Univerzita Hradec Králové

Fakulta informatiky a managementu

Katedra informatiky a kvantitativních metod

E2E Bot tester

Diplomová práce

Autor práce: Bc. MARTIN MALÝ
Studijní obor: Aplikovaná informatika

Vedoucí práce: Ing. PAVEL KŘÍŽ, Ph.D.

Hradec Králové 2022

Prohlášení

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury.

. . . . . .

Martin Malý

28. dubna 2022

Poděkování

Děkuji společnosti Feedyou s.r.o. a především Ing. Janu Dvořákovi za konzultace spojené s implementací diplomové práce, PhDr. Pavle Sychrové, Ph.D. za jazykovou korekturu a Ing. Pavlu Křížovi, Ph.D. za vedení práce.

Anotace

Předkládaná práce se jako hlavním tématem zabývá dostupnými testovacími nástroji na trhu a implementací vlastního end-to-end(E2E) testovacího nástroje, a to podle potřeby firmy Feedyou, s.r.o. Ve své teoretické části přináší na základě analýzy odborné literatury a dostupných zdrojů přehledný základní popis testovacích nástrojů, jejich přínosy a příklady jejich použití. Uvádí se zde také popis celého ekosystému bota, s kterým aplikace pracuje. Praktická část práce se zaměřuje na několik stanovených cílů. Mezi ně patří popsat dostupné testovací nástroje na trhu, zjednodušit testování a zvýšit časovou úsporu spojenou s automatizací a automatizovat testování botů, přičemž se tato práce zabývá službou Microsoft Azure Bot Service, která úzce souvisí s implementací testovací aplikace. V praktické části se přímo implementuje webová aplikace v Node.js a Reactu, která dokáže otestovat správnost odpovědí a jejich včasné posílání. Praktická část práce je tedy zaměřena na vytvoření uživatelsky přívětivé aplikace a zároveň oddělení vzhledové části od serverové, díky čemu může být aplikace používána přes REST API i jinou aplikací a testy se tak mohou automatizovat.

Annotation

Title: E2E Bot tester

This thesis as a main topic focuses on testing tools currently available on the market and the implementation of its own end-to-end(E2E) testing tool, as needed by a firm Feedyou, s.r.o. In its theoretical part it sheds light on a clear basic description of testing tools, their benefits and examples of their use based on an analysis of specialised literature and available resources. There is also a description of the entire bot ecosystem with which the application works. The practical part of the thesis focuses on several set purposes. Those purposes include how to describe the testing tools available in the market, to simplify testing and how to increase time savings associated with automation and to automate bot testing, while this thesis focuses on Microsoft Azure Bot Service, which is closely related to a test application implementation. The practical part implements a web application directly in the React and Node.js, which can test the accuracy of responds and their timely send-off. The practical part of the thesis is therefore focused on creating a user-friendly Application altogether with separating the visual part from the server part at the same time, thanks to which the Application may be used via REST API and/or another application and the tests may be automated thereafter.

Obsah

1 Úvod **1**

2 Cíl práce **3**

3 Chatboti **4**
3.1 Historie 4
3.2 Přístupy strojového učení 6
3.3 Zpracování přirozeného jazyka (NLP) 6
3.4 Pochopení přirozeného jazyka (NLU) 6
3.5 Umělé neuronové sítě 7
3.6 Trendy v automatizaci komunikace 8
3.7 Voiceboti 8
3.8 Platformy pro vývoj chatbotů 9
3.9 Microsoft Azure Bot Service 10
3.10 Slabé stránky a hrozby chatbotů 12
3.11 Příklady chatbotů dle využití 14
3.12 Užitečné nástroje pro chatboty 16
3.13 Testování chatbotů 18

4 Společnost Feedyou **27**
4.1 Technologie Feedyou Platform 27
4.2 Výrobní proces 30
4.3 Designer 31

5 Implementace testovacího nástroje E2E Bot tester **35**
5.1 Motivace 35
5.2 Výběr nejvhodnějšího řešení 35
5.3 Bot tester ve Feedyou Platformě 37
5.4 Popis funkčnosti Back End 37
5.5 Popis funkčnosti Front End 46

6 Souhrn výsledků **51**

7 Závěry a doporučení **52**

Literatura **53**

Seznam zkratek **55**

Přílohy **57**

Seznam obrázků

1 Bot framework emulator, zdroj: autor 12
2 Ukázka Botanalytics, zdroj: autor 17
3 Ukázka Botium box, zdroj: autor 20
4 Konverzační strom nákupního bota [1] 21
5 Uživatel vytvoří přizpůsobenou kytici [1] 22
6 Uživatel si vybere kytici k výročí [1] 23
7 Ukázka různých uživatelských vstupů [1] 24
8 Příklad toku konverzace shora v Excelu [1] 25
9 Přesné výroky, které se mají použít, jsou na samostatném listu nebo dokonce v samostatném souboru. [1] 25
10 Moduly Feedyou Platform, zdroj: autor 28
11 Procesní diagram, zdroj: autor 32
12 Designer, zdroj: autor 34
13 Ekosystém fungování bota ve společnosti Feedyou, zdroj: autor 38
14 Ukázka úspěšného testu, zdroj: autor 47
15 Ukázka JSON transcriptu, zdroj: autor 48
16 Ukázka neúspěšného testu, zdroj: autor 49
17 Ukázka porovnání JSONů, zdroj: autor 50

Seznam tabulek

1 Manuální vs. automatické testování, zdroj: autor 51

Seznam ukázek kódů

1 Jazyk Gherkin 19
2 Transcript 36
3 Posílání zpráv 39
4 Uložení hlavního response a timeoutu 40
5 Vrácení chyby 41
6 Odchozí zpráva 42
7 Konec scénáře 43
8 Časový rozdíl 44
9 Integrační transcript 45

1 Úvod

Současný život se odehrává většinou ve světě, v němž převládá automatizace. Její neustále se rozšiřující převaha je patrná ve většině oborů, v nichž je možné jakoukoli automatizaci využít. Hlavním důvodem tohoto trendu je umožňovat lidem, aby se věnovali zajímavějším a kreativnějším činnostem. Můžeme sledovat, že v IT světě se v posledních letech velmi rozšířili a získali oblibu virtuální asistenti. Pro mnoho osob může být ale překvapením, že chatboti nejsou žádnou novinkou. První z nich byl vytvořen již v roce 1950. Avšak za jejich aktuální velkou expanzí stojí zejména současná změna způsobu mezilidské komunikace. V současné době má telefon s podporou Short message service (SMS) přes 6 miliard lidí. [2] Využití chatbota stojí vždy pouze na fantazii jeho tvůrců a na firmách, které ho potřebují. Virtuální asistent může být využit na přijímaní nových zaměstnanců a na získávání informací o nich, anebo třeba na hlášení poruch v pronajatých bytových jednotkách. Samozřejmostí je dnes i virtuální obchodník, který maximalizuje váš obrat, protože sám řeší mnoho situací v internetových obchodech, v oblasti realit, automotive, u pojišťoven, finančních institucí nebo také virtuální operátorka, která vám zajistí neustálou podporu pro nejrůznější dotazy na různých webových stránkách. Společnost Feedyou s.r.o. tyto virtuální asistenty na míru vytváří. Protože jsou v oboru více jak 5 let a firma se rozrostla, vyrábí čím dál tím více chatbotů. Ve firmě proto vznikla potřeba zautomatizovat jejich testování. Právě tento přínosný a zajímavý proces tvorby a implementace nové automatické testovací aplikace pro chatboty se stala hlavním předmětem předkládané práce jako celku.

Je vhodné si přesněji popsat a uvědomit, proč je automatizace testování důležitá a jak vlastně v současné době testování nejčastěji probíhá. Celý proces začíná oslovením klienta, zda chce chatbota na míru. Navíc si nabízející firma provádí analýzu trhu, takže obvykle již ví, jakého chatbota klientovi nabídnout. Následně chatbota výrobní oddělení nadefinuje podle požadavků klienta. Avšak, aby mohl být otestován, je třeba, aby s ním někdo osobně komunikoval, ručně si s ním psal. Jedině tak je zjištěno, zda odpovídá dle očekávání a požadavků. V případě, že definici chatbota jakkoli změníme, musí být znovu ručně testován a odpovědi musí být znovu validovány. V této fázi je tedy hlavním ovlivňujícím faktorem člověk a může tak vzniknout obtíž, s níž je třeba pracovat – lidský faktor může chatbota otestovat nesprávně a dojde tak k produkci nefunkčního virtuálního asistenta, který kvůli chybám v odpovědích napáchá možná více škody než užitku. Řešením je právě testovací aplikace E2E Bot tester, která tento proces zautomatizuje. Konverzační specialista, který je zodpovědný za tvorbu a údržbu struktury konverzace, si chatbota prokliká a popovídá si s ním. Avšak navíc vznikne z této konverzace záznam, který se uloží do databáze. Poté je možno definici chatbota změnit. Následně si načteme v naší testovací aplikaci výše uvedený záznam a dáme spustit test. Během vteřiny je možno vysledovat, zda chatbot na nezměněné větvi odpovídá stále stejně. V případě, že by se odpovědi neshodovaly, test zahlásí chybu a ukáže nám, co očekával a co chatbot reálně odpověděl. Ihned je tedy jasné, co v definici virtuálního asistenta opravit.

Zdánlivě nemusí být takto získaná úspora času nikterak velká, ale vzhledem k tomu, že jsou někteří chatboti velice komplexní, může s nimi komunikace trvat

i několik desítek minut. Mohou mít také velké množství větví a je třeba každou testovat zvlášť. Z tohoto pohledu je tedy zřejmé, že časová úspora zařízená E2E Bot testerem může být obrovská.

2 Cíl práce

Cílem této práce je vytvořit aplikaci na end-to-end testování chatbotů pro společnost Feedyou. Na modelovém případě se ověřilo, že tento nový přístup řešení je velmi efektivní a postupně se bude do firmy zavádět. Vzhledem k tomu, že ve společnosti stavíme mnoho rozsáhlých chatbotů, časové úspory budou obrovské.

Doposud museli pracovníci z výroby a testeři bota proklikat a zjistit, zda po úpravách v definici chatbota funguje stromová a Natural language processing (NLP) část dle očekávání. Aplikace E2E Bot tester vytvořená v rámci předkládané práce to zvládne za okamžik, což nejen šetří čas, ale i snižuje potřebu kontroly lidským faktorem v testovacím procesu a konverzační specialisté se můžou věnovat kreativní činnosti, kterou program nezvládne. Tato časová úspora se netýká pouze našich chatbot specialistů, ale jde i o zákazníky, protože v některých případech si chatbota rozvíjejí samostatně. Bot nejen že posílá zprávy, ale i přistupuje do jiných systémů, což může E2E Bot tester pokrýt také pomocí zapamatování si správných odpovědí a jejich „přehrání“ ve chvíli testování. Naše implementace chatbotů představují často rozsáhlé stromové struktury a otestování každé větve může trvat i desítky minut.

V budoucnu směřuje společnost Feedyou k nabízení nově vyvinuté platformy přes Microsoft store do celého světa a tím tak vstupovat na globální trh nejen jako tvůrci chatbotů, ale i jako technologičtí specialisté, kteří nabízí řešení pro další firmy, které budou naše chatboty implementovat zákazníkům po celém světě. Tím by se celková úspora času dostala do úplně jiných rozměrů.

3 Chatboti

Označení chatbot se vztahuje na jakoukoli softwarovou aplikaci, která se zapojuje do dialogu s člověkem pomocí přirozeného jazyka. Tento termín se nejčastěji používá v souvislosti s aplikacemi, které konverzují prostřednictvím psaného jazyka, ale s pokroky v rozpoznávání řeči se to stále více jeví jako nesprávný rozlišovací znak. V popředí této technologie jsou nejviditelnější hlasově řízení digitální asistenti z tzv. Velké čtyřky: Siri od Apple, Cortana od Microsoftu, Alexa od Amazonu a nový asistent od Googlu. V návaznosti na to existuje mnoho tisíc textových chatbotů, které se zaměřují na konkrétní funkce umožňující nástroje na vytváření robotů pro široce používané platformy pro zasílání zpráv. [2]

Svět se začíná plnit chatboty, které se používají v nejrůznějších odvětvích jako jsou například výroba, bankovnictví, pojišťovnictví, internetové obchodování, telekomunikace, zdravotnictví, cestovní ruch, vzdělávání, státní správa a mnoho dalších. Umožňují přístup ke všem druhům služeb (například rezervace letů, kontrola příhodnosti počasí...apod.). Hlavním důvodem k zavádění těchto chatbotů je odlehčení kapacity pozornosti, kterou může člověk práci věnovat. Lidské zdroje jsou v každém odvětví cenné a jejich nahrazování virtuálními asistenty lidským zdrojům odlehčuje. Asistentovi je jedno kolik je hodin, kdo má jakou náladu či po kolikáté je zase znovu něco po nich vyžadováno. Vždy vám ochotně pomohou. Dobře navržený chatbot dokáže odbavit až 80% komunikace, ušetření času je tedy obrovské. [3]

3.1 Historie

Následující část textu přibližuje základní informace o historii chatbotů, které zde byly shrnuty na základě dostupných odborných zdrojů. [4]. V roce 1950 Alan Turing uvažoval, zda počítačový program může někdy hovořit se skupinou lidí, aniž by si uvědomili, že je jejich partner umělý. Tato otázka, nazvaná Turingův test, je mnohými považovaná za generativní myšlenku chatbotů. První chatbot se jménem Eliza byl zkonstruován v roce 1966. Simuloval při svých nastavených operacích, že je psychoterapeut a vracel věty uživateli v tázací formě. Jeho schopnost komunikovat byla omezená, ale byl zdrojem inspirace pro následný vývoj dalších chatbotů. Eliza používala porovnání vzorců a schéma výběru dopovědí založené na šablonách. Její nevýhodou byla omezená znalost, a tak mohla komunikovat pouze v určité oblasti témat. Dalšími nevýhodami byly neschopnost vést dlouhé diskuse, učit se a objevovat souvislosti.

V roce 1972 se objevil chatbot Parry, který působil jako pacient se schizofrenií. Parry byl považován za pokročilejší než Eliza, protože měl mít osobnost a lepší kontrolní strukturu. Definice jeho reakcí byly založené na systému předpokladů a emocionálních reakcí aktivovaných změnou povahy výpovědi uživatele. Parry byl použit v experimentu v roce 1979, kdy pět soudních psychiatrů provedlo dálnopisem rozhovor s pacientem a počítačem, aby se rozhodli, zda se jedná o počítačový program nebo skutečného schizofrenního pacienta. Psychiatři stanovili deset diagnóz. První psychiatr stanovil dvě správné diagnozy, druhý dvě nesprávné, třetí se domníval, že oba subjekty byly skutečnými pacienty a další dva diagnostikovali, že oba byli chat-

boti. Tento vzorek psychiatrů je však malý a význam zjištění není jasný. Parry měl omezené schopnosti, co se týče porozumění jazyka a schopnost vyjadřovat emoce. Dále měl nízkou rychlost reakce a nemohl se učit z konverzace.

První chatbot s využitím umělé inteligence byl v roce 1988 Jabberwacky. Byl napsán v CleverScriptu. Tento jazyk byl založen na tabulkových procesorech, a tak usnadnil vývoj chatbotů. K reakci používal kontextové porovnávání vzorců. Jabberwacky měl i přesto vysokou latenci a nemohl pracovat s velkým počtem uživatelů. [4]

Termín Chatterbot byl poprvé zmíněn v roce 1991. Jednalo se o umělého hráče TINYMUD, což byl virtuální svět v reálném čase pro více hráčů. Primární funkcí tohoto světa bylo chatovat. V průběhu času se ukázalo, že mnoho skutečných lidských hráčů dává přednost rozhovoru s Chatterbotem před skutečným hráčem. [4]

Dalším krokem vpřed v historii chatbotů bylo v roce 1995 vytvoření Alice (Artificial Linguistic Internet Computer Entity). První online chatbot inspirovaný Elizou. Alice byla založená na porovnávání vzorců, bez jakéhokoli skutečného vnímání celého rozhovoru. Alice měla 41 000 šablon a souvisejících vzorců, což byl obrovský počet ve srovnání s Elizou, která měla pouze 200 klíčových slov a pravidel. Alice však neměla inteligentní funkce a nedokázala generovat lidské odpovědi, které vyjadřují emoce nebo postoje. [4]

V roce 2001 došlo ke skutečné revoluci v technologii chatbotů s vývojem SmarterChild, který byl dostupný v aplikacích pro zasílání zpráv jako byli America Online a Microsoft Network (MSN). Bylo to poprvé, kdy chatbot začal pomáhat lidem s každodenními úkoly, protože mohl získávat informace z databází o televizním programu, sportovních výsledcích, cenách akcií, zpravodajství či počasí. Pro lidi bylo velice pohodlné přistupovat k těmto informačním systémům pomocí chatbotů. [4]

Dalším výrazným pokrokem bylo vytvoření chytrých osobních hlasových asistentů zabudovaných přímo do chytrých telefonů nebo domácích reproduktorů, kteří rozuměli hlasovým příkazům, mluvili digitálními hlasy a zvládali běžné úkoly jako například monitorování domácích zařízení, kalendářů, e-mailů atd. V dnešní době je neoblíbenější Siri od Applu, Watson od International Business Machines Corporation (IBM), Google Assistant od Googlu, Cortana od Microsoftu a Alexa od Amazonu. Všichni tito virtuální asistenti mají společné to, že se připojují k internetu a na rozdíl od svých předchůdců vytvářejí rychlejší a smysluplnější odpovědi. Jejich hlavní rozdíly budou shrnuty v následujícím textu. [4]

Siri, vyvinutá společností Apple v roce 2010, byla průkopníkem v oblasti osobních asistentů. Uživatelé s ní mohou komunikovat prostřednictvím zpráv nebo hlasových příkazů. Reaguje na požadavky pomocí různých internetových služeb. I přes to, že je Siri propracovaná, má několik slabin. Vyžaduje připojení k internetu. Je sice vícejazyčná, ale mnoho jazyků nepodporuje. Hlasová navigace funguje pouze v angličtině a má potíže s přízvukem uživatele a špatně snáší okolní hluk. [4]

V roce 2011 byl společností IBM vytvořen chatbot s názvem Watson. Rozuměl lidské řeči tak dobře, že dvakrát vyhrál v kvízové soutěži. V dalších letech umožnil Watson podnikům vytvářet lepší virtuální asistenty. Kromě toho byl Watson Healt navržen tak, aby pomáhal lékařům ve zdravotnictví diagnostikovat nemoci. Hlavní nevýhodou je podporování pouze angličtiny. [4]

Google Now, vyvinutý v roce 2012, byl původně používán k poskytování informací uživateli s ohledem na denní dobu, místo a preference. Google Assistant, který byl vyvinut v roce 2016, představuje další generaci Google Now. Má propracovanější umělou inteligenci s přívětivějším konverzačním rozhraním a poskytuje uživatelům informace předpovídající jejich požadavky. Nemá však žádnou osobnost a jeho otázky mohou porušovat soukromí uživatele, protože je propojen přímo s jeho účtem Google. [4]

Microsoft v roce 2014 navrhl osobní asistentku Cortana. Rozpoznává hlasové příkazy a provádí úkoly jako je identifikace času a polohy, notifikace, odesílání e-mailů a textů, vytváření a správa seznamů, chatování, hraní her a vyhledávání informací, které uživatel požaduje. Hlavní nevýhodou Cortany je, že může spustit program, který obsahuje malware. [4]

Ve stejném roce Amazon představil Alexu, která je zabudována do zařízení pro domácí automatizaci a zábavu a vytváří tímto Internet of Things (IoT) dostupnější pro lidi. Novinkou je, že vývojáři mohou používat Alexa Skills Kit k vytváření a publikování bezplatných nebo placených dovednosti Alexy. Hlavní nevýhodou jsou bezpečnostní rizika. [4]

3.2 Přístupy strojového učení

Chatboti, kteří využívají přístupy strojového učení namísto shody vzorů, extrahují obsah z uživatelského vstupu pomocí zpracování přirozeného jazyka (NLP) a disponují schopností učit se z konverzací. Berou v úvahu celý kontext dialogu, nejen aktuální zprávu, a nevyžadují předdefinovanou odpověď pro každý možný vstup uživatele. Obvykle potřebují rozsáhlý soubor trénování, jehož nalezení může představovat zásadní problém, protože dostupné soubory dat mohou být nedostatečné. [4]

Pro implementaci těchto chatbotů se často používají neuronové sítě. Modely založené na vyhledávání v neuronové síti, ve které jsou přiřazena skóre k jednotlivým odpovědím. Skóre slouží pro identifikaci nejpravděpodobnější odpovědi. Naproti tomu generativní modely syntetizují odpověď obvykle pomocí technik hlubokého učení. [4]

3.3 Zpracování přirozeného jazyka (NLP)

Zpracování přirozeného jazyka je obor umělé inteligence, který zkoumá, jak počítačové systémy mohou interpretovat a ovládat přirozený jazyk, a to v případě, že jde o text nebo řeč. Shromažďují se informace o porozumění lidskému jazyku sloužící k vytvoření vhodných technik pro počítačové systémy pro řízení lidského jazyka a provádění různých úkolů. Většina technik NLP se spoléhá na strojové učení. Skládají se z Pochopení Přirozeného Jazyka (NLU), které rozvíjí poslání porozumět textu a Generace Přirozeného Jazyka. Tato technika má odpovědnost za generování textu.[4]

