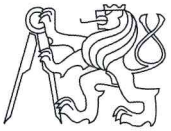
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Hlasové ovládání inteligentních domů iQtec
Student:	Matěj Hlaváček
Vedoucí:	Ing. Milan Kolář
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce zimního semestru 2018/19

Pokyny pro vypracování

Navrhněte a naimplementujte hlasové ovládání zařízení inteligentních domů iQtec pomocí technologie Amazon Echo:
1) Nastudujte API Amazon Echo a inteligentních domů iQtec a navrhněte jejich propojení.
2) Naimplementujte hlasové ovládání pro různá zařízení iQtec jako světla, žaluzie a vytápění.
3) Zautomatizujte nasazování na servery Amazon Web Services Lambda.
4) Otestujte aplikaci pomocí jednotkových a integračních testů.
5) Otestujte systém v reálné instalaci.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 19. února 2017

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Hlasové ovládání inteligentních domů iQtec

Matěj Hlaváček

Vedoucí práce: Ing. Milan Kolář

14. května 2017

Poděkování

V první řadě bych chtěl poděkovat panu Ing. Milanovi Kolářovi za příležitost pracovat na projektu, který mě zajímal a bavil, cennou podporu při realizaci a za ochotu vést mojí bakalářskou práci. Dále bych chtěl poděkovat Jakubovi Hlaváčkovi za velmi nápomocné rady při vývoji, Janu Vojtěškovi za vynikající připomínky a v neposlední řadě celé své rodině za podporu po celou dobu studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 14. května 2017 .....

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2017 Matěj Hlaváček. Všechna práva vyhrazena.
Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí, je nezbytný souhlas autora.

Odkaz na tuto práci

Hlaváček, Matěj. *Hlasové ovládání inteligentních domů iQtec*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Cílem této bakalářské práce je vyvinutí hlasového rozhraní pro ovládání inteligentních domů iQtec pomocí technologie Amazon Alexa. V práci je zahrnuta analýza a návrh celého systému, implementace dovedností pro zařízení Amazon Echo v jazyce Python, nasazování na servery Amazon Web Services Lambda, vytvoření autorizačního serveru podporujícího protokol OAuth a integrace do existující aplikace iQtec Architekt. Součástí práce je také realizace otevřené knihovny Askhome pro zjednodušení vytváření hlasových rozhraní inteligentních domů pomocí technologie Amazon Alexa.

Klíčová slova Hlasové ovládání, inteligentní domy, Amazon Echo, Amazon Web Services, OAuth, Python

Abstract

The goal of this bachelor thesis is to develop a voice interface for controlling iQtec smart homes using Amazon Alexa technology. The thesis includes analysis and design of the entire system, implementation of skills for Amazon Echo in Python, deployment to Amazon Web Services Lambda servers, creation of an authorization server supporting the OAuth protocol and integration into an existing application iQtec Architect. Part of the thesis is also the realization of the open-source Askhome library to simplify the creation of smart home voice interfaces using Amazon Alexa technology.

Keywords Voice control, smart homes, Amazon Echo, Amazon Web Services, OAuth, Python

Obsah

Úvod **1**
Obsah práce 1

1 Analýza **3**
1.1 Existující řešení 3
1.2 Porovnání technologií hlasového ovládání 3
1.3 Amazon Echo 4
1.4 Analýza API služby Alexa 4
1.5 Analýza API inteligentních domů iQtec 5
1.6 Funkce inteligentních domů iQtec 6
1.7 Návrh hlasového rozhraní 7
1.8 Analýza požadavků 8

2 Návrh **11**
2.1 Platforma 11
2.2 Jazyk 12
2.3 Architektura 12
2.4 Rozdělení do částí 14

3 Integrace do aplikace iQtec Architekt **15**
3.1 Účely aplikace 15
3.2 Komponenty 15
3.3 Generování definice hlasového rozhraní 16
3.4 Implementace 17

4 Knihovna Askhome **21**
4.1 Požadavky a účely knihovny 21

4.2 Styl kódu . . . 22
4.3 Postup realizace . . . 22
4.4 Rozhraní knihovny . . . 23
4.5 Testování . . . 24
4.6 Průběžná integrace . . . 25
4.7 Dokumentace . . . 25
4.8 Zveřejnění balíčku . . . 26

5 Smart Home Skill **29**
5.1 Použité technologie . . . 29
5.2 Implementace . . . 30
5.3 Testování . . . 31
5.4 Automatizace nasazování . . . 32

6 Custom Skill **35**
6.1 Účel Custom Skill . . . 35
6.2 Custom Skill API . . . 35
6.3 Interakční model . . . 36
6.4 Použité technologie . . . 36
6.5 Implementace . . . 37

7 Autorizační server **39**
7.1 Protokol OAuth . . . 39
7.2 Možné řešení autorizace . . . 41
7.3 Použité technologie . . . 41
7.4 Implementace . . . 42
7.5 Nasazení . . . 43

8 Uživatelské testování **45**
8.1 Testovací instalace . . . 45
8.2 Průběh testování . . . 45
8.3 Výsledky a komentáře . . . 46

Závěr **47**

Literatura **49**

A Seznam použitých zkratek **53**

B Obsah přiloženého CD **55**

Seznam obrázků

2.1 Diagram architektury systému hlasového ovládání iQtec 13

3.1 Ukázka vytváření hlasového rozhraní domu pomocí aplikace iQtec Architekt 17

4.1 Logo knihovny Askhome 21
4.2 Ukázka uživatelské příručky knihovny Askhome vygenerované pomocí nástroje Sphinx 27

7.1 Průběh autorizace v protokolu OAuth 41
7.2 Webové rozhraní autorizačního serveru iQtec 42

Seznam tabulek

1.1 Možné příkazy a dotazy a jejich podpora v jednotlivých dovednostech. 9

Seznam zdrojových kódů

3.1 XML konfigurace komponenty ztmavovatelného světla pro aplikaci iQtec Architekt. 18
3.2 Vygenerované XML definice hlasového rozhraní inteligentního domu z aplikace iQtec Architekt. 19
4.1 Příklad použití knihovny Askhome 23
4.2 Ukázka použití výjimek knihovny Askhome 24
5.1 Třída ztmavovatelného světla s metodou nastavení jasu . . . 31
5.2 Konfigurace nástroje Zappa pro nasazování Smart Home Skill 33
6.1 Zpracování požadavku nastavení procenta pomocí knihovny Flask-Ask 37

Úvod

Popularita automatizovaných, tzv. inteligentních domů je na vzestupu. Koncept Internet of Things (dále IoT), tedy idea propojených zařízení z běžného života, se začíná stávat skutečností. Od chytrého kartáčku na zuby přes automatické zabezpečení dveří se i další chytrá zařízení v domácnosti stávají čím dál běžnější.

Nabízí se zde skloubení technologií IoT a hlasového ovládání, které je nyní již dostatečně spolehlivé a praktické pro využití všedními uživateli. Hlasové ovládání je jedno z nejpřirozenějších a nejpohodlnějších, a díky pokroku ve vývoji neuronových sítí a větší výpočetní síle se začínají objevovat aplikace v mnoha odvětvích, mimo jiné i u inteligentních domů.

Jedním takovým zařízením je Amazon Echo, komerční produkt využívající technologii hlasového rozpoznávání Alexa. Amazon Echo slouží pro ovládání IoT technologií v domácnostech, zároveň poskytuje pro vývojáře aplikační rozhraní (dále API), kterým lze Echo naučit pracovat s novými zařízeními.

Obsah práce

V této bakalářské práci se budu zabývat analýzou přívětivého a použitelného hlasového rozhraní pro inteligentní domy iQtec a jeho implementací pro Amazon Echo. V rešeršní části budu prozkoumávat různé přístupy pro hlasové rozhraní, zanalyzuji API služby Alexa a ovladačů domů iQtec a navrhnu architekturu celého systému. V praktické části budu popisovat implementaci dovedností pro Amazon Echo v jazyce Python. Součástí práce bude také integrace do aplikace iQtec Architekt pro jednoduché navrhování hlasového rozhraní. Celý systém poběží na serverech Amazon Web

Services Lambda, bude tedy třeba aplikaci nasadit na tyto servery a tento proces automatizovat. Kvůli nutnosti autorizace uživatelů uvnitř vytvářeného systému budu vytvářet autorizační server podporující protokol OAuth. Nakonec provedu uživatelské testování použitelnosti na fyzické instalaci.

Protože chci svojí prací podpořit komunitu vývojářů pro Amazon Echo, část kódu zveřejním jako otevřenou knihovnu Askhome, která usnadňuje vytváření hlasového rozhraní inteligentních domů se službou Alexa.

Kapitola 1

Analýza

V této kapitole popíšu argumentaci k výběru technologie Amazon Echo, analýzu API služby Alexa a vytyčím požadavky na vytvářený systém.

1.1 Existující řešení

Ve světě již existují systémy pro hlasové ovládání inteligentních domů (například CastleOS[1]), domy značky iQtec ale zatím žádnou formu hlasového ovládání nepodporují.

1.2 Porovnání technologií hlasového ovládání

Hlavními hráči na trhu technologií hlasového ovládání jsou Amazon Echo a Google Home. Zatímco Amazon Echo je více rozšířené [1], hlasové rozpoznávání a interakce se zdá mít lepší výsledky u Google Home [2]. Google Home ale otevřel podporu pro vývojáře až v prosinci 2016, a to bylo pro tento projekt příliš pozdě, část funkcionality pro Amazon Echo jsem totiž už měl naimplementovanou dříve.

Za zmínku stojí také projekt Jasper[2], otevřená platforma pro vývoj aplikací hlasového ovládání. S tím jsou ale spojeny problémy správy vlastních serverů a tvorby vlastního hardwaru. Komerční technologie mají také vyšší kvalitu svých algoritmů na rozpoznávání a syntézu řeči.

Pro můj projekt jsem se z těchto důvodů rozhodl pro Amazon Echo.

[1] https://www.castleos.com/

[2] https://jasperproject.github.io

1.3 Amazon Echo

Amazon Echo je zařízení ve tvaru 23 cm vysokého a 8 cm širokého válce obsahující výkonné pole sedmi mikrofonů a reproduktor. Zařízení je napájeno přes kabel a připojeno k internetu přes technologii Wi-Fi. Po propojení s účtem Amazon přes mobilní nebo webovou aplikaci lze Amazon Echo aktivovat slovem „Alexa“ a vyslovením požadavku. Amazon Echo dokáže obsloužit různé dotazy a úkoly jako vyhledávání informací na internetu, přehrávání hudby a mimo jiné i ovládání inteligentních domů. [3]

Pro můj projekt jsem používal levnější alternativu Amazon Echo Dot, která kromě slabšího reproduktoru a menších rozměrů je však totožná s plnou verzí Amazon Echo. [3]

1.4 Analýza API služby Alexa

Se službou hlasového rozpoznávání Alexa, na které je Amazon Echo založeno, lze pracovat pomocí třech různých API pro tvorbu dovedností. Popíšu zde jejich výhody a nevýhody a důvody pro použití v mém projektu.

1.4.1 Custom Skill API

Custom Skills API je hlavní a nejflexibilnější možnost vývoje funkcionality pro Amazon Echo. Pro vytvoření Custom Skill musí vývojář nejprve nadefinovat tzv. interakční model, tedy věty, které aktivují požadovanou akci. Amazon Echo po vyvolání akce pošle požadavek na servery AWS Lambda, kde vývojářův kód provede svou logiku ovládání zařízení a vrátí odpověď, kterou Amazon Echo sdělí uživateli.

Nevýhodou je, že pro použití Custom Skill uživatel musí vždy ve svém příkazu zmínit jméno dovednosti nebo přímo vstoupit pomocí věty: „Alexa, open iQtec.“ Příkaz pro rozsvícení kuchyňského světla by mohl vypadat takto:

> „Alexa, tell iQtec to switch on the kitchen light.“

1.4.2 Smart Home Skill API

Pro podporu základních funkcí inteligentního domu, jako světla nebo vytápění, Amazon vytvořil API pro jednodušší implementaci oproti Custom Skills API. S předpřipraveným interakčním modelem vývojář naimplementuje pouze logiku ovládání svých zařízení a vše ostatní se vyřeší za něj.

