VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

CHATBOT V PODNIKOVÉM INFORMAČNÍM SYSTÉMU

CHATBOT IN AN ENTERPRISE INFORMATION SYSTEM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. MIROSLAV NOVÁK**
AUTHOR

VEDOUCÍ PRÁCE **doc. RNDr. JITKA KRESLÍKOVÁ, CSc.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

Zadání diplomové práce

Student: **Novák Miroslav, Bc.**
Program: Informační technologie Obor: Počítačová grafika a multimédia
Název: **Chatbot v podnikovém informačním systému**
Chatbot in an Enterprise Information System
Kategorie: Zpracování řeči a přirozeného jazyka

Zadání:

1. Seznamte se s problematikou chatovacích robotů, rozpoznávání textu a hlasu.
2. Analyzujte možnosti využití již existujících aplikačních rámců a dostupných služeb (například od společnosti Microsoft).
3. Seznamte se se systémem Product Information Management společnosti Allium, s.r.o. a navrhněte napojení chatovacího robota na tento systém.
4. Po dohodě s konzultantem společnosti Allium, s.r.o. zvolte vhodné vývojové prostředí a chatovacího robota implementujte.
5. Otestujte funkčnost robota na vhodném vzorku dat vybraném po dohodě s vedoucí.
6. Zhodnoťte dosažené výsledky a diskutujte další možnosti rozšíření vytvořené aplikace.

Literatura:

- WILKS, Yorick. *Machine Conversations*. London: Springer, 1999. ISBN 978-0792385448.
- SHEVAT, Amir. *Designing Bots: Creating Conversational Experiences.* O'Reilly Media, 2017. ISBN 978-1491974827.
- THEOBALD, Oliver. *Machine Learning for Absolute Beginners: A Plain English Introduction*. Independently published, 2017. ISBN 978-1520951409.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů zadání 1 - 3.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Kreslíková Jitka, doc. RNDr., CSc.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 22. května 2019
Datum schválení: 23. října 2018

Abstrakt

Tato diplomová práce řeší problematiku tvorby chatovacích robotů. Teoretická část práce představuje obecně koncept konverzačního rozhraní a analyzuje dostupné technologie k jeho tvorbě. Praktická část práce řeší návrh a implementaci konkrétního chatovacího robota, jehož cílem je být virtuálním asistentem v procesu výběru a nákupu zboží. Toho je dosaženo pomocí propojení chatovacího robota se systémem správy podnikových informací pomocí OData webových služeb. Jedním z největších problému bylo určení pořadí dotazovaných otázek týkající se vlastností produktů, přičemž pro implementaci byla využita teorie tvorby rozhodovacích stromů.

Abstract

This diploma thesis deals with problems of development of chatbots. The theoretical part of the thesis introduces the concept of the conversational interface in general and analyzes available technologies for its development. The practical part deals with the design and implementation of a particular chatbot, whose goal is to be a virtual assistant in the process of selecting and purchasing goods. This is accomplished by connecting the chatbot to the product information management system using OData web services. One of the biggest problems was to determine the order of questions asked about product properties. For the implementation was used decision tree theory.

Klíčová slova

chatovací robot, zpracování přirozeného jazyka, umělá inteligence, rozpoznávání textu a hlasu, správa podnikových informací, .NET

Keywords

chatbot, natural language processing, artificial intelligence, text and voice recognition, product information management, .NET

Citace

NOVÁK, Miroslav. *Chatbot v podnikovém informačním systému.* Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. RNDr. Jitka Kreslíková, CSc.

Chatbot v podnikovém informačním systému

Prohlášení

Čestně prohlašuji, že jsem tuhle diplomovou práci vypracoval samostatně pod vedením paní doc. RNDr. Jitky Kreslíkové, CSc. Další informace mi poskytl pan Ing. Martin Opršal. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Miroslav Novák
21. května 2019

Poděkování

Děkuji své vedoucí diplomové práce paní doc. RNDr. Jitce Kreslíkové, CSc. za odborné vedení této práce. Dále děkuji panu Ing. Martinu Opršalovi ze společnosti Allium s.r.o. za konzultace a odbornou pomoc při řešení práce. Poděkování patří také mé rodině, která mě podporovala po celou dobu studia a vytvořila mi podmínky, které mi umožnily úspěšně studovat. Speciální poděkování poté náleží mé přítelkyni, která mi poskytla laskavou podporou v těžkých dobách. Zapomenout nesmím ani na své přátele. Díky vzájemné podpoře a nezištné pomoci bylo možné překonat všechny nástrahy studia.

Obsah

Kapitola 1

Úvod

Chatovací roboti[1] mají potenciál změnit způsob, jakým my lidé komunikujeme s počítačem, tak jako se to dříve povedlo například webovým stránkám nebo mobilním zařízením. Hlavním komunikačním prostředkem mezi uživatelem a chatbotem je právě konverzace, která byla jedním z prvních způsobů dorozumívání mezi lidmi, proto je nám tolik přirozená. Jedním z důvodů zvýšeného zájmu o konverzační rozhraní je v současnosti velmi oblíbená oblast – internet věcí a s ním spojená chytrá zařízení, která se objevují všude kolem nás. Konverzační rozhraní se například skvěle doplňuje s chytrými domácnostmi a chytrými automobily.

Chatboty lze vnímat více způsoby. Například jako **vědní obor** snažící se vytvořit umělou inteligenci, která bude s uživatelem schopná vést uspokojivou konverzaci. Tato oblast výzkumu je v současnosti velmi aktuální a obzvlášť klíčoví hráči IT průmyslu se o ni velmi zajímají. Vzpomenout lze například každoroční soutěž Amazon Alexa Price, kde se vědní týmy z předních světových univerzit snaží vytvořit chatbota, který bude co nejlépe vést obecnou konverzaci na jakékoliv téma. První cenou v soutěži je odměna až 500 tisíc dolarů. Zajímavostí je, že v roce 2017 a 2018 se na druhém místě umístil tým Alquist z univerzity ČVUT. Jiný způsob, který se pro komerční využití hodí více, je vnímat chatboty jako nové a revoluční **uživatelské rozhraní**, pomocí kterého můžou uživatelé komunikovat se svými oblíbenými službami v prostředí chatovacích aplikací [27]. Tito chatboti standardně žijí v aplikacích jako Facebook Messanger, Slack nebo v prostředí webové aplikace, kde pomáhají uživatelovi řešit problémy nejrůznějšího typu.

V rámci diplomové práce je řešen návrh a implementace konkrétního chatbota, jehož cílem je usnadnit zákazníkům výběr a nákup zboží. Specifikace a řešení problému bylo konzultováno ve společnosti Allium s.r.o., která se specializuje na implementaci a instalaci podnikových informačních systémů. Navrhovaný chatbot bude integrován do systému správy podnikových informací, díky čemuž bude mít přístup k produktovým informacím dané firmy.

Struktura dokumentu

V kapitole 2 se nachází stručný úvod do problematiky tvorby chatbotů a je zde představena nezbytná teorie a terminologie, která se bude používat v následujících částech. Kapitola 3 se zabývá analýzou jednotlivých přístupů a technologií, které lze využít na tvorbu chatbotů. Kapitola 4 představuje systém správy podnikových informací, do něhož bude chatbot

[1] Pro chatovacího robota bude nadále používán pojem chatbot. V odborné literatuře se lze taktéž setkat s pojmy jako bot, smartbot, chatterbot, interaktivní agent nebo třeba konverzační rozhraní.

integrován a zároveň představuje jednotlivé funkční požadavky na chatbota. V kapitole 5 se nachází detailní návrh chatbota. Kapitola 6 se zabývá implementací a seznamuje s některými zajímavými problémy, které byly v rámci práce řešeny. Kapitola 7 představuje, jak byla ověřena funkcionalita chatbota. V kapitole 8 se poté nachází celkový souhrn diplomové práce a její závěr.

Kapitola 2

Stručný úvod do problematiky tvorby chatbotů

> A computer would deserve to be called intelligent if it could deceive a human into believing that it was human.
>
> Alan Turing

Cílem této kapitoly je uvést čtenáře do problematiky tvorby chatbotů a představit základní teoretická východiska pro jejich tvorbu. První část (2.1) seznamuje čtenáře se základy zpracování přirozeného jazyka. Další část (2.2) stručně popisuje historii chatbotů od prvního chatbota až po současnost. Třetí část (2.3) poté představuje typické případy užití chatbotů v současném světě, zatímco čtvrtá část (2.4) klasifikuje chatboty do několika kategorií. Poslední část (2.5) uvádí hlavní platformy, na kterých může chatbot s uživatelem komunikovat.

2.1 Zpracování přirozeného jazyka

Zpracování přirozeného jazyka[1] je věda, která je na pomezí věd zabývající se informatikou, umělou inteligencí a lingvistikou. Jejím cílem je analýza, generování a porozumění textů v přirozeném jazyce. Definovat tuto vědu lze i velmi jednoduše – jedná se jakoukoliv manipulaci s přirozeným jazykem pomocí počítače, přičemž přirozený jazyk je každý jazyk, který je používán pro každodenní lidskou interakci. Takovým jazykem je například angličtina nebo čeština. Opakem přirozeného jazyka je jazyk umělý. Příkladem můžou být programovací jazyky, které mají, oproti přirozeným, formálně definovanou syntaxi a sémantiku [19].

2.1.1 Problémy, které zpracování přirozeného jazyka řeší

Hlavní problémy, které zpracování přirozeného jazyka řeší, lze rozdělit do několika kategorií [11]:

[1] Zpracování přirozeného jazyka bývá v české i zahraniční odborné literatuře též označováno zkratkou NLP, tedy Natural language processing.

Syntaktické problémy

Tyto úkoly se týkají strukturálních vztahů mezi slovy. Jedná se například o indukci gramatiky, syntaktickou analýzu (angl. parsování) nebo segmentaci slov.

Sémantické problémy

Úkoly z této kategorie se zabývají významem a porozumění textu. Typickými problémy jsou poté strojový překlad, rozpoznání pojmenovaných entit, generování přirozeného jazyka nebo porozumění přirozenému jazyku.

Problémy týkající se zpracování řeči

Tyto úkoly mají společné to, že v nich figuruje lidský hlas. Typickým úkolem je rozpoznání hlasu, jehož cílem je definovat pro daný mluvený projev adekvátní textovou reprezentaci.

2.1.2 Souvislost s umělou inteligencí

Dříve byla většina systémů, které zpracovávaly přirozený jazyk, implementována pomocí zavedení pevně daných pravidel založených na jazykových strukturách. Tyto systémy bývají označovány jako **rule-based**, tedy systémy založené na pravidlech. Od 80. let minulého století se však začal prosazovat statistický model zpracování přirozeného jazyka. V současnosti se pro tuto oblast výzkumu používá termín **strojové učení**. Mluvíme tedy o metodách strojového učení ve zpracování přirozeného jazyka [11].

Strojové učení

Strojové učení (angl. machine learning) je podoblast umělé inteligence, která vychází ze statistických metod. Cílem strojového učení je vytvořit model, který bude dělat komplexní predikce či tvořit rozhodnutí. Samotný model je poté tvořený z algoritmů strojového učení a dat, které se nazývají trénovací data. Častý omyl bývá, že právě volba algoritmu je klíčový aspekt v kvalitě výsledného modelu. Naopak jsou to právě data, která nejvíce ovlivňuji úspěšnost výsledného modelu [30].

Existují 3 kategorie algoritmů strojového učení:

- **Učení s učitelem**

  Princip algoritmů z této kategorie spočívá v tom, že pro každý krok učení je známá požadovaná odezva a systém je tak okamžitě informován o aktuálním hodnocení poslední akce. Tyto algoritmy se používají na regresi a klasifikaci. Rozdíl mezi nimi je, že regrese má výstupní proměnou spojitou hodnotu, tj. číslo, zatímco při klasifikaci je výstupem diskrétní proměnná – kategorie. Mezi algoritmy patřící do této skupiny se řadí například lineární regrese, logistická regrese, neuronové sítě nebo tzn. Support vector machines [30, 33].

- **Učení bez učitele**

  Tato kategorie spočívá v nalezení podobností ve vstupních vektorech trénovací množiny – žádná podpůrná informace není obvykle dostupná. Tímto algoritmem je například k-means nebo k-nearest neighbors [33].

- **Posilované učení**

  Tato kategorie se od algoritmů učení s učitelem odlišuje tím, že systém provádí akce, po jejichž provedení dostává „odměnu“ – ta může být pozitivní i negativní a navíc různě veliká. Podstatné je, že každá akce ovlivňuje i následující akce a systém musí maximalizovat výsledný součet všech odměn [33].

Výhody použití strojového učení při zpracování přirozeného jazyka

Systémy, které jsou založené na strojovém učení, mají oproti systémům založených na pravidlech několik výhod [19]:

- Díky algoritmům učící se na velké množině dat jsou modely založené na strojovém učení více robustní, jelikož dokáží lépe reagovat na neznámé vstupy.
- Modely založené na strojovém učení získávají vyšší přesnost při dodání stále většího počtu trénovacích dat, přičemž komplexnost řešení zůstává zachována. Zatímco u systémů založených na pravidlech se vyšší přesnost získá přidáním dalších pravidel, čímž se však zvyšuje komplexnost systému a tedy i celá správa a modifikace systému.
- Z podstaty strojového učení vychází, že učící se procedury se automaticky zaměří na běžné případy. U dříve používaných systémů založených na pravidlech není však triviální tyto běžné případy identifikovat a formalizovat je.

2.2 Historie chatbotů

Pro pochopení současného zájmu o chatboty je vhodné nastudovat jejich historii a jejich novodobé směřování [25].

2.2.1 Turingův test

Před samotným popisem historie chatbotů je vhodné vysvětlit pojem **Turingův test**, který je často skloňován v souvislosti s chatboty a umělou inteligencí. Turingův test slouží na posouzení, zdali se systém chová inteligentně. Alan Turing se obecně zamýšlel nad tím, co znamená pro stroj myslet a dospěl k názoru, že na to nelze zodpovědět, jelikož pojmy stroj a myšlení nelze dávat dohromady. Navrhl proto empirický test, který funguje na principu, že se člověk dotazuje zároveň jiného člověka a počítačového programu, aniž by věděl, kdo je kdo. Pakliže nedokáže pomocí otázek jasně určit, kdo je člověk a kdo je počítačový program, je poté program označen za inteligentní [21].

Později se však ukázalo, že Turingův test není dostatečný. John Searle představil **Argument čínského pokoje**, který ukazuje, že pomocí pouhých odpovědí na dané otázky nelze prokázat, že je daný systém inteligentní. V tomto myšlenkovém experimentu figuruje uzavřená místnost plná textů v čínském jazyce a člověk v ní. Uživatelé pokládají boxu otázky v čínštině, člověk vevnitř boxu na základě algoritmu transformuje vstupní řetězec na výstupní řetězec. Takový systém se tváří jako inteligentní, avšak díky člověku vykonávající manuální práci ve skutečnosti systém čínsky nerozumí [4].

2.2.2 První chatbot ELIZA

Počítačový program ELIZA byl prvním experimentálním chatbotem, který byl dokončen v roce 1966. Jeho autorem je Joseph Weizenbaum z výzkumného institutu *MIT Computer*

Science and Artificial Intelligence Laboratory. Chatbot ELIZA ve skutečnosti nerozuměl úmyslu uživatele a ani nedokázal držet kontext mezi jednotlivými zprávami. Tento chatbot se skládal ze dvou částí. Samotného programu ELIZA, který sloužil k analýze slov, a datového souboru obsahující jednotlivá klíčová slova a transformace, který se nazýval Skript [31].

Nejznámější konfigurace chatbota byla nazývaná DOCTOR a často se při zmínce o botu ELIZA myslí právě DOCTOR. Chatbot simuloval konverzaci s terapeutem, který užívá rogeriánský přístup. DOCTOR k tomu využíval jednoduché techniky – opakoval uživatelovi jeho vlastní výroky a vytvářel nové otázky na základě klíčových slov, které nalezl ve větě. Úspěch bota zaskočil i samotného autora, který neočekával, že by mohli uživatelé vnímat bota jako skutečného člověka. Taktéž se přemýšlelo o využití programu DOCTOR pro opravdové terapeutické účely [31].

Přestože byl ELIZA první chatbot, přinesl některé techniky návrhu chatbotů, které se používají dodnes. Například, když program ELIZA nenalezl klíčové slovo v textu a tedy neznal správnou odpověď, odpovídal uživateli otázkou, aby udržel konverzaci. Typicky to byla otázka typu: *"To je velmi zajímavé, můžete mi o tom povědět více?"*.

2.2.3 Následující pokrok v oblasti chatbotů

Dalším významným chatbotem byl program **PARRY** vytvořený v roce 1972. Zatímco chatbot ELIZA simuloval terapeuta, PARRY simuloval pacienta trpícího paranoidní schizofrenií. Tento chatbot byl více komplexní a vážný. Ve stejném roce poté proběhla slavná konverzace bota ELIZA s PARRY.

Mezi další známé chatboty se řadí bot **Jabberwacky**, který byl vytvořený v roce 1981. Specifikum tohoto chatbota bylo, že se snažil napodobovat lidskou konverzaci a udržet ji s uživatelem co nejdéle. Primární účel bota Jabberwacky bylo ale především pobavení uživatelů. Dále lze zmínit chatbota **Dr. Sbaitso**, který byl vytvořen pro systém MS DOS. Tento bot, podobně jako ELIZA, simuloval psychoterapeuta. Známý je taktéž chatbot **A.L.I.C.E.**, který byl vytvořený v roce 1995. Tento bot získal spoustu ocenění. Nejvýznamnější byla trojnásobná výhra Loebnerovi ceny[2]. Kolem přelomu 21. století vznikl též bot **Smarterchild**, který byl integrován do platformy nazývané AOL Instant Messenger. V jednu dobu měl tento bot až 30 milionů uživatelů.

Revoluci a novodobý zájem o chatboty spustila však až společnost Apple v roce 2010 se svým virtuálním asistentem **Siri**, který byl integrován prvně do mobilních telefonů iPhone. Siri lze ovládat pomocí hlasu a nabízí spoustu uživatelských funkcí. Osobní asistent Siri umožňoval již od první verze vytáčet kontakty z adresáře, psát sms zprávy, plánovat události, vyhledávat informace na internetu a mnohé další služby. Mimo jiné, Siri spolupracuje i s aplikacemi třetích stran nainstalovaných v mobilním zařízení. Odpovědí na Siri byl systém **S Voice** od společnosti Samsung, který byl představený na mobilní zařízení Samsung S3 v roce 2012. Dalším asistentem byl systém **Google Now**, taktéž představený v roce 2012. S menším zpožděním v roce 2015 reagoval i Microsoft se svým virtuálním asistentem se jménem **Cortana**. Téhož roku byl představen i systém **Alexa** od společnosti Amazon. Alexa cílí především na ovládání chytré domácnosti. Uživatelé mohou komunikovat s Amazon Alexa pomocí inteligentního reproduktoru.

Nejnovější evoluci způsobila společnosti **Facebook** v roce 2016, když oznámila podporu chatbotů ve své chatovací aplikaci Facebook Messanger. Začátkem roku 2018 se na platformě

[2]Loebnerova cena je každoroční soutěž programů využívající techniky umělé inteligence za účelem simulace lidské konverzace. Soutěž je založena na Turingově testu [20]

Messanger vyskytuje více než 300 000 chatbotů[3]. Tato podkapitola byla převzata z [20, 25, 27].

2.3 Současné využití chatbotů

V současnosti existuje spoustu možností využití chatbotů ať již v podnikovém nebo zábavním průmyslu. Následující text obsahuje výčet těch nejdůležitějších oblastí využití. Je nutné taktéž poznamenat, že chatbot nemusí nutně náležet pouze do jedné kategorie.

Obchod

Jednou z oblasti využití chatbotů je elektronické obchodování. Konverzační chatboti obecně usnadňuji nakupování, jelikož díky interakci dochází k většímu zapojení zákazníka než v klasickém internetovém obchodě. Tito chatboti mohou přirozeně modelovat konverzaci s prodavačem v kamenném obchodě. Do této oblasti se může řadit například nakupování na Amazonu, ale taktéž třeba objednání pizzy nebo jízdy pomocí aplikace Uber integrované v oblíbeném konverzačním nástroji [27].

Příkladem takového bota je český chatbot Goodlok stejnojmenné firmy, která rozváží po Praze ovocné nápoje. Unikátní je, že se společnost Goodlok rozhodla nevytvářet webové rozhraní pro objednání nápojů – na řešení objednávek jim stačí jednoduchý bot pro Facebook Messanger platformu, viz obrázek 2.1.

Business neboli podnikové využití

Do této kategorie obecně spadají chatboti, kteří podporují uživatele ve vykonávání jejich práce. Patří sem tedy chatboti pro řízení lidských zdrojů (např. plánování dovolených), správu vybavení kanceláře (např. možnost kolektivního sbírání objednávek), podporu marketingu, obsluhu CRM systému a další aplikace [27].

Příkladem je bot Nikabot pro chatovací platformu Slack. Tento chatbot je určen především projektovým manažerům, jelikož umožňuje sledovat, na čem jeho tým zrovna pracuje nebo třeba kde v rámci projektu vznikají časové prodlevy [27].

Osobní asistenti

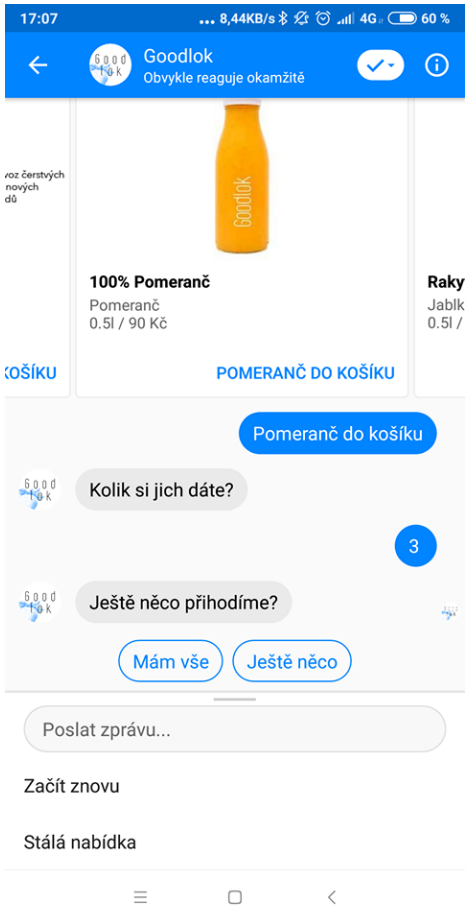
Tito boti se vyznačují tím, že zvyšují produktivitu uživatele a ulehčují mu život. Tito boti mohou spravovat uživatelův *to-do* list, pomáhat při cestování, dělat lékařské diagnozy nebo v roli osobního trenéra můžou pomoci uživateli zhubnout váhu a obecně podpořit zdravý životní styl nebo třeba spravovat uživateli finance [27].

Zajímavým příkladem takového osobního asistenta je chatbot Amy od společnosti x.ai. Účel tohoto bota je plánovat schůzky pomocí emailové komunikace. Funguje tedy jako virtuální sekretářka. Uživateli stačí přidat do kopie emailu právě bota Amy, která se dále postará o nalezení vhodného termínu schůzky pro obě strany. Poté pošle oběma stranám pozvánky a postará se o připomenutí jistou dobu před schůzkou [27].