3.4 Pochopení přirozeného jazyka (NLU)

Chatboti využívají porozumění přirozenému jazyku k načtení kontextu z nestrukturovaného uživatelského vstupu v lidském jazyce a k reakci na základně záměru aktu-

álního uživatele. Tři hlavní problémy vzniklé během procesu NLU jsou mechanismy myšlení, interpretace a obecné znalosti uživatele. NLU podporuje klasifikaci záměrů a extrakci entit, přičemž bere v úvahu kontextové informace. Entity mohou být definované systémem nebo uživatelem. Kontexty jsou řetězce, které ukládají objekt, na který se uživatel odkazuje. Model klasifikace záměrů může být předem trénovaný model, který byl vytvořen manuální klasifikací shromážděných textových zpráv od uživatelů do témat. [4]

Podobně lze model extrakce entit předem natrénovat ručním přidáváním entit do textových zpráv uživatele. Z tohoto důvodu lze vytvořit anotovaný tréninkový korpus přiřazením štítku ke každému slovnímu bloku. Poté, co byly modely natrénovány, mohou automaticky klasifikovat nové textové zprávy uživatele do záměrů a extrahovat entity. [4]

3.5 Umělé neuronové sítě

Vyhledávací a generativní chatboti používají několik druhů umělých neuronových sítí. Systém převezme vstup uživatele, vypočítá jeho vektorové reprezentace, předá jej jako prvky neuronové síti a vytvoří odpověď. Proces mapování slov do vektorů se nazývá vkládání slov. Několik metod může být jednoduchých nebo může používat techniky hlubokého učení. V systémech založených na vyhledávání jsou umělé neuronové sítě trénovány pomocí vektorů vstupu a záměrů uživatele. Berou jako vstup vstupní vektor uživatele a dávají pravděpodobnost každému záměru. Klasifikace entit v uživatelském vstupu je prováděna systémy Named entity recognition (NER), které mohou využívat i techniky hlubokého učení. [4]

Zatímco generativní chatboti jsou užiteční pro zapojení osoby do neformálních konverzací v otevřené doméně, nejsou ideální pro komunikaci v uzavřené doméně, pro kterou jsou vhodní chatboti jednající na základě pravidel. Hybridní chatboti používají jednak přístup založený na získávání a dále také generativní přístup k reakci na vstup uživatele, pokud neexistuje žádná shoda s některým z pravidel. [4]

Rekurentní neuronové sítě - Vývojáři chatbotů používají Recurrent Natural Networks (RNN), aby vzali v úvahu předchozí kontext v konverzaci. V RNN nový uživatelský vstup a informace z dřívějších dat zásobují neurony. Tímto způsobem smyčka předává znalosti z jedné části sítě do druhé. Pro odkázání na předchozí informace a učení se dlouhodobé závislosti se používají sítě Long Short Term Memory (LSTM), což je konkretní typ RNN. [4]

Sequence-to-Sequence model - Typický generativní model je Sequence-to-Sequence, který generuje cílovou sekvenci pohledem na zdrojovou sekvenci. Zdrojová sekvence je vstup uživatele a cílová sekvence je odpovědí chatbota. [4]

Deep Seq2seq modely - Generativní chatboti mohou mít lepší a lidštější výkon pokud jde model více do hloubky a má více parametrů, jako v případě hlubokých modelů Seq2seq obsahujících více vrstev sítí LSTM. [4]

3.6 Trendy v automatizaci komunikace

Jak je patrné nejen z dosavadního textu, chatbot technologie není ničím úplně novým. Proč tedy tak náhlý komerční zájem? Hlavním faktorem je, že se změnil svět a zejména se změnil způsob, jakým lidé komunikují. V současné době používá mobilní telefon s podporou SMS přes 6 miliard lidí. Aplikace pro zasílání zpráv používají více než 2 miliardy lidí: 49% lidí ve věku 18-29 let a 37% lidí ve věku 30-49 let. Samotný facebook messenger má 1 miliardu uživatelů. Ukazuje se, že jsme naprosto spokojeni s komunikací prostřednictvím krátkých typizovaných interakcí. Nevadí nám, že vedeme několik asynchronních konverzací současně. Toto je právě ideální prostředí pro chatboty.[2]

3.7 Voiceboti

Voicebot je virtuální hlasový asistent, který komunikuje přirozeným hlasem. Nereaguje jen na klíčová slova, ale je poháněn umělou inteligencí. Voicebot dokáže hovory nejen přijímat, ale i iniciovat. Dokáže hromadně oslovit libovolné množství zákazníků. Hlavní výhodou je jeho časově neomezená dostupnost a fakt, že se nikdy neunaví. Veškeré požadavky řeší v reálném čase. Další jeho důležitou schopností je, že dokáže pracovat s daty systémů vlastníka a posílá získané informace. Nastavit si můžeme jak ženský, tak i mužský hlas. Ukázkou využití voicebotů může být například sběr zpětné vazby, kdy voicebot obvolá okamžitě všechna čísla ze seznamu nebo třeba vyřeší reklamaci či stížnosti zákazníků společnosti. Další příklad je náborový proces, při kterém sám kontaktuje kandidáta, ověří kompetence a domluví schůzku ve volném termínu. Samozřejmostí je i sjednání pojištění díky předem definované sadě otázek. [5]

3.7.1 Rozpoznávání hlasu

Rozpoznání hlasu je hlavní součástí hlasového virtuálního asistenta. Schopnost převodu mluveného slova do psaného textu (Speech-to-text), je důležitým nástrojem pro celé fungování tohoto systému. Čím lepší převod mluveného slova na text, tím lépe může voicebot zareagovat. Chatbot a voicebot se vlastně moc neliší, jejich hlavním rozdílem je způsob získávání vstupních informací, dle kterých se bot následně rozhoduje, co vysloví či napíše. Logika za rozpoznáním hlasu už je stejná. Princip převodu řeči na text spočívá v tom, jak systém analyzuje zvukovou nahrávku. Algoritmy na rozpoznaní hlasu jsou staré již přes deset let, dříve volili mezi spojitým a diskrétním rozpoznáváním řeči. V diskrétní je nutné dělat mezi slovy velké hlasové mezery, které značně prodlužují komunikaci a jsou pro uživatele nepřirozené. V dnešní době již většina systémů pracuje se spojitou řečí. [6]

Princip rozpoznání řeči - Při rozpoznávání řeči musí algoritmus projít několika rozsáhlými kroky. Při mluvení vytváříme hlas tím, že vzduch prochází přes rozvibrované hlasivky. První krok je digitalizace vstupního signálu. Jako další krok se odstraní nežádoucí šum v pozadí, v některých případech je možné rozdělit frekvenční pásma, která se nadále budou samostatně zpracovávat. Pásma, ve kterých člověk neslyší nebo nemluví, se oříznou. Dalším krokem je sjednotit rychlost mluvení s daty s nimiž bu-

deme tyto digitalizované nahrávky dále porovnávat. Tento signál je rozdělen na velmi malé části, které mají délku setin až tisícin sekundy. Algoritmus porovná vzorová data se známými fóny ve vybraném jazyce. Fón je specifický zvuk, který vyjadřuje hlásku v zvoleném jazyce. Následující kroky nám již připadají snadné, stačí porovnat jednotlivé vzorky s fonémy daného jazyka a vybrat ten nejvhodnější. Opak je ale pravdou, protože realita je mnohem složitější, i když princip zůstává podobný. Algoritmus se snaží prozkoumat každý foném v souvislosti sousedních fonémů na základě velkého množství statistických dat, která jsou založená na velké databázi známých slov ve vybraném jazyce. Program nám tedy díky tomuto principu určí slovo, které se nejpravděpodobněji rovná tomu, co uživatel vyslovil. [6]

3.7.2 Přehled dostupných možností pro převod jazyku na text

V této podkapitole si probereme dostupné možnosti světových gigantů, které můžeme využít na rozpoznání jazyka na text. [6]

Google Cloud Speech API - Google Cloud Speech API umí rozpoznat více jak 80 jazyků, v kterých je i čeština. Zvuk zpracovává přímo buď z mikrofonu nebo ze zvukového souboru. K převodu je použit Deep Learning neuronové sítě, který se chlubí obrovskou přesností převodu zvuku na text. Rozpoznaný text vrací okamžitě a dobře počítá i s nežádoucími zvuky na pozadí. Tato platforma nabízí 60 minut rozpoznání zvuku měsíčně zdarma, dalších 15 sekund stojí přibližně 0,006 dolaru. Díky velkému množství dat a lidí, které na tomto API pracují, se rychle zdokonaluje. Například od roku 2017 již umí vracet vyslovené číslovky číslem a ne jen slovem. [6]

Bing speech API - Stejně jako Google, tak i Microsoft pracuje na rozpoznávání řeči, ale jeho API zaostává hlavně co se počtu jazyků týče. Nyní jich nabízí 29. Jejich API je přístupné z Androidu, iOS a Windows, nechybí ani zařízení třetích stran IoT. Rozpoznání řeči začne fungovat až po registraci na Microsoft Azure, díky které získáme klíč, který je k tomu nezbytný. Zdarma je pak prvních 5000 přenosů na jeden klíč. [6]

IBM Watson Developer Cloud - Speech to Text - IBM také umožňuje převod řeči na text, ale jen v 9 jazycích. API funguje s jakýmkoliv zařízením připojeným k internetu. Nicméně, když porovnáme řešení od Microsoftu nebo od Googlu, tak Watson výrazně zaostává. U IBM mají prvních 1000 minut zdarma, poté každá další minuta stojí 0.02 dolaru. [6]

3.8 Platformy pro vývoj chatbotů

Následující část textu se bude blíže zaměřovat na nejlepší bot platformy, které je možné použít pro výstavbu chatbotů. Tyto platformy jsou nástroje k vytváření a nasazení interaktivních chatbotů. Poskytují vývojové nástroje, jako jsou frameworky a sady nástrojů API, pro vytváření různých botů. Tito boti jsou škálovatelní, aby sloužili na více komunikačních platformách a na více zařízeních současně. Obsahují také funkce údržby pro řešení problému po jejich nasazení. Aby si produkt mohl říkat bot platforma, musí splňovat následující: Poskytovat možnost nasazení botů. Mít framework pro vývoj botů. Umožňovat uživatelům definovat chování a reakce

programu. Obsahovat uživatelské nástroje pro provádění údržby a aktualizaci publikovaných botů. [7]

Qualified je platforma pro live chat, chatbot a konverzační marketing pro společnosti, které používají Salesforce. Podnikové prodejní a marketingové týmy předních značek Business-to-Business (B2B) důvěřují společnosti Qualified, že rozšíří svůj prodejní kanál tím, že přemění své největší nevyužité marketingové aktivum, což jsou jejich webové stránky, na prodejní stroj prostřednictvím konverzací v reálném čase s hlavními návštěvníky a jejich cílovými účty.

ManyChat je platforma číslo 1 pro chat ve světovém marketingu. Už od roku 2016 začaly ManyChat využívat stovky tisíc firem po celém světě k automatizaci jejich Facebook Messenger Experience pro jejich Facebookové stránky.

IBM Watson Assistant je virtuální agent s umělou inteligencí, který zákazníkům poskytuje rychlé, konzistentní a přesné odpovědi na jakákoliv platformě pro zasílání zpráv, aplikaci, zařízení nebo kanálu. Pomocí umělé inteligence a zpracování přirozeného jazyka se Watson Assistant učí z konverzací se zákazníky, zlepšuje svou schopnost řešit problémy hned napoprvé a zároveň odstraňuje frustraci z dlouhého čekání, zdlouhavého vyhledávání a neužitečných chatbotů.

Haptik pomáhá podnikům vytvářet chatboty podporované umělou inteligencí pomocí modelů průmyslové umělé inteligence s inteligencí specifickou pro doménu, aby přesně porozuměly záměru uživatele a poskytovaly kontextové odpovědi řešící dotazy uživatelů od začátku do konce.

Yellow.ai je přední světová platforma pro konverzaci Customer Experience (CX), uznávaná skupinami Gartner, International Data Corporation (IDC) a G2 za lídra. Yellow.ai spojuje to nejlepší z umělé inteligence a lidské inteligence k automatizaci zákaznické zkušenosti pro velké podniky a nabízí rozlišení a informace v reálném čase a na vyžádání. Společnost Yellow.ai důvěřuje více než 700 globálních společností.

Verloop.io je přední světová platforma pro automatizaci zákaznické podpory, která podnikům umožňuje poskytovat svým zákazníkům skvělou podporu napříč kanály, včetně Voice, WhatsApp, Instagramu, webu, In-App a dalších. Verloop.io pomáhá značkám bezpečně, bez námahy a přesně škálovat jejich zákaznickou podporu a používá ho více než 5 000 značek po celém světě, včetně Decathlon, Cleartrip, Dar Al Arkan, Fetchr, ADIB, Nykaa, Lido Learning, DSP Mutual Fund, Rentomojo, Scripbox a další.

Microsoft Azure Bot Service umožňuje vývojářům vytvářet konverzační rozhraní na více kanálech, zatímco Language Understanding (LUIS) pomáhá vývojářům vytvářet přizpůsobené přirozené interakce na jakékoli platformě pro jakýkoli typ aplikace, včetně robotů. Následující text bude více věnován řešení od společnosti Microsoft, protože organizace, pro kterou je určena aplikace vyvíjená v rámci této práce, právě toto řešení využívá.

3.9 Microsoft Azure Bot Service

Organizace, pro kterou je aplikace určena, používá řešení od Microsoftu. Microsoft Azure Bot service je součást větší platformy Microsoft Azure, což je sada cloudových řešení, které nabízejí mnoho různých služeb pro firmy. Microsoft Azure Bot je pro

vývojáře cestou k urychlení tvorby botů. Služba umožňuje jak lokální, tak cloudový vývoj botů prostřednictvím integrovaného prostředí a zároveň poskytuje šablony pro další funkce. Microsoft také zahrnuje analytické nástroje pro získávání metrik, například využití API nebo počet aktivních uživatelů v určitý čas. Velká výhoda využití tohoto řešení je i integrace pro různé komunikační platformy jako je Skype, Teams, Slack nebo Facebook. [8]

3.9.1 Kognitivní Služby Microsoft

Další velkou výhodou jsou služby spojené s chatbotem. Ty jsou rozděleny do pěti hlavních schopností: Řeč, Znalosti, Vidění, Vyhledávání, Jazyk. [8]

Řeč - Tato sada schopností obsahuje všechny funkce související s hlasem a rozhraní Application Programming Interface (API) pro převod řeči na text, které převádí mluvená slova na text. Jsou k dispozici dvě verze, buď standardní, nebo nervová verze. Nervová verze je vyvinutější a přirozenější způsob vytváření syntetické řeči způsobující nižší poslechovou únavu. [8]

Znalosti - Tato sada schopností popisuje, jak lze botovi poskytnout znalostní bázi a jak jí vytvořit. Základ jsou záznamy v jeho znalostní databázi. Snaží se podat tu nejlepší odpověď a dodává k tomu také skóre spolehlivosti. Skóre je ukazatelem jistoty, že odpověď na otázku je správná. [8]

Vidění - Tato sada schopností poskytuje rozhraní API, které umožňuje vývojáři přístup k analýze obrázku a k určení jejich obsahu, stejně jako k detekci a převodu na text. API také poskytuje skóre spolehlivosti toho, zda daný obrázek představuje obsah pro dospělé, což umožňuje vývojáři zkontrolovat, co je považováno za reprezentativní. Mezi další schopnosti patří rozpoznávání obličejů, jejich detekce, určení emocí, extrahování psaného textu nebo mluveného slova osob. [8]

Vyhledávání - Tato sada schopností využívá vyhledávač Bing společnosti Microsoft a umožňuje funkce jako je vyhledávání zpráv, vyhledávání videa, vyhledávání na webu, automatické návrhy, vlastní vyhledávání, vyhledávání entit, vyhledávání obrázků a vizuální vyhledávání. [8]

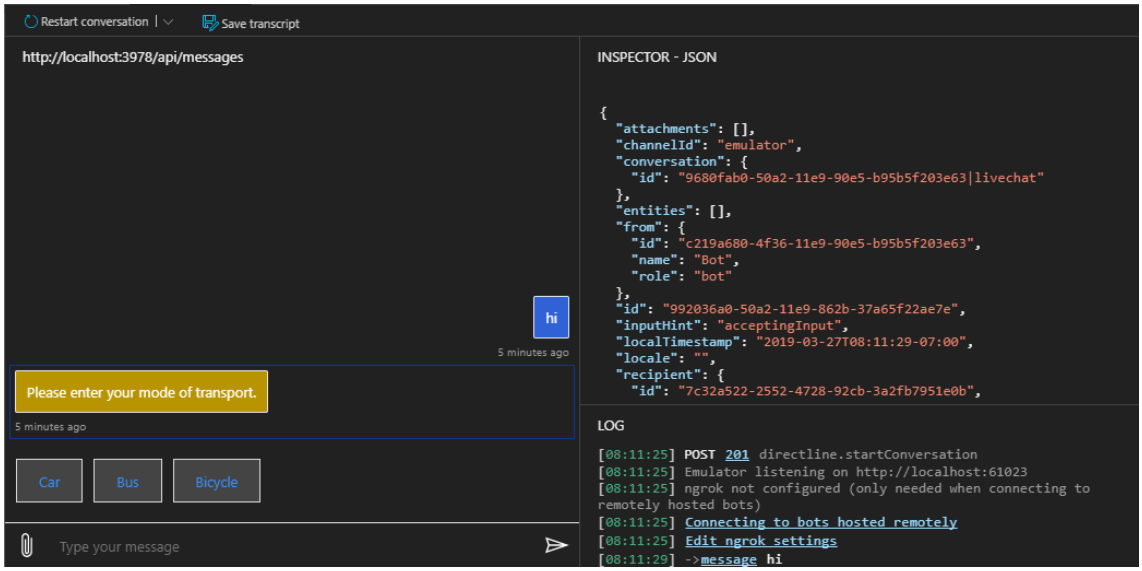
Jazyk - Tato sada schopností zpracovává analýzu vstupu a určuje vhodné reakce. Text Analytics API umožňuje vyhodnocení sentimentu (pozitivního nebo negativního) daného textu, extrahuje klíčová slova nebo fráze a určuje, který jazyk je použit. [8]

Nyní byly popsány možnosti tvorby botů. Dále je však ještě třeba věnovat pozornost aktuálnímu tématu, jímž je automatické testování odpovědí bota. Bylo otázkou, jak to uchopit. K dispozici byl *transcript*, neboli záznam konverzace, pod unikátním identifikátorem, přičemž každá konverzace má svůj. Nyní už jen otestovat, zda chatbot správně odpovídá. V další kapitole budou rozebrány možnosti a cesta, která vedla k inovativní vlastní implementaci aplikace.

