavádění a exekuci kódu uživatelské aplikace pomocí virtualizace infrastruktury. Aplikace jsou rozděleny do malých samostatných modulů, které jsou nasazovány a distribuovány po celé síti internetu věcí. Každý uzel v síti reprezentuje skupinu zařízení, které jsou spravovány konkrétním virtuálním strojem specifickým pro konkrétní potřeby skupiny zařízení internetu věcí jak je vidět na příkladu níže (Obrázek 11).

Tento přístup řeší architektonické požadavky, jako jsou vysokoúrovňové programování abstrakcí, vlastní správa a přizpůsobivost procesů a podpora transparentnosti v distribuovaných heterogenních infrastrukturách mezi jednotlivými zařízeními internetu věcí. Virtuální stroje lze rozdělit do dvou kategorií: virtuální stroje na úrovni middleware (virtuální stroje jsou umístěny mezi OS a aplikacemi) a druhou kategorii virtuálních strojů na systémové úrovni (nahrazují celý OS). Virtuální stroje na úrovni middleware přináší schopnost například paralelních procesů pro operační systém běžící pod nimi. Virtuální stroje na úrovni systému šetří prostředky, které by jinak OS využíval. [24][25]



Obrázek 11 - Schéma middleware řízeného virtuálními stroji. Zdroj: vlastní práce

3.3.2 Virtualizace

Skupina middleware řízeného virtuálními stroji zmiňuje problematiku virtualizace, která bude využívána později v praktické části. Virtualizace je proces, ve kterém je spouštěna virtuální instance počítačového systému ve vrstvě abstrahované od reálného hardwaru počítače.

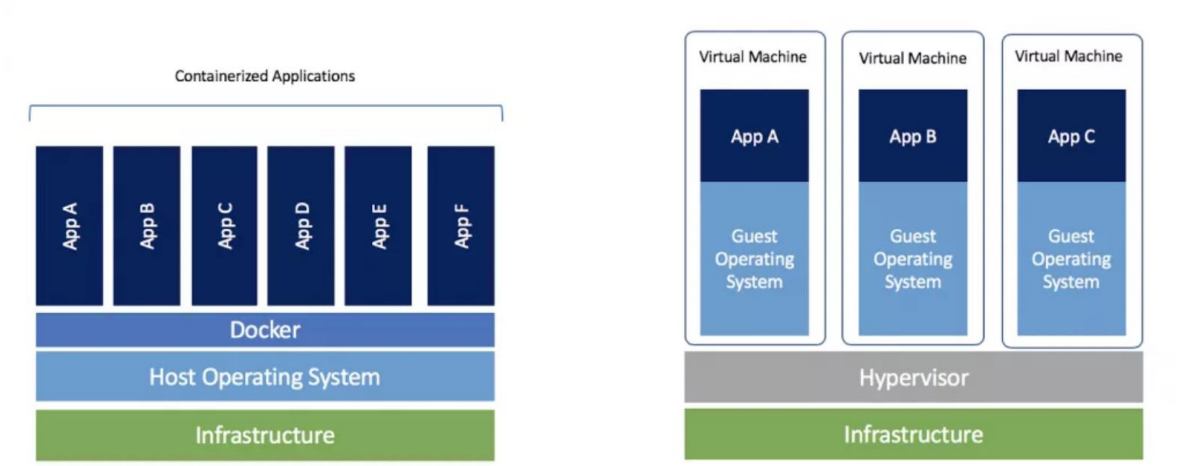
Nejčastější případ užití je více instancí jiných operačních systému vedle sebe, které umožňují uživateli využívat aplikace z jiných operačních systémů. Výhodou pro administrátory serverů je možnost provozovat různé operační systému na jednom fyzickém stroji a další podstatnou částí je pak rozdělení velkých systémů do menších částí a tak umožnit více uživatelům nebo aplikacím využívat stejný systém zároveň ve stejnou chvíli. Virtualizace také umožňuje izolování systémů jak mezi sebou tak od hostujícího operačního systému a ten je tak v bezpečí před procesy probíhajícími na jiném virtuálním stroji pod stejným hostitelem.

Z virtualizací se vážou tři důležité výrazy, které je potřeba pro pochopení problematiky vysvětlit virtuální stroj, hypervisor a kontejner. Virtuální stroj byl již z části popsán, jde o emulovaný počítačový systém, který běží na jiném systému, z pravidla jiném operačním systému. Přístup virtuální stroje do hostitelského systému je omezen a funguje spíše nad abstrakcí jednotlivých fyzických komponent, jako je CPU a paměť, která je s diskovým prostorem přiřazena virtuálnímu stroji. Virtuální stroj může mít i přístup k vstupně-výstupním zařízením jako jsou grafické karty, USB zařízení a další hardwarové komponenty. Pokud je virtuální stroj uložen na virtuálním disku, říká se tomuto diskový obraz, tento disk může obsahovat, všechny potřebné soubory a uložiště k rozběhnutí virtuálního stroje.

Druhým výrazem je hypervisor, který je jako vrstva mezi zařízením a virtuálním strojem. Hypervisory se rozdělují na dvě skupiny, skupina jedna, které se říká hypervisor “na železe” (z anglického bare metal), což je skupina hypervisorů, které běží přímo na hostitelském hardwaru a tak šetří prostředky z důvodu vynechání potřeby hostitelského operačního systému. Druhou skupinou jsou hypervisory typu 2, které jsou hostované na operačním systému a mohou být libovolně spuštěny a zastavováni jako běžná aplikace. Podskupinou druhého typu jsou KVM (Kernel-based virtual machine), tedy virtuální stroje řízené kernelem, které jsou hostovány přímo na jádru operačního systému a nikoliv na jeho aplikační vrstvě. [26][27]

Dalším vylepšením virtuálních strojů je využití vrstvy, která funguje jako most nebo kontejner pro jednotlivé virtualizované aplikace, to zaručuje lepší agilnost a šetří zdroje, rozdílná schémata můžeme vidět na příkladu níže (Obrázek 12), ve kterém je nalevo vrstva aplikace Docker, která běží v hostitelském operačním systému a přímo v ní pak jednotlivé aplikace. Na rozdíl od toho, standardní virtuální stroje využívají zmíněný Hypervisor, ve

kterém pak běží jednotlivé operační systémy s jednotlivými aplikacemi. Kontejner tedy funguje jako izolovaný proces, který sdílí stejný hostující operační systém, včetně jeho knihoven a dalších souborů potřebných k funkci. Kontejnery tedy nejsou naprosto nezávislé stroje, ale stále mohou být zpřístupněny z vnějšku a jejich účel se spíše zaměřuje na izolování konkrétních aplikací a programů než velkých víceúčelových serverů. [27][28]



Obrázek 12 - Docker kontejnery v porovnání s virtuálními stroji Zdroj: [27]

3.3.3 Bezpečnost

Vývoj internetu věcí už podle studie Cisco zmíněna v kapitole 1.1.1 Situace na trhu naznačuje, že internet věcí a jeho rozvoj bude mít na nás velký dopad a to z důvodu, že internet věcí používá zařízení z běžného života. Výhody, které přináší internet věcí jsou značné, ale i přes to je téma bezpečnosti velmi důležité. Dokonce bylo toto téma předmětem DEFCON a BlackHat konference, které se zaměřuje na otázky okolo hackingu. [29]

Zpráva, která byla vydána společností HP zmiňuje, že 70% zařízení internetu věcí jsou náchylné na prolomení bezpečnosti. Studie prováděla testy na deseti nejvíce využívaných zařízeních a našla 250 bezpečnostních chyb. V průměru, 25 bezpečnostních zranitelností bylo nalezeno pro každý z testovaných zařízení. Nejčastějšími chybami byly problémy s ochranou soukromí, nedostatečná autorizace, nedostatečné nebo naprosto chybějící šifrování síťového přenosu, nezabezpečené webové rozhraní a nedostatečná softwarová ochrana. Jednou z dnes už běžných scénářů může být nabourání do systému auta a to jak pro získání fyzického přístupu do něj - otevření dveří, tak převzetí úplné kontroly nad jedoucím vozidlem. [30][31]

Hlavní zájmem při vývoji middleware pro internet věcí je vyhýbání se bezpečnostních chyb a krádeži dat a fakt, že internet věcí nezahrnuje jen počítače, ale zařízení v každodenním životě tuto skutečnost ještě ukotvují. Studium nízko úrovňových protokolů k zajištění bezpečnosti a soukromí v centralizovaných a distribuovaných systémech scénářích pro internet věcí [32] zmiňuje cíle komunity a odborníků napravovat tyto chyby za účelem stálého zlepšování bezpečnosti. Jak bylo zmíněno, negativní dopad způsobený mnohými chybami v internetu věcí nesmí být při vývoji middleware zapomenut. Nedostatečná definice standardů a protokolů by mohla narušit vývoj a aktuální rozmach internetu věcí. [32][33]

3.3.4 Vývoj a distribuce dat

Aktuální dostupné middleware služby online, spadají do tří kategorií v rámci vývoje, jde o knihovny pro konkrétní programovací jazyk, spustitelný obraz v Dockeru, který funguje jako samostatná hotová platforma, která konzumuje REST API, nebo jiný způsob rozhraní pro přijetí dat a poslední je cloudové řešení bez nutnosti uživatele, jakkoliv něco lokálně měnit. Některé služby tedy vyžadují kompletní naprogramování vlastního řešení. Od middleware služeb se očekává, že data přijmou z koncového zařízení internetu věcí HTTP voláním, data poté zpracují v kontextu s ostatními ,tak ,aby byly strojově čitelná. Uloží je do vhodné databáze a dál pošlou informaci o agregaci nových dat v případě middleware založeného na událostech nebo jen vystaví REST či SOAP rozhraní pro vizualizaci na zaslání dotazu. REST je architektura rozhraní, která na bázi doporučení popisuje jak vytvořit rozhraní pro jednoduché vytváření, čtení, editaci a mazání dat. SOAP je protokolem a stojí za ním z části Microsoft, který již přesněji definuje jak sdílet data - XML soubory přes síť pomocí HTTP volání. [33]

Data se mohou ukládat lokálně do databáze nebo do nějakého veřejně dostupného katalogu otevřených dat, v druhém případě by měla tada splňovat jistou kvalitu. Kvalita uložených dat se dá hodnotit podle pěti bodového hodnocení podle Tim Berners-Leeho, v případě, že jsou data dostupná online, dostanou 1 bod, další bod za to, že budou strojově čitelná, tedy jsou procesovatelná počítačovým systémem ve vhodném strojově čitelném formátu - JSON, XML nebo nejrozšířenějším CSV. Při použití otevřeného formátu, tedy formátu, který jde otevřít v open-source programu, který je volně dostupný dostanou další bod. Další bod za použití RDF (Resource Description Framework - systém popisu zdrojů). RDF je ontologický jazyk,

vyvinutí konsorciem W3C pro popis dat. Poslední pátý bod dostanou data za použití zmíněného RDF s odkazy na ostatní související datasety. Middleware by měl umožňovat volně nakládat s přijatými daty a neomezovat tak uživatele. [34]

3.3.5 Správa zařízení

Správa samotných zařízení internetu věcí je velmi podstatnou součástí funkcionality middleware. Zařízení by měla být schopna detekovat a objevovat své sousedy - další zařízení internetu věcí v sítí v případě chytrých sítích. Pro účel této práce je zamýšleno používání zařízení, které jsou jednoduchá a jednoúčelová nemající žádný pojem o kontextu a sousedech. Nicméně zařízení samotná by měla být spolehlivá, tolerantní k chybám, adaptující se, optimalizována pro spotřebu zdrojů a schopnost sdílet všechny potřebné informace o svém stavu. Od správy zařízení se očekává možnost přidání nového zařízení, jeho monitoring, management, kontrola stavu a update firmware.