Notifikace

Notifikační chatboti mají za úkol oznamovat zajímavé události a mohou v budoucnosti nahradit notifikační emaily či notifikace z mobilních aplikací. Standardně mohou tito chatboti

[3] https://venturebeat.com/2018/05/01/facebook-messenger-passes-300000-bots/



Obrázek 2.1: Ukázka konverzace s chatbotem Goodlok, který umožňuje nákup ovocných džusů na platformě Facebook Messenger (snímek obrazovky).

uživateli posílat zprávy, které ho zajímají, tedy například novinky ze světa financí, sportu a podobně. Taktéž to můžou být notifikace ohledně snížení ceny produktu, růstu hodnoty akcií nebo dokonce oznámení, že dítě dorazilo v pořádku domů [27].

Příkladem je NBC News bot, který funguje na platformě Facebook Messanger, který pravidelně odesílá uživateli novinky ze světa podle jeho nastavených priorit.

Zákaznický servis či FAQ boti

V dnešní době se jedná o jeden z nejčastějších typů chatbotů. Princip spočívá v tom, že bot slouží v předních řadách zákaznické podpory, kde řeší požadavky a dotazy interních zaměstnanců nebo externích zákazníků. Podle experimentů může bot pokrýt až 40% všech interních a externích požadavků, což dokáže velmi ušetřit rozpočet firmy. V současnosti se často používá kombinace chatbota s aktivním týmem zákaznické podpory, který řeší požadavky, které bot nezvládne zodpovědět [27].

Integrační boti

Jedná se chatboty, kteří dokáží komunikovat se službou třetích stran a tím ji integrovat do své chatovací platformy.

Typickým příkladem je Saleforce CRM, který je integrován do platformy Slack. Jedna z výhod Saleforce CRM spočívá v tom, že uživatelé mohou v konverzaci hledat informace, které potřebují o konkrétních zákaznících, přímo v konverzaci se svým kolegou. Dalším příkladem je Statsbot, který sbírá informace z Google Analytics a jiných marketingových systémů a integruje je přímo do platformy Slack [27].

Hry a zábava

Poslední kategorií jsou boti, kteří se zaměřují čistě na pobavení uživatele. Druhů takovýchto botů může být hodně. Někteří chatboti můžou hrát s uživatelem hry na bázi hádanek a kvízu, další můžou posílat zábavné obrázky, zatímco jiní boti (především ti, kteří interagují s uživatelem pomocí hlasu) mohou vyprávět pohádky na přání [27].

2.4 Klasifikace chatbotů

Pro správný návrh chatbotů je vhodné uvést jejich klasifikaci do několika kategorií. Díky klasifikaci je poté možné nad botem uvažovat s vyšší abstrakcí. Správné zvolení kategorie je poté pro úspěch chatbota klíčové [27].

2.4.1 Klasifikace podle domény

Chatboty lze dělit podle domény, nad kterou pracují [27]:

1. Chatboti s **uzavřenou doménou** nebo také doménově specifičtí boti, kteří představují právě jednu službu či produkt. Tito chatboti dokáží vést konverzaci na téma, na které byly přizpůsobeni. Příkladem může být bot, který představuje cestovního asistenta. Tento chatbot dokáže pomoc s rezervací hotelu a letenek. Když se ho však uživatel zeptá na cokoliv, co není spojené s cestováním, dostane univerzální chybovou odpověď, že dané otázce nerozumí.

2. Chatboti (někdy též superboti) pracující na **otevřené doméně** dokáží vést s uživatelem konverzaci na jakékoli téma nebo případně pomoc s kterýmkoliv úkolem. Toho standardně dosahují textovými transformacemi, integrací více služeb nebo vyhledáváním vhodných odpovědí pomocí internetu. V dnešní době jde typicky o univerzální boty jako Siri.

V dnešní době se více využívá chatbotů pracujících na uzavřené doméně, jelikož dokáží modelovat lépe konkrétní službu a taktéž jsou jednoduší na tvorbu. Chatboti pracující na otevřené doméně se využívají často na konverzační účely [27].

2.4.2 Klasifikace podle typu konverzace

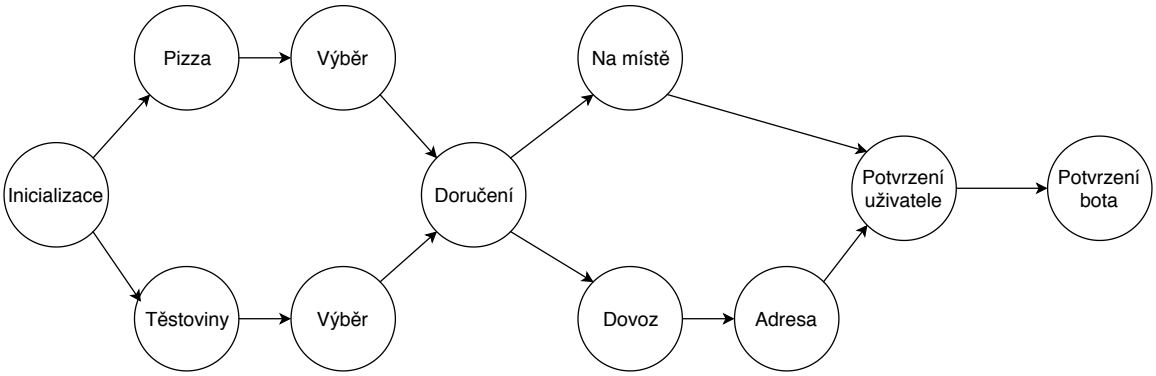
Existují základní dva typy konverzace [27]:

1. Konverzace zaměřená na **úkol**. Tento typ konverzace se vyznačuje tím, že uživatel chce po chatbotovi vykonání určité akce, kolem které se celá konverzace točí. Tyto konverzace je obecně lehčí modelovat. Chatbot při ní zpravidla následuje navržený konverzační tok[4], který udává jednotlivé kroky vedoucí k dosažení cíle. Příkladem může být objednání pizzy. Konverzační tok je vždy stejný – uživatel zvolí druh pizzy,

[4] V zahraniční literatuře se lze často setkat s výrazem flow.

poté se rozhodne, zdali chce pizzu doručit a vyplní adresu doručení, kterou poté potvrdí. Tento druh konverzace lze taktéž modelovat pomocí diagramů, viz obrázek 2.2.

2. Konverzace zaměřená na **téma**, jejímž cílem je výměna informací a nápadů. Tato konverzace je mnohem méně řízená, protože, jak v reálné diskuzi, uživatelé mohou přeskakovat z jednoho tématu na druhé, přičemž tato konverzace nemusí mít ani cíl. Tyto konverzace je složitější modelovat.



Obrázek 2.2: Diagram konverzačního toku chatbota sloužícího pro objednání jídla (inspirováno [27]).

2.4.3 Klasifikace podle tvorby odpovědi

Další klasifikace kategorizuje chatboty podle toho, jak tvoří odpovědi. Existují dvě možnosti [32]:

1. **Retrieval-based** metoda, která je založená na vyhledání vhodné odpovědi v databázi podle zvolených příznaků. Těchto nalezených odpovědí může být více. Zvolená metrika pro výběr poté může být například ohodnocení nejlepších odpovědí nebo výběr náhodné odpovědi ze skupiny. Právě náhodný výběr podobných odpovědí se využívá často, aby chatbot neodpovídal stále stejně a tím tedy působil více lidsky a méně strojově, což zlepšuje celkovou uživatelskou zkušenost.

2. **Generation-based** je metoda, která je charakteristická tím, že chatbot generuje pro každou otázku novou odpověď. Tedy takovou, kterou nemá nikde uloženou v databázi a zpravidla může odpovědět zcela novým způsobem. Tento přístup je ovšem složitý a musí překonávat spoustu problémů – například generování gramaticky správných vět.

Ve světě informačních technologií se používá zatím hlavně první metoda, jelikož je jednoduší na implementaci a vývojář má plnou kontrolu ohledně výstupní odpovědi. Lze taktéž říci, že Retrieval-based chatboti jsou více vhodní pro konverzace zaměřené na úkol (viz klasifikace výše), kdežto Generation-based chatboti plní dobrou funkci v obecných komunikačních tématech. Jelikož je generování odpovědí složitý úkol i pro výzkumné týmy, používají se taktéž kombinace obou principů.

2.4.4 Další dělení

Chatboti jdou dále kategorizovat, není však nutné ponořovat se do dalších podrobností. Například chatboty lze dělit podle počtu uživatelů se kterými komunikují – někteří chatboti mohou zároveň komunikovat v jednom kanálu s více uživateli. Jiné členění by mohlo být podle typu osobnosti bota a tak dále [27].

2.5 Platformy pro běh bota

Kanál nebo také platforma je místo, na kterém mohou uživatelé komunikovat s chatboty [20]. Před samotnou tvorbou chatbota je nutné si ve fázi analýzy promyslet, na jakou platformu bude chatbot cílit. Špatně cílený chatbot bude neúspěšný a uživatelé ho nebudou využívat. Tato práce se nezabývá analýzou jednotlivých platforem a doporučení, kde chatbota publikovat, pro přehled je zda však uveden seznam nejdůležitějších platforem současnosti [27]:

- Email,
- SMS,
- webová stránka,
- Facebook Messanger,
- Slack,
- Amazon Alexa,
- Microsoft Skype.

Kapitola 3

Dostupné technologie pro tvorbu chatbotů

Tato kapitola prozkoumává jednotlivé přístupy a technologie, které mohou být použity pro tvorbu chatbotů, a které se v současnosti aktivně používají. Jednotlivé přístupy byly analyzovány a klasifikovány do 3 kategorií podle úrovně znalostí, které jsou kladeny na vývojáře daného chatbota. První část (3.1) se zabývá vysokoúrovňovým přístupem. Tento přístup je charakteristický tím, že je autor schopný tvořit chatboty bez znalosti programování – je tedy určen primárně pro začátečníky. Technologie v této kategorii jsou často nazývány jako *Bot platforms*. Část druhá (3.2) řeší přístup střední úrovně, kde je již požadavek, aby měl vývojář základní znalostí programování. Technologie v této skupině jsou označovány jako *Bot frameworks*. Poslední část (3.3) se krátce zaměřuje na nízkoúrovňový přístup, který se vyznačuje tím, že nevyužívá žádné technologie specializované na tvorbu chatbotů [13].

3.1 Vysokoúrovňový přístup

Jedná se o nejnovější přístup k tvorbě chatbotů, který je spojován s podporou botů na platformě Facebook Messanger. Grafické rozhraní v těchto nástrojích může být použito k modelování celého konverzačního toku. Výhoda tohoto přístupu je především rychlost a jednoduchost návrhu, který zvládne i uživatel bez znalostí programovacích jazyků. Lze se tedy lépe zaměřit na modelování daných konverzačních toků, jelikož uživatel nemusí přemýšlet nad technickou stránkou. Nevýhodou je poté omezení v rámci daného nástroje, který nemusí podporovat vše, co tvůrce potřebuje modelovat. Další nevýhoda je poté fakt, že tyto nástroje bývají často placené a neumožňuji nasazení bota na vlastním serveru [20].

Služeb na vytvoření chatbota tímto způsobem existuje celá řada a lze říci, že rozdíly mezi nimi nejsou příliš velké. Mezi nejznámější patří zejména Chatfuel, Recast.ai, ManyChat, Pandorabots, Rebot.me, FlowXO, Reply.ai a mnohé další [20]. V následující části se nachází krátké představení jedné ze služeb.

3.1.1 Chatfuel

Chatfuel[1] je webový nástroj, který umožňuje tvorbu Facebook Messanger chatbotů modulárním způsobem. Hlavním cílem je tvořit boty na podporu marketingu a prodeje. Dále umožňuje vývojáři monitorovat chatbota a spravovat jeho výkonnost pomocí analytických

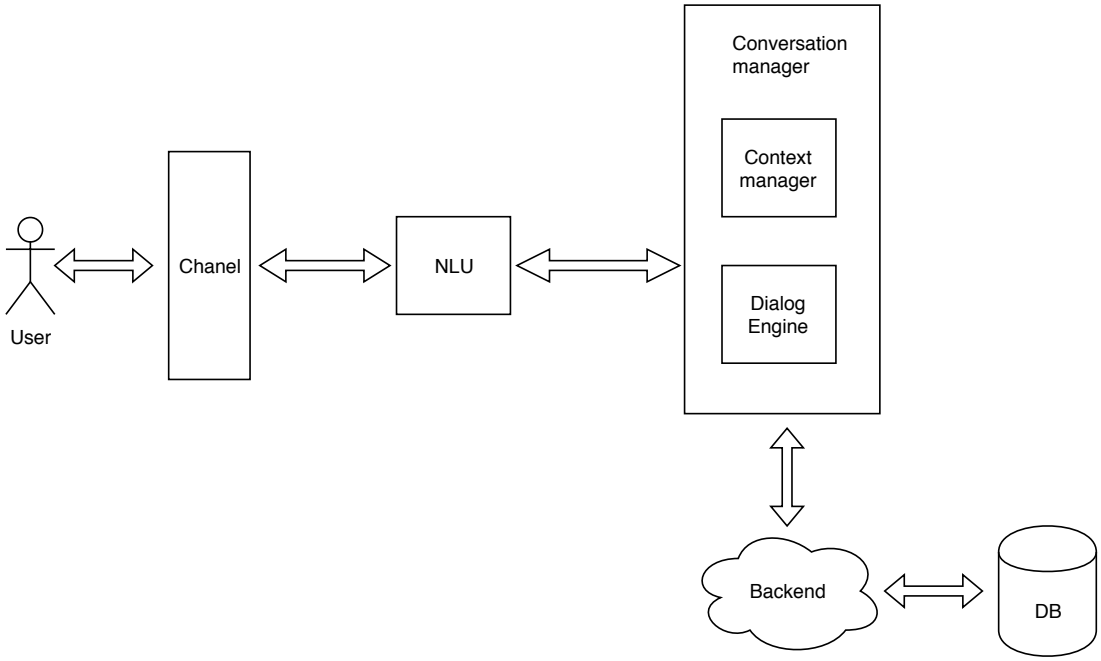
[1] Domovská stránka webové aplikace: https://chatfuel.com

nástrojů. Jedná se o komerční nástroj, který však nabízí licenci zdarma s určitým omezením [20].

3.2 Přístup střední úrovně

Tento přístup je charakteristický tím, že oproti minulému přístupu je nutné, aby měl vývojář znalosti programování. Jedná se o sadu nástrojů a před připravených kusů kódu umožnující programátorovi tvorbu chatbotů [13]. Oproti nejnižšímu přístupu se však liší tím, že vývojář nemusí mít znalosti ohledně zpracování přirozeného jazyka a nemusí znát metody strojového učení.

Na obrázku 3.1 je zobrazena architektura takového chatbota. Tento chatbot má dva základní moduly - modul pro porozumění zprávy uživatele a modul plnící funkci konverzačního manažera (někdy označován jako dialog engine). Po analýze dostupných technologií lze zjistit, že existují aplikační rámce, které v sobě integrují jak modul porozumění přirozeného jazyka, tak i konverzačního manažera (3.2.1). Dále existují technologie nabízející pouze porozumění přirozeného jazyka (3.2.2) a technologie umožnující tvorbu konverzačních manažerů (3.2.3) nezávisle od sebe.



Obrázek 3.1: Obecná architektura chatbota s využitím NLU služby (inspirováno [20]).

3.2.1 Aplikační rámce nabízející kompletní řešení

Jedná se o aplikační rámce, které integrují jak porozumění přirozeného jazyka, tak řízení konverzace a generování odpovědi pomocí konverzačního manažera. Tyto aplikační rámce nabízejí komplexní řešení pro tvorbu chatbotů. Standardně lze pomocí webového rozhraní vytvořit model pro zpracování přirozeného jazyka a definovat adekvátní odpovědi vůči záměrům uživatele. Tyto aplikační rámce nabízejí možnosti rozšíření funkčnosti (například komunikace s databází či webovými službami) pomocí klientských aplikací. Klientské aplikace často podporují spoustu platforem na vývoj. Některé aplikační rámce dokáží vygenero-

vat kostru klientské aplikace na základě definování záměrů ve webovém rozhraní. Nejčastěji aplikační rámce podporují klientské aplikace na platformě Node.js, Ruby, Java a PHP. Některé aplikační rámce (např. Amazon Lex) umožňují generovat klientské aplikace i pro mobilní zařízení na platformě Android a iOS.

Přehled aplikačních rámců

- **Dialogflow**

  Službu Dialogflow (dříve známou pod názvem API.AI) vlastní společnost Google a zároveň je zprostředkovaná pomocí infrastruktury společnosti Google. Společně s platformou od společnosti IBM se jedná o jednu z nejstarších platforem. V základní verzi lze tento aplikační rámec využít bezplatně [20].

- **IBM Watson Assistant**

  IBM Watson Assistant patří do rodiny IBM Watson, což je soubor několika služeb a technologií nabízející aplikaci umělé inteligence do podnikového prostředí. IBM Watson Assistant je IBM řešení pro tvorbu chatbotů [27]. V době psaní diplomové práce nabízí v beta verzi podporu českého jazyka[2].

- **Amazon Lex**

  Aplikační rámec Amazon Lex byl zpřístupněný programátorům v dubnu roku 2017. V jádru využívá model strojového učení, který využívá proprietární produkt Amazon Alexa. Amazon Lex se taktéž specializuje na rozpoznávání hlasu [2].

3.2.2 Porozumění přirozeného jazyka

V současnosti se k tvorbě chatbotů používají služby poskytující porozumění přirozeného jazyka[3], které dokáží extrahovat entity a záměry z vět. Běžně se jedná o webovou službu, která má publikované aplikační rozhraní. Vstupem do této služby je věta uživatele, výstupem jsou poté data ve formátu JSON reprezentující záměr uživatele a jednotlivé pojmenované entity v dané větě, viz obrázek 3.2. Většina klíčových společností v IT průmyslu, jako Google, Microsoft, Facebook či IBM, na těchto službách pracovala a následně je publikovala vývojářům k využití s různými licenčními podmínkami [20].

Přehled služeb poskytujících porozumění přirozeného jazyka

Služeb poskytujících porozumění přirozeného jazyka je celá řada. V následující části jsou krátce představeny ty nejdůležitější.

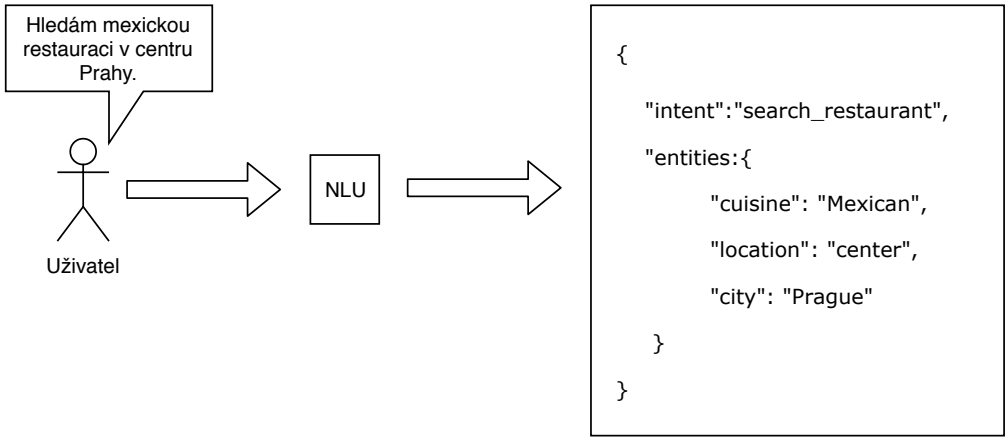
- **Wit.ai**

  Jedná se o službu, kterou vlastní společnost Facebook [27]. Zajímavostí je podpora českého jazyka[4] a možnost využít Wit.ai zdarma i pro komerční užití.

[2] Watson Assistant – podporované jazyky https://cloud.ibm.com/docs/services/assistant?topic=assistant-language-support

[3] Porozumění přirozeného jazyka – často označované jako NLU (Natural language understanding)

[4] Wit.ai – podporované jazyky https://wit.ai/faq



Obrázek 3.2: Zpracování přirozeného jazyka do strukturované podoby pomocí NLU služby. Výstupní formát poté odpovídá formátu programu Rasa NLU (zdroj vlastní).

- **LUIS**

  LUIS spadá do rodiny služeb Azure Cognitive Service od společnosti Microsoft. Cílem tohoto portfolia je přinášet do aplikací inteligentní služby. Příkladem je rozpoznávání řeči, analýza obrázků či inteligentní vyhledávání [26]. Služba LUIS bohužel nepodporuje češtinu. Bezplatná verze je omezena určitým objemem požadavků za sekundu.

- **Rasa NLU**

  Rasa NLU není online služba, jako produkty výše. Rasa NLU je program, který si lze nainstalovat a využívat NLU lokálně, případně si nasadit tento program na libovolný server. Rasa NLU je taktéž unikátní v tom, že je kompletně open-source [25].

3.2.3 Konverzační manažer

Konverzační manažer řídí tok konverzace. Vstupem do konverzačního manažera je sémantická reprezentace uživatelovy věty, přičemž tento systém rozhoduje o tom, jaká bude odpověď. Konverzační manažer taktéž spravuje konverzační kontext – standardně ve formátu klíč-hodnota. Dále tento systém umožňuje integraci s serverovými službami, od níž získává, případně zapisuje, data [20].

Přehled aplikačních rámců pro tvorbu konverzačních manažerů

- **Microsoft Bot Framework**

  Společně se službou LUIS nabízí Microsoft taktéž **Bot Framework**[5], což je aplikační rámec umožnující tvorbu chatbotů. Microsoft Bot Framework je vyvíjen jako open-source. Aplikační rámec umožňuje řídit konverzaci a je obecně na službě LUIS nezávislý. Vývojář tedy může použít jinou službu pro porozumění přirozeného jazyka nebo si vytvořit vlastní metriku (např. textovou transformaci) [27]. V aktuální čtvrté verzi je možné implementovat bota na dvou primárních platformách – C# a Node.js.

[5] Github aplikačního rámce Microsoft Bot Framework – https://github.com/microsoft/botframework-sdk

V beta verzi je poté možné využívat i platformy Python a Java. Provozovat chatbota lze poté na vlastním serveru nebo na platformě Microsoft Azure.

- **Rasa Core**

  Rasa Core je open-source aplikační rámec pro řízení konverzací, který je postaven na platformě Python. Lze ho nasadit na svém serveru. Zajímavá vlastnost je migrace – framework umožňuje využívat a dále rozšiřovat modely z jiných aplikačních rámců jako je Dialogflow [25].

3.3 Nízkoúrovňový přístup

Nízkoúrovňový přístup spočívá v tom, že vývojář nevyužívá žádné technologie a aplikační rámce, které jsou určeny pro tvorbu chatbotů. Výhoda spočívá v tom, že vývojář není závislý na technologiích třetích stran, které můžou měnit licenční podmínky a zároveň má volnost nad vlastní implementací a optimalizací daného chatbota. Nevýhodou je potom mnohem větší požadavek znalostí na vývojáře, který musí umět aplikovat metody zpracování přirozeného jazyka a strojového učení. Taktéž doba vývoje bude mnohem delší, z čehož vyplývají i výrazně větší finanční náklady na tvorbu takového chatbota. Po analýze lze říci, že existuje jen několik motivací, proč stavět chatboty nízkoúrovňové:

1. Požadavky na chatbota jsou natolik složité a specifické, že nejdou implementovat pomocí stávajících aplikačních rámců. Takový chatbot může být typicky konverzační, jehož cílem je simulovat přirozenou konverzaci.