Nevýhodou ovšem je, že API je omezené, a tak například dotazy na meteostanici obsloužit nejdou.

Stejný dotaz na rozsvícení světla přes Smart Home Skill by mohl být jednodušší:

„Alexa, switch on the kitchen light.“

1.4.3 Flash Briefing API

Amazon Echo poskytuje tzv. flash briefing, což je přehled aktuálních novinek a událostí. Pro přidání svých zpráv mohou vývojáři použít Flash Briefing API. Tuto funkcionalitu ale v mém projektu nevyužiji.

1.4.4 Použité řešení

Jednoduchost a robustnost Smart Home API je lákavá, ale takové řešení by nepokrylo všechny zařízení iQtec. Zároveň je tvorba interakčního modelu pro Custom Skill obtížná a nevytvořil bych ho tak dobře jako Amazon. Rozhodl jsem se proto použít obě varianty s co největším využitím Smart Home API a zbytek případů pokrýt pomocí Custom Skills API.

1.5 Analýza API inteligentních domů iQtec

PLC[3], neboli ovládací jednotka, domů iQtec poskytuje dvě možná rozhraní na posílání požadavků.

1.5.1 HTTP API

HTTP API je jednoduché rozhraní využívající parametrů URL, pro čtení a zápis nějaké hodnoty stačí znát adresu požadované proměnné a sestavit podle ní dotazový řetězec. Adresy proměnných se u každého domu liší, a proto je třeba pro každý znát příslušnou paměťovou strukturu PLC.

1.5.2 UDP API

Druhou možností je UDP API, které se používá například při připojení PLC přes mobilní síť GSM. Použití UDP API je o něco složitější, například pro identifikaci proměnné se používá hašovací funkce CRC.

[3]Programmable Logic Controller

1.5.3 Použité řešení

Protože v mém projektu budu využívat server GSM Bridge, který spravuje připojení jednotek a převádí mezi jednotlivými rozhraními, budu pracovat pouze s HTTP API.

1.6 Funkce inteligentních domů iQtec

V této části vyčtu všechny možnosti funkcí domů iQtec, které by mohly být hlasově ovládány.

Ovládání osvětlení

1. Zapnutí a vypnutí světel v místnosti nebo v celém domě.
2. Vypnutí světel se zpožděním.
3. Nastavení jasu světel.
4. Výběr scénáře osvětlení. Scénáře jsou přednastavené a pojmenované konfigurace světel. Například scénář „vaření“ by mohl zvýšit jas světel nad kuchyňskou deskou.
5. Nastavení módu zapínání náhodných světel. Tento mód slouží jako bezpečnostní opatření při opuštění domu pro vytvoření dojmu, že uvnitř stále někdo je.

Ovládání a dotazy na vytápění

1. Nastavení teploty na danou hodnotu.
2. Zvýšení nebo snížení teploty o danou hodnotu.
3. Nastavení automatického módu termostatu.
4. Zapnutí a vypnutí klimatizace.
5. Nastavení otáček fan-coilů (jednotek klimatizace).
6. Zjištění teploty a vlhkosti v místnosti.

Ovládání žaluzií

1. Stáhnutí nebo vytáhnutí žaluzií.
2. Posunutí žaluzií o mikro krok nahoru nebo dolů.
3. Nastavení úhlu žaluzií směrem na slunce.
4. Nastavení scénáře žaluzií. Tyto scénáře fungují podobně jako u osvětlení.

Dotazy na stav domu

1. Dotaz na otevřená okna.
2. Dotaz na zamčené dveře.
3. Dotaz na zapnutá světla.
4. Dotaz na celkovou spotřebu energie.
5. Dotaz na data o počasí z meteostanice, jako je například síla větru nebo intenzita slunečního záření.
6. Dotaz na celkový stav systémů domu.

Ostatní funkce

Domy iQtec umožňují další řadu funkcí, ty jsem ale nezmínil kvůli nepraktičnosti použití pomocí hlasového rozhraní. Například funkce kalendáře umožňuje plánovat chování systémů v budoucnu, nastavování je ale uživatelsky přívětivější v aplikaci pro osobní počítače.

1.7 Návrh hlasového rozhraní

1.7.1 Požadavky na rozhraní

Hlasové ovládání domů iQtec nemá nahradit ostatní ovládací rozhraní ani být jejich plnohodnotnou alternativou. Hlavním účelem projektu je zjednodušit a zpříjemnit zákaznickou interakci se systémy iQtec. Proto jsem při návrhu interakčního modelu upřednostnil intuitivnost a jednoduchost nad možnostmi nastavování všech různých parametrů.

1.7.2 Výsledný návrh

Všechny možné příkazy pro jednotlivé dovednosti, které jsem navrhl, jsou vypsány v tabulce 1.1. Jak je vidět, většina akcí je podporována v obou dovednostech. Custom Skill dokáže obsloužit všechny, až na akci vyhledání všech zařízení, což je způsob inicializace Smart Home Skill.

Díky velkému pokrytí možných akcí a jednodušší interakci je dovednost Smart Home Skill míněna jako primární způsob hlasového ovládání a Custom Skill jen jako doplněk pro komplexnější dotazy.

1.8 Analýza požadavků

1.8.1 Role

V systému budou vystupovat dvě role:

Uživatel Obyvatel inteligentního domu iQtec, který bude přes Amazon Echo udávat příkazy k ovládání domu.

Architekt Tvůrce hlasového rozhraní konkrétního domu. Bude určovat, která zařízení a pod jakým jménem se budou ovládat.

1.8.2 Funkční požadavky

Tvorba hlasového rozhraní Architekt může určit pomocí grafického nástroje, která zařízení iQtec budou moci být hlasově ovládaná, které vlastnosti bude zákazník moci nastavovat a pod jakým jménem se na zařízení bude odkazovat.

Autentizace uživatele Uživatel se může přihlásit na svůj účet iQtec přes aplikaci Alexa.

Hlasové ovládání Uživatel může hlasově ovládat zařízení svého domu, jak je popsáno v tabulce 1.1.

1.8.3 Nefunkční požadavky

Snadná rozšiřitelnost Systém bude snadné rozšířit o podporu dalších zařízení a modifikaci hlasového rozhraní.

Automatizace nasazování Nasazení nové verze bude automatizované a jednoduché.

Akce	Příklad příkazu	SHS[a]	CS[b]
Vyhledání všech zařízení	Discover my appliances	✓	
Zapnutí světla	Turn on light.	✓	✓
Vypnutí světla	Turn off light.	✓	✓
Nastavení jasu světla	Set light brightness to 50%.	✓	✓
Zvýšení jasu světla	Brighten light by 10%.	✓	✓
Snížení jasu světla	Dim light by 10%.	✓	✓
Nastavení scénáře osvětlení	Turn on bedtime scene.	✓	✓
Zapnutí náhodných světel	Turn on random lights.	✓	✓
Vypnutí všech světel	Turn off all lights.	✓	✓
Nastavení teploty	Set living room to 21°.	✓	✓
Zvýšení teploty	Increase temperature by 1°.	✓	✓
Snížení teploty	Decrease temperature by 1°.	✓	✓
Zapnutí klimatizace	Turn on AC.	✓	✓
Vypnutí klimatizace	Turn off AC.	✓	✓
Nastavení otáček fan-coilů	Set fan-coil speed 3.		✓
Zjištění nastavené teploty	What is kitchen set to?	✓	✓
Zjištění aktuální teploty	Tell me kitchen temperature.	✓	✓
Zjištění aktuální vlhkosti	Tell me kitchen humidity.		✓
Vytáhnutí žaluzií	Put bedroom blinds up.	✓[c]	✓
Stáhnutí žaluzií	Put bedroom blinds down.	✓[d]	✓
Posunutí žaluzií nahoru	Raise bedroom blinds.	✓	✓
Posunutí žaluzií dolů	Lower bedroom blinds.	✓	✓
Nastavení úhlu na slunce	Tilt blinds towards the sun.		✓
Nastavení scénáře žaluzií	Turn on blinds noon scene.	✓	✓
Dotaz na otevřená okna	Are any windows open?		✓
Dotaz na zamčené dveře	Is the front door locked?	✓	✓
Dotaz na rozsvícená světla	Are any lights on?		✓
Dotaz na celkovou spotřebu	Tell me current consumption.		✓
Dotaz na meteostanici	What is the weather like?		✓
Dotaz na stav domu	What is the house status?		✓
Seznam všech zařízení	List all appliances.		✓

Tabulka 1.1: Možné příkazy a dotazy a jejich podpora v jednotlivých dovednostech.

[a] Podpora ve Smart Home Skill

[b] Podpora v Custom Skill

[c] Ve Smart Home Skill se příkaz aktivuje pomocí „Turn on“, což je anglicky nekorektní, ale lepší než úplné vynechání funkcionality

[d] Viz poznámka c

Spolehlivost Systém bude stabilní a nebude docházet k nestandardnímu chování. Případné chyby budou zapisovány do logů.

Škálovatelnost Průměrná odezva systému bude rozumně dlouhá. Systém se bude přizpůsobovat zatížení a automaticky se škálovat.

Bezpečnost Veškerá komunikace pro ovládání inteligentních domů bude probíhat přes zabezpečený protokol HTTPS. Uživatel se bude autorizovat pomocí protokolu OAuth.

Kapitola 2

Návrh

V této kapitole se zaměřuji na architekturu celého systému a rozdělení do jednotlivých částí, které rozeberu detailněji dále.

2.1 Platforma

U typu platformy jsem neměl na výběr. Custom Skill sice může být hostován na externím serveru, Smart Home Skill ale musí být nasazen na serverech Amazonu AWS Lambda [4]. Výhodou je, že pro vývoj aplikací Alexa Amazon poskytuje hosting zdarma.

2.1.1 Amazon Web Services

Amazon Web Services (AWS) je rozsáhlá sbírka cloudových služeb jako výpočetní platformy, ukládání statických dat, databáze nebo síťové technologie. AWS je zdaleka nejpoužívanějším veřejným cloudovým poskytovatelem s podílem uživatelů 57% [5]. AWS operuje v 16 regionech po světě pro nejlepší dostupnost a odezvu, služby Alexa jdou bohužel využívat pouze ve třech, a to podle regionální verze Amazon Echo. [6]

2.1.2 AWS Lambda

AWS Lambda je výpočetní platforma, která umožňuje spouštět kód bez nastavování a spravování serverů. Tato služba je založena na konceptu architektury serverless, což znamená vytváření nového prostředí při každém požadavku. AWS Lambda tedy z uživatelského pohledu nemá žádnou trvalou infrastrukturu a vše je spravováno na straně Amazonu. Díky tomu se instance automaticky škálují podle využití. Výsledné rozhraní je čistě

funkcionální, kdy vstupním bodem je funkce s požadavkem jako argument a výstupem je její návratová hodnota. [7]

2.2 Jazyk

AWS Lambda prozatím podporuje pouze čtyři jazyky a to Node.js (JavaScript), Python 2.7, Java 8 a C# [7]. Z těchto možností je mi nejbližší Python i přes jeho brzy zastaralou verzi 2.7, která původně vyšla až v roce 2010. Python je zároveň používán v jiných projektech iQtec, a proto jsem pro programování Smart Home a Custom Skill zvolil tento jazyk.

2.3 Architektura

Architektura celého systému hlasového ovládání domů iQtec je znázorněna na obrázku 2.1. Tento diagram je zjednodušený a některé komponenty jsou pro přehlednost vynechány.

2.3.1 Popis komponent

PLC iQtec Ovladač zařízení inteligentního domu připojený k internetu nebo mobilní síti pomocí modulu GSM.

