UNIVERZITA KONŠTANTÍNA FILOZOFA V NITRE

FAKULTA PRÍRODNÝCH VIED

TVORBA AUTOMATIZOVANÉHO WORKFLOW PRE PODPORU PREVÁDZKY CISCO UNIFIED COMMUNICATION

DIPLOMOVÁ PRÁCA

2019 **Bc. DÁVID PAVELKA**

UNIVERZITA KONŠTANTÍNA FILOZOFA V NITRE

FAKULTA PRÍRODNÝCH VIED

TVORBA AUTOMATIZOVANÉHO WORKFLOW PRE PODPORU PREVÁDZKY CISCO UNIFIED COMMUNICATION

DIPLOMOVÁ PRÁCA

Študijný odbor:	9.2.9 Aplikovaná informatika
Študijný program:	Aplikovaná informatika
Školiace pracovisko:	Katedra informatiky
Školiteľ:	PaedDr. Martin Magdin, Ph.D.

Nitra 2019 Bc. Dávid Pavelka

POĎAKOVANIE

Ďakujem prof. Ing. Milanovi Turčánimu, CSc. z Katedry informatiky, bez ktorého by celá spolupráca s externou spoločnosťou nikdy nevznikla, spoločnosti Dimension Data v zastúpení Mgr. Erikom Urlandom, PhD., našim projektovým manažérom, za možnosť participovať na projekte a za vedenie pri realizácií projektu, Ing. Jánovi Kršákovi a Ing. Marcelovi Imrichovi, zamestnancom spoločnosti Dimension Data, za odborné rady a výpomoc pri realizácii projektu a v neposlednom rade môjmu školiteľovi PaedDr. Martinovi Magdinovi Ph.D., ktorý ma viedol pri písaní diplomovej práci a realizácii projektu.

ABSTRAKT

PAVELKA, Dávid: Tvorba automatizovaného workflow pre podporu prevádzky Cisco Unified Communication. [Diplomová práca]. Univerzita Konštantína Filozofa v Nitre. Fakulta prírodných vied. Školiteľ: PaedDr. Martin Magdin, Ph.D. Stupeň odbornej kvalifikácie: Magister odboru Aplikovaná informatika. Nitra: FPV, 2019. 78 s.

Automatizácia pomocou ChatOps riešenia je veľmi aktuálna téma v rámci oboru informačných technológií a vývoja automatizácií. Diplomová práca rieši implementáciu ChatOps riešenia pre podporu konfigurácie a prevádzky Cisco Unified Communications Manager-a (CUCM) s využitím open source event driven automatizačnej platformy StackStorm, implementácie ChatBot modulu s integráciou na platformu Webex Teams. V prvej časti diplomovej práci je popísané reálne využitie automatizačných nástrojov, vznik a význam ChatOps riešení a ich prínos pre DevOps. Je v nej spracovaný prehľad ChatBot riešení a opísané sú event driven platforma StackStorm, platforma CUCM a platforma Webex Teams. Druhá časť je zameraná na ciele diplomovej práce, stanovené v súlade so spoločnosťou Dimension Data. Tretia časť diplomovej práce popisuje konkrétne riešenie ChatOps so zameraním na programovacie a konfiguračné práce v rámci platformy StackStorm pri vykonávaní dopytov do CUCM databázy. Opisuje implementáciu riešenia pri konkrétnych scenároch použitia databázy. Štvrtá časť diplomovej práce obsahuje popis celých procesov konkrétnych scenárov, ich výsledky a bezpečnosť riešenia. V závere diplomovej práce sú uvedené výhody a nevýhody ChatOps riešenia a odporúčania pre prax.

Kľúčové slová: Automatizácia. ChatOps. ChatBot. StackStorm. Cisco Unified Communication Manager. Botkit. Webex Teams.

ABSTRACT

PAVELKA, Dávid: The creation of autamatized workflow for services of Cisco Unified Communications. [Diploma Thesis]. Constantine the Philosopher University in Nitra. Faculty of Natural Sciences. Supervisor: PaedDr. Martin Magdin, Ph.D. Degree of Qualification: Master of Applied Informatics. Nitra: FNS, 2019. 78 p.

Automation with the ChatOps solution is a very topical issue within the IT industry and automation development. The diploma thesis deals with implementation of ChatOps solution for configuration and operation support of Cisco Unified Communications Manager (CUCM) using the open source event driven automation platform StackStorm, implementation of ChatBot module with integration to Webex Teams. The first part of the thesis describes the real use of automation tools, the emergence and importance of ChatOps solutions and their contribution to DevOps. It features an overview of ChatBot solutions and describes the event driven StackStorm platform, the CUCM platform, and the Webex Teams platform. The second part is focused on the goals of the thesis, determined according to Dimension Data. The third part of the thesis describes a specific ChatOps solution focusing on programming and configuration work within the StackStorm platform when executing queries to the CUCM database. Describes the implementation of the solution for specific database usage scenarios. The fourth part of the thesis contains a description of the entire process of concrete scenarios, their results and the security of the solution. The thesis conclusion contains advantages and disadvantages of ChatOps solution and recommendations for practice.

Keywords: Automation. ChatOps. ChatBot. StackStorm. Cisco Unified Communication Manager. Botkit. Webex Teams.

OBSAH

ZOZNAM ILUSTRÁCIÍ A TABULIEK

ZOZNAM SKRATIEK A ZNAČIEK

WSDL **Web Services Description Language**	Je jazyk, ktorý opisuje, aké funkcie ponúka webová služba a spôsob, ako sa jej na to dopytovať. Zapisuje sa vo formáte XML a spravidla opisuje SOAP komunikáciu.
API **Application programming interface**	Aplikačné programovacie rozhranie súbor definícií podprogramov, komunikačných protokolov a nástrojov na tvorbu softvéru. Vo všeobecnosti ide o súbor jasne definovaných metód komunikácie medzi rôznymi zložkami, ktoré uľahčujú vývoj softvéru.
RPA **Robotic Process Automation**	Robotická procesná automatizácia je novovznikajúcou technológiou automatizácie obchodných procesov stavaná na práci softvérových robotov alebo umelej inteligencie.
XML **eXtensible Markup Language**	Značkovací jazyk vyvinutý konzorciom W3C. Umožňuje jednoduché vytváranie konkrétnych značkovacích jazykov a slúži na výmenu údajov medzi aplikáciami a na zverejňovanie dokumentov.
GUI **Graphical User Interface**	Grafické používateľské rozhranie, ktoré umožňuje ovládať elektronické zariadenie pomocou súboru interaktívnych obrazových prvkov, ktoré spúšťajú príkazy a umožňujú priamu interakciu so zariadením.
AXL **Administrative XML**	Je rozhranie založené na XML alebo SOAP, ktoré poskytuje mechanizmus na vkladanie, získavanie, aktualizáciu a odstraňovanie údajov z konfiguračnej databázy Unified Communication.
YAML **YAML Ain't Markup Language**	Je formát slúžiaci na serializáciu dát vo forme, ktorá je pre človeka ľahko čitateľná. Používa sa pre konfiguračné súbory, ale aj v aplikáciách, kde sa ukladajú alebo prenášajú dáta.

UUID **Universally Unique** **Identifier**	Je 128-bitové číslo používané na identifikáciu informácií v počítačových systémoch. Používa sa aj termín globally unique identifier (GUID).

ÚVOD

ChatOps je v súčasnosti čoraz viac používané slovo v rámci informatickej vedy a praxe. Zjednodušene sa dá ChatOps charakterizovať ako model kooperácie spájajúci ľudí, nástroje, procesy a automatizácie do jedného technologického, či pracovného postupu (workflow). Práve technologický postup využívajúci jednotlivé technológie a procesy charakterizuje ChatOps ako jedinečný model.

Samotné slovo ChatOps v sebe nosí anglický výraz chat vo význame komunikačného internetového kanálu, četu. V rámci využitia Cisco Unified Communication ako zdroja dátovej štruktúry máme možnosti použitia viacerých četových kanálov ako napríklad Webex Teams alebo Slack. V našej práci sme sa orientovali práve na využitie Webex Teams ako počiatočného bodu pre náš workflow.

Výsledok našej práce bude kompletné ChatOps riešenie pre podporu konfigurácie a prevádzky Cisco Unified Communications Manager (CUCM) s využitím automatizačnej platformy StackStorm a integrované na platformu Webex Teams.

V prvej kapitole diplomovej práce sa zaoberáme analýzou súčasného stavu z pohľadu reálnej aplikácie automatizácii v praxi a nástrojmi automatizácie. Analyzujeme aj vývoj, vznik a využitie ChatOps riešení a implementáciu ChatBotov do týchto riešení.

Druhá kapitola pojednáva ciele a podciele diplomovej práce zosúladené s implementovaním riešenia v spolupráci so spoločnosťou Dimension Data.

Tretia kapitola opisuje náš návrh riešenia danej problematiky. Demonštrujeme v nej tri základné scenáre použitia ChatOps riešenia a jeho aplikácie reálne v praxi pre podporu prevádzky CUCM s využitím event driven platformy StackStorm.

V štvrtej kapitole sa venujeme výsledkom diplomovej práce, priebehom jednotlivých scenárov a ich výsledkov, akými je napríklad aj skrátenie doby výkonu daných úloh používateľom a samozrejme aj bezpečnosti riešenia.

V závere popisujeme výhody a nevýhody nášho ChatOps riešenia pre aplikačnú prax.

1 ANALÝZA SÚČASNÉHO STAVU

Automatizácia je dôležitá súčasť nielen technických ale i prírodovedných, ekonomických a rôznych iných vedeckých smerov. Odkedy človek dokázal vytvoriť inteligentné riadiace systémy, snaží sa vytvárať aj automatizácie týchto systémov, automatizácie ich ovládania, či automatizácie pracovných postupov.

Automatizácia je technológia, ktorou sa proces alebo postup vykonáva s minimálnou ľudskou pomocou (Groover, 2014). Automatizácia, alebo automatické riadenie, je použitie rôznych riadiacich systémov pre prevádzkové zariadenia, ako sú stroje, procesy v továrňach, kotly a pece na tepelné spracovanie, prepínanie telefónnych sietí, riadenie a stabilizácia lodí, lietadiel a iných dopravných prostriedkov s minimálnym alebo zníženým ľudským zásahom (Rifkin, 1995). Veľký posun v automatizácii prišiel vďaka založeniu oddelenia automatizácie v roku 1947 v spoločnosti Ford. Počas tohto obdobia sa v priemysle rýchlo ujímali ovládače so spätnou väzbou, ktoré boli predstavené v 30. rokoch 20. storočia (Bennett, 1993). Práve tie mali významný prínos do algoritmizácie a vývoja výpočtovej techniky, ktorá je základom automatizácie.

Od polovice 20. storočia až dodnes nastal obrovský nárast automatizovaných procesov. Prakticky neexistuje odvetvie, ktoré by nezaznamenalo aspoň minimálny priamy či nepriamy zásah automatizácie. Dnes je to nielen priemysel, strojárstvo a výrobné procesy v ňom, ktoré sú často plne automatizované. Automatizácia je súčasťou aj poľnohospodárstva, finančného a bankového sektora, ale zasahuje aj do služieb, kde napríklad so zákazníkom na internetových stránkach a obchodoch komunikuje ChatBot, v oblasti zdravotníctva, kde pomáha diagnostikovať ochorenia a využitie má i v mnohých ďalších odvetviach.

1.1 TYPY AUTOMATIZAČNÝCH NÁSTROJOV

Rôznorodé využitie automatizačných procesov naprieč odvetviami svetového hospodárstva má za následok veľký počet vyvinutých automatizačných nástrojov, ktoré sú aplikované pre jednotlivé procesy.

Umelé neurónové siete

Umelé neurónové siete sú počítačové systémy vytvorené na princípe fungovania biologických neurónových sietí, ktoré tvoria mozog a nervový systém u živočíchov (van

Gerven a kol., 2018). Samotná neurónová sieť nie je algoritmus, ale skôr rámec pre mnoho rôznych algoritmov strojového učenia sa, ktoré spoločne pracujú a spracovávajú komplexné dátové vstupy (DeepAI, 2018).

Distribuované riadiace systémy

Distribuovaný riadiaci systém je počítačový riadiaci systém pre proces alebo zásah, zvyčajne s veľkým počtom riadiacich slučiek, v ktorých sú distribuované autonómne regulátory v celom systéme, ale s existujúcou centrálnou kontrolou. Je to zásadný rozdiel od systémov, ktoré používajú centralizované ovládanie umiestnené v riadiacej jednotke alebo v centrálnom kontrolnom počítači. Koncept distribuovaných riadiacich systémov zvyšuje spoľahlivosť a znižuje náklady na inštaláciu tým, že umiestňuje riadiace funkcie v blízkosti procesnej prevádzky so vzdialeným monitorovaním a dohľadom.

Nástroje používateľského rozhrania

Používateľské rozhranie (UI - z anglického „*User Interface*“) v oblasti priemyselného dizajnu je priestor interakcie medzi človekom a strojom. Cieľom tejto interakcie je umožniť efektívnu prevádzku a kontrolu stroja z prístupu človeka, zatiaľ čo stroj súčasne napája informácie, ktoré napomáhajú rozhodovaciemu procesu. Príklady tejto širokej koncepcie používateľských rozhraní zahŕňajú interaktívne aspekty operačných systémov, nástrojov, ovládacích prvkov operátorov ťažkých strojov a riadiacich procesov. Koncepčné aspekty použiteľné pri vytváraní používateľských rozhraní súvisia s disciplínami ako je ergonómia či psychológia. Používateľské rozhranie mechanického systému alebo priemyselnej inštalácie sa v angličtine označuje ako human–machine interface (HMI - anglicky rozhranie človek-stroj). Jedným z najznámejších používateľských rozhraní je graphical user interface (GUI), teda grafické používateľské rozhranie.

SCADA

Supervisory Control and Data Acquisition, skrátene SCADA, je architektúra riadiaceho systému, ktorá využíva počítače, sieťovú dátovú komunikáciu a grafické používateľské rozhranie pre riadenie kontrolných procesov s využitím aj periférnych zariadení. SCADA je jeden z najčastejšie používaných typov priemyselných riadiacich

systémov, avšak existujú obavy, že systémy SCADA sú ohrozené kybernetickými útokmi (Tsang, 2012).

Programovateľný logický automat

PLC z anglického Programmable Logic Controller, teda programovateľný logický automat je priemyselný digitálny počítač používaný pre automatizáciu procesov v reálnom čase. Uplatnenie našiel v procesoch vyžadujúcich si jednoduchosť a spoľahlivosť, ako je riadenie strojov, alebo výrobných liniek v továrňach.

Inštrumentácia

Prístrojové vybavenie alebo inštrumentácia (z anglického „*instrumentation*“) je spoločné pomenovanie pre senzoriku, monitoring, diagnostiku ale predovšetkým pre meranie a indikáciu zaznamenávania fyzikálnych veličín. Pojem inštrumentácia je aplikovateľný na samostatný prístroj zaznamenávajúci iba hodnoty teploty alebo komplexnú skupinu zariadení riadiaceho priemyselného systému. Jedno z najstarších použití meracieho zariadenia bolo nájdené v hrobke starovekého egyptského faraóna Amenhotepa I., pochovaného okolo roku 1 500 pred našim letopočtom (Nist Physical Measurement Laboratory, 2018).

Ovládanie pohybu

Ovládanie pohybu je kategória automatizácie, ktorá zahŕňa riadiace systémy alebo podsystémy zodpovedné za vyvolanie pohybu stroja. Medzi hlavné komponenty systémov na ovládanie pohybu patrí riadiaca jednotka pohybu, energetický zosilňovač a jeden alebo viacero primárnych posúvačov alebo ovládačov. Zvyčajne je pozícia, rýchlosť a frekvencia pohybu riadená zariadeniami ako je hydraulické čerpadlo, lineárny pohon, elektromotor alebo servo. Riadenie pohybu je dôležitou súčasťou robotiky a CNC obrábacích strojov.

Robotika

Robotika je interdisciplinárna oblasť vedy a techniky, ktorá zahŕňa strojárstvo, elektronické technológie, informačné technológie a rôzne iné. Robotika sa zaoberá návrhom, konštrukciou, prevádzkou a používaním robotov, ako aj počítačovými systémami pre ich kontrolu, senzorickú spätnú väzbu a spracovanie informácií. Robotika má využitie a potenciál v oblastiach ako je armáda, priemysel, poľnohospodárstvo, zdravotníctvo, domácnosť či nanorobotika. Od 60. rokov dvadsiateho storočia sa roboty

čoraz viac využívajú. V roku 2016 bol automobilový priemysel hlavným odberateľom priemyselných robotov s 52-percentným podielom na celkových tržbách (Robotic Industries Association, 2018).

1.2 LIMITÁCIE AUTOMATIZÁCIE

V súčasnosti nie je možné, aby boli všetky procesy plne automatizované. Nakoľko boli v posledných rokoch investované obrovské objemy financií do automatizácie procesov, je veľmi dôležité, aby na procesy dohliadal človek. Ak automatizácia dosiahne limit v podobe plnej funkčnosti zautomatizovania procesu, je ľudský faktor žiaduci na pozícii kontroly a dohliadania. Automatizácia totiž dosiahla v dnešnej dobe rozmery, pri ktorých sú zautomatizované rozsiahle procesy nakladajúce s obrovským množstvom tovarov a služieb, kde by pri malom zlyhaní alebo prerušení procesu mohlo dôjsť k rozsiahlym škodám na majetku, či službách alebo na poškodení mena spoločnosti.

Kontrola automatizácie

Dohliadanie nad automatizáciou môže mať rôzne formy. Pomerne často sa stretávame s nepretržitým vizuálnym kontrolovaním procesu. Táto forma dohľadu je častá v priemysle, najmä strojárstve, pri kontrole výrobných liniek alebo v logistike priemyslu. V softvérových automatizáciách je známa kontrola pomocou upozornení, ktoré majú podobu notifikácie. Notifikácia môže byť synchrónna alebo asynchrónna, časovo závislá alebo procesne závislá. Notifikácia môže byť uvedená vždy za väčším celkom procesu, na jeho začiatku a konci. Takáto notifikácia zabezpečí jednoduché diagnostikovanie problémového celku procesu. Pri takomto umiestnení notifikačných oznámení, či upozornení vravíme o procesne závislej notifikácii. Naopak časovo závislá notifikácia nám zabezpečuje presný dátum a čas prerušenia či chybného stavu na diagnostikovanie problémového procesu. Pri časovo závislej notifikácii vždy vieme presný čas, kedy nastal problém. Tento typ notifikácie môže nájsť uplatnenie pri procesoch, kde je žiaduca dôsledná znalosť času, do ktorého bol proces bezchybný. Uplatnenie notifikácie tohto typu môže byť i v sériovej výrobe, v softvérovej oblasti to môže byť napríklad odosielanie platieb.

Synchrónne notifikácie sú notifikácie, pri ktorých sa vykonávajú dva procesy súčasne. To znamená, že zatiaľ čo sa odosiela notifikácia s informáciou proces

automatizácie stále pokračuje. Synchrónne notifikácie sú často používané pre informačné oznámenia. Oznámenia nie sú spravidla len hlásenia chýb. Synchrónne notifikácie mávajú často charakter čisto informatívny. Informujú napríklad o aktuálnom stave procesu, o vykonanom celku alebo časti procesu, o počte spracovaných produktov. Naopak asynchrónny typ notifikácie môže byť použitý pri ukončení určitého celku procesu automatizácie alebo na samotnom konci automatizácie. Asynchrónna notifikácia sa totiž primárne zadáva medzi jednotlivými časťami automatizácie a zabezpečuje tak informovanie o vykonaní časti automatizácie s určitosťou, pretože proces sa do jej vykonávania dostane až po ukončení predchádzajúcej časti automatizácie.

Paradox automatizácie tkvie v tom, že čím efektívnejší je automatizovaný systém, tým rozhodujúcejší je ľudský faktor. Človek je menej zapojený do procesu systému, ale zároveň sa jeho zapojenie do procesu systému stáva kritickejším. V plne automatizovaných procesoch totižto zasahovanie človeka do systému prichádza v situácii, kedy sa vyskytne chyba v systéme alebo v jeho procese. Dôsledky tejto chyby sa budú znásobovať, pokým nedôjde k oprave alebo zastavení automatizovaného systému. A práve v takejto situácii sa očakáva zasiahnutie ľudského faktora do procesu.

1.3 VYUŽITIE AUTOMATIZÁCIE

Boom, ktorý nastal v automatizácii procesov, najmä v posledných rokoch dvadsiateho prvého storočia, znamená rozšírenie automatizácie a digitalizácie do pravdepodobne všetkých oblastí priemyslu a mnohých oblastí služieb i finančníctva.