Existuje několik přístupů ke správě zařízení a spíše tedy k jejich objevování. To nejjednodušší je manuální spojování s middleware, které je 1:1 pro všechnu správu. Sofistikovanějším způsobem je abstrakce skupin zařízení do sémantických modelů. Tento přístup je v sémantické architektuře řízené modely (Semantic Model Driven Architecture). Architektura MDA představuje koncept abstraktních zařízení, které mohou být spojeny s fyzickými zařízeními 1:1 nebo m:1. Informace a data o zařízeních a typ zařízení je popsán pomocí tohoto modelu. Nové zařízení se speciálními vlastnostmi a atributy mohou být přidány využívající stávající modely, ze kterých vytvoří pod-třídy přídávající právě tuto speciální vlastnost. Další možností jsou agenti procházející síť, kteří byly popsány v kapitole 1.3.1 Architektonické vzory middleware. [35][36]

4 Vlastní práce

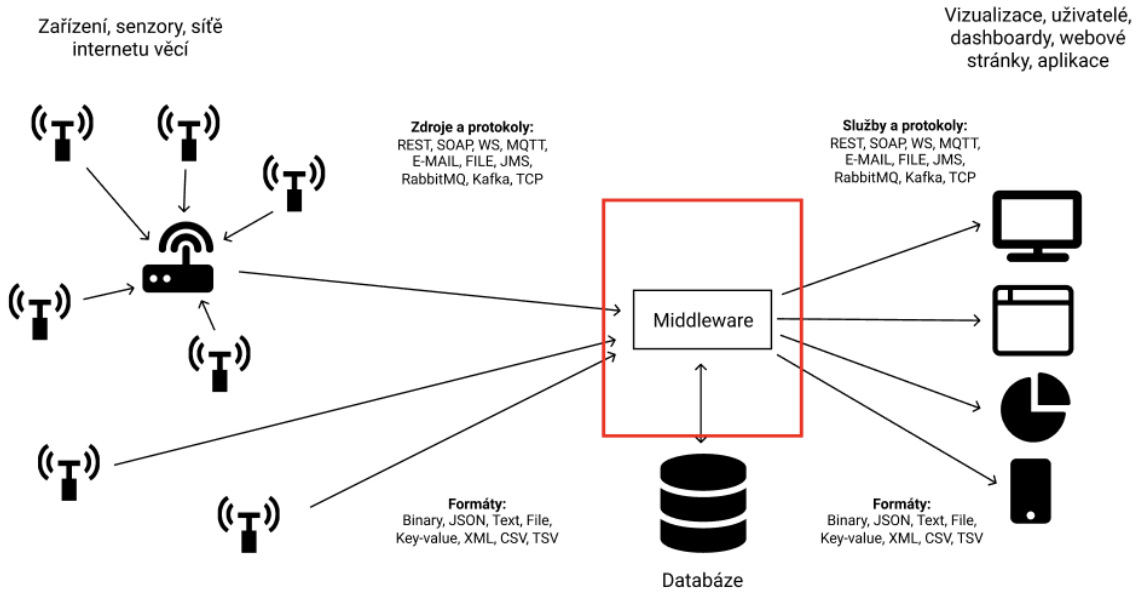
Ve vlastní práci budou systematicky popsány nejdříve požadavky na middleware, ze kterých budou vyvozená vhodná kritéria pro hodnocení a každé kritérium bude popsáno podle jaké škály bude hodnoceno.

V další kapitole budou zvoleny dostupné middleware služby, které budou představeny, jak z hlediska jejich architektury, funkcionalit, tak z hlediska všech souvislostí spojených s vývojem a používáním middleware služeb pro účel agregace. Na základě těchto požadavků a kritérií budou vybrané middleware služby zhodnoceny pomocí vícekriteriální analýzy variant, ze které bude vybrána vhodná kompromisní varianta pro konkrétní případ užití. Váhy a kritéria pro vícekriteriální analýzu variant budou konzultovány s experty v oblasti internetu věcí.

4.1 Požadavky na middleware

Za middleware pro účel této práce je považován online dostupný nástroj pro agregaci, transformaci a následnou distribuci dat internetu věcí. Požadavky vyplývající z lineární rešerše na základě užití middlewaru jsou popsány v této kapitole.

Na příkladu níže (Obrázek 13) je vidět navrhovaná architektura internetu věcí, nalevo jsou zařízení, senzory, měřiče internetu věcí, může jít i o aplikace a programy, ze kterých se získávají data, ty pomocí MQTT - nenáročný protokol pro internet věcí, zmiňovaného SOAP a REST rozhraní a webových služeb jsou odeslány do middleware. Tento middleware je předmětem práce a právě jeho alternativní služby budou v dalších kapitolách zhodnoceny. Pravá strana prezentuje koncové zobrazení výsledků. Tato architektura může mít mnoho dalších částí od před-analýzy jako Kafka, využití message brokera nebo více middleware, které by spolu komunikovaly.



Obrázek 13 - Schéma internetu věcí. Zdroj: vlastní práce

Na pořadí kritérií nezáleží pro žádnou z analýz v této práci, ale bylo zvoleno tak, že kopíruje proces vývoje samotného middleware. Počínaje licenci, pak zjišťováním informací z dokumentace, samotný vývoj software či konfigurace služby, podpora formátů v rámci systému služby, možné připojení nebo již připojené databáze pro uložení příchozích dat, distribuce získaných dat po jejich zpracování a transformaci a následná možnost vizualizace, která je součásti systému. Záměrně není zmíněn požadavek na rychlost nebo výkon, který pro případ užití této práci je nepodstatný. Test měření by nebyl nijak prokazatelný v malých dávkách a rozdíly by byly spíše v konkrétní implementaci ze strany vývojáře, druhým faktorem vynechání těchto požadavků je rozličnost robustnosti middleware služeb, které

mohou v některém případě data jen získat a dále nepoužít a v jiném uložit do cache či databáze a na pozadí provádět další funkce. Ve větších dávkách dotazů by pak šlo o porovnávání jazyků a kvalitě jejich garbage collectorů.

V dnešní době, ve které je velký rozmach začínajících firem - startupů a možnosti rozjíždět kampaně na crowdfundingových webech bude případem užití pro který se budou middleware služby hodnotit, malá začínající firma s 10 zaměstnanci z čehož většina jsou vývojáři. Pro neutralitu jde o zkušené vývojáře, kteří umí všechny potřebné programovací jazyky. Důvodem zvolení právě této menší firmy je ta, že větší firmy by pravděpodobně zvolily svoje řešení vyvinuté na míru jejich potřebám a tyto dostupné služby by tak byly buďto naprosto nevyužity nebo by posloužily jen jako základ pro vyvinutí právě onoho vlastního řešení.

4.1.1 Licence

Hlavním cílem, co se týče licence, je její otevřenost. Preferovány jsou open source projekty, které tak budou vyhovovat jaké malé firmě, která tak uvidí do kódu což usnadňuje pochopení problematiky, tak pro střední a velké firmy, které tento kód mohou použít jako základní stavební kámen pro vlastní řešení. Licencí je také myšlena případná cena za službu samotnou, která musí být minimálně přístupna jako časově omezená trial verze nebo demoverze ve které jde demonstrovat všechny funkcionality. Vyloučeny tak byly jakékoliv middleware služby, které nejsou volně dostupné jako třeba Kaa project webové stránky dostupné na www.kaaproject.org, který v době psaní této práce nebyl dostupný ke stažení zdarma.

Toto kritérium se hodnotí za plný počet bodů (10) v případě, že je naprosto volně přístupný, zdarma a jde o open source. Nižší hodnocení pak dostávají za trial a demoverzi, kde časově omezený trial se všemi funkcemi je preferován z důvodu podstaty demoverzí, které mívají jen část funkcionalit k vyzkoušení a to se pro případ užití malé firmy nehodí, která nemůže koupit software za 1000 dolarů a pak zdlouhavě řešit storno požadavek v případě neuspokojení. Také je třeba brát v potaz, že většina začínajících firem nemá finance na jakýkoliv externí software.

4.1.2 Dokumentace a komunita

K vývoji neodmyslitelně patří i soupis všech funkcionalit, rozpis celého API a popis všech metod v knihovně. Tato pasivní pomoc je první s čím se potencionální programátor setká, druhou je pak komunita, která je stejně podstatnou součástí vývoje. Na komunitu je pohlíženo

jako na fóra okolo služby, jejich velikost, rychlost zavírání bugů a rad (issues) na GitHub nebo podobné verzovací kontrole, počet issues ze kterých se dá také čerpat, pokud dokumentace není dostatečně detailní nebo se vyskytne nějaká neočekávaná chyba. Dokumentace je hodnocena, jak dobře strukturovaná je, jestli je kompletní a nechybí tam části s nápisem "bude doplněno" a jak dobře se v ní orientuje. V dnešní době existuje spoustu nástrojů na generování dokumentace přímo z kódu, jako jsou v Jave například JavaDocs. Příkladem špatné dokumentace je například OpenIoT od stejnojmenné organizace dostupný na https://github.com/OpenIotOrg/openiot, kde se po hledání nikde nepodařilo najít návod k instalaci a nasazení do provozu.

Nejvyššího bodového ohodnocení lze dosáhnout dokumentací, která bude systematická, bude dobře zvládnutá z UX pohledu a bude tam popsán celý proces od instalace po první demoverzi chodu se zařízením. Komunitní část kritéria bere v potaz existenci komunitního fóra, počet issues na GitHub je dvousečný, ale spíše naznačuje aktivní komunitu a produkt o který je zájem než problémy s vývojem samotným, důležitým faktem je rychlost zpracovávání issues, znovu by mohl být OpenIoT použit jako negativní příklad, kde jsou některé z issues bez reakce od roku 2014. Dalším faktorem, na který bude brán negativní pohled je zvolení přístupu více repositářů (více podstránek na GitHub) pro jednotlivé jazyky či zařízení, tento přístup dává smysl v případě, že jeden velký repositář by nebyl vhodný z důvodu velkých změn v kódu, málo kódu společného a obecná nesouvislost repositářů, ale tento přístup má také za následek rozdělenou komunitu dotazů a tedy pomalejší reakce na issues. Tento problém se dá utlumit vytvoření jednoho centrálního repositáře, který bude na ostatní odkazovat a na dotazy kladené pouze na jednom místě se v případě jistých duplicit bude rychle odpovídat.

4.1.3 Vývoj

Vývoj je po dokumentaci a komunitě druhý nejdůležitější kritérium v hodnocení. Vývojem se bere všechna konfigurace, od počáteční instalace, přes připojování zařízení, správu uživatelů po samotný vývoj a jeho provedení. Podstatné je kolik programovacích jazyků je podporovaných, jaké jazyky to jsou - například knihovna nebo služba psaná v jazyce Go, bude mít jistě skvělou správu paměti a bude velmi rychlá, nicméně se počtem expertů, kteří jsou k dispozici na trhu a pravděpodobností, že firma již takového experta vlastní nemůže rovnat mnohem rozšířenějším jazykům, jako jsou Python, Java a nebo JavaScript. Jakožto

případ užití pro malou začínající firmu je preferované vlastní řešení před naprostou černou skříňkou, která je ovládána přes UI a tedy knihovny dostanou relativně větší bodové hodnocení než uzavřená řešení. Vlastní skriptovací jazyky mohou pomoci svojí jednoduchostí pokud k ním přistupuje laický uživatel, zároveň to znamená, že je potřeba v nich vyškolit zaměstnance a tak jsou tedy znovu preferována vlastní řešení.

