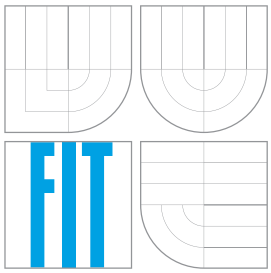
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

# AUTOMATIZOVANÉ ZPRACOVÁNÍ PROVOZNÍCH ZÁZNAMŮ V SYSTÉMU BEEEON

AUTOMATED PROCESSING OF LOG FILES IN BEEEON SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MAREK BEŇO

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PAVEL VAMPOLA

BRNO 2016

*Zadání bakalářské práce/18662/2015/xbenom01*

**Vysoké učení technické v Brně - Fakulta informačních technologií**

Ústav počítačových systémů Akademický rok 2015/2016

# Zadání bakalářské práce

Řešitel: **Beňo Marek**
Obor: Informační technologie
Téma: **Automatizované zpracování provozních záznamů v systému BeeeOn**
**Automated Processing of Log Files in BeeeOn System**
Kategorie: Analýza a testování softwaru

Pokyny:

1. Seznamte se se systémem BeeeOn vyvíjeným na FIT VUT, především s jeho serverovou částí.
2. Prostudujte dostupné nástroje pro flexibilní zpracování velkého množství provozních záznamů (logů).
3. Zvolte vhodný nástroj a navrhněte způsob jeho integrace do serverové části systému BeeeOn.
4. Proveďte integraci dle návrhu a zhodnoťte přínosy pro stabilitu a udržovatelnost systému.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 a 2 zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Vampola Pavel, Ing.**, UPSY FIT VUT
Konzultant: Puš Viktor, Ing., Ph.D., CESNET
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Zdeněk Kotásek, CSc.
*vedoucí ústavu*

# Abstrakt

Práca sa zaoberá spracovaním prevádzkových záznamov zo serverových aplikácií. Architektúra systému bola navrhnutá na základe štúdie dostupných technológií. Implementačná časť popisuje návrh jednotného formátu prevádzkových záznamov a implementáciu logovacej knižnice. Ďalej je popísaná inštalácia nástrojov, ich konfigurácia a nasadenie na server. Výsledkom je systém pre spracovanie prevádzkových záznamov navrhnutý s ohľadom na škálovateľnosť systému do budúcnosti. Systém bol otestovaný a nasadený v rámci fakultného projektu BeeeOn.



# Abstract

The paper concerns with processing of log files from server applications. System architecture is based on study of availible technologies. Firstly, design of unified log format and impelementation of unified logger library is described. Secondly, installation and configuration of used technologies and their integration is described. The result is log processing system designed to be scalable in the future. System was tested and integrated into project BeeeOn.



# Kľúčové slová

Internet vecí, BeeeOn, centralizované logovanie, prevádzkové záznamy, debugovanie, log súbory, spracovanie logov, elasticsearch, td-agent, kibana

# Keywords

Internet of Things, BeeeOn, centralized logging, logs, debugging, log files, log processing, elasticsearch, td-agent, kibana


# Citácia

BEŇO, Marek. *Automatizované zpracování provozních záznamů v systému BeeeOn*. Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Vampola Pavel.

# Automatizované zpracování provozních záznamů v systému BeeeOn

## Prehlásenie

Prehlasujem, že som túto prácu vypracoval samostatne pod vedením pána Ing. Pavla Vampolu a Ing. Viktora Puša Ph.D.. Uviedol som všetky literárne zdroje a publikácie, z ktorých som čerpal.

.....
Marek Beňo
17. mája 2016


## Poďakovanie

Ďakujem za vedenie a cenné rady počas vypracovávania práce vedúcemu Ing. Pavlovi Vampolovi. Ďakujem vedúcemu tímu server v BeeeOn Ing. Viktorovi Pušovi Ph.D. za konzultácie problematiky, ochotu a usmernenie počas vývoja.

# Obsah

# Kapitola 1

# Úvod

Práca sa zaoberá vytvorením systému spracovania prevádzkových záznamov v systéme Beeeon. BeeeOn je otvorený systém vyvíjaný komunitou na Fakulte informačních technologií VUT v Brně. Cieľom tohoto projektu je vytvoriť implementáciu inteligentnej domácnosti s orientáciou na otvorenosť riešenia a podporu zariadení tretích strán.

Prevádzkové záznamy predstavujú dátový výstup z aplikácií alebo procesov v systéme. Tento výstup predstavuje dátový tok, ktorý sa ukladá do takzvaného logovacieho súboru. Typický formát týchto súborov predstavuje súvislý tok jednotlivých záznamov udalostí. Záznamy sa ukladajú v súbore jedna udalosť na jeden riadok, kde udalosť sa skladá z formátovaných polí daných logovacím protokolom.

Cieľom systému spracovania prevádzkových záznamov je zjednotiť a optimalizovať spracovávanie týchto záznamov predovšetkým v serverovej časti systému BeeeOn. Centralizácia spracovávania záznamov uľahčuje prácu s týmito záznamami, čo má za následok jednoduchší náhľad do diania v interných častiach systému. Tieto dáta predstavujú vnútornú reprezentáciu stavu systému, možno ich použiť k vyhľadávaniu chýb v systéme alebo indikujú metriky systému. Všetky tieto aspekty vedú k dôležitosti spracovávania týchto dát a ich hodnote.

Pre spracovanie logov je určené centralizované riešenie podporujúce modulárny koncept. Koncept riešenia pozostáva z rozdelenia problému na jednotlivé podproblémy - oddeliteľné vrstvy nezávislé na ostatných, komunikujúce spolu prostredníctvom špecifikovaného formátu dát. Jednotlivé vrstvy reprezentujú tvorbu dát, ich zbieranie a jednotné úložisko a nástroje pre vizualizáciu a analýzu dát. Výsledné riešenie je následne navrhnuté na základe dostupných technológií a špecifikovaných požiadaviek, tak aby spĺňalo tento koncept.

Systém spracovania prevádzkových záznamov predstavuje praktickú integráciu centralizovaného logovania do systému BeeeOn, s ohľadom na určený koncept rozdelenia systému do vrstiev. Zaoberá sa využitím technológií podľa návrhu, ich implementáciou a konfiguráciou pre správnu funkčnosť. Riešenie predstavuje prínos pre užívateľov abstrakciou jednotlivých formátov dát, jednoduchosťou práce s nimi a prezentáciou formátovaných dát pre prehľad.

# Kapitola 2

# Centralizované logovanie

Centralizované logovanie je koncept správy prevádzkových záznamov (logov) [15], v ktorom je cieľom zjednotiť súčasti systému spracovávania prevádzkových záznamov do jednotnej architektúry. Architektúra spočíva z distribuovaných zdrojov dát v danom systéme, kolektorov zbierajúcich dáta, úložísk dát, sieťových prvkov zabezpečujúcich prenos dát a nástrojov pre analýzu dát. Typické problémy tohto systému predstavujú: problematika rozličných formátov dát, zabezpečenie koncových prvkov a prenášaných dát, škálovateľnosť architektúry.

Výhoda tohto riešenia spočíva v jednotnom úložisku dát, čo značne zjednoduší bezpečnosť a ochranu dát. Existujúce nástroje prevádzajú formátované správy do jednotnej internej reprezentácie, ktorá sa jednotne ukladá. Jednotný vnútorný formát zaručuje čitateľnosť dát pre užívateľa, ktorý nemusí poznať niekoľko špecifických formátov.

Tento prístup vyžaduje dodržanie niektorých zo základných konceptov, ktoré definujú návrhy potrebné dodržať pri tvorbe systému centralizovaného logovania.

**Zber a prenos dát** Hlavnou myšlienkou a hlavným prínosom tohoto konceptu je spracovávanie dát jednotne. Tento prístup nahrádza tvorbu záznamov a ich uskladnenie v rôznych formátoch, v rôznych médiách a komunikujúcich rôznymi protokolmi. Cieľom je zrušiť vzťah M:N pre pomer zdrojov dát a protokolov ich spracovania, zavedením jedného protokolu pre spracovanie dát, schopného spracovať viacero zdrojov a zaviesť jednotný formát v systéme.

Dôležitosť jednotného spracovávanie spočíva najmä v jeho škálovateľnosti. Pri systémoch skladajúcich sa z viacero aplikácií a viacero serverových zariadení sa vylučuje potreba spracovať rôzne aplikácie pre údržbu vzdialených záznamov. Dôležitým faktorom je aj dostupnosť záznamov, pri dynamických aplikáciach nie je vhodné používať replikáciu logovacích súborov pomocou časovaných úloh. V prípade centralizovaného riešenia je možno záznamy získať v takmer reálnom čase, oproti hodinovému, či dennému intervalu pri automatizovaných úlohách.

**Formát dát** Prevádzkové záznamy predstavujú tok, ktorý je možno priamo ukladať ako v prípade systémového žurnálu, do ktorého je možno pristupovať nástrojom `journalctl` alebo ich priame ukladanie do logovacích súborov.

V prípade uskladnenia prevádzkových záznamov v súboroch, dochádza k ich rotovaniu. Rotovanie súborov znamená zatvorenie aktuálneho súboru pre výstup a jeho zmena na základe času alebo jeho veľkosti. Tieto zmeny sa odzrkadľujú v názvoch súborov, ktoré obsahujú informáciu o dátume, v ktorom vznikli, čo umožňuje archiváciu alebo zmazanie

starých záznamov. Týmto spôsobom možno staré záznamy zmazať alebo ich možno archivovať, čo má za následok zmenšenie potrebného miesta na disku.

Typický formát súborov predstavuje System Log, ktorý samostatne predstavuje centralizáciu prevádzkových záznamov. Tento formát sa historicky osvedčil a v dnešnej dobe sa považuje za štandard, od ktorého sa odvíjajú ďalšie systémy spracovania.

**Proaktívne monitorovanie dát** Centralizácia formátu a ukladania dát znamená zjednodušenie ich spracovania a analýzy. V prípade chyby v aplikácii je možno tento stav detekovať a reagovať naň. Na základe dát možno vytvoriť pravidlá pre upozornenia užívateľov, čo znižuje čas potrebný k detekcií a náprave chyby.

V prípade zlyhania aplikácie nie je ale nutné čakať na prejav chyby pádom systému alebo iným následkom chyby, ale je možno detekovať anomálie v dátach a reagovať na ne. Práve proaktívny prístup je možný vďaka jednotnému formátu zavedenému centralizáciou a agregáciou týchto dát.

**Bezpečnosť a dôležitosť dát** Dôležitosť dát teda spočíva nielen v možnosti náhľadu do aktuálneho stavu systému, ale aj možnosti tieto dáta automaticky spracovávať a na ich základe vykonávať úlohy. Únik dát do nesprávnych rúk by mohol mať za následok zverejnenie podrobných informácií o systéme a zprístupnenie pripojeného systému útočníkovi.

Pre zabezpečenie dát je potrebné definovať spôsob prístupu k týmto údajom. Je potrebné zabezpečiť všetky rozhrania a užívateľské účty, ale aj prístupové body. V prípade infraštruktúry skladajúcej sa z viacerých serverov je potrebné definovať prístupové body, ktorými je možné prijímať dáta z týchto serverov. Pri prijatí dát je potrebné overiť identitu zdroja, aby nedošlo ku skompromitovaniu úložiska neplatnými dátami. Pri odosielaní dát vzniká potreba ich zabezpečiť, aby v prípade ich zachytenia nebolo možné ich čítať, a aby nebolo možné tieto dáta počas komunikácie pozmeniť.

Okrem zabezpečenia dát je dôležité vybrať typ a štruktúru dát, ktoré patria do záznamov. Úlohou záznamov je poskytnúť kľúčové informácie o jeho stave. Pre definovanie potrebných informácií a štruktúru záznamov je určený logovací protokol, ktorý špecifikuje formát záznamov.