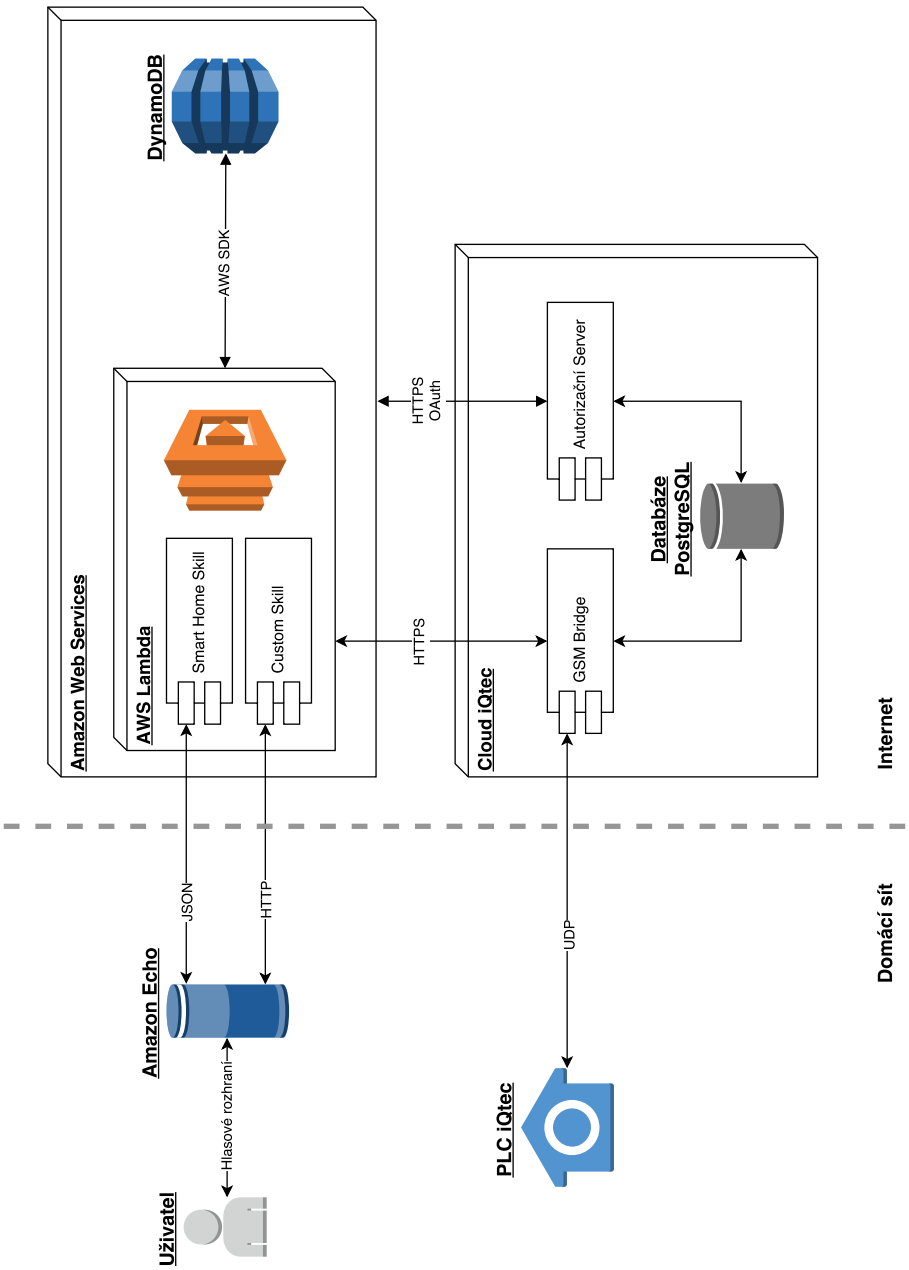
GSM Bridge Server, který se stará o přeposílání komunikace mezi PLC a ostatními službami.

Autorizační server Server zajišťující přihlašování uživatelů a poskytování přístupových tokenů k ovládání domu.

Databáze PostgreSQL Databáze pro uložení definic hlasových rozhraní, ke kterým se přistupuje přes GSM Bridge, a údajů o uživatelích pro autorizační server.

Smart Home a Custom Skill Jednotlivé dovednosti, které jsou nasazeny jako oddělené instance AWS Lambda. Zpracovávají požadavky od Amazon Echo a ovládají PLC domu přes GSM Bridge.

DynamoDB AWS databáze, která zde slouží jako cache definic hlasového rozhraní jednotlivých domů.



Obrázek 2.1: Diagram architektury systému hlasového ovládání iQtec

2.3.2 Průběh požadavku

1. Uživatel vysloví příkaz, který Amazon Echo zachytí a pošle službě Alexa na servery AWS pro hlasové rozpoznání.

2. Zpracovaný příkaz se podle typu předá Smart Home nebo Custom Skill v odpovídajícím formátu JSON nebo HTTP.

3. Dovednost pošle požadavek k ovládání domu na GSM Bridge s přístupovým tokenem identifikující uživatele.

4. GSM Bridge zkontroluje přístupový token a přepošle požadavek na PLC podle přípojení přes síť GSM nebo internet.

5. PLC provede požadovanou akci a odpoví s výslednou hodnotou zpět na GSM Bridge, který ji přepošle zpět dovednosti.

6. Smart Home nebo Custom Skill vytvoří podle výsledku odpověď a předá ji službě Alexa.

7. Zvukovou odpověď vytvořenou pomocí syntézy řeči pošle služba Alexa zpět na Amazon Echo, které ji přehraje uživateli.

2.4 Rozdělení do částí

V následujících kapitolách se budu detailněji zabývat jednotlivými částmi systému, které jsem realizoval. Nejprve se zaměřím na tvorbu hlasového rozhraní v nástroji iQtec Architekt. V další kapitole se budu zabývat otevřenou knihovnou Askhome, která usnadňuje práci se Smart Home API. Následně popíšu realizaci jednotlivých dovedností Smart Home Skill a Custom Skill. Nakonec rozeberu tvorbu autorizačního serveru a protokol OAuth, se kterým server pracuje.

Kapitola 3

Integrace do aplikace iQtec Architekt

V této kapitole popíšu moje úpravy aplikace iQtec Architekt pro navrhování hlasového rozhraní inteligentního domu.

3.1 Účely aplikace

Nástroj iQtec Architekt je aplikace pro operační systém Windows napsaná v jazyce C#, která slouží k návrhu systémů inteligentních domů iQtec. Aplikace má následující funkce:

- Definice všech zařízení inteligentního domu ovládaných systémem iQtec, kterou využívají další prvky sestavovacího procesu.
- Vyskládání paměti centrálního PLC iQtec pro ovládání a správu zařízení.
- Vytvoření rozhraní inteligentního domu ze strany uživatele pro aplikaci iQtec Asistent na operačním systému Windows.
- Definice HTTP a UDP rozhraní ovládání domu pro ostatní systémy iQtec.

3.2 Komponenty

Návrh inteligentního domu v aplikaci iQtec Architekt se vytváří pomocí tzv. komponent, které se umisťují do plánu domu. Každé komponentě jdou

zadat různá nastavení podle typu, například komponentě ovladače žaluzií lze nastavit časovou délku kroku nebo naklonění žaluzie.

Každá komponenta má svůj definiční soubor ve formátu XML, který obsahuje informace o typu komponenty, nastavitelných atributech v aplikaci nebo definice paměťových nároků na PLC pro použití instance komponenty.

3.2.1 Komponenty Alexa

Pro navrhování hlasového rozhraní jsem pro iQtec Architekt vytvořil několik nových komponent typu Alexa. Každá komponenta značí jeden typ hlasově ovladatelného zařízení. Instance komponenty drží data o aktivačním jménu zařízení, odkazech na proměnné ovládajících stav zařízení a několik dalších konfiguračních detailů jako velikost kroku při inkrementaci nebo dekrementaci procentuální hodnoty.

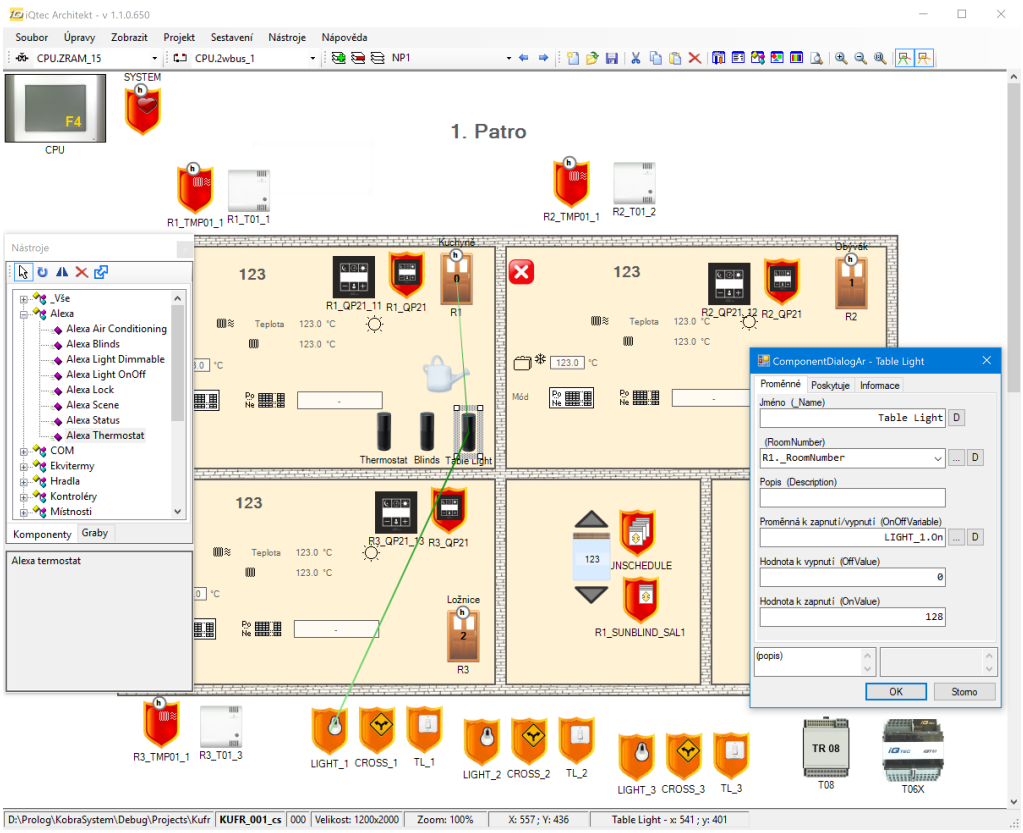
Každá komponenta typu Alexa také může být připojena na komponentu místnosti s položkou anglického názvu, tyto názvy se po sestavení předřadí aktivačnímu jménu zařízení. Například zařízení světla se jménem Table Light připojené na místnost Kitchen se bude aktivovat pomocí jména Kitchen Table Light. Tímto způsobem se může změnit jméno celé místnosti na jednom místě a nemusí se přejmenovávat každá komponenta zvlášť.

Ukázková XML konfigurace komponenty typu Alexa je vidět ve zdrojovém kódu 3.1, použití Alexa komponent uvnitř iQtec Architekta je znázorněno na obrázku 3.1.

3.3 Generování definice hlasového rozhraní

Po vytvoření komponent Alexa, iQtec Architekt při sestavení projektu vygeneruje definiční soubor ve formátu XML, který popisuje hlasové rozhraní všech zařízení domu. Tento soubor je využíván ostatními službami, jako Smart Home Skill, které podle něj vytvářejí požadavky na PLC pro ovládání zařízení. Zdrojový kód 3.2 ukazuje formát tohoto výstupního souboru.

Každý XML tag `appliance` uvnitř kořene značí jedno zařízení a jeho atributy nesou potřebná data o něm. Při porovnání zdrojových kódů 3.1 a 3.2 je vidět, jak každý výsledný atribut ztmavovatelného světla odpovídá položkám `variable` komponenty. Hodnoty atributů se přímo zadávají do formulářů v aplikaci. Výjimkou jsou speciální atributy `id` a `type`, které vznikly textovou transformací ze jména a typu komponenty. Tento přístup je výhodný, protože pro přidaní další Alexa komponenty se nemusí modifikovat zdrojový kód iQtec Architekta, ale stačí vytvořit nové XML komponenty.



Obrázek 3.1: Ukázka vytváření hlasového rozhraní domu pomocí aplikace iQtec Architekt

3.4 Implementace

Implementace generování definičního souboru spočívala v přidání logiky do funkce sestavení aplikace iQtec Architekt. V této funkci můj kód najde všechny Alexa komponenty pomocí dotazů LINQ. LINQ je součást .NET Frameworku jazyka C#, která umožňuje přímočaře získávat a transformovat data podobně jako jazyk SQL [8]. S LINQ stačí jednoduše vyfiltrovat všechny komponenty projektu metodou `Where` a transformovat pomocí metody `Select` do žádaného formátu.