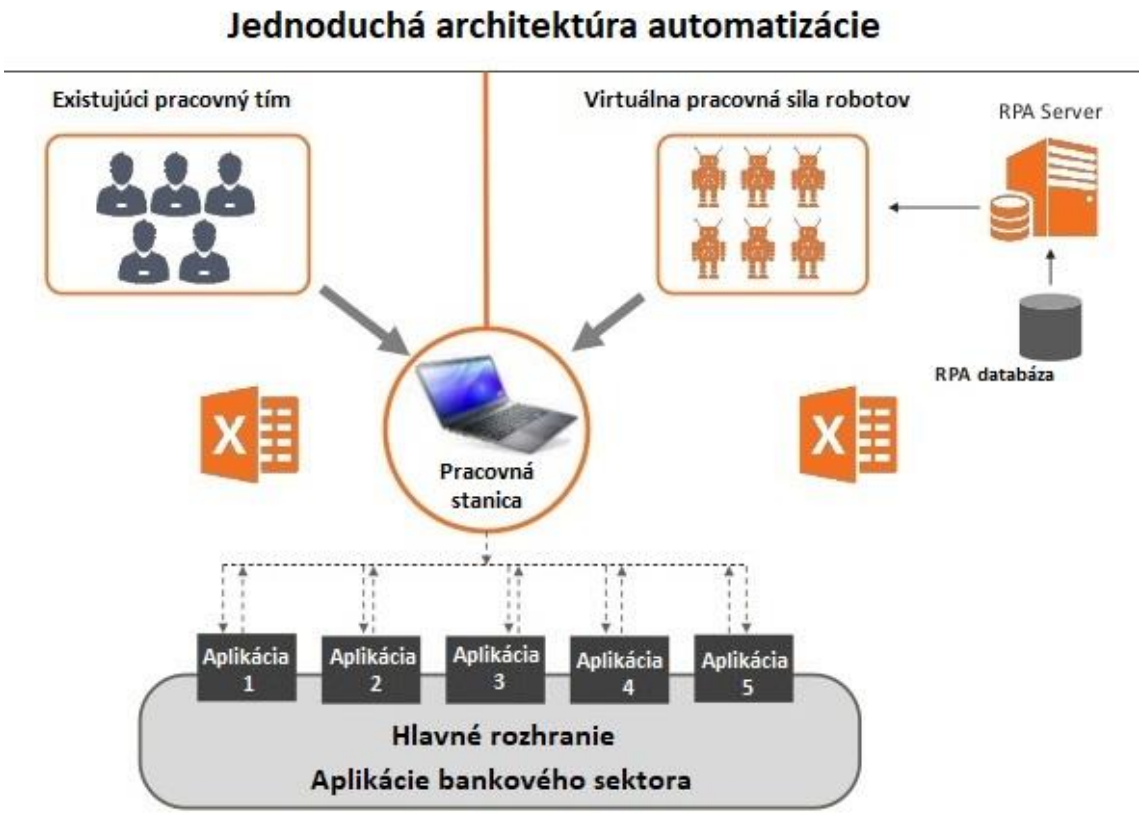
Kognitívna automatizácia

Kognitívna automatizácia je podmnožinou umelej inteligencie (Guhathakurta, 2018) umožňujúca kognitívne výpočty. Hlavným záujmom kognitívnej automatizácie je automatizácia administratívnych úloh a pracovných postupov, ktoré pozostávajú zo štruktúrovania neštruktúrovaných údajov (The Silicon Review, 2017).

Kognitívna automatizácia sa zameriava na replikáciu ľudských schopností, ktoré sa jej podľa Deloitte (2018) darí rýchlo a vo vysokej miere. Kognitívna automatizácia zahŕňa viaceré oblasti automatizácie, akou je spracovanie prirodzeného jazyka, výpočty v reálnom čase, algoritmy strojového učenia, rozsiahle analýzy údajov a učenie založené

na dôkazoch. Mnohé spoločnosti využili kognitívnu automatizáciu na spracovávanie dokumentov, extrakciu, syntézu a referovanie dát, manažment zákaziek, prijímanie a vyhľadávanie zákazníkov a zamestnancov, emailovú komunikáciu, či na nahradenie manuálnych činností a verifikáciu.

Na obrázku 1 môžeme vidieť jednoduchú architektúru automatizácie. Pracovný tím komunikuje s pracovnou stanicou, ktorá komunikuje s robotmi. Roboty bežia na RPA serveri a RPA databáze. Pracovná stanica ďalej pracuje s rôznym počtom aplikácií.



[1]

Obrázok 1 Jednoduchá architektúra automatizácie pre riešenia v bankovom a finančnom sektore.

Automatizácia priemyslu

Priemyselná automatizácia sa zaoberá predovšetkým automatizáciou procesov výroby, kontroly kvality a manipulácie s materiálom. Automatizácia priemyslu by mala nahradiť predovšetkým rozhodovanie človeka vo výrobnej sfére a manuálne rutinné

[1] Zdroj: Obrázok 1: https://www.indrastra.com/2018/09/Cognitive-Automation-004-09-2018-0006.html

úlohy počas procesu výroby, či v iných odvetviach priemyslu. Jeden z trendov v priemysle je využitie kognitívnej automatizácie na automatickú kontrolu kvality, či už po vizuálnej, alebo inej stránke. Automatizácia umožňuje firmám optimalizovať prevádzku priemyselných procesov. V súvislosti z automatizáciou sa v priemysle hovorí o štvrtej priemyselnej revolúcii.

„Slovensko patrí medzi svetových lídrov automatizácie. Podľa údajov Medzinárodnej federácie robotiky (IFR) za rok 2016 tu na desaťtisíc zamestnancov vo výrobnom sektore pripadalo 135 priemyselných robotov,“ (Vlachynský, 2018). Slovensko predbehlo krajiny ako Francúzsko, Švajčiarsko, Česko, Spojené kráľovstvo a Čínu. Celosvetový priemer robotov na desaťtisíc zamestnancov je len 74. V roku 2016 bolo na Slovensku nainštalovaných historicky najviac priemyselných robotov, viac ako 1 700 (Vlachynský, 2018).

Automatizácia dobývacích aktivít

V dobývacích aktivitách predstavuje automatizácia odstránenie ľudskej práce z procesu ťažby (Trounson, 2008). V súčasnosti je ťažobný priemysel stále v procese automatizácie. Avšak predovšetkým v rozvojových krajinách, ako sú najmä africké krajiny, ktoré sú známe rozsiahlou banskou činnosťou, je proces automatizácie v ťažobnej činnosti veľmi pomalý. Dôvodom sú nízke mzdové náklady, ktoré spôsobujú, že ťažobné firmy nie sú motivované k zefektívneniu procesov, teda ani k automatizácii.

Automatizácia podnikových procesov

Automatizácia podnikových procesov je automatizácia komplexných obchodných procesov (Gartner.com, 2019). Pozostáva z integrácie aplikácií, reštrukturalizácie pracovných zdrojov a využívania softvérových aplikácií v celej organizácii. Môže pomôcť zjednodušiť podnikanie, dosiahnuť digitálnu transformáciu, zvýšiť kvalitu služieb, zlepšiť poskytovanie služieb alebo znížiť náklady. Automatizácia podnikových procesov nájde uplatnenie v mnohých oblastiach podnikania vrátane marketingu, predaja a administratívy (TEBilion, 2018). V automatizácii podnikových procesov sa častokrát využíva umelá inteligencia.

Automatizácia logistiky

Automatizácia logistiky je aplikácia softvérových a hardvérových riešení na zlepšenie efektívnosti logistických činností a procesov. Predovšetkým sa jedná o činnosti v rámci skladu alebo distribučného centra, pričom širšie úlohy vykonávajú systémy riadenia dodávateľského reťazca a systémy plánovania podnikových zdrojov. Automatizácia nájde v logistických procesoch uplatnenie pri ukladaní a vyhľadávaní tovarov pomocou automatizovaných žeriavov, vozíkov, v systémoch triedenia, pri balení, paletovaní či rozbaľovaní materiálu alebo pri vážení prostredníctvom automatizovaných kontrolných váh. Na automatizáciu je v logistike priestor aj v administratíve logistiky, v skladovom hospodárstve a ostatných softvérových činnostiach.

Automatizácia maloobchodných služieb

V reštauračných zariadeniach sa automatizácia uplatňuje pri objednávaní tovarov a služieb. Spoločnosť McDonald's už pred rokmi zaviedla systém objednávania a platenia prostredníctvom inteligentných telefónov vo viacerých svojich prevádzkach, čím znížila potrebu veľkého množstva zamestnancov obsluhujúcich pokladňu (Jackson, 2011). Prevádzku plne automatizovaných kaviarní zasa spustila Texaská Univerzita v Austine (Hill, 2012). Mnohé kaviarne a reštaurácie už využívajú aplikácie pre inteligentné telefóny a tablety, aby zákazníci mohli objednávať a platiť za služby prostredníctvom svojich elektronických zariadení (Paskin, 2011). Niektoré reštaurácie majú automatizované dodávanie potravín zákazníkom pomocou automatizovaného pásu. V reštauračných zariadeniach možno registrovať aj používanie robotov namiesto personálu (Serkan, 2010).

V supermarketoch sa v posledných rokoch rýchlo ujíma systém samoobslužných pokladní. Tento spôsob obsluhy zákazníkov znižuje množstvo pracovnej sily na počet obslúžených zákazníkov. Ako automatizácia by sa dalo vnímať aj on-line nakupovanie prostredníctvom internetových obchodov a iných internetových spoločností poskytujúcich predaj. Súčet v pokladnici, platba a transakčný prevod sú totižto vykonávané pod určitým automatizovaným systémom.

Automatizácia verejných služieb

Verejný sektor služieb poskytovaných prevažne štátnymi zdrojmi je rozsiahla množina služieb, v ktorých sa nájde široký priestor na automatizáciu. V odpadovom hospodárstve to môžu byť napríklad automatizované vozidlá na zber odpadu alebo automatizácia zberu odpadu. Vo viacerých mestách funguje poloautomatizovaný zber odpadu. Znamená to, že v kontajneroch sú senzory snímajúce naplnenie, ktoré hlásia, kedy sa kontajner naplní a po zaznamenaní naplnenia, môže prísť vozidlo a vypratať kontajner. Tieto kontajnery sú polopodzemné a ich kapacita je tak omnoho väčšia ako pri klasických nadzemných kontajneroch. Nie je teda potrebné, aby ku kontajnerom chodilo vozidlo v pravidelných intervaloch pre niekedy zbytočne poloprázdne kontajnery. Alebo naopak, ak sú kontajnery príliš skoro naplnené, vozidlo môže prísť hneď ako to senzor zosníma. Správa komunálnych služieb je takto uľahčená, vozidlá nejazdia zbytočne, ale cielene, čo pomáha k celkovému manažovaniu chodu spoločnosti. Poloautomatizovaný spôsob zberu odpadu funguje aj v meste Nitra. Riaditeľ spoločnosti Nitrianske komunálne služby Ladislav Peniaško sa pre portál touchit.sk vyjadril, že vďaka dátam získaným z automatizovaných systémov môžu lepšie riadiť odpad v meste (NitraDen.sk, 2018).

Automatizácia domácností

Automatizácia v domácnostiach je často spojená s výrazmi inteligentná domácnosť alebo inteligentný dom. Sú rôzne procesy a činnosti, ktoré môžu byť v našich domácnostiach zautomatizované. Jedná sa napríklad o vykurovací systém, ovládanie vetrania či klimatizácie, riadenie osvetlenia, umelého i prirodzeného pomocou inteligentných žalúzií a roliet. Ďalej to môže byť obsluha elektrických spotrebičov. V oblasti bezpečnosti sú tu umiestňované detektory dymu, kvality ovzdušia či kamerový systém. Všetky tieto prvky spadajú do oblasti automatizovaných zariadení a činností a ich prínos má byť v uľahčení ľudských činností, zlepšení kvality života a zvýšení bezpečnosti.

1.4 CHATOPS AKO NÁSTROJ AUTOMATIZÁCIE PRACOVNÝCH POSTUPOV

ChatOps je nová operatívna paradigma – pracovný model, ktorý je už dnes na pozadí aplikovaný v bežných chatroomoch (četových miestnostiach) (StackStorm, 2019).

ChatOps je použitie četových klientov, ChatBotov (četbotov) a komunikačných nástrojov v reálnom čase na uľahčenie komunikácie týkajúcej sa vývoja softvéru a vykonávania vývoja tohto softvéru a zadaných úloh (Rouse, 2016).

Samotné slovo ChatOps sa skladá z dvoch častí, *„Chat“* a *„Ops“*. Anglické slovo *„Chat“* v pôvodnom význame krátkeho rozhovoru a neskôr v prenesenom význame komunikačného internetového kanálu, četu. *„Ops“* vzniklo skrátením anglického slova *Operation,* teda prevádzka, obsluha. ChatOps teda musíme chápať ako aplikáciu prevádzky, vykonávania niečoho, v rámci četovej miestnosti určitej komunikačnej aplikácie. Na strane komunikačnej aplikácie to môže byť aplikácia ako Skype, Webex Teams, Slack, Messenger, atď. Na strane vykonávania určitých funkcionalít alebo automatizovaných činností a úloh to môže byť napríklad nastavenie serveru, spustenie naprogramovaného kódu, alebo dopyt do databázy, pričom tieto úkony môžu nastať na rôznej platforme, či v rôznom automatizovanom skripte.

Existuje mnoho spôsobov, ako nasadiť prostredie ChatOps. Napríklad to môže byť notifikačný systém, ktorý odošle upozornenie o tom, že došlo k incidentu a v odpovedi bude klient četu používať plugin alebo vlastný skript na vykonanie vopred naprogramovaného príkazu (Rouse, 2016).

ChatOps je inovatívny model kooperácie spájajúci ľudí, nástroje, procesy a automatizáciu do jedného technologického postupu. Spôsob, akým je prevádzkovaný, uľahčuje a zefektívňuje komunikáciu v rámci tímov, automatizuje prevádzkové úlohy a prináša tak pozitívny dopad na efektivitu a produktivitu práce.

1.4.1 Vývoj a vznik riešení ChatOps

ChatOps má svoj pôvod vo vývojárskej platforme GitHub, ktorá podľa vlastných zdrojov hostí viac ako 31 miliónov vývojárov po celom svete (GitHub, 2019). GitHub je webová hostingová služba pre riadenie verziovania softvéru prostredníctvom verzovacieho softvéru Git. GitHub takto spája širokú škálu vývojárskych pracovníkov, ktorý nielen využívajú verzovaciu platformu ale aj zdieľajú svoje riešenia, komunikujú, riešia vzniknuté problémy pri vývoji softvéru a všetkými týmito aktivitami vytvárajú komunitu vývojárov, ktorá má podstatný význam pre vývoj softvéru. V tejto komunite sa rozšíril pojem ChatOps a platforma GitHub riešeniam ChatOps ponúka podstatný priestor.

1.4.2 Význam riešení ChatOps

Z hľadiska významu riešení ChatOps pre spoločnosť ako takú je dôležitý ich prínos do automatizácie rôznych softvérových riešení s možným dopadom na hardvérové nastavenia. Význam riešení ChatOps je preto vo vývoji softvéru, v podnikaní, komunikácii, či automatizácii, ktorou je ChatOps súčasťou.

ChatOps vznikol ako logický následok pokroku zaznamenaného v automatizácii za posledné roky a hľadania riešení na zjednodušenie opakovaných, čas vyžadujúcich si činností v rámci sektoru informačných technológií. Pre lepšie pochopenie uvedieme príklad softvérového riešenia s dopadom na hardvér: Ak máme nastavovať registrovanie zariadenia výpočtovej techniky do databázy manuálne, prostredníctvom prepracovaného GUI systému určitej spoločnosti, musíme spraviť niekoľko klikov, čo nám v konečnom výsledku zaberie určitý čas. Môže sa jednať napríklad o 10 minút. Ak máme takýchto zariadení registrovať 500 v rámci väčšej spoločnosti zaberie nám to 5 000 minút, čo predstavuje viac ako 83 hodín. Pri uvažovaní v rámci slovenského hospodárstva predstavuje 83 hodín približne polovicu času pracovného mesiaca zamestnanca. Toto všetko musí vykonať určitý zamestnanec tejto spoločnosti. Ak však zautomatizujeme registráciu týchto zariadení v rámci riešenia ChatOps a prinesie nám to v konečnom riešení pre 1 zariadenie 30 sekúnd komunikácie zamestnanca s ChatBotom ako súčasťou riešenia ChatOps, skrátime dobu vykonávania procesu človekom o 95% času predošlého riešenia.

Toto je potenciál reálneho prínosu ChatOps riešení do sektora infraštruktúry a vývoja informačných technológií. Je to skrátenie časovej náročnosti na vykonávané činnosti pomocou zautomatizovania úkonov. Lepšia komunikácia a spolupráca v rámci vývojárskych tímov, automatizovanie manuálnych činností, viditeľnosť a transparentnosť vývoja, nasadenia a diagnostiky v rámci softvérovej produkcie sú pozitívne dopady ChatOps v oblasti informačných technológii (Eric Sigler, 2014).

1.4.4 ChatOps ako rozšírenie DevOps

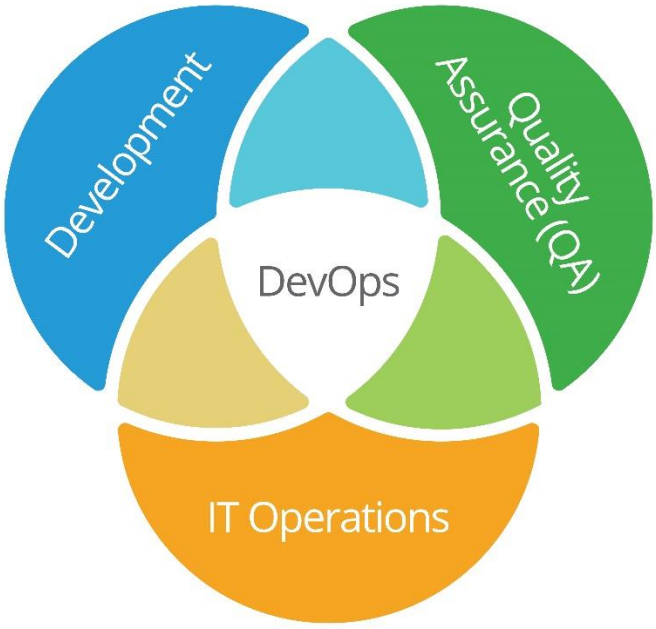
„DevOps je súbor postupov, ktoré kladú dôraz na spoluprácu a komunikáciu softvérových inžinierov a operácií v rámci IT a infraštruktúry s cieľom skrátiť čas na uvedenie produktu na trh. DevOps je zameranie sa hlavne na rýchle nasadenie funkcií

do produkcie, detekciu a nápravu vyskytnutých problémov bez narušenia ostatných služieb," (Zyane, 2017).

Slovo DevOps sa skladá z dvoch skratiek anglických slov a tými sú *„Dev"* a *„Ops"*. Skratka *„Dev"* predstavuje anglické slovo Development (vývoj) a skratka *„Ops"* je z anglického slova Operation (prevádzka). Toto slovné spojenie reprezentuje prístup pre celý hodnotový reťazec služby od vývoja až po prevádzku (BESTPRACTICE.SK, 2019). Techniky DevOps sa môžu nachádzať od zdrojovej kontroly až po testovanie v rámci agilného vývoja (The Agile Admin, 2019).

Použitie DevOps znamená kontinuálnu integráciu, kontinuálne dodanie a kontinuálne nasadenie softvéru. Práve pre takýto spôsob kontinuálnych procesov je ChatOps prínosom a použiteľných riešením.

Na obrázku 2 môžeme vidieť DevOps ako prienik 3 prvkov a to vývoja, IT operácií a zabezpečenia kvality.



[2]

Obrázok 2 DevOps ako prienik vývoja softvérového inžinierstva, prevádzky technológií a zaistenia kvality (QA)

[2] Zdroj: Obrázok 2: https://www.smartsheet.com/devops

1.4.5 Architektúra ChatOps

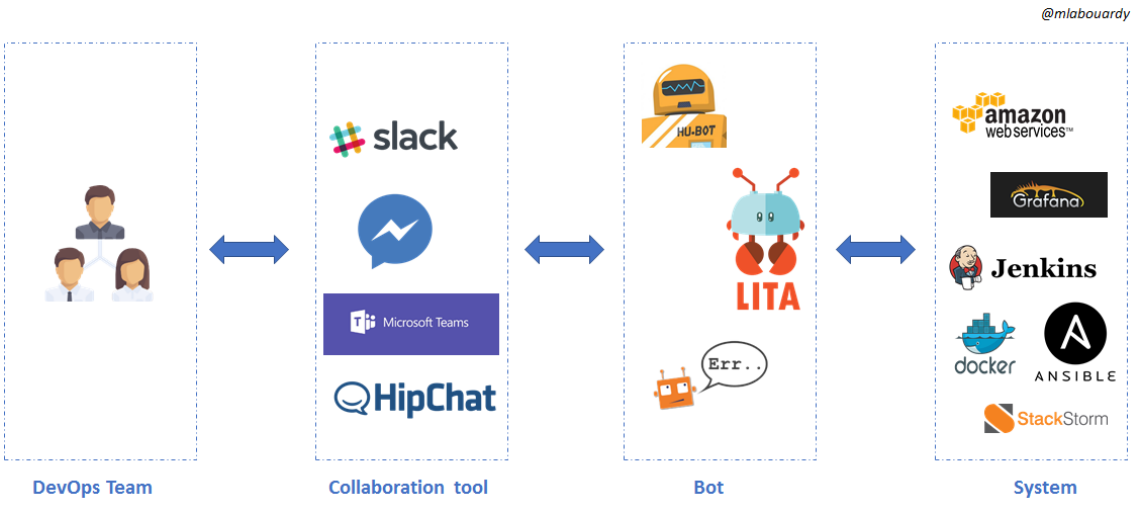
Architektúra riešení ChatOps sa skladá z troch hlavných komponentov.

V prvom rade sú to takzvané nástroje na spoluprácu. *„Sú to programy, četoví klienti, v ktorých sú prepojené zainteresované strany a tímy medzi sebou a tiež so systémami, na ktorých pracujú,“* (Zyane, 2017).

Ďalší nástroj v architektúre je ChatBot. ChatBot je robot, ktorý je jadrom metodológie ChatOps. *„Nachádza sa uprostred riešenia, medzi četovým programom a DevOps riešením, teda systémovou integráciou. Robot dostáva požiadavky od členov tímu v čete, na základe ktorých získava informácie z integrovaných systémov vykonaním sady príkazov (skriptov),“* (Zyane, 2017).

Tretia kľúčová časť ChatOps je systémová integrácia. Pod týmto pojmom si môžeme predstaviť konečnú platformu, ale aj spoluprácu event driven automatizačnej platformy a ďalších riešení, platforiem či aplikácii. *„Je to nástroj DevOps, ktorý umožňuje vyššiu produktivitu,“* (Zyane, 2017).

Na obrázku 3 môžeme vidieť možnú architektúru ChatOps riešenia s využitím platforiem, ktoré je možné zapojiť do ChatOps.