# Kapitola 3

# Systém Beeeon

Systém BeeeOn sa zaoberá problematikou inteligentnej domácnosti a vývojom vlastných hardvérových a softvérových riešení [22]. Zabezpečuje tak nielen elektrotechnické prvky potrebné pre tento systém, ale aj otvorené zdrojové kódy [3] potrebné pre vybudovanie open-source platformy pre IoT (Internet of Things, Internet vecí) zariadenia.

## 3.1 Internet of Things

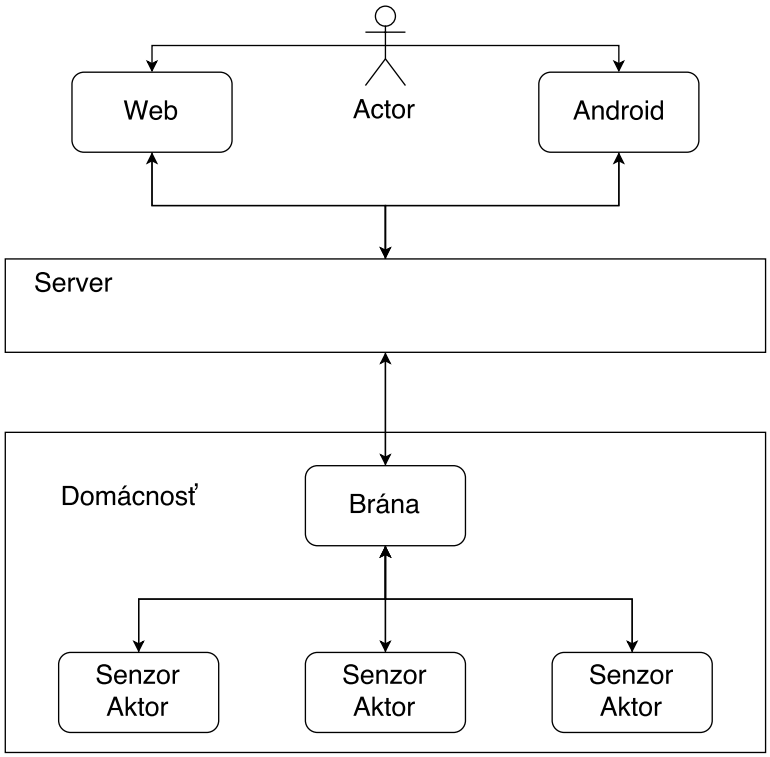
IoT predstavuje koncept siete, do ktorej sú zakomponované každodenné objekty, ktorým je pridaná istá miera inteligencie. Primárnym datovým zdrojom sú v tomto prípade senzory, ktoré majú za úlohu dodávať dáta do systému. Ovplyvnenie fyzického sveta zas ovplyvňujú reálny svet vykonávaním akcií [18]. Príkladom inteligentných zariadení teda sú rôzne domáce meteostanice, inteligentné televízory, prípadne chladničky. Rozšírenie týchto zariadení nachádza využitie v zdravotníctve, fitness, logistike alebo manufaktúre. Nárast v popularite a dostupnosti technológií pre vyššiu výdrž batérii a komunikáciu na dlhšie vzdialenosti, podporuje rozvoj týchto zariadení.

Špecifickosť tohoto systému spočíva v rôznych zariadeniach, typicky využívajúcich batérie pre napájanie, komunikujúcich vzdialene s malým obsahom dát. Práve dáta zohrávajú v systéme veľkú roľu, pretože sa využívajú na spracovanie systémom, pre rozhodovanie a informovanie užívateľa. Množstvo informácií v IoT a konektivita zariadenia sa však potýka s problémom so zabezpečením [16]. Bezpečnosť dát a zariadení užívateľa ostáva jedným z hlavných problémov tohoto konceptu.

## 3.2 Popis systému

Úlohou tohoto projektu je vytvoriť komplexný systém zabezpečujúci nielen hardvérové súčasti potrebné pre beh domácnosti, ale aj softvérové riešenie zabezpečujúce vnútornú logiku potrebnú pre prenos dát a riadenie domácnosti. Jedinečnosť systému spočíva v implementácií vlastných riešení kritických problémov, ako je vlastný komunikačný protokol a hardvérové zariadenia pre senzory a domácu bránu.

Výsledný koncept predstavuje inteligentnú domácnosť so všetkými súčasťami, potrebnými pre fungovanie celého systému a nevyžaduje dodatočné prvky tretích strán. Vďaka svojej jednoduchosti na inštaláciu je toto riešenie lákavé aj pre laickú verejnosť a zároveň vďaka otvorenosti zdrojových kódov a všetkých súčastí predstavuje priestor pre experimentáciu znalým užívateľom.




Obr. 3.1: Architektúra systému BeeeOn.

Obrázok 3.1 zobrazuje architektúru systému BeeeOn. Architektúra systému spočíva z hardvérových súčastí v domácnosti užívateľa: koncových prvkov predstavujúcich vstupy a výstupy systému a bránou predstavujúcou centrálny prvok zabezpečujúci komunikáciu so senzormi a prenos dát na server. Súčasťou systému sú serverové aplikácie predstavujúce vnútornú logiku, databáza určená ako úložisko užívateľských dát, automatizačný framework podporujúci konfiguráciu automatizovaných užívateľských úloh a užívateľské rozhrania zabezpečujúce prístup užívateľa do systému.

## 3.3 Koncové prvky

Systém BeeeOn pozostáva z dvoch typov koncových prvkov na užívateľskej strane: senzorov a aktorov. Senzory zabezpečujú základnú funkcionalitu systému meraním hodnôt a aktory slúžia na fyzickú interakciu s domácnosťou užívateľa.

Typická inteligentná domácnosť obsahuje niekoľko koncových prvkov, ktoré je možno ľubovoľne premiestňovať a prideľovať do daných miestností v domácnosti. Po spárovaní koncových prvkov s centrálnou bránou sa s ňou tieto koncové prvky prepoja a komunikujú s ňou namerané dáta alebo prijímajú príkazy na prepnutie stavu v prípade aktoru.

Systém BeeeOn zahŕňa vlastnú implementáciu senzorov a to ako hardvérových zariadení, tak aj ich softvéru. Softvérová implementácia zahŕňa vlastný komunikačný protokol „FIT Protokol“, ktorý zabezpečuje komunikáciu medzi bránou a integrovanými prvkami systému. Hardvérové zariadenia sa skladajú z bezdrôtových senzorov určených na meranie teploty a vlhkosti a senzoru tlaku vstavaného do centrálnej brány. Tieto zariadenia sú súčasťou zverejneného open-source projektu, avšak do systému je možno integrovať aj zariadenia

tretích strán ako napríklad Jablotron [7], Thermona [14] alebo openHAB [10].

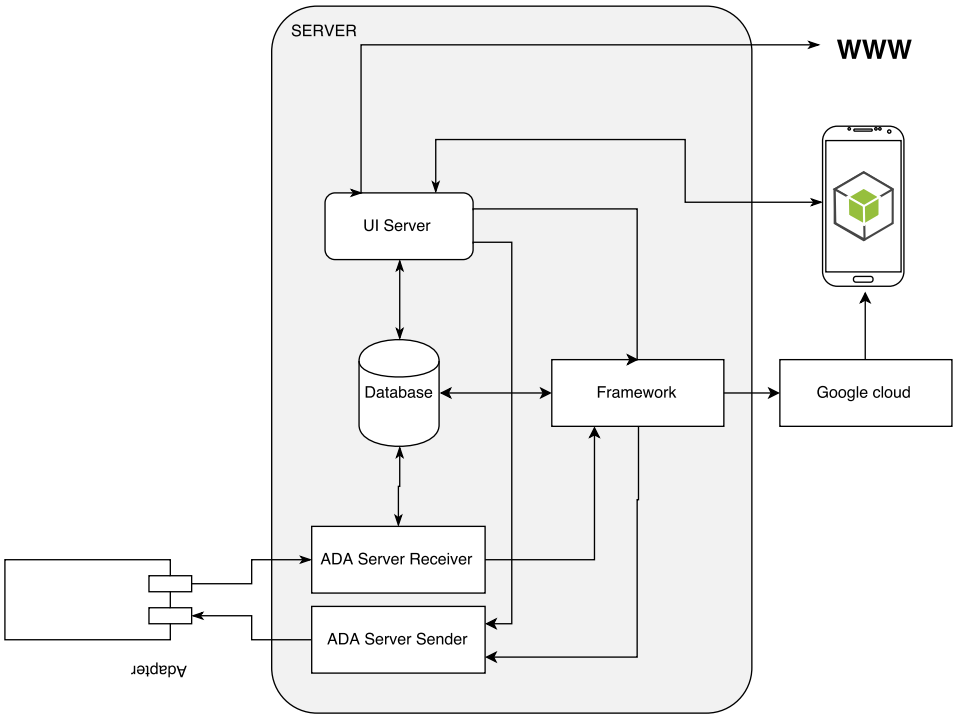
## 3.4 Brána

Brána predstavuje spojujúci článok medzi domácnosťou užívateľa a serverovou časťou zabezpečujúcou logiku pre riadenie domácnosti.

Brána v domácnosti slúži ako centrálny prvok, teda typická architektúra vyžaduje jednu bránu s pripojením na internet a do elektrickej siete. Hlavnou úlohou brány je zabezpečiť komunikáciu so serverovou časťou za účelom zasielania nazbieraných dát zo spárovaných koncových prvkov a prijímanie príkazov a nastavení z užívateľských rozhraní prostredníctvom serveru. V prípade výpadku spojenia je úlohou brány ukladať neodoslané dáta a v prípade obnovenia spojenia ich odoslať serveru. Bez priameho spojenia so serverom má domácnosť užívateľa veľmi obmedzené využitie a nie je možné využívať užívateľské rozhrania k ovládaniu domácnosti.

BeeeOn systém zahŕňa vlastnú implementáciu brány s hardvérovým riešením v jednotnom vizuálnom štýle podobnom šesťuholníkovému včeliemu plastu a softvérovým riešením pre ovládanie brány. Softvérové riešenie bežiace na bráne nazvané „adaapp“ zabezpečuje obojsmernú komunikáciu so senzormi aj bránou. Všetky komponenty brány, zahŕňajúce schému zapojenia dosky, 3D schémy obalu brány a softvér bežiaci na bráne, sú v rámci projektu verejné.osky, 3d schémy obalu brány a softvér bežiaci na bráne sú v rámci projektu verejné.

## 3.5 Server




Obr. 3.2: Serverová časť systému.

Serverová časť systému BeeeOn zabezpečuje vnútornú logiku a ukladanie dát z domácnosti. Súčasťou systému sú dva servery, fyzické zariadenia, na ktorých bežia serverové aplikácie. Vývojársky server ant-2.fit.vutbr.cz je určený na vývoj a ladenie systému pred jeho stabilizáciou a vydaním na produkčný server. Na produkčnom serveri cloud beží stabilná verzia systému a je určený na testovanie systému a integráciu inteligentných domácností.

Obrázok 3.2 zobrazuje serverovú časť systému BeeeOn a jej prepojenie s ostatnými súčasťami. Z obrázku možno pozorovať komplexnosť serverovej časti a teda dôležitosť ich rolí, ktorú zohrávajú v systéme. Najdôležitejšie prvky architektúry predstavujú ada_server zabezpečujúci spojenie s bránou v domácnosti a ui_server zabezpečujúci spojenie s užívateľskými rozhraniami. Tieto prvky tvoria jadro aplikačnej logiky serverovej časti a komunikujú s databázou určenou na ukladanie užívateľských dát. Ďalšou dôležitou súčasťou prepojenou so všetkými súčasťami je framework slúžiaci ako knižnica pre tvorbu automatizovaných úloh.

Serverové aplikácie predstavujú kľúčové role vo funkcionalite, keďže na ich behu závisí zvyšok systému. Kvôli širokej funkcionalite tieto časti produkujú množstvo prevádzkových záznamov, ktoré je vhodné sledovať a filtrovať za účelom sledovania ladiacich výstupov aplikácií. Tieto záznamy rozdelené do jednotlivých úrovní neobsahujú len informácie o zlyhaniach a vyvolaných výnimkách v týchto aplikáciach, ale aj stavové informácie o komunikácií a prenášaných dátach. Dôležitým typom správ sú záznamy „TRACE“ a „DEBUG“ využívané k lokalizácií chyby v prípade zlyhania aplikácie.