2. Vědecké a výzkumné účely – firma či jednotlivec se snaží pochopit jednotlivé principy, na nichž technologie pro tvorbu chatbotů stojí, přičemž je může chtít vylepšit.

3. Firma nebo jednotlivec chce vytvořit vlastní aplikační rámec nebo prostředí na tvorbu chatbotů.

3.3.1 Dostupné knihovny pro manipulaci s přirozeným jazykem

Knihoven umožnující zpracování přirozeného jazyka je celá řada. Majoritní většina používá programovací jazyk Python, najdou se však i výjimky v programovacím jazyce Java nebo C++.

- **NLTK**

  NLTK (Natural Language Toolkit) je základní knihovna určená pro manipulaci s přirozeným jazykem a je naprogramovaná v jazyce Python. Knihovna NLTK byla vytvořena v roce 2001 jako součástí kurzu výpočetní lingvistiky na Pensylvánská univerzitě. Knihovna poskytuje základní třídy pro reprezentaci dat, které jsou užitečné ve zpracováním přirozeného jazyka. Taktéž nabízí všechny běžné funkce užívané ve zpracování přirozeného jazyka jako syntaktickou analýzu (angl. parsing) nebo klasifikaci [19].

- **Apache OpenNLP**

  Apache OpenNLP[6] nabízí všechny základní funkce na zpracování přirozeného jazyka jako NLTK. Hlavním rozdílem je, že Apache OpenNLP funguje na platformě Java.

[6] Domovská stránka projetku – https://opennlp.apache.org/

- **spaCy**

  Jedná se alternativu k NLTK. Je to mnohem novější knihovna, která má jiný přístup než NLTK. Zatímco NLTK nabízí ke každému úkolu zpracování přirozeného jazyka několik algoritmů, které mohou být použitý, spaCy nabízí pouze jeden, který je podle autorů nejvíce vhodný. SpaCy na rozdíl od NLTK má v sobě integrovanou podporu pro tzn. word vector (viz 3.3.2), což je funkce umožnující reprezentovat slovo pomocí číselného vektoru [25].

3.3.2 Případová studie – implementace NLU služby

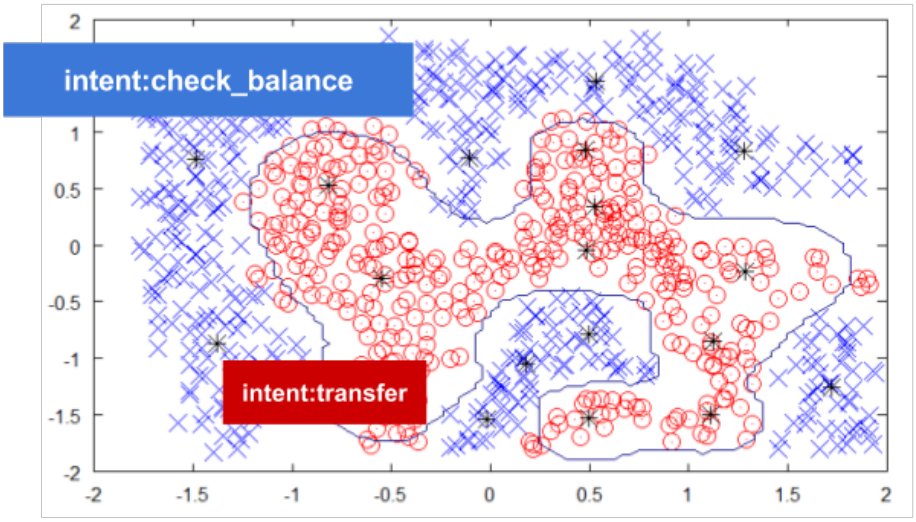
Cílem této podkapitoly je analýza implementace služeb sloužící na porozumění přirozeného jazyka. Tyto služby jsou v tvorbě chatbotů klíčové, a proto je vhodné chápat jak fungují. Porozumění přirozeného jazyka má na vstupu větu v přirozeném jazyce a výstupem jsou strukturovaná data popisující smysl věty, viz kapitola 3.2.2. Jelikož služby na porozumění jazyka jako LUIS či Wit.ai jsou proprietární, není možné zjistit, jaké technologie a metody využívají. Existuje však služba Rasa NLU, která je šířena jako open-source, u které lze zjistit, jak vnitřně funguje.

Rasa NLU využívá metody zvané Word vectors, která spočívá v převedení slova do vektorové reprezentace. Standardně se jedná o několika dimenzionální prostor, který často může obsahovat až 300 dimenzí. Jednou z důležitých vlastností tohoto prostoru je, že slova významově podobná leží v daném prostoru blízko sebe s tím, že tento rozdíl lze vypočítat. Naopak, rozdílná slova leží v prostoru daleko od sebe. Na výpočet takového prostoru lze použít knihovnu Word2vec od českého autora Tomáš Mikolova obsahující několik algoritmů určených k tomuto účelu nebo případně algoritmus GloVe[7]. Dále jsou nutná trénovací data. Například knihovna spaCy obsahuje předem připravený slovník vektorových reprezentací slov, který byl vytvořen pomocí algoritmu GloVe, přičemž k trénování byla použita kolekce *Common Crawl*[8], což je obrovská kolekce textových dat. Můžou být však použita například data extrahovaná z wikipedie. Použitím slovníku knihovny spaCy tedy člověk získává pro dané slovo číselnou reprezentaci.

Tímto způsobem Rasa NLU získává číselnou reprezentaci slov. Číselná reprezentace věty lze poté získat pomocí aritmetických operací, kdy lze jednotlivé číselné reprezentace slov ve větě sečíst a následně průměrovat. V Rasa NLU (a v podobných NLU službách) je nutné zadat příklady k jednotlivým záměrům, přičemž je důležité, aby jednotlivé ukázkové věty měly doopravdy stejný význam. Rasa NLU poté vytváří dvourozměrný prostor na základě jednotlivých číselných reprezentací ukázkových vět s určitou tolerancí a pakliže je poté vstupní uživatelova věta v tomto prostoru, je prohlášeno, že splňuje daný záměr [7]. Na obrázku 3.3 lze vidět ukázku prostoru, který si Rasa NLU vytváří. Graf na obrázku vyjadřuje, jak mohou vypadat hranice jednotlivých záměrů v číselné reprezentaci. Oblast vyplněná červeně představuje větu, jejímž účelem je provést transakci, zatímco modrá oblast vyjadřuje věty, v nichž je záměr zkontrolovat zůstatek na účtu [7].

[7] GloVe - GloVe je algoritmus strojového učení bez učitele, který je určený k získání vektorové reprezentace, jenž je vyvíjen na Stanfordské univerzitě [24].

[8] Common Crawl je volně přístupná kolekce dat, kterou lze použít na datovou analýzu - http://commoncrawl.org



Obrázek 3.3: Graf zobrazující záměry vět v číselné reprezentaci (převzato [7]).

Kapitola 4

Analýza a specifikace požadavků na chatbota

Tato kapitola analyzuje a specifikuje problém, který je řešený v této diplomové práci. V první části (4.1) je stručně představena společnost Allium s.r.o., která se podílela na řešení práce. Druhá část (4.2) uvádí čtenáře do problematiky správy podnikových informací. Třetí podkapitola (4.3) popisuje konkrétní systém zaměřující se na správu podnikových informací – systém Pimics společnosti Allium s.r.o. Poslední část (4.4) poté představuje funkční požadavky na chatbota řešeného v této práci.

4.1 Představení společnosti Allium s.r.o.

Společnost Allium s.r.o. působí na českém trhu od roku 1994. Od začátku svého působení se firma specializuje na implementaci, zavádění a správu **podnikových informačních systémů** daným zákazníkům. Prvním nasazeným softwarem společnosti se stal v roce 1996 systém řízení vztahů se zákazníky (tj. CRM) pro švýcarského zákazníka. V roce 2001 se Allium s.r.o. stalo partnerem společnosti Microsoft [1].

V současnosti zaměstnává firma několik desítek zaměstnanců a má pobočky v Praze, Brně, Ostravě a Bukurešti. Do portfolia společnosti spadají informační systémy jako správa podnikových informací, řízení skladu a optimalizace dopravy, řízení výroby, řízení vztahu se zákazníky, řízení lidských zdrojů, plánování podnikových zdrojů a další produkty. Tyto systémy pomáhá Allium implementovat a zavádět u koncových zákazníků s tím, že firma nabízí i dlouhodobou podporu hotového řešení. Společnost dále nabízí modulární zakázkový vývoj, který standardně rozšiřuje funkčnost stávajících systémů. Tyto systémy jsou postavené na platformách společnosti Microsoft jako jsou například Dynamics 365 nebo Office 365. [1].

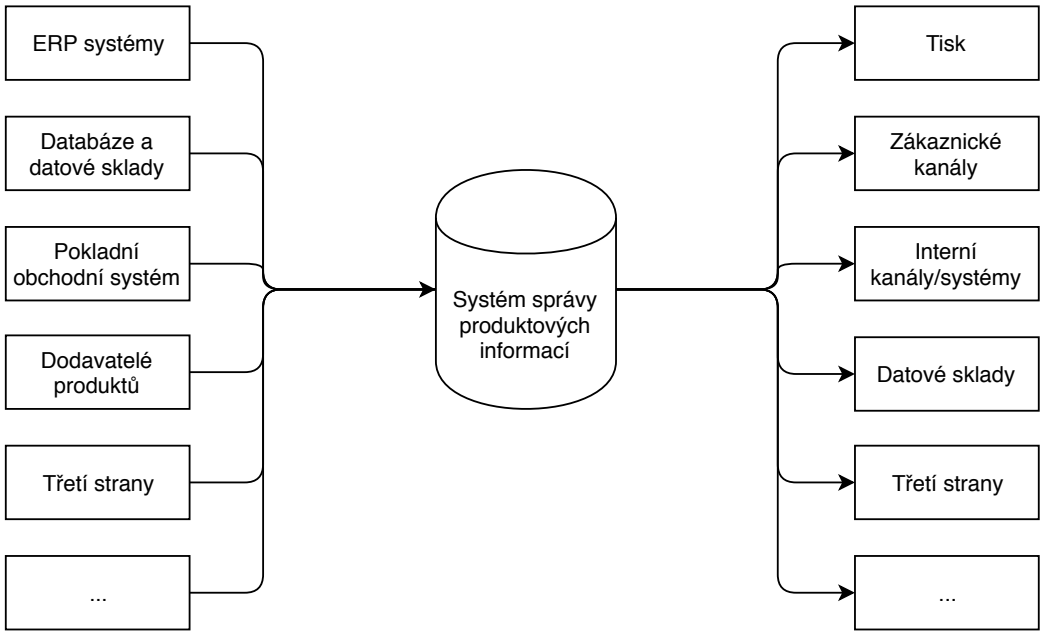
4.2 Úvod do systému správy podnikových informací

V této podkapitole je obecně představen princip správy podnikových informací.

4.2.1 Správa podnikových informací

Správa podnikových informací, zkráceně též **PIM** podle anglického Product Information Management, je relativně nový termín, jehož popularita rapidně vzrostla s popularitou in-

ternetových obchodů a obecně e-commerce. PIM systém se zajímá o procesy a technologie zajišťující centrální správu produktových informací se zaměřením na prodávání produktů prostřednictvím jednoho nebo více distribučních kanálů. Zkráceně řečeno, cílem systému PIM je shromáždit informace o produktech z více zdrojů a poté je sdílet několika kanálům [18]. Jednoduchá architektura obecného PIM systému je zobrazena na obrázku 4.1.



Obrázek 4.1: Architektura PIM systému s ukázkou možných vstupních zdrojů a výstupních odběratelů (inspirováno [18]).

Zdroje PIM systému

Zdrojem plnění PIM systému bývá především **ERP systém**[1], který řídí a spravuje výrobní, logistické a finanční zdroje či procesy společnosti. Téměř každá organizace, která má vlastní výrobu nebo případně prodává zboží, vlastní jeden či více ERP systémů. Tyhle systémy bývají standardně zdrojem všech produktů, které společnost nabízí. Identifikátor produktu použitý v ERP systému bývá většinou použitý jakožto jedinečný identifikátor i v PIM systému. Dalšími zdroji můžou být datové sklady, databáze či například pokladní obchodní systémy. Jinými zdroji můžou být dodavatelé produktů[2] či třetí strany [18].

Výstupní kanály PIM systému

Počet komunikačních, prodejních a distribučních kanálů se za posledních několik let díky internetu a mobilním zařízení rapidně rozrostl. Kanály mohou být rozděleny do **fyzických** a **digitálních**. Typickým fyzickým kanálem jsou prodejny, katalogy nebo obchodníci. Naopak digitální kanály jsou poté internetové stránky, mobilní aplikace nebo například profily na sociálních sítí. Všechny kanály potřebují informace o produktech pro úspěšnou propagaci, prodej a distribuci produktů. Jiné dělené kanálů lze poté vidět na obrázku 4.1. Na

[1] ERP systém je v českém jazyce označován jako systém plánování podnikových zdrojů .

[2] Počet dodavatelů může často velmi značný – podle studií dvě třetiny společností reportovala více než 100 dodavatelů [18].

obrázku si lze povšimnout, že ERP systémy mohou být nejen zdrojem informací, ale taktéž konzumentem například pro zpětnou validaci nebo finanční správu [18].

Motivace zavedení PIM systému

Zde je uveden krátký přehled některých benefitů, které plynou z využití PIM systému [18]:

- Rozšíření sortimentu. Pomocí strategie tzn. Dlouhého ocasu je možné nabízet díky internetovému obchodu více druhů zboží než v klasickém kamenném obchodě.
- Zkrácení doby uvedeni produktu na trh. Příkladem může být příprava produktového katalogu na rok 2019 pro několik různých trhů.
- Sjednocení uživatelské zkušenosti na všech kanálech.
- Zjednodušení komplexnosti organizace rozčleněním jednotlivých produktů do kategorií. Pro velké firmy obsahující stovky tisíc produktů a stovky dodavatelů je centrální systém správy nenahraditelný.

4.3 Seznámení s produktem Pimics společnosti Allium s.r.o.

Pimics je řešení společnosti Allium s.r.o., které slouží k vytváření a správě produktových informací. Charakteristické pro systém Pimics je to, že je plně integrován do ERP systému společnosti Microsoft. Pimics lze integrovat do ERP systému **Dynamics NAV**, který byl dříve nazýván pouze Navision nebo do systému **Dynamics 365**, což je nová cloudová[3] obdoba primárně desktopového Dynamics Nav. Dynamics 365 funguje na principu SaaS, tj. software jako služba, což je model nasazení, kdy se stará o provoz služby její poskytovatel [6].

4.3.1 Architektura produktu Pimics

Na obrázku 4.2 je zobrazena základní architektura produktu Pimics. V následující části jsou popsány jednotlivé komponenty tohoto systému [14]:

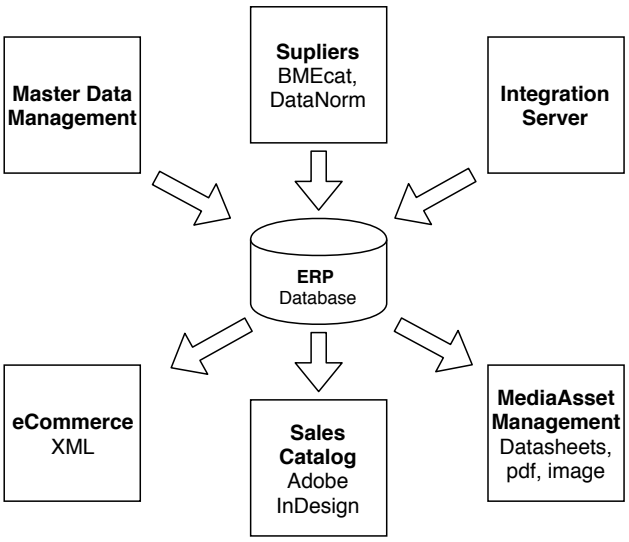
- **ERP/database**

  Jedná se o hlavní jednotku celého systému, která propojuje všechny další moduly. Jak bylo zmíněno výše, jádrem je ERP systém Dynamics NAV nebo Dynamics 365. V obou případech se jedná o produkt produktové řady Microsoft Dynamics, která se zaměřuje na podnikové informační systémy ERP a CRM pro menší až střední podniky. Klientská aplikace Dynamics NAV je k dispozici na hlavních platformách jako Windows, web či Android. Na obrázku 4.3 lze vidět základní architekturu systému Microsoft Dynamics NAV. Zapojení na obrázku obsahuje Windows klienta, dva webového klienty, Microsoft Dynamics Server Computer obsahující hlavní aplikační logiku aplikace a databázový SQL Server [6].

- **Master Data management**

  Jedná se o základní a nejdůležitější data a informace, které podnik vlastní.

[3] Cloudem je v této práci myšlen software, který je spuštěn na vzdáleném serveru. Cloudová služba je poté služba, která je zprostředkovaná vzdáleným serverem [29].



Obrázek 4.2: Architektura produktu Pimics společnosti Allium s.r.o. (inspirováno [14]).

- **Suppliers and Integration server**

  Jedná se o modul zajišťující výměnu dat mezi partnery nebo dodavateli. Modul umožňuje import dat o produktech pomocí standardizovaných mezinárodních formátů jako ETIM BMECat a DataNorm.

- **eCommerce**

  Jde o modul vykonávající publikování dat uložených v systému PIM pomocí webových služeb, které můžou využívat například internetové obchody. Jiné publikování může být pomocí emailů. Například v šabloně umožnující automatizované vydání zpravodaje (angl. newsletter).

- **Sales Catalog**

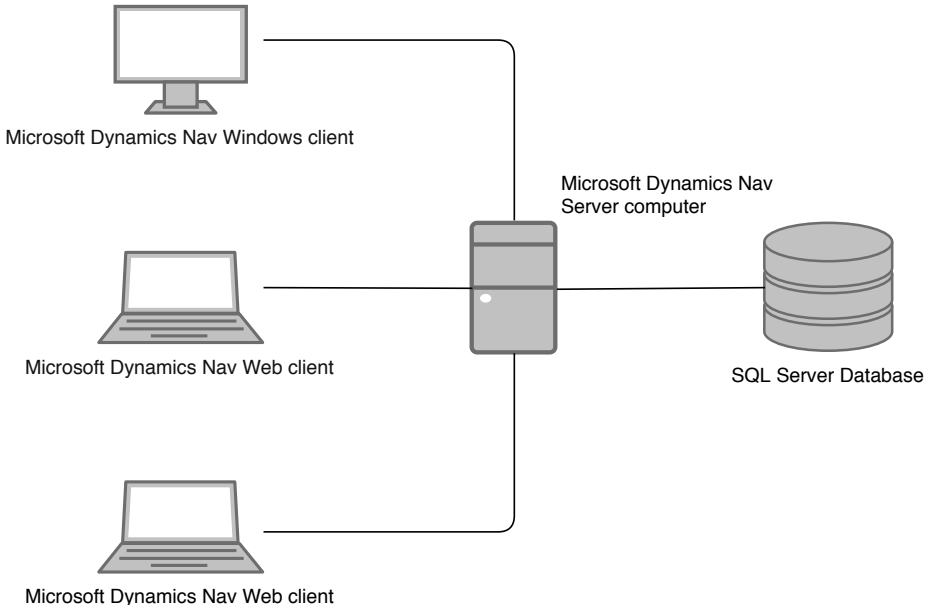
  Modul umožnující vytvoření elektronických katalogů a katalogů určených pro tisk.

- **MediaAsset Management**

  Modul MediaAsset Management zahrnuje práci s multimédii jako jsou obrázky, dokumenty ve formátu pdf nebo videa. Tyto multimédia lze taktéž přiřadit jednotlivým produktům v PIM systému.

4.3.2 Datový model produktových informací

Na obrázku 4.4 je zachycen zjednodušený model produktových informací, který slouží na základní seznámení se strukturou dat. K jednotlivým položkám bude vždy uveden příklad ze světa mobilních telefonů, který slouží k lepší představě o principu datového modelu. Je nutné poznamenat, že se jedná pouze o modelový případ. V případech, kdy se nasazuje systém správy podnikových informacích, je nutné sesbírat požadavky a navrhnout rozdělení na skupiny tak, aby korespondovalo s daným produktovým portfoliem [18].



Obrázek 4.3: Architektura systému Microsoft Dynamics Nav (inspirováno [6]).

Popis entit v datovém modelu

- **Item**

  Jedná se o koncovou položku katalogu, která obsahuje identifikační kód, podle něhož je možné tuto položku objednat. Příkladem takové položky může být *Samsung Galaxy S10 Dual SIM 128GB black*.

- **Keywords**

  Klíčová slova popisují danou položku a slouží ke kategorizaci a podpoře ve vyhledávání. Příkladem klíčového slova může být *mobilní telefon*, *mobil*, *Galaxy*, *Dual SIM* a podobně.

- **Features**

  Features jsou vlastnosti produktu. Každá vlastnost má svůj název, svoji hodnotu a typ (číslo, řetězec, logická hodnota). V kontextu mobilních telefonů se může jednat o vlastnosti typu *paměť = 64GB*, *váha = 157g* a podobně.

- **Item Group**

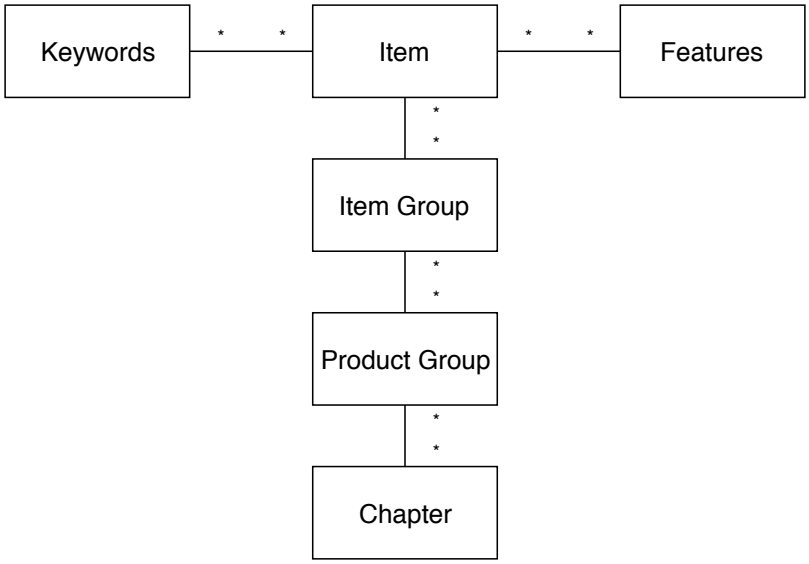
  Jedná se o skupinu předmětů. Často obsahuje jméno položky, které se dále dělí na konkrétní konfigurace. Příkladem je *Samsung Galaxy S10*.

- **Product Group**

  Product Group je skupina produktů, která sdružuje určité skupiny předmětů. V tomto případě to může být například skupina *Chytré telefony*.

- **Chapter**

  Chapter je nejvyšší skupina, která obsahuje skupiny produktů. V uvedeném případě například skupina *Mobilní telefony*.



Obrázek 4.4: Zjednodušený datový model produktových informací (zdroj vlastní).