3.9.2 Bot framework emulator

Tento emulátor slouží pro přímou komunikaci s chatbotem nasazeným buď lokálně nebo na serveru. Nejprve se zadá adresa serveru, kde bot běží, tím se k němu připojíme. Chatbot se nám spustí, v levém okně s ním můžeme chatovat a v pravém

vidíme detailní JSON, který reprezentuje příchozí zprávy. Zprávy nemusí obsahovat jenom text, mohou to být obrázky, ankety, galerie, tlačítka atd. V logu vidíme celou konverzaci. Pro restartování konverzace slouží tlačítko vpravo nahoře. Pro pozdější debug bota si můžeme uložit celou konverzaci tlačítkem *save transcript*. Na Obrázku 1 můžeme vidět, jak vypadá Bot framework emulator. V levé části obrázku vidíme konverzaci. V pravé dolní části se nachází jednotlivé požadavky a odpovědi. V pravé horní sekci můžeme vidět obsah komunikace jednotlivých zpráv. [9]



Obrázek 1: Bot framework emulator, zdroj: autor

3.10 Slabé stránky a hrozby chatbotů

Kromě jejich nesporných předností nejsou chatboti bez nevýhod a hrozeb. Zákazníci dobře znají komunikaci se společnostmi pomocí telefonů, e-mailů, newsletterů nebo webových stránek, zatímco aplikace messenger používají především pro svou soukromou komunikaci. Nový způsob interakce společností se zákazníky je prostřednictvím chatbotů v messengerech nebo samotných aplikacích. Není dobré všechny zákazníky hned tlačit do nového způsobu komunikace. V procesu transformace by měly být podporovány i tradiční platformy a spotřebitelé by měli být nenásilně povzbuzováni k používání nově vznikajících technologií a nástrojů. [4]

Zabezpečení dat je významnou rizikovou oblastí jak pro poskytovatele, tak pro uživatele. Společnosti jsou odpovědné za kvalitní ochranu a za vhodné nakládání s daty zákazníků, pokud poskytují samostatnou aplikaci chatbota. Protože však společnosti zpřístupňují svého chatbota na webech třetích stran, často se jim doručují data. Soukromí a bezpečnost dat musí být zachována zejména pokud jde o autentizační a platební systémy, kde se přistupuje k důvěrným, citlivým nebo finančním údajům. Zákazníci si navíc musí být vědomi toho, že když s nimi společnosti komunikují, shromažďují, ukládají a používají osobní údaje pro komerční a marketingové účely. [4]

3.10.1 Neporozumění záměru uživatele

Navzdory svému velkolepému vývoji chatboti často nedovedou rozpoznat záměr svého partnera. Je to asi jejich nejvýznamnější slabina, která se projevuje poměrně často. Pro uživatele je velmi frustrující, když chatbot nechápe jeho záměr. V závislosti na rozsahu chatbota se tato zranitelnost může ukázat jako škodlivá pro majitele chatbota. Například frustrující konverzace s chatbotem, sloužícím jako asistent prodeje, může zákazníka odradit. [4]

3.10.2 Toxický obsah v uživatelských vstupech chatbota

Toxický obsah může být vážnou nevýhodou pro poskytovatele a uživatele chatbotů. Tay, chatbot, který se zlepšuje prostřednictvím konverzace, vyvolal obrovskou kontroverzi, když byl napaden internetovými trolly na Twitteru. Po 16 hodinách svého využívání začal uživatelům posílat vysoce urážlivé tweety. Proto, i když je učení ze zkušenosti zdůrazňováno jako úspěšná strategie, chatbot musí mít ochranu, aby se vyhnul jejímu zneužití. [4]

3.10.3 Podvádění vůči chatbotům

V některých případech, v nichž se používají chatboti, je detekce podvodu kritická. Charakteristiky, které činí interakce chatbotů lidštějšími, vyvolávají nežádoucí strategické chování lidských podvodníků pro skrytí svého podvodu. Zjistilo se, že náznaky podvodu se liší v závislosti na konverzačních schopnostech chatbotů. Tyto zlepšené konverzační dovednosti přiměly uživatele k zapojení do strategických aktivit, které jsou kontraproduktivní pro odhalování podvodů. Chatboty podobné lidem mohou být neefektivní v případě, že je potřeba odhalovat, kdy jednotlivci lžou. [4]

3.10.4 Další faktory selhání

Dlouhé odpovědi, kde se zásadní informace skládají z jedné nebo dvou vět, které se skrývají mezi několika dalšími, mohou uživatele odradit od konverzace. Upřednostnit by se měla krátká a jasná sdělení. Uživatelské chyby v pravopisu mohou také znamenat obtíže při klasifikaci záměru. Vhodným řešením je mít mechanismus kontroly pravopisu jako krok předběžného zpracování vstupu od uživatele. [4]

Další neporozumění záměru může být způsobeno nestandardním vstupem uživatele. Jedná se například o nesprávně použité fráze, nevhodnou intonaci, špatnou výslovnost, užívání sarkasmu či humoru, o poruchy řeči, používání slangu či o nejrůznější syntaktické chyby a další podobné obtíže. Nedostatek osobnostních charakteristik na straně chatbota může také snadno odradit uživatele od dialogu. Toto riziko lze snížit tím, že chatbotovi dáte jméno a avatara. [4]

V jiných případech může být zklamání uživatele způsobeno nedostatkem jasné strategie chatbota a následným neefektivním vedením uživatele směrem ke komunikačním cílům. Někdy je chatbot navržen tak, aby sloužil konkrétnímu účelu a poté je upraven tak, aby vyhovoval další potřebě. V tomto případě bude ale mít dvojí osobnost, která vyvolá negativní pocity. [4]

3.11 Příklady chatbotů dle využití

Tato kapitola se bude věnovat oborům, v nichž mají chatboti největší potenciál. Využití najdeme hlavně v oblastech vzdělávání, zákaznických služeb nebo ve zdravotnictví. V neposlední řadě nesmíme zapomenout ani na robotiku a průmyslové použití.

3.11.1 Vzdělávací prostředí

Rostoucí poptávka po vzdělávání vede k vysoké konkurenci ve vysokoškolských institucích. Jedním z kritických důvodů sníženého zájmu o edukační činnosti a vysoké míry předčasného ukončování školní docházky je skutečnost, že když počet studentů roste, pomoc, kterou studenti dostávají od svého učitele, se snižuje. Chatboti mají schopnost poskytovat vzdělávací obsah a osobní asistenci, s kterou podporují další e-learningové edukační aktivity. [4]

Chatboti pro podporu učení mohou uchovat informace opakováním starých lekcí, když jejich obsah studentům chybí. Informace také shromažďují během kurzu, což pomáhá zlepšit proces učení a výuky. Studenti mají studium usnadněno, protože chatboti mohou odpovídat na otázky týkající se vzdělávacího obsahu. Chatbot může také pomoci studentům se školskou administrativou, jako je zápis do kurzu, harmonogram zkoušek, známky a další podrobnosti související se studiem, takže tlak na katedry univerzit se výrazně sníží. Ve výzkumu, prováděném univerzitou Informačních technologií ve Vietnamu v roce 2018, rostl počet studentů účastnících se univerzitního kurzu, protože studentům pomohl s registrací chatbot. [4]

Velký potenciál se skrývá taky ve výuce cizích jazyků. Výzkumy ukázaly, že studenti preferují sebevzdělávání pomocí chatbotů více a častěji než edukaci vedenou lidmi, protože se cítí sebevědoměji a mohou je kdykoli použít. V roce 2004 byl představen chatbot pro chatování se studenty angličtiny, který využívá zjednodušující přístup k logickému uvažování a vyvozování především prostřednictvím syntaktické a sémantické analýzy. Vytváří odpověď, která bere v úvahu kontext dialogu, znalosti a osobnost uživatele a znalost zdravého rozumu a zkušenosti z vyvozování. [4]

3.11.2 Služby zákazníkům

Rozvoj nových technologií způsobil, že lidé vzájemně komunikují jinak a stejně tak je to i se společnostmi. Elektronický obchod se vyvíjel a zcela změnil způsob, jakým společnosti prodávají své produkty. I v této oblasti však existují obtíže snižující kvalitu zákaznických služeb. Zejména v živých chatech může být čekací doba na odpověď zaměstnance příslušné společnosti dlouhá a odpovědi nemusí být vždy relevantní. [4]

Mnoho společností využívá chatboty k podpoře zákazníků. Péče o ně je k dispozici 24 hodin denně, což umožňuje spotřebitelům zveřejnit svůj požadavek bez ohledu na standardní provozní dobu, což zvyšuje spokojenost uživatelů. Jako příklad je možno uvést chatbota, který pomáhá s rozhodováním zákazníků, který produkt je pro ně nejvhodnější či další, který odpovídá na často kladené otázky. [4]

3.11.3 Zdravotnictví

Ve zdravotnictví jsou chatboti navrženi tak, aby pacientům poskytovali přizpůsobené informace o zdraví a terapii, produkty a služby související s určitými onemocněními a nabízeli diagnostiku a navrhovali léčbu na základě symptomů pacienta. Reální chatboti jsou využiti například k tomu, aby pomáhali zvyšovat informovanost lidí o rakovině, nebo se starají o emocionální zdraví svých uživatelů. Jiný zase připomíná pacientům, aby si vzali své léky a v neposlední řadě bylo nasazeno mnoho chatbotů, kteří poskytovali informace během pandemie COVID-19. [4]

Mezi výhody používání chatbotů pro zdravotní péči patří povzbuzování lékařského rozhodování a podpora lékařů, zlepšení fyzického cvičení, podpora kognitivně-behaviorální terapie a somatických poruch. Chatboti poskytují účinnou zdravotní léčbu s přesností stejnou jako člověk - lékař. [4]

V některých případech mohou být chatboti ke splnění přání pacientů vhodnější než lékaři - lidé, protože nejsou vůči pacientům zaujatí. A též ani pacienti nejsou vůči chatbotům zaujatí kvůli pohlaví, věku nebo rase. Chatboti se také nevyčerpají ani neonemocní; jsou nákladově efektivní a mohou pracovat nepřetržitě po celý den, což je užitečné zejména pro lidi, kteří mohou mít zdravotní problémy mimo pracovní dobu jejich lékaře. Mohou také komunikovat v různých jazycích, aby pomohli reagovat na specifické potřeby pacientů. [4]

3.11.4 Robotika

Nejdůležitější oblastí výzkumu chatbotů je rozhraní přirozeného jazyka, které je kritickou oblastí i pro fyzické roboty. Je to důvod proč právě u fyzických robotů najdeme velký počet různých aplikací přirozeného jazyka. I odborná literatura je plná příkladů zejména z oblasti vzdělávání, kde jsou roboti využiti. Není tedy výjimkou humanoid vyprávějící příběhy studentům nebo vysledované snížení úzkosti u studentů, které učí cizí jazyk fyzický robot působící jak lektor. V jiném případě zase fyzický robot třeba pomáhá batolatům při učení nových slov. [4]

3.11.5 Příklady průmyslového použití

V současné fázi technologického vývoje jsou chatboty již široce používány mnoha společnostmi a organizacemi.Je velice zajímavé podívat se na některé příklady z praxe. V bankovním sektoru chatboti mluví se zákazníky a mimo jiné poskytují informace o zůstatcích na jejich účtech, usnadňují platby z účtů, navrhují způsoby, jak ušetřit zdroje a pomáhají aktivovat karty. Zároveň pomáhají bance se sběrem zpětné vazby od zákazníků. [4]

IKEA, přední značka nábytku, spustila takzvaného ORC chatbota. Módní značka Zalando využívá chatbota pro sledování objednávek. PVR Cinemas, velký řetězec kin v Indii, používá k rezervaci vstupenek chatbota. Americká National Railroad Passenger Corporation používá pro rezervaci jízdenek chatbota Julie. Světová zdravotnická organizace používá k poskytování informací souvisejících s koronavirem chatbota zvaného World Health Organization (WHO) Health Alert. [4]

3.12 Užitečné nástroje pro chatboty

V této kapitole si popíšeme užitečné nástroje, které se hodí pro provozování botů.

3.12.1 Botanalytics

Botanalytics je webová aplikace, která jak už z názvu napoví, slouží pro analýzu bota. Jsou to statistiky založené na umělé inteligenci. Podporují všechny známé platformy botů jako Facebook Messenger, náš používaný Bot Framework Service, Google Asistant a mnoho dalších. Aplikace funguje tak, že si do ní napojíme našeho existujícího bota, a ona si začne ukládat všechny konverzace našich zákazníků. Na základě těchto informací nám dokáže zobrazit velmi užitečné informace. Dokáže sledovat konverzní poměry, toky konverzací a různé statistiky, jako jsou délka průměrné konverzace, počet zpráv za určité časové období, lokality uživatelů a další. Díky A/B testování s botanalytics se dokáží vytvořit lepší konverzační toky. Na Obrázku 2 můžeme vidět ukázku toku konverzací. [10]

3.12.2 Chatbottest

Další užitečný nástroj s názvem Chatbottest se zaměřuje na kvalitu vašeho chatbota. Na základě heuristického vyhodnocení a zkušeností této platformy byla vytvořena kompletní open source příručka s pokyny a otázkami, která otestuje jakýkoliv design chatbota. Tento nástroj dokáže najít problémy v interakci člověk - chatbot rozdělené do sedmi kategorií. [11]

Osobnost - Má chatbot jasný hlas a tón, který se hodí k uživatelům a probíhající konverzaci?

Onboarding - Chápou uživatelé, o čem je chatbot a jak s ním komunikovat od samého začátku?

Porozumění - Rozumí chatbot požadavkům, smalltalku, idiomům, emotikonům, apod.? Čemu je chatbot schopen porozumět?

Odpovědi - Jaké prvky posílá chatbot a jak dobře to dělá? Jsou relevantní pro daný okamžik a kontext?

Navigace - Jak snadné je projít konverzací s chatbotem? Cítíte se někdy ztraceni, když mluvíte s chatbotem?

Správa chyb - Jak dobře si chatbot poradí se všemi chybami, které se stanou? Dokáže se z nich vzpamatovat?

Inteligence - Má chatbot nějakou inteligenci? Dokáže si věci zapamatovat? Používá a řídí kontext jako člověk?

3.12.3 Dimon

Dimon je nástroj, který umožňuje vlastníkům chatbotů identifikovat a opravit problémy v konverzacích. Tato platforma nabízí automatické testování, které nám zkrátí testování z hodin na pár minut. Dále umí generovat scénáře, aby dosáhl co nejširšího pokrytí testy. Také nabízí sledování bota v reálném čase a zasílání upozornění při chybách v chatbotovi. Tato platforma by se nabízela k použití místo aplikace, která

Obrázek 2: Ukázka Botanalytics, zdroj: autor

se vyvíjí v této diplomové práci, ale bohužel nepodporuje Microsoft Bot Framework. Nyní jsou registrace do aplikace uzavřeny, takže nebylo možné si jí vyzkoušet. [12]

3.13 Testování chatbotů

Dnešní platformy pro tvorbu botů neposkytují v oblasti testování botů nikterak velkou podporu a to i přesto, že je testování pro zajištění kvality botů nezbytné. Většinou nabízejí jen konzoly, kde mohou vývojáři ručně otestovat, zda chatbot správně reaguje na vstupy NLP. Ruční testování sice pomáhá během vývoje, ale správný softwarový proces vyžaduje systematické a axiomatizovatelné testovací mechanismy. [13]

Manuální nebo automatické testování? Tohle není otázka buď a nebo. Zdroj pravdy nemusí být pouze jeden. Klíčem k vysoké kvalitě softwaru je právě kombinace testovacích technik. Ujistěme se, že rozumíme rozdílu mezi ověřením a validací. Ověření řeší problém: Stavíme to správně? Validace řeší problém: Stavíme správnou věc? V manuálním testování řešíme, zda chatbot splňuje požadavky. V automatizovaném testování nás zajímá, zda chatbot funguje tak, jak má. [14]

3.13.1 Manuální testování

Zabýváme-li se manuálním, tedy ručním, testováním, nelze jít v projektech chatbotů žádnou zkrácenou cestou. Manuální testování je samozřejmě nedílnou součástí dobré testovací strategie. Následně budou popsány hlavní techniky ručního testování. [14]

Davové testování - Nejefektivnější řešení, jak přivést chatbota až na hranici jeho limitů, je podrobit ho motivovanému a zkušenému zástupu lidských uživatelů. Pokud bude davové testování provedeno v několika kolech po sobě, přibude mnoho nových výroků, díky nimž lze chatbota vycvičit. Jestliže je chatbot spuštěn pro produkci, pak pravděpodobně neexistují v podstatě žádné technické námitky proti naplánování právě davového testování. Obvykle platí, že jsou uživatelé placeni za zjištěnou chybu. Je však třeba velice přesně definovat, co se klasifikuje jako chyba a co ne. Nedorozumění může být chyba, odpověď „promiň, nerozumím“ však chybou být s největší pravděpodobností nemusí a stává se pouze známkou toho, že chatbot potřebuje další školení. Taktéž zástup lidí, kteří se testování zúčastní, musí být volen vhodně a v závislosti na stavu projektu. Důležitými kritérii výběru by měly být jazyk a dialekt. [14]

Průzkumné testování - Platí zde stejný princip, jako pro výše uvedené davové testování. Není třeba definovat žádné přesné požadavky či testovací případy. Dav uživatelů udělá svou práci sám. [14]

A/B testování - I když je chatbot již aktivní, je stále třeba investovat do něho mnoho dalšího úsilí. Zejména zkoušet nejrůznější výroky a měřit změny v chování uživatelů. [14]

3.13.2 Automatizované testování

Automatizované testování může být časově mnohem bezpečnější, ale zároveň je to nekonečný vedlejší projekt. Automatizace testování je v poměru k její hodnotě tak

drahá, že by se měla používat střídmě. Lze ji považovat za způsob, jak zachytit a implementovat požadavky. Obecně platí, že čím větší je systém a čím je komplexnější, tím vyšší je Return on Investment (ROI) v automatizaci testování. Pro chatboty toto tvrzení platí jen částečně. [14]

Je vhodné si uvědomit, že je třeba počítat se zvýšenou pracovní zátěží vývojářů oproti testovacímu oddělení, protože automatizované testování vyžaduje velké úsilí při vývoji softwaru. Právě úkolem vývojářů softwaru je totiž převést testovací případy do spustitelného kódu. Automatizace testování se tedy obvykle týká oddělení vývoje softwaru a ne oddělení testování. [14]

Technické požadavky Oblastí, na kterou je dále třeba také myslet, jsou technické požadavky. Při předběžné práci se pomocí technických požadavků nahromaděné příběhy plní více či méně formalizovaným jazykem, užívaným k popisu požadovaných vlastností. Mezi technickými požadavky, testovacím oddělením a vývojem softwaru existuje velký prostor pro nedorozumění. Za účelem snížení nedorozumění je vhodné využít například Gherkin. Gherkin je obchodně čitelný jazyk specifický pro doménu, který umožňuje popsat chování softwaru bez podrobností o tom, jak je toto chování implementováno. Gherkin je tedy spojení mezi technickými požadavky, testovacím oddělením a vývojem softwaru s popisem sad funkcí. Při popisu funkcí pomocí Gherkin lze samotný popis použít jako dokumentaci požadavků i jako výchozí bod pro vývoj (systémové testy, integrační testy, unit testy atd.) [14]

Gherkin definuje velmi jednoduchou syntaxi s klíčovými slovy jako „Given“, „When“, „Then“. Na následujícím Ukázce kódu č. 1 je popsána funkce kávovaru:

```
Feature: Serve coffee
  In order to earn money
  Customers should be able to
  buy coffee at all times

  Scenario: Buy last coffee
    Given there are 1 coffees left in the machine
    And I have deposited 1 dollar
    When I press the coffee button
    Then I should be served a~coffee
```

Ukázka kódu 1: Jazyk Gherkin

Použití Gherkin minimalizujeme riziko neporozumění požadavkům pomocí formalizovaného jazyka. Prostřednictvím automatizace testování lze jistě dojít velkého posunu vpřed a obecně i snížit námahu mnoha zúčastněných stran. [14]

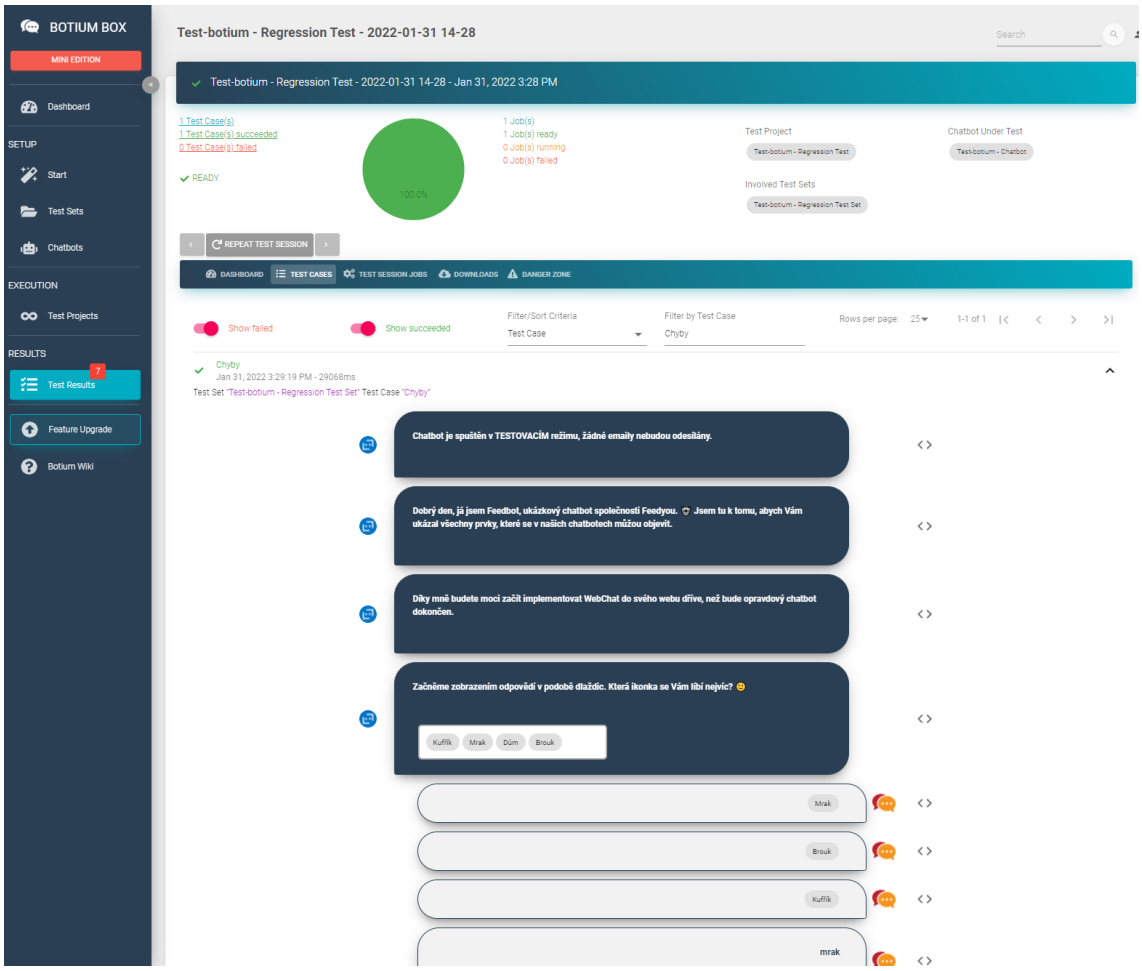
Přínos - Pro spuštění testovacích případů je sice zapotřebí vývojáře, který napíše kód pro popisy v tomto jazyce, ovšem pro testování chatbotů je možné tento krok přeskočit. [14]

Superschopnosti pro testery - Testovací případ je konverzace, kterou by chatbot měl být schopen sledovat. Testovací případy pro chatbota může sestavit každý, kdo je schopen konverzovat s chatbotem. Výsledkem je přepis konverzace. Následně je vhodné najít nástroj, který takto nahrané konverzace vezme a automaticky je paralelně spustí během několika sekund proti chatbotovi. Tento nástroj existuje a jmenuje se Botium. Zde se dostáváme k hlavní myšlence a výhodě, která plyne i z toho, co

bylo již uvedeno výše. Zatímco u většiny projektů ve vývoji softwaru obvykle testovací oddělení prostě na zajištění automatizace testování na vývojářích závisí, u chatbotů si vlastně může automatizaci testování provádět přímo samo testovací oddělení. [14]