Množstvo záznamov uľahčuje lokalizáciu chyby, ale predstavuje kapacitné problémy a potrebu pre kategorizáciu záznamov a potrebu ich filtrovať. Robustnosť týchto systémov v praxi potvrdzuje nutnosť pokročilého systému spracovania prevádzkových záznamom so zameraním sa na zjednodušenie lokalizácie vád a možnosť monitorovania dát v systéme.

**Ada_server** Ada_server je serverová aplikácia, ktorej úlohou je vykonávať a spracovávať správy z domácnosti užívateľa prostredníctvom brány. Je ďalej rozdelený na dve súčasti: receiver a sender, ktoré sú priamo zodpovedné za prijímanie a odosielanie dát. Úlohou časti receiver je spracovávať dáta prijaté z brány a zasielať ich do databázy alebo frameworku. Úlohou časti sender je prijímať užívateľské požiadavky z automatizačného frameworku alebo ui_serveru a zasielať ich na bránu. K spracovaniu obslužných rutín sa využíva blok obslužných vlákien, z ktorého sa vyberá vlákno na spracovanie požiadaviek a kontajner spojení, z ktorého sa vyberá spojenie s konkrétnym zariadením.

Ada_server obsahuje vlastnú implementáciu aplikácie pre vytváranie logov, ktorá spracováva ladiace výstupy, prebiehajúcu komunikáciu na sieťových pripojeniach a databáze. Kvôli existencií bloku obslužných vlákien je špecifický protokol rozšírený o číslo vlákna, z ktorého správa pochádza za účelom ladenia výstupu viacvláknovej komunikácie.

**Ui_server** Ui_server zabezpečuje spojenie medzi užívateľskými rozhraniami: android aplikáciou a webovým rozhraním. Spracováva užívateľské dáta a odosiela ich na framework, alebo odosiela užívateľské požiadavky prostredníctvom ada_serveru na bránu. Dáta ukladá do databázy pre budúce použitie.

Ui_server obsahuje vlastnú implementáciu logovacej knižnice, ktorá okrem štandartných polí do správ pridáva aj číslo vlákna pôvodu záznamu a číselnú reprezentáciu vrstvy správy.

**Databáza** Databáza slúži na uloženie informácií o užívateľovi, jeho zariadeniach a dátach z nich. Namerané dáta na senzoroch sa zasielajú na ada_server, kde sa ukladajú do data-

bázy. Následne sa dáta môžu čítať z databázy pre účely frameworku, ada_serveru alebo ui_serveru.

**BeeeOn Automation Framework** Framework predstavuje systém pre definíciu automatizovaných úloh. Pozostáva z riadiacich a komunikačných súčastí a samotných automatizačných úloh. Jeho úlohou je poskytovať podporu pre integráciu automatizovaných úloh do systému zabezpečujúcich kontrolu a sledovanie domácnosti podľa prednastavených pravidiel.

Framework nemá vlastnú logovaciu knižnicu a predstavuje serverovú časť vo vývoji, preto je vhodné v tejto časti integrovať univerzálnu logovaciu knižnicu. Formát správ vyžaduje okrem identifikácie jednotlivých vlákien, podobne ako v iných častiach serveru aj identifikáciu zdroja správy – automatizačnej úlohy.

## 3.6 Užívateľské rozhrania

Užívateľské rozhrania systému predstavujú Android aplikáciu [2] a webové rozhranie. Obidve rozhrania podliehajú jednotnému vizuálnemu štýlu a podporujú väčšinu zdieľanej funkcionality pre komfort užívateľa. Pre prihlásenie do systému užívateľské rozhrania vyžadujú interný BeeeOn účet.

Primárnymi funkciami sú registrácia domácej brány, párovanie senzorov s bránou a manažment domáceho systému. Hlavnou úlohou užívateľských rozhraní je poskytnúť užívateľovi prehľad o systéme a aktuálnych a historických dátach. Okrem tejto funkcionality poskytujú možnosti správy užívateľov, pridávanie a odoberanie zariadení.

# Kapitola 4

# Dostupné nástroje

Pre návrh vhodného riešenia je potrebná štúdia dostupných nástrojov na základe požiadaviek na systém so zameraním na kompatibilitu zvolených riešení. Kvôli rozdeleniu systému na tri vrstvy je potrebné brať ohľad na komunikáciu medzi vrstvami a teda formátom dát. Dostupným riešením je implementácia strednej vrstvy formou kolektoru, ktorý prevádza dáta na vnútornú reprezentáciu. V tomto prípade je možné podporovať niekoľko rôznych formátov vstupu a zároveň dodržiavať jednotnú internú reprezentáciu dát v centralizovanom úložisku.

Množstvo nástrojov však pracuje na príliš nízkej úrovni, kde je potreba zapuzdrenia alebo na príliš vysokej úrovni, kde je výsledkom komplikovaný a robustný systém nevhodný pre menšie projekty. Cieľom štúdie dostupných nástrojov je nájsť kompromis medzi týmito dvoma extrémami.

## 4.1 ELK stack

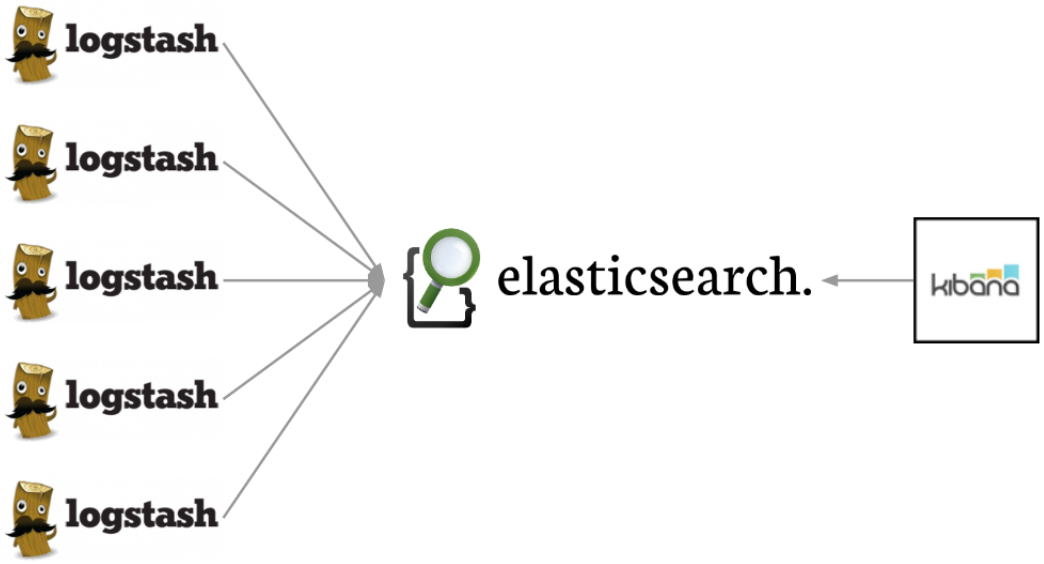
ELK (Elasticsearch Logstash Kibana) je open-source systém pre spracovanie a vizualizáciu prevádzkových záznamov. Poskytuje jednotné riešenie pre všetky potrebné vrstvy implementujúce spracovanie dát aj ich uloženie. Prevádzkové záznamy sa spracovávajú podľa konfiguračného súboru a ukladajú sa v internej reprezentácií. Rozšírenia tohto systému sú ale komerčné produkty, predstavujúce zásuvné module pre zabezpečenie, monitorovanie a notifikačný systém.

Architektúra pozostáva z úložiska dát Logstash, nástroju pre pokročilé vyhľadávanie Elasticsearch, webového rozhrania Kibana a komerčných zásuvných modulov do tohoto rozhrania.

Toto riešenie je populárne najmä vo veľkých spoločnostiach ako NASA, Github, The guardian, Klout, SOUNDCLOUD, kvôli svojej škálovateľnosti a pokročilým vyhľadávacím funkciám produktu Elasticsearch.

**Filebeat** Filebeat [5] je nástroj určený na sprostredkovanie - preposielanie dát v architektúre ELK. Je následníkom nástroja Logstash Forwarder, ktorý bol implementáciou Lumberjack protokolu pre zasielanie dát. Úlohou nástroja Filebeat je sledovať logovacie súbory a zasielať ich dáta do Logstash pre ďalšie spracovanie.

**Logstash** Logstash [9] slúži ako centralizované úložisko dát rôznych formátov a rôznych zdrojov.




Obr. 4.1: ELK architektúra prevzato z [4].

Hlavnou výhodou tohto nástroja je možnosť vytvárať spracované dáta rôznych formátov, pre ktoré je potreba konfigurácie. Konfiguračný súbor definuje štruktúru spracovávaných správ a predstavuje formát pre vnútornú reprezentáciu. Spracovanie rôznych zdrojov prebieha využitým konfiguračného súboru pre formátovanie vstupu a následné uloženie dáta určené na indexáciu a analýzu pomocou produktu Elasticsearch.

Podporované vstupy nezahŕňajú len prevádzkové záznamy aplikácií, ale možnosť ukladať a spracovávať aj monitorovacie, systémové dáta alebo dáta zo sociálnych sietí.

**Elasticsearch** Elasticsearch [4] je nástroj pre spracovávanie, vyhľadávanie a analýzu dát založený na knižnici Apache Lucene. Primárne je zameraný na dynamické spracovanie veľkého množstva dát s podporou pokročilého plnotextového vyhľadávania.

Vyznačuje sa svojou rýchlosťou v spracovaní dát, možnosťou distribúcie medzi viacero serverov a jednoduchej rozšíriteľnosti a škálovateľnosti.

Pre prácu s dátami je potrebné vytvoriť a nakonfigurovať službu Elasticsearch a nakonfigurovať úložisko dát – Logstash. Konfigurácia nových formátov dát je pomocou konfiguračného súboru v JSON formáte, čo umožňuje pridať nové zdroje dát, vytvoriť konfiguračný súbor, naindexovať ich a dáta sú pripravené na použitie.

Prístup k dátam je možno vytvoriť cez webové rozhranie Kibana, prípadne klientskými aplikáciami s podporou rôznych jazykov. Výhodou pre použitie tohto nástroja je podpora REST API, ktorá umožňuje operáciu s dátami pomocou JSON dotazov.

**Kibana** Kibana [8] je webové rozhranie zamerané na vizualizáciu a reprezentáciu dát.

Kibana umožňuje vykreslenie dát formou tabuliek, grafov, histogramov a máp. Pre tieto potreby primárne využíva indexované dáta z Logstash analyzované službou Elasticsearch. Vizualizácia dát je však možná aj z úložisk Hadoop, Beats ale aj z technológií tretích strán ako napríklad Apache Flume a Fluentd. Okrem prevádzkových záznamov možno analyzovať

aj dáta zo sociálnych sietí ako napríklad twitter alebo potvrdenia repozitárov hostovaných na Github.

Webové rozhranie poskytuje silný nástroj nielen ako priestor pre vizualizáciu dát, ale aj analýzu a ich jednoduchý export. Pomocou webového rozhrania možno pristupovať k dátam a manuálne vytvárať grafy, mapy a iné vizualizačné prvky a zároveň analyzovať dáta pomocou Elasticsearch.

Výsledkom je takzvaná informačná nástenka (dashboard), ktorá slúži na zobrazenie rozličných panelov a teda umožňuje celkový náhľad na výsledky dotazov spojené s grafickým prevedením jednoduchým pre pochopenie a ilustráciu dát. Tieto nástenky možno exportovať, zdielať alebo vkladať na ďalšie stránky.


Obr. 4.2: Kibana informačná nástenka prevzato z [8].

brázok 4.2 ukazuje informačnú nástenku s grafmi vizualizujúcimi dáta s využitím Elasticsearch pre operácie s dátami.