4.4 Funkční požadavky na chatbota

V této části jsou představeny požadavky na chatbota, který jakožto doménu bude používat PIM systém představený výše. Základní myšlenka je taková, že chatbot bude spravovat pomocí textového rozhraní klientům jejich objednávky a urychlovat a zjednodušovat proces nákupu. Tento chatbot bude napojený na systém Microsoft Dynamics NAV a Pimics, díky čemuž bude mít přístup k objednávkám uživatele a taktéž přístup k produktovému portfoliu firmy. Typickým uživatelem, který bude s chatbotem komunikovat, bude zákazník firmy, od níž chce objednat dané produkty, přičemž se bude jednat především o obchodní vztah B2B[4]. V budoucnu si však lze představit využití takového chatbota i na internetovém obchodě firmy určený koncovým spotřebitelům. Ideální stav by byl takový, že společnost, které běží podnikové systémy na systému Microsoft Dynamics NAV a PIM systému Pimics, by byla schopná bez dodatečného programování velmi rychle vytvořit chatbota, který by umožňoval snadný prodej produktů společnosti.

Na obrázku 4.5 se nachází diagram případů užití chatbota. Jediným aktérem je zde uživatel, který bude s chatbotem komunikovat. V následující části jsou krátce představeny jednotlivé případy užití, které zároveň tvoří funkční požadavky:

- **Autentizace uživatele**

  Autentizace je proces ověření identity subjektu. Chatbot musí být schopen ověřit, že komunikuje s uživatelem, za kterého se uživatel vydává. Na základě autentizace poté chatbot může daného uživatele oslovovat jménem, spravovat pro něho objednávky, tvořit cílené nabídky nebo případně posílat notifikace.

- **Vyhledání produktu**

[4]B2B – jedná se vztah tzn. business-to-business, který je specifický tím, že neobsahuje koncové spotřebitele, ale naopak právě dvě podnikající strany. Příkladem může být vztah výrobní firmy a jejího dodavatele materiálu.

Jedná se o klíčovou funkcionalitu pomocí níž může uživatel vyhledávat konkrétní produkt. Uživatel může vyhledávat produkt pomocí jména produktu nebo klíčových slov, které mohou být k produktu přiřazeny. Dále lze zobrazit všechny produkty v dané kategorii. Uživatel poté může daný produkt nebo produkty přidat do aktuální objednávky nebo si může zobrazit detail tohoto produktu.

- **Přidání produktu do objednávky**

  Do aktuální objednávky lze přidávat nové produkty včetně informace o počtu kusů.

- **Zobrazení aktuální objednávky**

  Jedná se o zobrazení produktů v aktuální nedokončené objednávce včetně počtu kusů a celkové ceny. Dále chatbot nabídne výzvu zákazníkovi k dokončení objednávky. Tento požadavek lze taktéž nazvat jako zobrazení nákupního košíku.

- **Odebrání produktu z objednávky**

  Chatbot umožní uživateli odebrat produkt z existující objednávky.

- **Dokončení objednávky**

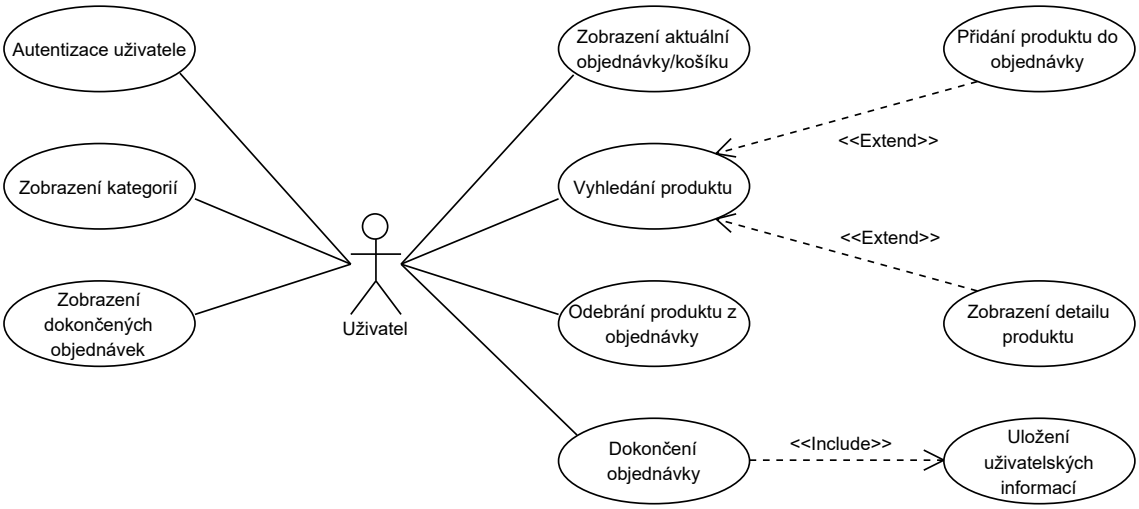
  Dokončení objednávky je cílový stav, kterého se snaží chatbot dosáhnout. Chatbot tedy musí provázet uživatele procesem nákupu tak, aby bylo procento vytvořených objednávek co nejvyšší. Samotné vytvoření objednávky poté sestává z výběru jednotlivých produktů a počet jeho kusů, které do dané objednávky spadá. Dále jsou potřeba informace o zákazníkovi, tj. jméno zákazníka, adresa a kontaktní údaje. Pokud chatbot tyto informace nemá, musí se uživatele doptat. Poté přichází na řadu rekapitulace a potvrzení objednávky. Jakmile uživatel potvrdí objednávku, chatbot vytvoří objednávku v ERP systému Microsoft NAV a pošle oznamující email zákazníkovi a příslušné osobě ve firmě.

- **Zobrazení dokončených objednávek**

  Tato funkcionalita umožňuje zákazníkovi získat historii ohledně svých nákupů. Chatbot zobrazí uživateli jeho poslední objednávky s tím, že uživatel může požádat o detail objednávky, která obsahuje jednak produkty v objednávce obsažené, ale taktéž například stav objednávky. Tedy zdali byla objednávka již vyzvednuta a podobně.

- **Zobrazení dostupných kategorií**

  Pro lepší orientaci v produktové nabídce společnosti si může zákazník vyžádat zobrazení všech produktových kategorií. Pomocí těchto kategorií poté může vyhledávat jednotlivé produkty.



Obrázek 4.5: Diagram případů užití chatbota (zdroj vlastní).

Kapitola 5

Návrh řešení

Tato kapitola představuje teoretický návrh chatbota, který bude v následující části diplomové práce implementován. První část kapitoly (5.1) představuje napojení chatbota na cílový informační systém. Druhá část seznamuje s návrhem konverzačních toků (5.2) a další část poté s technologiemi použitými pro implementaci (5.3). Čtvrtá část představuje návrh záměrů a entit pomocí služby LUIS (5.4) a poslední část popisuje celkovou architekturu řešení postavenou na cloudových technologií Microsoft Azure (5.5).

5.1 Návrh napojení chatbota na cílový systém

Při analýze možného napojení chatbota na systém Pimics, který je integrován v programu Microsoft Dynamics NAV (viz kapitola 4.3), lze zjistit, že existují dvě alternativní řešení. První řešení je, že chatbot bude mít přístup do databáze ERP systému Microsoft Dynamics NAV a do PIM systému Pimics. Druhé řešení je, že chatbot bude se systémem komunikovat pomocí vystavených webových služeb. Řešení umožnující propojení chatbota se systémem skrze **webové služby** je obecně vhodnější – jednotlivé výhody jsou rozebrány v části 5.1.1.

5.1.1 Porovnání alternativních způsobů propojení

V rámci diplomové práce bylo upřednostněno řešení, kdy bude chatbot komunikovat s PIM systémem pomocí **webových služeb**. Oproti variantě přímého napojení na databázi má toto řešení několik výhod:

- **Nezávislost** chatbota na databázovém serveru a tedy větší **robustnost**. Pakliže by se migroval databázový server na jiný, chatbot bude stále kompatibilní.
- **Jednoduchost**, jelikož není nutné pracovat s velmi rozsáhlým datovým modelem programu Microsoft Dynamics NAV.
- **Bezpečnost**. Není nutné dávat práva ke čtení a zápisu do produkční databáze další aplikaci, přičemž by mohly vzniknout potencionální bezpečnostní chyby.
- Budoucí propojení chatbota do systému **Microsoft Dynamics 365**. V současnosti totiž přímé napojení na databázi Microsoft Dynamics 365 není možné.

Pro úplnost je nutné dodat i jednu nevýhodu řešení propojení skrze webové služby. Tím je skutečnost, že webové služby poskytují již agregovaný pohled na databázi a ne vždy se lze dotázat na vše.

5.1.2 Webové služby

Webová služba nabízí způsob, jak si mohou dva systémy zapojené v jedné síti vyměňovat informace. Použitím webových služeb mohou systémy poskytnout informace přístupné jiným systémům skrze definování a publikování webového rozhraní. Toto rozhraní definuje, jaká data jsou k dispozici, jakého jsou typu a jak mohou být zpřístupněna [28].

Systém Microsoft Dynamics NAV umožňuje vytváření a publikování webových služeb dvou typů:

SOAP webové služby

Tyto webové služby používají pro přenos dat serializační formát XML. Samotný protokol SOAP poté definuje formát zpráv posílaných mezi systémy [29].

OData webové služby postavené na principu REST

REST[1] je architektonický styl založený na přenosu reprezentací zdrojů ze serveru na klienta. Webové služby, které se drží architektonického stylu REST, bývají nazývané jako RESTful webové služby. Běžně RESTful webové služby využívají pro přenos protokol HTTP – konkrétněji metody GET, POST, PUT a DELETE. V těle zprávy se používá serializační formát JSON, avšak může být nahrazen i formátem XML [29].

OData (tj. zkratka pro Open Data Protocol) je otevřený protokol vyvinutý společností Microsoft následující styl REST, který je alternativou pro SOAP webové služby. Tento protokol je standardizovaný u OASIS a je navržený pro jednoduché a standardizované dotazování tabulkových dat, přičemž ovládá i jejich vytvoření, úpravu či filtraci. Jako jiné RESTful webové služby, i OData webové služby používají identifikátory URI pro identifikaci zdrojů. Jako serializační formát je použit JSON nebo Atom Publishing Protocol [12].

Pro účel diplomové práce budou využívány právě **OData RESTful webové služby**, jelikož jsou doporučeny pro komunikaci s klientskou aplikací, zejména kvůli flexibilitě a snadnějšímu použití [6].

5.2 Návrh konverzačních toků

V kapitole 2.4.2 již bylo naznačeno, jak lze modelovat konverzace. Jednou z možností je diagram konverzačního toku. V literatuře se lze setkat i s názvem *stories*, tj. příběhy. Konverzační toky se používají k popisu dané části konverzace a zároveň umožňuji jednotlivé příběhy izolovat. Tedy místo rozvětvených toků se lze pouze odkázat na daný tok, případně rozdělit jeden tok na více toků – dílčích příběhů. Výhoda tohoto přístupu, kromě samotného návrhu a metodického způsobu, je, že každá konverzace může být izolovaná a lépe předávána napříč vývojovými týmy [27]. Z technického pohledu se lze dívat na toky taktéž jako na jednotlivé stavy, ve kterých se může konverzace nacházet. Konverzace může být právě v jednom stavu. Proto byl pro modelování konverzace použit **stavový diagram** pocházející z UML. Stavové diagramy popisující konverzaci, které jsou zobrazeny níže, však neobsahují koncové stavy, jelikož se vždy vrátí do výchozího stavu, kdy chatbot očekává příkazy uživatele.

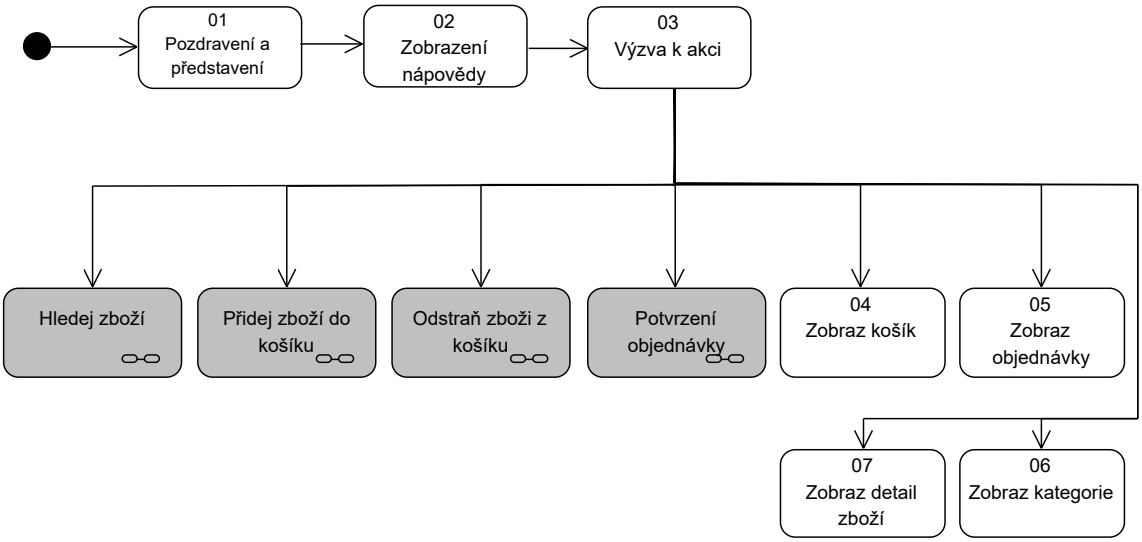
Při návrhu konverzačních toků je nutné smýšlet jako při návrhu kteréhokoliv jiného uživatelského rozhraní. Konverzační tok musí být intuitivní. Uživatel musí vědět, co se od

[1] REST – Representational State Transfer

něho očekává a jak se může v celém toku pohybovat. Často je vhodné opakovat hlášky vyzývající uživatele k akci a opakování, v jakém stavu konverzačního toku se uživatel právě nachází. V každém konverzačním rozhraní by se proto měly nacházet standardní možnosti typu zrušení současné akce, restartování konverzace nebo zobrazení nápovědy. Z důvodu přehlednosti diagramu toku však tyto akce nebudou do žádného uzlu zakresleny [27].

Hlavní konverzační tok

Na obrázku 5.1 je zobrazen hlavní konverzační tok konverzace. Bílé uzly představují konkrétní zprávu v konverzaci, zatímco šedé uzly reprezentují toky, které se dále dělí na další toky. V následující části budou představeny jednotlivé komponenty hlavního konverzačního toku.



Obrázek 5.1: Hlavní konverzační tok (zdroj vlastní).

1. **Pozdrav a představení**

   Pozdrav a představení je první interakce, při níž dochází chatbot a uživatel do styku, proto je velmi důležitá. Této úvodní části konverzace se říká taktéž *onboarding* [27]. Cílem je především definovat účel chatbota tak, aby bylo zcela zřejmé, s čím může uživateli pomoci.

2. **Zobrazení nápovědy**

   Další důležitou části začátku konverzace je představení, jak může uživatel používat chatbota. Toto představení je velmi důležité především na začátku konverzace, jelikož uživatel věnuje chatbotovi ještě plnou pozornost. Zároveň není vhodné uživatele zahrnout komplexním manuálem, ale spíše základními příkazy a případy použití [27]. Lze poznamenat, že pokud chatbot umožňuje ukládat stav o uživateli, je možné tyto úvodní části konverzace přeskočit v případě, kdy uživatel navštívil chatbota po druhé.

3. **Výzva k akci**

   V této části konverzace chatbot vybízí uživatele k akci. Typicky se jedná o větu typu: *"Jak Vám mohu pomoci?"*.

4. **Zobraz košík**

   V téhle části konverzace je zobrazen aktuální nákupní košík uživatele, který se skládá z jednotlivých položek a finální ceny. Chatbot po zobrazení košíku uživateli připomene, jak může odstranit zboží z košíku a jak potvrdit objednávku.

5. **Zobraz objednávky**

   Jedná se o možnost, kdy chatbot zobrazí aktivní objednávky, které uživatel vytvořil, včetně finální ceny.

6. **Zobraz kategorie**

   Jedná se o možnost, kdy chatbot zobrazí aktivní objednávky, které uživatel vytvořil, včetně finální ceny.

7. **Zobraz detail zboží**

   V případě, kdy uživatel zadá, že ho zajímá detail konkrétního zboží, chatbot zkontroluje, zdali zboží existuje. V případě existence zobrazí detail produktu, v opačném případě upozorní uživatele o neexistenci zboží v systému.

Hledej zboží

Obrázek 5.2 reprezentuje konverzační tok hledání zboží. Je nutné uvést, že se jedná pouze o vysokoúrovňový pohled, který popisuje základní myšlenku fungování. Tato myšlenka spočívá v tom, že chatbot nalezne určitou množinu produktů, které korespondují se vstupem zákazníka. Pakliže bude tato množina příliš veliká, nabídne chatbot možnost tuto množinu produktů redukovat. Redukce bude probíhat tak, že se bude chatbot ptát na jednotlivé společné atributy produktů a na základě odpovědi bude tyto produkty filtrovat. Problematika vyhledávání a získávání zpětné vazby je detailně rozebrána v kapitole 6.2.

1. **Hledej zboží**

   Samotný tok začíná tím, že uživatel zadá zboží, které ho zajímá. Chatbot následně prohledává zboží podle několika metrik.

2. **Našel jsem**

   V úspěšném scénáři chatbot našel konkrétní zboží, které uživatel hledal. V případě, že se nejedná pouze o jeden kus zboží ale vícero, chatbot zobrazí uživateli kolik potenciálních kusů zboží našel. Uživatel se poté může rozhodnout, zdali si nechá zobrazit všechny nalezené kusy zboží, nebo nechá chatbota, aby zjistil jeho preference a vyfiltroval pouze relevantní zboží.

3. **Zobraz vše**

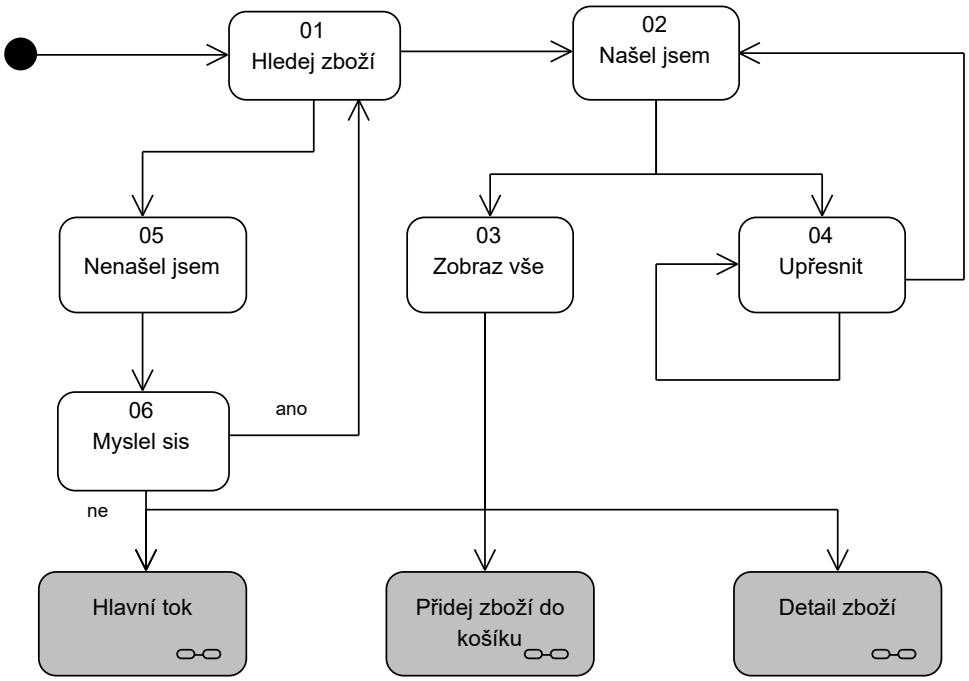
   Když uživatel vybere možnost zobraz vše, chatbot zobrazí uživateli všechno nalezené zboží. Počet nalezeného zboží ovlivňuje formát vykreslení. Do 10 položek je zobrazena karta zboží s obrázkem a základními informacemi, nad 10 položek pouze listový výčet všech produktů. Tyto produkty poté uživatel může přidat do svého nákupního košíku nebo si nechat zobrazit jejich detail. Konverzace se poté přesměruje zpět do hlavního toku.

4. **Upřesnit**

   V této části konverzace se chatbot doptává uživatele (získává zpětnou vazbu) na některé vlastnosti, které mu pomůžou redukovat počet nalezených produktů. Samotné doptávání končí, když se zúží množina potencionálních předmětů pod určitou hodnotu nebo v případě, kdy uživatel zadá, že chce zobrazit všechny filtrované předměty.

5. **Nenašel jsem**

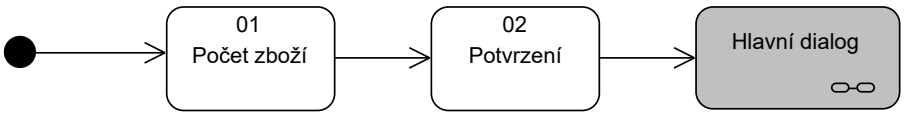
   V případě, kdy chatbot nenašel žádný produkt, nabídne uživateli nejpravděpodobnější slovo z celé databáze produktů, které asi myslel, aby zamezil překlepům uživatele. Pakliže uživatel souhlasí s novým slovem, chatbot pokračuje na krok 02, v opačném případě se konverzace přesune zpět na hlavní tok.



Obrázek 5.2: Konverzační tok hledej zboží (zdroj vlastní).

Přidej zboží do košíku

Na obrázku 5.3 je zobrazen konverzační tok reprezentující přidání zboží do košíku, do něhož se uživatel dostane z hlavní konverzačního toku. Dále následuje popis jednotlivých kroků.



Obrázek 5.3: Konverzační tok přidáni zboží do košíku (zdroj vlastní).

1. **Počet zboží**

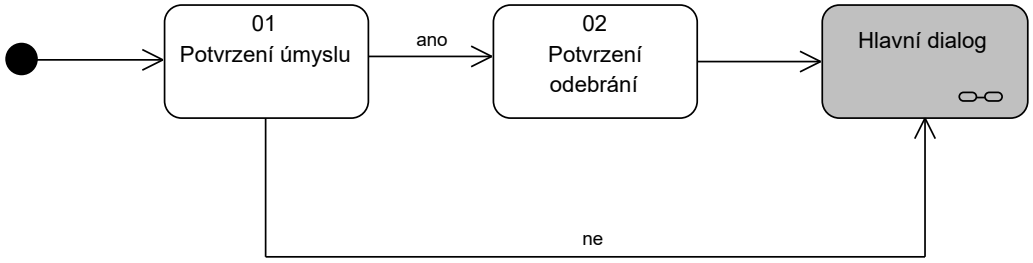
   V případě, kdy uživatel vybere zboží, které se má přidat do košíku, chatbot se ho zeptá, kolik kusů daného zboží si chce objednat.

2. **Potvrzení**

   V případě, kdy chatbot úspěšné ověří existenci zboží a jeho počet, zobrazí uživateli potvrzení, že bylo zboží přidáno do nákupního košíků a připomene, jak si může nákupní košík zobrazit. Konverzace poté pokračuje do hlavního dialogu.

Odstraň zboží z objednávky

Obrázek 5.4 reprezentuje konverzační tok odebrání produktu z nákupního košíků.



Obrázek 5.4: Konverzační tok odstranění zboží do košíku (zdroj vlastní).

1. **Potvrzení úmyslu**

   Když uživatel zadá zboží, které má být odstraněno z nákupního košíku, chatbot se ujistí a zeptá se, zdali si je uživatel jistý.