Botium úspěšně automatizuje proces testování chatbotů. Umožňuje syntetizovat počáteční sadu testovacích případů odvozených z tréninkových frází chatbota. Vygenerované testovací případy však berou v úvahu pouze základní toky konverzace a je třeba je rozšířit ručně. Cílem je tento manuální proces co nejvíce zautomatizovat. [13]

Botium je sada open-source komponent pro automatizované testování chatbotů. Tento nástroj zajišťuje, aby chatbot vykonával to, co vykonávat skutečně má. S testovaným chatbotem komunikuje přes konektory. Ty jsou k dispozici pro mnoho platforem chatbotů (jako DIalogFlow, Watson, Lex nebo Microsoft Bot Service) a lze přidávat další. Jejich hlavními funkcemi jsou testování toku konverzace, NLP modelu, End-to-end (E2E) založené na Selenium a Appium. Mimo funkcionální testy řeší také testy zátěžové, bezpečnostní a General Data Protection Regulation (GDPR). Samozřejmostí jsou integrace Continuous Integration (CI)/Continuous Deployment (CD) se všemi běžnými produkty jako jsou Jenkins, Bamboo, Azure DevOps Pipelines, IBM Toolchain a jiné. Na Obrázku 3 můžeme vidět aplikaci Botium. [15]

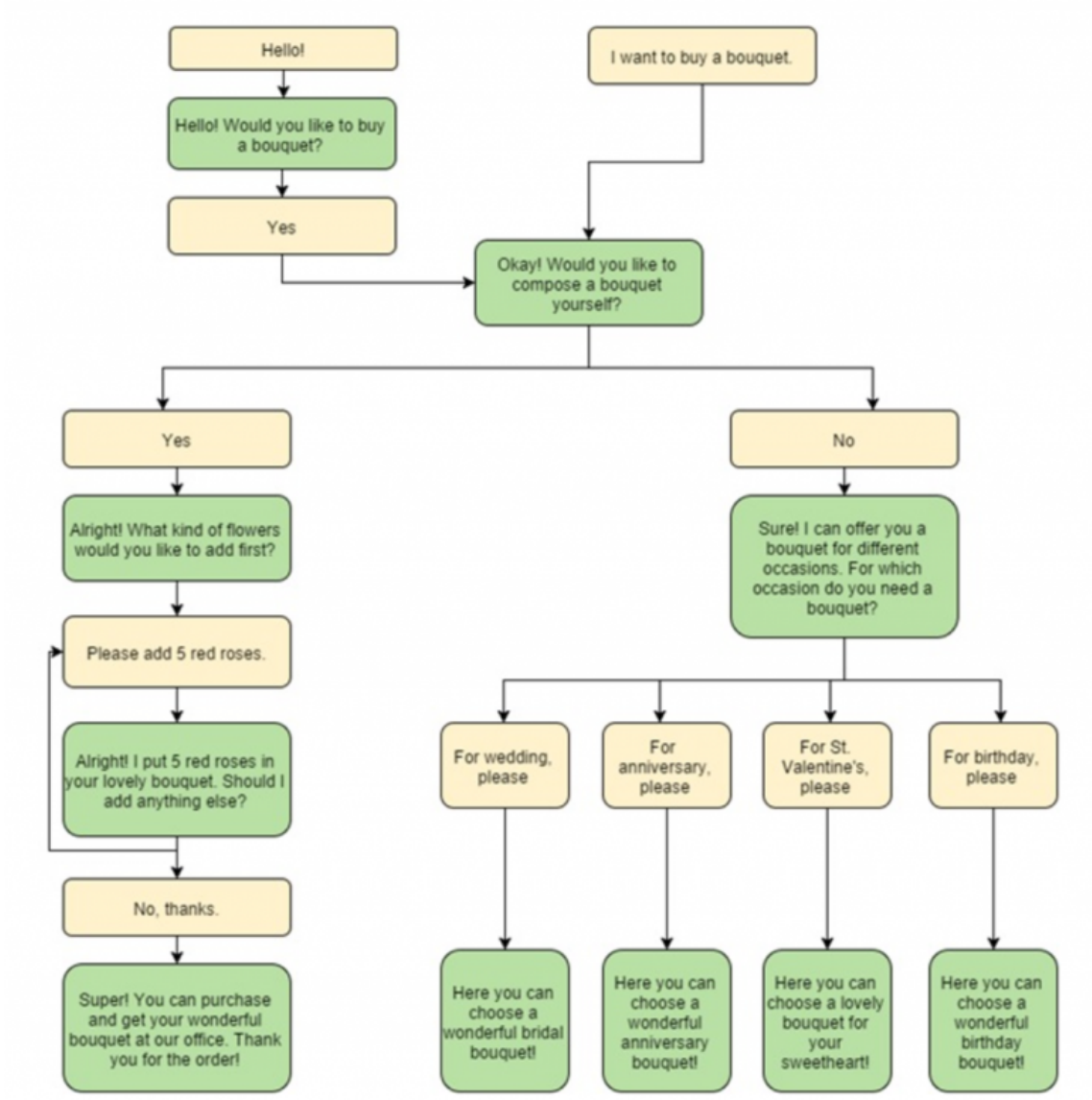


Obrázek 3: Ukázka Botium box, zdroj: autor

Testování chatbota je skutečně velmi těžké, a to i při použití správných nástrojů. V následujícím textu proto bude popsáno hledání testovacích scénářů a vše, co s tím souvisí. [1]

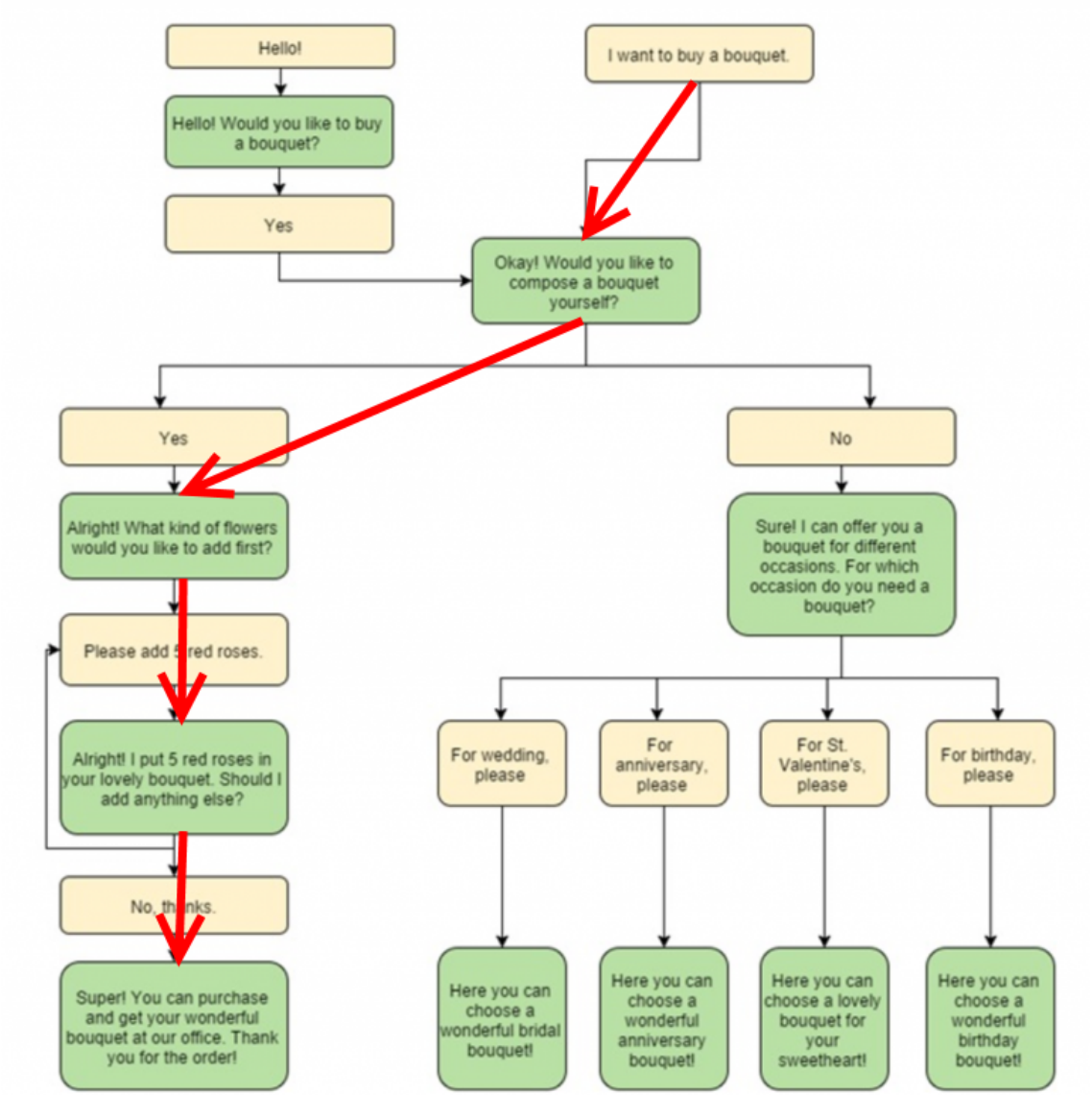
3.13.3 Hledání testovacích scénářů

Vizualizace konverzačního uživatelského rozhraní - Chatbot může sloužit jen jednomu specifickému účelu, ale může sloužit i více účelům a tedy ne pouze jednomu konkrétnímu. Pro takového víceúčelového chatbota je složité extrahovat tok konverzace (příklad Microsoft Zo). Tok konverzace lze vizualizovat ve vývojovém diagramu jak můžeme vidět na Obrázku 4. [1]



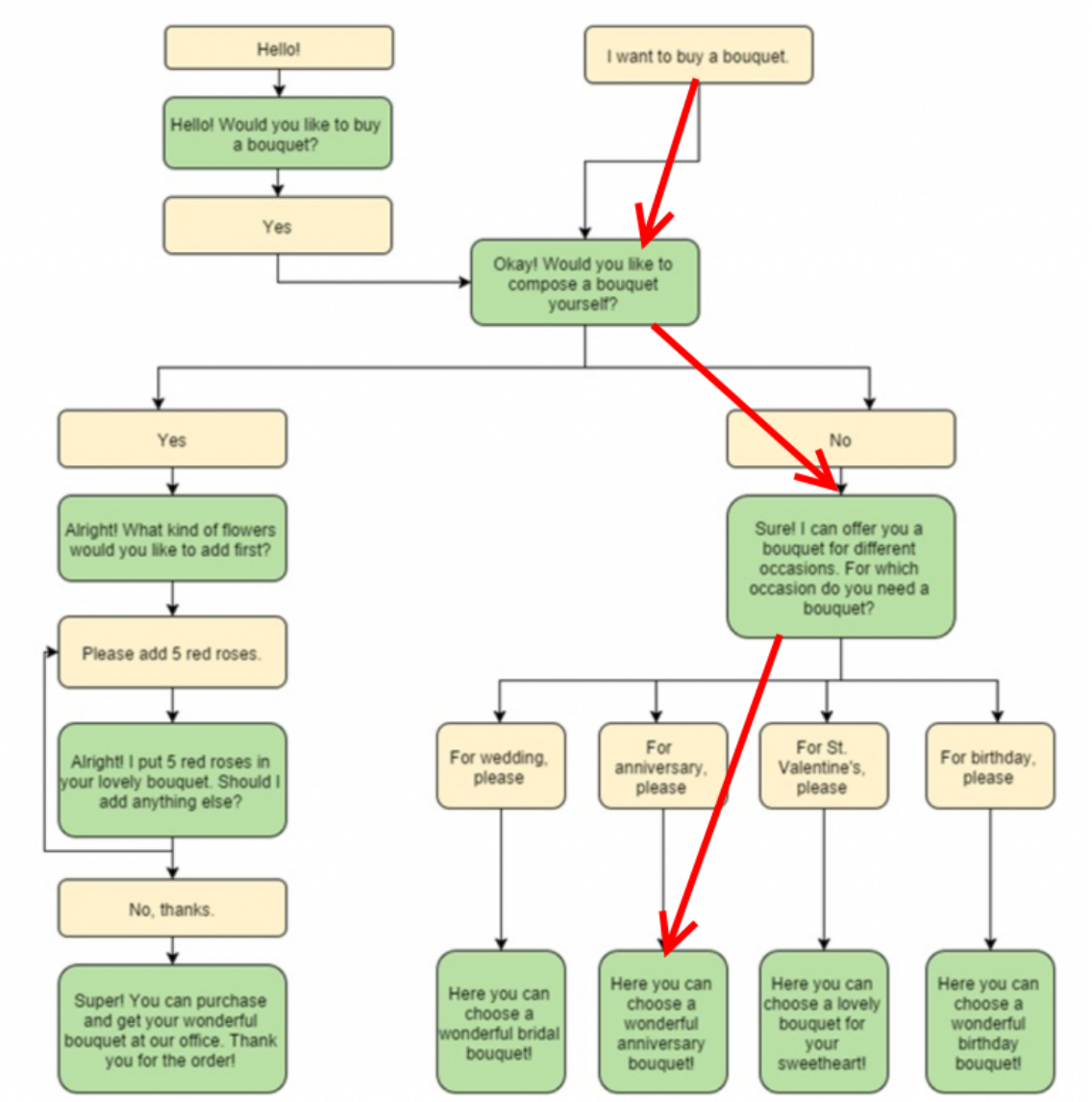
Obrázek 4: Konverzační strom nákupního bota [1]

Každá cesta vývojovým diagramem shora dolů je testovacím případem. Na Obrázku 5 můžeme vidět cestu konverzačním stromem, ve kterém si uživatel vytvoří přizpůsobenou kytici. Na dalším Obrázku 6 vidíme uživatele, který si vybral kytici k výročí. [1]



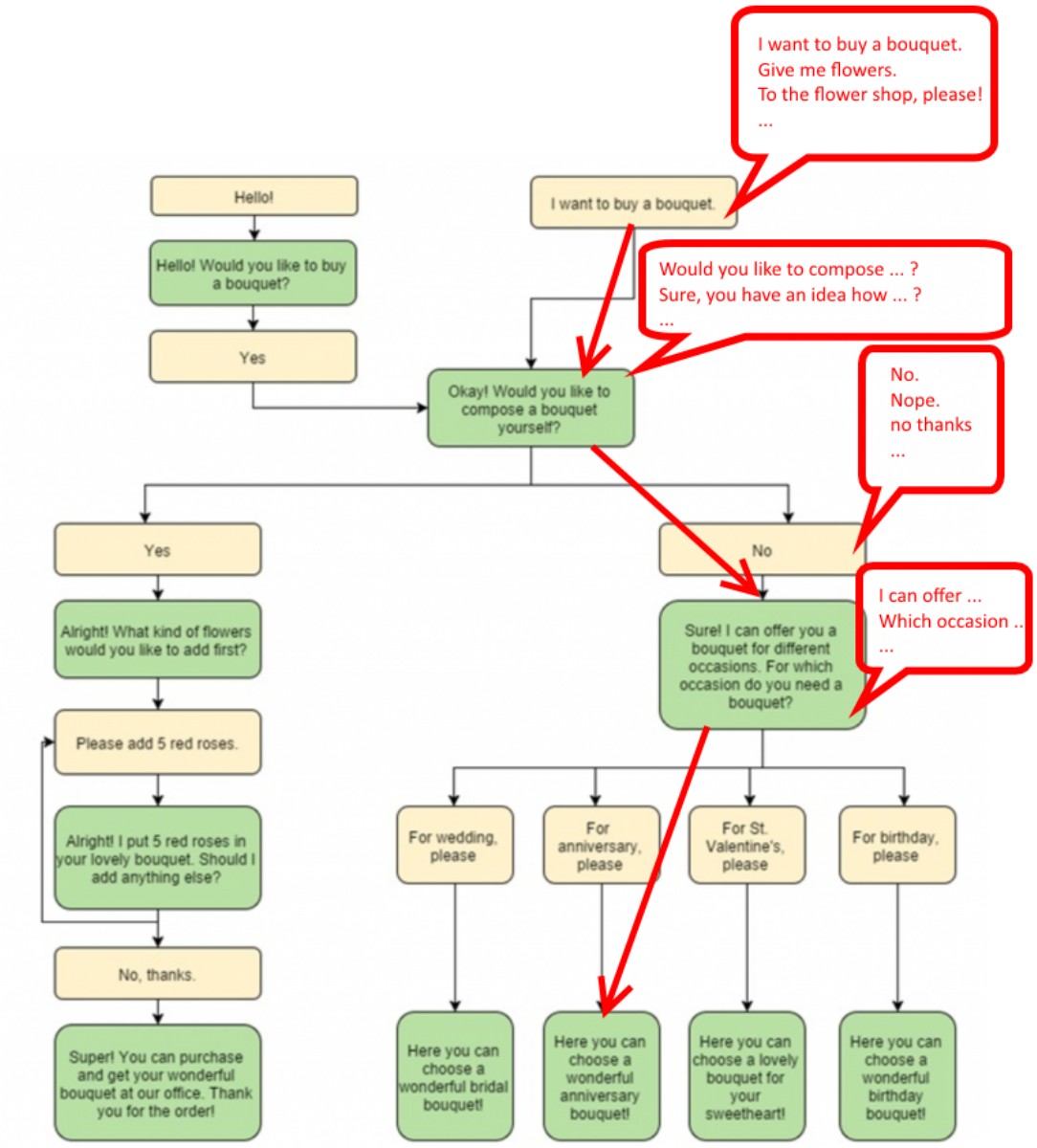
Obrázek 5: Uživatel vytvoří přizpůsobenou kytici [1]

Promluvy - To, co vývojové diagramy neukazují, jsou všechna možná vyjádření uživatelů, jichž je nekonečné množství. Z hlediska chatbota se možný uživatelský vstup nazývá „promluva“ a dobře navržený chatbot je trénován tak, aby rozpoznával většinu běžných promluv. To, co vývojové diagramy tak dobře neukazují, jsou výroky používané na druhé straně samotným chatbotem. Dobře navržený chatbot neodpovídá na stejné vstupy stále stejnými výstupy, ale poskytuje určité rozdíly v odpovědích na konverzaci. [1]



Obrázek 6: Uživatel si vybere kytici k výročí [1]

Pro každý uzel ve vývojovém diagramu je třeba zvážit různé vstupní a výstupní projevy. Vývojový diagram zobrazuje „šťastnou cestu“ v konverzaci, ve skutečnosti by se stejná konverzační cesta a testovací případ měly spokojit s většinou obvyklých výroků a kombinací výroků. Na Obrázku 7 můžeme vidět ukázku různých uživatelských vstupů. [1]



Obrázek 7: Ukázka různých uživatelských vstupů [1]

Variabilita kombinací nejrůznějších výroků je opravdu obrovská a jednu jedinou jednoduchou informací lze vyjádřit širokou škálou výroků. Příkladem může být obyčejné vyjádření ANO a NE, tedy souhlasu či zamítnutí. Vyjádření pro souhlas jsou například: yes, yes please, sure, do it, exactly, confirm, of course, sounds good, that’s correct, I don’t mind, I agree, ok. Vyjádření pro nesouhlas můžou být například tyto:

no, don't do it, definitely not, not really, thanks but no, not interested, I don't think so, I disagree, I don't want that, nok, nope, no thanks.

Tento seznam není zdaleka úplný, je tedy snadné si jednoduše domyslet, jak velké variace existují pro vyjádření složitějších záměrů. Je tedy více než jasné, že bez správného nástroje pro správu a spouštění identifikovaných testovacích případů, a to i těch nejzákladnějších, není možné tuto situaci vyřešit. [1]

Botium slouží jako podpora při tvorbě testovacích scénářů pro chatboty. Tyto scénáře lze definovat v souborech aplikace Excel, tok konverzace lze oddělit od výroků. Na Obrázku 8 můžeme vidět příklad toku konverzace shora v Excelu. Na Obrázku 9 můžeme vidět ukázku přesných výroků, které se mají použít v jednotlivých případech. [1]

	A	B
1	**User**	**Bot**
2	USER_BUY_BOUQET	
3		BOT_COMPOSE_YN
4	NO	
5		BOT_ASK_OCCASION
6	USER_ANNIVERSERY	
7		OK
8		
9		

Obrázek 8: Příklad toku konverzace shora v Excelu [1]

	A	B
1	**REFCODE**	**UTTERANCE**
2	USER_BUY_BOUQET	I want to buy a bouquet
3		Give me flowers
4		To the flower shop, please
5		Flowers, please
6		purches bouqet
7	BOT_COMPOSE_YN	Would you like to compose a bouquet yourself ?
8		You have an idea about the occasion ?
9		Is it for a special occasion ?
10	BOT_ASK_OCCASION	I can offer a bouquet for different occasions
11		What occasion do you need it for ?
12	USER_ANNIVERSERY	anniversery
13		I need it for an anniversery
14		
15		
16		

Obrázek 9: Přesné výroky, které se mají použít, jsou na samostatném listu nebo dokonce v samostatném souboru. [1]

Botium vytváří testovací skripty ze souboru (souborů) aplikace Excel a spouští je proti chatbotu. U odchylek od skriptů se testovací případ nezdaří a bude nahlášen. Botium se dobře integruje se stávajícími testovacími knihovnami (Mocha, Jasmine, Jest atd.), Také s kontinuálními integracemi, sestavením, nasazováním, testováním (Codeship, Travis, Bamboo, atd.). Dále se dobře integruje s chatbot platformami (Facebook, Slack, Microsoft Bot Framework, Hypertext Transfer Protocol (HTTP) endpoints, Dialogflow, Amazon Alexa a jiné). Botium obsahuje také technologicky nezávislou sbírku ukázkových výroků a konverzací, které by měl podporovat každý chatbot. Sbírka je dostupná pro různé jazyky. [1]

4 Společnost Feedyou

Společnost Feedyou se zabývá vývojem a rozvojem virtuálních asistentů (chatbotů a voicebotů) s cílem automatizovat neustále se opakující procesy, úkoly a komunikace v oblastech Human Resources (HR), Customer Service, Sales & Marketing.