Okrem vizualizácie a analýzy dát predstavuje webové rozhranie Kibana s integrovanými zásuvnými modulmi platformu pre správu užívateľských rolí, kontrolu zabezpečenia, monitoring a notifikačný systém na základe dát.

**Shield** Shield predstavuje subsystém ochrany dát a umožňuje vytvoriť bezpečnostnú vrstvu kontrolujúcu prístup k dátam.

Umožňuje zabezpečenie transportnej vrstvy a API vytvorením bezpečných prístupov nastavením certifikátom a zabezpečením pripojenia. Vo webovom rozhraní integruje systém správy rolí, ochranu dát prístupom na meno a heslo a poskytuje možnosti vytvorenia pravidiel na filtrovanie IP adries užívateľov. Systém rolí a zabezpečenie prístupu umožňujú vytvoriť pravidlá pre skupiny užívateľov zamedzujúce práva na čítanie/zápis dát a vytvorenie záznamu neoprávnených prístupov umožňujúci detekovať útoky.

**Watcher** Watcher je notifikačný systém umožňujúci upozornenia a monitorovanie ELK architektúry alebo dát. Systém je možné využiť na kontrolu integrity dát zo zdrojových zariadení, výskytu chýb na týchto zariadeniach alebo výskyt špecifických indikátorov v

záznamoch zo zariadení. Jedným z ďalších využití je monitorovanie využitia dostupných zdrojov akými sú dôležité faktory využitia CPU, RAM alebo diskového priestoru.

Jednoduchý prípad použitia predstavuje vytvorenie jednoduchého Elasticsearch dotazu, nastavenie podmienok dotazu a typu notifikácie. Jedným zo základných typov notifikácie je upozornenie emailom na výskyt chyby v prevádzkových záznamoch.

**Marvel** Nástroj pre monitorovanie stavu a živosti ELK bloku formou grafov vizualizovaných vo webovom rozhraní s podporou niekoľkých blokov súčasne. Poskytuje informácie o metrikách popisujúcich aktivitu a efektivitu bloku s možnosťami špecifických grafov pre jednotlivé dáta a náhľadom na stav bloku. Marvel tiež poskytuje možnosť pre zobrazenie historických dát.

**Found** Nástroj Elasticsearch ponúkaný ako služba hostovaná na serveroch spoločnosti elastic, s vytvorenými blokmi a uzlami pripravenými na použitie.

**Zhrnutie** Celý systém sa vyznačuje ako veľmi vhodné a expertné riešenie zadanej problematiky, ale svojou robustnosťou vykazuje závažné nedostatky. Inštalácia samotného systému je na jeho komplexnosť pomerne jednoduchá, problém nastáva v riešení konfliktov s inými bežiacimi službami. Konfiguráciou v/v rozhraní je možné vyriešiť problém bežiacich služieb na rovnakých portoch, avšak minimálne nároky na zdroje vyžadované pre beh zostávajú závažným problémom tohto nástroja.

Výhodami tohto riešenia zostávajú webové rozhranie poskytujúce pokročilé možnosti vizualizácie logov, škálovateľnosť a možnosť práce s veľkými množstvami dát distribuovaných na rozličných zdrojoch. Po konfigurácií sa vyznačuje najmä pokročilými možnosťami vyhľadávania a formátovania logovacích záznamov.

## 4.2 Fluentd

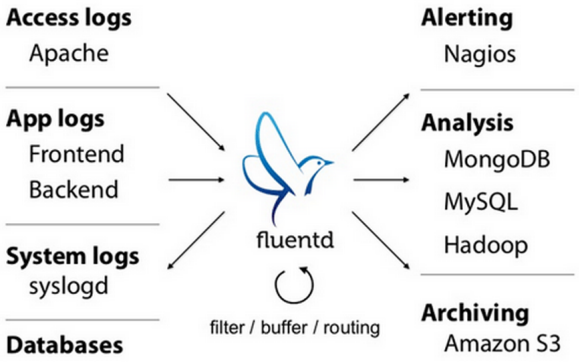
Fluentd [6] je open-source nástroj slúžiaci k zbieraniu dát v jednotnej logovacej vrstve. Špecializuje sa na vytvorenie jednotnej logovacej vrstvy zjednotením vstupov a ich prevedenie na vnútornú reprezentáciu, ktorú je možno jednotne uložiť alebo zaslať na spracovanie.

Hlavnou funkcionalitou je rozdelenie jednotlivých vstupov a spracovanie smerovania správ. Je možné spracovávať nielen rôzne typy vstupov ako je čítanie dát zo súboru alebo systémového žurnálu, ale aj vytvoriť komplexnú architektúru pozostávajúcu z inštancií Fluentd na vytvorenie centrálneho úložiska. Rozlíšenie jednotlivých vstupov zaručuje modulárny prístup s podporou zásuvných modulov špecifikujúcich formát vstupu alebo výstupu. Rozsiahla komunita poskytuje riešenia a podporu pre množstvo zdrojov dát a služieb na ich spracovávanie ako napríklad: heroku, amazon webservices, Google Cloud Platform a treasure ale aj pre ďalšie spracovanie službami Elasticsearch a Hadoop.

Pre oddelenie jednotlivých vstupov a ich spracovávanie používa hierarchiu značiek – „tagov“ , k určeniu zdroja záznamu a typu jeho spracovania.

Vďaka implementácií v jazykoch C a Ruby sa vyznačuje nízkymi pamäťovými nárokmi, zatiaľ čo poskytuje vysoký výkon. Pre zvýšenú spoľahlivosť implementuje vyrovnácie pamäte, opakované zasielanie dát pri zlyhaní a ošetrenie chybových stavov. Pri zasielaní dát podporuje zašifrované odosielanie a prijímanie dát, pre ich zabezpečenie voči odposluchu.

Použitie nástroja spočíva v nastavení konfiguračného súboru, ktorý pre jednotlivé zdroje a výstupy špecifikuje ich formát a parametre spracovania. Vytvorením jednotlivých blokov




Obr. 4.3: Fluentd architektúra, prevzato z [6].

pre spracovanie možno jednotným spôsobom spracovávať niekoľko rôznych vstupov, prípadne jeden vstup spracovávať viacerými výstupnými službami. Konfigurácia vstupov a výstupov je závislá na značkách zaručujúcich jednoznačnú identifikáciu.

## 4.3 Splunk

Splunk [12] predstavuje robustný nástroj zameraný na pokročilé spracovávanie a analýzu dát. Poskytuje možnosti rýchleho vyhľadávania a indexovania zozbieraných dát s možnosťami vizualizácie formou grafov a tabuliek.

Centralizované riešenie zahŕňa integráciu bezpečnostného a monitorovacieho modulu pre jednoduchú kontrolu stavu systému, uložených dát a reportovanie udalostí. Špeciálnou súčasťou systému je kolektor udalostí schopný zbierať dáta z rôznych zdrojov. Splunk poskytuje obmedzenú verziu a je primárne komerčný nástroj.

## 4.4 Scribe

Scribe [11] je server určený ku agregácií logovacích dát streamovaných v reálnom čase z veľkého množstva serverov. Scribe bol vyvinutý firmou Facebook a je implementovaný ako C++ server s open-source licenciou.

Architekúra Scribe má formu orientovaného grafu. Každý uzol predstavuje jeden server, na ktorom je nainštalovaný Scribe a slúži k agregácií dát a ich zasielaní na ďalší server. Výsledkom je centrálny server, ktorý agreguje a zbiera dáta z celej siete.

Táto architektúra sa vyznačuje spoľahlivosťou, vďaka technike opakovaného zasielania dát. V prípade, že klient nie je schopný odoslať dáta na ďalší server, tieto dáta si lokálne uloží a po náhodných intervaloch sa pokúsi pripojiť a znovu ich odoslať.

## 4.5 System log

System log (Syslog) [19] je systémový štandard pre vytváranie logovacích záznamov na unixových platformách. Štandard spočíva z vytvorenia správy o zázname zo zariadenia a jej zaslaní prostredníctvom démona syslogd na logovací server. Týmto spôsobom možno

zaznamenávať dáta nielen zo sieťových zariadení ale aj z aplikačnej vrstvy. Výhodou je centralizácia správ a možnosť vytvárania správ oddelene od ich spracovávania a zbierania.

Formát dát je špecifikovaný do troch súčastí:

**Facility** identifikácia zdrojového zariadenia dát

**Severity** úroveň dôležitosti dát

**Message** správa

## 4.6 Logovanie ako služba

Logovanie ako služba predstavuje model, kedy sa záznamy o prevádzke vytvárajú u užívateľa, ale zasielajú a spracuvávajú sa treťou stranou. V tomto prípade dochádza ku zjednodušeniu architektúry na užívateľskej strane spojením s komerčným riešením u poskytovateľa. Príkladom takýchto služie založených na predošlých nástrojoch sú riešenia ako Elastic cloud v ELK architektúre a treasure data využívajúce nástroj td-agent. Okrem nich existuje množstvo ďalších služieb, ktoré podporujú množstvo nástrojov ako napríklad loggly, papertrail logging alebo logentries.

# Kapitola 5

# Návrh riešenia

Spracovanie prevádzkových záznamov je možno rozdeliť na tri samostatné vrstvy komunikujúce spolu v špecifikovanom formáte dát. Tento prístup zaručí modularitu výsledného riešenia – teda možnosť jednotlivé vrstvy podla potreby zmeniť alebo rozšíriť.

Najnižšia vrstva súvisí s návrhom tvorby prevádzkových záznamov priamo v serverových aplikáciach, ich začlenením do jednotlivej logovacej vrstvy. V prípade najnižšej vrstvy sa jedna o špecifikáciu jednotného formátu správ, zmenou existujúcich protokolov a zavedením jednotnej logovacej knižnice. Komunikácia so strednou vrstvou by obsahovala neštrukturované dáta v textovej podobe vytvorené jednotlivými logovacími knižnicami. Úlohou strednej vrstvy je tieto dáta štrukturovať do jednotného vnútorného formátu pre možnosť jednotného spracovania. Ďalšou úlohou je tieto dáta jednotne uložiť lokálne, alebo v prípade záznamov na vzdialenom stroji ich bezpečne odoslať do úložiska. Do poslednej, najvyššej vrstvy tieto dáta prichádzajú v jednotnej štrukturovanej podobe a spracovávajú sa užívateľom podľa potreby.

## 5.1 Požiadavky na systém logovania

Systém pre spracovanie prevádzkových záznamov by mal zaručiť zjednodušenie prístupu a práce s prevádzkovými záznamami, preto je potrebné zaviesť základné požiadavky na systém, na ktoré je potreba prihliadať pri jeho návrhu a implementácií.

**Nástroje pre operácie s dátami** Hlavným požiadavkom je využívať open-source alebo bezplatné nástroje. Nástroje by mali zahŕňať funkcionalitu vyhľadávania dát pre konkrétne zariadenie v danom časovom období a jednoduchú možnosť filtrovania dát pre jednotlivé zariadenia. Interakcia užívateľa je vhodná formou grafického rozhrania alebo rozhrania príkazového riadku.

**Rozšíriteľnosť** Systém musí podporovať pridávanie nových formátov vstupov a zároveň akceptovať aktuálne vstupy. Jedným z ďalších základných požiadavkov je spracovávanie záznamov zo vzdialených serverov. Je potreba, aby navrhnutý systém bolo možné jednoducho rozšíriť pridaním ďalších nástrojov na spracovanie dát. Výsledné riešenie musí zahŕňať možnosť rozšírenia systému o ďalšie servery. V prípade slabej priepustnosti systému pri využití logovacích súborov je potreba implementovať databázové riešenie.

**Využitie systémových zdrojov** Robustnosť systému nesmie mať za dôsledok nadmerné využitie systémových zdrojov ako je procesorová časť, RAM a disková pamäť. Pri systémoch náročných na tieto zdroje je potreba ich optimalizovať, aby bolo využitie zdrojov minimálne, ale zároveň, aby nedošlo k obmedzeniu použiteľnosti celého systému. Kompletné riešenie nesmie zaberať viac ako desatinu kapacity pamäti RAM.