[3]

Obrázok 3 Architektúra ChatOps a možné využitie ChatBotov a rôznej infraštruktúry v riešeniach ChatOps

[3] Zdroj: Obrázok 3: https://chatbotsmagazine.com/how-chatops-can-help-you-devops-better-5-minutes-read-507438c156bf

1.4.6 Platformy podporujúce ChatOps riešenia

Vzhľadom na široké uplatnenie ChatOps v rámci vývoja softvéru informačných technológií sa rozširuje aj miera použitia ChatOps riešení a s tým narastá aj počet platforiem a produktov adaptovaných a vyvinutých na prácu s ChatOps.

V základnej trojprvkovej architektúre ChatOps máme četových klientov, ChatBotov a systémovú integráciu. Z hľadiska četových klientov tu máme niekoľko z nich, ktoré sú známe možnosťou aplikovania ChatOps:

- **Slack**: popredná četová platforma pre tímovú spoluprácu má viac ako 4 milióny denne aktívnych používateľov. Patrí medzi prvé platformy, ktoré integrovali robotov do svojho systému (Zyane, 2017).
- **HipChat**: vyvinutý spoločnosťou Atlassian slúžil na skupinové hovory, zdieľanie súborov, videohovory a zdieľanie obrazovky. V júli 2018 spoločnosť Atlassian oznámila zlúčenie a integráciu s vyššie spomenutou platformou Slack. Ukončenie celého procesu integrácie je oznámené spoločnosťou Atlassian na pätnásteho marca 2019 (Atlassian, 2018).
- **Facebook Messenger:** všeobecne známy pod pomenovaním Messenger, je aplikácia a platforma pôvodne vyvinutá ako Facebook Chat v roku 2008, neskôr premenovaná. V roku 2011 spoločnosť vyvinula samostatné aplikácie pre operačné systémy Android a iOS používané na smartfónoch (Stenovec, 2014).
- **Webex Teams:** je aplikácia pre nepretržitú tímovú prácu, ktorá umožňuje videohovory, skupinové správy alebo zdieľanie súborov (Cisco, 2019). Aplikácia je vyvinutá spoločnosťou Cisco, donedávna bola pomenovaná Cisco Spark. Aplikácii Webex Teams sa podrobne venujeme v podkapitole s názvom *Webex Teams*.

Ďalší kľúčový prvok ChatOps riešení sú roboty zvané ChatBoty. Podrobne sa ChatBotom a ich platformám budeme venovať v podkapitole s názvom *ChatBot ako súčasť riešenia ChatOps*.

Tretia časť ChatOps je samotná systémová integrácia. ChatOps riešenia ponúkajú široké spektrum platforiem a systémov, do ktorých ich je možné integrovať. Existuje niekoľko oblastí integrácie ChatOps riešení. V oblasti firemného zaznamenávania je to

JIRA, OTRS alebo TeamForge, ďalej sú to verzovacie systémy ako GitHub, Bitbucket, GitLab. V oblasti infraštruktúry Packer, Swarm alebo Docker. Známe integrácie sú aj pre platformy a systémy Amazon, Jenkins, Ansible a veľmi populárna je event driven technológia StackStorm, ktorej sa podrobne budeme venovať v podkapitole *Platforma StackStorm*.

Uplatnenie ChatOps riešení je veľmi široké a vzhľadom na rastúci záujem o automatizáciu softvérových riešení a lepšiu kooperáciu v tíme budú možnosti využitia rôznych platforiem v blízkej budúcnosti naďalej rásť.

1.5 CHATBOT AKO SÚČASŤ RIEŠENIA CHATOPS

Súčasťou riešenia a architektúry ChatOps je ChatBot. Názov vznikol spojením anglického slova *„chat“*, vo význame krátkeho rozhovoru, neskôr v prenesenom význame internetového kanálu a slova *„bot“*, anglického ekvivalentu slova „robot“. ChatBot v riešeniach ChatOps je vykonáva a spracováva komunikáciu s používateľom v rámci četového programu, prijíma dáta (žiadosti, dopyty) od používateľa a naopak prijíma výstupné dáta z infraštruktúry ChatOps riešenia naspäť do četového programu, kde ich zobrazí používateľovi.

Prvý známy ChatBot bol vydaný v roku 1966. Joseph Wiezenbaum vytvoril počítačový program ELIZA, v ktorom použil svoj novovytvorený procesuálny jazyk SLIP (Symmetric List Processor) (Weizenbaum, 1966). ELIZA fungovala na princípe rozoberania viet na kľúčové slová a substituovania kľúčových slov, podľa ktorých hľadala predefinované frázy, na ktorých už mala predpripravené odpovede (ChatBot.org, 2019). Od vytvorenia programu ELIZA uplynulo v počítačovej vede dlhé obdobie a tak vzniklo mnoho ďalších ChatBotov, akými sú Mitsuku, Rose, Eviebot či Tutor. ChatBot je častokrát chápaný ako program, ktorý komunikuje s človekom prirodzeným ľudským jazykom. ChatBot však nemusí mať a nemá len takéto využitie. ChatBot môže byť súčasťou ChatOps ako menšieho alebo rozsiahleho riešenia problematiky. ChatBot môže byť využitý na výpis dát z infraštruktúry zariadení, môže byť použitý pri zásahu do infraštruktúry či databázy, pri vytvorení nových záznamov v nich ale aj pri spúšťaní programov, aplikácií, spúšťaní konfigurácií, atď. Využitie ChatBota v rámci ChatOps prináša nové chápanie tohto nástroja v rámci počítačovej vedy.

1.5.1 ChatBoty využívané v ChatOps

Existuje niekoľko ChatBotov, ktoré je možné zapracovať do riešení ChatOps:

- **Hubot**: Bol vyvinutý spoločnosťou GitHub, Inc. na účely automatizácie firemnej chatroom. Neskôr z neho vznikol open-source napísaný v CoffeeScript na platforme Node.js. Hubot je štandardizovaný spôsob zdieľania skriptov medzi všetkými robotmi (The GitHub Blog, 2019).
- **Lita**: Je napísaný v jazyku Ruby. Je silne inšpirovaný Hubotom a má veľmi komplexný zoznam doplnkov, čo znamená, že môže byť integrovaný do mnohých četových platforiem ako sú Slack, Facebook Messenger a mnohé ďalšie (Zyane, 2017).
- **Cog**: Bol vyvinutý spoločnosťou Operable a používa zreťazenie príkazov (*commad-pipeline*) systému Unix. Môže byť aplikovaný na Twitter, GitHub alebo Cog chat.
- **Errbot**: Je napísaný v jazyku Python. Cieľom produktu je uľahčenie vytvárania vlastných pluginov, pre lepšiu aplikovateľnosť na rôzne funkcionality (Petrover a kol., 2019).
- **BotKit**: Popredný vývojársky nástroj na vytváranie četových robotov, aplikácií a vlastných integrácií pre významné četové platformy. Momentálne sa používa viac ako 10 000 BotKit robotov (botkit.ai, 2019). Používa Open Source knižnice a je napísaný v jazyku Node.js.

1.6 PLATFORMA STACKSTORM

V riešeniach ChatOps sú dôležité DevOps nástroje automatizácie procesov systémovej integrácie. Medzi takéto nástroje patria event driven automatizačná platforma. Ich hlavnou úlohou je vykonávanie a spravovanie automatizačných a operačných procesov. Event driven automatizačná platformy môžu byť v rámci ChatOps medzistupňom k širšej systémovej integrácií častokrát na API nástroje konečných služieb a platforiem.

StackStorm je event driven automatizačná platforma slúžiaca na integráciu a automatizáciu služieb a nástrojov. Spája existujúce infraštruktúry a aplikačné prostredie, čoho dôsledok je jednoduchšia automatizácia daného prostredia. StackStorm je výkonná open-source automatizačná platforma, ktorá spája aplikácie,

služby a pracovné postupy. Je ľahko rozšíriteľný, flexibilný a vyvíjaný pre implementáciu na DevOps a ChatOps riešenia (StackStorm, 2019).

Platforma StackStorm pomáha automatizovať bežné prevádzkové vzory a modely akými sú spúšťanie systémových zlyhaní, diagnostika na fyzických uzloch, identifikácia a overovanie zlyhania hardvéru ale aj kontinuálne nasadenie. Na všetky tieto prevádzkové vzory a modely umožňuje StackStorm aplikovať vlastnú architektúru opretú o „rule“ (pravidlá), „workflow“ (pracovné postupy) a „actions“ (akcie). (StackStorm).

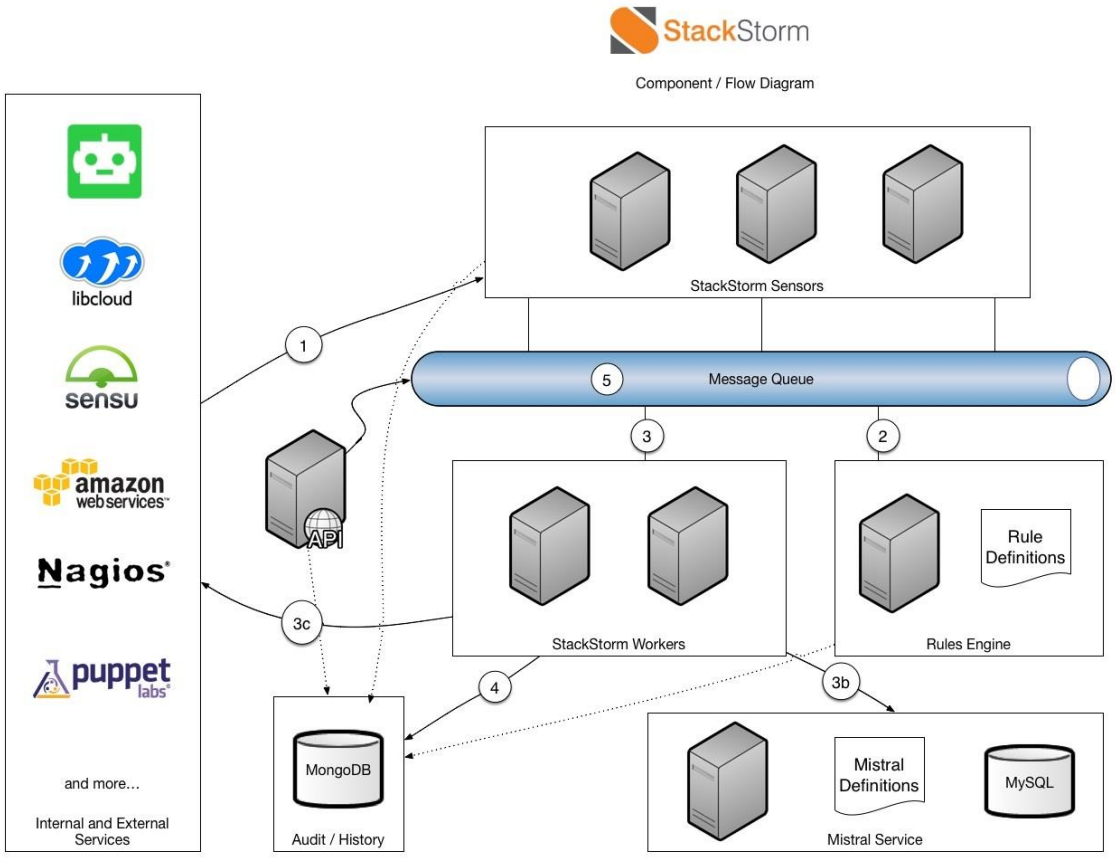
1.6.1 Architektúra platformy StackStorm

Ako sme už uviedli, architektúra platformy StackStorm je opretá o niekoľko prvkov, ktoré sa navzájom podieľajú na konečnom riešení. Všetky pravidlá, postupy a akcie sú tvorené ako kódy, skripty. Architektúru platformy StackStorm tvorí sedem hlavných komponentov:

- **Sensors** : Sú to senzory, alebo snímače pre vstupnú a výstupnú integráciu, ktoré prijímajú a sledujú udalosti. Keď nastane udalosť z externých systémov a je spracovaná senzorom, do systému sa vyšle Trigger.
- **Triggers**: Po slovensky spúšťače, sú reprezentácie externých udalostí, môžu byť generické (časovače) alebo integračné (JIRA aktualizácia). Nový typ Trigger môže byť definovaný zapísaním doplnku Sensor.
- **Actions**: Akcie sú odchádzajúce integrácie StackStormu. Sú tvorené skriptami, častokrát Python pluginmi a metadátovým súborom, ktorý spravidla obsahuje niekoľko riadkov. Akcie môžu byť vyvolané priamo používateľom prostredníctvom rozhrania CLI alebo API, alebo môžu byť použité a volané ako súčasť pravidiel a pracovných postupov.
- **Rules**: Sú to pravidlá, ktoré mapujú spúšťače do akcií, definujúc postup, podmienky prechodu a odovzdávanie dát.
- **Workflows**: V preklade pracovné postupy, spájajú akcie pričom určujú poradie, podmienky prechodu a odovzdávanie dát.
- **Packs**: Balíky zjednodušujú správu StackStormu a zdieľanie pripojiteľného obsahu. Používatelia môžu vytvárať vlastné balíky, zdieľať ich na platforme Github alebo odosielať na burzu StackStorm.

- **Audit trail**: Audítorská stopa zabezpečuje zaznamenávanie a uchovávanie vykonaných akcií s presnými podrobnosťami o spustení a výsledkoch (StackStorm, 2019).

Na obrázku 4 je znázornený diagram architektúry platformy StackStorm. Pod číslom 1 znázornený chod udalostí z externých služieb zaznamenávaných StackStorm senzormi. Tieto udalosti sú ďalej, v trase číslo 2, porovnávané v definícii pravidiel a spúšťačmi služby StackStorm sú spúšťané akcie. Spracované akcie sú ukladané vo fronte správ (číslo 3). Tu sa nám architektúra vetví na procesy pod číslami 3c a 3b. Proces pod pojmom 3c je komunikácia akcií platformy StackStorm ďalej na externé služby a proces 3b znázorňuje prístup na Mistral služby v MySQL databáze. Prístupová história a audítorská stopa sú ukladané v Mongo databáze.



4

Obrázok 4 Diagram architektúry platformy StackStorm

[4] Zdroj: https://docs.stackstorm.com/overview.html

1.6.2 Jazyk Python ako integrácia platformy StackStorm

Platforma StackStorm je schopná spracovávať udalosti z rôznych externých zdrojov. Jednotlivé komponenty architektúry sú napísané prevažne v jazyku Python, preto je nutná inštalácia tohto jazyka na operačný systém pred inštaláciou platformy StackStorm. Avšak používateľom vyvíjané akcie môžu byť napísané v rôznych programovacích jazykoch.

Jazyk Python je voľne šíriteľný programovací jazyk, ktorý umožňuje širokú integráciu v rôznych službách (python.org, 2019).

1.7 CISCO UNIFIED COMMUNICATION MANAGER

Služba Cisco Unified Communications Manager (CUCM) je platforma na riadenie podnikových hovorov a manažment relácií, ktorá spája ľudí kdekoľvek s použitím ľubovoľného zariadenia (Cisco, 2019). CUCM je jadrom collaboration platformy spoločnosti Cisco, ktorá umožňuje jednoduchú komunikáciu prostredníctvom hlasu, videa a správ na ľubovoľnom zariadení a na ľubovoľnom operačnom systéme. Zjednodušením komunikácie Cisco uľahčuje organizáciám verejného sektora riadiť efektivitu, zvyšovať výkonnosť a znižovať náklady. CUCM ponúka rozširovanie flexibility pomocou centralizovaného nástroja na správu povolení a podávania správ. Zlepšuje integráciu mobilných a stolových telefónov a zariadení Cisco Jabber a interoperabilitu medzi systémami a zariadeniami (Cisco, 2019).

Služba CUCM môže byť použitá ako konečná platforma v riešeniach ChatOps.

1.8 WEBEX TEAMS

ChatOps potrebuje na svoje fungovanie četoví program, na ktorý je možné aplikovať ChatBota a sfunkčniť tak celý proces ChatOps. Služba Webex Teams je takýmto četovým klientom.

Webex Teams je aplikácia pre nepretržitú tímovú prácu umožňujúca videohovory, skupinové správy, zdieľanie súborov a inovatívnu prácu s whiteboard (bielou tabuľou) (Cisco, 2019). Aplikácia Webex Teams bola vytvorená v roku 2018 zlúčením dvoch služieb spoločnosti Cisco a to Cisco Spark a Cisco Webex pod jednu službu s názvom Webex Teams (Chan, 2019). Webex Teams je aplikácia používaná naprieč mnohými spoločnosťami po celom svete na čet, videohovory, zdieľanie súborov a rôzne ďalšie

záležitosti uľahčujúce spoločnostiam správu, tímovú spoluprácu a vývoj. ChatBot, ktorý spadá do riešení uľahčujúcich spoluprácu, správu a vývoj, patrí medzi služby, ktoré je možné aplikovať v rámci aplikácie Webex Teams.

2 CIELE DIPLOMOVEJ PRÁCE

Cieľom záverečnej práce je implementácia kompletného ChatOps riešenia pre podporu konfigurácie a prevádzky Cisco Unified Communications Manager-a (CUCM) s využitím open source event driven automatizačnej platformy StackStorm, implementácie ChatBot modulu s integráciou na platformu Webex Teams.

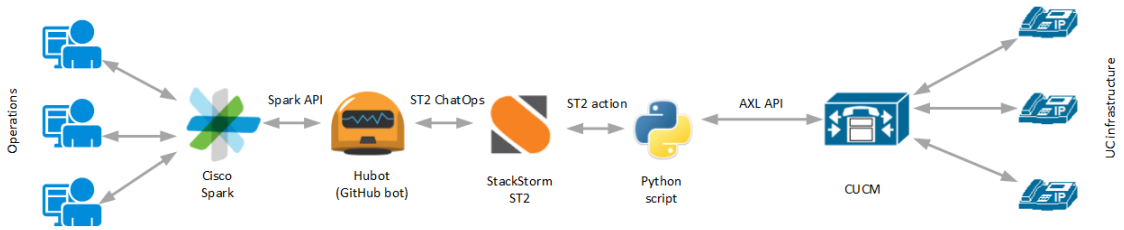
Podciele:

- naštudovať a informovať sa o inovatívnych riešeniach ChatOps a možnostiach práce s ChatOps,
- naštudovať si dokumentáciu event driven automatizačnej platformy StackStorm, implementáciu ChatBot modulov v rámci Webex Teams a podporu ChatOps riešení v rámci CUCM
- oboznámiť sa a zanalyzovať softvérové platformy StackStorm, Webex Teams a potencionálne použiteľných ChatBot modulov,
- zautomatizovať konkrétne konfiguračné scenáre pre CUCM, ktoré budú detailne špecifikované v priebehu prípravnej fázy,
- konzultovať postup projektu s členmi tímu a riešiť vzniknuté problémy,
- vyhodnotiť zautomatizované scenáre a reálne možnosti prevádzky ChatOps riešení.

3 TVORBA CHATOPS RIEŠENIA PRE PODPORU KONFIGURÁCIE A PREVÁDZKY CISCO UNIFIED COMMUNICATIONS MANAGER

Ako je spomenuté v predchádzajúcej kapitole *Ciele diplomovej práce* pre naše riešenie ChatOps boli zvolené platforma Webex Teams ako četový program, platforma StackStorm ako event driven automatizačný nástroj a platforma CUCM ako systémová integrácia projektu. V tejto kapitole opíšeme pracovné postupy pri tvorbe riešenia ChatOps pre podporu prevádzky CUCM.

Na obrázku 5 môžeme vidieť prvotný návrh architektúry nášho riešenia ChatOps pre podporu prevádzky CUCM. Naľavo máme používateľov, ktorí komunikujú s botom (plánované je použitie Hubota platformy GitHub) umiestneným v aplikácii Cisco Spark (dnes zjednotenej pod názvom Webex Teams). Bot ďalej komunikuje s platformou StackStorm, ktorý vykonávaním skriptov písaných v jazyku Python vyvolá pomocou API rozhrania AXL dopyt do CUCM databázy. CUCM následne vykoná dopyt priamo na zariadeniach v Cisco infraštruktúre.



Obrázok 4 Prvotný návrh architektúry riešenia ChatOps pre podporu prevádzky CUCM

3.1 IMPLEMENTÁCIA PLATFORMY STACKSTORM V CHATOPS

Rozhodnutie použiť event driven automatizačnú platformu StackStorm v rámci našej implementácie riešenia ChatOps bolo pre širokú škálu možností vývoja, ktoré táto platforma ponúka ale aj pre možnú integráciu v rámci systému CUCM pomocou už existujúcich knižníc v rámci programovacieho jazyka Python. Platforma StackStorm má prepracovanú architektúru a spôsob správy pomocou jednotlivých balíčkov akcií. Tieto balíčky si dokáže vytvárať vývojár sám, slúžia na skompletizovanie celých procesov do jedného administratívneho celku, ktoré môže neskôr uploadovať do GitHubu alebo StackStorm burzy či jednoducho prenášať v rámci svojich pracovných potrieb.

3.1.1 Inštalácia platformy StackStorm

Systémové nároky StackStorm platformy si vyžadujú prácu s unixovým operačným systémom ako je Ubuntu, RHEL alebo CentOS Linux. Platforma nie je podporovaná na iných distribúciách Linuxu. Podporovaná je iba 64-bitová architektúra. Pred inštaláciou samotnej platformy StackStorm je dôležité mať v operačnom systéme nainštalovaný programovací jazyk Python. Tabuľka 1 obsahuje zoznam podporovaných verzií systému Linux spolu s príkladmi Vagrant Boxes a Amazon AWS, ktoré vývojári platformy StackStorm používajú na testovanie.

Tabuľka 1 Zoznam podporovaných verzií systému Linux

Linux (64-bit)	Vagrant Box	Amazon AWS AMI
Ubuntu 14.04	bento/ubuntu-14.04	Ubuntu Server 14.04 LTS (HVM)
Ubuntu 16.04	bento/ubuntu-16.04	Ubuntu 16.04 LTS - Xenial (HVM)
RHEL 7 / CentOS 7	bento/centos-7.4	Red Hat Enterprise Linux (RHEL) 7.2 (HVM)
RHEL 6 / CentOS 6	bento/centos-6.7	Red Hat Enterprise Linux (RHEL) 6 (HVM)

Minimálne hardvérové nároky na vývoj a prevádzku platformy StackStorm sú zobrazené v tabuľke 2.