Jak bylo představeno v kapitole 1.3.2 Virtualizace, od middleware služeb se očekává, dodání funkcionality jako knihovna rozšiřující vlastní projekt, v případě nutnosti rozjetí externího programu bude toto plně v kompetenci Docker kontejnerů, které se postarají o celou instalaci a tak nasazení na vlastní servery bude triviální. Akceptovatelnou variantou je také cloudové řešení pro které není potřeba nic lokálně hostovat. Problém s cloudovým řešením by mohl nastat v případě, že síť mezi zařízeními a middleware službou nemá přístup k internetu. V tomto případě je cloudová služba nemožná.

4.1.4 Správa uživatelů a zařízení

Podstatným kritériem je správa uživatelů přistupujících do middleware a správa zařízení samotných. Oba faktory by měly být implementovány ze strany middleware a uživatel by tak byl odstíněn od řešení bezpečnosti, abstrahování zařízení a komunikace. Od správy uživatel se očekává možnost přidělování práv, které by měla být vlastně definovaná, přidělování okruhů zařízení, jak uživatel nebo alespoň administrátor by měl být schopný změnit uživateli heslo, nebo ho obnovit. Z oblasti bezpečnosti by měla komunikace probíhat zašifrovaně pomocí HTTPS a uživatelská hesla by měla být uložena jako hash a ne prostý text.

Jak bylo zmíněno v kapitole 1.3.3. Bezpečnost, veškerá komunikace by měla být zašifrována a data by se neměly mezi zařízeními posílat pomocí prostého textu. Vhodné je rozpoznání zařízeních samotných, možnost zobrazení stavu zařízení, jeho baterie a obecné formátování stavu zařízení internetu věcí. Middleware pro účely této práce je izolován od sítě internetu věcí a na jednotlivá zařízení se připojuje napřímo, pro budoucnost by byla vhodné zavést Device discovery (Objevování nových zařízení) o kterém se zmiňuje kapitola 1.3.5 Správa zařízení. Vzhledem k architektuře middleware použitých v této práci - architektura řízená událostmi - by měli zařízení být schopné posílat data a co je podstatné, middleware služba tyto volání přijímat.

Hodnocení bude probíhat tedy na základě zmíněných hledisek, podstatné bude, jak je správa provedena, jestli v rámci UI, což je preferováno nebo přes volání API, které tímto může donutit vývojáře implementovat mnohem více funkcionalit a funkcí, než pro svou potřebu nutně potřebují. Možné rozšíření tohoto kritéria je možnost implementace dalších architektur, které jsou vhodné pro větší a rozsáhlejší sítě.

4.1.5 Podpora formátů

Jak bylo zmíněno v kapitole 1.3.4 Vývoj a distribuce dat všechny datové formáty, které nesou data přicházející ze zařízení internetu věcí by měly spadat pod strojově čitelné formáty. Toto kritérium hodnotí, s jakými formáty se dá pracovat v rámci konkrétní middleware služby. Také se v potaz berou možnosti, kterými mohou data proudit do middleware služby a podpora různých služeb třetích stran jako jsou další middleware, převážně orientované na zprávy (anglicky middleware Message broker - message oriented middleware) jako jsou již zmíněný RabbitMQ, Kafka, JMS které se starají o toleranci chyb v komunikaci a Kafka od Apache Foundation může sloužit jako middleware pro transformaci velkých dat sám o sobě.

Hodnocení bude na základě počtu podporovaných formátu, kde preferováno je podpora, která je již součástí knihovny nebo konkrétní služby, toto zajistí, že vývojář nemusí implementovat vlastní parsování, ale o vše je postaráno v rámci služby.

4.1.6 Databáze

Klíčovou částí pro agregaci dat a jejich analýzu je data ukládat po jejich přijetí a transformaci. Middleware, které jsou knihovnami samozřejmě umožňují připojit libovolnou databázi, preferováno je znovu řešení, kde je podpora některé databáze již součástí služby. Bodové hodnocení přidává možnost výběru mezi podporou SQL a NoSQL databáze, naopak negativní hodnocení dává naprostá nemožnost připojit jakoukoliv databázi s faktem, že je uživatel nucen používat neznámou databázi na pozadí. Toto řešení je vhodné pro rychlé prototypování aplikací, ale pro případ užití této práce, by to bylo z dlouhodobého hlediska na škodu, skutečnost, že se do databáze nedá připojit napřímo.

Protože většina middleware ze své podstavy podporuje odesílání dat pomocí HTTP volání, je možné i při in-memory databázích a chybějící podpoře jiné databáze možné připojit databázi

třetí strany za pomocí vyvinutí dalšího malého middleware, který by se staral pouze a jen o vystavování REST rozhraní pro konkrétní databázi.

4.1.7 Distribuce dat

Kritérium distribuce dat vyplývá z rešerše v kapitole 1.3.4 Vývoj a distribuce dat, ve kterém jsou zmíněny formáty a fungování middleware architektury pro tento projekt. Na rozdíl od kritéria podpora formátu, který řeší data, které přichází do systému middleware a s jakým formátem dat se pracuje vně systému, kritérium distribuce dat řeší možnosti, jakými lze data odesílat ze systému, jaké jsou podporované formáty a možnosti nastavení.

Lépe hodnoceno bude řešení, které je součástí systému, počet podporovaných formátů a další možnosti nastavení.

4.1.8 Vizualizace

Výsledná vizualizace dat není očekávanou součástí middleware služeb pro agregaci a analýzu dat a velké firmy pravděpodobně sáhnout po vlastním řešení využitím průmyslového standardu Kibana od společnosti Elastic. Menší firmy a tedy případ užití v rámci této práce ocení tuto možnost vizualizovat agregované informace, jak už pro finální zobrazení, tak se takováto možnost podporované vizualizace dá využít pro zjednodušení správy zařízení, které mohou tímto zobrazovat počet chyb, počet zařízení obecně, jejich stav, stav baterie, případně všechny obecné informace bez nutnosti implementace dalšího řešení třetích stran.

V tomto kritériu vizualizace bude kladen důraz na propracovanost a možnost konfigurace jednotlivých grafů, jestli celý dashboard funguje v reálném čase a preferované je znovu řešení, které je součástí a nevyžaduje startování další služby.

4.2 Vybrané Middleware

V této kapitole budou představeny zvolené middleware služby, jejich funkcionalita architektura a možnosti podle kritérií, které budou zhodnoceny v následující kapitole. Hlavním výběrovým kritériem pro výběr middleware služeb byla jejich dostupnost a to preferovanou variantou byl open source, možnost odzkoušení v rámci časově omezené trial verze nebo funkcionality omezená demoverze - program byl k dispozici na zkoušku bez

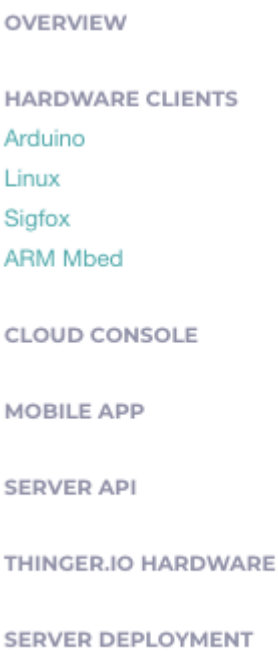
vkládání kreditních karet. Tímto se vyloučili možnosti jako AWS - Amazon Web Services se svým IoT Core, Microsoft Azure - IoT Hub a možnost vytvoření vlastní middleware služby za použití ELK souboru od Elasticu - ElasticSearch pro vyhledávání, jako databáze a pro analýzu, LogStash pro logování a sběr dat a Kibana pro vizualizaci. Tyto řešení jsou velmi robustní a není jich třeba pro případ užití malé firmy. Middleware služby byly také vybírány podle aktuálního vývoje, poslední update nemohl být starší než pár měsíců a zároveň být vyvíjen společností nebo skupinou nikoliv jednotlivcem, tento požadavek vyloučí malé nevyhledatelné middleware, kterého bude mnoho. Tyto malé projekty mohou přinést hodnotu, ale pro případ užití malé firmy, která chce pokračovat v dlouhodobém vývoji a tento produkt aktivně využívat je pravděpodobnost uzavření vývoje produktu, který je zastřešen jednotlivcem moc vysoká.

Bylo vybráno 5 middleware služeb, které jsou dostupné online. V praktické části již byly zmíněny některé ostatní jako Kaa a OpenIoT a proč nebyly vybrány. Pořadí, ve kterém jsou služby představeny je zcela náhodné.

4.2.1 Thinger.io

Thinger.io je jediný zástupce cloudového řešení, které je plně hostované, tato služba umožňuje i zavedení lokální stroje pomocí Dockeru. Thinger umožňuje škálovatelnou cloudovou infrastrukturu, kterou je možné jednoduše pomocí uživatelského rozhraní administrátorské konzole konfigurovat. Thinger se řadí mezi open source middleware služby, které nejsou vyvinuté konkrétně na žádný hardware od konkrétního výrobce a jako zařízení, které bude připojeno, je možné připojit i notebook. Celé cloudové řešení je dostupné za poplatek. Je možné službu vyzkoušet pomocí účtu zdarma, který je limitován na 2 zařízení, 4 dashboardy, 4 endpointy a 4 buckety, které fungují v Thinger.io jako databáze, která je součástí řešení a tedy pro ukládání dat jako jsou časové řady pro teplotu nebo vlhkost v čase. Tyto data mohou být zobrazeny pomocí dashboardu, uschovány nebo exportovány pro další offline analýzu. Kód je pod licencí MIT. Za rozšíření počtu zařízení, bucketů a endpointů se cena pohybuje od 3.95 EUR za měsíc pro 20 zařízení, 19.95 EUR měsíčně za 100 zařízení po startovací cenu pro korporátní a velké projekty od 200 EUR pro kterou je potřeba kontaktovat Thinger.io tým pro dohodnutí finální ceny podle potřeb klienta.

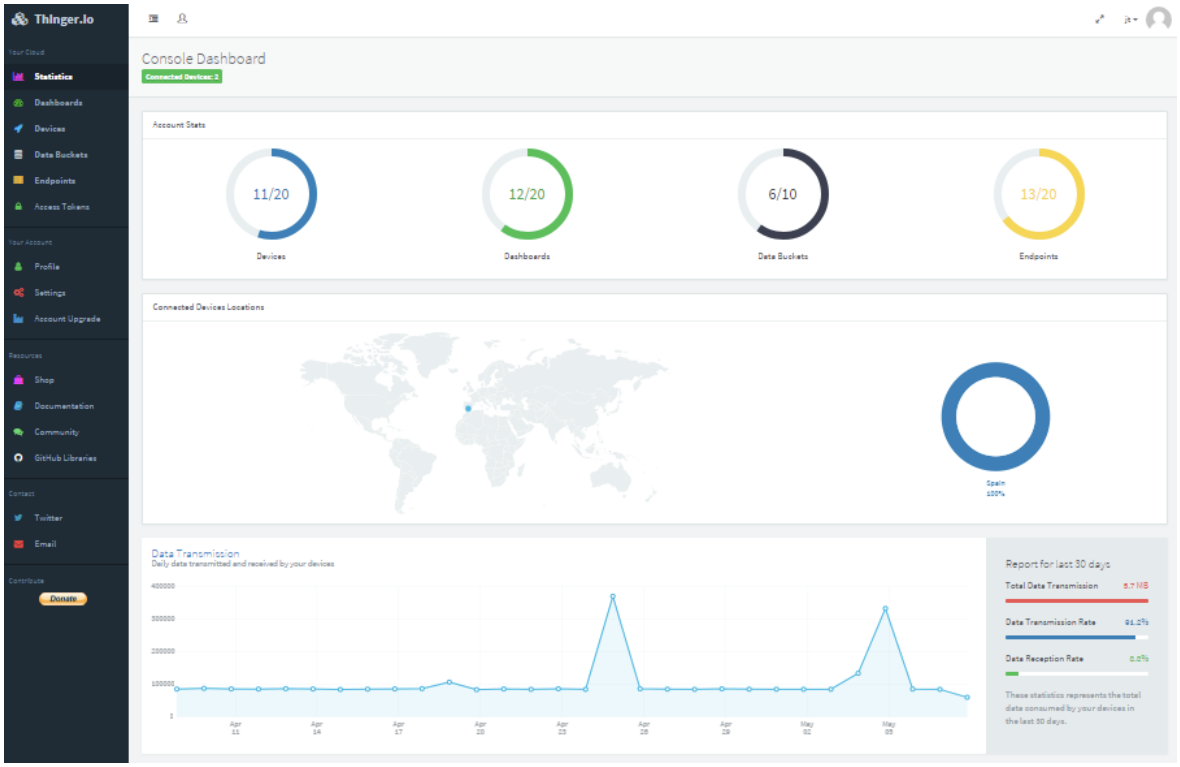
Dokumentace je detailní a obsahuje sekce pro všechny podporované zařízení, jak je vidět na obrázku níže (Obrázek 14), každá z těchto sekcí obsahuje detailní popis všech kroků a možností, které dané zařízení nebo platforma podporuje. Thinger.io má aktivní komunitu na svém fóru, kde odpovědi od ostatních členů jsou v řádu hodin. GitHub je v několika repozitářích a není jisté, který repozitář je hlavní, všechny mají obecně málo sledování a issue v řádech jednotek.