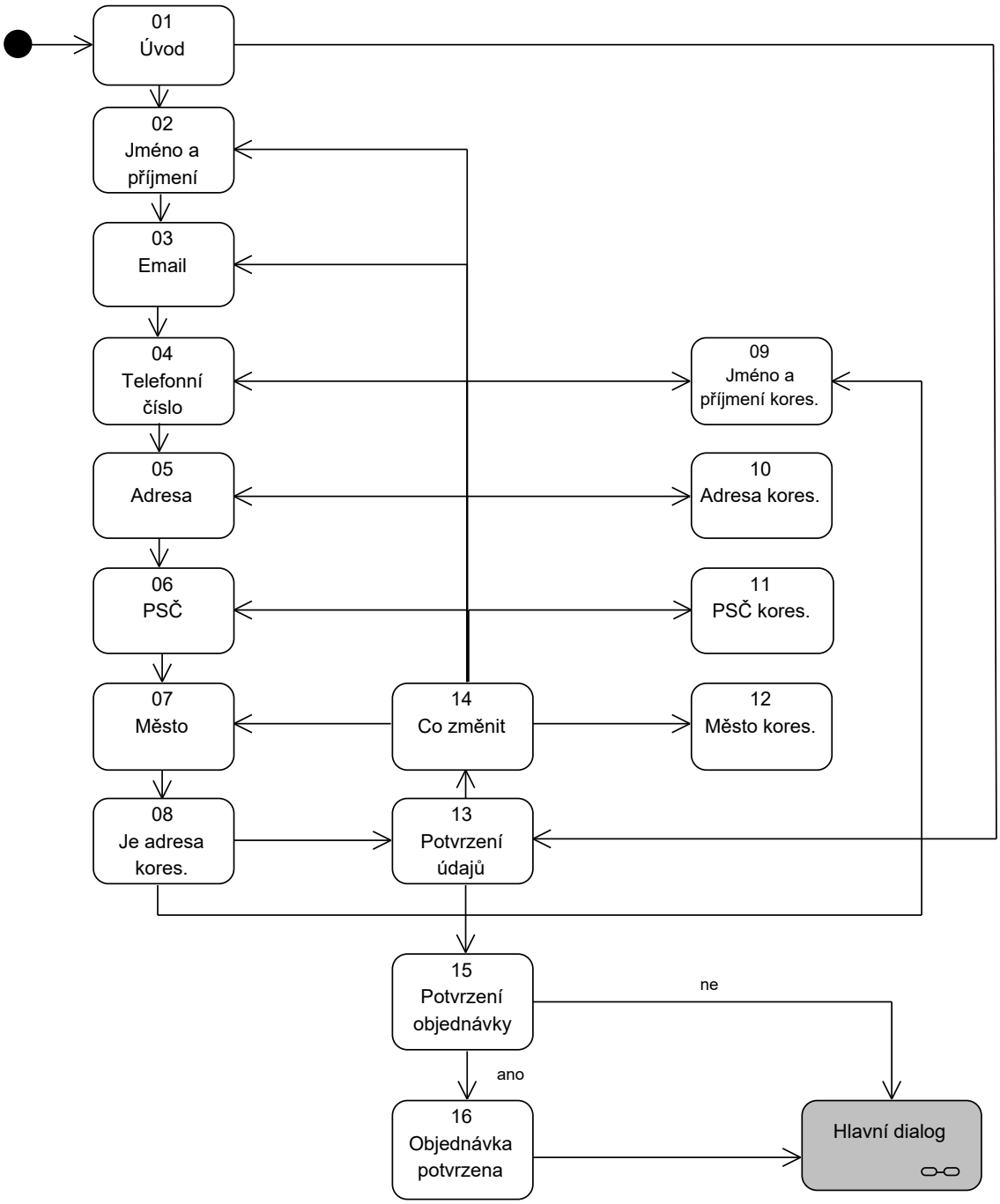
2. **Potvrzení odebrání**

   Chatbot potvrdí odebrání zboží z nákupního košíku. Konverzace dále pokračuje do hlavního dialogu.

Potvrzení objednávky

Na obrázku 5.5 lze vidět konverzační tok potvrzení objednávky. V rámci tohoto toku je nutné nejprve vyhodnotit, zdali má chatbot k dispozici informace o uživatelovi, které jsou nezbytné pro vytvoření objednávky. Pakliže tyto informace nemá (tj. nejsou uloženy v databázi), je nutné je zjistit, aby mohla proběhnout validní objednávka. Mezi tyto informace se řadí jméno a příjmení, email, telefonní číslo, adresa, PSČ a město. Další důležitou informací je, zdali se zadaná adresa trvalého bydliště (nebo sídla firmy) shoduje s adresou korespondenční. V případě, kdy uživatel odpoví negativně, je přesměrován do části konverzace, v níž se chatbot doptá i na jeho korespondenční adresu – viz krok 08 na obrázku 5.5. Jakmile chatbot získá všechny požadované informace (krok 13), ověří správnost uživatelských dat. V případě, že uživatel nalezne chybu ve svých datech, chatbot mu umožní vybrat v kroku 14 atribut, který zadal nesprávně a ten poté opravit. Tento přístup je aplikován proto, že v konverzačním rozhraní je velmi jednoduché udělat překlep a nebylo by uživatelsky přívětivé začít s procesem získávání dat od znova.

Jakmile uživatel potvrdí svá zadaná data, mohou být tato data uložena do databáze, aby je uživatel nemusel vyplňovat při nové objednávce znovu. Následně se chatbot dotáže na správnost celé objednávky, kdy zobrazí aktivní nákupní košík, finální cenu a informace o uživateli nutné k zaslání objednávky. V případě potvrzení je objednávka zaslaná na zpracování a uživateli se zobrazí informace o jejím úspěšném založení.



Obrázek 5.5: Konverzační tok potvrzení objednávky (zdroj vlastní).

5.3 Použité technologie

Kapitola 2 ukázala dostupné technologie pro tvorbu chatbotů. Cílem této podkapitoly je představit konkrétní technologie, které budou použity pro implementaci a vysvětlit důvody k jejich výběru.

V kapitole 2 byly kategorizovány dostupné technologie do 3 skupin podle požadovaných znalostí vývojáře: vysokoúrovňový přístup, přístup střední úrovně a nízkoúrovňový. Pro účel diplomové práce se nejvíce hodí přístup střední úrovně. Oproti vysokoúrovňovému nabízí podporu plného přizpůsobení dle požadavků. Vysokoúrovňové přístupy taktéž limitují ve výběru platformy, na níž bude chatbot nasazen.

Pro účel diplomové práce taktéž není příliš vhodný nejnižší přístup, který je charakteristický tím, že nevyužívá žádné nástroje na tvorbu chatbotů. Tímto způsobem stráví vývojář spoustu času vývojem nástrojů, které jsou pro něho nachystány.

5.3.1 Aplikačního rámec pro tvorbu konverzačního manažera

Při výběru technologie bylo bráno v potaz doporučení spolupracující firmy. Allium s.r.o. preferovala technologie od společnosti Microsoft. Jedním z důvodů této preference bylo, že firma má postavená všechna řešení nad technologiemi společnosti Microsoft a ráda by udržela celkovou kontinuitu vývoje.

Pro výběr konverzačního manažera byl zvolen aplikační rámec **Microsoft Bot Framework**[2], který byl již částečně představen v kapitole 3.2.3. Výhodou oproti jiným aplikačním rámcům je, že vývojář jednotlivé konverzační toky programuje sám a má tedy větší možnost přizpůsobit konverzaci požadavkům. Další výhodou tohoto rámce je, že je nezávislý na použité službě pro porozumění přirozeného jazyka. Microsoft Bot Framework nabízí možnost výběru programovacího jazyka pro vývoj. Konkrétně ve stabilní verzi nabízí možnost volby programovacího jazyka C# a Javascript. V rámci diplomové práce byl zvolen programovací jazyk C#. Jedním z rozhodovacích faktorů byla rozsáhlejší dokumentace a lepší dostupnost studijních materiálů.

Pro lokální vývoj chatbota byl dále používán nástroj Bot Framework Emulator. Tento nástroj umožňuje snadné napojení na lokálně spuštěného chatbota. Díky tomuto nástroji je možné chatbota ladit.

5.3.2 Porozumění přirozeného jazyka

V předchozích kapitolách bylo porozumění přirozeného jazyka již částečně představeno včetně několika dostupných služeb, viz kapitola 3.2.2.

Výběr služby

V rámci diplomové práce byla zvažována především služba LUIS od společnosti Microsoft a Rasa NLU od společnosti Rasa.

Mezi **výhody** Rasa NLU v porovnání se službou LUIS patří:

- Jedná o open-source nástroj a lze tedy analyzovat jaké algoritmy a přístupy služba používá.
- Jednoduchost samotného řešení, kdy lze všechny záměry a entity definovat v jednom souboru.
- Možnost konfigurace zřetězeného zpracování. Lze měnit, jaká knihovna podporující strojové učení se použije. Na výběr je knihovna Tenserflow a spaCy.

[2]Github repozitář projektu lze nalézt zde: https://github.com/Microsoft/BotBuilder

- Lze přidat podporu českého jazyka. Ve výchozím stavu sice Rasa NLU češtinu neobsahuje, ale dá se doinstalovat.
- Při provozu na vlastním serveru nejsou dodatečné náklady na provoz.

Mezi **nevýhody** poté patří:

- LUIS je v porovnání s Rasa NLU mnohem komplexnější nástroj. Rasa NLU například neumožňuje online testování, zabezpečení, testování v dávkách a logování.
- Větší pracnost při nasazení na vlastním serveru, jelikož je nástroj vyvíjen v programovacím jazyce Python. Není nutné řešit vlastní provoz, protože LUIS funguje jako služba. V případě požadavku, aby Rasa NLU fungovala v cloudové výpočetní platformě Microsoft Azure, se nabízí možnost nasadit aplikaci jako Docker kontejner a zabezpečení řešit přímo v Microsoft Azure.
- Rasa NLU oproti LUIS nenabízí předdefinované záměry (pozdravy, loučení, restartování konverzace) a předdefinované entity (jako třeba automatické detekce lokací, jmen, dat).

Po diskuzi se zadavatelem práce byl nakonec vybrán **LUIS** jakožto služba poskytující porozumění přirozeného jazyka pro vyvíjeného chatbota. Mezi hlavní důvody byla zejména jednoduší práce s tímto nástrojem a taktéž fakt, že Microsoft Bot Framework je více připraven na práci s LUIS. Nicméně, v budoucnosti lze stále v případě potřeby LUIS nahradit za Rasa NLU.

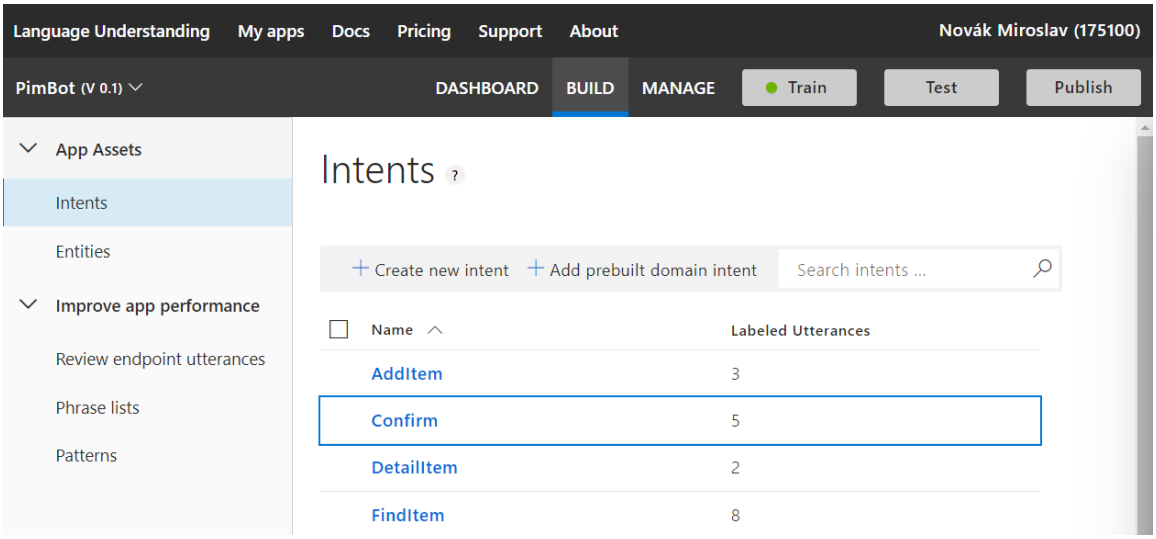
Práce se službou LUIS

Aplikace, které chtějí využívat LUIS, s touto službou komunikují pomocí definovaných HTTP požadavků s tím, že tato služba dostává na vstupu větu v přirozeném jazyce, výstupem je poté strukturovaný popis ve formátu JSON. LUIS obsahuje spoustu předdefinovaných záměrů a entit, které jsou doporučeny používat [15].

LUIS se ovládá primárně přes webové uživatelské rozhraní nazvané **Luis Portal**, který je znázorněn na obrázku 5.6. Důležitá poznámka je to, že všechna data lze z Luis Portal exportovat a v případě potřeby importovat do nové aplikace. Proces vytvoření nového záměru je následující:

1. Vývojář na hlavní stránce webové aplikace zvolí možnost "Vytvoř nový záměr" a tento záměr pojmenuje.
2. Vývojář k danému záměru definuje několik typických výroků, které tento záměr popisují. Těmto větám se říká také *utterances*. Příkladem je záměr "Otevírací doba" a výrok popisující tento záměr: "Kdy zítra otevírá obchod?".
3. Vývojář natrénuje model a poté ho publikuje.

Nově lze LUIS ovládat i skrze sadu nástrojů příkazové řádky. Jedním z těchto nástrojů je Luis CLI, který umožňuje spravovat celý model.



Obrázek 5.6: Luis Portal (snímek obrazovky).

5.3.3 QnA Maker

QnA Maker je další z rodiny nástrojů Azure Cognitive Service, obdobně jako LUIS. QnA Maker je webová služba, která umožňuje vytvořit znalostní bázi ve formátu otázka-odpověď. Služba QnA Maker odpovídá na otázky uživatelů (vstupující v přirozeném jazyce) s tím, že vyhledává nejlepší shodu ve znalostní bázi. Příkladem použití takových znalostních bází může být FAQ[3], produktové příručky, podpůrné dokumenty či tvorba chatbotů bez znalosti programování. Obdobně jako LUIS, má QnA Maker primární webové uživatelské rozhraní [16].

Hlavní rozdíl mezi službou QnA Maker a LUIS je, že QnA Maker na základě vstupu hledá nejlepší shodu a vrátí odpověď k dané shodě, zatímco LUIS extrahuje záměr a entity z uživatelovy věty. Při tvorbě chatbotů se často používá kombinace obou technologií. QnA Maker se poté používá pro běžné otázky, na které je generována vždy stejná odpověď. Například otázka: *"Jaká je otevírací doba obchodu?"*. LUIS se poté používá pro otázky, jejichž odpověď je již specifická pro konkrétního zákazníka. Typická otázka může být: *"Jak to vypadá s mojí objednávkou?"*.

V případě diplomové práce však QnA Maker řeší problematiku **Small talks**, což lze do češtiny volně přeložit jako běžná konverzace. Jedná se konverzaci, která nemá žádný význam a neřeší žádný případ použití chatbota. Nicméně, podpora těchto rozhovorů dokáže razantně zlepšit celkový dojem z používání chatbota a taktéž dokáže definovat jeho osobnost. Uživatelé chatbota rádi zkoušejí a pokládají mu nejrůznější otázky, přičemž jsou zvědaví, co jim chatbot odpoví. Typickou běžnou konverzací jsou věty: *"ahoj"*, *"jak se máš"*, *"nemám tě rád"* a podobně [27].

Řešení bylo navrženo tak, že pomocí služby LUIS bude natrénovaná datová sada otázek. V případě, že chatbot pošle požadavek službě LUIS a ta vrátí záměr SmallTalk, chatbot se dotazuje služby QnA Maker, kde pošle požadavek se stejnou větou uživatele, přičemž QnA Maker vrátí odpověď, kterou chatbot uživateli zobrazí. Datová sada byla použita z beta verze projektu BotBuilder-PersonalityChat[4]. Na výběr jsou 3 datové sady, které se dělí podle

[3] Frequently Asked Question - často kladené otázky

[4] Github repozitář projektu: https://github.com/Microsoft/BotBuilder-PersonalityChat

osobnosti chatbota – profesionální, přátelský a zábavný. Správná volba osobnosti chatbota je pro úspěšnou marketingovou komunikaci nezbytná [27]. Pro účel chatbota vytvářeného v rámci této diplomové práce se nejvíce hodí osobnost profesionální, jelikož chatbot bude komunikovat především s koncovými zákazníky. V rámci implementace byla poté datová sada upravena tak, aby mohla být použita pro LUIS a QnA Maker.

5.3.4 BotFramework-WebChat

Pomocí aplikačního rámce Microsoft Bot Framework (viz 5.3.1) lze vytvořit serverovou část chatbota, kterého lze poté napojit na některé kanály, které tento framework podporuje (například Facebook Messanger, Skype a podobně). Ve spolupráci s firmou Allium s.r.o. byl domluven jako cílový kanál webová aplikace.

Pro implementaci klientské aplikace, která bude komunikovat s chatbotem, byl použit BotFramework-WebChat[5] klient. Existují dvě varianty, jak tohoto klienta použít. Jednou z možností je vložit do běžné HTML stránky skript v programovacím jazyce Javascript, který zajistí rendrování webového chatu včetně všech nutných komponent jako jsou dialogová okna či tlačítka. Druhou možností je poté použít komponentu naprogramovanou v knihovně React.js. Pro účely diplomové práce bude použita první varianta, tedy ta bez nutnosti knihovny React.js. Jednou z dobrých vlastností klienta BotFramework-WebChat je možnost plně upravit vzhled webového chatu pomocí kaskádových stylů.

Před samotným použitím tohoto klienta je nutný takzvaně direct-line token, který lze získat v Microsoft Azure Portal v nastavení nasazeného chatbota. V kódu 1 lze vidět ukázku použití.

```
<!DOCTYPE html>
<html>
  <body>
    <div id="webchat" role="main"></div>
    <script src="https://cdn.botframework.com/
        botframework-webchat/latest/webchat.js">
    </script>
    <script>
      window.WebChat.renderWebChat({
        directLine: window.WebChat.createDirectLine({
            token: 'YOUR_DIRECT_LINE_TOKEN'
        }),
        userID: 'YOUR_USER_ID'
      }, document.getElementById('webchat'));
    </script>
  </body>
</html>
```

Kód 1: Ukázka rendrováni webového chatu pomocí knihovny BotFramework-WebChat (převzato [10]).

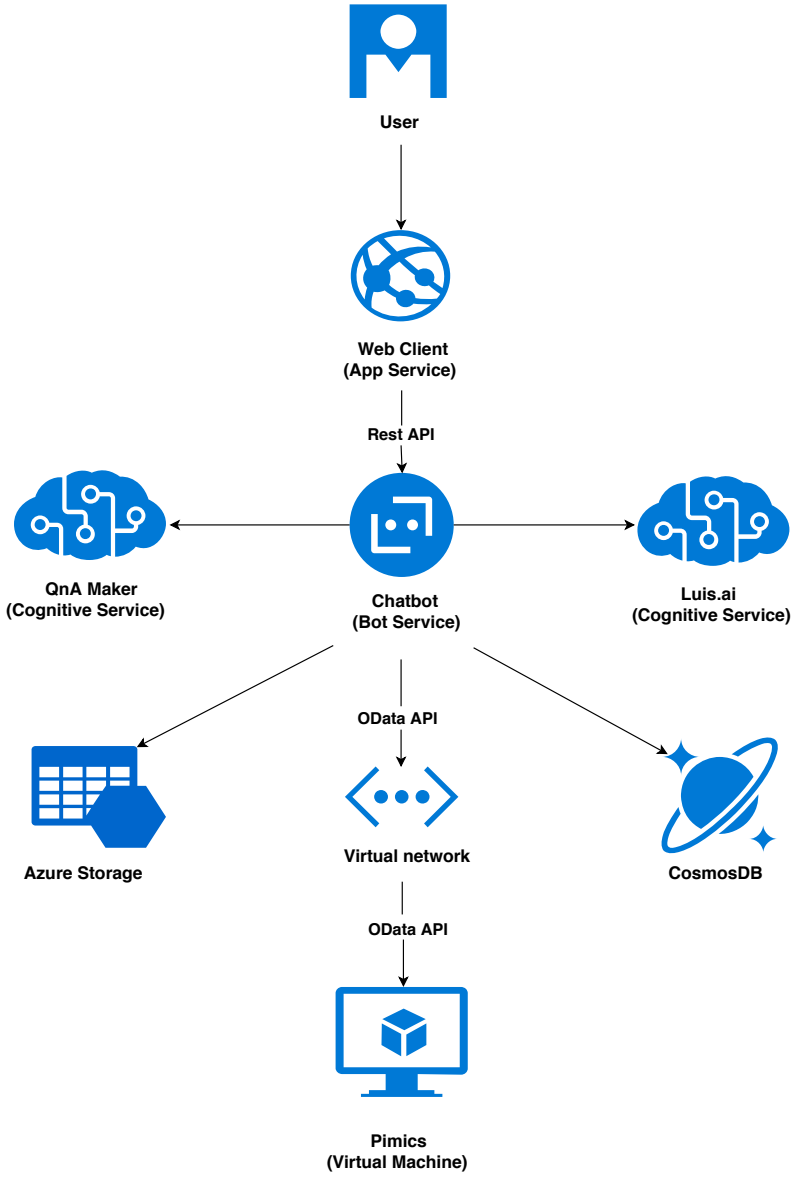
[5]Github repozitář projektu: https://github.com/Microsoft/BotFramework-WebChat

5.4 Návrh záměrů a entit pomocí služby LUIS

Záměry a entity byly vytvořeny pomocí aplikace Luis Portal a poté exportovány z důvodů jiného budoucího použití a persistence. V příloze C jsou představeny tyto navržené záměry včetně popisu a příkladu výroku.

5.5 Výsledná architektura řešení

Na obrázku 5.7 je zobrazena výsledná architektura řešení chatbota. Všechny komponenty tvořící řešení jsou nasazeny na platformě Microsoft Azure, což je cloudová platforma umožnující tvorbu, nasazení a správu služeb z nabízeného portfolia. Jednotlivé služby jsou poté spuštěny v data centrech společnosti Microsoft.



Obrázek 5.7: Výsledná architektura řešení (zdroj vlastní).

5.5.1 Stručný popis komponent

Většina služeb byla již v předchozích částech návrhu podrobně popsána, proto bude mít následující popis především účel souhrnu.

- **Webový klient (Web Client)**

  Jedná se o jedinou komponentu, se kterou uživatel přichází do přímého styku. V zásadě jde běžnou webovou aplikaci, s níž uživatel komunikuje pomocí HTTP protokolu. Tato komponenta dále komunikuje s chatbotem pomocí REST API. Více se lze dočíst v kapitole 5.3.4. Z hlediska platformy Microsoft Azure se jedná o službu **App Service**, což je služba umožnující vytvářet a hostovat webové aplikace, mobilní serverové systémy a rozhraní REST API s tím, že koncový programovací jazyk si volí vývojář. Výhoda tohoto řešení je mimo jiné automatické škálování a vysoká dostupnost služby [5].