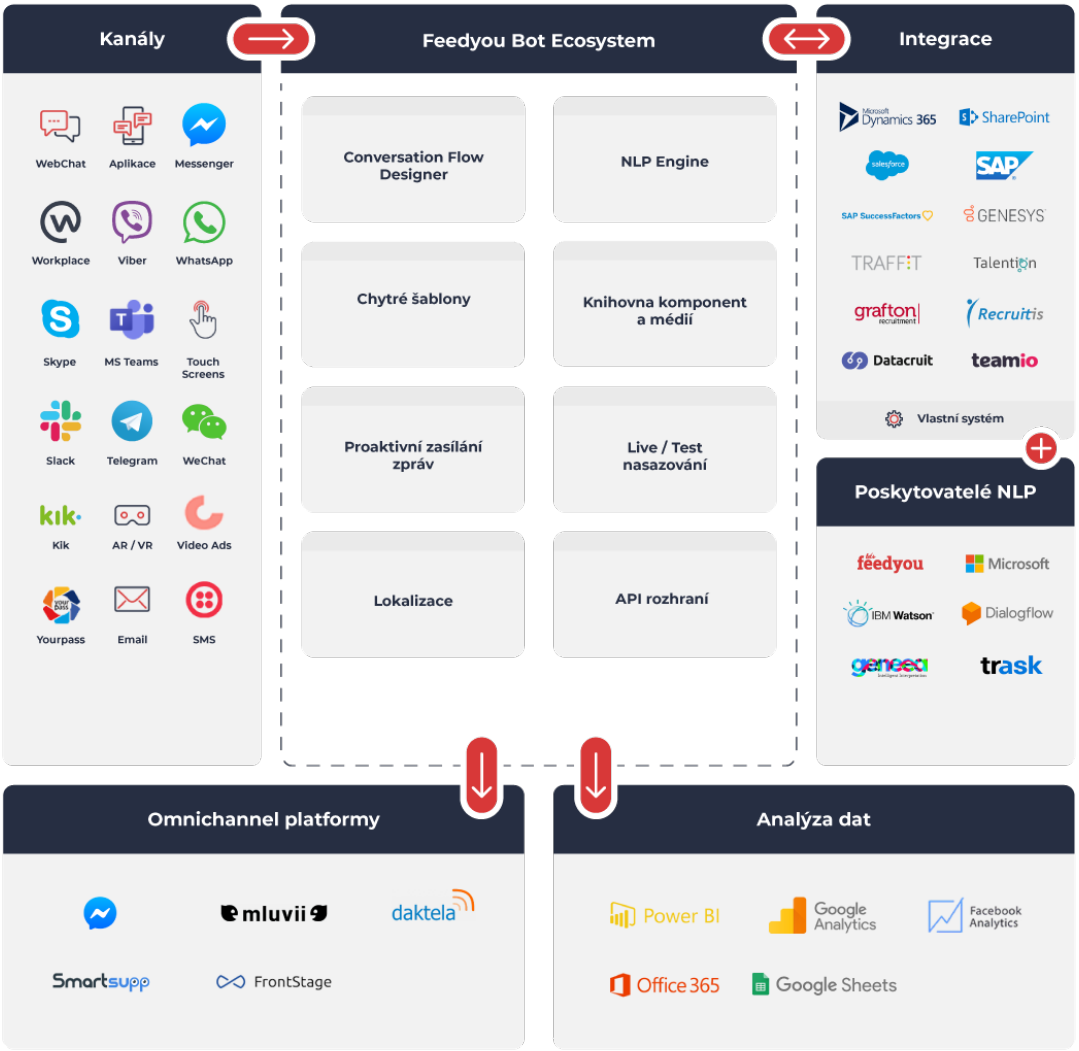
Feedyou má přes 200 úspěšných implementací ve 30 zemích Evropy. Spolupracuje se značkami jako je Dr.Max, Lufthansa InTouch, Swiss Life, Information and Communications Technology (ICT) Operator, IKEA, České radiokomunikace (ČRA), Institut klinické a experimentální medicíny (IKEM), ŠKODA AUTO, Continental, Knorr-Bremse, Grafton Recruitment, Gi Group, Česká pošta, Ipsos Group, Heimstaden, Professional Parcel Logistic (PPL), Sportisimo, Chèque Déjeuner a řadou dalších. Globální výzkumná a poradenská organizace Gartner Inc. je v lednu 2020 umístila na 1. místo v kategorii Conversational Platforms s hodnocením 5.0 spokojenosti jejich zákazníků. Jsou vítězem prestižních Microsoft Awards 2019, kategorie: Umělá inteligence (AI). Firma Feedyou získává ocenění po celém světě již od roku 2018.

Staví virtuální asistenty na prověřených technologiích od společnosti Microsoft. Provoz jednotlivých instancí je zpravidla na Microsoft Azure cloud. Od roku 2017 jsme technologickým partnerem Microsoft CZ/SK. V roce 2021 získali jedno z nejvyšších partnerství – Microsoft IP Co-Sell Ready Partner, které jim umožňuje využívat pro jejich zákazníky maximum benefitů, ať obchodních (ECIF), tak technologických. Podporují v současné době 104 jazyků – arabština, arménština, baskičtina, bengálština, čínština, čeština, dánština, holandština, angličtina, perština, finština, francouzština, galicijština, němčina, řečtina, hindština, maďarština, indonéština, irština, italština, japonština, korejština, litevština, malajština, nepálština, norština, polština, portugalština, rumunština, ruština, srbština, slovenština, slovinština, španělština, švédština, tagalog, tamilština, thajština, turečtina, ukrajinština a mnoho dalších. Dále rozšiřují portfolio především malých Central and Eastern Europe (CEE) jazyků.

Vytváří a napojují vlastní NLP (Natural Language Processing) modely, postavené na manuálním i samoučícím algoritmu (AI). Ty fungují na bázi neuronových sítí, které jsou schopné rozpoznat i například sentiment člověka. Napojují rovněž NLP modely třetích stran (Microsoft Language Understanding (LUIS), Geneea, SentiSquare, Trask Solutions, IBM Watson či Google Cloud). Mají multi-channel support. Jejich chatboti jsou schopni fungovat na webových stránkách, mobilních aplikacích, digitálních wallet kartách, omni-channel platformách typu mluvii, Daktela, Atlantis FrontStage, SmartsUpp a dále na instant messaging aplikacích typu Facebook Messenger, Viber, WhatsApp, Slack, Skype, Microsoft Teams, WeChat, Kik.

4.1 Technologie Feedyou Platform

Feedyou Platform je cloudové Software as a service (SaaS) (Software as a Service) řešení pro efektivní tvorbu, správu a rozvoj virtuálních asistentů (chatbotů a voicebotů) založených na bázi konverzačního stromu a NLP (Natural Language Processing) vč. strojového učení a napojování na různé komunikační kanály a informační systémy přes API. Na Obrázku 10 můžeme vidět jednotlivé moduly Feedyou platformy.



Obrázek 10: Moduly Feedyou Platform, zdroj: autor

Feedyou Platform se skládá z následujících modulů:

- Instance samotného jádra virtuálního asistenta (zpracovává příchozí a odchozí zprávy na základě aktuální verze konverzačního stromu, udržuje stav jednotlivých uživatelů a jejich pozice ve stromu nebo historii konverzace).
- Aplikace Conversation Designer pro kolaborativní vytváření a správu konverzačního stromu virtuálního asistenta (schválená verze je odeslána do instance virtuálního asistenta).
- NLP Designer Training – Součástí Conversation Designer je i modul pro vytváření modelů v napojení na konverzační strom virtuálního asistenta a následné trénování dle uživatelských dat z provozu.
- Služby Microsoft Bot Service pro napojení komunikačních kanálů.
- Feedyou NLP engine.
- Analytický modul (Power BI Embedded).
- Voice konektor.
- API rozhraní pro napojení systémů třetích stran.

Uživatelé Feedyou Platform mají možnost postavit si vlastní konverzační stromy virtuálních asistentů v aplikaci Conversation Designer, případně mohou zvolit některou z předpřipravených šablon komunikace pro specifické použití (Recruitment, Onboarding, Employee & Customer Support, Employee Survey, Presales, Net Promoter Score (NPS)). Zároveň je možné do samotné instance virtuálního asistenta napojit NLP modely pro porozumění psaným zprávám uživatele (Feedyou používá primárně svou NLP technologii, sekundárně napojuje NLP modely třetích stran jako jsou Microsoft LUIS, Geneea, SentiSquare, Trask Solutions, Google Cloud nebo IBM Watson).

Smyslem Feedyou Platform je umožnit uživateli jednoduchým způsobem postavit konverzační strom virtuálního asistenta v aplikaci Conversation Designer, kterou přemění na objektový zápis JSON a spustí ve vybraném komunikačním kanálu (více viz Podporované kanály).

Feedyou Platform pomáhá zároveň udržovat přehled nad jednotlivými konverzacemi uživatelů s virtuálními asistenty a umožňuje na základě získaných dat asistenty průběžně vylepšovat nebo měnit (v rámci například A/B testování). Všechna tato data Feedyou Platform zobrazuje pro každého virtuálního asistenta zvlášť v analytickém modulu Power BI Embedded s možností zapojit data z různých zdrojů třetích stran (Google Analytics, Facebook Analytics, . . .) a případně je exportovat do Google Sheets / Office 365.

Použité technologie: Backend Feedyou Platform i samotného virtuálního asistenta je vytvořen pomocí JavaScriptového prostředí Node.js a ověřených technologií jako jsou Microsoft Bot Framework nebo Express. Frontend využívá ověřenou kombinaci moderních reaktivních knihoven jako React nebo Redux.

Infrastruktura: Pro provoz celého řešení je využíván Microsoft Azure Cloud, kde každá platforma / virtuální asistent běží v oddělené skupině prostředků, což snižuje některá bezpečnostní rizika a usnadňuje správu. Všechny tyto skupiny jsou vedeny pod předplatným společnosti Feedyou, která se stará o správu nákladů. Aktuálně je společnost schopna pomoci zákazníkovi s nasazením celého řešení do jeho vlastní infrastruktury, nicméně má to následující specifika. Zákazník si bude sám v rámci svého Microsoft Azure účtu hostovat Node.js aplikaci se samotným jádrem virtuálního asistenta a databázi všech dat o uživatelích (historie zpráv, aktuální pozice v konverzačním stromu, apod.). Ve firemním Microsoft Azure Cloudu bude muset ale nadále běžet Feedyou Platform, tedy nástroj pro správu konverzačních stromů, NLP, statistiky a služba Microsoft Bot Service zpracovávající všechny jak příchozí, tak odchozí zprávy pro všechny kanály kromě WebChatu. Dále ještě na firemním účtu poběží služby pro rozpoznávání obsahu psaných zpráv uživatele (NLP), které jsou nasazené buď v Microsoft Azure Cloudu nebo v infrastruktuře firemního NLP dodavatele.

Natural Language Processing (Feedyou NLP Engine): Mezi hlavní přednosti NLP patří porozumění psanému textu úzce propojeného s konverzačním stromem. Využívá nejnovějších machine learning a lingvistických technik. Přínosná je také správna NLP a konverzačního stromu na jednom místě včetně přetrénování NLP modelu na jedno kliknutí. Engine má podporu více jak sto jazyků včetně češtiny nebo slovenštiny a detekci jazyka. Důležitá je maximální otevřenost, neboli možnost volby enginu od Feedyou nebo třetí strany, pokud je například již využívána (Microsoft LUIS, Geneea, SentiSquare, IBM Watson, Trask Solutions, Google Cloud, Facebook Wit.ai). NLP běží jako součást virtuálního asistenta, tedy například i ve vlastním cloudu a má vysoké možnosti škálování.

Feedyou Platform je unikátní tím, že každého daného virtuálního asistenta lze zároveň spustit ve více komunikačních kanálech najednou, přičemž výsledky ze všech těchto kanálů se scházejí na jednom místě a zároveň všechny změny provedené na asistentovi se automaticky projeví ve všech kanálech.

Aktuálně lze virtuální asistenty provozovat současně na těchto kanálech: Feedyou WebChat, Feedyou iFrame, mobilních aplikacích (iOS a Android), digitálních wallet kartách, chatovacích aplikacích typu Facebook Messenger, Viber, WhatsApp, Slack, Skype (for Business), Microsoft Teams, Facebook Workplace, WeChat, Kik či na omnichannel (call centre) platformách jako (O2) Mluvii, Daktela, Atlantis FrontStage, SmartsUpp a dalších. V neposlední řadě jsme schopni napojit firemní chatboty na personalizované video reklamy a kampaně (Motionlab). Pracuje se nyní na napojení ZenDesku a dále na kanálech typu RCS/RBM a Apple Business Chat.

4.2 Výrobní proces

Proces výroby bota začíná tak, že si obchodník domluví s potenciálním klientem rámcově obsah zakázky a sepíše, jaký má být chatbotův účel. Pokud bude potřeba integrace na zákazníkův systém a obchodník toto ví, rovnou klienta požádá o specifikaci API, na které se bude bot připojovat. Jako další krok je konzultace obchodníka s vedoucím výroby, který odhadne náklady na základě dodané specifikace. Obchodník

si schválí s klientem cenu zakázky, zařídí veškerou smluvní dokumentaci. Ve chvíli, kdy jsou vyřízeny všechny formality, propojuje obchodník klienta s účetním oddělením. Doladí se smlouvy a dodatky pro klienty a rovněž se vystaví faktura.

Projektový manažer nadále komunikuje s klientem za účelem upřesnění podrobných specifikací zadání, které obsahují materiály nutné pro začátek práce na zakázce. Dále je nutné stanovit termíny, kdy se začne na zakázce pracovat a k jakému datu dojde k zaslání prototypu. Poté probíhá samotná výroba chatbota v designeru, který bude dále popsán v kapitolách níže. Po dokončení struktury bota se odešle klientovi prototyp k připomínkování. Klient chatbota schválí, nebo ho připomínkuje. V dalším kroku je tedy případně potřeba zapracovat připomínky. Pokud dojde v rámci připomínek ke změně v rozsahu zadání, informuje účetní oddělení klienta o více nákladech na úpravu. Pokud klient souhlasí, předá se informace k vytvoření faktury. Když klient souhlasit nebude, změny se zapracovávat nebudou. Následně se chatbot předá klientovi. Potom už se jen hlídají pravidelné platby za tuto službu. Na Obrázku 11 můžeme vidět procesní diagram výroby chatbota ve společnosti Feedyou.

4.3 Designer

Při založení společnosti Feedyou ještě Microsoft neměl dobrý nástroj na definici bota, a proto byl vytvořen vlastní designer. Designer umí držet historii verzí, vkládat komentáře pro týmovou práci, nechybí ani jazykové mutace a hlavně jednoduché vytváření stromové struktury. Díky tomuto nástroji nemusí chatboty pro zákazníky tvořit jen vývojáři, ale školí se na ně konverzační specialisté.

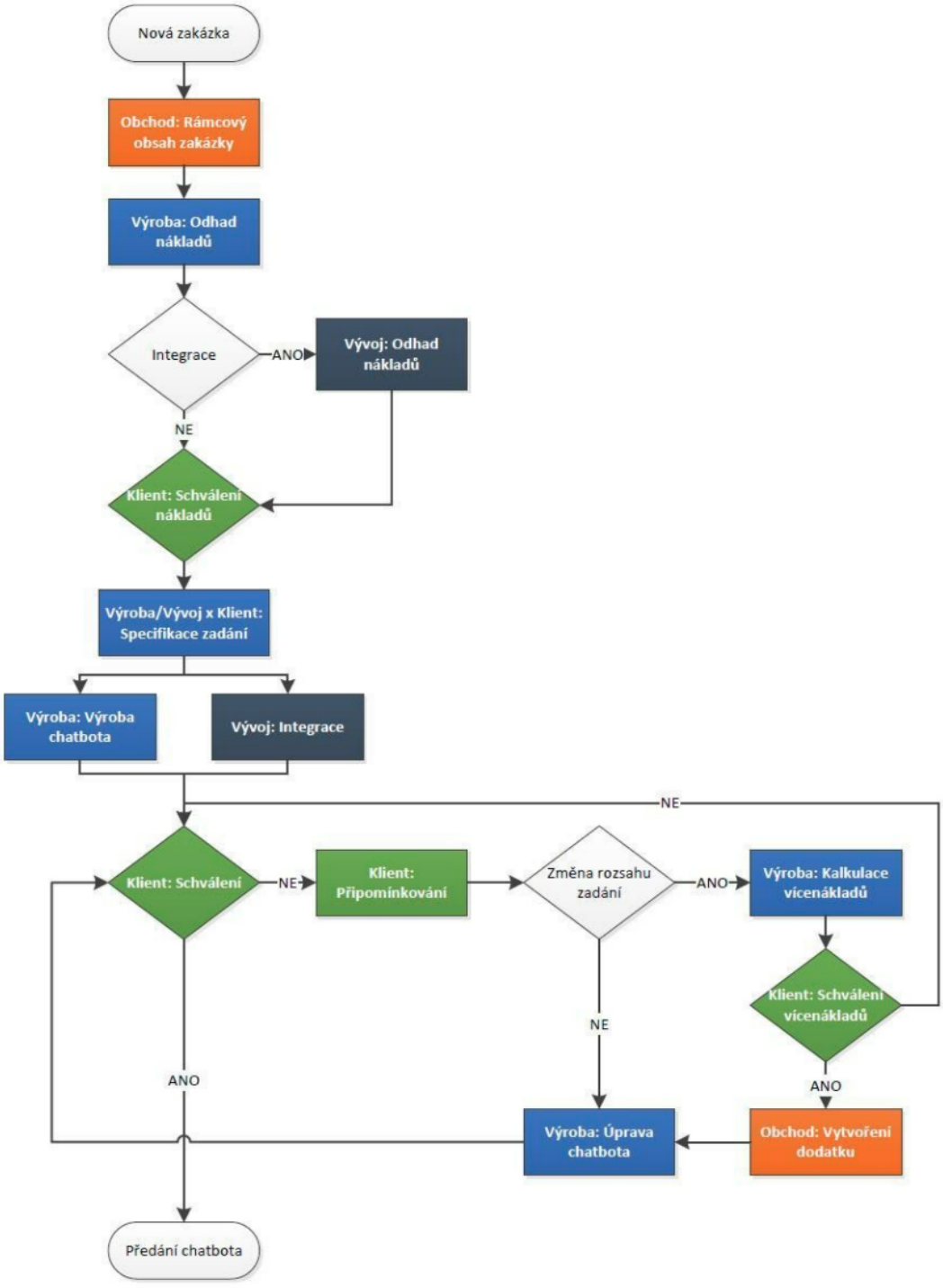
V designeru můžeme tvořit bota v podobě JavaScript Object Notation (JSON). Po kliknutí na přidávací tlačítko kroku se nám zobrazí dvanáct možností, pomocí nichž si můžeme přidat do stromu bota různé komponenty, které následně tvoří jeho výslednou funkčnost. Tyto nástroje si nyní stručně popíšeme. Začneme nástrojem „Zpráva“. Je to text, který píše virtuální asistent uživateli. Nevyžaduje žádnou reakci, popřípadě ani odpověď uživatele. Můžeme do ní vložit obrázky nebo gify. Mezi jednotlivými zprávami (i dalšími textovými kroky) je časová mezera, aby virtuální asistent působil přirozeně.

Dále máme „Closed Question“. Uzavřená otázka je vhodná v případě, že je žádoucí uživateli nabídnout tlačítka s definovanými odpověďmi. Uživatel nebude mít možnost vložit vlastní text, ale musí si vybrat z jedné z možností.

„Open question“ vyžaduje odpověď uživatele. Pokud uživatel na otázku neodpoví, virtuální asistent nepokračuje v komunikaci. Lze vložit validace na vstup od uživatele.

„Confirm question“ s možností odpovědí Ano/Ne je dalším nástrojem. Hlavní výhodou potvrzovací otázky je, že má přednastavená synonyma. Tzn. pokud není vypnutý input (uživatel může psát) a uživatel napíše například „ano“, „jj“ apod., odpověď se zaregistruje jako Ano. Další výhodou je vytvoření uzavřené otázky typu Ano/Ne na pár kliknutí.

Další možností, která se nabízí, je „Dialog“. Komunikace virtuálního asistenta je z dialogů poskládána. Je to tedy další z nástrojů, který můžeme vložit do stromové struktury. Je to vlastně složka již vložených kroků, které můžeme dále znovu a znovu využívat.



Obrázek 11: Procesní diagram, zdroj: autor

„E-mail“ se využívá k odeslání konkrétních informací na zadaný e-mail klienta, nebo k odeslání e-mailu uživateli - například připomínka pro odeslání životopisu personalistovi dané firmy.

„Attachment“ se využívá k nahrání přílohy do virtuálního asistenta. Obvykle se jedná o životopis, případně certifikát nebo jiné ověření a také o fotografii.