Tabuľka 2 Minimálne hardvérové požiadavky platformy StackStorm

Testovanie	Nasadenie (produkcia)
dvojjadrový procesor 2GB RAM 10GB úložisko	štvorjadrový procesor viac ako 16GB RAM 40GB úložisko

Štandardne služba StackStorm a s ňou súvisiace služby používajú nasledovné porty TCP:

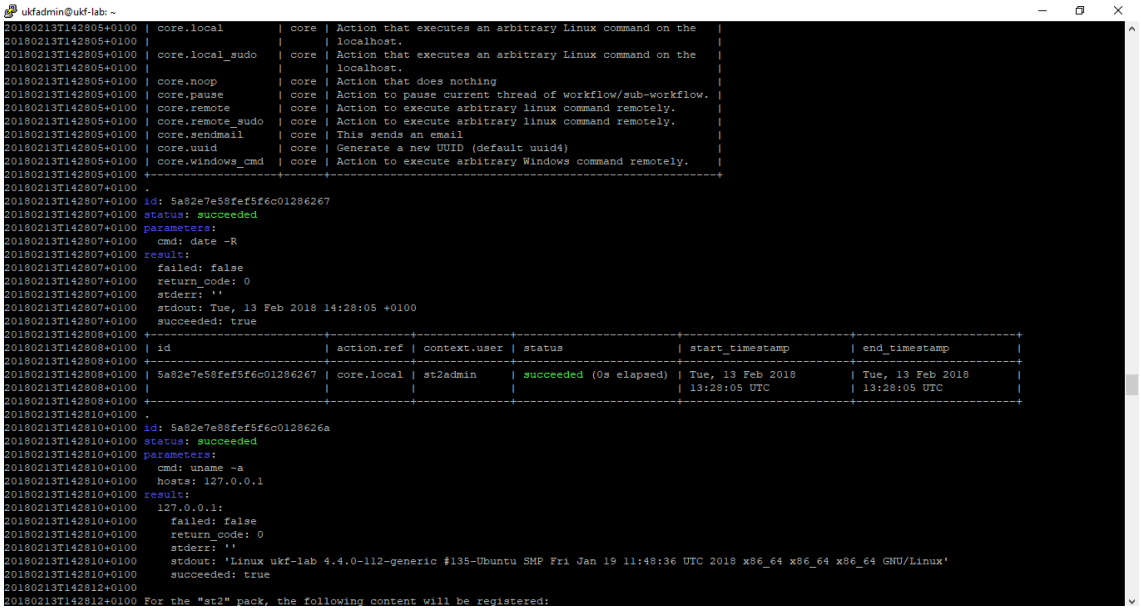
- nginx (80, 443)
- mongodb (27017)
- rabbitmq (4369, 5672, 25672)

- postgresql (5432)
- st2auth (9100)
- st2api (9101)
- st2stream (9102)

Pre správnu funkčnosť platformy StackStorm je dôležité, aby tieto porty nevyužívali iné služby a neblokovali tak platformu StackStorm pri správnej inštalácii a spustení.

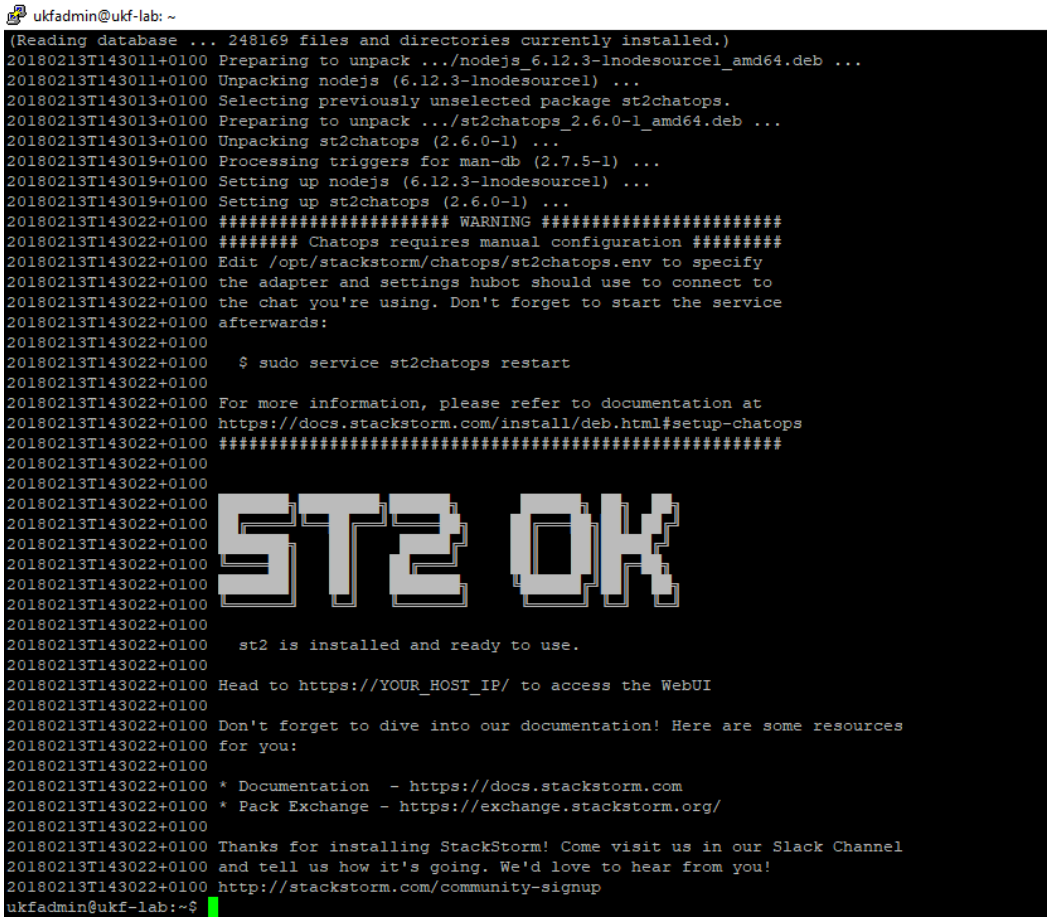
V našej práci sme využili prostredie operačného systému Ubuntu 16.04.3 LTS, ktorý sme nainštalovali na virtuálny prístroj v laboratóriu spoločnosti Dimension Data v Bratislave. Pre virtuálny prístroj boli nakonfigurované hardvérové nastavenia podľa systémových požiadaviek dokumentácie platformy StackStorm.

Na obrázkoch 6 a 7 môžeme vidieť priebeh inštalácie platformy StackStorm na virtuálny prístroj v laboratóriu počas vývoja nášho projektu. Inštalácia zaberie niekoľko minút času, je dôležité postupovať podľa pokynov v rámci dokumentácie produktu zverejnenej na webovej stránke platformy StackStorm.



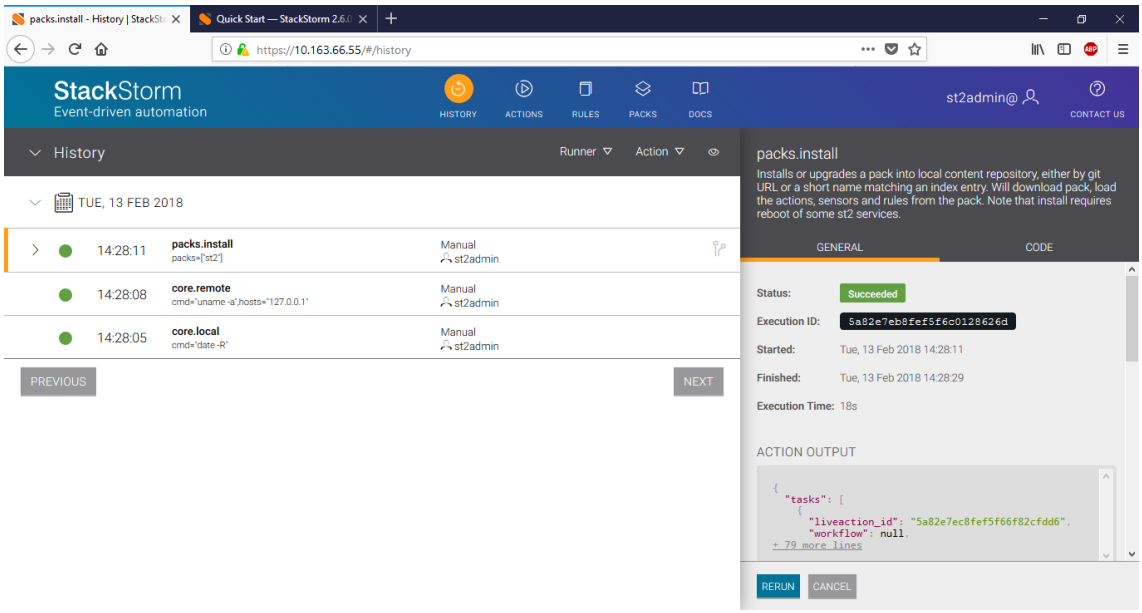
Obrázok 5 Priebeh inštalácie platformy StackStorm

Po dobehnutí inštalácie StackStorm vypíše oznamovaciu hlášku o úspešnej inštalácii „st2 is installed and ready to use“, čo v preklade znamená „st2 je nainštalovaný a pripravený na použitie“. Hláška je umiestená pod grafickým názvom „ST2 OK“ (obrázok 7).



Obrázok 6 Koniec inštalácie platformy StackStorm s hláškou o úspešnom ukončení

Po úspešnej inštalácii produktu máme možnosť otvoriť webový prehliadač, zadať IP adresu serveru, na ktorý sme nainštalovali platformu StackStorm, v našom prípade 10.163.66.55 a zobraziť GUI produktu. Na obrázku 8 máme možnosť vidieť toto GUI rozhranie platformy StackStorm po úspešnej inštalácii. V hornom strednom menu je zvolená položka „*History*“, po ktorej zakliknutí sa nám zobrazia všetky doteraz spustené akcie služby StackStorm. Toto zobrazenie je možné vďaka audítorskej stope, ktorú v sebe architektúra platformy má implementovanú. Ako vidíme na obrázku 8, počas úspešnej inštalácie sa už vykonali 3 akcie a tými sú *packs.install*, *core.remote* a *core.local*. Vykonanie týchto akcií je súčasťou inštalácie a nie je ich potrebné manuálne zadať.



Obrázok 7 Východiskový stav GUI rozhrania platformy StackStorm po úspešnej inštalácii

3.2 IMPLEMENTÁCIA RIEŠENIA CHATOPS NA PLATFORME CUCM

Pomocou platformy CUCM dokážeme vytvárať zásahy priamo do databázy Cisco zariadení. Dokážeme vytvoriť nové zariadenie, aktualizovať status zariadenia, aktualizovať zariadenie, vymazať zariadenie, vytvoriť asociácie a rôzne iné. Všetky zásahy v rámci databázy je možné vykonať pomocou CUCM platformy. Preto použitie CUCM platformy v rámci ChatOps riešenia nám umožní manipuláciu so zariadenia v databáze a vytvára nám tak široké možnosti aplikovateľnosti riešenia v rozličných prípadoch. Pre funkčné pracovanie s databázou potrebujeme správny WSDL súbor. Tento súbor typu XML je definíciou funkcií webovej služby a možností dopytov na tieto služby. V našom prípade sme WSDL súbor získali stiahnutím z podstránky spoločnosti Cisco, *developer.cisco.com*, ktorá je určená pre vývoj a prácu s AXL API na platforme CUCM. Z tejto stránky sme stiahli aj najnovšiu verziu schémy architektúry práce s CUCM databázou *axl-schema-11-5.*

3.2.1 Implementácia platformy StackStorm pre prácu s platformou CUCM

Pre prácu s platformou CUCM v rámci platformy StackStorm je nutné pracovať pomocou API. V rámci našej práci sme použili dva typy API pre rôzne typy zásahov a výpisov z databázy.

AXL API

AXL je skratka odvodená z anglického výrazu „*Administrative XML*“. Toto API je defaultné Cisco rozhranie slúžiace na automatizáciu volaní a dopytov do CUCM databázy. Je vyvinuté a spravované spoločnosťou Cisco pre prácu z CUCM. AXL API umožňuje robiť niekoľko typov zmien v databáze CUCM pomocou XML a rozhrania SOAP. Pre správnu funkcionalitu AXL API musíme implementovať potrebnú funkcionalitu z knižníc *requests, urllib3*, *lxml* a *ssl*. Taktiež je potrebné definovať odkaz na funkčný WSDL súbor, ktorý je typu XML. Definujeme aj pripojenie na klienta, jeho lokalizáciu a prístupové údaje. V nasledujúcej ukážke kódu 1 môžeme vidieť implementáciu dopytu pomocou AXL API pri použití v akcii platformy StackStorm.

```
from zeep import Client
from zeep.cache import SqliteCache
from zeep.transports import Transport
from zeep.exceptions import Fault
from zeep.plugins import HistoryPlugin
from requests import Session
from requests.auth import HTTPBasicAuth
from urllib3 import disable_warnings
from urllib3.exceptions import InsecureRequestWarning
from lxml import etree
from st2common.runners.base_action import Action
disable_warnings(InsecureRequestWarning)
username = [redacted]
password = [redacted]
host = [redacted]'
wsdl = 'file:///home/ukfadmin/axlsqltoolkit/schema/current/AXLAPI.wsdl'
location = 'https://{host}:8443/axl/'.format(host=host)
binding = "{http://www.cisco.com/AXLAPIService/}AXLAPIBinding"
session = Session()
session.verify = False
session.auth = HTTPBasicAuth(username, password)
transport = Transport(cache=SqliteCache(), session=session, timeout=20)
history = HistoryPlugin()
client = Client(wsdl=wsdl, transport=transport, plugins=[history])
service = client.create_service(binding, location)
```

Ukážka kódu 1 Implementácia dopytu pomocou AXL API

RIS API

RIS API je skratka odvodená z anglického výrazu „*Real-Time Information Port*“, ktorý tvorí názov tohto API. RIS API je vyvinuté spoločnosťou RisPort70. Poskytuje vyhľadávanie aktuálneho stavu pripojenia telefónov, zariadení a aplikácií pripojených k CUCM. Toto API slúži na dopyt po informáciách v reálnom čase v rámci databázy. V našej práci využívame tento typ API pre získavanie informácií o aktuálne registrovaných zariadeniach s ich aktuálnym statusom.

Pre správnu funkcionalitu RIS API musíme implementovať potrebné knižnice ako je *suds.client* a *ssl*, ďalej je potrebné definovať odkaz na funkčný WSDL súbor a samozrejme definovať pripojenie na klienta, jeho lokalizáciu a prístupové údaje. V nasledujúcej ukážke kódu 2 môžeme vidieť implementáciu dopytu pomocou RIS API pri použití v akcii platformy StackStorm.

```
from suds.client import Client
import ssl
import sys
from st2common.runners.base_action import Action
ssl._create_default_https_context = ssl._create_unverified_context
url = 'file:///home/ukfadmin/RISSservice.wsdl'
client = Client(url,
location='https://[redacted]/realtimeservice2/services/RISService',
                username=[redacted],
                password=[redacted])
stateInfo = None
```

Ukážka kódu 2 Implementácia dopytu pomocou RIS API

Po správnej implementácii kódu sa nám vracajú v oboch prípadoch použitia API korektné dáta. Formát je však v oboch prípadoch mierne odlišný, čo ale nespôsobuje problémy pri práci, pretože dokážeme korektne pristupovať k návratovým hodnotám.

3.3 DEFINOVANIE SCENÁROV POUŽITIA CHATOPS

V rámci cieľov našej práce sme si zadefinovali tri hlavné prípady použitia ChatOps riešenia v rámci prevádzky CUCM. Tieto scenáre sú základom pre prácu s databázou CUCM a majú demonštrovať možnosti nášho ChatOps riešenia pre podporu prevádzky Cisco CUCM. Tri hlavné scenáre sú: výpis aktuálne registrovaných a neregistrovaných

zariadení, pridanie nového zariadenia do databázy a update v databáze. Okrem hlavných scenárov sa nám môžu objaviť vedľajšie scenáre, ktoré doplnia celkový charakter ChatOps riešenia.

3.3.1 Výpis aktuálne registrovaných a neregistrovaných zariadení

Zariadenia evidované v databáze môžu mať rôzny status, ktorý definuje ich aktuálny stav. Tento status môže byť nedefinovaný (anglicky „*nullable*“) alebo definovaný, pričom sa do statusu definujú hodnoty „*Registered*“ a „*Unregistered*“. Statusy typu „*Registered*“ a „*Unregistered*“ predstavujú stav, v akom sa zariadenie nachádza. Status „*Registered*“ vyjadruje registrované zariadenie, teda zariadenie, ktoré bolo po pridaní do databázy aktivované a používané. Status „*Unregistered*“ vyjadruje zariadenie neregistrované, to je také zariadenie, ktoré síce bolo pridané do databázy, ale stále nebolo aktivované a používané. V tomto scenári nás zaujíma práve status typu „*Registered*“ a „*Unregistered*“.

Vzhľadom na to, že AXL API nám v tomto prípade nevracia požadované hodnoty, pracuje s databázou, ale nie s aktívnymi parametrami, funkciami alebo vlastnosťami používateľov a zariadení, teda dáta nie sú reálne v čase, musíme pre tento scenár použiť RIS API. Zadefinujeme RIS API volanie na našu databázu a do výstupu nášho kódu priradíme návratovú hodnotu z databázy ako výsledok. V nasledujúcej ukážke kódu môžeme vidieť prístup do databázy pomocou RIS API a následný dopyt na zariadenie so statusom „*Registered*“.

```
from suds.client import Client
import ssl
ssl._create_default_https_context = ssl._create_unverified_context
url = 'file:///home/ukfadmin/RISSservice.wsdl'
client = Client(url,
        location='https://[redacted]/realtimeservice2/services/RISService',
        username=[redacted],
        password=[redacted])
  stateInfo = None
  criteria = client.factory.create('CmSelectionCriteria')
item = client.factory.create('SelectItem')
#criteria.MaxReturnedItems = 500
criteria.MaxReturnedDevices = 500
criteria.DeviceClass = 'Phone'
```

```
criteria.Status = 'Registered'
criteria.SelectBy = 'Name'
result = client.service.selectCmDevice(stateInfo, criteria)
print result
```

Ukážka kódu 3 Dopyt do databázy prostredníctvom RIS API na registrované zariadenia

Kód v ukážke kódu 3 nám vráti zoznam zariadení v databáze so statusom „*Registered*“. V nasledujúcej ukážke 4 môžeme vidieť výsledok tohto volania. Pre rozsiahlosť výsledku uvádzame len časť z výsledku. V piatom riadku od spodu ukážky kódu 4 môžeme vidieť definíciu atribútu „*Status*“ ako „*Registered*“.

```
(CmDevice){
Name = "SEP0050600CB8E6"
IpAddress = "10.163.66.151"
DirNumber = "750-Registered"
DeviceClass = "Phone"
Model = 607
Product = 494
BoxProduct = 0
Httpd = "Yes"
RegistrationAttempts = 0
IsCtiControllable = True
LoginUserId = "gray"
Status = "Registered"
StatusReason = 0
PerfMonObject = 2
DChannel = 0
Description = "Meeting Room Grey"
```

Ukážka kódu 4 Časť z výstupu volania do CUCM na registrované zariadenia

Výstup z databázy nám zobrazuje všetky údaje o zariadeniach. Preto je dôležité pristupovať k týmto údajom a selektovať ich v rámci Python kódu, tak dokážeme zobraziť pre konečného používateľa iba pre neho dôležité dáta.

V ukážke kódu 3 je znázornený dopyt do databázy prostredníctvom skriptu v jazyku Python, tento dopyt však nie je ešte súčasťou platformy StackStorm. Pre sfunkčnenie takéhoto volania v rámci platformy StackStorm potrebujeme vytvoriť StackStorm akciu. StackStorm akciu vytvárame v predpripravenom balíčku (pack). Tento

balíček si vytvoríme sami. Balíček si vytvoríme v zložke „*Packs*“ zadaním nasledovných príkazov:

mkdir hello_st2

cd hello_st2

mkdir actions

mkdir rules

mkdir sensors

mkdir aliases

mkdir policies

touch pack.yaml

touch requirements.txt.

Potom vytvoríme metadátový súbor balíčku *pack.yaml.* V ukážke kódu 5 môžeme vidieť zadefinovanie metadátového súboru pre náš nový StackStorm balíček.

```
---
ref: ukf
name: ukf
description: cisco_spark-botkit-st2 Chatops solution Student version
keywords:
   - chatops
   - botkit
   - st2
   - webex teams
version: 0.1.0
author: Michal Dvorak, David Pavelka
email: dvorak159@gmail.com
```

Ukážka kódu 5 Vytvorenie metadátového súboru pre nový StackStorm balíček (pack)

Vytvorením balíčku dodržiavame predefinovanú architektúru platformy StackStorm. Takéto balíčky (*packs*) umožňujú jednoduchšiu správu pri práci s event driven platformou StackStorm. Odteraz všetky potrebné súbory, akcie, reťaze akcií (*chains*) môžeme vytvárať v rámci daného nami vytvoreného balíčku.

Samotné vytvorenie akcie zahŕňa vytvorenie dvoch súborov. Jeden súbor je tvorený kódom, ktorý je zodpovedný za vykonanie samotných príkazov. Je to hlavná časť akcie, ktorá je programovaná vývojárom, teda tá časť, ktorá obsahuje samotný kód, ktorý vykoná danú akciu. Tento súbor býva častokrát písaný v programovacom jazyku

Python, ale je možné vytvárať tieto súbory a ďalej vyvíjať potrebnú logiku akcie v rôznych programovacích jazykov. Druhý súbor je metadátový súbor typu YAML. Je to súbor, ktorý opisuje akciu a jej vstupy a taktiež je dôležitý pri vytváraní akcie a pri všetkých zmenách vykonaných v kóde, teda pri aktualizácii akcie.