- **Chatbot**

  Chatbot je služba vytvořená pomocí Microsoft Bot Framework, viz 5.3.1, která komunikuje s ostatními komponentami a tím tvoří centrální uzel architektury. Z hlediska Microsoft Azure se jedná o službu Azure Bot Service. Tato služba umožňuje jednoduché nasazení chatbota, konfiguraci a nastavení výstupních kanálů [3].

- **LUIS**

  LUIS je služba na extrahování záměrů a entit z věty, viz kapitola 5.3.2.

- **QnA Maker**

  Účel služby QnA Maker v rámci diplomové práce je poskytnout přístupnou znalostní bázi pro řešení problémů tzn. běžných konverzací, viz 5.3.3.

- **CosmosDB**

  CosmosDB[6] je globální distribuovaná dokumentová (jinými slovy taktéž NoSQL) databáze. Pro účel diplomové práce slouží tato databáze pro ukládání uživatelských dat a dokončených objednávek. Jedná se o dočasné řešení, v budoucnu by se měla uživatelská data a objednávky ukládat přímo do systému Microsoft Dynamics NAV a systému Pimics. Nicméně, tyto systémy v současnou chvíli nejsou zcela připraveny.

- **Azure Storage**

  Azure Storage[7] je cloudové úložiště určené k zápisu a čtení masivního množství dat. V rámci diplomové práce slouží k logování (též žurnálování nebo ukládání) všech konverzací mezi uživatelem a chatbotem z důvodů zpětné analýzy chování chatbota a pro jeho případné vylepšení.

- **Pimics**

  Jedná se o virtuální stroj, v němž je nainstalován systém Microsoft Dynamics NAV s modulem Pimics. Microsoft Dynymics NAV zároveň publikuje několik OData služeb, s nimiž může chatbot komunikovat. Více informací o propojení je uvedeno 5.1.

[6] Domovská stránka služby CosmosDB – https://azure.microsoft.com/services/cosmos-db
[7] Domovská stránka služby Storege – https://azure.microsoft.com/services/storage

Kapitola 6

Implementace chatbota včetně popisu řešení některých problémů

Tato kapitola popisuje implementaci chatbota a některé problémy, které se v průběhu implementace objevily. První část kapitoly (6.1) se věnuje implementaci propojení chatbota s ERP a PIM systémem pomocí OData webových služeb. Druhá část (6.2) popisuje implementaci vyhledávacího modulu, včetně základního popisu teorie rozhodovacích stromů, která byla pro implementaci použita. Další část (6.3) představuje řešení překlepů, kterých se uživatelé dopouštějí při psaní vyhledávaných produktů. Čtvrtá část (6.4) představuje strukturální a behaviorální model aplikace chatbota za účelem popsání způsobu řízení konverzace. Poslední část (6.5) poté uvádí, jak byl vyřešen požadavek na autentizaci uživatele.

6.1 OData webové služby

Jak bylo představeno v dřívějších kapitolách, chatbot komunikuje s ERP systémem Microsoft Dynamics NAV a modulem Pimics pomocí RESTful webových služeb, které využívají protokol OData.

6.1.1 Publikování webových služeb

Aby bylo možné využívat webové služby, je nutné je nejdříve v systému Microsoft Dynamics NAV publikovat. Samotné publikování se poté skládá z několika kroků:

V prvním kroku je nutné povolit dotazování na OData webové služby, které je ve výchozím stavu zakázané. Využívá se k tomu program Microsoft Dynamics NAV Server, který slouží na konfiguraci tohoto ERP systému. Po zvolení dané lokální instance je nutné otevřít nastavení OData Services a povolit OData služby verze 4.

Samotné publikování služby probíhá již v programu Microsoft Dynamics NAV, konkrétně v nastavení: *Departments/Administration/IT Administration/Services/Web Services*. Pro publikování konkrétní služby je nezbytné znát **Object ID** daného objektu. Všechny objekty v systému mají své unikátní identifikační číslo v rozsahu 1 – 999 999 999. V rámci diplomové práce se však pracuje pouze s objekty typu *Page*, což jsou objekty umožnující interakci s uživatelským rozhraním – reprezentují daný pohled v systému NAV [6]. Nalezení konkrétního identifikačního čísla poté spočívá v tom, že se nalezne daná informace v uživatelském rozhraní a poté si vývojář zobrazí informace o dané stránce, v níž bude uvedeno i identifikační číslo daného objektu.

Při publikování webové služby je nutné taktéž zadat jméno této služby. Toto jméno je velmi důležité, protože slouží jako identifikátor služby v URL adrese. Je proto nutné zachovat jistou konvenci při pojmenování publikovaných služeb. Služba se poté nachází na URL, které je tvořené podle následujícího schématu:

http://host:port/NAV/ODataV4/CompanyId/WebServiceName

Příkladem je vygenerovaná lokální URL adresa:

http://pimics-chatbot:7048/NAV/ODataV4/
Company('CRONUS%20International%20Ltd.')/ItemsPIM

V případě, kdy je nutné posílat požadavky mimo síť virtuálního stroje (tj. z okolního světa), v němž je spuštěn Microsoft Dynamics NAV, je potřeba nahradit doménové jméno virtuálního stroje za plně specifikované doménové jméno počítače (angl. FQDN), na němž je instance spuštěna. V případě diplomové práce je tedy koncová URL adresa webové služby následující:

http://pimicschatbot.westeurope.cloudapp.azure.com:7048/NAV/ODataV4/
Company('CRONUS%20International%20Ltd.')/ItemsPIM

6.1.2 Popis OData webových služeb

Pro účely dokumentace jsou v tabulce 6.1 zobrazeny publikované služby včetně jejich identifikačního čísla a jména.

Object Type	**Object ID**	**Service Name**
Page	4006640	FeaturesPIM
Page	4006532	ItemGroupsPIM
Page	4006545	ItemsPIM
Page	4043478	KeywordsPIM
Page	4006595	PicturesPIM
Page	4006550	ProductGroupsPIM

Tabulka 6.1: OData webové služby publikované systémem Microsoft Dynamics NAV.

- **FeaturesPIM**

  Služba, která vrací všechny vlastnosti všech produktů. Důležitý je zejména atribut *Code*, který je identifikátorem produktu, a atribut *Description*, který popisuje o jakou vlastnost se jedná. Další atribut *Value* popisuje hodnotu vlastnosti.

- **ItemGroupsPIM**

  Tato služba vrací všechny skupiny předmětů s tím, že významným atributem je *Code*, který definuje identifikaci kategorie a *Description*, který popisuje název kategorie.

- **ItemsPIM**

  ItemsPIM je služba, která vrací všechny koncové produkty uložené v systému. Atribut *No* je poté identifikátor předmětu.

- **KeywordsPIM**

  Jedná se o službu, která vrací všechna klíčová slova ke všem produktům. Významným atributem je *Code*, který definuje produkt, ke kterému se zboží váže, a *Keyword_ID* definující identifikátor klíčového slova. Atribut *Keyword* poté obsahuje hodnotu klíčového slova.

- **PicturesPIM**

  Tato služba vrací všechny obrázky, které se vážou k produktům. Důležitý je atribut *Number*, který definuje identifikátor obrázku a atribut *Content*, který obsahuje samotný obrázek ve formátu *Base64*. Obrázky nejsou nijak komprimované (tj. mimo rámec standardních formátů) a je nutné je na klientské aplikaci chatbota dynamicky zmenšovat, pokud překročí maximální velikost. Pro stručnost se text diplomové práce tímto problémem nezabývá.

- **ProductGroupsPIM**

  Jedná se o službu, která vrací všechny produktové kategorie. Atribut *Code* definuje identifikátor kategorie a *Description* poté jeho popis.

6.1.3 Klient webových služeb

Byly zvažovány dva způsoby implementace klienta OData webových služeb. Prvně byl na implementaci použit oficiální modul Visual Studia 2017 nazvaný **OData Connected Service**. Cílem tohoto modulu je vygenerovat klienta OData služeb v jazyce C#. Princip je takový, že uživatel předloží modulu URL služeb a modul vygeneruje požadovaný kód. Nicméně, tento modul měl spoustu nedostatků. Jedním z nedostatků bylo to, že modul nezvládal autentizaci webové služby. Druhým problémem byla samotná kvalita vygenerovaného kódu. Na podporu výše zmíněných 6 webových služeb tento modul vygeneroval více než 10 000 řádků zdrojového kódu, který navíc nebyl přeložitelný.

Nakonec byla využita knihovna **Simple.OData.Client**[1], která je odlehčenou variantou OData Connected Service. S využitím této knihovny byla vytvořena třída `ODataClient Singleton`, která poskytuje jiným třídám instanci třídy `ODataClient`. Třída je navržena podle návrhového vzoru Jedináček. Instanci třídy `ODataClient` poté využívají navázané třídy, které reprezentují danou službu. V kódu 2 lze vidět využití knihovny pro získání všech produktů z PIM serveru.

```
1 var client = ODataClientSingleton.Get();
2 var items = await client
3         .For(Constants.Company).Key(Constants.CompanyName)
4         .NavigateTo(Constants.ItemsServiceEndpointName)
5         .FindEntriesAsync();
```

Kód 2: Využití knihovny Simple.OData.Client pro posílání požadavků na PIM server.

[1]Odkaz na Github repozitář: https://github.com/simple-odata-client/Simple.OData.Client

6.2 Vyhledávací modul

Vyhledávací modul je část chatbota, která pomůže nalézt uživateli zboží, které si poté může vložit do nákupního košíku. Jedná se implementaci navrženého konverzačního toku *Hledej zboží*, viz 5.2. Modul se skládá ze dvou částí – vyhledání množiny předmětů na základě vstupu uživatele (6.2.2) a získávání zpětné vazby na konkrétní vlastnosti produktů za účelem redukce potencionálních předmětů (6.2.3).

6.2.1 Vysvětlení problému

Implementace modulu, který umožňuje vyhledání konkrétního produktu, byl nejsložitější problém řešený v rámci diplomové práce. Podstata problémů spočívá v tom, že PIM systémy mohou obsahovat až statisíce koncových produktů [21] a uživatel si přeje objednat právě jeden z nich. Nelze však počítat s tím, že uživatel ví, který konkrétní produkt si přeje – v PIM databázi jsou jako koncové produkty uloženy většinou konkrétní konfigurace daného produktu (například *Samsung Galaxy S10+ Dual SIM 128GB black*). Uživatel běžně ví, o jakou skupinu předmětů má zájem (tj. například mobilní telefony) a jaké má preference (tj. cena, velikost displeje apod.). Chatbot zde vstupuje v roli asistenta, který má pomoci s výběrem.

Dalším aspektem problému je, že není dopředu známá vstupní doména. Chatbot může být potenciálně připojen na různé PIM systémy. PIM systém jedné společnosti může obsahovat nábytek, v jiné třeba cyklistické potřeby. Proto není snadné dynamicky vytvořit model strojového učení, který by sloužil k optimalizaci a urychlení vyhledávání. Implementace se proto snažila o co možná největší míru abstrakce procesu tak, aby byl systém zcela nezávislý na vstupní doméně.

6.2.2 Vyhledání množiny produktů

Po analýze možností, jak lze vyhledat produkt na základě vstupu uživatele, lze vyvodit dva přístupy:

- **Přístup shora dolů**

  Tento přístup spočívá v tom, že chatbot nalezne co největší množinu potencionálních předmětů, kterou vyhledal podle vstupu uživatele. Tuto množinu poté dále redukuje na základě získávání zpětné vazby od uživatele.

- **Přístup zdola nahoru**

  V tomto přístupu chatbot vyhledá co nejmenší množinu předmětů, tedy tu, která nejvíce odpovídá předmětu vyhledávání. Pakliže by uživatel nenašel požadovaný předmět v dané vyhledané množině, může pomocí získávání zpětné vazby tuto množinu rozšířit.

V rámci diplomové práce byl zvolen přístup **shora dolů**, především z důvodů, aby se nestalo, že by se produkt, který si uživatel chce objednat, vůbec neobjevil ve výsledku vyhledávání. V přístupu shora dolů je větší pravděpodobnost, že se tam právě onen vyhledávaný produkt nachází.

V aplikačním kódu se nachází třída `ItemService`, která obsahuje metodu `GetAllItemsBy MatchAsync`, která očekává na vstupu uživatelův popis toho, co vyhledává. Tato metoda postupně vyhledá všechny produkty podle jména produktu (tj. Item description), všechny produkty podle shod se jménem kategorie a všechny produkty podle shod klíčových slov.

Všechny tyto 3 nalezené množiny produktů následně spojí pomocí matematické operace sjednocení množin.

6.2.3 Získávání zpětné vazby

Tato část aplikace má za cíl redukovat množinu nalezených předmětů pomocí doptávání se uživatele na konkretní vlastnosti, viz *features* ve zjednodušeném datovém modelu 4.3.2. Chatbot v současnosti podporuje 3 typy vlastností, na které se může zeptat. Typ vlastnosti je uveden i v PIM systému a lze ho získat pomocí webových služeb. Bohužel, typ vlastnosti je uveden často nesprávně. Mnohdy je uveden typ *Alphanumeric*, ale vlastnosti obsahují číselné hodnoty. Proto je nutné dynamicky zjišťovat typ vlastnosti. Pro tyto účely slouží metoda `SetAndCheckType()` třídy `FeatureToAsk`. Samotné typy vlastností podporované chatbotem jsou:

- **Číselný typ**

  Tento typ reprezentuje číselnou hodnotu vlastnosti. Chatbot prvně agreguje jednotlivé hodnoty vlastnosti (například váha) a spočítá jejich medián. Na výpočet mediánu slouží metoda `GetMedianValue()` třídy `FeatureToAsk`. Následně se chatbot uživatele zeptá na otázku typu *"Co si myslíte o váze"* a nabídne mu 2 možnosti odpovědi. Do hodnoty mediánu a nad velikost mediánu. Na základě zvolené hodnoty poté chatbot redukuje množinu nalezených hodnot a proces pokračuje.

- **Textový typ**

  Tento typ reprezentuje výčet. Chatbot v tomto případě dynamicky agreguje všechny hodnoty daného typu (vztahující se k dané množině) a zeptá se uživatele na konkrétní hodnotu výčtu. Příkladem je například barva, která může nabývat hodnot modrá, červená apod.

- **Logický typ**

  Posledním typem je logický typ obsahující hodnoty *true* a *false*. Chatbot agreguje hodnoty analogicky s textovým typem.

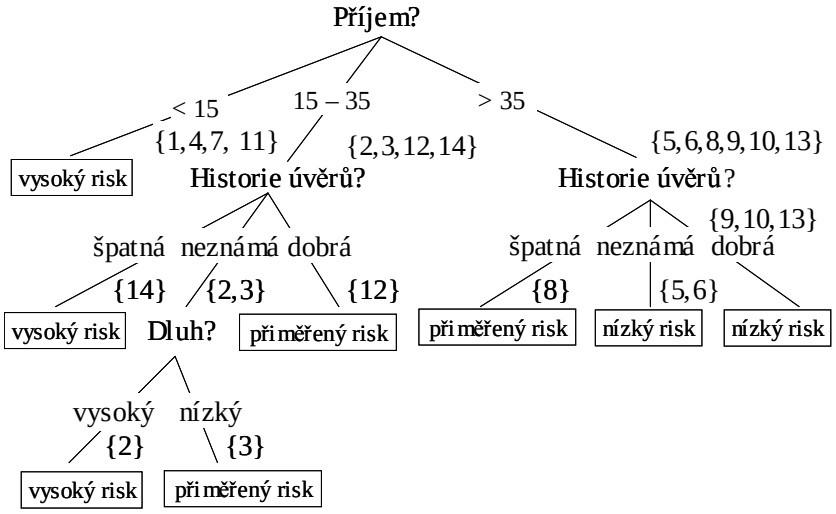
Výběr doptávaných vlastností

Dalším důležitým problémem je určení pořadí otázek specifikující vlastnosti. Jednotlivé produkty často obsahují velkého množství vlastností, běžně se jedná o více než 30 vlastností popisující daný produkt. Určení správného pořadí návaznosti otázek je zcela klíčové, jelikož cílem chatbota je v co **nejmenším** počtu kroků (tj. otázek) dovést uživatele k produktu, který si přeje objednat. Není uživatelsky přívětivé ptát se zákazníka na 30 otázek než chatbot dojde ke zdárnému výsledku. V průběhu implementace bylo zjištěno, že problematika volby vlastností, na které se zeptat, bez znalosti domény, není zcela triviální. Problém byl nakonec vyřešen pomocí **rozhodovacích stromů**.

Rozhodovací strom

K implementaci výběru pořadí dotazovaných vlastností byla využita teorie tvorby rozhodovacích stromů. Rozhodovací strom je algoritmus strojového učení, konkrétně jde o učení s učitelem, využívaný hlavně ke klasifikaci (ale umožňuje i regresi). Výhoda tohoto algoritmu, v porovnání s jinými klasifikátory (například Bayesovská klasifikace), je intuitivnost

řešení a jednoduchá možnost vizualizace stromu. Rozhodovací strom obsahuje dva typy uzlů – rozhodovací uzly, které kontrolují hodnoty vstupních vlastností a kořenové uzly, které přiřadí vstup k dané kategorii. Díky svým vlastnostem je často používaný v případech, kdy nelze určit priorita atributů vstupní datové sady [19]. Ukázkový rozhodovací strom lze vidět na obrázku 6.1. V uzlech tohoto stromu jsou zobrazeny otázky, mezi uzly možné odpovědi a v listech stromu výsledné kategorie.



Obrázek 6.1: Příklad jednoduchého rozhodovacího stromu (převzato [33]).

K výběru vhodných pořadí atributů se využívají zejména dvě kritéria – informační zisk (angl. information gain) a Gini index. V diplomové práce bylo využito právě kritérium informačního zisku, jehož výpočet vychází z výpočtu **entropie** (neboli míry neuspořádanosti). V prvním kroku vedoucí k výpočtu informačního zisku daného atributu je nutné vypočítat entropii celé datové sady (množiny příkladů), v případě diplomové práce všech vyhledaných produktů. Entropii datové sady lze vypočítat [33]

$$E(MP) = \sum_{i=1}^{n} -p(o_i)\log_2(p(o_i)) \tag{6.1}$$

kde $E(MP)$ je entropie datové sady (množiny příkladů), n je počet všech výsledných kategorií a $p(o_i)$ je pravděpodobnost výskytu odpovědi (kategorie).

Ve druhém kroku se poté spočítá entropie daného atributu [33]

$$E(A) = \sum_{i=1}^{m} \frac{MP_i}{MP} E(MP_i) \tag{6.2}$$

kde $E(A)$ je entropie atributu, m je počet hodnot atributu. Atribut A rozděluje množinu MP na m podmnožin, z nichž každá obsahuje MP_i případů.

Posledním krokem je poté výpočet informačního zisku [33]

$$zisk(A) = E(MP) - E(A) \tag{6.3}$$

kde $zisk(A)$ je informační zisk atributu A.

Samotný výpočet informačního zisku provádí metoda `ComputeInformationGain()` třídy `FeatureToAsk`. Tato metoda získává na vstupu seznam aktuálních produktů, na jejichž základě vypočítá pro danou vlastnost informační zisk. Výpočet vychází ze vzorců uvedených výše, implementovány jsou však některé **optimalizace** zjednodušující výpočet. Především není nutné počítat jednotlivé pravděpodobnosti výskytu odpovědi (kategorie). V případě diplomové práce je odpověď (tj. koncový uzel stromu) právě jedna, jelikož každý koncový produkt je v databázi právě jednou. Tím pádem je pravděpodobnost výskytu vždy stejná a rovná se převrácené hodnotě počtu všech vstupujících produktů.

Při výběru atributů (vlastností) pro konstrukci rozhodovacích stromu se v rámci diplomové práce nebere v potaz pouze informační zisk, ale taktéž i **předdefinovaná priorita**. V systému Pimics lze u jednotlivých vlastností vybrat jejich prioritu (důležitost) pomocí číselné reprezentace. Čím větší číslo v hodnotě *order*, tím má vlastnost větší prioritu. Typicky však správce systému označí několik vlastností (například délka, šířka a výška), které mají větší prioritu. Prvně se otázky řadí podle definované priority a pakliže mají stejnou prioritu, použije se vypočítané kritérium informačního zisku. Ve zdrojovém kódu chatbota poté existuje metoda `GetAllFeaturesToAsk`, která je definovaná v třídě `ItemService`, a která na vstupu očekává seznam produktů. Metoda na základě těchto předmětů vrátí seřazený seznam otázek, na které se má chatbot zeptat. V kódu 3 lze vidět ukázku z metody `GetAllFeaturesToAsk`, která provádí výše zmíněné operace.

```
1  filteredFeatureToAsk.ForEach(i => i.SetAndCheckType());
2  filteredFeatureToAsk.ForEach(i =>
3          i.ComputeInformationGain(items.ToList()));
4
5  var orderedFeaturesToAsk = filteredFeatureToAsk
6          .OrderByDescending(i => i.Order)
7          .ThenByDescending(i => i.InformationGain)
8          .ToList();
```

Kód 3: Ukázka kódu z metody `GetAllFeaturesToAsk`.

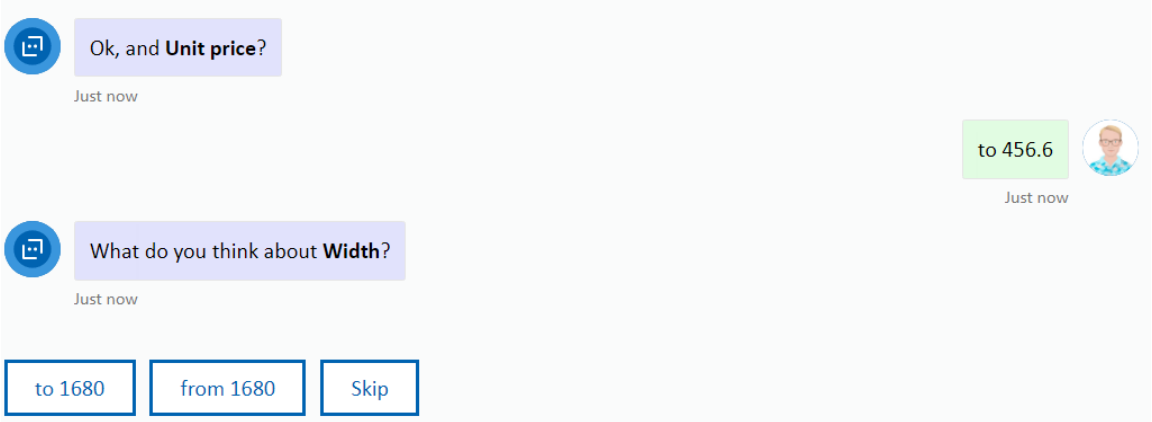
Samotný rozhodovací strom se poté tvoří pomocí známého iterativního algoritmu **ID3**. Zjednodušeně lze algoritmus představit v následujících 4 krocích [8]:

1. Výpočet entropie (informačního zisku) každého atributu a množiny dat S.
2. Rozdělení množiny S na podmnožiny s využitím atributu s nejmenší entropii (tj. největším informačním ziskem).
3. Vytvoření uzlu rozhodovacího stromu obsahující tento atribut.
4. Pokračování rekurzivně na zbývajících podmnožinách.