Pro zápis XML souboru jsem použil novější třídu `XDocument`, která na rozdíl od staršího postupu s `XmlDocument` bohatě podporuje dotazy LINQ ve svém API k tvorbě XML souborů [9]. Díky tomu jsem mohl data z přetransformovaných Alexa komponent přímo namapovat na atributy XML elementů.

```xml
<?xml version="1.0" encoding="utf-8"?>
<component>

  <property name="ID" value="Alexa Light Dimmable" />
  <property name="Class" value="alexa" />
  <property name="Category" value="alexa" />
  <property name="Info" value="Alexa ovladač ztmavovatelného
  ↪ světla" />

  <property name="IconFile" value="alexa.png" />
  <property name="IconWidth" value="21" />
  <property name="IconHeight" value="50" />

  <!-- Common Alexa variables -->
  <variable name="_Name" info="Aktivační jméno"
  ↪ type="string128" value="Light" access="P" />
  <variable name="RoomNumber" type="roomnumber"
  ↪ requires="room" value="CPU.DummyRoom" access="P"/>
  <variable name="Description" info="Popis" type="string128"
  ↪ value="" access="P" />

  <!-- Custom Alexa variables -->
  <variable name="OnOffVariable" info="Proměnná k
  ↪ zapnutí/vypnutí" type="variable" access="P" />
  <variable name="OffValue" info="Hodnota k vypnutí"
  ↪ type="byte" value="0" access="P" />
  <variable name="OnValue" info="Hodnota k zapnutí"
  ↪ type="byte" value="128" access="P" />
  <variable name="PercentageVariable" info="Proměnná k
  ↪ ovládání jasu" type="variable" access="P" />
  <variable name="MaxPercentageValue" info="Maximální hodnota
  ↪ jasu" type="byte" value="100" access="P" />

</component>
```

Zdrojový kód 3.1: XML konfigurace komponenty ztmavovatelného světla pro aplikaci iQtec Architekt.

```xml
<?xml version="1.0" encoding="utf-8"?>
<data>
  <appliance id="kitchen_table_light"
  ↪ type="alexa_light_onoff" name="Kitchen Table Light"
  ↪ onoffvariable="1/6/2" offvalue="1" onvalue="2"/>
  <appliance id="bedroom_light" type="alexa_light_dimmable"
  ↪ name="Bedroom Light" onoffvariable="1/5/2" offvalue="1"
  ↪ onvalue="2" percentagevariable="1/5/13"
  ↪ maxpercentagevalue="100"/>
  <appliance id="bedtime_scene" type="alexa_scene"
  ↪ name="Bedtime Scene" controlvariable="1/10/2"
  ↪ onvalue="7"/>
  <appliance id="all_lights" type="alexa_light_onoff"
  ↪ name="All Lights" onoffvariable="1/10/2" offvalue="1"
  ↪ onvalue="2"/>
  <appliance id="bedroom" type="alexa_thermostat"
  ↪ name="Bedroom" curtempvariable="1/16/10"
  ↪ curhumidityvariable="1/16/14"
  ↪ targettempvariable="1/16/30"
  ↪ controlvariable="1/16/58"/>
  <appliance id="blinds" type="alexa_blinds" name="Blinds"
  ↪ controlvariable="1/12/5" percentagevariable="1/0/6"
  ↪ maxpercentagevalue="100"/>
</data>
```

Zdrojový kód 3.2: Vygenerované XML definice hlasového rozhraní inteligentního domu z aplikace iQtec Architekt.

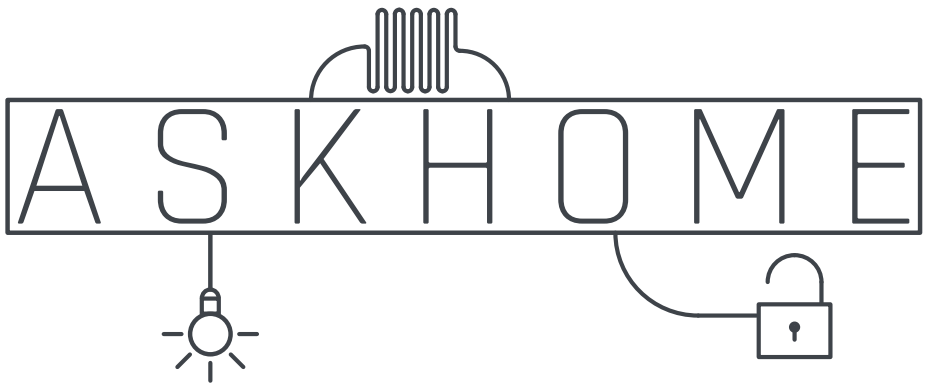
Kapitola 4

Knihovna Askhome

V této kapitole rozeberu návrh a implementaci otevřené knihovny pro Python Askhome.

4.1 Požadavky a účely knihovny

Hlavním účelem knihovny je zjednodušit práci se Smart Home API. Protože Smart Home API je rozhraní využívající formátu JSON, práce s ním znamená velké množství kódu, které zpracovává a generuje JSON data. Například ukázkový kód z oficiální dokumentace Smart Home API [10] obsahuje 60 řádek pro zpracování a vytváření odpovědi na požadavek a neobsa-



Obrázek 4.1: Logo knihovny Askhome

huje žádnou logiku ovládání zařízení. Při mém původním pokusu vytvářet Smart Home Skill bez jakékoli pomocné knihovny, kód rychle narostl do nepřehledné změti příkazů pro práci se surovými JSON daty.

Z těchto důvodů jsem si na knihovnu Askhome vytyčil následující požadavky:

- Jednoduchá definice ovládaných zařízení a jejich poskytovaných akcí.
- Zpracování JSON požadavků do objektů.
- Možnost vytvoření odpovědí pomocí metod objektu požadavku nebo vyvolání výjimek.

4.2 Styl kódu

Výhodou zveřejnění knihovny Askhome je, že komunita vývojářů pro Amazon Echo může kód vylepšovat. Protože je repozitář nahraný na serveru GitHub[4], může kdokoli zaslat tzv. pull request, tedy žádost o přijetí vylepšení kódu do hlavní verze. Kvůli možnosti modifikace knihovny ostatními vývojáři jsem se snažil psát kód co nejpřehledněji a nejčistěji.

Výsledný kód jsem proto psal podle oficiální stylové příručky pro Python PEP 8 [11] a podle dodatků ve stylové příručce od společnosti Google [12]. Dodržování pravidel jsem kontroloval pomocí nástroje pylint[5].

Při návrhu rozhraní knihovny jsem postupoval podle rad v knize Hitchhiker's Guide to Python [13] od Kennetha Reitze, autora velmi populární knihovny Requests[6].

4.3 Postup realizace

Při realizaci Askhome jsem postupoval podle metody Test-driven development, tedy přístupu k vývoji softwaru, kde prvním krokem je vždy napsání testu místo samotného kódu. Tímto způsobem jsem mohl nejdříve napsat test s použitím knihovny a měnit rozhraní, dokud jsem s ním nebyl spokojen. Až potom jsem přistoupil k samotné implementaci, kdy už bylo v testech předepsáno, jak se má kód chovat. Výhodou tohoto přístupu byla jednoduchá možnost měnit rozhraní knihovny, zároveň psaním příkladů použití v testech jsem mohl dobře odhalit možné budoucí problémy rozhraní.

[4] https://github.com/mathead/askhome
[5] https://www.pylint.org
[6] http://python-requests.org

4.4 Rozhraní knihovny

Výsledné rozhraní popíšu na ukázce zdrojového kódu 4.1.

```
1  from askhome import Appliance, Smarthome
2
3  class Light(Appliance):
4      @Appliance.action
5      def turn_on(self, request):
6          pass  # Logika pro zapnutí světla
7
8  home = Smarthome()
9  home.add_appliance('light1', Light, name='Kitchen Light')
10
11 lambda_handler = home.lambda_handler
```

Zdrojový kód 4.1: Příklad použití knihovny Askhome

Tento příklad je plně funkčním kódem pro Smart Home Skill se stejnou funkcionalitou jako 60 řádkový kód z oficiální dokumentace [10]. Na řádce 3 se definuje zařízení `Light` s jednou možnou akcí zapnutí. Akce se označí pomocí dekorátoru `@Appliance.action` a typ akce se odvodí ze jména metody. Inspiroval jsem se webovým frameworkem CherryPy[7], kde se podobně podle jmen metod určí URL webového pohledu.

Na řádce 9 se přidá instance ovladatelného světla v kuchyni do objektu domu `Smarthome`. Na následující řádce se už jen vystaví reference na funkci, která bude sloužit jako vstupní bod pro stroj AWS Lambda. Knihovna Askhome dokáže z těchto definic odpovědět na požadavek `DiscoverAppliancesRequest`, tedy zjištění všech podporovaných zařízení. Na požadavek `TurnOnRequest`, tedy zapnutí určitého zařízení, Askhome zavolá metodu `turn_on` se zpracovaným objektem požadavku `request`.

Askhome také umožňuje odpovědět na požadavky pomocí metod objektu třídy `Request`, který podle typu požadavku poskytuje co nejjednodušší rozhraní. Například při tvorbě odpovědí na požadavky typu `GetTargetTemperatureRequest`, tedy zjištění nastavení teploty termostatu, metoda `response` přijímá nepovinné argumenty jako aktuální mód nebo rozmezí cílové teploty.

Pokud při vykonávání řídící logiky nastane nějaká chyba nebo uživatel zadá nepřípustné hodnoty, stačí vyvolat příslušnou výjimku. Například

[7] http://cherrypy.org

v ukázce zdrojového kódu 4.2 se kontroluje, zda uživatel požádal o nastavení teploty termostatu v rozmezí 15 až 25 °C. Vyvoláním výjimky se zajistí uživatelsky přívětivá odpověď jednotky Amazon Echo, v tomto případě:

„I can only set the temperature between 15 and 25 degrees.“

```
1 from askhome.exceptions import ValueOutOfRangeError
2
3 class Heater(Appliance):
4     @Appliance.action
5     def set_target_temperature(self, request):
6         if request.temperature not in range(15, 25):
7             raise ValueOutOfRangeError(15, 25)
```

Zdrojový kód 4.2: Ukázka použití výjimek knihovny Askhome

4.5 Testování

Výhodou aplikací nasazených na AWS Lambda je, že mají jeden jasný vstup a výstup ve formátu JSON. Díky tomu se velmi dobře testují. Většina jednotkových testů Askhome je založena na skutečných JSON dotazech Smart Home API z oficiální dokumentace [14]. Tím se zajistí nejen správnost logiky uvnitř kódu knihovny, ale i správnost interakce s okolím.

Pro testování jsem místo obvyklého nástroje ze standardní knihovny Pythonu unittest[8] použil testovací framework pytest[9]. Pytest má oproti unittest několik výhod jako detailnější výpisy výsledku testů, jednodušší definice testů a testování výroků pomocí konstrukce `assert`.

4.5.1 Pokrytí kódu testy

Dalším nástrojem, který jsem využil během testování, je Coverage.py[10]. Tento balíček slouží k měření pokrytí kódu testy. Funguje tak, že při průběhu testování pro každou řádku testovaného kódu zkontroluje, zda byla interpretrem vykonána, nebo ne. Tímto způsobem lze ověřit, jestli testování vyzkouší opravdu všechny možnosti a nezapomíná například na nějaký podmíněný blok kódu. Procentuální vyjádření pokrytí u jednotlivých

[8] https://docs.python.org/2/library/unittest.html
[9] http://pytest.org
[10] http://coverage.readthedocs.io

souborů může pomoci najít místa, kde testy chybí. Stoprocentní pokrytí sice neznamená absolutní správnost kódu, ale jistě indikuje vysokou kvalitu testů [15].

Kvůli tomu, že knihovna Askhome je zamýšlena pro využití veřejností, rozhodl jsem se pro stoprocentní pokrytí. Mezi otevřenými knihovnami je to známka kvality, která může nalákat některé vývojáře.