Pre vytvorenie akcie musíme poznať, aké chceme mať vstupy do akcie a aké chceme mať výstupy z akcie. Vstupy do akcie sú zadefinované v metadátovom súbore. V ukážke kódu 6 môžeme vidieť metadátový súbor pre akciu na dopyt po registrovaných zariadeniach. Pri tejto akcii očakávame na vstupe iba jeden údaj. Ten môžeme vidieť pod názvom „*message*“ v ukážke kódu 6. Vstup je typu „*string*“, je to povinný údaj a má nultú pozíciu.

```
---
name: "getRegiAction06"
runner_type: "python-script"
description: "Get registered gadgets."
enabled: true
entry_point: "getRegiAction06.py"
parameters:
  message:
    type: "string"
    description: "Message to print."
    required: true
    position: 0
```

Ukážka kódu 6 Metadátový súbor pre dopyt na registrované zariadenia.

Vstupy nám prichádzajú zo StackStorm triger a akcie sú spúšťané v reťazi akcií („*actionChain*“) ako StackStorm workflow. Údaje prichádzajú do trigeru odchytávaním pomocou StackStorm senzoru, ktorý čaká na adrese webhook „*https://10.163.66.55/webhook/cisco_spatk_chatops“* na príchod správy z bota. Bot je nasadený v četovej miestnosti aplikácie Webex Teams. Akonáhle napíšeme správu určenú pre bot, táto správa sa odošle na webhook adresu, kde je odchytená senzorom. Ďalej pokračuje proces podľa architektúry StackStorm platformy cez rule, triger až k akcií resp. k reťazi akcií, ktorá spúšťa jednotlivé akcie podľa toho, ako ich má zadefinované. Reťaz akcií je workflow nášho StackStorm projektu.

Samotná reťaz akcií je zadefinovaná vývojárom. Môže ale nemusí byť použitá v procese ChatOps riešenia. V našom prípade sú vo všetkých scenároch použité reťaze

akcií. Tento postup zavádzame z toho dôvodu, aby sme mohli jednoducho upravovať riešenie vzhľadom na to, že v rámci jedného scenára sa nám vykonáva viacero dopytov do CUCM platformy. Každá StackStorm akcia, ktorá je súčasťou reťazi akcií, je v našom prípade zodpovedná za vykonanie dopytu do CUCM platformy. Vytvorenie reťazi akcií nám zabezpečí vytvorenie dvoch súborov typu YAML, jeden z nich je metadátový, druhý popisuje samotnú postupnosť akcií v reťazi akcií.

V ukážke zdrojového kódu 7 môžeme vidieť metadátový súbor pre vytvorenie reťazi akcií na dopyt po registrovaných a neregistrovaných zariadeniach. V prvom riadku je definovaný názov reťazi, v druhom riadku je umiestnený popis reťazi akcií. Veľmi dôležitý údaj je *„entry_point*“, ktorý nám hovorí, aký súbor, v ktorom je definovaná postupnosť akcií, má byť spustený. Nižšie sú umiestnené parametre, v tomto prípade sa jedná o dva parametre *„test_word“* a *„room_id“.* Prvý zo spomenutých parametrov je zodpovedný za správny vstup z četovej miestnosti, musí sa jednať o reťazec *„Registered“* alebo *„Unregistered“.* Druhý zo spomenutých parametrov je zodpovedný za prenos identifikačného čísla četovej miestnosti.

```
---
        name: "chainPhone"
        description: "Simple Action Chain workflow"
        runner_type: "action-chain"
        entry_point: "chains/chainPhone.yaml"
        pack: ukf
        enabled: true
        parameters:
          test_word:
                type: "string"
                required: true
                description: "Registered/Unregistered"
           room_id:
                type: "string"
                required: true
                description: "Room ID for spark"
```

Ukážka kódu 7 Metadátový súbor pre vytvorenie actionChain

Druhý súbor potrebný pre vytvorenie reťazi akcií popisuje logiku postupnosti akcií. Môžeme ho vidieť na ukážke kódu 8.

```
---
  chain:
    -
      name: "printRegistered"
      ref: "default.getRegiAction06"
      parameters:
        message: "{{test_word}}"
      on-success: "printUnRegistered"
      on-failure: "error"
    -
      name: "printUnRegistered"
      ref: "default.getUnReg01"
      parameters:
        message: "{{test_word}}"
      on-success: "sendReportToSpark"
      on-failure: "error"
    -
      name: "sendReportToSpark"
      ref: "cisco_spark.send_message"
      parameters:
        text: "Here you go, I found this:
                    \n{{printUnRegistered.stdout}}{{printRegistered.stdout}}"
        roomId: "{{room_id}}"
      on-failure: "error"
    -
      name: "error"
      ref: "cisco_spark.send_message"
      parameters:
        text: "There are no devices of status {{test_word}}"
        roomId: "{{room_id}}"
  default: "printRegistered"
```

Ukážka kódu 8 Logika procesu actionChain

V prvom scenári vytvárame reťaz akcií, ktorá bude schopná vrátiť hodnoty z platformy z CUCM pre registrované aj neregistrované zariadenia. Preto ja naša reťaz akcií tvorená štyrmi akciami. Prvá akcia s názvom *„printRegistered“* je zodpovedná za dopyt do platformy CUCM pre získanie návratovej hodnoty obsahujúcej zoznam registrovaných zariadení. Druhá akcia s názvom *"printUnRegistered"* nám vracia z CUCM návratovú hodnotu obsahujúcu zoznam neregistrovaných zariadení. Ďalšia akcia

v poradí je akcia s názvom *"sendReportToSpark"*, ktorá je zodpovedná za odoslanie výstupných hodnôt z akcií späť do četovej miestnosti v aplikácii Webex Teams. Štvrtá akcia s názvom *„error“* je vykonaná v prípade chybných výsledkov. Hláška *„error“* identifikuje chybu a posiela chybovú hlášku ChatBotu.

Prvá akcia *„printRegistered“* sa skladá z metadátového súboru, ktorý je popísaný vyššie a znázornený v ukážke kódu 6. Logika akcie je naprogramovaná v súbore *„getRegiAction06.py“*. V tomto súbore sú inicializované a importované potrebné knižnice na prácu s RIS API a CUCM, nachádza sa tu autentifikácia pripojenia prostredníctvom RIS API do CUCM. Výsledok, ktorý sa nám vracia z databázy. je priradený premennej *„result“*. Tento výsledok obsahuje zoznam všetkých zariadení v databáze so statusom *„Registered“* a vracia sa nám z databázy ako rozsiahly reťazec typu Tuple. Ďalej je naprogramované spracovanie tohto výsledku. Vzhľadom na to, že sa nám vracia z databázy veľmi rozsiahly reťazec s rôznymi dátami, ktoré nie sú pre konečného používateľa dôležité, musíme tieto dáta odfiltrovať od nepotrebných atribútov a priradiť do výsledku akcie v takom jazyku, ktorý je čitateľný pre konečného používateľa.

Na ukážke kódu 9 sú znázornené časti naprogramovanej akcie pre dopyt na registrované zariadenia v databáze. Pre skrátenie ukážky z nej boli vynechané časti ako inicializácia a import knižníc, autentifikácia, deklarácia premenných a zvyšné priradenie do premenných, ktoré je obdobné predchádzajúcim priradeniam.

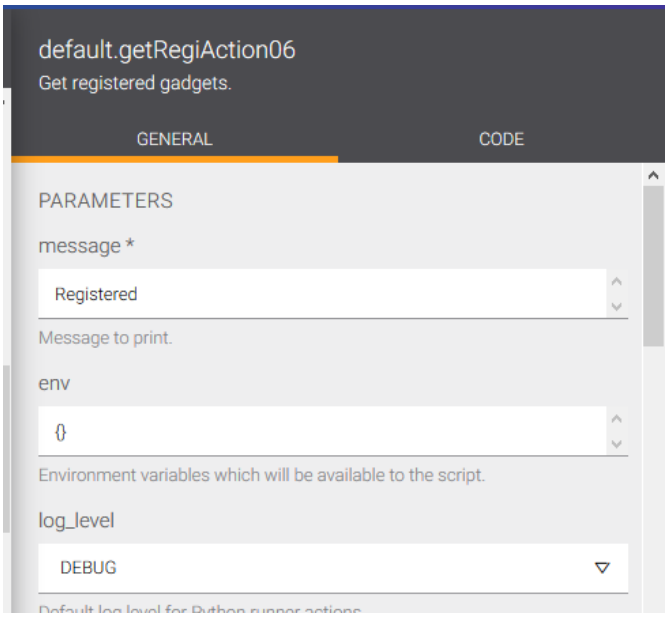
```
#inicializacia a import kniznic
#autentifikacia
stateInfo = None
criteria = client.factory.create('CmSelectionCriteria')
item = client.factory.create('SelectItem')
criteria.MaxReturnedDevices = 500
criteria.DeviceClass = 'Phone'
criteria.Status = 'Registered'
criteria.SelectBy = 'Name'
result = client.service.selectCmDevice(stateInfo, criteria)
class getRegiAction04(Action):
    def run(self, message):
    #deklaracia premennych
    #podmienka
     if message == 'Registered':
        test = result
```

```
        counter = 0
        counter2=1
        outResult = ''
        while counter<len(result[0].CmNodes.CmNode[2].CmDevices. CmDevice):
            print '-----------------------'
            print 'USER', counter2
            print 'Name:',
            result[0].CmNodes.CmNode[2].CmDevices.CmDevice[counter].Name
            nameRslt =
            result[0].CmNodes.CmNode[2].CmDevices.CmDevice[counter].Name
             print 'IP Address:',
            result[0].CmNodes.CmNode[2].CmDevices.CmDevice[counter].IpAddress
            ipAddressRslt =
            result[0].CmNodes.CmNode[2].CmDevices.CmDevice[counter].IpAddress
            print 'Description:',
            result[0].CmNodes.CmNode[2].CmDevices.CmDevice[counter].Description
            descRslt =
           result[0].CmNodes.CmNode[2].CmDevices.CmDevice[counter].Description
            print 'Login User Id:',
           result[0].CmNodes.CmNode[2].CmDevices.CmDevice[counter].LoginUserId
            loginUserIdRslt =
           #zvysne priradenie do premennych
            counter += 1
            counter2 += 1
        return (message, name, nameRslt, ipAddress, ipAddressRslt, desc, descRslt,
       loginUserId, loginUserIdRslt, line, lineRslt)
    return (False, message, outResult)
```

Ukážka kódu 9 Logická časť akcie na dopyt po registrované zariadenia v databáze

Pre registráciu našej novovytvorenej akcie musíme zadať príkaz „*st2 action create my_action_metadata.yaml*“, ktorým vytvoríme akciu a následne príkaz „*st2ctl reload --register-actions*“, ktorým obnovíme všetky akcie aktuálne zaregistrované v našej platforme StackStorm. Druhý z príkazov musíme zadať aj pri akomkoľvek zásahu do kódu akcie alebo do metadátového súboru pre obnovenie a načítanie nového kódu do platformy StackStorm.

Výsledok akcie „*getRegiAction06“* môžeme vidieť aj po spustený samostatnej akcie mimo reťazi akcií v GUI rozhraní nášho StackStormu. Po zadaní správneho vstupu, kľúčového slova „*Registered*“ sa nám vráti aktuálny zoznam zariadení so statusom „*Registered*“. Vstup do akcie môžeme vidieť na obrázku 9.



Obrázok 8 Vstup do akcie getRegiAction06.py

Po správnom zadaní kľúčového parametra sa nám vráti výsledok tak, ako to je naprogramované v akcii. Vráti sa nám zoznam zariadení so statusom „*Registered*“, odfiltrovaný od nepotrebných dát. Pre jednoduchú prácu s výsledným zoznamom sme naprogramovali jeho výpis aj vo výstupe (output), aj vo výsledku (result) akcie. Na obrázku 10 môžeme vidieť výsledok akcie. Vypisujeme päť údajov, ktoré pokladáme za dôležité vedieť pre používateľa. Sú to: meno (name), IP adresa (IP Address), popis (Description), prihlasovacie meno (Login User Id) a typ zariadenia (Device type). Na obrázku 10 môžeme vidieť, že sa nám vrátili dve zariadenia aktuálne v databáze so statusom „*Registered*“.

```
----------------------
USER 1
Name: SEP0050600CB8E6
IP Address: 10.163.66.185
Description: Meeting Room Grey
Login User Id: gray
Device type: 607
----------------------
USER 2
Name: SEP5006AB819BDC
IP Address: 10.163.66.131
Description: Marcel | Cisco 8845 | 207 | REC
Login User Id: mimrich
Device type: Cisco 8845
```

Obrázok 9 Výstup (output) akcie getRegiAction06.py

Druhá akcia, ktorej úlohou je dopyt do databázy pre vrátenie zoznamu všetkých neregistrovaných zariadení, je veľmi podobná prvej akcii. Pre vytvorenie tejto akcie sme tiež najskôr zadefinovali metadátový súbor. Ďalší súbor, ktorý sa zaoberá samotnou logikou akcie, je veľmi podobný kódu pre dopyt na registrované zariadenia. Rozdiel je v kritériách volania RIS API, kde namiesto „*criteria.Status = 'Registered'*“ je kritérium „*criteria.Status = 'UnRegistered'*“. Rozdielna je aj podmienka pred vstupom do spracovania výsledkov, kde namiesto podmienky „*if message == 'Registered'*“ je umiestnená podmienka „*if message == 'UnRegistered'*“.

Tretia akcia v reťazi akcií „*sendReportToSpark*“ je predefinovaná akcia platformy StackStorm, získaná z balíčka Cisco Spark, ktorý je možné doinštalovať do vlastnej nainštalovanej platformy StackStorm, pretože je verejne dostupný. Táto akcia zodpovedá za odoslanie výsledkov z predchádzajúcich akcií naspäť ChatBotu do aplikácie Webex Teams.

Akcia „*error*“ je popísaná vyššie.

Po vytvorení reťazi akcií so všetkými akciami, ktoré v reťazi potrebujeme použiť, môžeme reťaz akcií registrovať v StackStorm platforme. Použijeme nasledovné príkazy:

 # registrácia akcie

st2 action create opt/stackstorm/packs/default/actions/nazovAkcie.meta.yaml

 # kontrola dostupnosti akcie

st2 action list --pack=examples

 # spustenie akcie

st2 run examples.nazovAkcie.

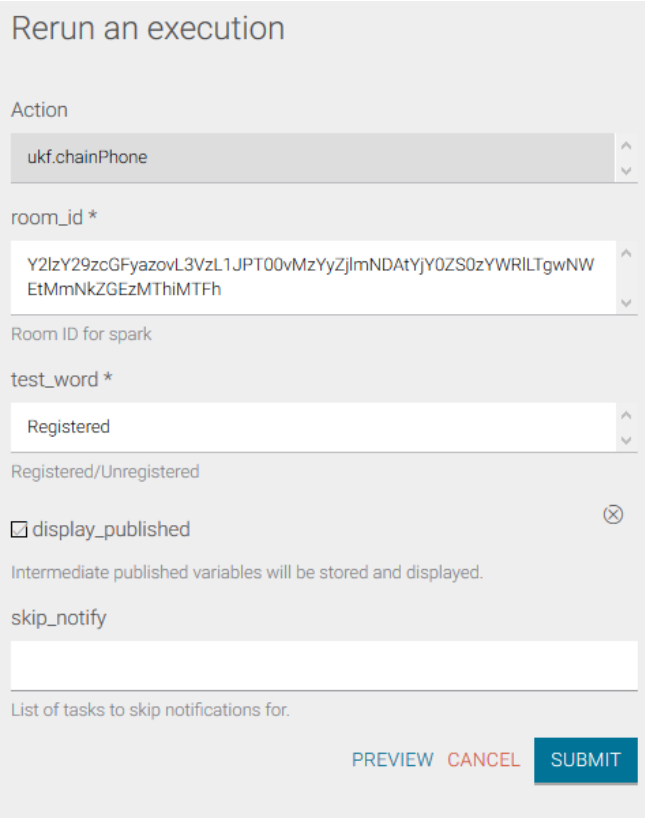
Po zadaní týchto príkazov môžeme s reťazou akcií pracovať. Pri ďalšej práci nesmieme zabudnúť, že po akejkoľvek zmene v metadátových súboroch reťazi akcií a samotných akcií v nej volaných musíme zadať príkaz „*st2 action update <action.ref> <action.metadata.file>“*, ktorým aktualizujeme reťaz akcií alebo príkaz „*st2ctl reload --register-all“*, ktorým aktualizujeme všetky zmeny v našej StackStorm platforme.

Našu reťaz akcií pre prvý scenár sme nazvali *chainPhone* a nachádza sa v našom balíčku *ukf*. Po spustení celej reťazi akcií sa nám dostaví potrebný výsledok. Na obrázku 11 môžeme vidieť priebeh spúšťania jednotlivých akcií v postupnosti zadefinovanej v reťazi akcií v GUI rozhraní StackStorm.

Obrázok 10 Priebeh akcie chainPhone v GUI rozhraní StackStorm

Vzhľadom na to, že naša reťaz akcií je spúšťaná z četového rozhovoru používateľa aplikácie Webex Teams a nášho ChatBota, vstup do akcií vyzerá odlišne ako pri spustení akcií priamo z GUI rozhrania. Na obrázku 12 môžeme vidieť vstup do akcie pre dopyt na zariadenia so statusom „*Registered*“ s jedným vyplneným parametrom vstupu navyše a tým je *room_id* parameter. Tento parameter obsahuje jedinečný identifikátor našej četovej miestnosti v aplikácii Webex Teams, ktorý je ukazovateľom, kam sa majú výsledky na konci reťazi akcií vrátiť.

Parameter *room_id* je vyplnený pomocou StackStorm triggeru, ktorý ho získava z ChatBota nainštalovaného na aplikácii Webex Teams. Celý tento proces je riešený v diplomovej práci Bc. Michala Dvořáka, ktorý participoval na čiastkovom riešení tohto projektu.

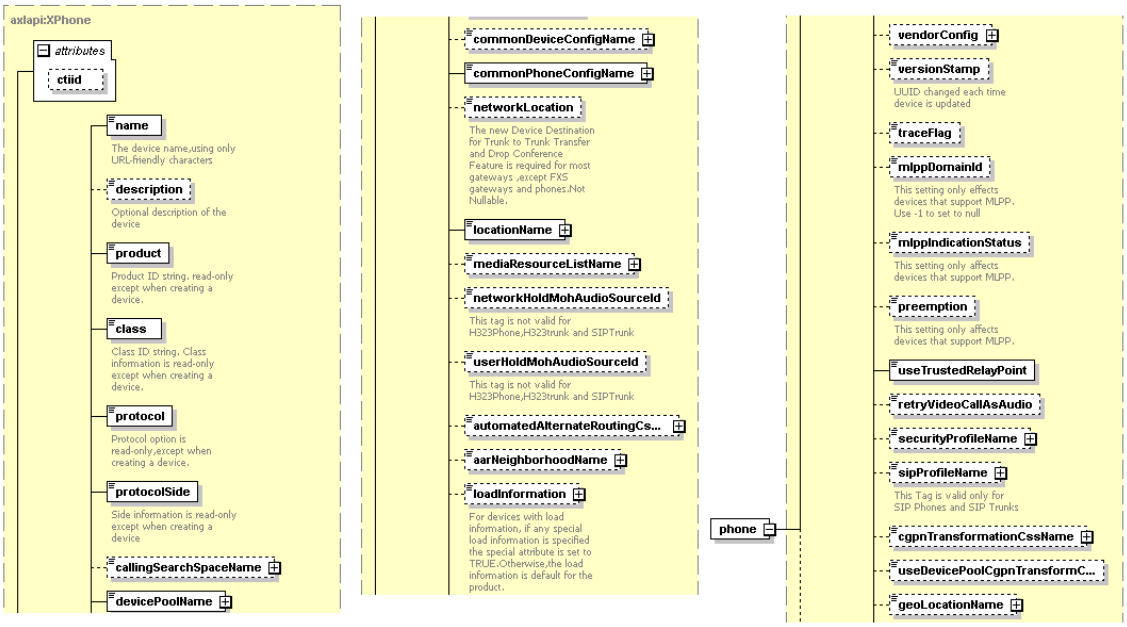


Obrázok 11 Vstup do akcie s vyplneným parametrom room_id

3.3.2 Pridanie nového zariadenia do databázy

V Cisco databáze sa nachádza niekoľko zariadení, s ktorými dokážeme pracovať. Na to, aby sme s nimi dokázali pracovať, musíme ich vedieť v tejto databáze vytvoriť, teda pridať ich do databázy. Akékoľvek nové zariadenie je potrebné evidovať a pridať do databázy. Tento proces riešime v našom druhom scenári, ktorý je popísaný v tejto podkapitole.

Po analýze celého scenára zisťujeme, že nepotrebujeme pracovať s dátami reálne v čase a na prácu nám bude postačovať klasické API od spoločnosti Cisco AXL API. Vďaka tomuto API vytvoríme potrebné pripojenie a získame potrebné dáta na prácu z druhým scenárom. Na to aby sme mohli pridať nové zariadenie do databázy, musíme poznať architektúru AXL API, ktorá je popísaná v jeho dokumentácii. Schéma AXL API dostupná na webovej stránke *„https://developer.cisco.com/docs/axl-schema-reference/“* nám ponúka širokú škálu metód pre prácu s databázou. V tomto scenári nás zaujíma metóda *„addPhone“*. Na obrázku 13 môžeme vidieť menšiu časť zo schémy metódy *„addPhone“*, pomocou ktorej dokážeme pridať nové zariadenie do databázy.