„Storage“ je název pro hodnotu, která se ukládá do souboru s veškerými daty, která virtuální asistent shromáždil (tzv. data sheet). Standardně storage obsahují kroky uživatele, které je žádoucí zaznamenat, například Otevřená/Uzavřená otázka. Ukládat do storage lze také manuálně.

„Switch“ funguje na bázi podmínek. Základní podmínkou je Default a ostatní je nutné Nastavit/Přidělit. Pokud je splněno některé z přidělených kritérií (podmínek), pokračuje uživatel v dané větvi. Pokud není splněno zadané kritérium, pokračuje uživatel dále větví Default. Obsahem podmínek je vždy obsah ve storagích.

„Action“ lze vložit do konkrétního místa v rámci komunikační struktury. Akcí lze například ukončit dialog, popřípadě celou konverzaci. Nejčastěji se využívá akce právě pro skončení dialogu, aby komunikace nepokračovala dál - pomocí End Conversation. Toto nastavení kompletně ukončí konverzaci.

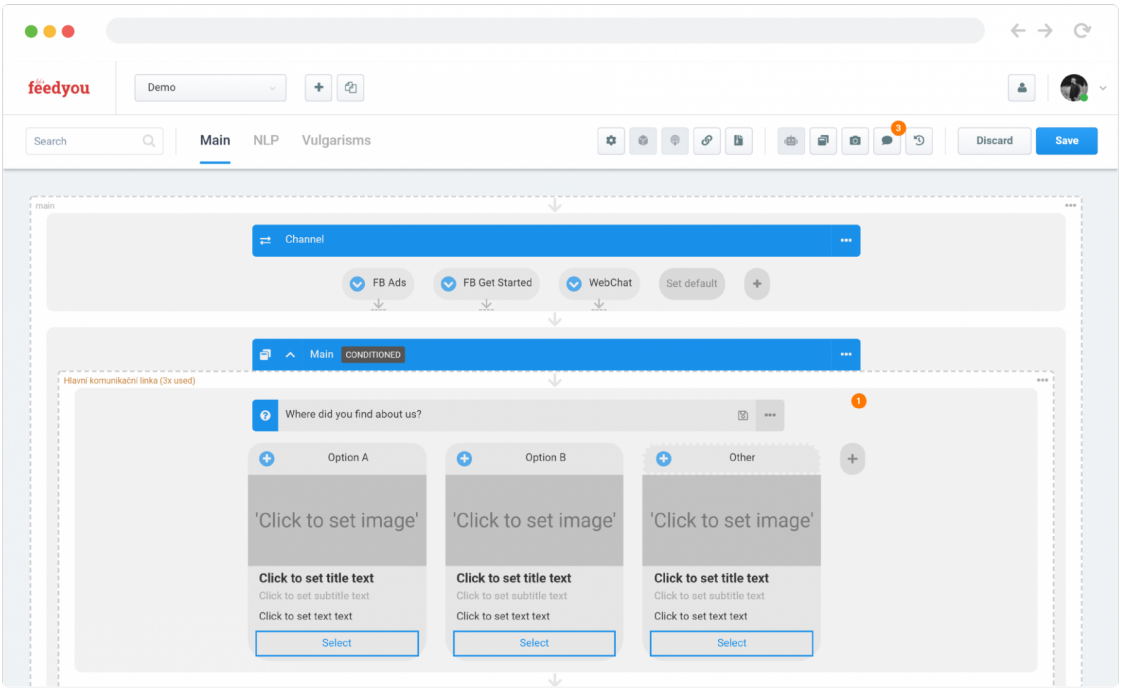
„Timeout“ je nástrojem, pomocí něhož lze limitovat čas, do kterého musí uživatel projít danou sekcí (dialogem) bota. Pokud to do daného časového limitu nestihne, spustí se specifický dialog, po jehož konci virtuální asistent pokračuje následujícím krokem, nebo tam, kde uživatel skončil. Pokud uživatel projít sekci s Timeoutem stihne, dialog je pro vypršení času přeskočen a Virtuální asistent (VA) pokračuje standardně dál.

„Integration“ představuje způsob, jak pracovat s uživatelskými daty virtuálního asistenta získanými typicky z externích systémů. Používá se například pro úpravu dat virtuálního asistenta, odeslání dat do systému, získání dat pro zobrazení v carouselu, nebo vyvolání akce v externím systému. Integrační Krok se spustí ve chvíli, kdy uživatel dojde v konverzaci do bodu, kde je Krok definován.

„CopyPaste“ slouží pro kopírování částí bota do jiných míst, případně i do jiného bota.

V době psaní tohoto textu by se dalo konstatovat, že Microsoft stále ještě nemá opravdu dobrý nástroj pro tvorbu chatbotů bez práce s JSONem. Microsoft Bot Framework Composer je určen spíše pro programátory, protože se tam pracuje přímo s JSONem a vše nelze pohodlně naklikat. A samozřejmě chybí mnoho potřebných vlastností, které už náš designer má a který neustále vylepšujeme. Do budoucna je v plánu tento nástroj dát na Microsoft Store, aby naší Feedyou Platform (designer) mohli používat lidé po celém světě a mohli si tak pohodlně vytvořit boty pro sebe, svoji firmu nebo jiným firmám na zakázku. Samozřejmě ne však zadarmo.

Designer si umí držet historii verzí, takže se můžeme vracet k našim změnám. Dále má komentáře, což je ideální na týmovou tvorbu botů. Samozřejmostí jsou i jazykové mutace, NLP editor, pomocí kterého nasadíte chytré NLP modely a díky jednoduchému editoru je můžete sami rozvíjet a ručně trénovat. Naprosto běžné je následné vytvoření instance bota, který můžeme vyzkoušet v novém okně. Na Obrázku 12 můžeme vidět ukázku designéru.



Obrázek 12: Designer, zdroj: autor

Každý nový uživatel (firma) dostane svou vlastní instalaci Feedyou Platform, do které se přihlašuje a autorizuje prostřednictvím specializované platformy Auth0 za pomocí podporovaných adaptérů jako jsou Facebook Account, Email + heslo, Google Suite, Microsoft AD, apod.

5 Implementace testovacího nástroje E2E Bot tester

Testovací nástroj je rozdělený na dvě aplikace, které spolu navzájem komunikují, ale fungují naprosto nezávisle. RESTové rozhraní najdeme ve složce server. Je to šablona, která je vygenerována pomocí příkazu express-generator. [16] Klientskou aplikaci nalezneme ve složce client. Je to šablona vygenerovaná z balíčku tokyo-free-white-react-admin-dashboard. [17] Tato knihovna velice usnadní vývoj aplikace, protože vývojář nemusí řešit vzhled komponent. Ty jsou již připravené v ukázkových stránkách. V této šabloně byla i šablona pro konverzační chat, který byl pro vývoj testovacího nástroje pro chatboty ve firmě Feedyou taktéž nadšeně využit. Ve složce src najdeme zobrazovací komponenty, které pouze vykreslí *response* ze serveru pro uživatele příjemným způsobem. Každá aplikace se pouští zvlášť příkazem *npm start* v rootu složky client a server. Nyní již můžeme pokračovat ve vývoji a změny se nám hned reflektují do webového prohlížeče. Klientská aplikace běží na adrese: http://localhost:3001 a serverová na http://localhost:3000.

5.1 Motivace

Hlavní motivací celého procesu práce práce a tvorby sytému automatického testování bota je zefektivnění práce ve společnosti Feedyou a odchycení chyb ještě před nasazením do produkce. Doposud museli pracovníci z výroby a testeři bota proklikat a zjistit, zda po úpravách v definici chatbota funguje stromová a NLP část dle očekávání. Vyvíjená aplikace to zvládne za okamžik, což šetří čas a odstraňuje lidský faktor z testování. Naše implementace chatbotů jsou často rozsáhlé stromové struktury, které by zaměstnanci trvalo otestovat i desítky minut. Další velkou motivací je i samotné testování designeru, který generuje *JSON* dle vytvořené stromové struktury. Abychom měli jistotu, že naše vylepšení v tomto nástroji neovlivní fungování bota, náš tester má v plánu napsat testovací stromové struktury, které prověří, zda po nasazení nové verze chatbot odpovídá stále stejně.

5.2 Výběr nejvhodnějšího řešení

Ve společnosti je už velké množství implementovaných chatbotů, které je potřeba testovat. Hlavní motivací je zefektivnění práce ve společnosti a podchycení chyb ještě před nasazením do produkce.

První a nejjednodušší možnost je testovat chatboty ručně po každé změně struktury. Tento způsob nevyžaduje žádnou implementaci, a proto je nyní ve společnosti využíván. Má však mnoho nedostatků, protože testování zabere mnoho času. I kdyby měl zaměstnanec všechny scénáře předem připravené, odpovídal rychle a též rychle kontroloval výstupy, nebude ve skutečnosti nikdy opravdu rychlým. Chatbot sám totiž zdržuje, aby se podobal živému uživateli, a zobrazuje nám, že právě píše. Čas, po který je informace zobrazena, je přímo úměrný počtu znaků ve zprávě.

Další možností by bylo využít testovací nástroje jako třeba ElasTest, který se používá na testování webových aplikací. [18] Tento nástroj podporuje klikání, čekání a dokáže porovnávat očekávané chování s aktuálním. Byla by to vlastně simulace uživatele, který si píše přímo s chatbotem. Nevýhodou tohoto řešení jsou přibývající finanční náklady. Zprávy se přes náš webchat musí posílat přes Microsoft Bot Service, který ale není zadarmo. Tato vrstva umí z chatů jako je Facebook, Whatsapp, Viber přeložit zprávy a poslat je dál jako jednotný *request*, který už jde na náš firemní server. Náš server dle definice bota odpoví na definovanou Uniform Resource Locator (URL) z body příchozího *requestu*, což opět dále navyšuje cenu, protože Microsoft Bot Service tuto *response* přeloží a pošle do příslušného komunikátoru.

Třetí možností, jak boty testovat, je použít nástroj Bot Framework Emulator. [9] V tomto nástroji lze testovat definovaného bota a dívat se na jeho odpovědi i v podrobnějším měřítku jako JSON. Aplikace umožňuje uložit celý *transcript*, neboli záznam komunikace, který je možné i později nahrát a ručně porovnávat dřívější odpovědi. Toto řešení nebylo stále vyhovující, protože bylo potřeba celou tuto oblast zautomatizovat. Tato možnost se sice obejde bez Microsoft Bot Service a nemusí se za ní platit, protože voláme přímo Back End (BE) bota, ale nijak nám v automatizaci nepomůže. [19]

Jako nejvhodnější řešení bylo vybráno využití přímého volání našeho serveru, který funguje přes RESTové rozhraní. Ten pak odpovídá na námi definovanou URL. Hlavní výhodou je vlastní režie posílání zpráv, volání rozhraní třetích stran, ukládání scénářů a po změně bota vyzkoušení, zda stále bot odpovídá správně dle uloženého scénáře.

Nyní již máme k dispozici záznam z konverzace, neboli *transcript*, a můžeme tak porovnat příchozí odpovědi od chatbota. Aplikace má dva vstupy. Prvním je *apiUrl*, protože chatbot může běžet v různých prostředích, na testu nebo přímo u zákazníka. Druhý parametr je *userId*. I když název trochu mate, je to unikátní id konverzace a každé zahájení konverzace má svoje nové vygenerované userId. Aktuální získávání tohoto id je trochu uživatelsky nepřívětivé a musíme si ho vytáhnout z logovací databáze nebo z logu běžícího bota na serveru. Do budoucna bude v aplikaci možné dodělat vytvoření scénářů pro jednotlivé uživatele, kteří si ho potom jen vyberou ze selectboxu a spustí test. Ten nyní porovnává hlavně vrácené texty. Do budoucna se budou hlídat i časy reakce. Dle dostupného scénáře se dopočítají časy mezi odpověďmi s nějakou rezervou, a když bot neodpoví do tohoto času, aplikace test přeruší, vrátí chybu a pošle záznam, kam až konverzace došla. Díky tomu přesně vidíme, kde a kdy přestal bot odpovídat, a také hned víme, v jakých místech si zkontrolovat definici bota. Na následující Ukázce kódu č. 2 můžeme vidět pro představu zjednodušený scénář, ve skutečnosti se posílá o mnoho více atributů, které následně porovnáváme s aktuální příchozí odpovědí z našeho bota.


```
[
   {
      "Timestamp":"2021-12-04T17:03:24.118Z",
      "Direction":"out",
      "Text":"Here's a~message with a~picture carousel and buttons that lead to
    the web."
```

```
    },
    {
        "Timestamp":"2021-12-04T17:03:23.318Z",
        "Direction":"in",
        "Text":"Ok"
    },
    {
        "Timestamp":"2021-12-04T17:03:20.139Z",
        "Direction":"out",
        "Text":"We can begin?"
    },
    {
        "Timestamp":"2021-12-04T17:03:20.041Z",
        "Direction":"out",
        "Text":"Hello, I'm Feedbot, Feedyou's sample chatbot.I am here to show you
     all the elements that may appear in our chatbots. "
    }
]
```

Ukázka kódu 2: Transcript

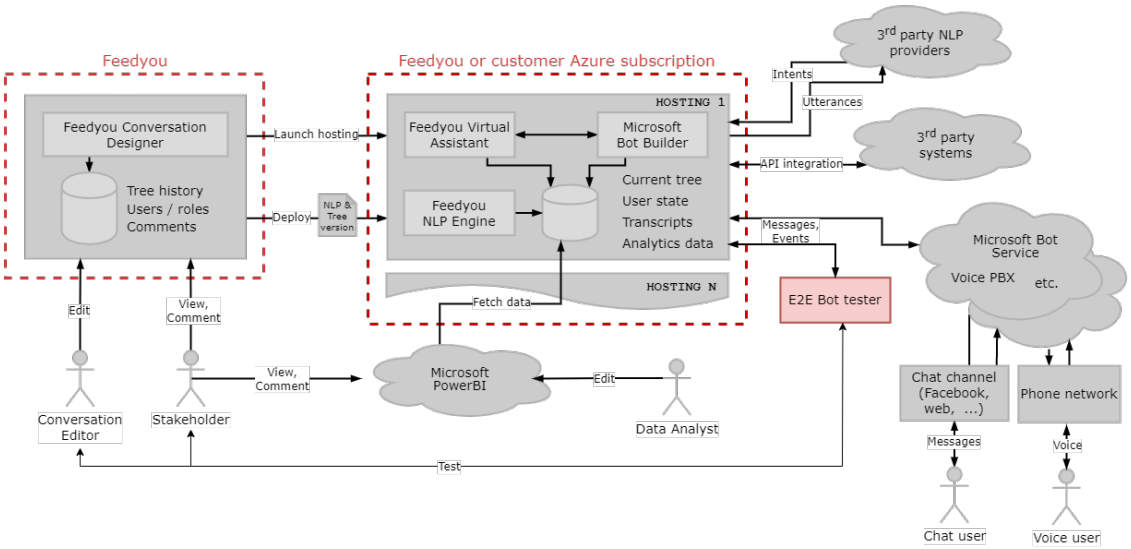
Dalším rozšířením je volání rozhraní třetích stran. Mnoho implementovaných chatbotů si dotahuje informace ještě z jiných systémů. Mohou to být informace o počasí, o zákaznících, o produktech nebo například výsledná cena sjednaného pojištění. Aplikace bude umožňovat i mockovat tyto systémy, protože tento testovací nástroj testuje primárně naší definici bota a ne API třetích stran. V další kapitole ještě dále popíšu, jak nyní testujeme volání API třetích stran.

5.3 Bot tester ve Feedyou Platformě

Pro lepší představení fungování bota je přiložen obrázek ekosystému, který funguje ve společnosti FeedYou. Na obrázku 13 můžeme vidět celou Feedyou platformu, v níž je znázorněné, jak spolu komunikují jednotlivé systémy ekosystému.

5.4 Popis funkčnosti Back End

Aplikace je napsaná v Node.js s využitím knihovny Express js. Jedná se o RESTové rozhraní, které umí komunikovat s naším interním RESTovým rozhraním, kde běží naši chatboti. Toto rozhraní má více prostředí, je možné ho pro vývojové účely pustit i lokálně, protože zde jsou nadefinovaní testovací boti. V cílovém adresáři najdeme několik složek a souborů. Nás bude hlavně zajímat app.js, ve které je nastavení RESTového rozhraní. Probíhá zde registrace adres, které následně budeme používat. Adresa start slouží pro zahájení a ukončení konverzace, to je ta, kterou budeme volat v klientské aplikaci. Další je adresa *messages*, která se přidává do *serviceUrl* při posílání zpráv chatbotovi, a na ni mám též budou chodit zprávy. Dále se zde nastavuje logování, Cross-Origin Resource Sharing (CORS) politika, cookies, error handler a další.



Obrázek 13: Ekosystém fungování bota ve společnosti Feedyou, zdroj: autor

Další důležitá složka nese název routes. V indexu je výchozí routa, kterou k ničemu nepoužíváme. V souboru start je kód, který spouští start konverzace. V souboru messages je hlavní logika aplikace, která pracuje s příchozími zprávami, posílá odchozí a oba tyto typy zpráv porovnává se scénářem.

BE aplikace spustíme tak, že zavoláme URL aplikace/start s dvěma parametry. První je apiUrl, který říká na jaké URL adrese virtuální asistent běží, druhý parametr je userId, který se nyní používá jako jednoznačný identifikátor konverzace. Pomocí těchto parametrů si bot tester vytáhne z databáze *transcript*, neboli scénář, který budeme testovat. Když se nám ho nepodaří vytáhnout, vrátíme zprávu, že je prázdný, a nepokračujeme dál.

Ve složce src můžeme najít vytažené funkce z adres. V souboru *beginIntroDialog* nalezneme parametrický *request* pro začátek dialogu. Posílá se v něm typ event s názvem *beginIntroDiadlog* společně s adresou, ve které máme uložené *userId* a *conversationId*, což je unikátní id konverzace. Další potřebný atribut je *serviceUrl*, jinak by bot nevěděl, kam nám má odpovědět. Z důvodu toho, že byl požadavek, mít více vláknovou aplikaci, v adrese si pošleme unikátní testovací id, které jsme si vytvořili na začátku testu. Je to klíč v mapě, která je umístěná v souboru *temporaryData*. Je to obyčejná javascriptová mapa, kde se pod klíči(*testId*) schovávají veškeré informace o probíhajícím i dokončeném testu. Nalezneme tam dotažený *transcript*, průběžně ukládanou konverzaci, apiUrl a aktuální index *transciptu*. Díky tomuto průběžnému ukládání informací do mapy, může vedle sebe běžet více konverzací a aplikaci nevadí, že jsou puštěné najednou. V souboru utils nalezneme funkci na tvorbu testId a další přepoužitelné funkce, které se následně volají v souboru *messages*. V souboru *sendMessage* najdeme parametrický *request* na zasílaní obyčejných zpráv chatbotovi. Je velice podobný jako *request* pro začátek konverzace, jen s rozdílem, že typ je *message*.

Při každém použití bota se nám *transcript* zaznamenává, a my tak můžeme vidět historii komunikace, analyzovat ji a sbírat případně různá data spojená s naším využitím. Příchozí *transcript* se vyčistí od prázdných příchozích zpráv a od první testovací zprávy, která se do transcriptu neukládá, ale přijde na testovacím prostředí. Dále je *transcript* převrácen, protože je vyžadováno obrácené pořadí zpráv. Aplikace si vygeneruje unikátní *testId* dle zvoleného algoritmu a uloží jej do mapy, kde se postupně ukládá konverzace. Dále si tam uložíme *transcript*, *apiUrl*, index, hlavní *response* a *timeout*. Po té zavoláme první *request* na chatbota, který nastartuje naší komunikaci. Na následující Ukázce kódu č. 3 můžeme vidět ukázku posílání zpráv se servisní URL, na kterou nám bot zpětně odpoví.

```
const sendMessage = async (apiUrl, text, testId, ownUrl) => {
  await fetch(`${apiUrl}/api/messages/custom?code=${process.env.AUTH_CODE}`, {
    method: "post",
    body: JSON.stringify({
      type: "message",
      text: text,
      sourceEvent: {
        mockIntegration: true,
      },
      address: {
        user: {
          id: testId,
        },
        conversation: {
          id: testId,
        },
        serviceUrl: `http://${ownUrl}/messages?testId=${testId}`,
      },
    }),
    headers: { "Content-Type": "application/json" },
  });
};
```

Ukázka kódu 3: Posílání zpráv

Další možné řešení, které se nabízelo, bylo mít jen jeden endpoint, který by začal konverzaci a zároveň byl i jako *serviceUrl*. V čem je ale u takového řešení problém? Hlavní důvod pro zvolení dvou endpointů byl ten, že potřebujeme sbírat informace o testování a postupně je ukládat. Abychom mohli čekat na další zprávu, musíme zaslat *response*, jinak by aplikace nemohla nadále reagovat. Bylo potřeba oddělit komunikace server na server a klient na server. Klienta nezajímají průběžné výsledky testování, jemu stačí až konečný výsledek. Proto vznikla routa *messages*, kterou si uvedeme jako *serviceUrl* ve startovací události a pak už si komunikuje server sám s tímto endpointem, aniž by o tom klient věděl. Nabízí se otázka, jak po konci testu ukončit *request* v endpointu start, když už chatbot komunikuje pouze s endpointem *messages*. A nyní přichází jedna z velkých výhod funkcionálního programování. Ve startu jsme si na konci vytvořili mapu, do které jsme pod vygenerovaným testId uložili všechny potřebné informace, včetně dotaženého *transcriptu*. O tom nyní mluvit

nebudeme, to je popsáno níže. Hlavní co nás nyní zajímá je, že si do mapy pod atribut můžeme uložit i funkce, tak proč si tam neuložit celý *response*? Javascript si pamatuje reference, takže zavolání *reponse.json()* v jiném endpointu aplikace, uloženého v mapě, je možné. Díky tomu můžu ukončit *response* endpoitnu start v endpointu *messages*.