Obrázek 14 - Sekce Thinger.io dokumentace. Zdroj: [37]

Vývoj probíhá v rámci uživatelského rozhraní, přes které je možné vše nakonfigurovat. Služba využívá vestavěné databáze, kterou není možné měnit v rámci systému a tedy jedině nastavení externí se softwarem, který by tuto databázi spravoval. Připojování k zařízením umožňuje využití přístupových klíčů a je velice jednoduché. Jako zařízení může figurovat i stolní počítač a implementace a připojení není tedy nijak omezeno. Volání API pro konfiguraci je možné, ale v době psaní této práce je přes něj možné jen přidávat a odebírat uživatele a zařízení, zbytek funkcionalit v dokumentaci chybí. Služba podporuje jen formát JSON, který dokáže přijímat přes vystavené API endpointy REST rozhraní. Výsledná data analýzy lze znovu jen pomocí JSON formátu odeslat pomocí HTTP volání, tato funkce je v menu uživatelského rozhraní nazvána “Endpoints”.

Thinger.io podporuje základní vizualizaci, kterou je jednoduché nastavit, od kruhových grafů, přes mapy výskytu zařízení, příklad uživatelského rozhraní a vizualizace je vidět níže (Obrázek 15).



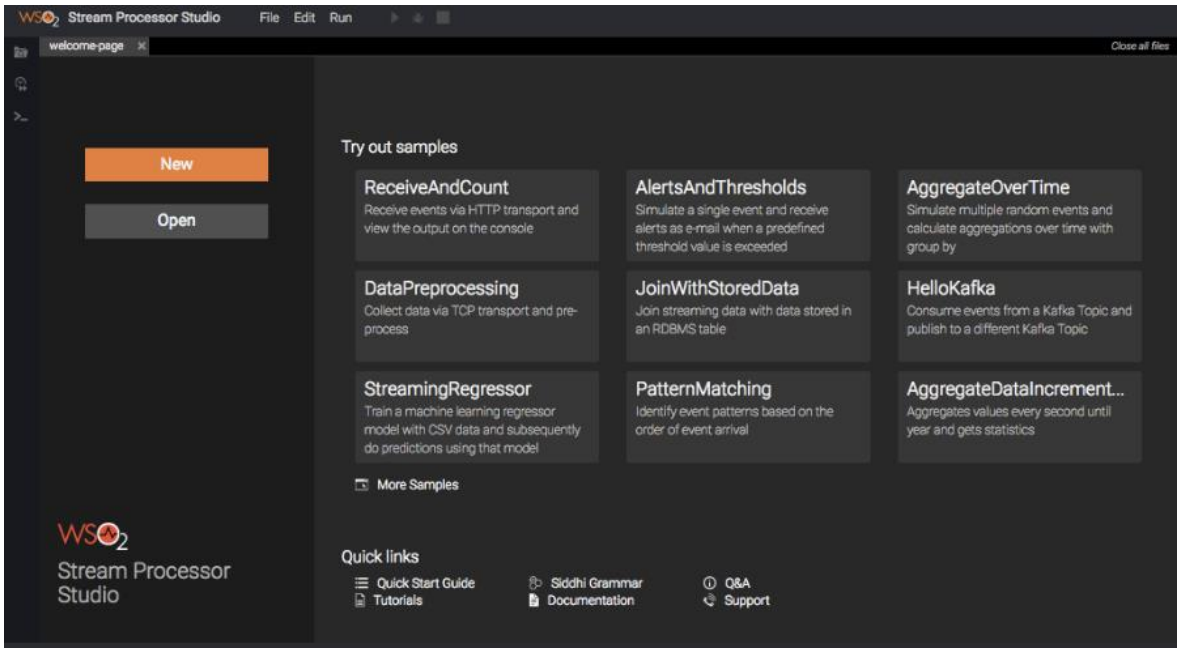
Obrázek 15 - UI Thinger.io cloudové platformy. Zdroj: [37]

4.2.2 WSo2 IoT

WSo2 je stream procesor od stejnojmenné společnosti, který se zaměřuje na enterprise analýzu dat pro velké společnosti. Služba je postavená na open source stream procesoru Siddhi, který je využíván společnostmi, jako je UBER a další. WSo2 uvádí, že je jejich stream procesor schopný zvládnout 100 000 událostí za vteřinu s malou velikostí nasazovacího balíčku. WSo2 je open source s trial verzí, která není dohledatelná, jak dlouho trvá. Instalace se dá provést mnoha způsoby přes AWS nasazení, Kubernetes po Docker, které ale nebyly možné vůbec v rámci trialu spustit. Instalace je možná v trialu pomocí .pkg souboru využívající Java rozhraní pro běh. WSo2 vyvíjí celé řešení pro měření výkonu, API management, správu uživatelů, IoT servery, datovou analýzu a zpracování datového streamu, které je použito pro účely této práce. Dokumentace je seskupená do nelogických kategorií, které jsou špatně naformátované. Každá kategorie v hlavním menu má po zvolení další

podkategorie, které mají další podkategorie. GitHub společnosti WSo2 má v době psaní této práce 280 repositářů, které tak fragmentují komunitu a jejich dotazy.

WSo2 funguje na bázi vývoje vlastních algoritmů, které jsou psány v online vývojovém editoru přímo v jednom z nainstalovaných kontejnerů. Editor má vlastní formátování a je přehledný, příklad UI jak vypadá ihned po startu prostředí je vidět níže (Obrázek 16).



Obrázek 16 - UI Thinger.io cloudové platformy. Zdroj: [37]

Pro vývoj se používá Siddhi Streaming SQL Language, který je vidět na příkladu níže (Obrázek 17). Jde o derivát C jazyka s kombinací s SQL dotazy přímo v kódu. Tento fakt znesnadňuje hledání expertů na trhu pro takto nestandardní jazyk.

```
@App:name("CargoWeightApp")

define stream CargoStream (weight int);

@sink(type='log', prefix='LOGGER')
define stream OutputStream(weight int, totalWeight long);

@info(name='CargoWeightQuery')
from CargoStream
select weight, sum(weight) as totalWeight
insert into OutputStream;
```

Obrázek 17 - Příklad kódu WSo2 IoT. Zdroj: [38]

WSo2 podporuje přijímání zpráv ve formátech JSON, XML, v binárních streamech, prostém textu a jako key-value páry. Tyto formáty jsou získány, jak z koncových zařízení internetu věcí protokoly TCP, HTTP či MQTT, ale také další škálou možností jako je Kafka, RabbitMQ, JMS a email. Stejný výčet možností a formátů je v rámci distribuce dat po provedené analýze. Pro svůj vývoj využívá buďto vestavěnou databázi nebo podporu připojení MongoDB, Solr, HBase a dalších. V rámci vývoje je všechna komunikace logována do zobrazené konzole.

Zařízení jsou připojovány pomocí zmíněných možností volání a služeb, ale jejich správa není nijak v rámci stream procesoru možná, stejná podpora je tak i pro správu uživatelů, kteří se musí v základní verzi připojit vždy jen pod jedním - administrátorským účtem. Pro rozšíření podpory správy uživatel a zařízení je nutné zahrnout do systému další služby od WSo2.
Služba podporuje základní typy grafů a mapu pro zobrazení geograficky vázaných dat. S grafy je možné vytvářet vlastní widgety pro vkládání na stránky a generovat reporty.

4.2.3 MainFlux

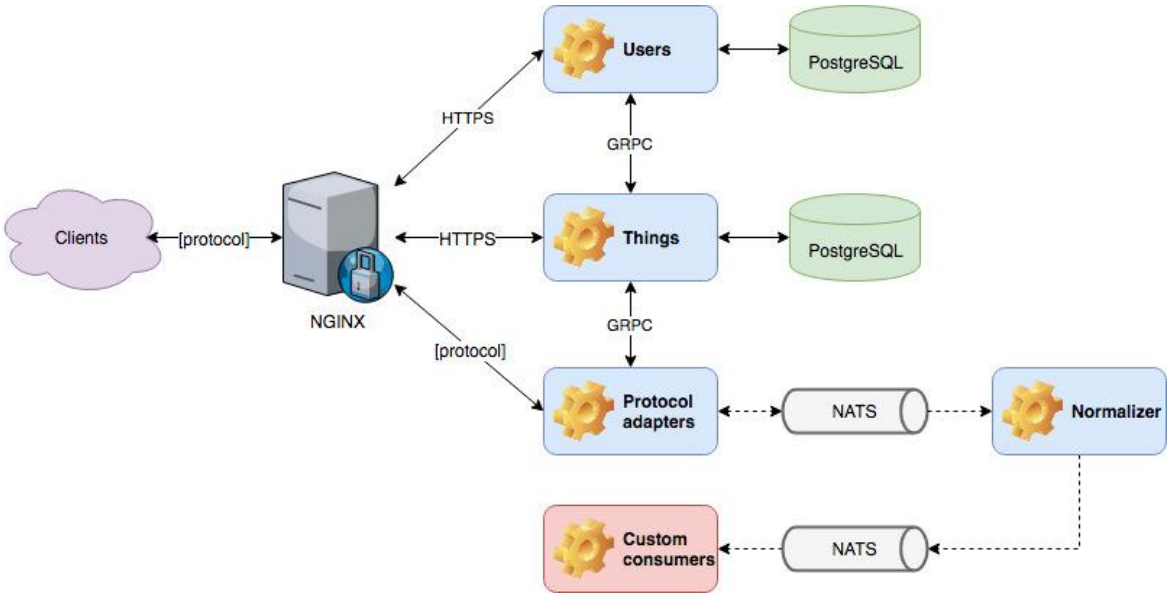
MainFlux je middleware služba a první zástupce s potřebou vývoje vlastního serveru kompletně bez uživatelského rozhraní. Jde o open source službu, která je zcela zdarma, vyvíjena stejnojmennou společností. Společnost službu vydává pod licencí Apache. Dokumentace je rozdělena na 12 položek, které mají další sekce. GitHub projektu je v jednom repozitáři s 50 otevřenými a 235 zavřenými issues. Repositář má v době psaní této práce 424 a poslední commit je 3 dny starý. Dokumentace používá všude příklady API volání napsané jako curl příkaz a není možnost zvolit příklady v jiných jazycích. Celá služba je napsána jazyce Go.