4.6 Průběžná integrace

Průběžná integrace podle Martina Fowlera znamená pravidelné spouštění sestavovacího procesu projektu, často ověřeném automatickými testy [16]. U knihovny Askhome jsem se rozhodl pro průběžnou integraci kvůli jednoduchému ověření funkčnosti výše zmíněných požadavků pull request. Když je na server GitHub zaslán pull request pro Askhome, průběžná integrace prověří bezchybnou instalaci balíčku, úspěšný výsledek automatických testů a stoprocentní pokrytí testy. Pokud se nesplní některý z těchto požadavků, pull request se odmítne a neprojde systémem, dokud žadatel problémy neopraví.

Pro Askhome jsem zvolil službu průběžné integrace Travis CI[11]. Tato služba se snadno propojí s repozitářem na serveru GitHub a pro otevřené projekty je zdarma. Travis CI jsem navíc ještě zkombinoval se službou codecov[12], která dokáže generovat statistiky o pokrytí kódu testy.

4.7 Dokumentace

Kvalitní dokumentace otevřeného projektu je velmi důležitá, bez ní by použití cizím vývojářem znamenalo časově náročné prozkoumávání zdrojového kódu. Dokumentace by měla být obsáhlá, ale zároveň stručná a přehledná.

4.7.1 Uživatelská příručka

Pro knihovnu Askhome jsem vytvořil uživatelskou příručku pomocí nástroje Sphinx[13]. Sphinx slouží ke generování dokumentace primárně softwarových projektů napsaných v jazyce Python a dokáže automaticky vytvořit přehled API z dokumentačních řetězců. Zdrojové soubory Sphinx jsou formátovány

[11] https://travis-ci.org/mathead/askhome
[12] https://codecov.io/gh/mathead/askhome
[13] http://www.sphinx-doc.org

značkovacím jazykem reStructuredText, pomocí kterých lze vytvořit dokumentaci ve formátech HTML, PDF, LaTeX a dalších. Ukázka vygenerované dokumentace ve formátu HTML je vidět na obrázku 4.2.

4.7.2 Dokumentace ve zdrojovém kódu

Pro dokumentační řetězce ve zdrojovém kódu Askhome jsem dal přednost standardu Google [17] před standardem Sphinx, protože je podle mého názoru výrazně čitelnější. Například výčet parametrů se neřeší přes speciální značky `:param:`, ale odsazováním, které více odpovídá stylu jazyka Python. Pro podporu těchto dokumentačních řetězců bylo potřeba přidat do Sphinx rozšíření napoleon[14].

4.7.3 Hosting dokumentace

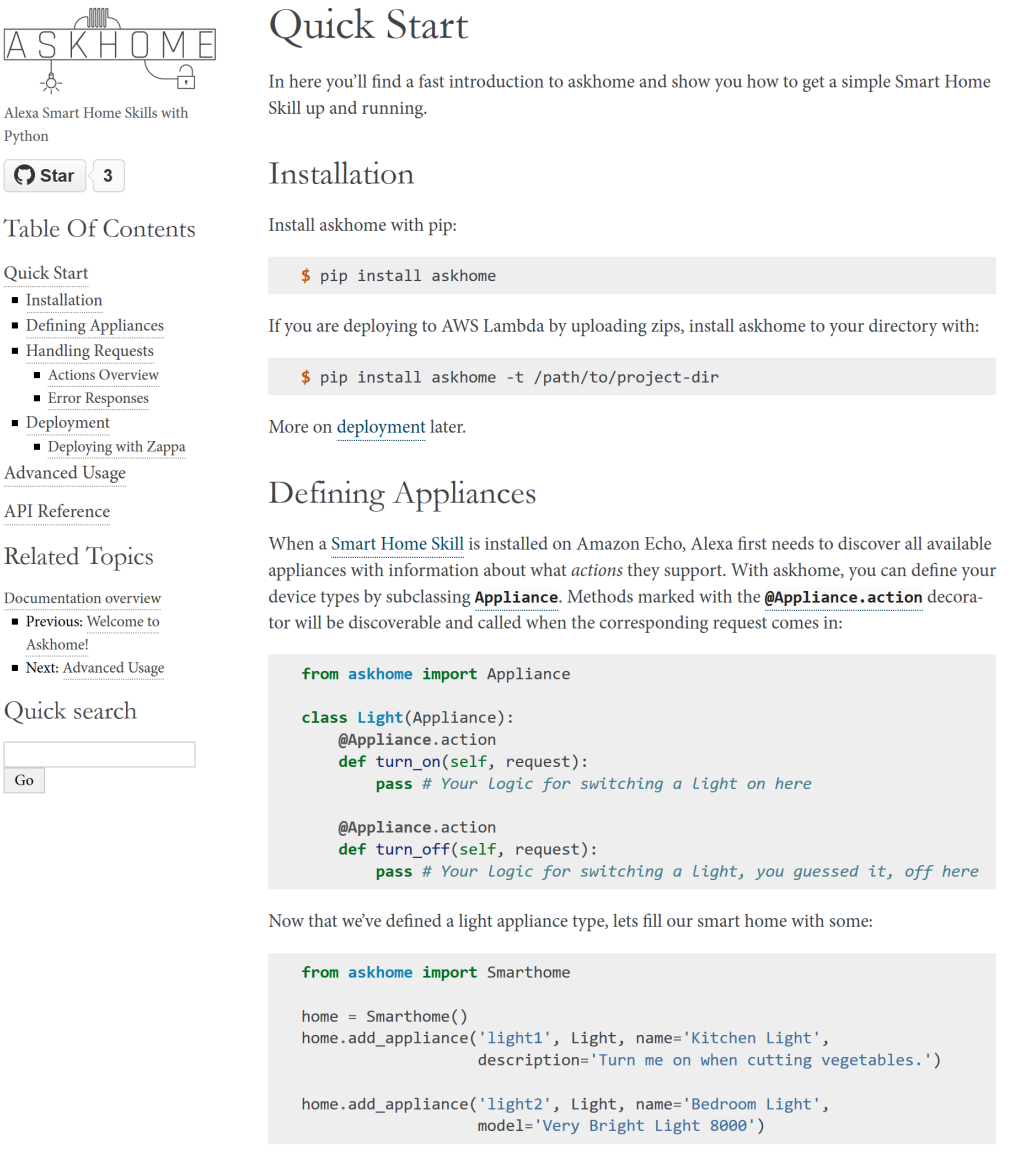
Pro mojí dokumentaci jsem využil službu Read the Docs[15], která poskytuje hosting zdarma pro otevřené projekty. Read the Docs se umí také propojit se serverem GitHub tak, že při každé změně zdrojových souborů dokumentace se automaticky spustí sestavení na serverech Read the Docs. Tímto způsobem se nemusí nijak řešit nasazování dokumentace, služba Read the Docs vše obstará sama.

4.8 Zveřejnění balíčku

Python Package Index (zkráceně PyPI) je oficiální repozitář balíčků pro jazyk Python. Pro zveřejnění knihovny Askhome na PyPI bylo potřeba vytvořit strukturu balíčku, což znamenalo mimo jiné vytvoření souboru setup.py obsahujícího metadata o balíčku jako popis, verze nebo požadované závislosti. Po nahrání balíčku na PyPI je nyní možné Askhome nainstalovat pomocí nástroje pip, který je zahrnutý v instalacích jazyka Python, následovně:

```
$ pip install askhome
```

[14] http://www.sphinx-doc.org/en/stable/ext/napoleon.html
[15] http://askhome.readthedocs.io



ASKHOME

Alexa Smart Home Skills with Python

Star 3

Table Of Contents

Quick Start
- Installation
- Defining Appliances
- Handling Requests
  - Actions Overview
  - Error Responses
- Deployment
  - Deploying with Zappa

Advanced Usage

API Reference

Related Topics

Documentation overview
- Previous: Welcome to Askhome!
- Next: Advanced Usage

Quick search

Go

Quick Start

In here you'll find a fast introduction to askhome and show you how to get a simple Smart Home Skill up and running.

Installation

Install askhome with pip:

```
$ pip install askhome
```

If you are deploying to AWS Lambda by uploading zips, install askhome to your directory with:

```
$ pip install askhome -t /path/to/project-dir
```

More on deployment later.

Defining Appliances

When a Smart Home Skill is installed on Amazon Echo, Alexa first needs to discover all available appliances with information about what *actions* they support. With askhome, you can define your device types by subclassing **`Appliance`**. Methods marked with the **`@Appliance.action`** decorator will be discoverable and called when the corresponding request comes in:

```
from askhome import Appliance

class Light(Appliance):
    @Appliance.action
    def turn_on(self, request):
        pass # Your logic for switching a light on here

    @Appliance.action
    def turn_off(self, request):
        pass # Your logic for switching a light, you guessed it, off here
```

Now that we've defined a light appliance type, lets fill our smart home with some:

```
from askhome import Smarthome

home = Smarthome()
home.add_appliance('light1', Light, name='Kitchen Light',
                   description='Turn me on when cutting vegetables.')

home.add_appliance('light2', Light, name='Bedroom Light',
                   model='Very Bright Light 8000')
```

Obrázek 4.2: Ukázka uživatelské příručky knihovny Askhome vygenerované pomocí nástroje Sphinx

Kapitola 5

Smart Home Skill

V této kapitole se budu zabývat realizací Smart Home Skill pro ovládání domů iQtec.

5.1 Použité technologie

Pro Smart Home Skill jsem samozřejmě použil knihovnu Askhome popsanou v minulé kapitole. Díky tomu mohl být kód krátký a stačilo řešit pouze detaily zpracovávání definičního XML souboru a spojení se servery iQtec.

Celý kód jsem vyvíjel ve virtuálním prostředí vytvořeném pomocí virtualenv[16], ve kterém je izolovaná instalace Pythonu, čímž se zajistí stejná funkčnost na různých strojích a zjednoduší se tím nasazování.

Jako knihovnu pro posílání HTTP požadavků jsem zvolil Requests[17] od dříve zmíněného Kennetha Reitze, jednu z nejpoužívanějších knihoven pro Python vůbec [18]. Requests na své hlavní stránce vyzdvihuje jednoduchost použití, podporu SSL verifikace a transparentního HTTPS protokolu.

Pro práci s autorizací přes protokol OAuth jsem použil nadstavbu nad Requests, Requests-OAuthLib[18]. Průběh OAuth je sice jednoduchý a mohl bych ho naimplementovat sám pouze s knihovnou Requests, práce s Requests-OAuthLib je ale přímočará a přidání další závislosti v tomto případě ničemu nevadí.

[16] https://virtualenv.pypa.io
[17] http://python-requests.org
[18] https://github.com/requests/requests-oauthlib

5.1.1 DynamoDB

DynamoDB je databáze typu NoSQL založená na architektuře AWS. NoSQL je termín používaný k popisu nerelačních databází, které mají výhody snadného vývoje a škálovatelného výkonu. Tyto databáze využívají datových struktur jako dokumenty, grafy nebo mapy. Oproti tradičním relačním databázím většinou databáze NoSQL prioritizují horizontální škálování před robustností. [19]

Pro DynamoDB jsem se ale nerozhodl kvůli dobré škálovatelnosti, protože neočekávám velkou zátěž. Hlavním faktorem volby byla dobrá podpora DynamoDB uvnitř instancí AWS Lambda a pro vyvíjení aplikací Alexa je zdarma.

Pro práci s DynamoDB jsem využil knihovnu boto3[19], která slouží jako rozhraní se službami AWS. Knihovna je přímo podporována a doporučena Amazonem [20].

5.2 Implementace

Díky knihovně Askhome jsem mohl naprogramovat třídy přímo odpovídající komponentám Alexa z aplikace iQtec Architekt. Například ve zdrojovém kódu 5.1 je třída ztmavovatelného světla, která dědí od jednoduchého světla `OnOffLight` a přidává funkcionalitu nastavení jasu.

Všechny třídy mají společného předka `IQtecAppliance`, který definuje metody `send_value` a `read_value` pro posílání požadavků na PLC inteligentního domu. Tato třída také poskytuje atribut `properties`, který obsahuje všechny hodnoty příslušného tagu `appliance` z definičního XML souboru.