Obrázok 12 Časť schémy metódy addPhone

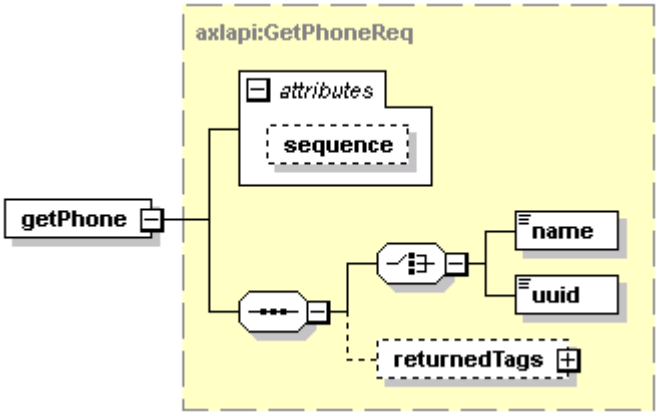
Ako je evidentné z obrázka, metóda „*addPhone*“ má široké spektrum atribútov, ktoré môžu, ale nemusia byť vyplnené. Rozdiel medzi povinnými (*mandatory*) atribútmi a nepovinnými atribútmi je taký, že v schéme metódy vedie k povinným atribútom plná čiara a k nepovinným atribútom vedie prerušovaná čiara. Povinné atribúty musia byť pri danom volaní použité, zatiaľ čo nepovinné môžu. Vzhľadom na rozsiahlosť možných atribútov, sme pre demonštráciu riešenia zvolili spôsob pridávania nového zariadenia iba s povinnými atribútmi. Pre pridanie zariadenia ale musíme tieto povinné atribúty získať. Preto prvá akcia druhého scenára má za úlohu získanie potrebných údajov z už existujúcich predefinovaných šablón, ktoré sa používajú pre pridanie nového zariadenia.

Metóda, ktorá nám dokáže získať potrebné dáta z týchto šablón sa nazýva „*getPhone*“. Vytvoríme teda dopyt do databázy prostredníctvom využitia metódy „*getPhone*“. Pri použití AXL API sa vyskytujú problémy so spracovaním dopytu implementujúceho doteraz používané knižnice. Pre lepšiu funkcionalitu akcií preto prechádzame s používania knižnice „*suds-jurko*“ a implementujeme nové knižnice „*ZEEP*“. V ukážke kódu 10 môžeme vidieť časť kódu, ktoré je zodpovedná za implementáciu správnych knižníc pre lepšiu prácu z AXL API.

```
from zeep import Client
from zeep.cache import SqliteCache
from zeep.transports import Transport
from zeep.exceptions import Fault
from zeep.plugins import HistoryPlugin
from requests import Session
from requests.auth import HTTPBasicAuth
from urllib3 import disable_warnings
from urllib3.exceptions import InsecureRequestWarning
from lxml import etree
```

Ukážka kódu 10 Implementácia potrebných knižníc pre lepšiu prácu z AXL API

Metóda „*getPhone*“ má jednoduchú architektúru. Povinný parameter je iba „*name*“ a nepovinný parameter „*returnedTags*“. Parameter „*returnedTags*“ obsahuje niekoľko atribútov. Ak nie je v dopyte spomenutý, dopyt nám vráti všetky dáta, ktoré o dopytovanom prvku obsahuje. Ak je vyplnený, vráti nám len tie dáta, ktoré v atribútoch definujeme. Na obrázku 14 môžeme vidieť schému metódy „*getPhone“* bez zobrazenia nepovinných atribútov nepovinného parametra „*returnedTags*“.



Obrázok 13 Schéma metódy getPhone

V našom prípade si sám používateľ v četovej miestnosti pri komunikácii s ChatBotom zadefinuje aká šablóna má byť vybraná, aby bolo nové zariadenie pridané do databázy pomocou tejto šablóny. Údaj obsahujúci meno používateľom vybranej šablóny máme na vstupe do akcie „*getPhoneZ*“, v ktorej budeme volať metódu „*getPhone*“. Pri volaní metódy „*getPhone*“ následne zvolíme ako parameter „*name*“ používateľom zadefinovanú šablónu a vyvoláme dopyt. Výsledok dopytu je celý zoznam údajov, ktoré o danej šablóne databáza uchováva. Na ukážke kódu 11 môžeme vidieť

časť výsledku dopytu metódy „*getPhone*“ do databázy. Žltou farbou sú vyznačené dáta, ktoré budeme ďalej spracovávať pre pridanie nového zariadenia. Pre rozsiahlosť údajov je v ukážke uvedený iba začiatok výsledku dopytu.

```
{   'return': {
     'phone': {
       'name': 'TMP_8845',
       'description': None,
       'product': 'Cisco 8845',
       'model': 'Cisco 8845',
       'class': 'Phone Template',
       'protocol': 'SIP',
       'protocolSide': 'User',
       'callingSearchSpaceName': {
          '_value_1': None,
          'uuid': None
       },
       'devicePoolName': {
          '_value_1': 'Default',
          'uuid': '{1B1B9EB6-7803-11D3-BDF0-00108302EAD1}'
       },
       'commonDeviceConfigName': {
          '_value_1': None,
          'uuid': None },
```

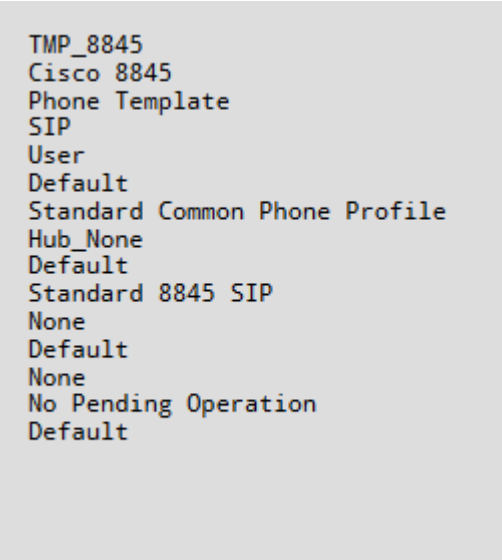
Ukážka kódu 11 Časť z výsledku dopytu metódy getPhone do databázy

Z výsledných dát vyberieme potrebné údaje, ktoré priradíme na výstup akcie „*getPhoneZ*“, aby sme ich mohli uložiť na vstup metódy „*addPhone*“ a ďalej mohli s nimi pracovať. V ukážke kódu 12 môžeme vidieť časť akcie getPhoneZ. V šiestom riadku je umiestnené volanie metódy „*getPhone*“, následne pokračuje spracovanie potrebných údajov a ich výpis do výsledku akcie.

```
class getPhoneZ(Action):
    def run(self, template):
      outResult = 'Getting specific template was not successfull'
      try:
        outResutlt  = 'Getting specific template was successful'
        resp = service.getPhone(name=template)
        print (resp['return'].phone.name)
        print(resp['return'].phone.product)
        print(resp['return'].phone['class'])
        print(resp['return'].phone.protocol)
```

Ukážka kódu 12 Časť kódu akcie getPhoneZ

Na obrázku 15 môžeme vidieť výstup z akcie „getPhoneZ“, ktorá nám získa potrebné dáta o šablóne. Spracované dáta ďalej vstupujú do akcie „*addPhone*“.



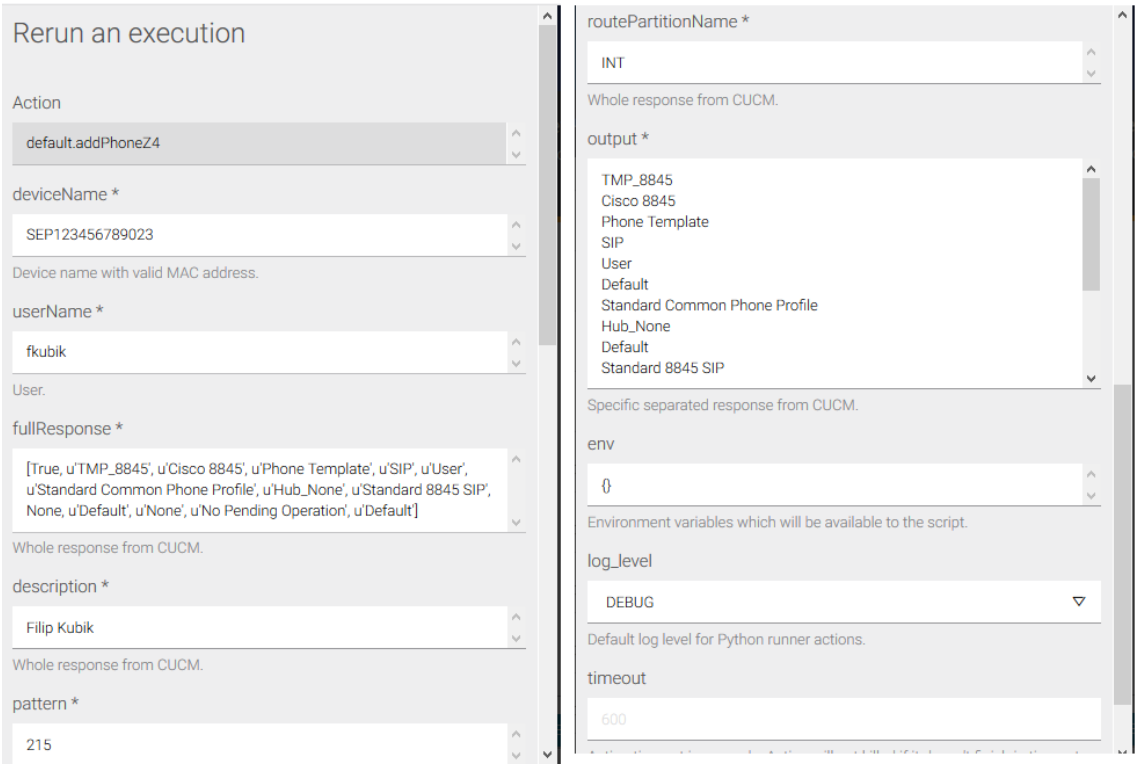
Obrázok 14 Výstup z akcie getPhoneZ obsahujúci potrebné údaje o šablóne

K samotnému zariadeniu potrebujeme pridať aj používateľa z databázy, aby sme pridali nové zariadenie pre už existujúceho používateľa. Preto pred spustením akcie *„addPhone“* vykonáme dopyt do databázy, aby sme získali potrebné dáta o používateľovi. V ChatBote nám používateľ zadá používateľské meno (*„userName“*), ktoré nám príde na vstupe do akcie *„getUserZ*“. Táto akcia má za úlohu získať potrebné dáta z databázy o používateľovi, ktorému chceme priradiť nové zariadenie. Akcia *„getUserZ*“ má v sebe naprogramovaný dopyt do databázy metódou *„getUser*“, pomocou ktorej získame potrebné dáta. V ukážke kódu 13 môžeme vidieť časť kódu akcie *„getUserZ*“, v ktorej je naprogramovaný dopyt do databázy pomocou metódy *„getUser*“. Metóda má zadefinované parametre *„returnedTags“*, aby sme v návratových hodnotách mali len pre nás dôležité údaje. Ďalej tieto údaje spracovávame a dáme ich do výsledku akcie, aby mohli byť použité na vstupe do akcie *„addPhone*“.

```
        class getUserZ(Action):
            def run(self, user):
              out2Result = 'Action did not get results'
              try:
                resp  = service.getUser(userid = user, returnedTags = {
                    'firstName': '',
                    'lastName': '',
                    'primaryExtension': {
                       'pattern': '',
                       'routePartitionName': ''
                    } ,
                     'associatedDevices': ''
                  })
                firstName = resp['return'].user.firstName
                lastName = resp['return'].user.lastName
                pattern = resp['return'].user.primaryExtension.pattern
                routePartitionName =
resp['return'].user.primaryExtension.routePartitionName
```

Ukážka kódu 12 Časť kódu akcie getUserZ

Po získaní všetkých potrebných údajov z predošlých akcií „*getPhoneZ*“ a „*getUserZ*“ môžeme pristúpiť k vytvoreniu akcie „*addPhone*“, pomocou ktorej pridáme nové zariadenie do databázy. Všetky potrebné dáta, ktoré sme získali, použijeme na vstupe do akcie „*addPhone*“. Na obrázku 16 môžeme vidieť spustenie akcie „*addPhone*“ v reťazi akcií s potrebnými dátami, ktoré boli získané v predchádzajúcich dvoch akciách reťazi akcií „*getPhoneZ*“ a „*getUserZ*“.



Obrázok 15 Spustenie akcie addPhone v reťazi akcií s potrebnými dátami

V kóde akcie „*addPhone*“ je zadefinovaný dopyt do databázy prostredníctvom AXL API. Do deklarovaných premenných rozdelíme údaje získané zo vstupu do akcie a tieto premenné vložíme na potrebné miesta kódu pri vykonaní dopytu do databázy pomocou metódy „*addPhone*“. V ukážke kódu 14 môžeme vidieť časť akcie „*addPhone*“. Z dôvodu veľkého rozsahu kódu sú zámerne vynechané oblasti priradenie údajov do premenných a zvyšná časť metódy, kde sú priradené údaje k atribútom metódy.

```
class addPhone(Action):
    def run(self, deviceName, userName, fullResponse, description, pattern, routeParti$
      name = deviceName
      # priradenie udajov zo vstupu do zvysnych premennych
      try:
          resp  = service.addPhone(phone={
          'name': name,
          'description': description +' | '+ iproduct + ' | '+ pattern,
          'product': iproduct,
          'class': 'Phone',
          'protocol': iprotocol,
          'protocolSide': iprotocolSide,
          'devicePoolName': idevicePoolName,
```

Ukážka kódu 13 Časť akcie addPhone

Po úspešnom pridaní zariadenia nám StackStorm vráti nami naprogramovaný výpis, hlášku o úspešnom pridaní zariadenia do databázy s príslušným UUID novovytvoreného zariadenia, ktorú môžeme vidieť na obrázku 17.

```
Device was added with UUID:
{F18D8ACC-9953-85E9-E52D-DB525E947F97}
```

Obrázok 16 Hláška o úspešnom pridaní zariadenia do databázy

Pri všetkých vytvorených akciách sme vytvárali metadátové súbory a vykonali potrebné príkazy, tak ako to je podľa dokumentácie platformy StackStorm opísané v predchádzajúcej podkapitole *Výpis aktuálne registrovaných a neregistrovaných zariadení.*

Celý proces priebehu reťazi akcií s pridaním nového zariadenia do databázy môžeme vidieť na obrázku 18. V prípade vyskytnutia sa chyby počas priebehu reťazi akcií sa spustí akcia „*error*“, ktorá odošle chybovú hlášku používateľovi do aplikácie Webex Teams.

10:48:22	**ukf.chainAddPhone** UserName="fkubik",MacAdress="5006AB819B34",...	**ukf.chatopsAddPhone** core.st2.webhook	
10:48:22	getPhone	**default.getPhoneZ** template="TMP_8845"	
10:48:26	getUser	**default.getUserZ** user="fkubik"	
10:48:29	addPhone	**default.addPhoneZ4** userName="fkubik",deviceName="SEP5006AB819B34",...	
10:48:33	sendReportToSpark	**cisco_spark.send_message** text="Completed successfully on device SEP5006AB819B3	

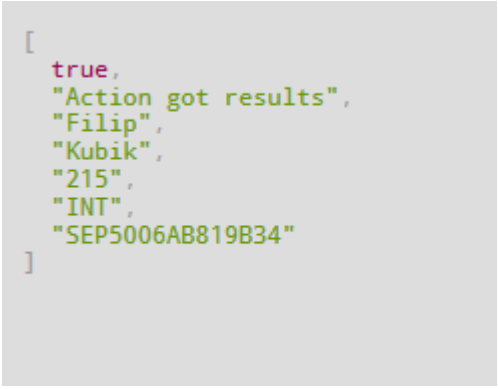
Obrázok 17 Priebeh akcií v reťazi akcií chainAddPhone

3.3.3 Update existujúceho zariadenia v databáze

Na aktualizáciu (update) existujúceho zariadenia v databáze je v dokumentácii AXL API popísaná metóda „*updatePhone*“. Táto metóda má v schéme uvedené rozsiahle možnosti použitia nepovinných aj povinných parametrov. Druhý scenár sme zadefinovali tak, že pred spustením samotnej metódy „*updatePhone*“ zistíme z databázy používateľa, ktorému chceme aktualizovať zariadenie. Na tento dopyt využijeme AXL metódu

„*getUser*“. Okrem aktualizácie zariadenia je pre úplnosť riešenia v databáze zároveň potrebné aktualizovať aj asociovanú linku. Tú dokážeme aktualizovať pomocou metódy „*updateLine*“. V reťazi akcií sa nám teda okrem akcie „*sendReportToSpark*“ budú nachádzať ďalšie tri akcie, ktoré použijú dopyty do databázy pomocou metód „*getUser*“, „*updateLine*“ a „*updatePhone*“.

Metóda „*getUser*“ je popísaná v predchádzajúcej podkapitole s názvom *Pridanie nového zariadenia do databázy.* Používateľ nám v aplikácii Webex Teams pomocou ChatBota zadá meno používateľa, ktorého zariadenie chce aktualizovať. Tento údaj prichádza na vstup do akcie „*getUser*“, kde sa ďalej s ním pracuje, ako je už popísané v predchádzajúcej kapitole. Výsledky z akcie „*getUser*“ prichádzajú ďalej na vstupy do akcií „*updatePhone*“ a „*updatePhoneRR*“, kde sú pomocou nich vykonané potrebné dopyty do databázy za účelom aktualizácie zariadenia a jeho asociovanej linky. Výsledný výstup z akcie „*getUser*“ s príslušnými parametrami môžeme vidieť na obrázku 19.



Obrázok 18 Výsledok akcie getUser s potrebnými dátami

Po akcii „*getUser*“ je v reťazi akcií zadefinované spustenie akcie „*updatePhone*“, v ktorej je naprogramovaný dopyt do databázy pomocou metódy „*updateLine*“. Na vstupe využijeme parametre získané z predchádzajúcej akcie „*getUser*“, ktoré nám vstupujú do premenných *pattern, alertingName, asciiAlertingName.* Následne vykonáme dopyt do databázy pomocou metódy „*updateLine*“, ktorá nám aktualizuje údaje na linke asociovanej so zariadením používateľa zadaného na začiatku scenára. V ukážke kódu 15 môžeme vidieť časť kódu akcie „*updatePhoneZ*“, v ktorej vykonávame dopyt pomocou metódy „*updateLine*“.

```
class updatePhoneZ(Action):
    def run(self, pattern, alertingName, asciiAlertingName):
      routePartitionName = 'INT'
      iroutePartitionName = routePartitionName
      ipattern = pattern
      ialertingName = alertingName
      iasciiAlertingName = asciiAlertingName
      try:
        resp  = service.updateLine(
          pattern = ipattern,
          routePartitionName = iroutePartitionName,
          alertingName = ialertingName,
          asciiAlertingName = iasciiAlertingName,
          )
        print(resp['return'])
```

Ukážka kódu 14 Časť kódu akcie updatePhoneZ s metódou dopytu updateLine

Po vykonaní druhej akcie v poradí v rámci zadefinovanej reťazi akcií môžeme prejsť k aktualizovaniu samotného zariadenia používateľa. Tento proces je naprogramovaný v akcii s názvom „*updatePhoneRR*“, ktorá používa na dopyt do databázy metódu „*updatePhone*“.

Na vstupe do akcie nám prichádzajú dáta z predošlej akcie „*getUser*“, ktoré priraďujeme do premenných *name*, *alertingName, pattern* a *routePartitionName*. Následne vykonáme dopyt do databázy pomocou metódy „*updatePhone*“, ktorá nám aktualizuje potrebné údaje zariadenia, ktoré sme zadefinovali v dopyte. V ukážke kódu 16 môžeme vidieť časť z kódu akcie „*updatePhoneRR*“, ktorá je zodpovedná za dopyt do databázy.

```
class updatePhoneRR(Action):
    def run(self, name, alertingName, pattern, routePartitionName):
      idisplayName = alertingName
      ilineTextLabel = alertingName + pattern
      ipatern = pattern
      iroutePartitionName = routePartitionName
      try:
        resp  = service.updatePhone(name = name, lines = {  'line' : {
                    'index' : '1',
                    'display' : idisplayName,
                    'label' : ilineTextLabel,
```

```
                'dirn' : {'pattern': pattern,
                    'routePartitionName': iroutePartitionName
                    }
                }
    print(resp['return'])
```

Ukážka kódu 15 Časť akcie updatePhoneRR

Po úspešnom zbehnutí tretej akcie z reťazi akcií s názvom „*chainUpdatePhone*“ môžeme výsledok odoslať do aplikácie Webex Teams pomocou akcie „*sendReportToSpark*“.

```
{2A86FDB6-C9A4-4F3D-115F-A41758229AA2}
```

Obrázok 19 Výsledok akcie updatePhoneRR obsahujúci iba UUID aktualizovaného zariadenia

Výsledok akcie „*updatePhoneRR*“ obsahuje iba UUID zariadenia, ktoré bolo aktualizované (obrázok 20). Preto je v reťazi akcií zadefinovaná podmienka, že ak akcia „*updatePhoneRR*“ zbehne úspešne na vstup do akcie „*sendReportToSpark*“ sa pridá text: „Complete successfully on device: [UUID zariadenia].” Ak by akcia nezbehla úspešne do aplikácie Webex Teams by sa odoslala chybová hláška pomocou akcie „*error*“.

Celý proces reťazi akcií „*chainUpdatePhone*“ je zobrazený na obrázku 21.

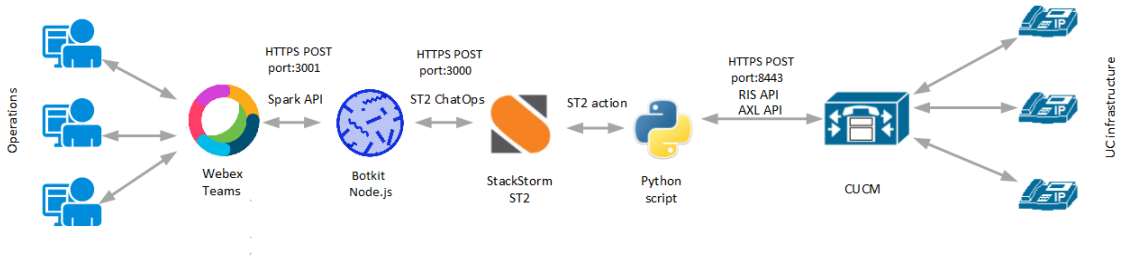
08:51:52	**ukf.chainUpdatePhone** pattern="fkubik",...	**ukf.chatopsUpdatePhone** core.st2.webhook	
08:51:53	getUser	**default.getUserZ** user="fkubik"	
08:51:56	updatePhone	**default.udpatePhoneZ3** pattern="215",asciiAlertingName="Filip Kubik",...	
08:51:59	updatePhoneRR	**default.updatePhoneRR** pattern="215",routePartitionName="INT",...	
08:52:02	sendReportToSpark	**cisco_spark.send_message** text="Result: \n True",...	