Vývoj, jak bylo nastíněno, je v rámci vytvoření celého vlastního řešení. Projekt je nainstalován pomocí Docker a funguje jako knihovna a API. Je možné využít integrovanou příkazovou řádku pro testování a základní nastavení. Potřeba vlastního serveru zároveň umožňuje vývoj jakéhokoliv řešení, včetně přijímaných formátů a možností distribuce dat dále. Možnosti pro komunikaci se službou jsou HTTP volání, WebSocket, MQTT a protokol CoAP. Nativně podporované databáze jsou InfluxDB, Cassandra, PostgreSQL a MongoDB, které jsou součástí Docker kontejneru. Na příkladu níže (Obrázek 18) vidíme curl příkaz pro vytvoření uživatele.

```
curl -s -S -i
 --cacert docker/ssl/certs/mainflux-server.crt
 --insecure -X POST -H
"Content-Type: application/json"
 https://localhost/users -d '{"email":"john.doe@email.com", "password":"123"}'
```

Obrázek 18 - Příklad kódu MainFlux. Zdroj: [39]

Hlavní architektura je rozdělena na tři hlavní celky - users, things a channel - uživatele, věci a kanál. User reprezentují reálné uživatele systému, kteří se přihlašují pomocí emailu a hesla, ty použije pro získání autorizačního tokenu. Po přihlášení, může uživatel spravovat zařízení a kanály ke kterým má oprávnění. Thing entita, která reprezentuje zařízení internetu věcí nebo aplikaci třetích stran, která využívá platformu pro komunikaci s dalšími zařízeními, které jsou připojené do platformy. Poslední entita Channel reprezentuje komunikační kanál, kterým všechny zařízení ve službě komunikují. Mainflux využívá open source zprávový systém NATS jako páteřní komunikační protokol. Systém NATS umožňuje komunikaci pomocí jakéhokoliv formátu. Celá architektura je na schématu níže (Obrázek 19).



Obrázek 19 - Architektura Mainflux systému. Zdroj: [39]

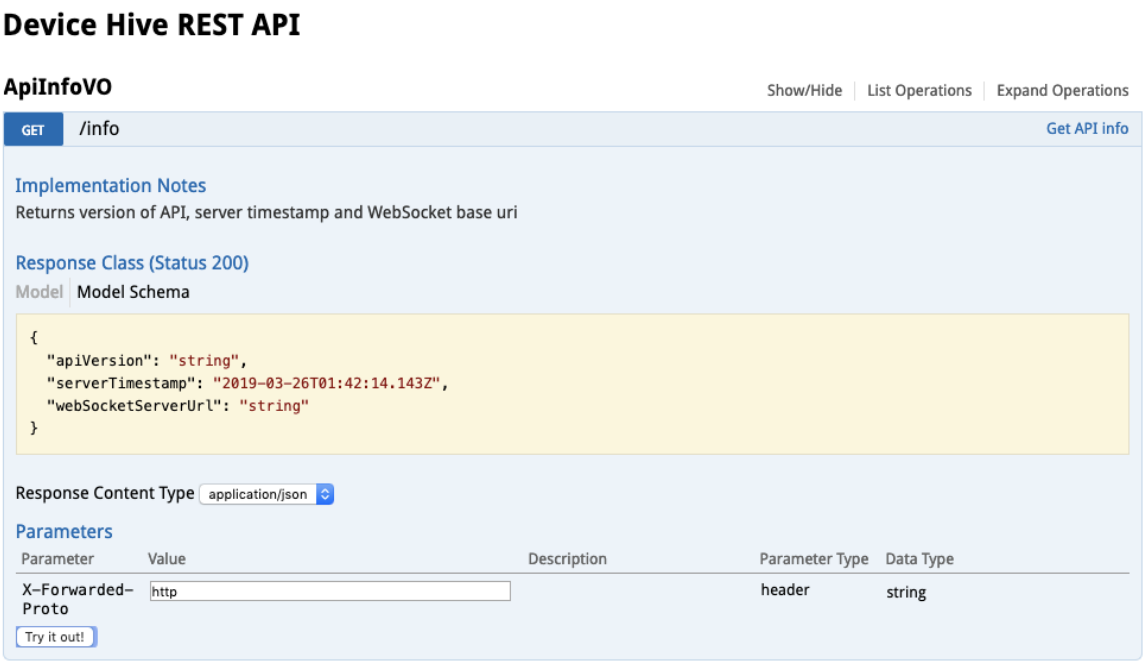
Mainflux využívá aplikaci třetí strany pro vizualizaci Grafana, která je součástí instalačního balíčku jako Docker kontejner.

4.2.4 DeviceHive

DeviceHive nabízí open source middleware službu, která jako předešlý MainFlux vyžaduje implementaci vlastního řešení. DeviceHive je platforma pro komunikace a správu chytrých zařízení. Služba je vyvíjena společností DataArt, která nabízí další produkty pro chytrou energii, automatizaci domova, vzdálené měření, telemetrii, vzdálenou kontrolu a monitorovací systémy. DeviceHive je cloudová platforma, kterou je možné nasadit i lokálně pomocí virtualizace Docker a jeho kontejnerů. Je zaměřená na využití paradigmatu microservice,

která zajišťuje nastavení, správu, kontrolu a analýzu chování chytrých zařízení internetu věcí pomocí různých protokolů oproti konfiguračnímu API, dalšími službami firmy DeviceHive je datová validace a sběr, strojové učení nebo umělá inteligence například pro analýzu video materiálu. DeviceHive nabízí vše bez poplatků pod licencí Apache 2.0. Službu si je možné vyzkoušet bez instalace s využitím online demoverze.

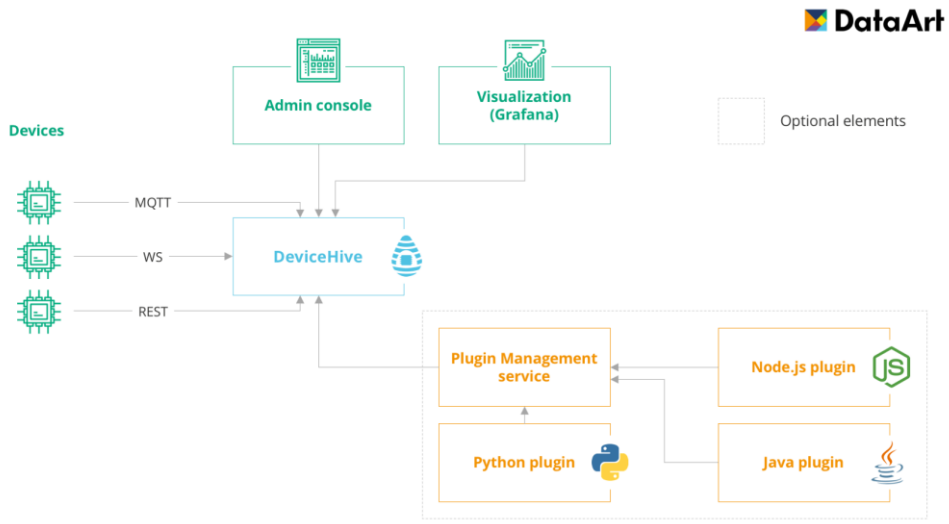
GitHub DeviceHive je rozdělený do více repositářů podle jazyka knihovny ze kterých je největší knihovna v jazyce Java s aktivním odpovídáním na issue ve kterých je 38 otevřených a 70 zavřených. Jako jediná služba má popsáno celé API včetně všech parametrů a odpovědí s využitím nástroje Swagger pro popis API (Obrázek 20). Dokumentace je rozdělena do 20 hlavních částí, které mají další sekce podle zařízení, typu entity nebo protokolu.



Obrázek 20 - API popsané v nástroji Swagger. Zdroj: [40]

Vývoj probíhá, jak bylo zmíněno implementací vlastního řešení a administrátorská konzola v rámci uživatelského rozhraní zde plní informativní charakter ačkoliv je v ní možné mnoho funkcí správy zařízení a zasílání informací konfigurovat. Uživatelské rozhraní je vázané pouze na formát JSON, tato limitace neplatí v případě implementace vlastního serveru. Knihovny jsou napsané v programovacích jazycích Python, Java, JavaScript a Go, ze kterých

je Java největší. Jak je patrné podle architektury (Obrázek 21) data lze přijímat pomocí zmíněného REST rozhraní, web servisy nebo protokolu MQTT. DeviceHive má podporu databáze PostgreSQL zabudovanou do jednoho z Docker kontejnerů. Data je možné odesílat a distribuovat dál pomocí HTTP volání, ale bez implementace vlastních algoritmů nelze provádět žádná analýza nad daty. Služba využívá, stejně jako MainFlux nástroje třetí strany Grafana.



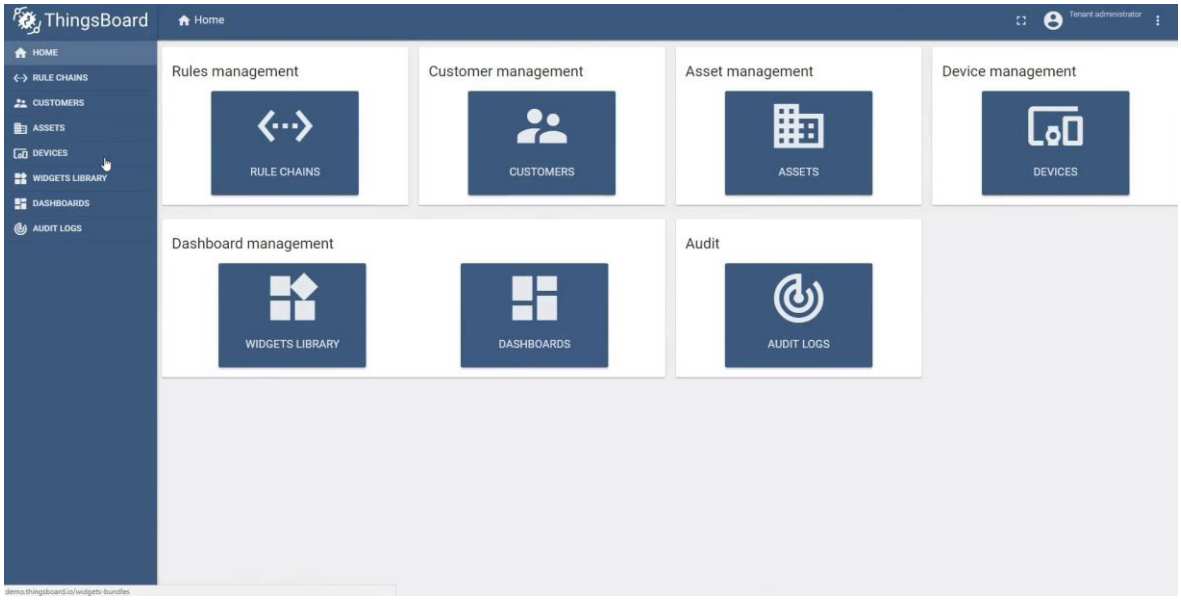
Obrázek 21 - Architektura DeviceHive systému. Zdroj: [40]

4.2.5 ThingsBoard

ThingsBoard je plně kompatibilní s oběma přístupy vývoje a to implementace vlastního řešení a UI. Služba je pod Apache licencí vyvíjena společností ThingsBoard jako open source. Vývojáři uvádí, že ThingsBoard cluster může obsluhovat milióny zařízení, je lehce konfigurovatelný díky svém programování přes UI a velmi škálovatelný. Dokumentace ThingsBoard je přehledná a velmi detailní, již první přehled popisuje rozdílné nasazení mezi monolitickým přístupem a přístupem microservices a rozdílný přístup mezi jednotlivou aplikací a nastavením celého clusteru. GitHub má nejvíce hvězd ze všech a vývoj je stále velmi aktivní, to vychází nejen z počtu pull requestů - 12, ale posledního commitu před týdnem a počtu issues (413 otevřených a 695 zavřených), aktivita této úrovně na GitHub přesahuje všechny ostatní služby dohromady. ThingsBoard používá pro podporu své

komunity i blog a online chat v nástroji Gitter. ThingsBoard nabízí placenou verzi pro enterprise řešení, které rozšiřují funkcionalitu.