V příkladu 5.1 na řádce 6 zjistím identifikátor proměnné, jež ovládá jas a na kterou bude následně posílán požadavek. Na řádce 7 a 9 pracuji s maximální hodnotou jasu, podle které přeškáluji požadované procento. Potom už stačí poslat požadavek pomocí metody `send_value`, která automaticky zkontroluje, zda se hodnota správně nastavila a případně vyvolá výjimku `UnableToSetValueError`.

K posílání požadavků na PLC vytvářím instanci `OAuth2Session` z knihovny Requests-OAuthLib a inicializuji ji s přístupovým tokenem, který získám z požadavku od Amazon Echo. Podle tohoto přístupového tokenu autorizační server pozná, ke kterému uživateli požadavek patří. Instance `OAuth2Session` má po inicializaci stejné rozhraní, jako knihovna Requests a stejně jednoduše se s ní posílají HTTP požadavky.

[19] https://github.com/boto/boto3

```
class DimmableLight(OnOffLight):
    xml_type = 'alexa_light_dimmable'

    @Appliance.action
    def set_percentage(self, request):
        variable = self.properties['percentagevariable']
        max_val = self.properties['maxpercentagevalue']

        value = request.percentage / 100.0 * float(max_val)
        self.send_value(variable, value)
```

Zdrojový kód 5.1: Třída ztmavovatelného světla s metodou nastavení jasu

5.2.1 Využití DynamoDB

Protože instance AWS Lambda je fyzicky situována v Severní Americe, požadavky z ní na servery iQtec umístěné v České Republice mohou mít odezvu v řádech sekund. Bylo proto potřeba omezit veškerou komunikaci na minimum.

Jedním způsobem jak snížit odezvu bylo ukládání definičního XML. Bez toho by se tento soubor musel na začátku každého požadavku stáhnout, jinak by nebylo jak identifikovat ovládané zařízení. Řešením bylo tedy ukládání souboru do databáze DynamoDB. Protože jediným identifikátorem uživatele, který požadavek přes Amazon Echo vyvolal, je přístupový token, soubor je uložen s tímto tokenem jako klíč. Databáze DynamoDB v tomto smyslu tedy slouží jako cache definičních XML souborů.

5.3 Testování

Za testovací framework jsem zvolil stejně jako v knihovně Askhome pytest. Ve většině testů jsem použil knihovnu mock[20], která umožňuje simulovat chování nějaké funkce, třídy nebo celého modulu a nahradit jej logikou, kterou lze dobře testovat.

Při testování Smart Home Skill jsem využil mock pro nahrazení knihovny Requests tak, aby žádné skutečné HTTP požadavky nebyly vysílány do sítě. Místo toho jsem nasimuloval chování PLC inteligentního domu a kontroloval, zda komunikace probíhá bezchybně. Na závěr testu simulované funkce kontroluji počet volání a argumenty, se kterými byly zavolány.

[20] https://github.com/testing-cabal/mock

Pro otestování jednotlivých zařízení jsem v aplikaci iQtec Architekt vygeneroval definiční XML s každou Alexa komponentou. Tento soubor v testech načtu a nechám podle něj kód Smart Home Skill vytvořit instance jednotlivých zařízení. Potom zkouším každé zařízení zpracovat požadavek od Amazon Echo a kontroluji jeho JSON odpověď. Tímto se ujistím o správném propojení rozhraní mého kódu, aplikace iQtec Architekt a Smart Home Skill API.

5.4 Automatizace nasazování

Na AWS Lambda lze nasazovat pomocí webového formuláře, kam se nahraje ZIP archiv s kódem a všemi jeho závislostmi. Tento způsob je ale pomalý a pracný, a proto jsem chtěl nasazování zautomatizovat.

5.4.1 Možné nástroje

Prvním nástrojem, se kterým jsem experimentoval je Apex[21]. Apex je napsán v jazyce Go a jeho hlavní předností je schopnost emulovat jazyky, které AWS Lambda nepodporuje. Instance Lambda se definují pomocí JSON souborů, podle kterých je Apex vytvoří a nahraje na ně archiv s kódem. Možná nastavení jsou však omezená a například automatické propojení s eventy Smart Home API nadefinovat nelze.

Proto jsem přestoupil k nástroji Serverless[22]. Serverless funguje nad technologií Node.js a oproti nástroji Apex je mnohem populárnější a rozsáhlejší. Kromě AWS Lambda podporuje i další poskytovatele jako Google Cloud Functions nebo Azure Functions. Při nasazování používá Serverless službu AWS CloudFormation, pomocí které lze vytvářet a propojovat téměř všechny prostředky AWS. Tento způsob je sice mnohem silnější, ale zároveň výrazně pomalejší.

Nakonec jsem ale nepoužil ani Serverless, protože jsem narazil na problém balíčků pro Python, které jsou potřeba zkompilovat. Například knihovna lxml kvůli výkonu obsahuje část kódu v jazyce C a musí být během instalace na cílovém stroji zkompilována. Na AWS Lambda běží speciální operační systém Amazon Linux a předkompilovat balíčky pro něj by bylo náročné. Kvůli tomu jsem se nakonec rozhodl pro nástroj Zappa, který je tento problém schopný vyřešit.

[21] https://github.com/apex/apex
[22] https://serverless.com

5.4.2 Zappa

Zappa je nástroj v jazyce Python pro nasazování tzv. WSGI aplikací na AWS Lambda. WSGI je standardní rozhraní mezi jazykem Python a webovými servery, většina webových frameworků komunikuje právě pomocí tohoto rozhraní. Zappa dokáže transformovat HTTP požadavky z AWS Lambda na požadavky typu WSGI.

Vyvíjený Smart Home Skill sice WSGI nepoužívá, tato funkcionalita šla však pomocí konfiguračních souborů odebrat. Zappa se může konfigurovat soubory ve formátu JSON nebo YAML, použitá konfigurace ve formátu YAML je vidět na ukázce 5.2.

Velkou výhodou při nasazování nástrojem Zappa je integrace s virtuálním prostředím virtualenv. U ostatních nasazovacích nástrojů jsem musel buď pomocí skriptu nebo manuálně kopírovat závislosti do adresáře projektu aby se společně s kódem nahrály na AWS Lambda. Oproti tomu Zappa prohledá nainstalované balíčky virtuálního prostředí, kompilované balíčky vymění za své předkompilované pro OS Amazon Linux a všechno automaticky přidá do nahrávaného archivu.

```
1 dev:
2   s3_bucket: smart-home-skill-dev-deploy
3   lambda_handler: main.handler
4   aws_region: us-east-1
5   timeout_seconds: 20
6   memory_size: 128
7   keep_warm: false
8   touch: false
```

Zdrojový kód 5.2: Konfigurace nástroje Zappa pro nasazování Smart Home Skill

Kapitola 6

Custom Skill

V této kapitole popíšu realizaci dovednosti Custom Skill, která doplňuje Smart Home Skill.

6.1 Účel Custom Skill

Protože některé dotazy jako zjištění vlhkosti nebo spotřeby elektřiny nešly pomocí Smart Home Skill obsloužit, Custom Skill měl tyto speciální požadavky doplnit. Zároveň bylo potřeba podporovat veškerou funkcionalitu Smart Home Skill, aby si uživatel nemusel pamatovat příslušnost příkazu k dovednosti. Tímto způsobem si stačí vědět, že jednoduché příkazy lze rovnou sdělit Smart Home Skill a složitější až po vstupu do Custom Skill.

Kvůli nutnosti manuální tvorby interakčního modelu je ale výsledný Custom Skill méně robustní a uživatelsky přívětivý než Smart Home Skill s předpřipraveným modelem od Amazonu. Zároveň většinu funkcí inteligentních domů iQtec ve Smart Home Skill podporovat jde, a proto je význam Custom Skill v tomto projektu spíše experimentální. Podle budoucí zpětné vazby se bude moci buď rozšířit, nebo úplně zrušit.

6.2 Custom Skill API

Na rozdíl od Smart Home Skill API, pro používání Custom Skill API je potřeba vytvořit interakční model, který definuje aktivační věty pro každý příkaz, tzv. intent. Každý intent reprezentuje jednu akci uživatelské interakce a může obsahovat tzv. sloty, které slouží jako parametry. Pak už stačí v kódu každý intent se sloty zpracovat, podobně jako volání funkce s argumenty.

Například příkaz „Alexa, ask iQtec what is the humidity in my bedroom?“ se podle mého interakčního modelu interpretuje jako GetHumidityIntent s hodnotou bedroom ve slotu appliance. Jak již bylo řečeno, pro použití Custom Skill musí uživatel nejprve do dovednosti vstoupit, nebo jako v tomto příkladu zmínit její jméno. [21]

6.3 Interakční model

Interakční model pro Custom Skill se definuje pomocí souboru ve formátu JSON, Amazon ale poskytuje webovou aplikaci Skill Builder pro interaktivní vytváření modelu.

Pro vytvoření intentu se musí zadat několik příkladných aktivačních vět, pomocí kterých se služba naučí rozpoznat žádanou akci. V aktivačních větách se mohou specifikovat sloty. Každý slot je nějakého typu, pro který je potřeba také zadat několik příkladů, aby služba věděla, jaká slova může na pozicích slotů očekávat.

Pro každý slot označený jako povinný se také musí vytvořit výzva, kterou se Echo zeptá na doplnění, pokud uživatel slot původně nespecifikoval.

V mém případě jsem měl jeden typ slotu Appliance, který zachycoval zařízení k ovládání. Každý intent, až na obecné dotazy na celý dům, odpovídal akci jednoho zařízení, podobně jako ve Smart Home Skill. Tímto způsobem jsem mohl vytvořit intenty pro již naimplementovanou funkcionalitu ze Smart Home Skill s podobnými aktivačními větami a doplnit ještě nepodporované akce ve stejném formátu.

6.4 Použité technologie

Při tvorbě Custom Skill jsem vycházel ze zdrojového kódu Smart Home Skill a použil stejné knihovny a nástroje včetně Askhome, Requests-OAuthLib a Zappa.

6.4.1 Flask-Ask

Propojení s Custom Skill API mi zjednodušila knihovna Flask-Ask[23], rozšíření pro populární webový framework Flask[24]. Flask-Ask dokáže pomocí dekorátorů navázat funkce a jejich argumenty na odpovídající intenty a sloty, podobně jako v knihovně Askhome.

[23] https://github.com/johnwheeler/flask-ask

[24] http://flask.pocoo.org

6.4.2 Zappa

Ve Smart Home Skill jsem nástroj Zappa používal pouze k automatizaci nasazování. Protože ale Custom Skill lze nastavit na komunikaci pomocí protokolu HTTP, Zappa zde tím pádem plnil svoji hlavní funkci transformace HTTP požadavků na rozhraní WSGI, se kterým pracuje framework Flask.

6.5 Implementace

Protože Custom Skill je v podstatě rozšíření Smart Home Skill, obsahuje velkou část stejné logiky. Kvůli tomu jsem mezi oběma dovednostmi sdílel definici zařízení pomocí potomků třídy `Appliance` z knihovny Askhome. Takto je ovládací logika na jednom místě a kód dovedností pouze spojuje jednotlivá rozhraní.

Pro označení podporovaných metod zařízení jsem vytvořil dekorátor `@custom_skill_action`, fungující podobně jako `@Appliance.action` z knihovny Askhome. Pokud přijde požadavek na ovládání zařízení, které nemá odpovídající metodu označenou pomocí tohoto dekorátoru, Custom Skill odpoví chybovou hláškou nepodporované akce.