**Bezpečnosť dát** Množstvo pripojených zariadení a zdrojov vstupu znamená možné zraniteľnosti v systéme [20]. Užívateľské dáta je potreba zabezpečiť proti cudziemu prístupu, neautorizovaný užívateľ nesmie získať prístup k dátam. Dáta užívateľov nemôžu byť skompromitované počas zasielania, počas komunikácie je potrebné dáta šifrovať. Dáta musia pochádzať z overených zdrojov, nie je prípustné, aby boli počas komunikácie pozmenené [17].

## 5.2 Tvorba dát

Prevádzkové záznamy vznikajú v jednotlivých serverových aplikáciach, teda je potrebné zaručiť kompatibilitu naprieč jednotlivými súčasťami. Každá časť kladie rôzne požiadavky a implementuje rozličný prístup k prevádzkovým záznamom, teda je potreba zjednotiť tento proces. Riešením je vytvorenie univerzálnej logovacej knižnice, alebo zmena existujúcich riešení a ustanovenie jednotného protokolu.

Požiadavky na knižnicu zahrňujú potrebu rozlíšenia úrovní tvorby prevádzkových záznamov kvôli potrebe zachytávania záznamov na nízkej úrovni za účelom ladenia chýb na jednotlivých častiach serveru a zároveň kompromis voči prostriedkom využívaným knižnicou pri bežnej prevádzke.

Táto knižnica by mala za účel zapúzdrenie akcií a práce s logovacou vrstvou, zjednotenie procesu tvorby logov naprieč systémovými aplikáciami a následne jednoduchšiu implementáciu spracovávania prevádzkových záznamov a práce s logovacou vrstvou.

Ďalšou požiadavkou je odlíšenie zdrojov záznamov. Pre jednotlivé serverové aplikácie, automatizačné úlohy alebo ďalšie časti systému je vhodné rozlíšiť pôvod záznamu. Pridaním značiek je zaistená jednoduchá lokácia pôvodu a možnosť záznamy filtrovať spolu s ďalšími časťami záznamu.

Jednotlivý logovací formát teda možno ustanoviť na základe týchto polí:

**Timestamp** Pokročilé nástroje pre prácu s dátami podporujú automatické generovanie časových razítok podľa času prijatia záznamu. Tento čas ale nie je totožný s časom vytvorenia záznamu, a je potrebný pre prehľadnosť dát v logovacích súboroch. Nástroje určené na spracovanie záznamov podporujú vlastné definovanie tohoto formátu..

**Tag** Predstavuje identifikátor úlohy, ktorá vytvorila záznam. S ohľadom na architektúru systému BeeeOn je potreba rozlíšiť server pôvodu, serverovú aplikáciu a identifikátor časti alebo úlohy. Vytvorením tejto hierarchie možno v centralizovanom úložisku dát rozlíšiť dáta pochádzajúce z rôznych serverov a aplikácií a zároveň poskytuje možnosť pre filtráciu záznamov z jednej aplikácie. Príklad hierarchickej štruktúry pre serverovú aplikáciu ada_server bežiacu na produkčnom a vývojarskom serveri s časťami sender a receiver:

```
ant-2.ada_server.sender
                .receiver
```

```
cloud.ada_server.sender
                .receiver
```

**Thread ID** Identifikátor vlákna, v ktorom došlo k vytvoreniu záznamu. Tento identifikátor možno jednoducho získať pomocou systémového volania gettid. Účelom tohto identifikátoru je rozlíšiť zdroje záznamov vo viacvláknových programoch, najmä v prípade dynamického pridelovania obslužných rutín vláknam.

**Origin** Identifikátor počiatku správy využíva makrá pre-processoru slúžiace k identifikácií riadku zdrojového kódu, z ktorého daná správa vznikla, čo má za následok značné zníženie času potrebného pre lokalizáciu chyby. Konkrétny formát obsahuje zdrojový súbor, oddeľovač „:“ a číslo riadku. Pre tento identifikátor sa využívajú makrá „___FILE___“, „___LINE___“ a taktiež možno použiť makro „___FUNCTION___“ v tele správy.

**Level** Identifikátor úrovne dôležitosti spracovávaného záznamu. Priame využitie je pri spracovávaní, pri ktorom je možné záznamy filtrovať, alebo agregovať podľa ich úrovne dôležitosti. Pri kritických záznamoch je možné ich spracovávanie zvláštnym spôsobom, prípadne špecifikácia upozornení na tieto záznamy

Návrh jednotlivých úrovní záznamov, podľa dôležitosti od najmenej kritických až po najviac kritické:

- TRACE – najhustejšie záznamy určené na sledovanie toku programu
- MSG – záznamy pre ladenie sieťového prenosu, obsahujúce dáta komunikácie
- DEBUG – husté záznamy určené pre ladenie aplikácie obsahujúce pomocné premenné
- INFO – informačné záznamy obsahujúce stavové premenné
- WARNING – neočakávané stavy aplikácie, ktoré nemajú veľký vplyv na jej beh
- ERROR – chyby aplikácie, ktoré umožňujú jej ďalší beh
- FATAL – chyby aplikácie, ktoré vedú k následnému ukončeniu aplikácie

Tieto úrovne sú založené na štandartoch zavedených v protokoloch Apache [1] a Syslog [13] upravené na aktuálny systém rozšírením o úroveň MSG za účelom spätnej kompatibility s aktuálnymi riešeniami.

**Message** Telo záznamu zobrazujúce samotnú správu s informáciou o udalosti alebo komentár a stavové premenné.

## 5.3 Zber a uloženie dát

Vytvorené prevádzkové záznamy sa prostredníctvom logovacej knižnice zapisujú do súboru alebo na štandartný výstup. V prípade spustenia aplikácie ako služba v systeme je tento štandardný výstup zapisovaný do systémového žurnálu. Tieto záznamy z týchto logovacích súborov alebo žurnálu je potrebné odoslať do centralizovaného úložiska. V prípade logovacích súborov nie je možné akceptovať statickú definíciu ciest k týmto súborom, ale

je potrebné brať do úvahy rotáciu logovacích súborov. Táto problematika je rozšírená o potrebu zabezpečeného odosielania v prípade aplikácií bežiacich na vzdialených serveroch.

Ďalšou úlohou je tieto záznamy spracovať a preformátovať na jednotnú reprezentáciu. Z dôvodu nekompatibility existujúcich logovacích knižníc na serverových aplikáciach je potreba akceptovať rôzne formáty záznamu. Jednotný formát dát s rozlíšením jednotlivých vstupov dovoľuje jednotné ukladanie týchto dát vo forme súborov alebo databázy. Pri menšom objeme dát je vhodné záznamy ukladať do súborov a využiť komprimáciu pre staršie súbory. Implementácia databázy vyžaduje ďalšie systémové zdroje a zväčšuje robustnosť riešenia, ale poskytuje lepšiu škálovateľnosť. Jednotný formát je potrebné prispôsobiť nástrojom na spracovanie dát vo vyššej vrstve.

Pre účely zasielania dát do jednotného úložiska je potrebné využiť Filebeat, pôvodne nazvaný ako Logstash Forwarder. Úlohou tohoto programu je zaslať vytvorené dáta pomocou lumberjack protokolu do jednotného úložiska. Jednotné úložisko v tejto architektúre reprezentuje program Logstash. Tento program využíva grok výrazy pre prevod neštrukturovaných dát na vnútornú formu.

## 5.4 Spracovanie a operácie s dátami

Formát dát predstavuje presnú špecifikáciu jednotlivých polí prevádzkového záznamu a určuje štruktúru záznamu. Jednotlivé položky tejto štruktúry sú závislé na serverových častiach, avšak musia sa podrobiť jednotnému návrhu. Jedným z hlavných cieľov je možnosť zoskupiť a vyfiltrovať tieto dáta na všetky záznamy týkajúce sa špecifického zariadenia. Aby bolo toto možné, je potrebné udržiavať kompletné informácie priamo v zázname a vylúčiť závislosť na predošlých záznamoch. Viacriadkové záznamy je potrebné teda zlúčiť do jedného alebo definovať viacriadkový jednotný formát, ktorý umožňuje lepšie spracovanie záznamov nástrojmi.

Reprezentácia dát v jednotnej forme predstavuje možnosť prezentovať užívateľovi spracované dáta v prehľadnej forme. Pri presnej špecifikácií formátu dát je možné tieto dáta použiť na vykreslenie grafov historických hodnôt jednotlivých stavových premenných aplikácií.

Vďaka jednotnému formátu je taktiež možné využívať jednotlivé polia záznamov pre porovnávanie parametrov jednotlivých aplikácií. Výsledkom je možnosť využiť tieto dáta pre udržiavanie agregovaných metrík, možných pre indikáciu nesprávnej činností aplikácií. Jedným z takýchto metrík môže byť počet správ s úrovňou dôležitosti „ERROR“ alebo „FATAL“ indikujúcich zlyhanie aplikácie vyžadajúce zásah vývojára.

Na spracovanie dát a operáciami s nimi sa v tejto vrstve využíva program Elasticsearch, ktorý zabezpečuje plnotextové vyhľadávanie v takmer reálnom čase. Pre vizualizáciu a export dát je vhodné webové rozhranie Kibana, ktoré spája architektúry a teda poskytuje možnosť pre dotazy a vyhľadávanie dát. Výsledok týchto hľadaní možno vizualizovať formou informačných nástenieck.

# Kapitola 6

# Implementácia

Táto kapitola sa zaoberá využitím technológií podľa návrhu. Cieľom je predstaviť kroky potrebné pre inštaláciu a nakonfigurovanie funkčného systému spracovávania prevádzkových záznamov na danom serveri.

## 6.1 Použité technológie a nástroje

Pôvodný návrh implementácie počítal s architektúrou ELK stack teda implementáciou jednotlivých nástrojov Elasticsearch, Logstash a Kibana. Toto riešenie sa radí medzi veľmi populárne vďaka svojím pokročilým funkciám a možnosti škálovať riešenie s rozširujúcou sa architektúrou. Zvykom je prideliť samostatný server na logovacie služby, prípadne prideliť samostatný server každej službe. Vďaka rozdeleniu riešenia na tieto služby možno potom celý systém škálovať na riešenie spočívajúce zo stoviek až tisícov serverov.

Problémom tohto riešenia je ale jeho náročnosť na zdroje, architektúra systému BeeeOn sa skladá z dvoch serverov s obmedzenými systémovými zdrojmi, ktoré sa prideľujú medzi množstvo aplikácií. S ohľadom na tento problém a požiadavky na minimalizáciu zdrojov nastala teda zmena v architektúre. Alternatívou ku produktu Logstash je Fluentd, respektíve td-agent, ktorý zahŕňa funkcionalitu Logstash a vyznačuje sa nízkou spotrebou systémových zdrojov. Táto zmena ovplyvnila aj najnižšiu vrstvu a transportnú vrstvu, keďže td-agent možno využívať aj v server-client architektúre viacerých serverov, kedy jedna inštancia td-agent slúži ako kolektor a druhá ako odosielateľ.

### 6.1.1 systemd

Systemd je systémový nástroj určený na správu a spúšťanie užívateľských služieb, zavedenie operačného systému. Stav tohoto démona je možné zmeniť použitím nástroja systemctl, ktorý má za úlohu manipulácie s užívateľskými službami a to ako ich spustenie alebo zastavenie, ale aj konfiguráciu automatického spustenia pri štarte pomocou príkazu enable.