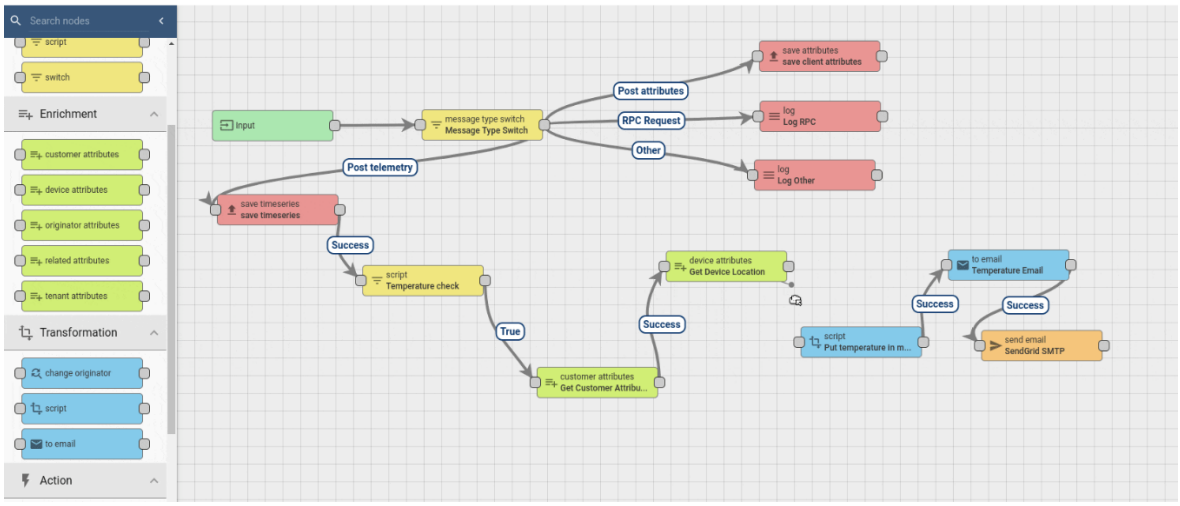
Aplikace je nasazena pomocí Dockeru a vše je možné nastavit přes administrátorské uživatelské rozhraní, které je vidět níže (Obrázek 22). ThingsBoard umožňuje funkcionalitu "Assets", které reprezentují skupiny, konkrétně budovy zákazníků a jejich zařízení. Všechna komunikace mezi zařízeními je šifrována. Vývoj v UI je vázán pouze na jeden formát - JSON, které je možné získat přes běžné protokoly HTTP, MQTT. ThingsBoard uvádí dvě podporované databáze a to Cassandra jako zástupce NoSQL řešení a PostgreSQL pro SQL databáze. Cassandra jako NoSQL je vhodná pro telemetrická data. Služba připravuje brzkou podpory další NoSQL databáze AWS DynamoDB. Pro testovací a development účely je možné použít databázi HSQLDB.



Obrázek 22 - UI ThingBoard. Zdroj: [41]

ThingsBoard umožňuje nastavení flow události pomocí Rule Engine frameworku. Ten se skládá, ze tří hlavních komponent, Message - zprávy - která reprezentuje příchozí událost, událostí může být nejen data z chytrého zařízení internetu věcí, ale také změna životního cyklu zařízení, volání vystaveného REST API, RPC požadavek a další nastavitelné vyvolání. Rule Node - funkce, která se provádí s každou příchozí událostí, které mohou provádět agregaci, transformaci, analýzu nebo filtraci dat nebo vyvolat další události a poslední komponenta je Rule Chain - spojení mezi jednotlivými Rule Node, které vyznačují a

kontrolují tok událostí systémem. Funkce v Rule Node, jsou psané v programovacím jazyce JavaScript. Příklad Rule Engine je níže (Obrázek 23).



Obrázek 23 - UI skriptování flow. Zdroj: [41]

Výsledná data jsou omezena formátově na JSON, která mohou být poslána dále na zpracování nebo uložena do databáze. ThingsBoard umožňuje vytváření grafů v reálném čase, vytváření widgetů a v placené verzi generování reportů.

4.3 Zhodnoceni Middleware

V této kapitole budou v návaznosti na předchozí, ve které byly vybrány middleware služby internetu věcí zhodnoceny jednotlivá řešení pro konkrétní případy užití. Hlavním případem je malá začínající firma s 10 zaměstnanci, která chce proniknout do odvětví internetu věcí a sběru dat z nich. Mají zájem tento projekt dál rozvíjet a mají kapacity vývojářů pro vyvinutí vlastního řešení. Middleware služby budou hodnoceny pomocí vícekriteriální analýzy variant, pro kterou budou nejprve určeny body na škále od 1 do 10, kde 10 je nejlepší hodnota. Následně podle případu užití budou zvoleny vhodné váhy pro hodnocení podle jednotlivých kritérií, které vychází jak z poznatků z lineární rešerše, zvoleného případu užití tak také s rad expertů věnujících se problematice internetu věcí na České Zemědělské Univerzitě a ve společnosti Openmatics věnující se internetu věcí pro správu vozových parků, kteří měli vhodné rady a návrhy z praxe pro hodnocení. Po hodnocení bude zvolena vhodná kompromisní varianta či popsány možnosti vývoje na základě výsledků pro jednotlivé případy užití. Škály hodnocení jsou definovány v jednotlivých kapitolách každého kritéria a hodnocení bude probíhat. Kritéria budou číslovány podle pořadí v první části praktické části

pro lepší přehlednost, která nenaznačuje jejich váhu ani preferenci. Zvolená čísla a jejich zkratky pro jednotlivá kritéria jsou v tabulce níže (Tabulka 2).

Tabulka 2 - Číslování kritérií a jejich zkratky. Zdroj: vlastní práce

č.	Kritéria	Zkratky
1	Licence	Licen
2	Dokumentace a komunita	Dokum
3	Vývoj	Vývoj
4	Správa uživatelů a zařízení	Správa
5	Podpora formátů	Formát
6	Databáze	Datab
7	Distribuce dat	Distri
8	Vizualizace	Vizu

4.3.1 Thinger.io

Thinger.io je open source middleware, ale jako jediný poskytuje pouze demoverzi omezenou na 2 zařízení, to oproti ostatním je nejhorší v této kategorii a proto dostává 1 bod. Dokumentace je rozdělena do sekcí podle zařízení což splňuje očekávání logických celků, GitHub je rozdělen do několika repositářů ze kterých není patrné, který repositář je hlavní, všechny mají velmi malou sledovanost a issues v řádech jednotek. Za systematickou knihovnu, fragmentovaný GitHub a komunitní fórum ve kterém je rychlá odezva (v hodinách) dostává 7 bodů.

Majoritní část vývoje probíhá přes uživatelské rozhraní, zároveň je možná konfigurace přes vystavený REST API, které v době psaní této práce je možné přidávat a odebírat uživatele a zařízení. Uživatelské rozhraní je funkční a vhodné na rychlé prototypování a umožňuje využít tokenů pro autentizaci přístupu, zařízení dostává 6 bodů. Ve správě uživatelů je možné připojit více uživatelů, přiřadit jim jednotlivá zařízení, ale není možné přiřazovat práva skupinou, zařízení je možné autentizovat pomocí tokenu přístupu, ovládat zasíláním JSON dávky a přijímat JSON data. Správa uživatelů a zařízení je průměrná - 5 bodů. Podpora formátů je na nízké úrovni s jediným podporovaným formátem JSON. Thinger.io používá

vestavěnou databázi a není možné připojit jinou bez vytvoření vlastního externího řešení, vestavěná databáze je vhodná pro prototypování, ale ne v dlouhém horizontu. Data je možné dál distribuovat pomocí HTTP volání v JSON formátu, které je dostačující, ale zároveň neumožnuje více variant. Služba podporuje jen základní vizualizaci pomocí základních grafů - kruhový, spojový a geografickou mapu. Všechna hodnocení jsou vidět níže (Tabulka 3)

Tabulka 3 - Hodnocení Thinger.io. Zdroj: vlastní práce

1	Thinger.io	Licen	Dokum	Vývoj	Správa	Formát	Datab	Distri	Vizu
		3	7	6	7	3	4	5	3

4.3.2 WSo2 IoT

WSo2 IoT je služba zaměřující na enterprise analýzu s mnoha zpoplatněnými moduly. Pro účel agregace dat je určen Stream Processor, který je - jako ostatní moduly - open source. WSo2 je možné nainstalovat pomocí knihovny Java zdarma na omezenou dobu, ostatní možnosti instalace nejsou v trial verzi možné. Trial verze je preferovaná nad demoverzí a fakt, že Java instalace je možná neomezeně dlouho na nekomerční využití je za kritérium licence 6 bodů. Dokumentace byla nejhorší ze všech hodnocených služeb po stránce systematičnosti, GitHub má 280 repositářů, které tedy drasticky fragmentují komunitu s jejich dotazy, bez jiné platformy na koncentraci - fórum, chat - dostává nízké 3 body.

Vývoj je v online vývojovém prostředí v Siddhi Streaming SQL Language, tato skutečnost neumožňuje žádné verzování kódu a bude nemožné hledat experty v této oblasti, kterých bude málo, a budou velmi drazí, proto dostává nejnižší ohodnocení - 3 body. Správa uživatelů není možná a správa zařízení také ne, zařízení by bylo možné autentizovat v rámci kódu větvením a posíláním tokenů v rámci dávek, ale toto není zamýšlená implementace ze strany vývojářů služby. Správa uživatelů a zařízení je možná po dokoupení modulů na toto určených, hodnocení je tedy nízké - 2. WSo2 nabízí absolutně největší škálu formátů a možností posílání, které jsou vypsány v předešlé kapitole popisující WSo2. Databáze je možné připojit externí, nebo využít jednu ze zabudovaných - in-memory databáze, MongoDB, Solr či HBase. Podpora vizualizace je na nižší úrovni jako Thinger.io. Všechny body za jednotlivá kritéria jsou v tabulce níže (Tabulka 4).

Tabulka 4 - Hodnocení WSo2 IoT. Zdroj: vlastní práce

2	WSo2 IoT	Licen	Dokum	Vývoj	Správa	Formát	Datab	Distri	Vizu
		6	3	3	2	10	4	7	3

4.3.3 Mainflux

Mainflux je plně zdarma online nástroj vycházející pod licencí Apache jako open source. Dokumentace služby Mainflux je napsaná v krocích, které nejsou dobře čitelné, příklady API volání jsou napsané v curl dávce a není možné zvolit jiný jazyk, GitHub je v jediném repositáři, který je aktivní jak po stránce vývoje (poslední commit před 3 dny) tak počet issue, kterých je 50 otevřených a 235 zavřených. Kombinace ne ideální dokumentace a dobrého GitHubu je 6 bodů.

Aplikace společně s DeviceHive vyžaduje vlastní řešení s tím rozdílem, že Mainflux nemá žádné UI. Mainflux funguje jako knihovna pro vlastní implementaci, díky tomuto je podpora formátů neomezená a jejich distribuce, záleží na konkrétní implementaci v projektu vývojáře, ale možnosti kritérií jsou vždy preferovány nativní proto 8 bodů pro distribuci a formáty. Nativně podporované databáze jsou InfluxDB, Cassandra, PostgreSQL a MongoDB, které jsou součástí Docker kontejneru. Uživatel může spravovat zařízení jen, na které má oprávnění, všechna komunikace je šifrovaná. Správa zařízení umožňuje nastavovat kanály pro

zařízení, autentizovat zařízení to je lepší průměr za 7 bodů. Mainflux pro vizualizace používá nástroje Grafana od třetí strany. Celé hodnocení služby je v tabulce níže (Tabulka 5).