Na příkladu zdrojového kódu 6.1 je vidět zpracování požadavku nastavení procenta pomocí knihovny Flask-Ask. Na první řádce se nastaví, na který intent bude funkce odpovídat a v argumentu `convert` se specifikují typy jednotlivých slotů. Na další řádce dekorátor `@handle_exceptions` zajišťuje uživatelsky přívětivé odpovědi při vyvolání výjimky. Na řádce 4 se volá funkce `handle_action`, která zinicializuje zařízení podle jména

```
1  @ask.intent('SetPercentageIntent', convert={'percentage':int})
2  @handle_exceptions
3  def set_percentage(appliance_name, percentage):
4      appliance, request = handle_action(appliance_name,
5                                         'set_percentage')
6      request.percentage = percentage
7      appliance.set_percentage(request)
8
9      return statement('OK, I set %s to %s percent' %
10                      (appliance_name, percentage))
```

Zdrojový kód 6.1: Zpracování požadavku nastavení procenta pomocí knihovny Flask-Ask

a zkontroluje, zda podporuje žádanou akci. Dále se nastavuje objekt typu `CustomSkillRequest`, který kopíruje rozhraní třídy `Request` z knihovny Askhome, a proto se může použít jako argument pro metodu `set_percentage`. Nakonec se už jen odpoví uživateli pomocí funkce `statement` z knihovny Flask-Ask

Tímto způsobem se obstará většina požadavků týkajících se jednotlivých zařízení. Obecné dotazy na celý dům, jako aktuální spotřeba nebo otevřená okna, řeším pomocí speciálního typu zařízení `alexa_status`. Ve výčtu všech zařízení najdu první s tímto typem, a pak požadavek zpracuji stejně jako ostatní.

Kapitola 7

Autorizační server

Tato kapitola se týká serveru zajišťujícím autorizaci a autentizaci uživatelů iQtec pomocí protokolu OAuth 2.0.

7.1 Protokol OAuth

Protokol OAuth vznikl v roce 2007 jako standard autorizace cizích aplikací pro přístup k soukromým zdrojům uživatelů bez zadávání hesla do cizí aplikace. [22]

Jako příklad oficiální stránka OAuth uvádí speciální klíče pro luxusní auta. Tyto klíče jsou určeny pro hotelovou obsluhu a oproti normálním klíčům neumožňují jet s autem více než pár kilometrů. S některými klíči nelze otevírat kufr, zatímco jiné blokují adresář v palubním počítači. Každý z těchto klíčů slouží k předání jistých práv třetí straně stejně jako u protokolu OAuth. [22]

7.1.1 OAuth 2.0

Druhá verze protokolu OAuth vyšla v roce 2012 jako kompletní přepracování prvního vydání [23]. OAuth 2.0 už neobstarává bezpečnost přenosu, a díky tomu je značně jednodušší. Rovněž obsahuje zjednodušený průběh autorizace a změněnou terminologii. Protože je nyní světovým standardem, dále se budu zabývat pouze verzí OAuth 2.0.

7.1.2 Role v OAuth

V protokolu OAuth vystupují čtyři různé role [24][25]:

Uživatel Majitel zdroje, který poskytuje oprávnění ke svému účtu.

Klient Aplikace třetí strany, která žádá o autorizaci.

API Server používaný pro přístup k informacím uživatele nebo chráněnému zdroji.

Autorizační server Server zajišťující rozhraní pro přihlášení uživatele a přijmutí nebo odmítnutí přístupových požadavků. Autorizační server může být v menších implementacích stejný jako API server.

7.1.3 Průběh autorizace

OAuth podporuje několik různých typů průběhu, já se dále budu zabývat pouze nejpoužívanějším způsobem Authorization Code Grant pro webové aplikace.

Na obrázku 7.1 je vidět typický průběh autorizace v protokolu OAuth [24]:

0. Nejdříve musí být klientská aplikace zaregistrována na autorizačním serveru, odkud dostane svůj soukromý a veřejný identifikační kód.

1. Klient přesměruje prohlížeč na přihlašovací stránku autorizačního serveru s parametrem svého veřejného identifikačního kódu.

2. Uživatel se přihlásí a potvrdí autorizační požadavek.

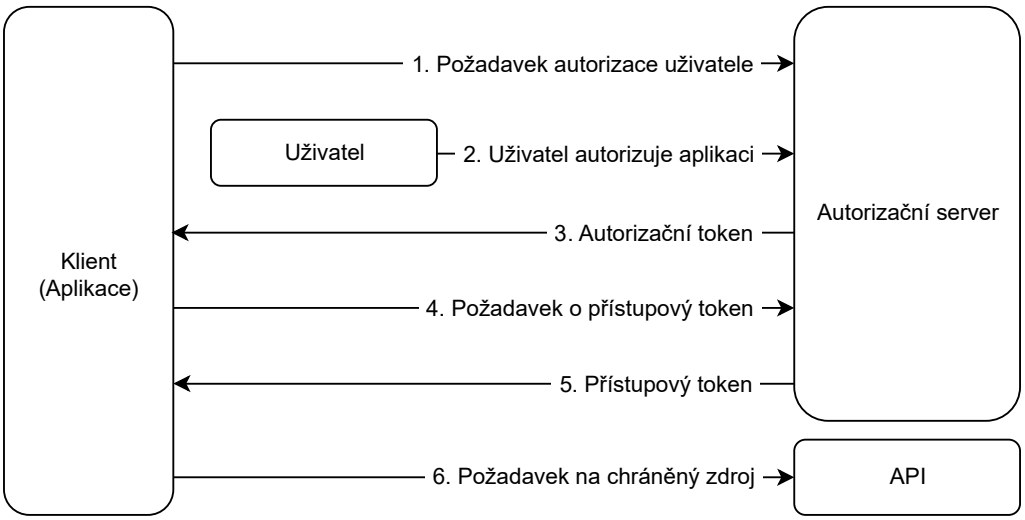
3. Autorizační server přesměruje prohlížeč zpět na stránku klienta s autorizačním tokenem.

4. Klientský server zašle požadavek pro výměnu autorizačního tokenu za přístupový se svým soukromým identifikačním kódem.

5. Autorizační server zašle zpět přístupový a obnovovací token.

6. Klient nyní může využívat API pomocí svého přístupového tokenu.

7. Po vypršení platnosti přístupového tokenu může klient zažádat o nový se svým obnovovacím tokenem.



Obrázek 7.1: Průběh autorizace v protokolu OAuth

7.2 Možné řešení autorizace

Služby Alexa pro autorizaci a identifikaci uživatelů používají protokol OAuth, při vytváření Smart Home Skill dokonce Amazon autorizační server vyžaduje [26]. Amazon ale nabízí pro aplikace, které vlastní autorizaci nepotřebují, svého poskytovatele OAuth, který se jednoduše nastaví přes webové rozhraní AWS.

V mém případě jsem ale autorizaci uživatelů iQtec potřeboval a poskytovatel OAuth od Amazonu nestačil. Šlo by sice manuálně propojovat účty uživatelů iQtec s jejich účty na Amazonu, to by ale znamenalo nahrávání konfiguračního XML při každé změně. Proto jsem se rozhodl vytvořit vlastního poskytovatele OAuth napojeného na ostatní systémy iQtec. Tento poskytovatel je zatím používán pouze pro autorizaci služeb Alexa, do budoucna ale není problém řešení rozšířit pro další aplikace, případně na protokol OAuth převést ostatní způsoby autentizace uvnitř systémů iQtec.

7.3 Použité technologie

Pro autorizační server iQtec jsem vybral webový framework Django[25] pro jazyk Python, protože několik dalších služeb iQtec už tento framework používají a mám s ním zkušenosti.

[25]https://www.djangoproject.com

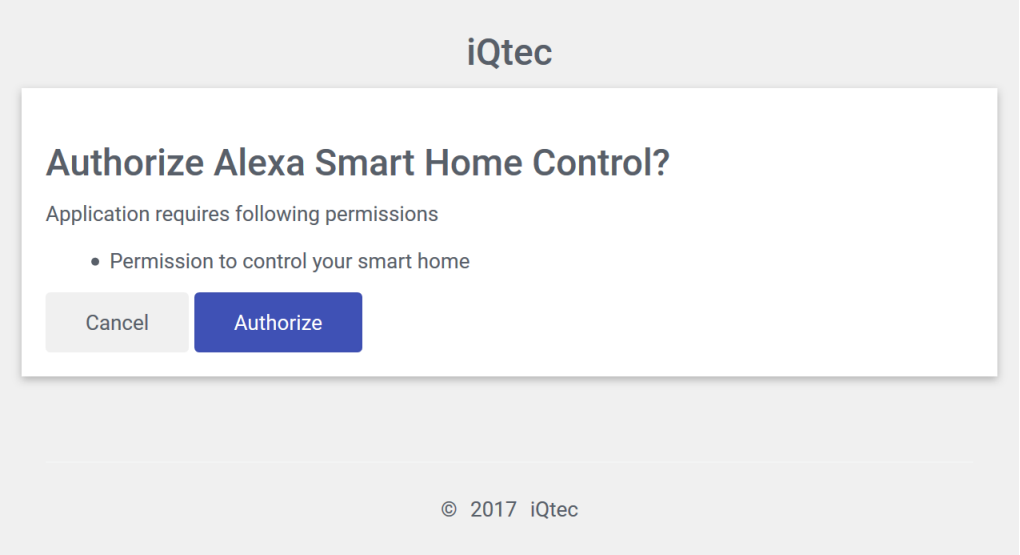
Pro Python existuje několik knihoven implementujících protokol OAuth, já jsem zvolil populární knihovnu OAuthLib[26]. OAuthLib a framework Django jsou propojeny knihovnou django-oauth-toolkit[27], která je přímo doporučena v dokumentaci OAuthLib.

Jako webový server a databázi jsem použil již nainstalované technologie Apache[28] a PostgreSQL[29].

7.4 Implementace

S knihovnou django-oauth-toolkit pro vytvoření autorizačního serveru stačilo správně nakonfigurovat nastavení OAuth jako délku platnosti přístupových tokenů a nasměrovat adresy URL na předem připravené pohledy autorizačního rozhraní.

Pro sladění uživatelského rozhraní se zbytkem webových portálů iQtec jsem přepsal HTML šablony tak, aby používaly styly CSS, které jsem převzal z jiného projektu. Výsledný vzhled je vidět na obrázku 7.2.



Obrázek 7.2: Webové rozhraní autorizačního serveru iQtec

[26] https://github.com/idan/oauthlib
[27] https://github.com/evonove/django-oauth-toolkit
[28] http://httpd.apache.org
[29] https://www.postgresql.org

Klientské aplikace lze registrovat přes administrační nástroj Django Admin, který je zahrnut ve frameworku Django a slouží k jednoduché správě definovaných modelů. Pro propojení se službou Alexa stačilo v mém administračním rozhraní vytvořit novou aplikaci a vygenerovaný veřejný a soukromý kód zkopírovat na stránku správy služby Alexa na AWS.

7.5 Nasazení

Při nasazování autorizačního serveru jsem čelil problému s protokolem HTTPS, a to z důvodu, že služba Alexa povoluje autorizaci OAuth jen přes šifrované spojení s certifikátem podepsaným od důvěryhodné certifikační autority [26]. Musel jsem proto vyřešit podporu tohoto protokolu.

7.5.1 SSL certifikáty

Protokol HTTPS funguje pomocí bezpečnostní vrstvy SSL, kde každé spojení je zašifrováno veřejným klíčem SSL certifikátu [27]. Každý certifikát musí být podepsán, a to buď sám sebou (self-signed) nebo certifikační autoritou. Pokud podpis není od důvěryhodné certifikační autority, většina prohlížečů ukáže upozornění, a proto je žádoucí mít podpis od takové autority.

Protože iQtec zatím na žádné ze svých stránek HTTPS nepodporoval, neexistoval žádný SSL certifikát a musel jsem ho zajistit.

7.5.2 Let's Encrypt

Jako certifikační autoritu jsem zvolil Let's Encrypt[30], která jako jedna z mála poskytuje podepsané certifikáty automatizovaně a zdarma [28]. Většina ostatních autorit požaduje za podpis finanční částku a ověření identity skrz byrokratický proces. Let's Encrypt nabízí nástroj CertBot[31], který automatizuje udělování certifikátu po instalaci na cílový server. Díky tomu bylo získání podepsaného certifikátu rychlé a bezproblémové.

7.5.3 Doména autorizačního serveru

Kvůli tomu, že autorita Let's Encrypt uděluje certifikáty pouze na základě doménových adres, musel jsem zaregistrovat pro autorizační server jméno.

[30] https://letsencrypt.org
[31] https://certbot.eff.org

Využil jsem již existující adresy iqtec.cz a přidal jsem DNS záznam pro subdoménu auth.iqtec.cz.