Úloha tohoto nástroja v systéme logovania spočíva z dvoch obdobných úloh:

- Spustenie serverových aplikácií, ktorých výstup sa loguje ako služby
- Spustenie nástrojov systému logovania ako služby

Spustenie jednotlivých serverových aplikácií ako služby v systemd prináša výhodu v automatickej správe služby, možnosti reštartu pri zlyhaní a jednotného výstupu aplikácie. Štandartný výstup takejto služby je uložený do systémového žurnálu, ktorého obsah je

dostupný pomocou nástroja journalctl. K výstupu serverovej aplikácie sa potom pridajú značky, pomocou ktorých je možno v žurnále vyhľadávať. Pre priamy prístup k záznamom systemctl implementuje funkcionalitu prepínača -f, podobne ako v systémovej utilite tail. Tento prepínač umožňuje sledovanie záznamov aplikácie v reálnom čase, obdobne ako pri výstupe do súboru. Výstup aplikácie bez pridaných značiek možno získať nasledovne:

```
journalctl -u beeeon-ui-server
|7|~139835043800832:~29.2.2016~12:41:8:184~major ver.: 1 minor ver.: 0
|7|~139835043800832:~29.2.2016~12:41:8:184~access granted
|7|~139835043800832:~29.2.2016~12:41:8:585~DB:insertNewUser
|7|~139835043800832:~29.2.2016~12:41:8:585~Error: Cannot execute query.
```

Spustenie samotných nástrojov určených pre spracovávanie prevádzkových záznamov ako službu v systemd má obdobné výhody ako pri serverových aplikáciach. Tieto služby možno jednoducho spustiť, reštartovať a zistiť ich stav. Dôležitými funkcionalitami sú znovuzavedenie aplikácie v prípade jej zlyhania a automatické spustenie služby pri štarte operačného systému. Rovnako ako ostatné aplikácie aj logovacie nástroje vytvárajú prevádzkové záznamy, ktoré je potrebné v prípade zlej konfigurácie alebo chyby prehľadávať pre lokalizáciu chyby.

### 6.1.2 td-agent

Td-agent je stabilná verzia distribuovaného balíčku Fluentd. Td-agent slúži na jednotné spracovávanie rozlične štrukturovaných správ. Td-agent na rozdiel od Fluentd distribúcie obsahuje inicializačné skripty, optimalizovanú správa pamäti a inštaluje sa pomocou balíčkov.

**Úloha td-agent v systéme logovania** Úlohou td-agent je spojiť spracovávanie prevádzkových záznamov zo serverových aplikácií v systéme BeeeOn, ktoré ukladajú záznamy do logovacích súborov alebo systémového žurnálu. Okrem lokálnych záznamov je potrebné prijímať záznamy zo vzdialených serverov. V tomto prípade sa vytvorí klient-server architektúra, kedy obidva servery obsahujú inštaláciu td-agent, jedna inštancia td-agent figuruje ako klient – zasielateľ a druhá ako server – kolektor. Toto riešenie predpokladá škálovanie architektúry do budúcna a podporuje viacero serverov hierarchicky prijímajúcich dáta z nižších vrstiev a zasielajúcich dáta do vyšších vrstiev.

Pre rozšírenie základnej funkcionality nástroja td-agent sa využívajú zásuvné moduly (pluginy), ktoré základná inštalácia neobsahuje a vyžadujú samostatnú inštaláciu a konfiguráciu. Na prijímanie dát zo systémového žurnálu slúži plugin `fluent-plugin-systemd`. V prípade potreby preposielania vzdialených záznamov je datový prenos cez verejnú sieť potrebné zabezpečiť. Túto funkcionalitu podporuje `fluent-plugin-secure-forward`. Pre napojenie td-agenta na zvyšok EFK architektúry(Elasticsearch, Fluentd, Kibana) je potrebné zasielanie dát do Elasticsearch, ktoré implementuje `fluent-plugin-elasticsearch`.

**Td-agent vs Logstash** Pôvodný návrh systému logovania počítal s kompletnou ELK architektúrou, avšak po zvážení požiadaviek na systém a dostupných technológií došlo k nahradeniu nástroja Logstash nástrojom td-agent. Toto riešenie je taktiež populárne a občas komunitou označované ako EFK architektúra. Tieto dva nástroje predstavujú rovnakú funkcionalitu a podobné vlastnosti, ale implementačne sa značne odlišujú.

Td-agent využíva CRuby s výkonnostne náročnými časťami iplementovanými v jazyku C, zatiaľ čo Logstash využíva JRuby, ktorý vyžaduje podporu jazyka Java. Závislosť na jazyku Java znamená vyššiu náročnosť na pamäť, čo je priamo v rozpore s požiadavkami na systém logovania.

Pre rozlíšenie jednotlivých zdrojov dát a ich štruktúry Logstash používa konfiguračný súbor so štruktúrou pre datový vstup, výstup a filtrovanie správ. Pre spracovanie správ podporuje grok, zatiaľ čo td-agent potrebuje pre túto funkcionalitu komunitný plugin.

Td-agent využíva konfiguračné súbory, ktoré obdobne obsahujú sekcie pre dátový zdroj a filter. Rozdiel spočíva v rozlíšení jednotlivých správ pomocou značiek, na základe ktorých dochádza k ich ďalšiemu spracovaniu.

Jedným z kritických nedostatkov td-agent bola absencia podpory prostredia Windows v skorších verziách, ale kvôli orientácií systému logovania BeeeOn na unixové prostredie toto nepredstavuje problém. Novšie verzie td-agent prinášajú podporu pre prostredie Windows. Td-agent rovnako ako Logstash podporuje priamu integráciu so zvyškom ELK architektúry a podporuje značné množstvo rozšírení.

**Inštalácia** Inštalácia tohto nástroja nevyžaduje manuálnu konfiguráciu a pozostáva zo sledovania krokov oficiálneho návodu. Stabilná distribúcia td-agent prichádza spolu s inštalačným skriptom, ktorý je možno spustiť a automaticky nainštalovať td-agent.

```
curl -L https://toolbelt.treasuredata.com/sh/install-redhat-td-agent2.sh \
    | sh
```

Po vykonaní tohto príkazu je možno td-agent manuálne spustiť, prípadne využiť inicializačného skriptu, ktorý je súčasťou inštalácie.

V systéme BeeeOn sa na serverovej časti k obsluhe služieb používa systemd. Systemd unit nie je súčasťou inštalácie, ale je možno využiť inicializačné skripty. Pre správu nástroja td-agent je teda možné spustiť ho ako jednotku v systemd (systemd unit), konkrétne ako službu (service). Pomocou systemctl možno povoliť spustenie služby pri štarte a službu spustiť, zastaviť alebo reštartovať. Syntax príkazov, pomocou ktorých možno ovládať jednotky systemd:

```
systemctl enable unit
systemctl start unit
systemctl stop unit
systemctl restart unit
```

V základnej konfigurácií obsahuje td-agent tieto cesty:

- Konfiguračná zložka /etc/td-agent/
- Konfiguračný súbor td-agent /etc/td-agent/td-agent.conf
- Datová zložka /var/log/td-agent/
- Logovací súbor td-agent /var/log/td-agent/td-agent.log

**Kontrola nastavenia** Základná konfigurácia po inštalácií obsahuje položku určenú pre manuálne zasielanie správ. Táto sekcia definuje source, teda vstup dát, ktoré sa následne spracovávajú nástrojom td-agent.

```
#HTTP Input
<source>
  type http
  port 8972
</source>
```

Pre spracovanie týchto dát je potreba pridať sekciu match, táto sekcia definuje typ spracovania pre tieto správy. Príklad pre spracovanie manuálnych dát na kontrolu do súboru a na štandartný výstup:

```
#debug output to file
<match debug.file>
type file
path /var/log/td-agent/dbg
</match>

#debug output to stdout
<match debug.**>
type stdout
</match>
```

Overenie tejto funkcionality možno otestovať manuálnym vložením dát na definovaný HTTP Input. Pre jednoduchý test postačia tieto príkazy:

```
curl -X POST -d 'json={"data":"test"}' http://localhost:8972/debug.stdout
curl -X POST -d 'json={"data":"test"}' http://localhost:8972/debug.file
```

Tieto príkazy pozostávajú z POST dotazu na adresu HTTP vstupu s parametrom, ktorý obsahuje značku definujúcu typ spracovávania a dátami definujúcimi zaslanú správu. Po ich vykonaní dôjde k spracovaniu správ na základe značiek. V prípade štandartného výstupu sa správy zapíšu na štandartný výstup v prípade priameho spustenia v príkazovom riadku, v prípade spustenia td-agent ako služby sa zapíšu do logovacieho súboru td-agent. V prípade výstupu do súboru sa záznamy zapíšu do súboru špecifikovaného parametrom path, konkrétne umiestneného do logovacej zložky.

**Konfigurácia** Konfigurácia td-agent spočíva zo špecifikácií sekcií pre špecifikáciu vstupu, výstupu a sekcií pre spracovanie záznamov. Tieto sekcie spočívajú z:

- source - datový vstup
- filter, buffer - sekcie zaručujúce spolahlivosť procesu
- match - sekcia pre spracovanie dát na základe značky a typu

Konfigurácia štandartne spočíva z konfiguračného súboru td-agent, avšak pre podporu modularity systému možno vytvoriť ďalšie konfiguračné súboru a pomocou direktívy `@include` možno tieto súbory spojiť do jednej konfigurácie.

Konfigurácia systému logovania využíva plugin tail pre sledovanie súboru a plugin systemd pre sledovanie systémového žurnálu. Pre spracovanie dát sa používa Elasticsearch plugin.

**Plugin tail** Plugin in_tail zabezpečuje datový vstup zo súboru a je obsiahnutý v jadre Fluentd, teda nie je potrebné ho manuálne inštalovať.

Funkcionalitou je podobný unixovej utilite `tail` s prepínačom -f, teda sleduje daný súbor a akékoľvek zmeny zasiela na výstup. Konfigurácia tohto pluginu spočíva zo špecifikácie cesty k súboru v parametri `path`, pozičného súboru určného k uloženiu aktuálnej pozície v sledovanom súbore `pos_file` a značky určenej k identifikácií správ pre spracovanie. Posledným a najdôležitejším je parameter `format`, td-agent podporuje množstvo predom definovaných formátov, avšak manuálne protokoly je potrebné formátovať pomocou špecifikáci regulárneho výrazu.

Špecifikácia formátu pomocou regulárneho výrazu vyžaduje použitie pomenovaných záchytných skupín, ktoré definujú kľúče a regulárnych výrazov, ktoré definujú hodnoty vo výslednom zázname.

```
format [(?<time.*>)] (?<tag>.*) [(?<thread_id.*)] (?<location>.*:.*) \
 (?<severity>.*) (?<message>.*)
```

V prípade zasielania neštrukturovanej správy možno formát nešpecifikovať. Príklad konfigurácie pre serverovú aplikáciu ada_server bežiaci na serveri ant-2 v systéme BeeeOn:

```
<source>
  tag beeeon.ant-2.ada_server
  type tail
  path /var/log/ada_server/*.log
  pos_file /var/log/td-agent/ada_server.log.pos
  format none
</source>
```

**systemd-plugin** Systemd plugin ma za úlohu poskytnúť datový vstup pomocou ktorého možno dynamicky načítavať dáta zo systémového žurnálu. Jeho inštalácia je pomocou nástroja `td-agent-gem`:

```
td-agent-gem install fluent-plugin-systemd
```

Konfigurácia tohto pluginu spočíva v uvedený parametru `pos_file`, podobne ako pri plugine in_tail, ktorého úlohou je sledovať aktuálny index v systémovom žurnále. Ďalším parametrom je špecifikácia parametru `filter`. Zatiaľ čo je možné prijímať záznamy aj bez jeho špecifikácie, kvôli množstvu záznamov obvykle sa vyskytujúcom v systémovom žurnáli je vhodné špecifikovať jednotku, ktorej dáta sú konkrétne dôležité. Tento parameter používa nastavenie obdobné využitiu nástroja journalctl. Nastavenie parametru `strip_underscores` je dôležité kvôli zasielaniu dát do nástroja Kibana, ktorý využíva znaky „_“ pred názvom poľa pre špecifikáciu vlastných premenných. Príklad konfigurácie pluginu:

```
<source>
  type systemd
  path /var/log/journal
  pos_file /var/log/td-agent/systemd.pos
  filters [{ "_SYSTEMD_UNIT": "beeeon-ui-server.service" }]
  tag beeeon.ant-2.ui_server
  strip_underscores true
</source>
```

**Elasticsearch plugin** Plugin `fluent-plugin-elasticsearch` slúži ako datový výstup pre Elasticsearch. Jeho konfiguráciou možno odosielať dáta Logstash formáte. Jeho inštalácia je možná pomocou príkazu:

```
td-agent-gem install fluent-plugin-elasticsearch
```

Konfigurácia tohto pluginu spočíva zo špecifikácie adresy služby Elasticsearch, nastavenia parametru `logstash_format`, ktorý slúži k špecifikácií vyžadovaných polí a nastavenia parametru `include_tag_key`, ktorý slúži na vynútené pridanie značky do obsahu správy.

```
<match beeeon.ant-2.ada_server>
  type elasticsearch
  host localhost
  port 9200
  logstash_format true
  include_tag_key true
</match>
```

### 6.1.3 Elasticsearch

Úlohou Elasticsearch v systéme spracovania prevádzkových záznamov je poskytnúť možnosť analyzovať dáta. Pomocou tohto nástroja je v týchto dátach možné vyhľadávať pomocou dát uložených v jednotlivých dokumentoch - indexoch.