Tabulka 5 - Hodnocení Mainflux. Zdroj: vlastní práce

3	Mainflux	Licen	Dokum	Vývoj	Správa	Formát	Datab	Distri	Vizu
		10	6	6	7	8	8	8	7

4.3.4 DeviceHive

Jak bylo zmíněno v předchozí kapitole, DeviceHive vyžaduje vytvoření vlastního řešení. DeviceHive je middleware služba distribuována pod licencí Apache 2.0 jako open source. Dokumentace služby je velmi kvalitní, systematická, rozdělena do 20 částí po logických celcích. GitHub není ve stejném repositáři, ale je validně rozdělen podle jazyka knihovna, ze kterých je největší Java s 38 otevřenými a 70 zavřenými issue. Kritérium dokumentace a komunity tak dostává plných 10 bodů. DeviceHive podporuje 4 velké programovací jazyky - Python, Java, JavaScript a Go - tento fakt upřednostňuje DeviceHive před Mainflux, která nepodporuje žádné klientské knihovny a celá služba je psaná v Go, což je méně rozšířený jazyk než JavaScript, Java či Python.

Data lze přijímat pomocí REST, MQTT nebo Web Service v jakémkoliv formátu, odeslání dat není z důvodu tvoření vlastní implementaci nijak omezeno, proto obě kritéria jsou ohodnocena 8 body. Správa zařízení a uživatelů je možná jak přes API volání, tak v UI. To umožňuje autentizaci zařízení, šifrování přenosu a volení práv pro jednotlivé zařízení pro každého uživatele. DeviceHive má zabudovanou podporu PostgreSQL, který je součástí instalace v jednom z kontejnerů. Služba využívá, stejně jako Mainflux nástroje třetí strany Grafana pro vizualizaci. Celé bodové ohodnocení je vidět v tabulce níže (Tabulka 6).

Tabulka 6 - Hodnocení DeviceHive. Zdroj: vlastní práce

4	DeviceHive	Licen	Dokum	Vývoj	Správa	Formát	Datab	Distri	Vizu
		10	10	8	8	8	7	8	7

4.3.5 ThingsBoard

ThingsBoard je open source distribuovaný pod Apache licencí. Dokumentace ThingsBoard je přehledná a velmi detailní, GitHub má nejvíce sledování ze všech a vývoj je stále velmi aktivní, to vychází nejen z počtu pull requestů - 12, ale posledního commitu před týdnem a počtu issues (413 otevřených a 695 zavřených), za to dostává hodnocení maximální - 10 bodů.

Vývoj je plně kompatibilní s oběma přístupy a to implementace vlastního řešení a UI konfigurace. Tyto standardní přístupy rozšiřuje o způsob skriptování a kontrolu datového toků pomocí UI. Tento framework se nazývá Rule Engine a ThingsBoard je ohodnocen 10 body. Správa zařízení a uživatelů tady splňuje všechny předpoklady middleware služeb a dodává funkcionalitu Nemovitostí (Assets), které dovolují seskupovat zařízení a uživatele například podle zákazníka. Přijímaný formát a práce ve službě, včetně Rule Engine je omezena pouze na JSON, distribuce dat je díky vlastnímu řešení neomezena a podporovanými protokoly jsou HTTP a MQTT. Na rozdíl tedy od vlastních řešení DeviceHive a Mainflux, která jsou neomezená a přináší ještě nativní podporu proto 9 bodů. Podporované databáze jsou cassandra, HSQLDB, PostgreSQL a připravovaná je DynamoDB. ThingsBoard umožňuje vytváření grafů v reálném čase, vytváření widgetů a v placené verzi generování reportů.

Tabulka 7 - Hodnocení ThingsBoard. Zdroj: vlastní práce

5	ThingsBoard	Licen	Dokum	Vývoj	Správa	Formát	Datab	Distri	Vizu
		10	10	10	10	3	10	9	9

4.3.6 Určení vah

Na základě literární rešerše byly určeny vhodné váhy pro jednotlivá kritéria pro konkrétní případ užití malé začínající firmy. Váhy byly diskutovány s experty z České Zemědělské Univerzity a společnosti Openmatics zabývající se zařízeními internetu věcí. Saatyho metoda pro zvolení vhodným vah nebyla použita, tato metoda je převážně vhodná, pokud váhy jsou určovány jednotlivcem což není tento případ. Jako nejvíce podstatnými kritérii se jevily kritéria Dokumentace a komunita a Vývoj, které jsou nejpodstatnější pro účel vývoje middleware služby s hodnotou 0,2. Další v pořadí kritéria Licence, Podpora formátů a Distribuce dat, těmto kritériím byla zvolena váha 0,15. Licence je klíčovým kritériem pro

menší firmy, které v začátcích nemají dostatečně financí, Podpora formátů a Distribuce dat je podstatná z hlediska variability možností správy homogenních dat. S hodnotou 0,1 kritérium Databáze a poslední s hodnotou 0,05 Vizualizace a Správa uživatelů a zařízení. Hodnoty zvolených vah jsou v tabulce níže (Tabulka 8).

Tabulka 8 - Váhy vícekriteriální analýzy variant. Zdroj: vlastní práce

Middleware	Kritéria							
	Licen	**Dokum**	**Vývoj**	**Správa**	**Formát**	**Datab**	**Distri**	**Vizu**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Normalizovaná hodnota váhy kritéria	0,15	0,20	0,20	0,05	0,15	0,10	0,15	0,05

4.3.7 Metoda váženého součtu

Pro metodu váženého součtu byly určeny v předchozí části body pro každé kritérium každé middleware služby. Všechny body pro metodu váženého součtu jsou v tabulce níže (Tabulka 9)

Tabulka 9 - Metoda váženého součtu. Zdroj: vlastní práce

Middleware		**Kritéria**							
		Licen	**Dokum**	**Vývoj**	**Správa**	**Formát**	**Datab**	**Distri**	**Vizu**
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	**Thinger.io**	3	7	6	7	3	4	5	3
2	**WSo2 IoT**	6	3	3	2	10	4	7	3
3	**Mainflux**	10	6	6	7	8	8	8	7
4	**DeviceHive**	10	10	8	8	8	7	8	7
5	**ThingsBoard**	10	10	10	10	3	10	9	9

Prvním krokem po určení bodového hodnocení je vybrání ideální a bazální varianty tedy minimum a maximum každého kritéria. Hodnoty ideální a bazální varianty jsou zapsány v tabulce níže (Tabulka 10).

Tabulka 10 - Metoda váženého součtu. Zdroj: vlastní práce

Varianty	Kritéria							
	Licen	Dokum	Vývoj	Správa	Formát	Datab	Distri	Vizu
	1	2	3	4	5	6	7	8
Ideální varianta	10	10	10	10	10	10	9	9
Bazální varianta	3	3	3	2	3	4	5	3

Ideální a bazální varianta je použita pro výpočet standardizované kriteriální matice dílčích užitků. Pro výpočet dílčího užitku Thinger.io pro kritérium Licence vezmeme jeho bodovou hodnotu 3, od ní odečteme bazální variantu sloupce - 3, to celé vydělíme výsledkem (ideální varianta - bazální varianta). Vyjde nám příklad (3 - 3) / (10 - 3) který se rovná 0 a tu zapíšeme do další tabulky. Tento postup opakujeme pro každou buňku, výsledek je v tabulce níže (Tabulka 11).

Tabulka 11 - Metoda váženého součtu. Zdroj: vlastní práce

Middleware		Kritéria							
		Licen	Dokum	Vývoj	Správa	Formát	Datab	Distri	Vizu
		1	2	3	4	5	6	7	8
Normalizovaná hodnota váhy kritéria		0,15	0,20	0,20	0,05	0,15	0,10	0,15	0,05
1	Thinger.io	0,00	0,57	0,43	0,63	0,00	0,00	0,00	0,00
2	WSo2 IoT	0,43	0,00	0,00	0,00	1,00	0,00	0,50	0,00
3	Mainflux	1,00	0,43	0,43	0,63	0,71	0,67	0,75	0,67
4	DeviceHive	1,00	1,00	0,71	0,75	0,71	0,50	0,75	0,67
5	ThingsBoard	1,00	1,00	1,00	1,00	0,00	1,00	1,00	1,00

Dalším krokem po naplnění tabulky váženého součtu je vypočítat celkový užitek varianty. Ten se vypočítá skalárním součinem hodnot v řádku s hodnotami zvoleným vah. Výsledkem je tabulka celkových užitků níže (Tabulka 12).

Tabulka 12 -Výsledek vícekriteriální analýzy variant. Zdroj: vlastní práce

Middleware		Celkový užitek variant	Pořadí
1	**Thinger.io**	0,231	5
2	**WSo2 IoT**	0,289	4
3	**Mainflux**	0,672	3
4	**DeviceHive**	0,833	2
5	**ThingsBoard**	0,900	1.

Na základě vícekriteriální analýzy variant metody váženého součtu, klíčových kritérií, zvolených vah pro případ užití malé začínající firmy vyšla služba ThingsBoard jako kompromisní varianta s nejvyšším užitkem s hodnotou 0,9. Jako druhý se umístila služba DeviceHive s 0,833 a dále (seřazeno podle pořadí) Mainflux, WSo2 IoT a Thinger.io.

5 Výsledky a diskuse

Z provedené vícekriteriální analýzy variant vychází jako nejlépe hodnocená kompromisní varianta middleware služba ThingsBoard. Pokud se bere v potaz případ užití malá začínající firma, je pro ni tato služba velice zajímavá. Firma v začátcích nemá finance na koupi robustního hotového řešení a pro tento middleware zdarma by měli možnost zutilizovat volné kapacity ze svých 10 zaměstnanců. Vhodnost ThingsBoard by se umocnila existencí uživatelského rozhraní, které je mnohem přístupnější netechnickým uživatelům a nabízí tak vhodné prezentační materiály pro potencionální klienty. Druhou navrhovanou možností by mohla být implementace vlastního řešení, které je vyvíjeno velkou nadnárodní společností s podporou knihoven ve 4 hlavních jazycích – Python, Java, JavaScript a Go. Obě varianty mají skvělou dokumentaci s detailním popisem všech klíčových aspektů, tento fakt podporuje lepší podpora na portálu GitHub, na kterém ThingsBoard aktivně reaguje. ThingsBoard by byl zvláště vhodný v případě nutnosti rychlého prototypování řešení jak pro demonstraci nápadu, testu proveditelnosti nebo v rámci ověření konceptu. Rychlé prototypování je umožněno grafickým skriptovacím frameworkem Rule Engine.

V případě doporučení řešení pro střední až velké korporátní společnosti je potřeba analyzovat rozdílnou situaci. Velké společnosti vždy preferují vyvinutí vlastního systému na míru jejich specifickým požadavkům, pro tento příklad by byla vhodná druhá kompromisní varianta DeviceHive. Díky pokrytí programovacích jazyků svými knihovnami - Python, Java, JavaScript, Go - pokryjí většinu běžného trhu při hledání expertů.