7.5.4 Konfigurace Apache

Nástroj CertBot umí do jisté míry automaticky konfigurovat servery Apache pro používání HTTPS, v mém případě ale běželo na stroji několik instancí Apache a CertBot neupravil konfigurační soubory správně. Doposud se rozlišovalo mezi instancemi pomocí čísel portů, s novou doménou jsem ale chtěl používat pro autorizační servery porty 80 a 443, standardní pro tomu odpovídající protokoly HTTP a HTTPS. Vytvořil jsem proto další instanci Apache, která poslouchala na portech 80 a 443 a rozdělovala pomocí modulu mod_proxy požadavky na příslušné porty podle příchozích domén. Tento proxy server se zároveň postaral o šifrování SSL a dále posílal pouze rozšifrované požadavky HTTP, takže vlastní webové servery už HTTPS nemusely řešit.

Kapitola 8

Uživatelské testování

Součástí mé práce bylo testovaní reálné použitelnosti hlasového ovládání, a to dvěma zaměstnanci firmy iQtec a třemi členy rodiny.

8.1 Testovací instalace

Testování probíhalo na fyzické instalaci, která obsahovala PLC iQtec připojené k žárovce, termostatu a relé ovládající žaluzie. Další funkcionalita byla virtuálně simulována. Pro hlasové ovládání byla použita jednotka Amazon Echo Dot druhé generace, která oproti standardnímu zařízení Amazon Echo nemá tak výkonný reproduktor, mikrofon však mají obě jednotky totožný.

8.2 Průběh testování

Při testování jsem každého účastníka požádal o následující úkoly:

1. Rozsvítit a zhasnout světlo.
2. Zhasnout všechna světla.
3. Nastavit jas světla na 50%.
4. Nastavit scénář osvětlení.
5. Zjistit teplotu a vlhkost místnosti.
6. Nastavit teplotu termostatu na 21° C.
7. Zvýšit teplotu termostatu o 1° C.

8. Stáhnout a vytáhnout žaluzie.

9. Nastavit úhel žaluzie směrem na slunce.

Před každým testováním jsem účastníkovi vysvětlil jak celý systém funguje a jak hlasové rozhraní používat, ale formulace jednotlivých úkolů jsem explicitně nezmiňoval a nechal ji na účastnících.

Každý úkol měli účastnící provést nejdříve pomocí Smart Home Skill pokud jej podporoval, a poté i přes Custom Skill.

8.3 Výsledky a komentáře

Odezva účastníků byla kladná, hlasové ovládání by si každý z nich dokázal představit jako jejich budoucí doplněk do domácnosti. S většinou úkolů neměli výrazný problém a funkcionalita systému se jim líbila.

Někteří měli potíže s hlasovým rozpoznáváním kvůli jejich anglické výslovnosti. Největším problémem bylo ovládání žaluzií, kdy Amazon Echo interpretovalo slovo „blinds“ (žaluzie) jako „lights“ (světla). Protože se ale hlasové rozpoznávání služby Alexa používáním postupně učí rozumět výslovnosti jednotlivých uživatelů, očekával bych, že tento problém po čase zmizí. Například mým příkazům Amazon Echo rozumí již velmi dobře.

Při používání Custom Skill také často docházelo k chybnému rozpoznání jména zařízení, které se lišilo jen minimálně od vysloveného, což vedlo k hlášce neexistujícího zařízení. Je to způsobeno tím, že ve slotech interakčního modelu nelze omezit množinu možných slov. Služba Alexa tak může chybně rozpoznat do slotu jména zařízení jakékoliv jiné slovo. Oproti tomu ve Smart Home Skill jsou všechna jména zařízení definována a hlasové rozpoznávání tedy funguje lépe. Toto chování zároveň s nutností vstoupení do Custom Skill odradilo některé účastníky, přáli si mít veškerou funkcionalitu ve formátu Smart Home Skill.

Jeden účastník zmínil, že by systém mohl být užitečný pro hendikepované uživatele, kteří mají problémy interagovat s fyzickými ovladači.

Další účastník celý systém přirovnal k elektrickému ovládání okének auta. Podle něj si na pohodlí člověk zvykne rychle, a pak už se mu nechce vracet zpět. Stejně tak by si dokázal představit pohodlí hlasového ovládání.

Závěr

Cílem této bakalářské práce bylo navrhnout a implementovat systém hlasového rozhraní pro inteligentní domy iQtec, který je dobře použitelný a do budoucna rozšiřitelný. Věřím, že tyto cíle byly úspěšně splněny.

Díky kvalitní technologii hlasového rozpoznávání Alexa je výsledná použitelnost dobrá, i když ji jazyková bariéra do jisté míry omezuje. Integrace do aplikace iQtec Architekt umožňuje rychle vytvořit hlasové rozhraní nových nebo existujících inteligentních domů. Přidání podpory pro další zařízení iQtec je díky návrhu systému jednoduché a bezproblémové.

Části systému také mohou pomoci ostatním budoucím projektům. Vytvořený autorizační server může sloužit jako centrální autentizační služba dalších produktů iQtec. Otevřená knihovna Askhome výrazně zjednodušuje práci se Smart Home Skill API, což bylo v této práci potvrzeno, a díky kvalitní dokumentaci může ušetřit čas ostatním vývojářům pro Amazon Echo.

Systém by do budoucna mohl být rozšířen o další zařízení a komplexnější interakce s nimi. Možným vylepšením by také mohl být nástroj pro uživatele umožňující upravovat hlasové rozhraní pro své potřeby. Jestliže služba Alexa začne podporovat hlasové rozpoznávání v českém jazyce, bude třeba části systému přeložit. Pokud by se v budoucnu ukázala jiná hlasová technologie jako lepší, mohl by na ni být celý systém přemigrován.

Celkový průběh realizace systému byl velmi zajímavý a rád bych proto pokračoval v jeho vývoji, a to jak z hlediska rozvoje komerčního produktu, tak i knihovny Askhome, kterou bych chtěl dále udržovat a postupně vylepšovat.

Literatura

[1] Soper, T.: Amazon Echo sales reach 5M in two years [online]. 2016, [cit. 2017-02-07]. Dostupné z: http://www.geekwire.com/2016/amazon-echo-sales-reach-5m-two-years-research-firm-says-google-competitor-enters-market/

[2] Moynihan, T.: OK, Alexa: A Google Home Versus Amazon Echo IQ Test [online]. 2016, [cit. 2017-02-07]. Dostupné z: https://www.wired.com/2016/11/ok-alexa-google-home-versus-amazon-echo-iq-test/

[3] Echo & Alexa Devices [online]. [cit. 2017-02-07]. Dostupné z: https://www.amazon.com/Amazon-Echo-And-Alexa-Devices/b?ie=UTF8&node=9818047011

[4] Amazon.com, Inc.: *Hosting a Custom Skill as a Web Service* [online]. [cit. 2017-02-15]. Dostupné z: https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/docs/developing-an-alexa-skill-as-a-web-service.

[5] Weins, K.: Cloud Computing Trends: 2017 State of the Cloud Survey [online]. 2017, [cit. 2017-03-11]. Dostupné z: http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2017-state-cloud-survey

[6] Cloud Computing with Amazon Web Services [online]. [cit. 2017-03-11]. Dostupné z: https://aws.amazon.com/what-is-aws/

[7] Amazon.com, Inc.: *What Is AWS Lambda?* [online]. [cit. 2017-03-11]. Dostupné z: http://docs.aws.amazon.com/lambda/latest/dg/welcome.html.

[8] Introduction to LINQ [online]. [cit. 2017-04-24]. Dostupné z: https://msdn.microsoft.com/en-us/library/bb397897(v=vs.110).aspx

[9] Effective Xml Part 1: Choose the right API [online]. 2011, [cit. 2017-04-24]. Dostupné z: https://blogs.msdn.microsoft.com/xmlteam/2011/09/14/effective-xml-part-1-choose-the-right-api/

[10] Amazon.com, Inc.: *Steps to Create a Smart Home Skill* [online]. 2017, [cit. 2017-02-15]. Dostupné z: https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/docs/steps-to-create-a-smart-home-skill.

[11] van Rossum, G.; Warsaw, B.; Coghlan, N.: *PEP 8 – Style Guide for Python Code* [online]. 2001, [cit. 2017-04-02]. Dostupné z: https://www.python.org/dev/peps/pep-0008/.

[12] Patel, A.; Picard, A.; Jhong, E.: *Google Python Style Guide* [online]. [cit. 2017-04-02]. Dostupné z: https://google.github.io/styleguide/pyguide.html.

[13] Reitz, K.; Schlusser, T.: *The Hitchhiker's Guide to Python: Best Practices for Development*. O'Reilly Media, první vydání, ISBN 978-1491933176.

[14] Amazon.com, Inc.: *Smart Home Skill API Reference* [online]. 2017, [cit. 2017-02-15]. Dostupné z: https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/docs/smart-home-skill-api-reference.

[15] Why Shooting for 100% Test Coverage is Important [online]. 2011, [cit. 2017-03-15]. Dostupné z: https://lionfacelemonface.wordpress.com/2011/09/24/why-shooting-for-100-test-coverage-is-important/

[16] Fowler, M.: Continuous Integration [online]. 2006, [cit. 2017-03-07]. Dostupné z: https://www.martinfowler.com/articles/continuousIntegration.html

[17] *Example Google Style Python Docstrings* [online]. [cit. 2017-04-02]. Dostupné z: http://sphinxcontrib-napoleon.readthedocs.io/en/latest/example_google.html.

[18] PyPI Ranking [online]. [cit. 2017-04-20]. Dostupné z: http://pypi-ranking.info/alltime

[19] What is NoSQL? [online]. [cit. 2017-04-20]. Dostupné z: https://aws.amazon.com/nosql/

[20] Getting Started with AWS and Python [online]. 2010, [cit. 2017-04-20]. Dostupné z: https://aws.amazon.com/articles/3998

[21] Amazon.com, Inc.: *Understanding Custom Skills* [online]. [cit. 2017-03-11]. Dostupné z: https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/overviews/understanding-custom-skills.

[22] Hammer-Lahav, E.: OAuth Introduction [online]. 2007, [cit. 2017-04-24]. Dostupné z: https://oauth.net/about/introduction/

[23] Differences Between OAuth 1 and 2 [online]. [cit. 2017-04-24]. Dostupné z: https://www.oauth.com/oauth2-servers/differences-between-oauth-1-2/

[24] Microsoft Corporation: *The OAuth 2.0 Authorization Framework* [online]. 2012, [cit. 2017-04-24]. Dostupné z: https://tools.ietf.org/html/rfc6749.

[25] Parecki, A.: OAuth 2 Simplified [online]. [cit. 2017-04-24]. Dostupné z: https://aaronparecki.com/oauth-2-simplified/

[26] Amazon.com, Inc.: *Linking an Alexa User with a User in Your System* [online]. 2017, [cit. 2017-04-02]. Dostupné z: https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/docs/linking-an-alexa-user-with-a-user-in-your-system.

[27] RTFM, Inc.: *HTTP Over TLS* [online]. 2000, [cit. 2017-04-02]. Dostupné z: https://tools.ietf.org/html/rfc2818.

[28] Hartley, M.: Let's Encrypt: The Good and the Bad [online]. 2016, [cit. 2017-04-24]. Dostupné z: http://www.datamation.com/security/lets-encrypt-the-good-and-the-bad.html

Příloha A

Seznam použitých zkratek

API Application Programming Interface

AWS Amazon Web Services

CRC Cyclic redundancy check

CSS Cascading Style Sheets

GSM Global System for Mobile Communications

HTTP Hypertext Transfer Protocol

HTTPS HTTP Secure

HTML Hypertext Markup Language

JSON JavaScript Object Notation

LINQ Language Integrated Query

OS Operating System

PDF Printable Document Format

PLC Programmable Logic Controller

PyPI Python Package Index

RFC Request for Comments

SSL Secure Sockets Layer

UDP User Datagram Protocol

SDK Software Development Kit

WSGI Web Server Gateway Interface

XML Extensible Markup Language

YAML YAML Ain't Markup Language

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `askhome` ..... adresář balíčku Askhome
    - `askhome` ..... zdrojové kódy knihovny Askhome
    - `docs` ..... zdrojové kódy dokumentace knihovny Askhome
    - `test` ..... zdrojové kódy testů knihovny Askhome
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `BP_Hlaváček_Matěj_2017.pdf` ..... text práce ve formátu PDF