V práci byla provedena optimalizace tohoto algoritmu. Ta spočívá v tom, že není nutné vytvořit celý rozhodovací strom se všemi větvemi. Využívá se principu odloženého vyhodnocování (angl. lazy evaluation), kdy se vypočítá pouze kořen rozhodovacího stromu dané podmnožiny, kterou uživatel zvolil. V praxi to pak vypadá, že uživatel zvolí odpověď na danou otázku týkající se vlastnosti, poté zavolá se metoda `FilterItemsByFeature()` třídy

`ItemService`, která vrátí redukovanou množinu produktů. Nad touto množinou se poté znovu vypočítá kořen rozhodovacího stromu.

Na obrázku 6.2 je zobrazena ukázka výsledné aplikace. Chatbot získává zpětnou vazbu ohledně ceny a posléze šířky s tím, že nabídne uživateli některé hodnoty.



Obrázek 6.2: Ukázka fungování získávání zpětné vazby (snímek obrazovky).

6.3 Řešení překlepů

Další zajímavý problém bylo řešení překlepů či chyb, kterých se uživatel dopouští v komunikaci s chatbotem. Příkladem může být, pokud uživatel napíše *"please find me a char"*. Uživatel se dopustil překlepu a pravděpodobně měl na mysli *chair*, což v češtině odpovídá židli. Chatbot by měl projevovat určitou inteligenci a odpouštět uživatelům drobné překlepy, což obecně napomůže lepší uživatelské zkušenosti.

Obecně neexistuje jen jedna cesta, jak podobné problémy řešit. Zvažovány byly především dva algoritmy:

- **Algoritmus Levenshteinovi vzdálenosti**

  Mezi dvěma textovými řetězci $s1$ a $s2$ je Levenshteinova vzdálenost (někdy též označována jako editační vzdálenost) minimální počet editovacích operací, které jsou nutné k transformaci řetězce $s1$ na $s2$. Mezi editovací operace se řadí vložení znaku do řetězce, odstranění znaku z řetězce a nahrazení znaku v řetězci. Pro příklad, Levenshteinova vzdálenost mezi řetězci *pes* a *ves*, je právě 1, zatímco mezi řetězci *pes* a *boj* je 3 [22].

- **Algoritmus Metaphone**

  Algoritmus Metaphone je fonetický algoritmus, jehož autor je Lawrence Philips, který publikoval první verzi tohoto algoritmu v roce 1990. Vstupem do tohoto algoritmu je slovo v anglickém jazyce, výstupem poté jeho výslovnost ve formě klíče. Dvě slova, které se vyslovují stejně, mají na výstupu stejný klíč. Například, pokud je zavolána funkce Metaphone se vstupním řetězcem *programmer*, výstupním klíčem je *PRKRMR*. Stejný výstupní klíč funkce poskytne i nad vstupním řetězcem s vynechaným písmenem *programer* [9].

V rámci diplomové práce byl zvolen algoritmus **Levenshteinovi vzdálenosti**, jelikož poskytoval při zběžném testování lepší výsledky[2]. Nicméně, pro budoucí užití lze uvažovat o kombinaci obou přístupů. Celý proces řešení překlepů funguje tedy tak, že v případě, kdy vyhledávaný předmět nebyl nalezen, chatbot porovná pomocí Levenshteinovi vzdálenosti hledaný předmět s předměty uloženými v databázi. Chatbot poté zobrazí uživateli produkt, který má s vyhledávaným předmětem nejmenší Levenshteinovou vzdálenost s otázkou: *"Myslel jste tento předmět?"*

6.4 Struktura chatbota a řízení konverzace

Tato kapitola představuje strukturální (6.4.1) a behaviorální (6.4.2) pohled na aplikaci chatbota.

6.4.1 Struktura chatbota

Na obrázku 6.3 je zobrazen diagram balíčků popisující základní strukturu aplikace chatbota. Diagram balíčků zde tvoří vysokoúrovňový pohled na architekturu serverové části chatbota. Dále následuje krátký popis jednotlivých balíčků.

- **Bot**

  Balík *Bot* představuje hlavní balík, který zpracovává vstupní požadavky.

- **Dialogs**

  Tento balík v sobě obsahuje třídy, které řídí jednotlivé konverzační toky. Příkladem je třída `AddItemDialog`, která řídí dialog týkající se přidání nového předmětu do objednávky.

- **State**

  Balík *State* v sobě obsahuje třídy, které reprezentují stav neboli kontext chatbota. Stav chatbota je nutné ukládat z důvodu, aby uživateli zůstal zachován kontext, když přijde na stránku chatbota znovu. Příkladem je třída `CustomerState`, která reprezentuje informace o uživatelovi.

- **Services**

  Balík *Services* zapouzdřuje hlavní (business) logiku chatbota a manipulaci s daty. Tento balík například vyhledává jednotlivé produkty nebo vytváří otázky, na které se má chatbot uživatele zeptat.

- **Constants**

  Tento balík obsahuje třídy, které obsahují konstanty chatbota. Může se jednat o názvy záměrů služby LUIS, ale taktéž i celou textaci chatbota.

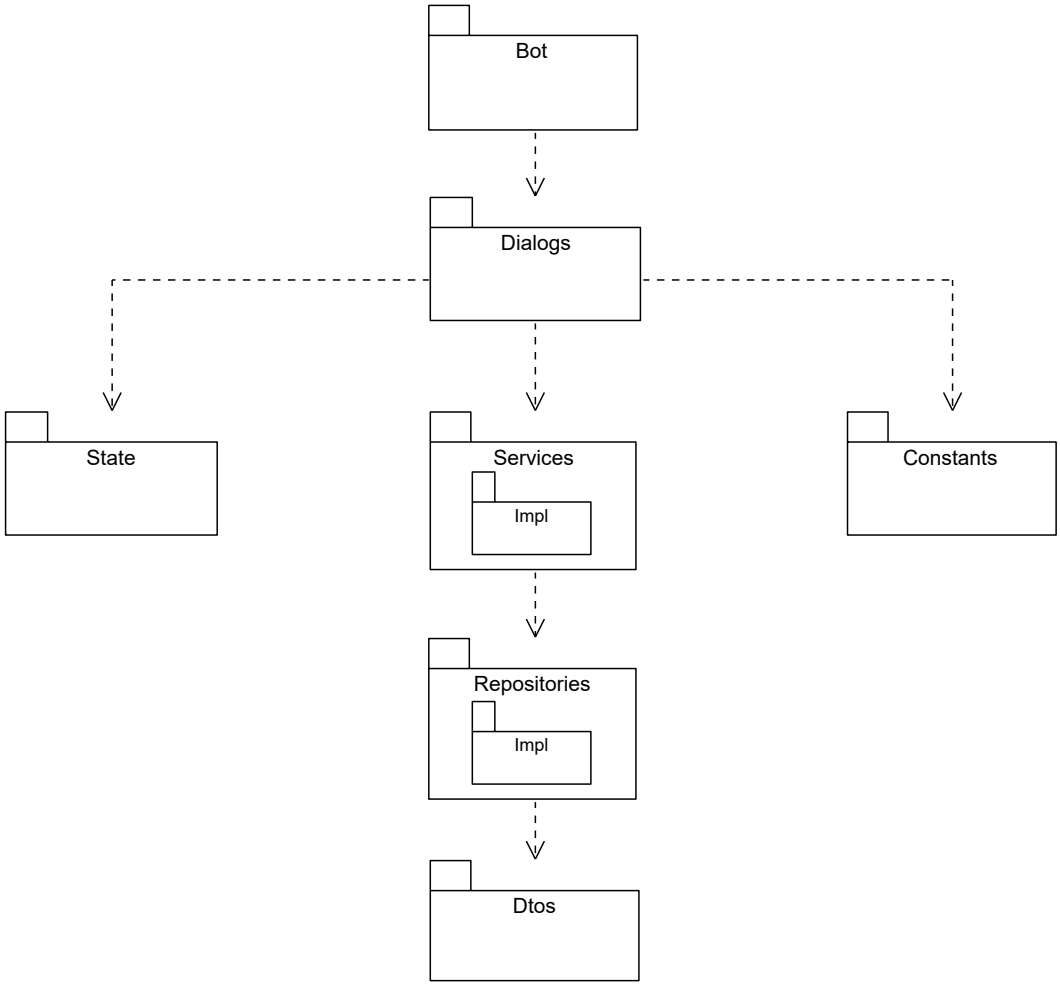
- **Repositories**

  Účel tříd v balíku *Repositories* je získávat data z webového serveru. Balík *Services* a *Repositories* jsou schválně separovány, aby nebyla hlavní aplikační logika závislá na způsobu dotazování dat. Díky této architektuře je možné v budoucnu místo webových služeb použít například databázový server.

[2]Samotná implementace v programovacím jazyce C# byla převzata z https://www.dotnetperls.com/levenshtein

- **Dtos**

  Balík *Dtos* (*data transfer objects*) v sobě obsahuje komunikační třídy, které jsou používány pro komunikaci mezi balíkem *Services* a *Repositories*.



Obrázek 6.3: Diagram balíčků serverové části chatbota (zdroj vlastní).

6.4.2 Řízení konverzace

Předchozí kapitola představila strukturu chatbota pomocí balíčků. Následující část stručně popisuje, které konkrétní třídy v aplikačním kódu se podílejí na řízení konverzace. Hlavní třída, která zpracovává vstupní požadavky, se nazývá `PimBot`. Tato třída implementuje rozhraní `IBot`, které pochází z aplikačního rámce Microsoft Bot Framework. Třída `PimBot` poté implementuje metodu `OnTurnAsync`, která má za úkol zpracování vstupního požadavku. Dále tato metoda zjišťuje záměr vstupního řetězce pomocí služby LUIS a vypisuje uvítací zprávu uživateli. Metoda posléze přenechává řízení třídě `MainDispatcher`, která slouží jako řadič.

Třída `MainDispatcher`, dědící z třídy `ComponentDialog`, kontroluje kontext konverzace. V případě, kdy je rozpracován nějaký dialog, třída přenechá řízení třídě reprezentující daný dialog. V případě, kdy není rozpracován žádný dialog (tj. na počátku konverzace, případně

po skončení jiného dialogu), třída na základě záměru uživatele spustí daný dialog reprezentovaný třídou dialogu. Například, pokud je rozeznán záměr `FindItem`, je spuštěn dialog reprezentovaný třídou `FindItemDialog`.

Jednotlivé třídy reprezentující dialogy jsou typu `WaterfallDialog`. To znamená, že tyto dialogy definují sled událostí, které jdou sekvenčně za sebou. Jednotlivé události jsou pak od sebe rozděleny uživatelovým vstupem. Na obrázku 6.4 je zobrazen sekvenční diagram, který popisuje komunikaci instancí tříd za účelem nalezení produktu. Tento obrázek shrnuje popis řízení komunikace uvedený výše.

6.5 Autentizace uživatele

V kapitole 4.4 byl představen požadavek na autentizaci uživatele, tedy že chatbot musí být schopen ověřit totožnosti uživatele. Jeden z důvodů je personalizace. Například ukládání osobních údajů, aby je uživatel nemusel znovu zadávat, případně persistenci nákupního košíků. Forma autentizace je poté závislá na výstupním kanálu chatbota, pomocí něhož komunikuje s uživatelem. Například, pokud by byl výstupní kanál chatbota Facebook Messanger, není nutné řešit autentizaci, jelikož se o ní stará samotný Facebook. V případě diplomové práce je však hlavní platforma webová aplikace.

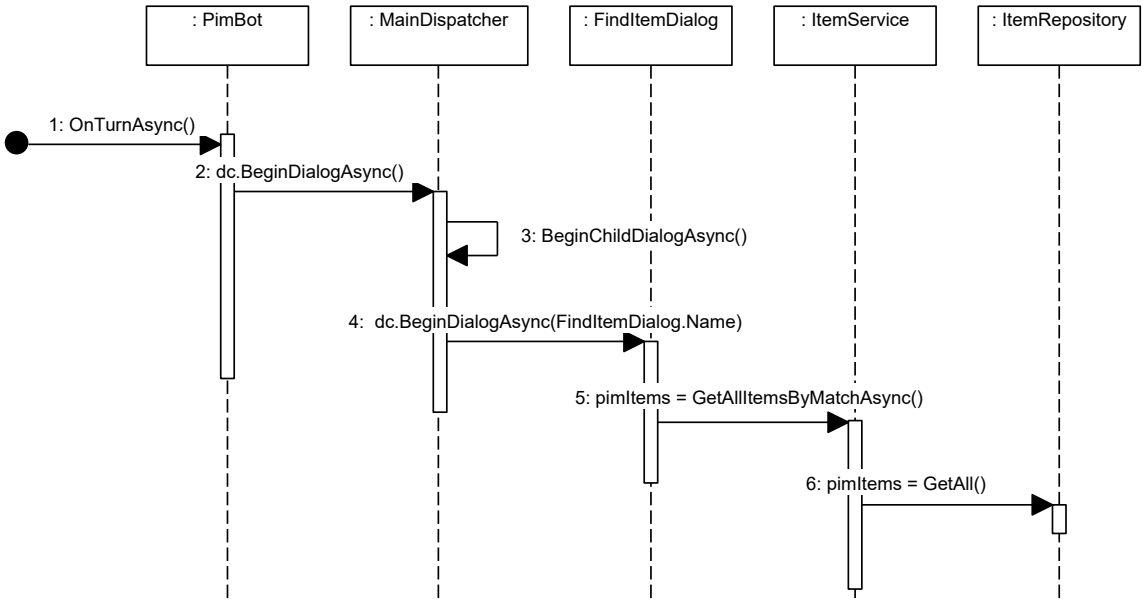
První myšlenka autentizace uživatele byla taková, že by byly některé funkce chatbota veřejné, zatímco k jiným by se musel uživatel přihlásit. Například, pokud by uživatel dokončil objednávku, chatbot by vyzval uživatele k přihlášení s tím, že by uživatel doplnil přímo v chatu (nebo pomocí Microsoft Adaptive Cards) své přihlašovací údaje. Nicméně, tato možnost se ukázala pomocí Microsoft Bot Framework jako nereálná. Microsoft Bot Framework z důvodu bezpečnosti neumožňuje skrytí uživatelovy zprávy, jako například umožňuje HTML pomocí textového pole s hvězdičkami. To znamená, že by chatbot vyzval uživatele k zadání uživatelského jména a hesla, ale heslo by nebylo nikterak skryté, což není rozhodně správný postup. Jeden z důvodů, proč Microsoft Bot Framework nepodporuje skrytí vstupního uživatelova textu je skutečnost, že se framework snaží, aby uživatel nepsal citlivé údaje do chatu, jelikož většina chatbotů má nastaveno logování, tím pádem by se logovala i citlivá data uživatele.

Microsoft Bot Framework nicméně podporuje autentizaci pomocí protokolu OAuth. To znamená, že je nutné využívat jinou externí službu, která zapouzdřuje a ověřuje totožnost uživatele. Microsoft doporučuje především službu na správu identit nazvanou Azure Active Directory, ale lze použít i jiné OAuth poskytovatele jako Google, Github či Facebook [3].

Autentizace uživatele přes protokol OAuth však není v rámci diplomové práce vhodná, jelikož se v budoucnu počítá se správou uživatelů pomocí systému PIM, který však v současnosti tuto funkcionalitou nepodporuje. Jako nejsnadnější a nejpraktičtější řešení se ukázalo neřešit autentizaci pomocí chatbota, nýbrž samostatně v rámci **webového klienta**. V kapitole 5.3.4 byla představena technologie BotFramework-WebChat, která je použita pro implementaci klientské webové aplikace chatbota. Před rozhraní webového chatu je předsazen přihlašovací formulář. V případě úspěšné autentizace poté webový klient vyrendruje daný webový chat s tím, že je tomuto chatu předán identifikátor uživatele (ukázka je v kódu 1). Samotný chatbot (tj. serverová aplikace) poté dostává identifikátor uživatele. Není tedy již nutné, aby chatbot cokoliv ověřoval. Chatbot poté používá tento identifikátor uživatele k přístupu k databázi CosmosDB (viz 5.5), v níž jsou uložena uživatelská data.

Tato jednoduchá webová aplikace pro ověření totožnosti uživatele a zobrazení chatovacího okna je vytvořena pomocí aplikačního rámce ASP.NET Core. Pro tvorbu uživatelského rozhraní webové aplikaci je využit aplikační rámec Bootstrap 4. Samotná autentizace slouží

pouze pro demonstrační účely a v současnosti podporuje 3 uživatele, kteří jsou uloženi na pevno v kódu webové aplikace. Jak bylo řečeno výše, v budoucnu se počítá, že bude k autentizaci využit PIM server.



Obrázek 6.4: Sekvenční diagram popisující zpracování požadavku "hledání produktu" (zdroj vlastní).

Kapitola 7

Testování funkčnosti chatbota

Tato kapitola se věnuje testování, což je proces, který je nezbytný pro ověření správné funkčnosti chatbota. Samotné testování je rozděleno na tři úrovně. První část (7.1) se zabývá automatizovaným testování extrakcí záměrů a entit pomocí LUIS. Druhá část (7.1) testování se poté zaměřila na jednotkové testování serverové části chatbota. Poslední část (7.3) se věnuje akceptačnímu testování, které bylo provedeno u firmy Allium s.r.o.

7.1 Testování služby LUIS

Služba LUIS byla detailně představena v kapitole 5.3.2. Její správné fungování je nezbytné pro celkové správné chování chatbota, jelikož pomáhá zjišťovat uživatelův záměr. Testování této služby má dva důležité dopady. Prvním je samotné ověření správného fungování služby. Druhým a významnějším dopadem je, že usnadňuje **regresní testování**. Služba LUIS je totiž velmi citlivá na změny modelu. Někdy stačí přidat nový záměr a přidat ukázkové věty (angl. utterance), které jsou v konfliktu s jinými ukázkovými větami jiného záměru. To může mít za důsledek nesprávné rozpoznání původního záměru. Při jakékoliv změně modelu je proto vhodné otestovat, zdali nebyla narušena původní funkcionalita.

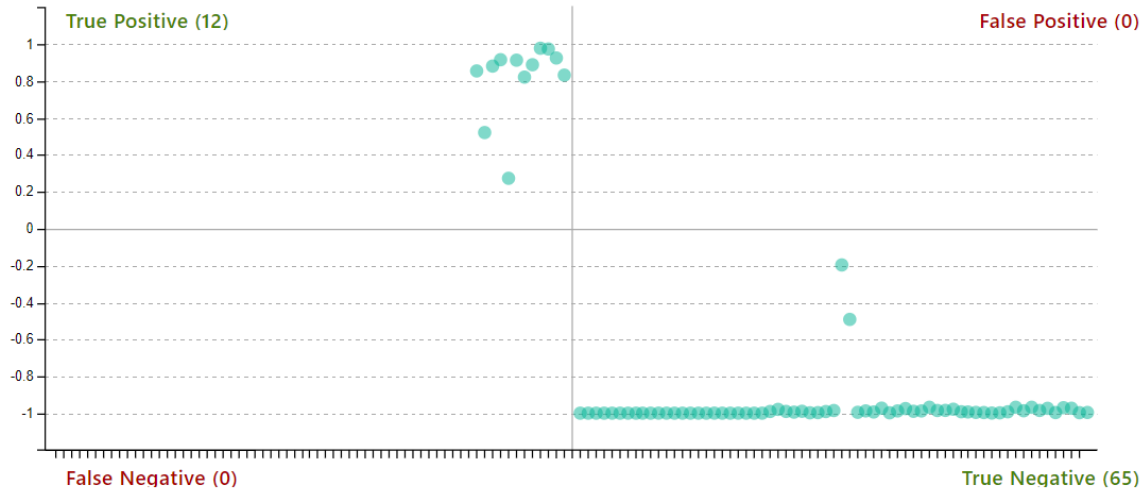
```
[
   {
      "text":"Find xiaomi redmi note 5",
      "intent":"FindItem",
      "entities":[
         {
            "entity":"item",
            "startPos":5,
            "endPos":24
         }
      ]
   }
]
```

Kód 4: Ukázka testovací sady služby LUIS.

Testování se provádí skrze rozhraní webové aplikace Luis Portal. K dispozici jsou dva typy testování vytvořeného modelu. První je přes panel nazvaný *Single testing panel*. Tento panel umožňuje napsat větu, která vstoupí do modelu. Výstupem je poté zachycený záměr a rozpoznané entity včetně výsledného skóre. Druhým a více praktickým způsobem je testování v dávkách skrze panel nazvaný *Batch testing panel*. Tato funkce umožňuje vytvořit sadu testů, které se poté spustí a ověří správnost modelu. Sadu testů je nutné vytvořit mimo aplikaci, a poté ji do aplikace nahrát. Tato sada má jasně definovanou syntaxi zápisu s tím, že cílový formát je typu JSON. V kódu 4 je zobrazena ukázková testovací sada obsahující jeden test. Tento test kontroluje, zdali model pozná záměr `FindItem` ze vstupní věty. Celkově bylo v rámci diplomové práce vytvořeno **78 testů** služby LUIS, které pokrývají všechny definované záměry a entity použité v aplikaci, viz příloha C.

Pro vyhodnocení výsledků slouží v aplikaci Luis Portal speciální panel. Tento panel zobrazuje pro každý záměr graf, který ukazuje jednotlivé testovací věty a jejich číselné ohodnocení. Na základě těchto grafů lze poté optimalizovat jednotlivé záměry, jelikož vývojář přehledně vidí, které testovací věty mají pro daný záměr podobné ohodnocení. Na obrázku 7.1 je zobrazen graf reprezentující vyhodnocení záměru `Confirm`. Ve kvadrantu grafu nazvaného *True Positive* jsou zobrazeny právě ty testovací věty, které byly modelem označeny správně za záměr `Confirm`. Naopak v kvadrantu *True Negative* jsou zobrazeny testovací věty, které byly správně označeny, že nejsou záměru `Confirm`.

Testování služby LUIS pomocí aplikace Luis Portal má nicméně jeden nedostatek. V současnosti není možné spouštět testovací scénáře přes Luis CLI, tedy přes nástroj příkazové řádky a ani vzdáleně skrze aplikační rozhraní. V současném stavu lze spouštět testy pouze přes webové rozhraní služby.



Obrázek 7.1: Vyhodnocení testování v aplikaci Luis Portal – graf pro záměr Confirm (snímek obrazovky).

7.2 Jednotkové testování

Jednotkové testování (nebo taktéž unit testing) je proces automatizovaného testování samostatných komponent v programu jako jsou metody, objekty nebo třídy. Jednotkové testování by mělo pomocí různých vstupů pokrýt celou funkcionalitu dané komponenty. Test jednotky

by poté měl být nezávislý a měl by pokrývat pouze danou jednotku, nikoliv její integraci s jinými částmi systému [29].

Aby bylo možné testovat pomocí jednotkových testů, bylo nezbytné navrhnout chatbota odpovídajícím způsobem tak, aby jednotlivé části na sobě byly nezávislé. Jeden z důvodů, který byl vzpomenut dříve, byl ten, aby test pokryl pouze daný kus funkcionality. Druhý a závažnější důvod byl, aby bylo možné použit princip mockování[1]. V rámci toho, že jsou na sebe některé komponenty závislé, tedy že třída potřebuje instanci jiné třídy, aby mohla pracovat, je nutné tuto závislou třídu mockovat. Například třída `ItemService` v aplikaci chatbota potřebuje ke svému fungování objekt implementující rozhraní `IItemRepository`. Rozhraní `IItemRepository` zapouzdřuje získávání dat z externího serveru. Pro otestování správné funkčnosti třídy `ItemService` tak, aby nebyla nutná komunikace s PIM serverem, je nutné rozhraní `IItemRepository` mockovat.