Ostatní varianty měly ve vícekriteriální analýze variant větší odstup než první dvě varianty mezi sebou, to bylo způsobeno převážné nedostatečným hodnocením ve stěžejních kritériích, kterými byly Dokumentace a komunita a Vývoj. Poslední dvě varianty WSo2 IoT a Thinger.io měli špatně zpracovnou vizualizaci a limitované možnosti připojení na externí databáze.
Porovnání výsledků vícekriteriální analýzy variant s jinými zdroji a články nelze, protože není možné nalézt skrze běžné vyhledávání podobného porovnávání. Online jsou k nalezení představení jednotlivých middleware služeb, ale nikoliv jejich porovnání. Na internetu bylo nalezení spousty zdrojů, které porovnávají middleware služby z hlediska jejich architektur a implementací těchto řešení.

6 Závěr

Cílem práce bylo zhodnocení middleware služeb sloužících k agregaci, transformaci a distribuci dat získaných ze zařízení internetu věcí.

V první části vlastní práce byly analyzovány klíčové požadavky na middleware služby vyplývající z literární rešerše. Každé zvolené kritérium bylo představeno, popsáno na základě jakých poznatků vyplývá a definován způsob a postup hodnocení. Byl přesně definován rozsah, do kterého middleware jako téma práce zasahuje a čeho se netýká a zároveň byl popsán případ užití malé začínající firmy, pro který jsou kritéria a jejich váhy volena. Dokumentace a způsob vývoje konkrétní middleware služby, byly společně nejvýznamnějšími kritérii jakožto klíčové parametry pro vývoj middleware řešení.

V další kapitole byly vybrány middleware služby podle základních kritérií: dostupné online, open source, je možné je vyzkoušet v rámci časově omezené trial verze nebo demoverze bez nutnosti vkládání kreditní karty. Middleware služby muselo být také aktivně podporované ze strany vývojářů a poslední update nemohl být starší než pár měsíců. Na základě těchto výchozích požadavků bylo vybráno 5 služeb - Thinger.io s vývojem v UI, WSo2 IoT s podporou mnoha formátů a UI vývojovým prostředím, Mainflux služba pro implementaci vlastního řešení, DeviceHive s detailní dokumentací a middleware ThingsBoard, který nabízel vlastní UI skriptování pomocí Rule Engine frameworku. Služby byly představeny a dále popsána, jejich funkcionalita, architektura a další charakteristiky podle zvolených kritérií.

V poslední části byly vybrané middleware služby zhodnoceny pomocí vícekriteriální analýzy a vyhodnoceny vhodné kompromisní varianty aplikované na jednotlivé případy užití. Pro případ užití malé začínající firmy se jako vhodné kompromisní varianty jevily ThingsBoard, který měl ve vícekriteriální analýze variant největší užitek a DeviceHive. ThingsBoard je vhodný pro rychlé prototypování díky UI skriptování v Rule Engine frameworku, na které jde navázat vlastní implementací pro tento robustní nástroj, nabízející správu zařízení, uživatelů a celých skupin v podobě funkce Asset management. DeviceHive je vhodný jak pro malou firmu, tak pro případ větších společností, které chtějí pracovat na vlastním řešení s podporou open source middleware služby, která je vyvíjena nadnárodní společností. DeviceHive má svou knihovnu ve více jazycích - Python, JavaScript, Java a Go. Na tyto výsledky a poznatky

je možné navázat reálnou implementací projektu, který bude sbírat data ze zařízení internetu věcí a pomohly by tak při vybírání vhodné middleware služby.

7 Seznam použitých zdrojů

1. R. H. Weber, (2010). "Internet of Things - New Security and Privacy Challenges". Computer Law & Security Review 26: 23-30.
2. KUMAR, S. Manor & MAHALAKSHMI, K. & XAVIER, P. A pervasive study on applications and technologies of internet of things (IOT). International Journal of Mechanical Engineering and technology (IJMET). 2019. ISSN: 0976-6359.
3. ALI, H. Ali & ALI, A. Hesham & BADAWY, M. Mahmoud. Internet of Things (IoT): Definitions, Challenges and Recent Research Directions. International Journal and Computer Applications. 2015. 128. ISSN: 975-8887.
4. ATZORI, Luigi & IERA, Antonio & MORABITO, Giacomo. The Internet of Things: A Survey. Computer networks. 2010. ISSN: 2787-2805.
5. EVANS, Dave. The Internet of Things: How the Next Evolution of the Internet Is Changing Everything. Kalifornie: Cisco Whitepapers. 2011. [online]. [cit. 15. 9. 2018]. <https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf>
6. LUETH, L. Knud. State of the IoT 2018: Number of IoT devices now at 7B - Market accelerating. [online]. [cit. 16. 9. 2018]. <https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/>
7. BHUVANESWARI, V. & PORKODI, R. The Internet of Things (IoT) Applications and Communication Enabling Technology Standards: An Overview. International Conference on Intelligent Computing Applications, ICICA. 2014. DOI: 324-329. 10.1109.
8. SALAZAR, Jordi & SILVESTRE, Santiago. Internet věcí. Praha: České vysoké učení technické v Praze Fakulta elektrotechnická. 2017. ISBN: 978-80-01-06231-9.
9. SIGFOX. miniUNI produkt [online]. [cit. 20. 9. 2018]. <https://partners.sigfox.com/products/miniuni>
10. COLITTI, Lorenzo & GUNDERSON, H. Steinar & KLINE, Erik & REFICE, Tiziana. Evaluating IPv6 Adoption in the Internet [online]. [cit. 5. 10. 2018]. <https://storage.googleapis.com/pub-tools-public-publication-data/pdf/36240.pdf>
11. HUSTON, Geoff. IPv4 Address Report [online]. [cit. 5. 10. 2018]. <http://www.potaroo.net/tools/ipv4/>

12. DEERING, S. & HINDED, R. Internet Protocol, Version 6 (IPv6) Specification [online]. [cit. 5. 10. 2018]. <https://tools.ietf.org/html/rfc2460>
13. MILLE, Lawrence. Internet of Things for Dummies. John Wiley and Sons, Ltd., 2017 ISBN: 978-1-119-34992-1.
14. HASSAN QUSAY F. Internet of Things A to Z: Technologies and Applications.Wiley-IEEE Press, 2018. ISBN: 978-1-119-45674-2.
15. BURIAN, P. Internet inteligentních aktivit. Praha: Grada, 2014. ISBN: 978-80-247-5137-5.
16. CARRARA, W. & ENZERINK, S. & FRÉDERIQUE, O. & COSMINA, R. & STEENBERGEN, van E. Open Data Goldbook for Data Managers and Data Holders. European data portal, 2018 [online]. [cit. 22. 10. 2018]. <https://www.europeandataportal.eu/sites/default/files/european_data_portal_-_open_data_goldbook.pdf>
17. SHARON, F & SHIRI, A. Metadata for Research Data: Current Practices and Trends. Dublin Core Metadata Initiative, 2018. ISSN 1939-1366.
18. CISCO. Cisco Global Cloud Index: Forecast and Methodology, 2016–2021. Kalifornie: Cisco Whitepapers. 2018. [online]. [cit. 23. 10. 2018]. <https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf>
19. Elgendy, N & Elragal, A. Big Data Analytics: A Literature Review Paper. Lecture Notes in Computer Science, 2014. ISSN: 8557. 214-227.
20. MONGODB, Hadoop and MongoDB. MongoDB, 2019 [online]. [cit. 24. 10. 2018]. <https://www.mongodb.com/hadoop-and-mongodb>
21. HE, Y. & LEE, R. & HUAI, Y. & SHAO, Z. & JAIN, N. & ZHANG, X. & XU, Z.: RCFile: A Fast and Space-efficient Data Placement Structure in MapReduce-based Warehouse Systems. IEEE International Conference on Data Engineering (ICDE), 2011. ISSN: 1199–1208.
22. YADAV, Lata S. & SOHAL, A. Review Paper on Big Data Analytics in Cloud Computing. Researchgate, 2017.
23. GILL, C. & SMART, D. W. Middleware for robots?. AAAI Spring Symposium on Intelligent Distributed and Embedded Systems. Stanford: 2002.

24. RAZZAQUE, Mohammad A. & MILOJEVIC, M. & PALADE, A. & CLARKE, S. (2015). Middleware for Internet of Things: a Survey. IEEE Internet of Things Journal, 2015. DOI: 10.1109/JIOT.2015.2498900.
25. SARANYA, M. & PRIYA, Anusha A. A Study on Middleware Technologies in Cloud Computing. International Journal for Innovative Research in Science & Technology, 2017. ISSN: 2349-6010
26. JONES, S. Docker: The Ultimate Beginners Guide To Learning: The Basics Of Docker. CreateSpace Independent Publishing Platform, 2016. ISBN: 9781537103008
27. FONG, J. ARE CONTAINERS REPLACING VIRTUAL MACHINES? Docker blog, 2018 [online]. [cit. 15. 2. 2019]. <https://blog.docker.com/2018/08/containers-replacing-virtual-machines/>
28. DOCKER. Welcome to the Docs - Docker. San Francisco: Docker Inc., 2016 [online]. [cit. 15. 2. 2019]. <https://docs.docker.com/engine/understanding-docker>
29. MCMILLAN, P. Attacking The Internet of Things (using time) DEFCON, 2014 [Online]. [cit. 15. 2. 2019]. <https://defcon.org/images/defcon-22/dc-22-presentations/Mcmillan/DEFCON-22-Paul-Mcmillan-Attacking-the-IOT-Using-timing-attacks.pdf>
30. RAWLINSON, K. HP Study Reveals 70 Percent of Internet of Things Devices Vulnerable to Attack. Kalifornie, 2014 [Online]. [cit. 15. 2. 2019]. <https://www8.hp.com/us/en/hp-news/press-release.html?id=1744676#.WEZYAtRU5_k>
31. DHANJAMI, N. Abusing the Internet of Things. Blackhat, 2014 [Online]. [cit. 15. 2. 2019]. <https://www.blackhat.com/docs/asia-14/materials/Dhanjani/Asia-14-Dhanjani-Abusing-The-Internet-Of-Things-Blackouts-Freakouts-And-Stakeouts.pdf>
32. ROMAN, R. & ZHOU, J. & LOPEZ, J. On the features and challenges of security and privacy in distributed internet of things. Computer Networks, 2013. DOI: 10.1016/j.comnet.2012.12.018.
33. MINERAUD, J., et al. A gap analysis of Internet-of-Things platforms. Computer Communications, 2016. DOI: 10.1016/j.comcom.2016.03.015.
34. HENDLER, J. A Primer on Machine Readability for Online Documents and Data. data.gov blog, 2012. [Online]. [cit. 22. 2. 2019]. <https://www.data.gov/developers/blog/primer-machine-readability-online-documents-and-data>

35. ALBUQUERQUE, C. & CAVALCANTI, A. & FERRAZ, S. F. & FURTADO, Paula, A. A Study on Middleware for IoT. Int'l Conf. Internet Computing and Internet of Things. CSREA Press, 2015. ISBN: 1-60132-439-1.
36. BANDYOPANDHYAY, S. & SENGUPTA, M. & MAITI, S. & DUTTA, S. Role Of Middleware For Internet Of Things: A Study. International Journal of Computer Science & Engineering Survey, DOI: 2011. 2. 10.5121/ijcses.2011.2307.
37. THINGER IO, The Open Source Platform for the Internet Of Things [online]. [cit. 2. 3. 2019]. <http://docs.thinger.io/>
38. WSO2 IOT. Stream Processor Documentation [Online]. [cit. 2. 3. 2019]. <https://docs.wso2.com/display/SP430/Stream+Processor+Documentation>
39. MAINFLUX. What is Mainflux? [Online]. [cit. 2. 3. 2019]. <https://mainflux.readthedocs.io/en/latest/>
40. DEVICEHIVE. Three Steps To IoT [Online]. <https://docs.devicehive.com/docs>
41. THINGSBOARD. ThingsBoard Documentation [Online]. [cit. 2. 3. 2019].