**Inštalácia a Konfigurácia** Inštalácia nástroja Elasticsearch prebieha pomocou nasledovného príkazu:

```
sudo yum -y install elasticsearch
```

Pre následnú konfiguráciu tejto služby je potrebné upraviť konfiguračné súbory. Jedným z konfiguračných súborov je súbor `/etc/elasticsearch/elasticsearch.yml`, v ktorom možno špecifikovať nastavenia služby a súbor `/etc/elasticsearch/logging.yml` pre nastavenie samotných záznamov tejto služby.

**curl API** Pre prácu s Elasticsearch neni potrebné webové rozhranie kibana, komunikácia je možná aj pomocou json dotazov na adresu Elasticsearch bloku. Pomocou týchto príkazov možno pridávať alebo mazať nové dokumenty, meniť nastavenia alebo zistiť stav bloku. Príkaz pre zistenie stavu bloku:

```
curl -XGET 'http://localhost:9200/_cluster/health?pretty=true'
```

**Elasticsearch template** Pre formátovanie jednotlivých vstupných polí využíva Elasticsearch mapovanie nastavené v jednotlivých indexoch. Nastavením mapovania možno určiť datový typ poľa alebo nastaviť jeho spracovanie pomocou analyzátoru. Vo väčšine prípadov je predvolené nastavenie správne, avšak značky a iné polia obsahujúce znaky „-“ a „.“ sú analyzované. Analýzou týchto polí dochádza k ich rozdeleniu na časti podľa týchto znakov, čo je nevhodné najmä pri vyhľadávaní značiek alebo polí hostname.

Špecifikácia mapovania zaručuje definovania datového typu a analyzovania polí. Pre túto špecifikáciu sa využíva Elasticsarch PUT API na vytvorenie predvolenej šablóny. Východzie mapovanie polí pre daný index možno nájsť príkazom:

```
curl -XGET localhost:9200/logstash-2016.04.17/_mapping/?pretty
```

Výsledkom je štruktúra zobrazujúca nastavenie daného indexu a jeho typov. Špecifikáciou šablóny možno vytvoriť predvolené hodnoty, ktoré sa aplikujú pri vytvorení nového indexu a to v prípade logstash-* nového indexu pre každý deň. Pre vytvorenie a zmazanie šablóny možno použiť tieto príkazy:

```
curl -XPUT 'http://localhost:9200/_template/simple-template' \
-d@simple-template.json
curl -XDELETE 'http://localhost:9200/_template/simple-template'
```

Odpoveďou v prípade úspechu je správa „acknowledged:true“ a výsledok možno skontrolovať zobrazením aktuálnych šablón:

```
curl -XGET 'http://localhost:9200/_template/
```

### 6.1.4 Kibana

Úloha nástroja Kibana spočíva v prínose jednoduchého užívateľského rozhrania, pre operácie s dátami.

**Inštalácia a konfigurácia** Pre inštaláciu nástroja kibana možno využiť nasledovný príkaz:

```
sudo yum -y install kibana
```

Predvolená cesta ku konfiguračnému súboru je: `/opt/kibana/config/kibana.yml`. V prípade lokálneho serveru je potrebné zmeniť nastavenie „server.host“ na „localhost“. Nástroj kibana je vhodné nastaviť a spustiť ako službu systemd. Pre skompletovanie funkcionality je potreba doinštalovať nginx proxy.

**Použitie** Kibana je previazaná s nástrojom Elasticsearch a poskytuje webové rozhranie, pomocou ktorého možno jednoducho v dátach vyhľadávať a zobrazovať ich. Táto funkcionalita je dostupná v záložke Discover. Okrem funkcie webového rozhrania spojeného s Elasticsearch je hlavnou úlohou Kibany vizualizácia a formátovanie dát do užívateľsky prívetivej formy. V záložke Visualize je možno vytvárať vizualizácie dát a v záložke Dashboard možno vytvoriť kompletnú informačnú nástenku zobrazujúcu objekty vizualizácií.

**Discover** Spojenie nástrojov Kibana a Elasticsearch má za následok prepojenie funkcionality a možností vyhľadávania s webovým rozhraním pomocou textového poľa, do ktorého je možno vkladať dotazy. Týmto spôsobom sa podporuje možnosť vkladať dotazy so syntaxou knižnice Lucene, bez nutnosti rozhrania v príkazovom riadku a pokročilého formátovania dotazov. Tieto dotazy možno pre komfort užívateľa pomenovať a uložiť, čo zaručuje rýchly prístup k často požadovaným informáciám. Okrem manuálnej špecifikácie dotazu možno vytvoriť filter na základe poľa, kliknutím v detaile poľa.

Vyhľadávané dáta tak možno filtrovať na základe dostupných polí podľa formátu dát. V prípade výberu jednotlivých polí možno výsledky vyhľadávania upraviť a zobrazovať iba vybrané časti záznamu. Okrem výberu polí možno vybrať časový rozsah záznamov a to ako pevný dátum aj relatívnu dobu. Táto funkcionalita je špeciálne užitočná pri sledovaní záznamov za posledný čas, nakoľko možno jednoducho vizuálne zistiť množstvo vytvorených záznamov v priebehu časového intervalu. V spojení s možnosťou automatickej aktualizácie tak existuje možnosť zobraziť formátované záznamy za posledných niekoľko minút.

**Visualize** Zbierané dáta nie je možné len vyhľadávať a zobrazovať, ale je možné na ich základe vytvárať vizualizačné objekty. Medzi vizualizačné objekty patria nielen rôzne druhy grafov, ale aj tabuľky, mapy, poznámky a čítače metrík. Tieto objekty možno vytvárať na základe nového vyhľadávania, alebo použiť uložené vyhľadávanie.

**Dashboard** Z jednotlivých vizualizačných objektov možno vytvoriť informačnú nástenku (dashboard). Tieto nástenky poskytujú možnosť pre zoskupenie vizualizácií a predstavujú primárnu funkciu - reprezentáciu formátovaných dát prehľadne a vizuálne atraktívne.

### 6.1.5 Nginx

Úlohou nginx je slúžiť ako webový server, ktorý sprístupňuje službu Kibana. Pomocou tejto konfigurácie je možno povoliť pripojenie len z lokálnej siete a nastaviť autentifikáciu užívateľa. Nginx teda plní úlohu reverznej proxy.

**Inštalácia** Inštalácia nástroja spočíva z nainštalovania nástroja z repozitáru pomocou príkazu:

```
sudo yum -y install nginx
```

Po inštalácií nástroja je potrebné špecifikovať užívateľa pre prístup do Kibany. Tento užívateľ sa vkladá do súboru `/etc/nginx/htpasswd.users`, ktorý je špecifikovaný v nastaveniach. Po vykonaní príkazu je potrebné špecifikovať heslo.

```
sudo htpasswd -c /etc/nginx/htpasswd.users admin
```

**Konfigurácia** Nastavenie konfiguračného súboru `/etc/nginx/nginx.conf` využíva vkladanie obsahu konfiguračných súborov z konfiguračnej zložky:

```
include /etc/nginx/conf.d/*.conf;
```

Konfigurácia proxy pre Kibanu teda spočíva z umiestnenia súboru `kibana`.conf do zložky `/etc/nginx/conf.d`. Obsah konfiguračného súboru spočíva z nastavenia lokálnej adresy, autorizácie užívateľov a nastavenia proxy. Po nastavení konfiguračného súboru je možné povoliť a spustiť nginx ako služby v systemd a webové rozhranie kibana je dostupné na špecifikovanej adrese.

### 6.1.6 Jednotná logovacia knižnica

Na základe návrhu riešenia a jednotného logovacieho formátu je súčasťou systému jednotná logovacia knižnica, ktorej úlohou je poskytnúť rozhranie pre tvorbu prevádzkových záznamov na základe požiadavkvo na systém logovania.

Implementácia tejto knižnice je v jazyku C++ s podporou štandardu C++11, čo zaručuje kompatibilitu s jednotlivými časťami systému BeeeOn. Práca s knižnicou predstavuje vytvorenie objektu `unified_logger`, ktorý obsahuje nastavenia knižnice. Pomocou dvojice funkcií alebo makier, možno priamo špecifikovať správu na štandartný výstup alebo do súboru. Výhodou je minimalizácia potrebného vstupu užívateľa doplnením hlavičky správy knižnicou a podpora streamového operátoru. Implementácia pomocou streamového operátoru vyžaduje ošetrenie prekladania reťazcov na výstupe vo viacvláknovom prostredí. Synchronizáciu záznamov a prekladanie reťazcov ošetruje objekt `locked_stream`, ktorý sa

uzamyká pre jednotlivé záznamy a zaniká s posledným operátorom „«“ vo výraze. Knižnica tak podporuje správy rôzneho formátu z rôznych datových typov a nevyžaduje presné parametre funkcie pre logovanie. Logovacia knižnica bola využitá pri implementácií frameworku [21] pre automatizačné úlohy a tým integrovaná do systému BeeeOn. Príklad práce s knižnicou:

```
Unified_logger logger("framework");
logger.LOGOUT("temperature", "DEBUG") << "Entered config state" \
<< std::endl;
logger.LOGOUT("manager", "ERROR") << "Can't load humidity module" \
<< std::endl;
```

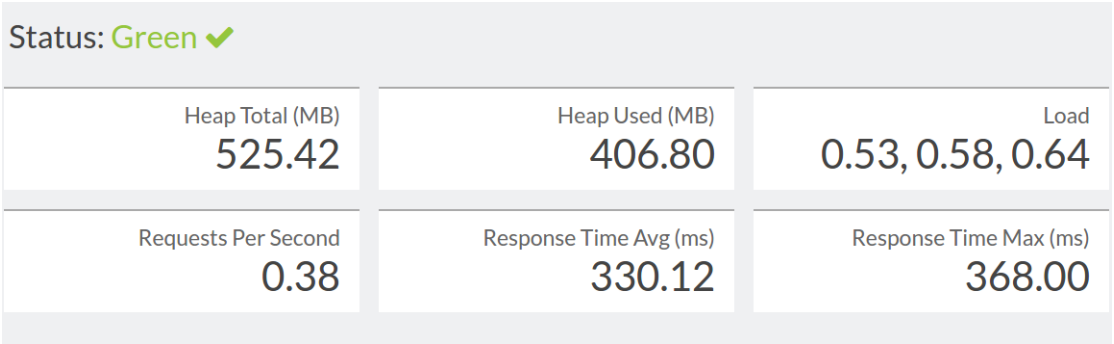
Príloha Štrukturovaná správa je príkladom výsledného štrukturovaného záznamu.

## 6.2 štatistiky a testovanie

Táto sekcia je zameraná na prezentáciu parametrov implementovaného riešenia počas jeho testovania a ich porovnanie s výslednými parametrami.

Navrhované riešenie bolo implementované na testovacom serveri, kde bolo integrované do systému BeeeOn. Systém spracovania logov bol sprístupnený tímu vývojárov, vďaka čomu priniesol pridanú hodnotu pri spracovaní prevádzkových záznamov. Počas nasadenia na testovacom serveri bolo celkovo spracovaných viac než 300 miliónov záznamov s najväčšou hustotou záznamu viac než 1.3 milióna záznamov za sekundu.