Obrázok 20 Úspešný proces reťazi akcií chainUpdatePhone

3.4 PROCES PREPOJENIA EXISTUJÚCICH RIEŠENÍ

V rámci našej práce sme vytvorili funkčné riešenie dopytu do databázy pomocou platformy StackStorm. Celé ChatOps riešenie ale zahŕňa aj napojenie ChatBota do aplikácie Webex Teams a napojenie ChatBota na platformu StackStorm. Túto časť ChatOps riešenia rieši vo svojej diplomovej práci Bc. Michal Dvořák. V rámci architektúry

riešenia bol za ChatBot platformu vybratý bot Botkit pracujúci na Node.js základe. Nastali mierne zmeny oproti prvotným návrhom riešenia a konečná architektúra nášho ChatOps riešenia sa dostala do podoby, ktorú môžeme vidieť na obrázku 22.

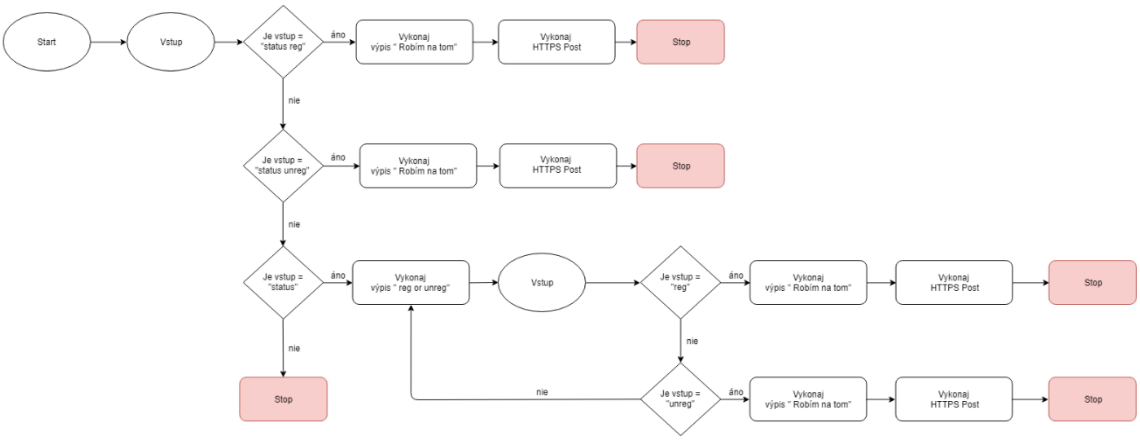


Obrázok 21 Výsledná architektúra ChatOps riešenia

Zmeny nastali aj v pôvodnom návrhu práce iba s AXL API, keď sme zistili, že na prácu s databázou reálne v čase je potrebné RIS API. Zmena nastala aj pri četovej platforme, po zlúčení Cisco Spark a Cisco Webex pod jednu službu s názvom Webex Teams.

Prvý scenár - Výpis aktuálne registrovaných a neregistrovaných zariadení

Z hľadiska riešenia v platforme StackStorm sa v prvom scenári očakáva na vstupných hodnotách iba jedno slovo, ktoré musí byť buď „*Registered*“ alebo „*UnRegistered*“. Na tomto princípe bola zadefinovaná aj komunikácia v aplikácii Webex Teams. Naformulované pravidlá komunikácie používateľa s ChatBotom definujú, kedy sa má aké volanie spustiť. Pre prvý scenár sme vytvorili tri možnosti výpisu a tými sú: výpis všetkých registrovaných zariadení, výpis všetkých neregistrovaných zariadení a výpis registrovaných aj neregistrovaných zariadení zároveň. Pre každú možnosť sme definovali kľúčové slová *status reg*, *status unreg*, *status*. Na obrázku 23 môžeme vidieť diagram komunikácie s ChatBotom. Tento scenár je jediný z našich troch scenárov, ktorý využíva RIS API na dopyt do databázy. Po zadaní jedného z týchto kľúčových slov nám má ChatOps vo výpise vrátiť požadované výsledky odfiltrované do podoby čitateľnej a zrozumiteľnej pre používateľa. Práca sa týmto pre používateľa obmedzí iba na výpis kľúčového slova a automatizácia ChatOps mu vráti potrebné dáta.



Obrázok 22 Diagram komunikácie s ChatBotom – prvý scenár

Druhý scenár - Pridanie nového zariadenia do databázy

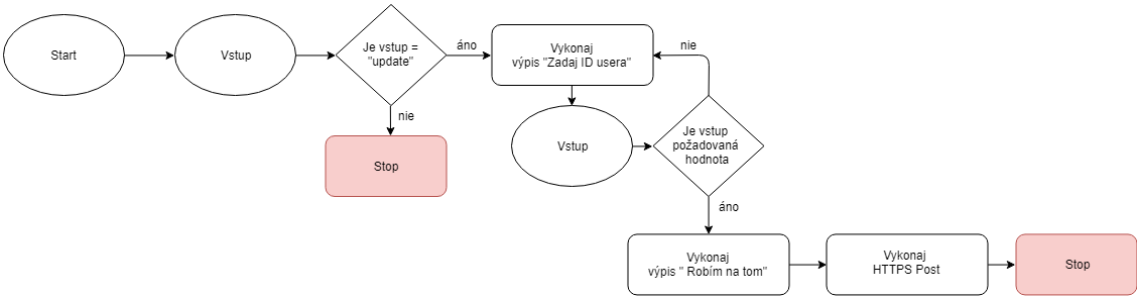
Pridanie nového zariadenia do databázy si vyžaduje viacero údajov, ktoré potrebujeme získať od používateľa. Po zadaní kľúčového slova *add* v rámci konverzácie s botom v četovej miestnosti aplikácie Webex Teams, sa dostávame do procesu zbierania údajov pre pridanie nového zariadenia do databázy. V prvom rade musí používateľ vybrať šablónu, podľa ktorej sa majú vyplniť jednotlivé údaje. Vybranie šablóny sme pre používateľa obmedzili na zadanie jednej z troch číslic (1, 2, 3) pre výber príslušnej šablóny. Ďalej musí používateľ zadať MAC adresu, ktorá je vzápätí overovaná na valídnosť a taktiež musí zadať ID (identifikátor) používateľa. Všetky tieto dáta zadané používateľom nám prichádzajú na vstupoch do akcií v platforme StackStorm. Meno šablóny je priradené na vstupe do akcie „*getPhone*“ ako nami vytvorený povinný parameter *template*, kde sa ďalej pomocou neho vykoná dopyt do databázy na získanie šablóny. Získaním šablóny získame potrebné dáta na vytvorenie zariadenia pomocou predefinovaných parametrov, preto tieto údaje ďalej použijeme vo výsledkoch akcie. ID používateľa vstupuje ako parameter do akcie „*getUser*“, aby sme mohli získať potrebného používateľa, ktorému máme priradiť nové zariadenie. Do akcie vstupuje ako povinný parameter *user* a je kľúčovým parametrom akcie na dopyt do databázy. Pomocou tohto parametra získame vo výstupe z databázy ďalšie potrebné dáta, ktorými sú *firstName*, *lastName*, *pattern*, *routePartitionName* a *associatedDevices*. Všetky získané údaje z akcií „*getUser*“ a „*getPhone*“ použijeme na vstupoch do akcie „*addPhone*“ ako povinné parametre.

Práca používateľa sa pri druhom scenári takto obmedzí iba na zadanie troch parametrov a potvrdenie akcie. Všetky ostatné úkony nutné pre vytvorenie zariadenia sú plne automatizované a používateľovi už len príde informácia do aplikácie Webex Teams o úspešnom alebo neúspešnom pridaní zariadenia.

Tretí scenár - Update existujúceho zariadenia v databáze

Pri treťom scenári sme využili možnosti použitých platforiem a vytvorili sme plne automatizovaný scenár. Pre vstup do tejto akcie stačí používateľovi zadať kľúčové slovo *update*. Po zadaní tohto kľúčového slova sa dostávame do konverzácie s botom, v ktorej nás bot vyzve na zadanie ID používateľa. Po zadaní tohto parametra už pokračuje plne automatizovaný scenár, pri ktorom sa údaje o používateľovi získané z akcie *„getUser“* priradia na vstupoch do akcií *„updatePhone“* a *„updatePhoneRR“*, aby sa mohli aktualizovať potrebné dáta o používateľovi uložené na zariadení a asociovanej linke. Po uskutočnení celého vyššie spomenutého procesu sa používateľovi odošle do aplikácie Webex Teams hláška o úspešnom, alebo neúspešnom vykonaní aktualizácie.

Pre vykonanie tohto scenára nám stačí zadať jedno kľúčové slovo *add* a jeden parameter ID používateľa a celý proces prebehne bez akéhokoľvek ďalšieho zásahu používateľa. Preto je tento scenár ukážkovým príkladom uľahčenia práce pre používateľa pomocou ChatOps riešenia. Na obrázku 24 môžeme vidieť diagram komunikácie používateľa s ChatBotom pri zadávaní tretieho scenára.



Obrázok 23 Diagram komunikácie používateľa s ChatBotom

4 VÝSLEDKY RIEŠENIA A ICH ZHODNOTENIE

Cieľom našej práce bolo vytvorenie troch scenárov, na ktorých by sme demonštrovali ChatOps riešenie pre podporu prevádzky CUCM s využitím platformy Webex Teams a event driven platformy StackStorm. Vytvorili sme tri prípady použitia nášho kompletného ChatOps riešenia na troch základných úkonoch vykonávaných s databázou, ktorými sú výpis z databázy, pridanie nového zariadenia do databázy a aktualizácia údajov v databáze. Týmto riešením demonštrujeme kompletnosť využitia ChatOps riešenia pri pokrytí úkonov v rámci používania CUCM platformy. Uľahčenie práce používateľa, teda zvyčajne pracovníka určitej firmy, nad platformou CUCM je jednoznačné. Naše riešenie ChatOps odbremení používateľa od monotónnej práce postupného zisťovania a následného vypĺňania jednotlivých parametrov pri práci s CUCM v rámci daného GUI rozhrania. Namiesto vyhľadávania zariadení, šablón a ostatných potrebných údajov v rámci CUCM stačí používateľovi zadať zopár kľúčových slov do aplikácie Webex Teams v rámci konverzácie s našim ChatBotom menom Ellie. Ellie pomocou nášho ChatOps riešenia jednoznačne skráti čas výkonu daných úkonov nad databázou. Používateľ zadaním jedného z kľúčových slov *status*, *add* alebo *update* v rámci konverzácie s Ellie spustí proces zbierania dát, ktoré sú spravidla obmedzené na jedno slovo (tretí scenár) až štyri odpovede (druhý scenár) plus potvrdenie spustenia procesu, čím spustí samotný proces úkonov, ktorých konečným výsledkom je vytvorenie daného scenára. Po ukončení úkonov nad databázou sa používateľovi vráti požadovaná odpoveď o úspešnom alebo neúspešnom vykonaní procesu v rámci databázy. Celý proces môže trvať niekoľko sekúnd až zopár minút v závislosti od viacerých faktorov (rýchlosť zadania údajov používateľom, rýchlosť siete, typ scenára). Toto je jednoznačné ušetrenie času, ktorý by používateľ inak strávil nad zbieraním a vypĺňaním dát v GUI rozhraní pre CUCM.

Naše ChatOps riešenie bolo zadávané, spracovávané a vytvárané v kooperácii so súkromnou spoločnosťou, slovenskou pobočkou medzinárodného koncernu, Dimension Data. Výsledok projektu mal byť odprezentovaný na workshope v sídle pobočky spoločnosti Dimension Data v Bratislave pred vedením spoločnosti, zamestnancami spoločnosti (obchodný zástupca, náš projektový manažér, náš kooperačný tím, ...) a potencionálnymi zákazníkmi, teda používateľmi takéhoto ChatOps

riešenia v rámci praxe. Pri workshope sme dospeli k záveru, že takéto riešenie ChatOps automatizácie má veľký potenciál pre využitie v praxi nielen v rámci platforiem CUCM a Webex Teams, ale aj v rámci iných počiatočných a koncových platforiem pri súčasnom použití našej logiky a architektúry ChatOps riešenia s využitím event driven platformy StackStorm a existujúceho prepojenia s botkitom. Na obrázku 25 môžeme vidieť záber z workshopu uskutočneného v Bratislave.

Obrázok 24 Workshop na tému ChatOps (prezentujúci Bc. Dávid Pavelka)

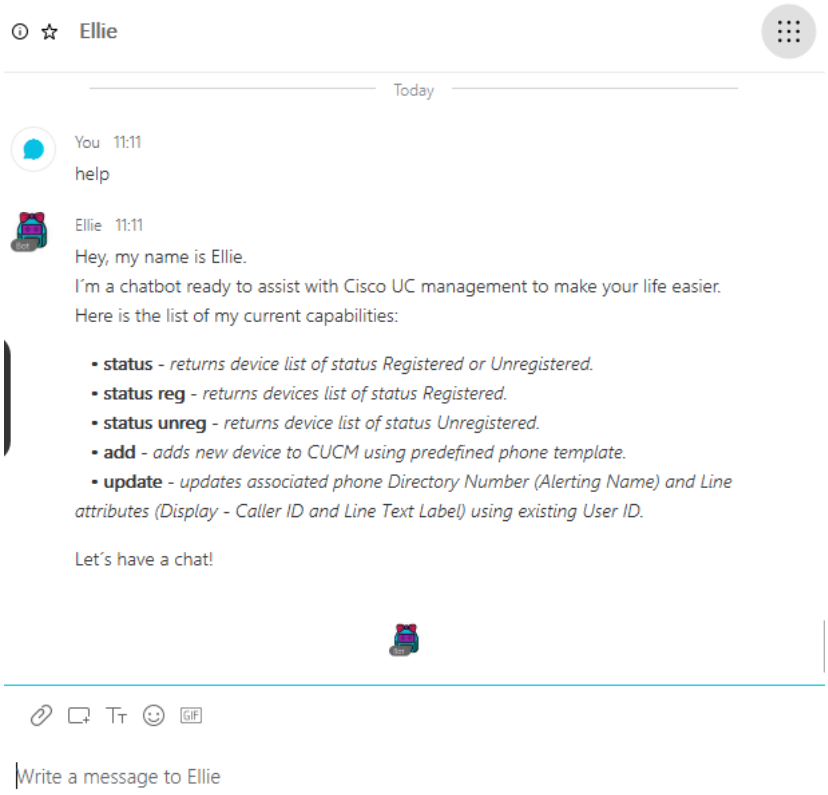
4.1 VÝSLEDKY JEDNOLIVÝCH SCENÁROV

V rámci aplikácie Webex Teams si vytvoríme četovú konverzáciu s existujúcim botom Ellie. V tejto konverzácii môžeme písať kľúčové slová priamo botovi bez špeciálneho označenia. V prípade, ak by sme Ellie pridali do skupinovej četovej miestnosti, musíme začať konverzáciu s Ellie označením umiestneným na začiatku správy ako „*@Ellie*“. Pre jedného používateľa je vhodnejšie použitie konverzácie one-to-one (teda jedna k jednej), čím zabezpečíme rýchlejšie písanie správ samotným používateľom.

Významnou výhodou nášho riešenia je jeho dostupnosť. Na vykonanie potrebných scenárov nepotrebujeme inštalovať žiadne rozsiahle aplikácie, riešenia alebo knižnice. Stačí nám nainštalovaná jediná aplikácia Webex Teams. Tú môžeme mať nainštalovanú na svojom počítači, ale aj smartfóne. Momentálne je pre operačné

systémy smartfónov dostupná verzia aplikácie Webex Teams zadarmo. Z hľadiska softvérovej, alebo hardvérovej náročnosti je riešenie nenáročné. Stačí spĺňať bežné hardvérové požiadavky pre aplikáciu Webex Teams na svojom počítači, alebo smartfóne a mať internetové spojenie. Práve vďaka internetovému spojeniu, ktoré je jedinou podmienkou na chod riešenia, je jeho dostupnosť veľmi rozsiahla. V konečnom riešení nám stačí mať aplikáciu Webex Teams a internetové spojenie a môžeme s botom komunikovať a vytvárať scenáre doslova odkiaľkoľvek.

Pre uľahčenie komunikácie s botom sme vytvorili príkaz „*help*“, po ktorom zadaní nám Ellie vypíše možnosti kľúčových slov, respektíve príkazov, ktoré máme možnosť zadať. Na obrázku 26 môžeme vidieť výsledok komunikácie po zadaní kľúčového slova „*help*“. Ellie nám vráti základný popis o sebe a zoznam možných príkazov.

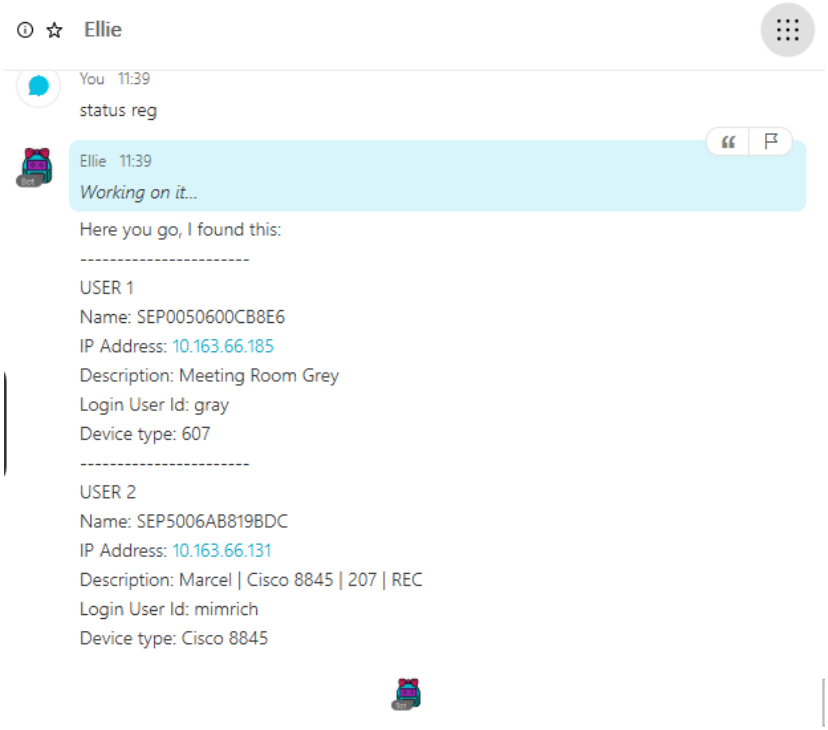


Obrázok 25 Výsledok komunikácie po zadaní kľúčového slova „help“

Prvý scenár - Výpis aktuálne registrovaných a neregistrovaných zariadení

Ako možno vidieť z obrázka 26, prvý scenár je zahrnutý v troch príkazoch: *status*, *status reg, status unreg*. Po zadaní príkazu *status* nám Ellie vráti doplňujúcu vetu, aby sme zadali kľúčové slovo *reg* pre prípad, ak chceme vrátiť zoznam registrovaných zariadení, alebo kľúčové slovo *unreg,* ak chceme vrátiť zoznam neregistrovaných

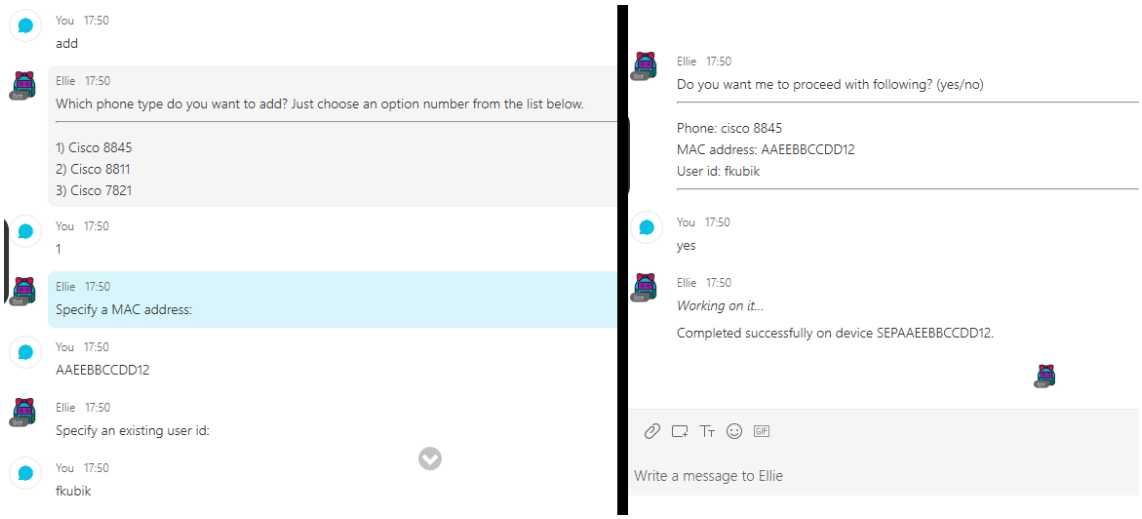
zariadení. Po zadaní príkazu *status reg*, Ellie vráti zoznam všetkých zariadení so statusom „*Registered*“ (môžeme vidieť na obrázku 27) a po zadaní príkazu *status unreg* nám vráti zoznam všetkých zariadení so statusom „*Unregistered*“.