Stejný postup byl využit i pro *timeout*. Ten byl potřeba zavést z několika důvodů. Hlavní důvod byla situace, kdy by chatbot přestal odpovídat, pak by testovací aplikace nemohla dále pokračovat v činnosti a *response* by nikdy nepřišla, protože by pořád čekala na odpověď od chatbota. Proto si můžeme nastavit čas, po který má testovací aplikace čekat. Čas může být fixní, například 3 vteřiny, nebo ho můžeme dopočítat podle transcriptu, kde je čas jednotlivých zpráv a přidat k tomu nějakou časovou toleranci. Když chatbot odpověď nestihne, nemusíme nutně ukončit testování, ale stačí vypsat ke zprávě o kolik jednotek času přišla později. Obtíž byla v tom, jak tento *timeout* ovládat. V tom nám zase pomohl funkcionální přístup. V momentě uložení *timeoutu* do mapy, začne *timeout* běžet v event loop nezávisle na probíhající komunikaci mezi naší aplikací a chatbotem a odpočítává nastavený čas. Po jeho uplynutí se spustí funkce, definovaná v *timeout*. Nyní máme ve funkci uložený *response* ze startu, který zašle v JSONu chybu, že čas vypršel. Další komplikace nastává, když komunikace běží správně, ale nám běží na pozadí *timeout*, který by zaslal chybu. To znamená, že musíme při doručení zprávy na endpoint *messages timeout* resetovat. To uděláme tak, že ho jen znovu nastavíme a uložíme do mapy, a díky tomu, že jsme to stihli dřív, než vypršel, komunikace běží dále. Tento čas můžeme nastavit dle našich potřeb. Nyní se nastavuje jako konstanta v souboru. Na následující Ukázce kódu č. 4 můžeme vidět ukázku uložení *timeoutu* a hlavního *response*.

```
data.set(testId, {mainRes: res, \textit{transcript}, index: 0, apiUrl, timeout
: setTimeout(() => res.json({ error: 'timeout' }), TIME_OF_TIMEOUT)});
```

Ukázka kódu 4: Uložení hlavního response a timeoutu

V prvním *requestu* je v body takzvaná *serviceUrl*, aby bot věděl, na jakou URL má poslat odpověď. Tímto se nyní dostáváme k druhému endpointu aplikace (aplikace/-messages). Protože jsme si do *serviceUrl* poslali jako parametr i unikátně vygenerované *testId*, můžeme si dle něj vytáhnout z mapy konkrétní test, ve kterém již máme uložený testovací scénář a další potřebné atributy. Chatbot posílá zpátky zprávy jako POST na *serviceUrl*, v jehož body přijde text, type a *attachments*. V textu je obsah příchozí zprávy, v type je typ zprávy. Může být buď *typing* nebo *message*. Aby mohl bot simulovat svoje psaní a působit tak věrohodněji, posílá zprávy typu *typing*, a my v našem chatu zobrazíme, že píše.

Tyto zprávy nejsou pro test nijak důležité, proto při příchodu posílám *response* a tím tento *request* ukončím a čekám na další zprávu. Když se nejedná to *typing*, jedná se o *message*. Podíváme se do transcriptu, zda na indexu, který jsme si uložili do mapy očekáváme vstupní nebo výstupní zprávu. Jestliže očekáváme vstupní, pošleme *request* na bota ze zprávou z transcriptu, dále si pošleme zase *testId*, abychom si dle něj mohli zase stáhnout ty správné údaje. Díky tomuto přístupu nám může běžet více testů najedou a aplikace je tak paralelní. Další možností ukládání dat, která se nabízela, byla databáze. Nakonec jsme si řekli, že databáze je zbytečně velký nástroj,

jehož potenciál bychom nevyužili. Dalšími variantami byly například ukládání do souborů nebo zasílání každé *response* hned klientovi a nechat práci na něm. I v těchto případech by se mohlo stát, že po spuštění více testů najedou, by se ovlivňovaly. Po zaslání *requestu* si do mapy pod klíčem *testId* uložíme zprávu, kterou jsme nyní poslali. Následně zvýšíme index o 2, protože když posíláme zprávu a zároveň nám jedna přišla, tak děláme vlastně dva kroky v *transcriptu* najednou. Na následující Ukázce kódu č. 5 můžeme vidět, jak se vyhodnocuje vrácení chyby.

```
if (
  JSON.stringify(
    conversationWithLastMessage[conversationWithLastMessage.length - 1]
      .message
  ) !== JSON.stringify(omitTranscript)
) {
  const conversationWithTimeDiffTranscript =
    getConversationWithTimeDiffTranscript(
      conversationWithLastMessage,
      transcript
    )(conversationWithLastMessage);

  mainRes.json({
    error: `JSON message not equals JSON from transcript`,
    data: {
      apiUrl,
      index,
      transcript,
      conversation: conversationWithTimeDiffTranscript,
    },
  });
  res.sendStatus(200);
}
```

Ukázka kódu 5: Vrácení chyby

Když je další zpráva v transcriptu příchozí, zkontrolujeme zda se tato zpráva rovná zprávě, která má podle scénáře přijít. Jestliže se neshodují, hlavní *request* se ukončí a pošle *response* s chybou. A my tak hned poznáme, v čem náš chatbot nefunguje. Vidíme, co nám vrátil, co jsme očekávali a můžeme se na to hned v opravě zaměřit. Algoritmus porovnání porovnává celé JSON objekty. Ze začátku se porovnávali jen zprávy, ale z další analýzy vyplynulo, že bude nejlepší si ukládat celé *response* z bota a porovnávat ho s celým objektem z transcriptu. Tento objekt ze scénáře musel být ořezán. Několik atributů není totiž třeba porovnávat, protože slouží jen jako dočasná nebo informativní data pro danou zprávu. Pomocí JSON.stringify si převedeme celý objekt zprávy na text a porovnáme se scénářem. Všechny tyto informace pošleme ve výsledném *response*, takže nám klientská aplikace může zobrazit jejich rozdíl, a tak lehce opravíme strukturu chatbota, aby fungoval jako dříve.

Když jsou zprávy stejné, aplikace pokračuje dál a zkontroluje, zda už jsme nedošli na konec scénáře. Jinak bychom čekali dál na zprávu, která by nikdy nepřišla. Když jsme na konci, vrátíme hlavní *response* z prvního *requestu*, ve kterém je zpráva

o výsledku testu a data. V nich najdeme konverzaci, kterou pak můžeme vykreslit na Front End (FE), a *transcript*, dle kterého komunikace probíhala. Když nám aplikace skončí chybou, můžeme se podívat, co bylo dál v plánu, a kde se zprávy nerovnaly. Další možností, která by se mohla do budoucna implementovat je, že by se při nerovnosti zpráv uložila jak originální, tak příchozí zpráva a testování by pokračovalo dál. Do *response* by pak šel konečný počet chyb a test by se nemusel opakovat několikrát, i když by tam bylo nesrovnalostí více. Dalším efektivním přínosem by byla například cílená změna definice chatbota. Chceme tak otestovat, zda oproti minulému scénáři odpovídá v některých místech jinak. Na to by se právě použil mechanismus, který byl popsán výše. Ten by se nezastavil u první chyby, ale vypsal by je všechny najednou. Dále by mohla přibýt možnost tento *transcript* nahradit tím nově vygenerovaným testem, a to v případě, že s výsledkem souhlasíme. Výběr *transcriptů* pro jednotlivé uživatele se bude vytvářet později v další fázi vývoje této aplikace.

Když nám přijde zpráva od chatbota, je potřeba v další podmínce kontrolovat, zda ve scénáři na indexu plus jedna je také příchozí zpráva. Když ano, tak jen tu první zprávu zapíšeme do mapy, ukončíme *request* a čekáme na další. Když je ale po příchozí zprávě další zpráva odchozí zprávou, tak už by nám nic nepřišlo. Musíme tedy do mapy zapsat příchozí zprávu a následně hned odeslat odchozí zprávu, kterou zase zapíšeme do mapy a index přičteme o 2, protože zase děláme dva kroky najednou. Podporu pro dvě po sobě jdoucí zprávy od uživatele nemáme, protože ve všech chatbotech má uživatel možnost napsat pouze jednu zprávu, nebo kliknout jen na jednu možnost, pak už zase na řadě chatbot, který může zasílat kolik zpráv za sebou chce. S tím je v aplikaci počítáno. Na následující Ukázce kódu č. 6 vidíme, co se děje, když je na řadě odchozí zpráva. Funkce *responseErrorIfNotEqualsMessages* vyhodnotí, zda se zprávy rovnají. Další podmínka zkontroluje, zda přijde další zpráva z bota. Když ne, zkontroluje, zda není na řadě zpráva od uživatele a případně jí pošle botovi. Poslední podmínka *isEndOfTranscript* vyhodnotí, zda už jsme na konci *transcriptu* a ukončí porovnávání.

```
else if (transcript[index]?.Direction === OUT) {
      const omitReqBody = omitAttributesForEquals(req.body);

      responseErrorIfNotEqualsMessages(temporaryData, text, res, omitReqBody);

      if (transcript[index + 1]?.Direction === OUT) {
        setDataAndClearTimeoutSendRes(
          testId,
          temporaryData,
          res,
          1,
          appendMessage({
            direction: OUT,
            message: omitReqBody,
          })
        );
      } else if (transcript[index + 1]?.Direction === IN) {
        const appendOutInMessages = compose(
```

```
        appendMessage({
          direction: IN,
          text: transcript[index + 1]?.Text,
        }),
        appendMessage({
          direction: OUT,
          message: omitReqBody,
        })
      );

      await sendMessage(
        apiUrl,
        transcript[index + 1]?.Text,
        testId,
        req.headers.host
      );
      setDataAndClearTimeoutSendRes(
        testId,
        temporaryData,
        res,
        2,
        appendOutInMessages
      );
    }

    if (isEndOfTranscript(transcript, index)) {
      responseSuccess(temporaryData, res, text, req.body);
    }
  }
```

Ukázka kódu 6: Odchozí zpráva

Další, co chceme kontrolovat, je čas zpráv. Mohlo by se nám stát, že čekáme příchozí zprávu od bota, která ovšem nikdy nepřijde, ale naše aplikace bude stále čekat. Kvůli tomu by se nikdy *response* neposlala a FE by se nedozvěděl, kde se stala chyba. Další motivací pro *timeout* je, že chceme sledovat rychlost příchozích zpráv. Díky *transcriptu* si vypočítáme, jak rychle zprávy chodí. Čas *timeoutu* máme nastavený v konstantě. Tento *timeout* spustíme při prvotním *requestu*, který začíná naší testovací konverzaci. Protože je uložený jako atribut v mapě, běží na pozadí, i když aplikace čeká na příchozí zprávu. Před každým uložením zprávy do mapy, ať už odchozí nebo příchozí, se zavolá *clearTimeout* a nastaví se nový, který se zase uloží zpátky do mapy a zase čeká, zda bude server bota odpovídat. Na následující Ukázce kódu č. 7 můžeme vidět funkci *responseSuccess*, která se zavolá po té, co dojdeme ve scénáři nakonec.

```
const responseSuccess = (temporaryData, res, text, message) => {
  const { apiUrl, index, transcript, mainRes, conversation } = temporaryData;

  const conversationWithLastMessage = appendMessage({
    direction: OUT,
```

```
    text,
    message,
  })(conversation);
  const conversationWithTimeDiffTranscript =
    getConversationWithTimeDiffTranscript(
      conversationWithLastMessage,
      transcript
    )(conversationWithLastMessage);

  mainRes.json({
    info: "Testing was successful.",
    data: {
      apiUrl,
      index,
      transcript,
      conversation: conversationWithTimeDiffTranscript,
    },
  });
  res.sendStatus(200);
};
```

Ukázka kódu 7: Konec scénáře

Jak už bylo řečeno výše, bylo potřeba počítat časový rozdíl mezi zprávami příchozími a zprávami z *transcriptu*. Díky tomuto výpočtu můžeme vidět, jak rychle přichází zprávy z bota s porovnáním s rychlostí, jak přicházeli při tvorbě *transcriptu*. Můžeme tedy porovnat rychlost jednotlivých nasazených instancí, protože někteří boti běží na serverech u zákazníka a jeho pomalé reakce by mohli svádět na naši platformu. Samotný výpočet je složen ze dvou kroků a děláme ho vždy těsně před ukončením hlavního *requestu*. Nejdříve vypočítáme rozdíl mezi příchozími zprávami z *transcriptu* a uložíme do každé zprávy zvlášť. Stejně porovnáme i zprávy z konverzace a uložíme jejich rozdíly do daných zpráv. V druhém kroku vezmeme jejich rozdíly a odečteme je od sebe a tak nám vznikne časový rozdíl mezi scénářem a právě testovanou konverzací. Na následující ukázce kódu č. 8 můžeme vidět jak tyto funkce vypadají.

```
const addTimeDiff = (conversation) =>
  mapIndexed((item, idx) => {
    if (directionP(item) === OUT && directionP(conversation[idx - 1]) === OUT) {
      const timeDiff =
        new Date(item.timestamp) - new Date(conversation[idx - 1].timestamp);
      return { ...item, timeDiff };
    }
    return item;
  });

const addTimeDiffBetweenTranscript = (transcript) =>
  mapIndexed((item, idx) => {
    if (timeDiffP(item)) {
      const timeDiffTranscript = timeDiffP(item) - timeDiffP(transcript[idx]);
```

```
      return { ...item, timeDiffTranscript };
    }
    return item;
  });

const getConversationWithTimeDiffTranscript = (conversation, transcript) =>
  compose(addTimeDiffBetweenTranscript(transcript), addTimeDiff(conversation));
```

Ukázka kódu 8: Časový rozdíl

Dalším tématem, který bude ještě rozebrán podrobněji jsou integrace. Je to hlavní výhoda oproti konkurenčním nástrojům, které je neumožňují testovat. Feedyou platforma umožňuje i integrace zákazníkovi na míru, ale my jsme se zatím zabývali pouze těmi nejčastějšími, což jsou http integrace. Jako první krok pro implementaci této podpory testování integrací musela být úprava do Feedyou platformy. Při analýze problému testování integrací jsme nalezli řešení, které se nakonec zdálo nejlepším. V první řadě se muselo implementovat logování integrací. Logování zpráv, neboli *transcript*, již byl hotový. O něm je v této práci už hovořeno mnohokrát. Integrace se nyní tedy logují také do databáze a můžeme si je pro danou konverzaci vytáhnout stejně jako *transcript*. Na následující Ukázce kódu č. 9 můžeme vidět integrační *transcript*. Logujeme si metodu poslání, *request* body a headers, *response* body a status. V *response* body vidíme to, co přišlo jako odpověď při provolání URL, která se také ukládá do logovacího scénáře.

```
{
  "PartitionKey": "537d2d45-afff-41ea-a55f-67724fb991a1",
  "RowKey": 8351493524882039,
  "Timestamp": "2022-03-28T22:27:55.674Z",
  "Method": "GET",
  "RequestBody": "",
  "RequestHeaders": {},
  "ResponseBody": {
    "userId": 1,
    "id": 1,
    "title": "delectus aut autem",
    "completed": false
  },
  "ResponseStatus": 200,
  "Url": "https://jsonplaceholder.typicode.com/todos/1"
}
```

Ukázka kódu 9: Integrační transcript

Po jeho dotažení do bot testera si spojíme *transcript* zpráv s logem integrací a seřadíme dle timestampu. Tím nám vznikne scénář obohacený o integrace. Další úpravou ve Feedyou platform, kromě logování integrací, byla ještě úprava integračního kroku. Při zahájení testování konverzace voláme event *beginIntroDialog*. Jak jste si mohli všimnout v předchozím kódu, tělo *requestu* obsahuje ještě atribut *sourceEvent*, ve kterém se nachází *mockIntegration*, jehož hodnota je *true*. Jak již plyne z jeho názvu, slouží pro mockování integrací. Při testu není nutné volat znova integraci, protože jí

máme již uloženou ve scénáři. Chceme testovat bota a ne integraci třetí strany. Pro případ otestování integrace třetích stran přidáme parametr, který nastaví *mockIntegration* na *false*, a integrace se bude reálně provolávat i při testování konverzace. V integračním kroku ve Feedyou platform to funguje tak, že když přijde *mockIntegration*, která je nastavená na *true*, http krok zavolá testera na zaslanou *serviceUrl* a ten mu zpátky vrátí status a body z *transcriptu*. Bot následně běží dál a pokračuje v konverzaci. Tuto odpověď z integrace si označíme jako type *integration*, abychom ho pak mohli na FE odlišit od běžných zpráv z bota.

Jako úložiště dočasných dat byla využita běžná javascriptová mapa, která má funkce get, set atd., aby nemuselo být pracováno s databází, která v tomto případě není potřeba. Mapa slouží pouze jako úložiště dočasných dat pro vícevláknové zpracování testů. Díky průběžnému ukládání zpráv dle unikátního *testId* může běžet více testů najedou a nijak se neovlivňují. Jako zabezpečení provolání *requestů* bota slouží klíč, který musí mít aplikace nastavené v souboru .env.

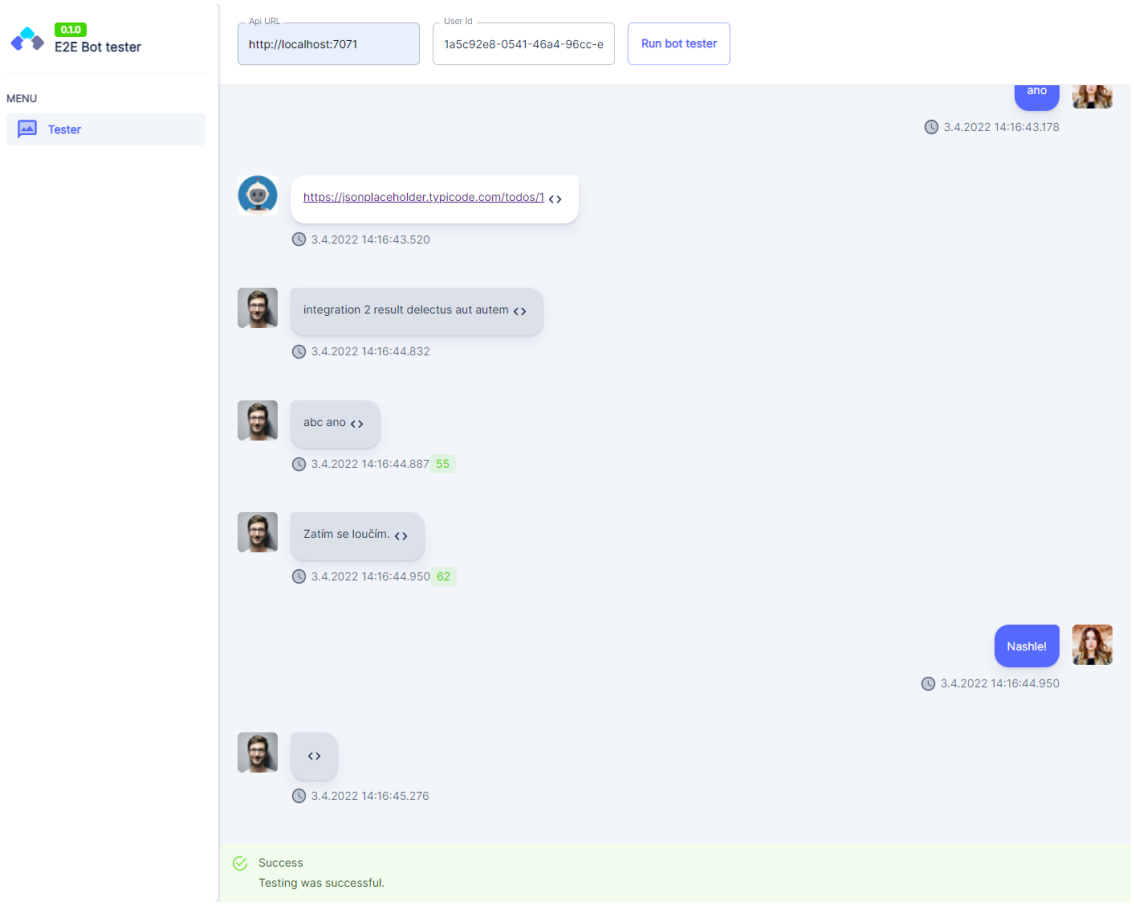
Na závěr této kapitoly si ještě popíšeme, jak je potřeba volat naší testovací aplikaci. Metodou POST provoláme *request* na adresu start?apiUrl=umístění chatbota a userId=unikátní id konverzace. Jako odpověď na tento *request* přijde JSON, ve kterém najdeme *transcript*, komunikaci a informace o výsledku testu.

5.5 Popis funkčnosti Front End

Frontendová aplikace je napsaná pomocí Reactu a TypeScriptu. React byl vybrán, protože s ním již máme bohaté pracovní zkušenosti. Tato aplikace je hlavně pro uživatele k interpretování výsledků. Byla vytvořena s pomocí knihovny material-ui, která má většinu potřebných komponent pro složení aplikace. Tato knihovna je nyní velice oblíbená a rozšířená mezi programátory. Slouží hlavně těm, kteří nechtějí ztrácet čas stylováním a opětovnou použitelností komponent. Tyto komponenty jsou už napsané v React a mohou se tak jen naimportovat do projektu a rovnou používat a vývojář se tak může plně soustředit na funkcionalitu aplikace.