Najväčšiu záťaž na serverové zdroje predstavujú nástroje Elasticsearch a Kibana. Pomocou konfigurácie je možno nastaviť systém tak, aby bolo možno pracovať s dátami zo systému ale zároveň aby bolo minimalizované využitie systémových zdrojov. Grafy využitia systémových zdrojov vygenerované pomocou nástroja `PNP4Nagios` možno nájsť v prílohe Grafy využitia systémových zdrojov. Obrázok 6.1 zobrazuje výrez zo stránky zobrazujúcej stavové informácie o nástroji Kibana počas používania.




Obr. 6.1: Kibana status.

Integráciou zo systémom BeeeOn som došiel k záveru, že systém spracovania prevádzkových záznamov je schopný spracovať množstvo dát produkované serverovými aplikáciami. Možnosti integrovaného riešenia však ďaleko presahujú aktuálne využitie, čo je výhodné vzhľadom na budúce škálovanie systému BeeeOn a jeho rozšírenie na viacero serverov.

# Kapitola 7

# Záver

Cieľom bakalárskej práce bolo zoznámiť sa s dostupnými nástrojmi v oblasti spracovania prevádzkových záznamov a na ich základe navrhnúť a implementovať riešenie jednotného systému spracovávania týchto záznamov.

Prvá časť práce spočívala v štúdií dostupných nástrojov vhodných pre systém BeeeOn a zo špecifikácie požiadaviek na systém spracovania záznamov. Na základe týchto nástrojov a požiadaviek došlo k rozvrhnutiu systému do jednotlivých častí špecializovaných na konkrétne úlohy: tvorba dát, zbieranie a uloženie dát a spracovanie a vizualizácia dát. Tieto nástroje boli nasadené na testovacom serveri a spracovávali prevádzkové záznamy zo systému BeeeOn.

Pre tvorbu dát bol špecifikovaný jednotný logovací formát, ktorý špecifikuje formát správy vytvorenej v aplikácií, z ktorej sa zbierajú prevádzkové záznamy. Pre vytvorenie správ v tomto formáte bola vyvinutá jednotná logovacia knižnica, ktorá zabezpečuje tvorbu správ vo viacvláknových aplikáciach.

Zvyšnú časť systému tvorí trojica nástrojov td-agent, Elasticsearch a kibana. Td-agent slúži k zjednoteniu spracovávania záznamov. Pomocou tohto nástroja sa záznamy z rozličných zdrojov preposielajú v jednotnom formáte do nástroja Elasticsearch. Elasticsearch slúži pre analýzu a vyhľadávanie v záznamoch a nástroj kibana poskytuje webové rozhranie pre užívateľov.

Na základe výsledkov možno povedať, že zvolené riešenie je vhodné pre praktickú aplikáciu v systéme BeeeOn a vďaka zvoleným nástrojom poskytuje do budúcnosti vysokú možnosť škálovania. Modulárny prístup taktiež poskytuje priestor pre rozšírenie systému o ďalšie nástroje podľa potreby. V prípade budúceho rozšírenia systému o ďalšie servery je možné záznamy z týchto serverov jednoducho pridať do architektúry a taktiež zahrnúť dáta z ďalších serverových aplikácií. Možným rozšírením do budúcnosti je pridanie spracovávania záznamov zo zariadenia brán BeeeOn.

Aplikácia zvoleného riešenia je možná nielen v systéme BeeeOn, ale aj v iných projektoch, v ktorých dochádza ku spracovaniu väčšieho množstva dát a ich prezentácie vývojárom alebo užívateľom. Stanovením jednotného formátu prevádzkových záznamov a ich systému spracovávania došlo k zjednodušeniu práce vývojárov s týmito záznamami. Výhodou integrácie webového rozhrania do systému je vizualizácia dát a zjednodušenie práce so záznamami pre neznalého užívateľa.

# Literatúra


[1] *Apache log4php* [online]. 2016 [cit. 2016-4-30]. Dostupné z: https://logging.apache.org/log4php/docs/introduction.html.

[2] *BeeeOn Beta* [online]. 2016 [cit. 2016-4-30]. Dostupné z: https://play.google.com/store/apps/details?id=com.rehivetech.beeeon.

[3] *BeeeOn Repositories* [online]. 2016 [cit. 2016-4-30]. Dostupné z: https://github.com/BeeeOn.

[4] *Elasticsearch* [online]. 2016 [cit. 2016-4-30]. Dostupné z: https://www.elastic.co/products/elasticsearch.

[5] *Filebeat* [online]. 2016 [cit. 2016-4-30]. Dostupné z: https://www.elastic.co/products/beats/filebeat.

[6] *Fluentd* [online]. 2016 [cit. 2016-4-30]. Dostupné z: http://www.fluentd.org/.

[7] *Jablotron* [online]. 2016 [cit. 2016-4-30]. Dostupné z: http://www.jablotron.com/sk/.

[8] *Kibana* [online]. 2016 [cit. 2016-4-30]. Dostupné z: https://www.elastic.co/products/kibana.

[9] *Logstash* [online]. 2016 [cit. 2016-4-30]. Dostupné z: https://www.elastic.co/products/logstash.

[10] *openHAB* [online]. 2016 [cit. 2016-4-30]. Dostupné z: http://www.openhab.org/.

[11] *Scribe* [online]. 2016 [cit. 2016-4-30]. Dostupné z: http://www.scribesoft.com/.

[12] *Splunk* [online]. 2016 [cit. 2016-4-30]. Dostupné z: https://www.splunk.com/.

[13] *syslog : Severity level* [online]. 2016 [cit. 2016-4-30]. Dostupné z: https://en.wikipedia.org/wiki/Syslog#Severity_level.

[14] *Thermona* [online]. 2016 [cit. 2016-4-30]. Dostupné z: http://www.thermona.cz/.

[15] Chuvakin, D. A. A.; Schmidt, K. J.; Phillips, C. *Logging and Log Management : The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management*. Elsevier, 2012. ISBN 978-1-59749-635-3.

[16] Hu, F. *Security and Privacy in Internet of Things (IoTs) : Models, Algorithms, and Implementations*. CRC Press, 2016. ISBN 978-1-4987-2318-3.

[17] Kent, K.; Souppaya, M. : Guide to Computer Security Log Managment. online, September 2006. Dostupné z: http://csrc.nist.gov/publications/nistpubs/800-92/SP800-92.pdf

[18] Kopetz, H. *Real-Time Systems : Design Principles for Distributed Embedded Applications*. Springer, 2011. ISBN 978-1-4419-8237-7.

[19] Matoušek, P. *Síťové aplikace a jejich architektura*. VUTIUM, 2014. ISBN 978-80-214-3766-1.

[20] McAfee : McAfee Labs 2016 Threats Predictions. Technická zpráva, McAfee Labs, november 2015. Dostupné z: http://www.mcafee.com/us/resources/reports/rp-threats-predictions-2016.pdf

[21] Novák, M. *Rozšiřitelný systém pro automatizaci domácností* Brno: Vysoké učení technické v Brně, Fakulta informačních technologií, 2016. Vedoucí práce Vampola Pavel.

[22] Čížek, J. Na brněnské FIT se rodí BeeeOn. Univerzální chytrá domácnost [online]. február 2015 Dostupné z: http://www.zive.cz/clanky/na-brnenske-fit-se-rodi-beeeon-univerzalni-chytra-domacnost/sc-3-a-181423/default.aspx.

# Prílohy

# Zoznam príloh

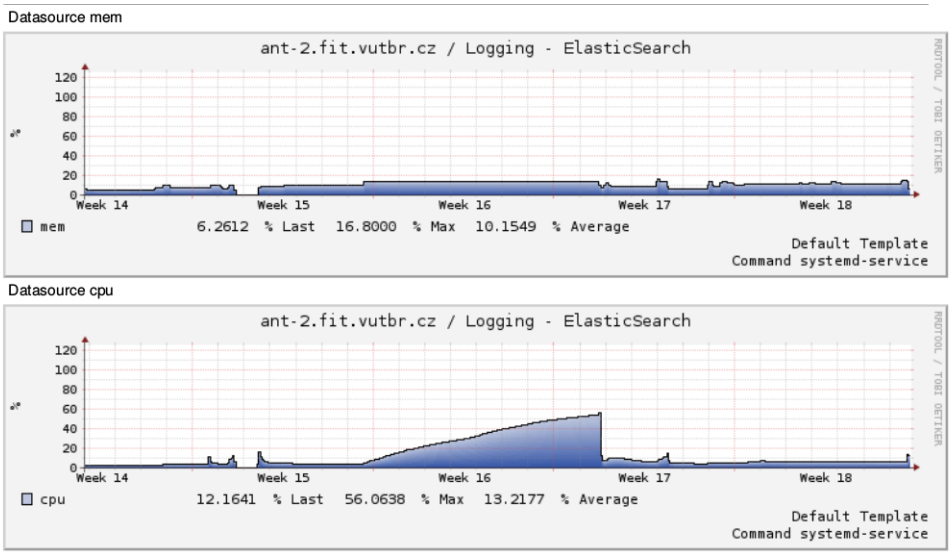
# Príloha A

# Štrukturovaná správa

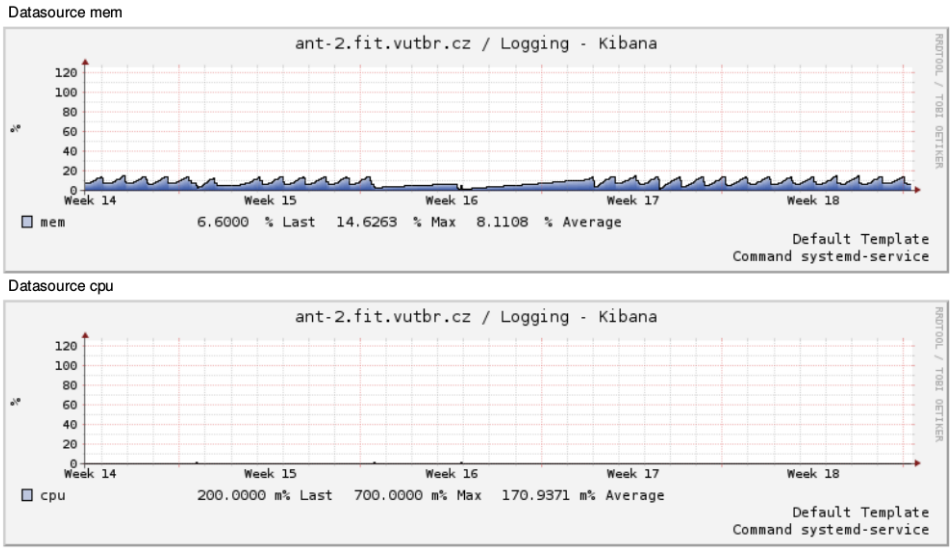

```
{
  "_index": "logstash-2016.05.05",
  "_type": "fluentd",
  "_id": "AVSBZ6bRSbHjTUj_EIcL",
  "_score": null,
  "_source": {
    "timestamp": "2016-05-05 14:51:22:958",
    "tag": "beeeon.ant-2.baf",
    "thread": "140073639077632",
    "src": "src/TriggerTaskInstance.cpp:33",
    "severity": "ERROR",
    "message": "Run of FireHazard instance was not successful.",
    "@timestamp": "2016-05-05T14:51:22.959Z"
  },
  "fields": {
    "@timestamp": [
      1462459882959
    ]
  },
  "highlight": {
    "tag": [
      "@kibana-highlighted-field@beeeon.ant-2.baf
      @/kibana-highlighted-field@"
    ]
  },
  "sort": [
    1462459882959
  ]
}
```

# Príloha B

# Grafy využitia systémových zdrojov




Obr. B.1: Využitie systémových zdrojov - Elasticsearch




Obr. B.2: Využitie systémových zdrojov - Kibana