Obrázok 26 Výsledný zoznam po zadaní príkazu „status reg“

Druhý scenár - Pridanie nového zariadenia do databázy

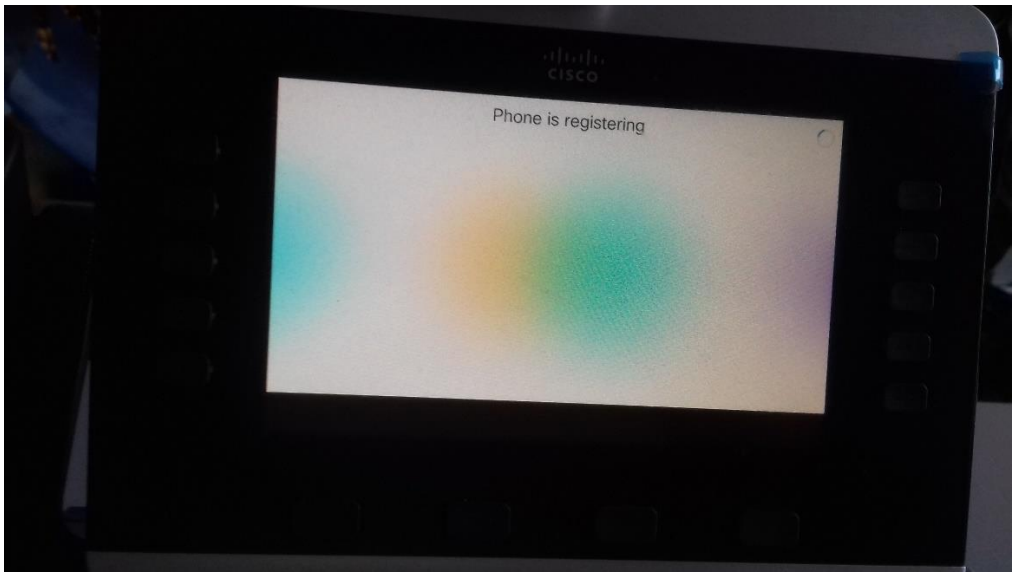
Pre pridanie zariadenia do databázy použijeme príkaz *add.* Po jeho zadaní musíme odpovedať Ellie na otázky o MAC adrese zariadenia, používateľovi a musíme vybrať, aká šablóna sa má pri pridaní použiť. Tieto zodpovedané hodnoty nám prechádzajú do akcií platformy StackStorm ako premenné pre vykonanie jednotlivých dopytov. Na obrázku 28 môžeme vidieť postup scenára pri pridaní nového zariadenia do databázy.



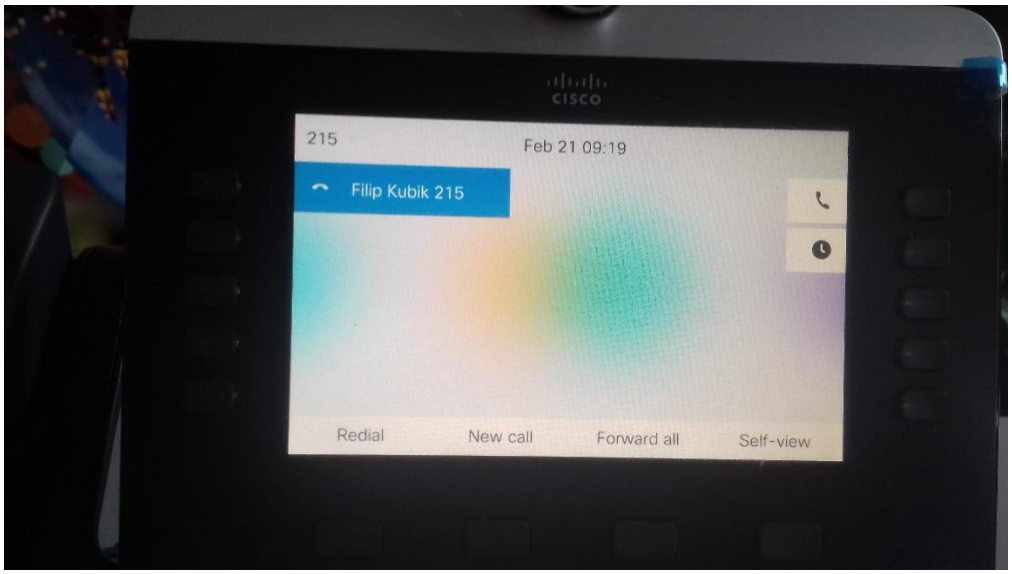
Obrázok 27 Pridanie nového zariadenia do databázy

Celý scenár možno odpozorovať aj na samotnom zariadení. Na obrázkoch 29 až 31 môžeme vidieť postupnosť krokov pri pridaní zariadenia. Najskôr zariadenie nie je pridané a svieti čisto biely displej (obrázok 29), neskôr sa zariadenie registruje, o čom hovorí oznamovacia veta na vrchu displeja (obrázok 30) a nakoniec je zariadenie registrované na konkrétneho používateľa (obrázok 31). Kompletný scenár od momentu začatia komunikácie po registrovanie zariadenia trvá do dvoch minút, čo je značné ušetrenie času oproti tomu, čo by používateľ musel vynaložiť na manuálne pridanie zariadenia do databázy pomocou GUI rozhrania.

Obrázok 28 Fyzické zariadenie pred registráciou v databáze



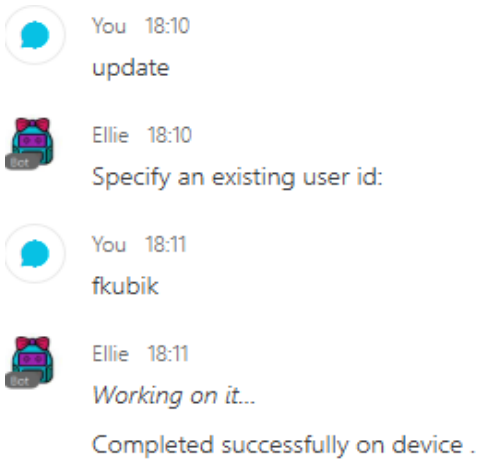
Obrázok 29 Fyzické zariadenie počas priebehu registrácie



Obrázok 30 Fyzické zariadenie po úspešnom zbehnutí druhého scenára

Tretí scenár - Update existujúceho zariadenia v databáze

Tretí scenár, ktorý aktualizuje údaje existujúceho zariadenia podľa údajov daného používateľa, je plne automatizovaný scenár. Teda tento scenár nepotrebuje na svoje vykonanie žiadne ďalšie dáta, alebo inštrukcie, iba jeden jediný príkaz a meno používateľa, ktoré zadáme po spustení scenára príkazom *update*. Na obrázku 32 môžeme vidieť použitie tretieho scenára v praxi.



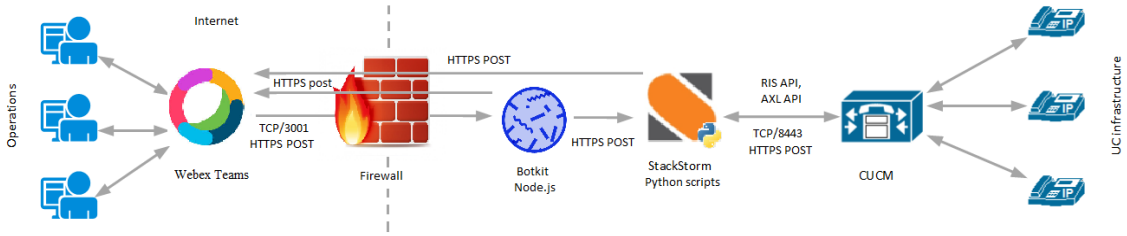
Obrázok 31 Priebeh tretieho scenára

4.2 BEZPEČNOSŤ RIEŠENIA

V rámci bezpečnosti riešenia dochádza k spojeniu firemných aplikácii s cloudovou službou. Kritický bod riešenia vzhľadom na bezpečnosť sa nachádza práve medzi cloudovou službou Webex Teams a firemnými aplikáciami. V tejto časti riešenia prebieha komunikácia cez HTTPS post, čo znamená, že komunikácia je šifrovaná. Ďalším prvkom bezpečnosti je *Secret Key* umiestnený v posielaných dátach, ktorý zamedzuje nežiadúcim požiadavkám. Takisto obmedzenie prístupu iba na jeden port je zamedzením vstupu dovnútra riešenia. Architektúru nášho ChatOps riešenia s vyznačenými bezpečnostnými prvkami môžeme vidieť na obrázku 33.

Problém, ktorý môže nastať v tomto riešení je DDoS útok na server. Tento problém je však bezpečnostným problémom aj v rozsiahlych známych riešeniach významných spoločností.

Naše riešenie, ktoré je prepojením rozličných systémov, cloudovej a firemnej služby, je napriek svojej rozsiahlosti veľmi bezpečné.



Obrázok 32 Architektúra ChatOps riešenia s ohľadom na prvky bezpečnosti

ZÁVER

Prínosom nášho ChatOps riešenia je možné praktické využitie logiky a architektúry nášho riešenia na rôzne koncové a počiatočné platformy, kde môžeme nahradiť platformu CUCM a Webex Teams inými platformami.

Architektúra a logika nášho riešenia tým dostáva flexibilnosť rôznorodosti použitia v rámci aplikačnej praxi informačných technológií. Pri použití platformy CUCM sme demonštrovali jednoznačné uľahčenie pracovných úkonov na vykonávanie scenárov, ktoré sa reálne v praxi používajú. Takéto uľahčenie častokrát monotónnych pracovných úkonov zvýši nielen komfort pracovníkov, ale aj urýchli spôsob výkonu práce a teda zníži časovú náročnosť, čo má v konečnom dôsledku pozitívne dopady na cenu práce. V praktickom využití však treba brať ohľad na rozsiahlosť riešenia. Ak má byť z finančného hľadiska ChatOps automatizácia prínosom, musí byť prepočítaná nákladovosť vývoja ChatOps vzhľadom k pozitívnemu dopadu tohto riešenia na cenu práce. Pretože práve cena vývoja automatizácie vo všeobecnosti je jedným z dôvodov prečo súkromné spoločnosti takéto riešenia odkladajú.

Z technologického hľadiska je veľkou výhodou rôznorodosť využitia platformy StackStorm a jej možného napojenia na koncové aplikácie, jej výborne spracovaná architektúra a možnosť písania vlastného kódu v širokom spektre programovacích jazykov. Ako možná nevýhoda tejto platformy by sa mohla objaviť viazanosť základu StackStorm na programovací jazyk python. ChatBot Botkit má zasa rozsiahle možnosti implementácie na četové aplikácie. Bezpečnosť, ktorej sa v sektore informačných technológii dnes kladie veľký dôraz, je v našom riešení veľmi dobre zvládnutá voči útokom zvonka.

Napriek možným nedostatkom nášho riešenia a ChatOps automatizácii vo všeobecnosti vidíme veľké možnosti aplikovania týchto typov automatizácii ako riešení, ktoré v konečnom dôsledku uľahčia nielen pracovný život ľuďom, či už v súčasnosti, ale predovšetkým v budúcnosti. Pretože veľká časť budúcnosti patrí softvérových automatizáciám.

ZOZNAM BIBLIOGRAFICKÝCH ODKAZOV

ATTLASIAN. 2018. Frequently Asked Questions. Atlassian. [online] Atlassian. 26. júl 2018. [cit. 2019-02-03]. Dostupné na internete: <https://www.atlassian.com/partnerships/slack/faq#faq-da2b66a1-53d3-4c4e-a405-467d961336f7>.

BENNETT, Stuart. 1993. A History of Control Engineering 1930-1955. Londýn: Peter Peregrinus Ltd. 23 s. 1993. ISBN 0-86341-280-7.

BESTPRACTICE.SK. 2019. DevOps. [online] OMNICOM, s.r.o. 2019. [cit. 2019-01-03]. Dostupné na internete: <https://www.bestpractice.sk/sk/Best-practice/DevOps.alej.>

BOTKIT.AI. 2019. Building Blocks for Building Bots. Botkit. [online] XOXCO Inc. 2019. [cit. 2019-03-03]. Dostupné na internete: <https://botkit.ai/>.

CISCO. 2019. Cisco Unified Communications Manager. https://www.cisco.com. [online] Cisco. [cit. 2019-06-03]. Dostupné na internete: <https://www.cisco.com/c/en/us/solutions/industries/government/us-government-solutions-services/unified-communications/unified-communication-manager.html>.

CISCO. 2019. Cisco Unified Communications Manager (CallManager). https://www.cisco.com. [online] Cisco. [cit. 2019-06-03] Dostupné na internete: <https://www.cisco.com/c/en/us/products/unified-communications /unified-communications-manager-callmanager/index.html>.

CISCO. 2019. Cisco Webex Teams. Cisco Webex. [online] Cisco. 2019. [cit. 2019-02-03]. Dostupné na internete: <https://www.webex.com/team-collaboration. html>.

CISCO. 2019. Cisco Webex Teams - Make teamwork your best work. https://www.webex.com. [online] Cisco. [cit. 2019-06-03]. Dostupné na internete: <https://www.webex.com/team-collaboration.html>.

DEEP AI, INC. 2018. deepai.org. DeepAI. [online] 24. december 2018. [cit. 2018-24-12]. Dostupné na internete: <https://deepai.org/machine-learning-glossary-and-terms/neural-network>.

DELOITTE. 2018. www2.deloitte.com. Deloitte. [online] Deloitte Touche Tohmatsu Limited, LBG. 11. december 2018. [cit. 2019-02-01]. Dostupné na internete: <https://www2.deloitte.com/us/en/pages/deloitte-analytics/solutions/robotics-and-cognitive-automation.html#>.

SIGLER, Eric. 2014. What Is ChatOps?. https://www.pagerduty.com. [online] Pagerduty. 2. december 2014. [cit. 2019-28-02]. Dostupné na internete: <https://www.pagerduty.com/blog/what-is-chatops/>.

GARTNER.COM. 2019. Business Process Automation - Gartner IT Glossary. Gartner.com. [online] Gartner.com. 20. január 2019. [cit. 2019-20-02]. Dostupné na internete: <https://www.gartner.com/it-glossary/bpa-business-process-automation>.

GITHUB. 2019. Built for developers. https://github.com/. [online] GitHub, Inc. 2019. [cit. 2019-27-02]. Dostupné na internete: <https://github.com/>.

GROOVER, Mikell P. 2014. Fundamentals of Modern Manufacturing: Materials, Processes, and Systems. Hoboken: JOHN WILEY & SONS, INC. 2014. ISBN 978-0470-467002.

GUHATHAKURTA, Rahul. 2018. www.indrastra.com. Indastra. [online] 4. september 2018. [cit. 2019-16-01]. Dostupné na internete: <https://www.indrastra.com/2018/09/Cognitive-Automation-004-09-2018-0006.html>. ISSN 2381-3652.

HILL, David J. 2012. Automation Comes To The Coffeehouse With Robotic Baristas. singularityhub.com. [online] Singularity Education Group, 9. máj 2012. [cit. 2019-19-02]. Dostupné na internete: <https://singularityhub.com/2012/05/09/automation-comes-to-the-coffeehouse-with-robotic-baristas/#sm.0001fv5mp2emyf5710e8m0cj3hv1t>.

CHAN, Stephanie. 2019. Everything you need to know about Cisco's new Webex. https://newsroom.cisco.com. [online] Cisco, 18. apríl 2019. [cit. 2019-06-03]. Dostupné na internete: <https://newsroom.cisco.com/feature-content?type=webcontent&articleId=1922422&CAMPAIGN=Collaboration&Country_Site=us&POSITION=Social+Media&REFERRING_SITE=Twitter&CREATIVE=CiscoCollab>.

CHATBOT.ORG. 2019. ChatBot. ChatBot.org. [online] Your Future Is Now, 2019. [cit. 2019-03-03]. Dostupné na internete: <https://www.chatbots.org/chatbot/>.

JACKSON, Brian. 2011. McDonald’s automation a sign of declining service sector employment. www.itbusiness.ca. [online] ITBusiness.ca, 19. máj 2011. [cit. 2019-18-02]. Dostupné na internete: <http://www.itbusiness.ca/blog/ mcdonalds-automation-a-sign-of-declining-service-sector-employment/20398>.

NIST PHYSICAL MEASUREMENT LABORATORY. 2018. A Walk Through Time - Early Clocks. https://www.nist.gov. [online] 25. december 2018. [cit. 2018-25-12]. Dostupné na internete: <https://www.nist.gov/pml/walk-through-time-early-clocks>.

NITRADEN.SK. 2018. Nitra získala cenu za najinovatívnejší systém zberu odpadu na Slovensku. nitraden.sk. [online] NitraDen.sk, 6. október 2018. [cit. 2019-23-02]. Dostupné na internete: <https://nitraden.sk/nitra-ziskala-cenu-za-najinovativnejsi-system-zberu-odpadu-na-slovensku/>.

PASKIN, Becky. 2011. New Pizza Express app lets diners pay bill using iPhone. www.bighospitality.co.uk. [online] William Reed Business Media Ltd. 17. jún 2011. [cit. 2019-19-02]. Dostupné na internete: <https://www.bighospitality.co.uk/Article/2011/06/17/New-Pizza-Express-app-lets-diners-pay-bill-using-iPhone>.

PETROVER, Tali Davidovich, BINET, Guillaume a GROENEN, Nick. 2019. Errbot. Errbot.io. [online] Errbot.io. 2019. [cit. 2019-03-03]. Dostupné na internete: <http://errbot.io/en/latest/>.

PYTHON.ORG. 2019. Python. Python. [online] The Python Software Foundation . [cit. 2019-05-03]. Dostupné na internete: <https://www.python.org/>.

RIFKIM, Jeremy. 1995. The End of Work: The Decline of the Global Labor Force and the Dawn of the Post-Market Era. New York: G. P. Putnam's Sons. 66,75 s. 1995. ISBN 0-87477-779-8.

ROBOTIC INDUSTRIES ASSOCIATION. 2018. Robot density rises globally. https://www.robotics.org. [online] 2. august 2018. [cit. 2018-26-12]. Dostupné na internete: <https://www.robotics.org/content-detail.cfm/Industrial-Robotics-News/Robot-density-rises-globally/content_id/7002>.

ROUSE, Margaret. 2016. Definition Chatops. techtarget.com. [online] TechTarget, 30. september 2016. [cit. 2019-24-02]. Dostupné na internete: <https://searchitoperations .techtarget.com/definition/ChatOps>.

SERKAN, Toto. 2010. Wheelie: Toshiba's new robot is cute, autonomous and maybe even useful. techcrunch.com. [online] Verizon Media, 23. január 2010. [cit. 2019-20-02]. Dostupné na internete: <https://techcrunch.com/2010/03/12/ wheelie-toshibas-new-robot-is-cute-autonomous-and-maybe-even-useful-

video/?guccounter=1&guce_referrer_us=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnLw&guce_referrer_cs=ZRZJ5bCnDXLwvVeq6tIdvA>.

STACKSTORM. 2019. StackStorm Overview. StackStorm. [online] StackStorm. [cit. 2019-04-03]. Dostupné na internete: <https://docs.stackstorm.com/overview.html>.

STACKSTORM. 2019. What is ChatOps? https://docs.stackstorm.com. [online] StackStorm, 2019. [cit. 2019-24-02]. Dostupné na internete: <https://docs.stackstorm.com/chatops/chatops.html>.

STENOVEC, Timothy. 2014. The Real Reason Facebook Is Forcing You To Download Messenger. HuffPost. [online] Verizon Media, 13. augusta 2014. [cit. 2019-02-03]. Dostupné na internete: <https://www.huffingtonpost.com/2014/08/13/facebook-messenger_n_5674703.html>.

TEBILION. 2018. Three Reasons Why Your Business Needs To Automate Its Sales Process. TEBilion. [online] TEBilion, 27. jún 2018. [cit. 2019-21-02]. Dostupné na internete: <https://www.tebillion.com/en/blog/111-three-reasons-why-your-business-needs-to-automate-its-sales-process>.

THE AGILE ADMIN. 2019. What Is DevOps? The Agile Admin. [online] The Agile Admin, 2019. [cit. 2019-01-03]. Dostupné na internete: <https://theagileadmin.com/what-is-devops/>.

THE GITHUB BLOG. 2019. What is Hubot. Hubot. [online] The GitHub Blog. 2019. [cit. 2019-03-03]. Dostupné na internete: <https://hubot.github.com/.

THE SILICON REVIEW. 2017. Automate Complex Workflows Using Tactical Cognitive Computing Coseer. thesiliconreview.com. [online] The Silicon Review, 30. júl 2017. [cit. 2019-16-01]. Dostupné na internete: <https://thesiliconreview.com/magazines/automate-complex-workflows-using-tactical-cognitive-computing-coseer/>.

TROUNSON, Andrew. 2008. Rio to trial automated mining. The Weekend Australian. [online] 19. január 2008. [cit. 2019-19-02]. Dostupné na internete: <https://www.theaustralian.com.au/business/mining-energy/rio-to-trial-automated-mining/news-story/d889f879e5d54ed73772fbd77c5d59b2>.

TSANG, Rose. 2012. https://gspp.berkeley.edu/. [online] 13. august 2012. [cit. 2018-25-12]. Dostupné na internete: <http://gspp.berkeley.edu/iths/Tsang_SCADA Attacks.pdf>.

GERVEN, Marcel van a BOHTE, Sander. 2018. Artificial Neural Networks as Models of Neural Information Processing. Lausanne: Frontiers Media, 2018. 36 s. ISBN 978-2-88945-401-3.

VLACHYNSKÝ, Martin. 2018. Slovensko patrí k lídrom automatizácie, no s robotmi netreba súperiť (Trend). INESS. [online] 1. máj 2018. [cit. 2019-17-01]. Dostupné na internete: <http://www.iness.sk/sk/slovensko-patri-k-lidrom-automatizacie-no-s-robotmi-netreba-superit-trend>.

WEIZENBAUM, J. Eliza. 1966. A Computer Program For the Study of Natural Language Communication Between Man And Machine. MIT. 1966, ročník 9, číslo 1.

ZYANE, Rania. 2017. How ChatOps Can Help You DevOps Better. Chatbots Magazine. [online] A Medium Corporation, 18. máj 2017. [cit. 2019-28-02]. Dostupné na internete: <https://chatbotsmagazine.com/how-chatops-can-help-you-devops-better-5-minutes-read-507438c156bf>.

ZOZNAM PRÍLOH

Príloha A – Optické médium obsahujúce diplomovú prácu vo formáte PDF