Aby bylo možné některé komponenty v aplikaci chatbota mockovat, bylo nutné použít návrhový vzor Vkládání závislostí (angl. Dependency injection), který umožňuje takzvané obrácení závislostí (angl. inversion of controll). Aplikační rámec ASP.NET Core nabízí jistou podporu tohoto návrhového vzoru. V hlavní konfigurační třídě nazvané `Startup` lze v metodě `ConfigureServices` registrovat služby do kontejneru služeb, ke kterým mají komponenty přístup skrze konstruktor. Těmto registrovaným službám lze poté nastavit životnost (angl. lifecycle nebo taktéž scope). V rámci diplmové práce je využívána životnost typu Jedináček (angl. Singleton), což znamená, že při registraci je vytvořena jedna instance dané třídy, kterou kontejner služeb obsahuje po celý běh aplikace [17].

Název testovací sady (třídy)	**Počet testů**
CategoryServiceTest	4
CommonUtilTest	6
FeatureServiceTest	6
ItemServiceTest	25
KeywordServiceTest	5

Tabulka 7.1: Detail sad jednotkových testů.

Pro vytvoření samostatných jednotkových testů je použit aplikační rámec NUnit. Tento rámec slouží pro tvorbu jednotkových testů na platformě .NET. Svoji podobou je podobný technologii nazvané JUnit, která slouží pro tvorbu jednotkových testů na platformě Java. Dále je pro testování využita knihovna **Moq**[2], která slouží pro jednoduché vytváření mocků. V rámci implementace jednotkových testů vznikly dvě pomocné třídy: třída `FakeDataGenerator`, která slouží pro generování testovacích dat a třída `MockService Generator`, jejímž účelem je generování mockovaných služeb. Celkově bylo vytvořeno **46 jednotkových testů**. Detail jednotlivých sad je zobrazen v tabulce 7.1. Tyto testy jsou zaměřeny na služby, tedy třídy v balíků *Services*, které zapouzdřují hlavní funkcionalitu, která je poskytnuta třídám, které řídí tok konverzace. Zajímavý je například test nazvaný `GetAllFeaturesToAsk_Values_Test()` nacházející se ve třídě `ItemServiceTest`, který otestuje správné pořadí otázek, na které se má chatbot zeptat a taktéž jednotlivé možné odpovědi, které chatbot v konverzaci nabídne.

[1] Mock je objekt, který simuluje chování jiného objektu. Mock implementuje stejné rozhraní jako simulovaný objekt [29]. Pojmem mockování je v práci myšleno vytváření takových objektů.

[2] Github repozitář knihovny: https://github.com/moq/moq4

7.3 Akceptační testování

Akceptační testování je proces ověření, že výsledný program splňuje své původní požadavky. Toto testování bývá často posledním stupněm při validaci programu a běžně se provádí manuálně u koncového zákazníka (tj. zadavatele) [23].

V rámci diplomové práce vznikla sada testů, která vychází přímo z funkčních požadavků na chatbota, viz 4.4. Každý funkční požadavek je pokryt právě jednou testovací sadou. Tyto testy byly poté testovány společně s firmou Allium s.r.o., čímž se ověřila požadovaná funkčnost chatbota. Jednotlivé testovací sady včetně testů jsou k nalezení v protokolu akceptačního testování v příloze D.

Kapitola 8

Závěr

Cílem diplomové práce byl návrh a implementace chatbota, který usnadňuje zákazníkům vyhledání a výběr konkrétního produktu, který si přeje zákazník zakoupit. Chatbot dále provádí zákazníka celým procesem vytvoření objednávky včetně sběru požadovaných zákaznických údajů. V první části práce byla provedena analýza zaměřená na tvorbu konverzačních rozhraní a dostupných technologií, které jsou vhodné na jejich tvorbu. Posléze byla navržena integrace chatbota se systémem Pimics společnosti Allium s.r.o., což je systém správy podnikových informací, jehož cílem je správa produktového portfolia firmy. Samotná integrace byla navržena pomocí OData webových služeb. Dále byly navržené jednotlivé konverzační toky chatbota pomocí stavových diagramů. Navržená architektura chatbota se poté skládá z několika samostatných komponent, které jsou nasazeny v cloudové platformě Microsoft Azure.

Samotná implementace je rozdělena do dvou celků. První část tvoří serverovou část chatbota a je vytvořena pomocí programovacího jazyka C# za použití aplikačního rámce Microsoft Bot Framework. K rozpoznání záměru uživatele a extrakci důležitých entit ze vstupní věty uživatele je použita služba LUIS. Chatbot dále využívá službu QnA Maker, která slouží na podporu běžných rozhovorů (tzn. smalltalks). Jedním z největších problémů řešených v práci bylo získávání zpětné vazby ohledně vlastností dané skupiny předmětů za účelem užšího výběru. Pro výběr pořadí dotazovaných vlastností byla použita teorie tvorby rozhodovacích stromů. Konkrétně byl implementován algoritmus ID3 sloužící na konstrukci rozhodovacího stromu. Druhá část řešení tvoří webová aplikace, která řeší autentizaci uživatele a rendrování webového chatu. K implementaci je využita technologie ASP.NET Core, BotFramework-WebChat a Bootstrap 4.

Chatbot splňuje všechny funkcionální požadavky, které byly v průběhu řešení práce přesně specifikované. Ověření funkcionality bylo ověřeno pomocí testování. Byly vytvořeny 3 úrovně testů. První sada automatizovaně testuje extrakci záměru a entit pomocí služby LUIS. Druhá sada je tvořena automatizovanými jednotkovými testy serverové části chatbota. Poslední část je tvořena akceptačními testy, které byly provedeny manuálně u spolupracující firmy.

Následující rozvoj

O chatbota, který byl vytvořený v rámci diplomové práce, již projevil zájem potencionální zákazník zadavatele práce. V následujících týdnech tedy proběhne testování chatbota přímo u zákazníka na jeho datové infrastruktuře. Díky ukládání všech konverzací do databáze

a zpětné vazbě zákazníka vznikne spousta podmětů k vylepšení chatbota. Budoucí rozvoj se poté zaměří na:

- Vytvoření více jazykových mutací chatbota. V současnosti je podporována pouze angličtina.
- Optimalizaci dotazů na systém správy podnikových informací včetně podpory rychlé vyrovnávací paměti (tj. cashe) za účelem zvýšení rychlosti odezvy chatbota.
- Podporu složitějších vzorů při hledání konkrétního produktu tak, aby bylo možné s hledaným produktem rovnou zadat požadované vlastnosti.

Literatura

[1] *Allium - O nás*. [Online; navštíveno 30.12.2018].
URL https://www.allium.cz/cz/o-nas

[2] *Amazon Lex*. [Online; navštíveno 28.12.2018].
URL https://en.wikipedia.org/wiki/Amazon_Lex

[3] *Azure Bot Service Documentation*. [Online; navštíveno 07.05.2019].
URL https://opdhsblobprod02.blob.core.windows.net/contents/5f0dd15031744758b5df82fd6be87f9a/1d7ab6ecd5717f32d8f416cfb9869089?sv=2015-04-05&sr=b&sig=MhP%2FWH2KdwRS6NxWRfGD7GJ5OCzsKMWQCvGDZ9%2FXXco%3D&st=2019-05-07T19%3A31%3A05Z&se=2019-05-08T19%3A41%3A05Z&sp=r

[4] *Chinese room*. [Online; navštíveno 28.12.2018].
URL https://en.wikipedia.org/wiki/Chinese_room

[5] *Dokumentace ke službě Azure App Service – kurzy, reference k rozhraní API*. [Online; navštíveno 23.4.2019].
URL https://docs.microsoft.com/cs-cz/azure/app-service/

[6] *Dynamics NAV - Developer and IT-Pro Help*. [Online; navštíveno 29.12.2018].
URL https://docs.microsoft.com/en-us/dynamics-nav/opbuildpdf/TOC.pdf?branch=live

[7] *How intent classification works in NLU*. [Online; navštíveno 28.12.2018].
URL https://mrbot.ai/blog/natural-language-processing/understanding-intent-classification/

[8] *ID3 algorithm*. [Online; navštíveno 01.05.2019].
URL https://en.wikipedia.org/wiki/ID3_algorithm

[9] *Metaphone*. [Online; navštíveno 02.04.2019].
URL https://en.wikipedia.org/wiki/Metaphone

[10] *Microsoft/BotFramework-WebChat: A highly-customizable web-based client for Azure Bot Services*. [Online; navštíveno 21.4.2019].
URL https://github.com/Microsoft/BotFramework-WebChat

[11] *Natural language processing*. [Online; navštíveno 18.11.2018].
URL https://en.wikipedia.org/wiki/Natural_language_processing

[12] *Open Data Protocol* . [Online; navštíveno 04.1.2019].
URL https://en.wikipedia.org/wiki/Open_Data_Protocol

[13] *Popular Chatbot Frameworks*. [Online; navštíveno 28.12.2018].
URL https://discover.bot/bot-talk/beginners-guide-bots/popular-chatbot-frameworks/

[14] *Product Information Management (PIM)*. [Online; navštíveno 10.1.2019].
URL https://www.pimics.com/en-US/Product-modules/Product-Information-Management

[15] *What is Language Understanding (LUIS)?* . [Online; navštíveno 20.4.2019].
URL https://docs.microsoft.com/en-us/azure/cognitive-services/luis/what-is-luis

[16] *What is QnA Maker?* [Online; navštíveno 21.4.2019].
URL https://docs.microsoft.com/cs-cz/azure/cognitive-services/qnamaker/overview/overview

[17] *Dependency injection in ASP.NET Core* . [Online; navštíveno 03.05.2019].
URL https://docs.microsoft.com/cs-cz/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-2.2

[18] Abraham, J.: *Product information management*. Cham: Springer, [2014], ISBN 978-3-319-04884-0.

[19] Bird, S.; Klein, E.; Loper, E.: *Natural language processing with Python*. Cambridge [Mass.]: O'Reilly, druhé vydání, c2009, ISBN 978-0-596-51649-9.

[20] Janarthanam, S.: *Hands-On Chatbots and Conversational UI Development*. Birmingham: PacktPublishing Ltd., 2017, ISBN 978-1-78829-466-9.

[21] Jurafsky, D.; Martin, J. H.: *Speech and language processing*. Upper Saddle River, N.J.: Pearson Prentice Hall, druhé vydání, 2009, ISBN 978-0131873216.

[22] Manning, C. D.; Raghavan, P.; Schutze, H.: *Introduction to information retrieval*. New York: Cambridge University Press, 2008, ISBN 978-052-1865-715.

[23] Myers, G. J.; Badgett, T.; Thomas, T. M.; aj.: *The art of software testing*. Hoboken, N.J.: John Wiley, druhé vydání, c2004, ISBN 978-047-1469-124.

[24] Pennington, J.; Socher, R.; Manning, C. D.: GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, s. 1532–1543, [Online; navštíveno 30.12.2018].
URL http://www.aclweb.org/anthology/D14-1162

[25] Raj, S.: *Building chatbots with Python*. New York, NY: Springer Science Business Media, 2018, ISBN 978-1484240953.

[26] Shaikh, K.: *Developing bots with QNA maker service*. New York, NY: Springer Science Business Media, 2018, ISBN 978-1-4842-4184-4.

[27] Shevat, A.: *Designing bots*. Boston: O'Reilly, 2017, ISBN 978-149-1974-827.

[28] Sommerville, I.: *Software engineering*. 9th ed. Boston: Pearson, 2011, ISBN 978-0-13-703515-1.

[29] Sommerville, I.: *Software engineering*. Boston: Pearson, tenth edition vydání, [2016], ISBN 978-0-13-394303-0.

[30] Theobald, O.: *Machine Learning for Absolute Beginners: A Plain English Introduction*. Independently Published, 2018, ISBN 9781549617218.

[31] Weizenbaum, J.: *Computer power and human reason*. New York, N.Y.: Penguin, 1984, c1976., ISBN 01-402-2535-8.

[32] Wu, Y.; Wu, W.; Li, Z.; aj.: Topic Augmented Neural Network for Short Text Conversation. *CoRR*, ročník abs/1605.00090, 2016, [Online; navštíveno 18.11.2018], 1605.00090.
URL http://arxiv.org/abs/1605.00090

[33] Zbořil, F.; Zbořil, F.: *Základy umělé inteligence - Studijní opora*. FIT VUT v Brně, Interní materiál, 2012, [Online; navštíveno 30.12.2018].
URL
https://www.fit.vutbr.cz/study/courses/IZU/private/oporaizu-esf-5a.pdf

Příloha A

Obsah přiloženého paměťového média

- /
  - `pimBotDp/` ..... adresář obsahující zdrojové kódy serverové části chatbota
  - `pimBotDpTest/` ..... adresář obsahující zdrojové kódy jednotkových testů
  - `pimBotDpWebChat/` ..... adresář obsahující zdrojové kódy klientské části chatbota
  - `doc/`
    - `dataset/` ..... adresář obsahující datovou sadu k službě QnA Maker a testy pro službu Luis.ai
    - `img_src/` ..... adresář obsahující zdrojové soubory obrázků
    - `tex_src/` ..... adresář obsahující zdrojové soubory textu práce
  - `dp.pdf` ..... text diplomové práce ve formátu pdf

Příloha B

Návod k manuální instalaci

V následující části je představen návod na spuštění chatbota na lokálním PC z důvodů vývoje a testování. Chatbot má defaultně nastavenou komunikaci s Luis.ai a QnA službu. V případě, kdy je nutné nahradit Luis.ai službu za jinou (například z důvodů financování), je nutné použít CLI nástroj MSBot[1].

B.1 Prerekvizity

- Nástroj Bot Framework Emulator (V4).
- Multiplatformní nástroj příkazové řádky .NET Core command-line tools[2].

B.2 Instalační postup

1. Přístup do adresáře obsahující zdrojové kódy serverové části chatbota.

   ```
   cd PimBotDp/src
   ```

2. Sestavení aplikace chatbota pomocí nástroje dotnet.

   ```
   dotnet build
   ```

3. Lokální spuštění chatbota. Chatbot bude defaultně poslouchat na adrese localhost:3978.

   ```
   dotnet run
   ```

4. Připojení na chatbota pomocí aplikace Bot Framework Emulator. Otevření nabídky File -> Open Bot Configuration a zvolit soubor PimBotDP.bot

[1] Github repozitář nástroje MSBot – https://github.com/microsoft/botbuilder-tools/tree/master/packages/MSBot

[2] Github repozitoř nástroje dotnet cli: https://github.com/dotnet/cli

Příloha C

Navržené záměry a entity

Záměr	Entita	Počet výroků	Příklad výroku	Popis
AddItem	item	3	add xiomi redmi note 5	Záměr reprezentující přidání zboží do nákupního košíku.
Confirm		5	confirm order	Záměr popisuje potvrzení objednávky a přesun k jejímu dokončení.
DetailItem	item	2	show me detail 55697	Záměr, který představuje akci, kdy si chce uživatel zobrazit detail vyhledaného předmětu.
FindItem	item	8	looking for phones	Záměr vyjadřuje vyhledání předmětu nebo skupiny předmětů.
None		1	none	Tento záměr slouží k odchycení případu, kdy nebude použit žádný jiný záměr.
RemoveItem	item	4	remove item	Záměr reprezentuje odstranění daného předmětu z nákupního košíku.
ShowCart		5	show my cart	Záměr představuje akci, kdy si chce uživatel zobrazit svůj aktuální nákupní košík.
ShowCategories		4	show all categories	Tento záměr popisuje událost, kdy chce uživatel zobrazit všechny dostupné kategorie zboží.

ShowOrders		4	show orders	Záměr sloužící pro zobrazení všech aktivních objednávek.
SmallTalk		567	I love you	Tento záměr slouží pro rozpoznání běžných konverzací.
Utilities. Cancel		13	I want to cancel	Jedná se o předdefinovaný záměr, který slouží k rozpoznání obecného záměru uživatele zrušit současnou akci.
Utilities. Help		15	Help me please	Předdefinovaný záměr sloužící k rozpoznání, že uživatel vyžaduje pomoc.

Tabulka C.1: Navržené záměry a entity ve službě Luis.ai.

Příloha D

Protokol akceptačního testovaní

Autentizace uživatele

Testovací scénář	Očekávaný výsledek	Stav
1.1: Uživatel přistoupí na domovskou stránku chatbota.	Uživateli je zobrazen přihlašovací formulář.	Prošel
1.2: Uživatel zadá správné přihlašovací údaje do přihlašovacího formuláře	Uživatel je přihlášen.	Prošel
1.3: Uživatel zadá nesprávné přihlašovací údaje do přihlašovacího formuláře.	Uživatel nebyl přihlášen a je mu zobrazeno upozornění na nesprávné údaje.	Prošel
1.4: Uživatel je přihlášen do systému.	Uživatel je přesměrován do webového chatu, kde ho chatbot pozdraví jeho uživatelským jménem.	Prošel
1.5: Uživatel stiskne tlačítko odhlásit ve webovém chatu.	Uživatel je odhlášen ze systému a přesměrován na domovskou stránku, kde je mu zobrazen přihlašovací formulář.	Prošel

Tabulka D.1: Testovací sada: Autentizace uživatele.

Vyhledání produktu

Testovací scénář	Očekávaný výsledek	Stav
2.1: Uživatel vyhledává jeden konkrétní produkt podle jeho specifického názvu.	Chatbot zobrazí nalezený produkt a nabídne možnost přidat produkt do košíku (objednávky) nebo zobrazit detail produktu.	Prošel
2.2: Uživatel vyhledává skupinu předmětů podle klíčového slova.	Chatbot zobrazí, že nalezl několik potencionálních produktů a nabídne možnost zobrazit všechny produkty nebo získat zpětnou vazbu ohledně atributů.	Prošel
2.3: Uživatel vyhledává skupinu předmětů podle jména kategorie.	Chatbot zobrazí, že nalezl několik potencionálních produktů a nabídne možnost zobrazit všechny nebo získat zpětnou vazbu ohledně atributů.	Prošel
2.4: Uživatel chce zobrazit všechny produkty z nalezené skupiny produktů.	Chatbot zobrazí nalezené předměty a nabídne možnost přidat produkt do košíku (objednávky) nebo zobrazit detail produktu.	Prošel
2.5: Uživatel chce zobrazit detail daného vyhledaného předmětu.	Chatbot zobrazí detail nalezeného produktu.	Prošel
2.6: Uživatel vybere možnost, že se chce u nalezených produktů nechat chatbotem doptat na vlastnosti.	Chatbot vybere nejvíce relevantní otázku a zeptá se uživatele na společnou vlastnost produktů.	Prošel

Tabulka D.2: Testovací sada: Vyhledání produktu.

Přidání produktu do objednávky

Testovací scénář	Očekávaný výsledek	Stav
3.1: Uživatel přidá existující produkt do objednávky (košíků).	Chatbot se uživatele zeptá, kolik kusů daného produktu si přeje objednat.	Prošel
3.2: Uživatel přidá neexistující produkt do objednávky.	Chatbot uživatele upozorní, že daný produkt neexistuje v rámci obchodu a proto nebyl přidán do objednávky	Prošel
3.3: Uživatel odpoví číselným údajem, kolik kusů zboží si přeje objednat.	Chatbot potvrdí uživateli objednávku.	Prošel
3.4: Uživatel odpoví nečíselným údajem, kolik kusů zboží si přeje objednat.	Chatbot uživatele upozorní, že musí zadat číselný údaj.	Prošel

Tabulka D.3: Testovací sada: Přidání produktu do objednávky.

Zobrazení aktuální objednávky

Testovací scénář	Očekávaný výsledek	Stav
4.1: Uživatel zobrazí objednávku (košík), která je prázdná.	Chatbot uživatele upozorní, že je objednávka zatím prázdná.	Prošel
4.2: Uživatel zobrazí objednávku obsahující položky.	Chatbot zobrazí uživateli všechny produkty uložené v objednávce včetně počtu kusů a zároveň upozorní, jak lze objednávku potvrdit.	Prošel

Tabulka D.4: Testovací sada: Zobrazení aktuální objednávky.

Odebrání produktu z objednávky

Testovací scénář	Očekávaný výsledek	Stav
5.1: Uživatel odstraňuje z objednávky produkt, který v ní **není** obsažen.	Chatbot uživatele upozorní, že objednávka neobsahuje daný produkt.	Prošel
5.2: Uživatel odstraňuje z objednávky produkt, který v ní **je** obsažen.	Chatbot se uživatele zeptá, zdali si je jistý s odebráním produktu.	Prošel
5.3: Uživatel odpovídá chatbotovi, že si **je** jistý odebráním zboží z objednávky.	Chatbot odebere produkt z objednávky a potvrdí tuto akci uživateli.	Prošel
5.4: Uživatel odpovídá chatbotovi, že si **není** jistý odebráním zboží z objednávky.	Chatbot odebere produkt z objednávky a potvrdí tuto akci uživateli.	Prošel

Tabulka D.5: Testovací sada: Odebrání produktu z objednávky.

Dokončení objednávky

Testovací scénář	Očekávaný výsledek	Stav
6.1: Uživatel dokončí objednávku bez uložených uživatelských údajů.	Chatbot vyzve uživatele k zadání osobních údajů nutných k objednávce.	Prošel
6.2: Uživatel vyplňuje osobní informace nutné k objednávce.	Chatbot se zeptá na to, zdali korespondenční adresa je stejná jako adresa uživatele. V případě, kdy uživatel odpoví, že není, chatbot se dále zeptá i na korespondenční údaje.	Prošel
6.3: Uživatel zadá všechny informace nutné k objednávce.	Chatbot ověří správnost uživatelových informací tím způsobem, že je všechny vytiskne a zeptá se uživatele, zdali jsou v pořádku.	Prošel
6.4: Uživatel dokončí objednávku s uloženými uživatelskými údaji.	Chatbot ověří správnost uživatelových informací tím způsobem, že je všechny vytiskne a zeptá se uživatele, zdali jsou v pořádku.	Prošel
6.5: Uživatel zvolí odpověď, že jeho informace nejsou v pořádku.	Chatbot se zeptá uživatele, které jeho údaje nejsou v pořádku.	Prošel
6.6: Uživatel potvrdí své osobní údaje.	Chatbot ověří celou objednávku tím, že vytiskne uživatelské informace a celý obsah objednávky včetně finální ceny.	Prošel
6.7: Uživatel **potvrdí** objednávku.	Chatbot odešle objednávku a tuto skutečnost potvrdí uživateli.	Prošel
6.8: Uživatel **nepotvrdí** celou objednávku.	Chatbot anuluje objednávku.	Prošel

Tabulka D.6: Testovací sada: Dokončení objednávky.

Zobrazení dokončených objednávek

Testovací scénář	Očekávaný výsledek	Stav
7.1: Uživatel zobrazí dokončené objednávky.	Chatbot zobrazí všechny dokončené objednávky, včetně finálních cen a data vytvoření.	Prošel
7.2: Uživatel zobrazí dokončené objednávku (prázdné).	Chatbot uživatele upozorní, že doposud nebyla vytvořena žádná objednávka.	Prošel

Tabulka D.7: Testovací sada: Zobrazení dokončených objednávek.

Zobrazení dostupných kategorií

Testovací scénář	Očekávaný výsledek	Stav
8.1: Uživatel zobrazí všechny dostupné kategorie produktů.	Chatbot uživateli zobrazí všechny dostupné kategorie produktů.	Prošel

Tabulka D.8: Testovací sada: Zobrazení dostupných kategorií.