Na začátku aplikace máme dvě pole, do kterých zadáme *apiUrl* a *userId*, následně klikneme na tlačítko vpravo, které provolá náš BE a otestuje chatbota, podle podmínek, které byly popsány výše. Při čekání na odpověď se nám zobrazí *loader* v modálním okně. Po přijetí *response* zobrazíme konverzaci a pod ní výrok s výsledkem testu. Původně jsme chtěli, aby aplikace zobrazovala obrázky a ovládací prvky stejně jako WebChat, ale po hlubší analýze jsme zjistili, že by to bylo hodně implementačně náročně a hlavně úplně zbytečné. Rozhodli jsme se ukazovat jen texty zpráv. Každá zpráva má na svém konci tlačítko s ikonou kódu, po kliknutí na ikonu kódu se nám zobrazí modální okno. V něm máme tři záložky. Originální *transcript*, originální zpráva a porovnání těchto dvou *JSONů*. Chat se zobrazí celý najedou, není v tomto případě potřeba simulovat živou konverzaci, protože tento FE slouží pouze pro výsledky testů. Dále vidíme u každé zprávy datum a čas, kdy přišla a časový rozdíl mezi tím, který byl u zprávy v transcriptu a tím, kdy reálně při testování přišel. Díky tomu můžeme testovat i rychlost prostředí, na kterém je bot spuštěn. Na Obrázku 14 můžeme vidět ukázku úspěšného testu konverzace. Ikona robota znamená

provolání integrace, v jehož těle můžeme vidět provolanou URL adresu. Podrobnosti o integraci si můžeme zobrazit při kliku na ikonu kódu v pravé části zprávy.

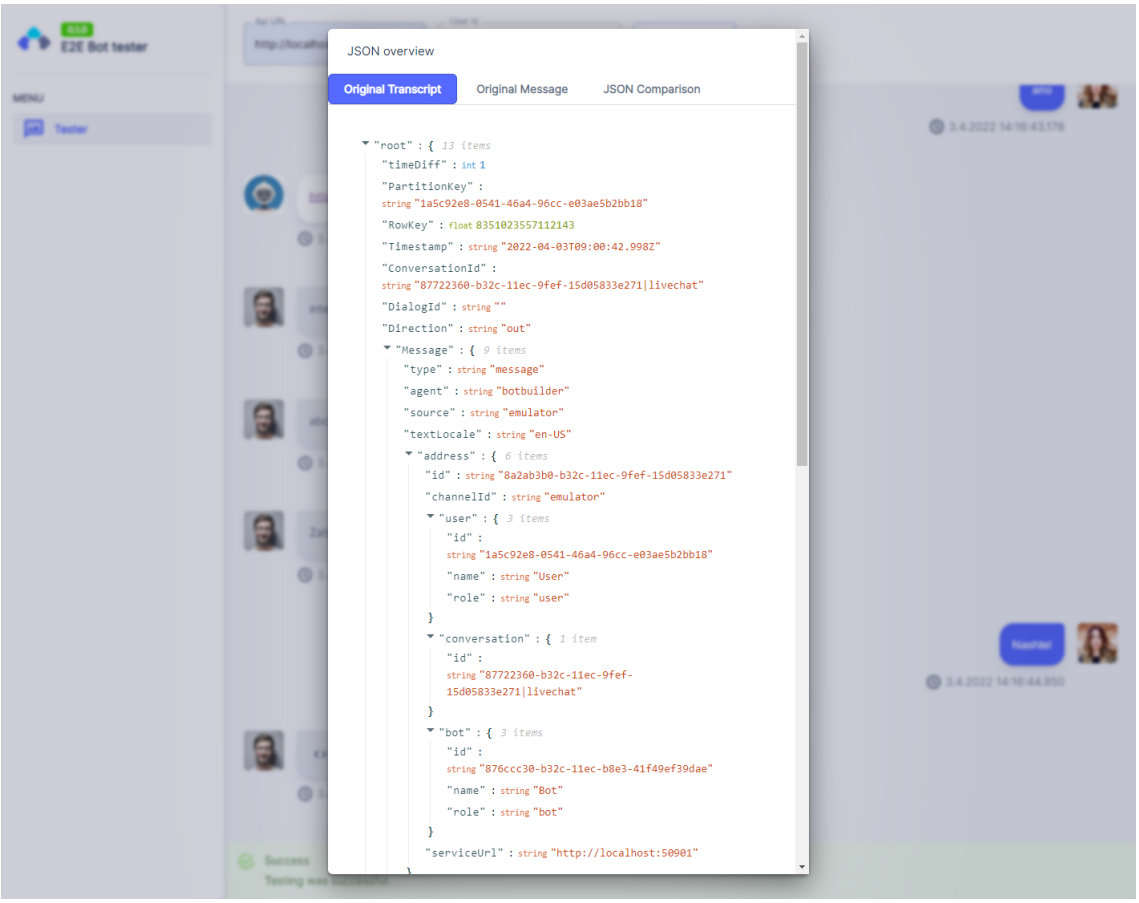


Obrázek 14: Ukázka úspěšného testu, zdroj: autor

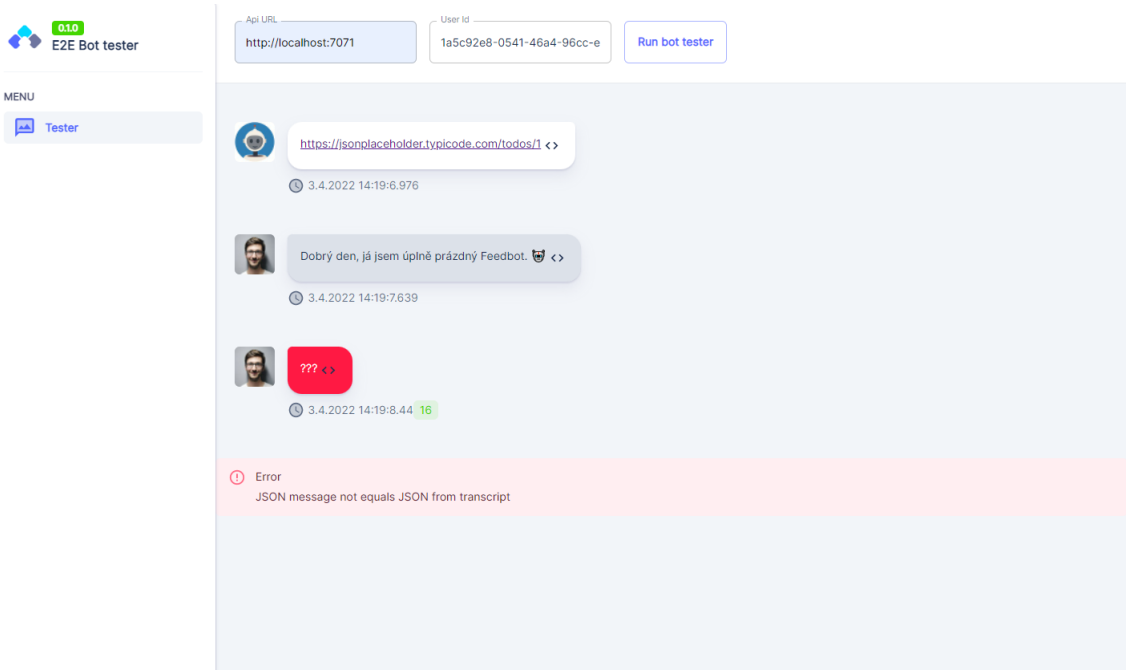
V případě, že si chceme prohlédnout JSON transcriptu, u jednotlivých zpráv, můžeme kliknout na ikonu kódu v pravé části zprávy. Po jehož rozkliknutí se nám zobrazí modální okno, jak můžeme vidět na Obrázku 15. V případě zobrazení originální zprávy, která přišla při testování bota, se stačí překliknout na záložku *original message*.

Na Obrázku 16 si ukážeme případ, kdy se zprávy nerovnají. V tomto případě přijde z BE chyba a zčervená poslední zpráva. Na Obrázku 17 můžeme vidět porovnávaní JSONů, kde je na první pohled patrné, že přišla jiná zpráva, než byla očekávána. Červená značí, co reálně přišlo, a zeleně je zpráva, která byla očekávána. Přepínačem show si můžeme vypnout ostatní atributy, které jsou stejné. Může se stát, že u složitějších zpráv bude atributů k porovnání mnohem více a přepínač nám může pomoci v přehlednosti.

V další fázi vývoje aplikace bude nasazen zvlášť BE a FE. Serverová část aplikace se bude používat na testování botů v různých prostředích. Aby mohl být využit úplný potenciál této aplikace, zavedeme ho do Feedyou platformy. Bude v designeru bota,

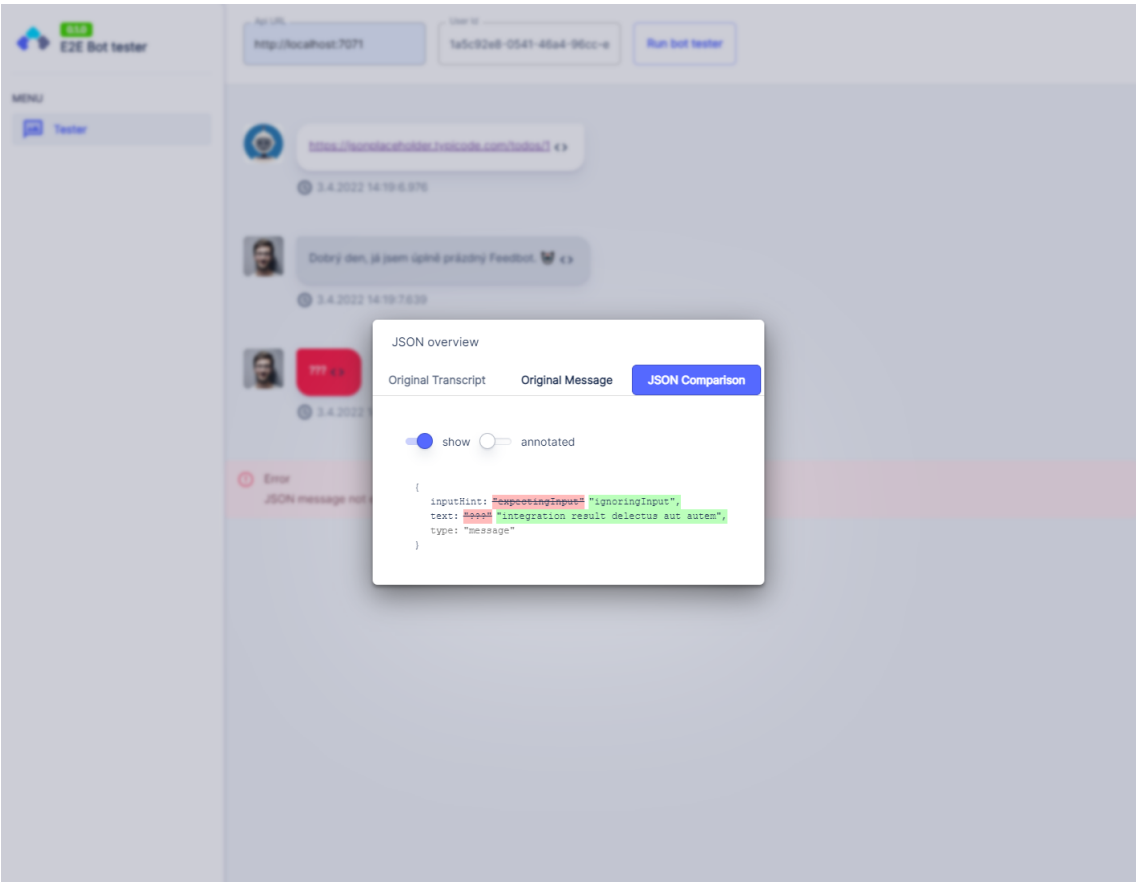


Obrázek 15: Ukázka JSON transcriptu, zdroj: autor



Obrázek 16: Ukázka neúspěšného testu, zdroj: autor

kde si již nyní můžeme stavět konverzační strom. Tato implementace tak přinese uživatelům designéru nové možnosti testování bota. Dříve to šlo jen ručně, což bylo hodně zdlouhavé. Pro daného chatbota si vytvoříme několik testovacích scénářů, které se provolají při každé změně chatbota. V tomto seznamu bude vidět, zda připravené scénáře stále procházejí dle očekávání. V případě, že nějaký test spadne, budeme se moci prokliknout do FE, kde již uvidíme rozdíl mezi *transcriptem* a originální zprávou, jako můžeme vidět na Obrázku 17.



Obrázek 17: Ukázka porovnání JSONů, zdroj: autor

6 Souhrn výsledků

Testování botů obecně zabírá mnoho času, který by mohli využít uživatelé Feedyou platformy efektivněji. Nyní si představíme hypotézu o skutečné úspoře času při zavedení nového systému automatického testování v reálné výrobě chatbotů v naší společnosti. Obecně by se po každé změně bota měly přetestovat všechny jeho větvě, zda odpovídá stále tak, jak očekáváme, ale na to většinou nezbývá čas, a chyby se odhalí až na produkci, což už stojí zákazníka peníze a naší společnost dobrou pověst. V následující tabulce si ukážeme, kolik času měsíčně stojí testování. Data v této tabulce pochází ze statistik našeho výrobního oddělení. Zkratka Full Time (FT) znamená plný úvazek.

Tabulka 1: Manuální vs. automatické testování, zdroj: autor

čas testování(v hod)	čas vyhodnocení automatického testu (v hod)	počet botů	manuální testování za měsíc	automatické testování za měsíc
5	0.5	12	60	6
10	1	14	140	14
20	2	20	400	40
30	2.5	24	720	60
40	3	16	640	48
60	4	14	840	56
celkem		**100**	**2800**	**224**
počet úprav za měsíc		1.8	5040	403.2
počet FT			30	2.4
úspora FT	**27.6**			

Boty jsme rozdělili podle složitosti. Jednotlivé řádky představují kategorie botů, které jsme seřadili podle komplexnosti testování. V prvním sloupci můžeme vidět čas v hodinách za měsíc, který se stráví testováním. V dalším sloupci můžeme vidět, kolik zabere času práce s automatickým testováním, vytváření scénářů a jejich přepis. Vedle je vidět aktuální počet botů v naší společnosti podle jejich komplexnosti. Z toho nám vyjde další sloupce, kde jsou hodiny za manuální testování za měsíc při jedné úpravě na každém botovi a v posledním sloupci jsou hodiny, které se budou muset věnovat automatickému testování za měsíc při jedné úpravě na každém botovi. Další důležitá proměnná je počet úprav měsíčně na jednotlivých botech. V průměru je to 1.8 úpravy za měsíc. Vynásobíme-li vzájemně jednotlivé důležité položky, snadno zjistíme, že testování chatbotů musí pro naši společnost každý měsíc na celý svůj úvazek dělat 30 osob. Porovnáme-li pak tento výsledek s položkami týkajícími se časové dotace, kterou lidé stráví testováním při zavedení automatického testování chatbotů, snížíme hodnotu o více než 27 osob na úplný úvazek. Tento markantní rozdíl tedy umožní minimálně 27 lidem celý svůj pracovní úvazek věnovat tvořivějším věcem a testovací rutinu přenechat nové aplikaci.

7 Závěry a doporučení

Předkládaná diplomová práce si kladla za svůj hlavní cíl přehledně shrnout základní poznatky z oblasti testování chatbotů a dále také ve své praktické části přiblížit vlastní implementaci aplikace E2E Bot tester, která právě tuto oblast efektivně řeší. I přesto, že nejsou chatboti již úplně novou technologií, jedná se o velmi aktuální téma úzce spojené v současnosti s velmi diskutovaným tématem úspory rutinně stráveného času testujících osob. Jednotlivé kapitoly práce byly zaměřeny nejen na historii sledované oblasti, ale i na zpracování přirozeného jazyka, umělé neuronové sítě, jejich silné a slabé stránky a zejména na samotný proces testování chatbotů.

Jedním z dílčích cílů práce bylo též porovnání aktuálně dostupných nástrojů pro testování a zhodnocení jejich přínosů a negativ oproti novému řešení, jehož implementace v rámci společnosti Feedyou je v práci popisována. Hlavním konkurentem tohoto řešení byla aplikace Botium, u níž se však ukázalo, že nesplňuje jeden z hlavních požadavků společnosti Feedyou, což je testování integrací.

Tyto integrace jsou důležitou součástí stromové struktury chatbotů, protože velká část botů je napojena na nějaký externí systém, z kterého si načítá různá data. Hlavním přínosem E2E Bot testeru je, že úzce spolupracuje s vlastním řešením Feedyou platformy, která je postavená na Microsoft Azure Bot Service. Díky tomu si může při testování vytáhnout scénáře integrací a následně je testovat, zda vracejí očekávaný výsledek. Možné je tyto odpovědi brát ze scénáře i v případě, že nechceme provolávat rozhraní třetích stran.

Nová aplikace má využití nejen u chatbotů, ale také u voice botů, které nyní aktivně vyvíjíme a brzo budou součástí Feedyou platformy. Protože se hlasoví boti překládají na text a komunikace se ukládá stejně jako u chatbota, můžeme pouštět testy i pro hlasové boty.

Implementací vlastní funkční a efektivní aplikace na míru společnosti Feedyou byl cíl práce splněn. Hlubší analýzou bylo zjištěno, že žádné z jiných dostupných řešení by v plné míře nevyhovovala deklarovaným potřebám a testování by nebylo účinné tak, jak je v praxi třeba. Aplikace bude následně integrována do Feedyou platformy a nabízena jako její součást po celém světě přes Microsoft Store. To znamená, že při současném trendu rozrůstání chatbotů, může aplikace v budoucnu ušetřit obrovské množství času firmám po celém světě. Tím, že chatboti budou stále odpovídat správně, se bude navyšovat spokojenost zákazníků a zisky firem, které virtuální asistenty používají.

Literatura

[1] Treml, F. How to Test a Chatbot — Part 3: The Quest for Test Cases. 2021. Available from: https://chatbotsmagazine.com/how-to-test-a-chatbot-part-3-the-quest-for-test-cases-cb7a90238a40

[2] Dale, R. The return of the chatbots. volume 22, no. 5, 2016: pp. 811–817, ISSN 1351-3249, 1469-8110, doi:10.1017/S1351324916000243, publisher: Cambridge University Press. Available from: https://www.cambridge.org/core/journals/natural-language-engineering/article/return-of-the-chatbots/0ACB73CB66134BFCA8C1D55D20BE6392

[3] Perez-Soler, S.; Juarez-Puerta, S.; Guerra, E.; et al. Choosing a Chatbot Development Tool. volume 38, no. 4, 2021: pp. 94–103, ISSN 0740-7459, doi:10.1109/MS.2020.3030198, place: Los Alamitos Publisher: Ieee Computer Soc WOS:000664984000012. Available from: https://www.webofscience.com/wos/woscc/full-record/WOS:000664984000012

[4] Adamopoulou, E.; Moussiades, L. Chatbots: History, technology, and applications. volume 2, 2020: p. 100006, ISSN 2666-8270, doi:10.1016/j.mlwa.2020.100006. Available from: https://www.sciencedirect.com/science/article/pii/S2666827020300062

[5] Procházka, F. Voicebot - Virtuální hlasový asistent. 2022. Available from: https://feedyou.ai/cs/voicebots/

[6] Michal, B. Implementace hlasového asistenta mobilního telefonu pro nevidomé. 2017, accepted: 2017-06-07T13:22:07Z Publisher: České vysoké učení technické v Praze. Vypočetní a informační centrum. Available from: https://dspace.cvut.cz/handle/10467/68599

[7] Best Bot Platforms Software 2022: Compare Reviews on 160+. 2022. Available from: https://www.g2.com/categories/bot-platforms

[8] Lennartsson, R.; Edqvist, J. *Chat Bots & Voice Control : Applications and limitations of combining Microsoft's Azure Bot Service and Cognitive Services' Speech API*. 2019. Available from: http://urn.kb.se/resolve?urn=urn:nbn:se:kau:diva-72450

[9] Jonathan Fingold. Test and debug bots using the Bot Framework Emulator - Bot Service. 2021. Available from: https://docs.microsoft.com/en-us/azure/bot-service/bot-service-debug-emulator

[10] Botanalytics | AI powered Chatbot Analytics and Voice Analytics. 2022. Available from: https://botanalytics.co

[11] Martín, J. Chatbottest – bezplatný průvodce, který vám pomůže pochopit, co váš chatbot dělá špatně. 2022. Available from: https://chatbottest.com/#learn-more

[12] Dimon. Dimon - Bot Performance Assurance. 2022. Available from: https://dimon.co

[13] Bravo-Santos, S.; Guerra, E.; de Lara, J. Testing Chatbots with Charm. In *Quality of Information and Communications Technology*, edited by M. Shepperd; F. Brito e Abreu; A. Rodrigues da Silva; R. Pérez-Castillo, Communications in Computer and Information Science, Springer International Publishing, 2020, ISBN 978-3-030-58793-2, pp. 426–438, doi:10.1007/978-3-030-58793-2_34.

[14] Treml, F. How to Test a Chatbot — Part 2: Superpowers for the testers. 2021. Available from: https://chatbotsmagazine.com/how-to-test-a-chatbot-part-2-superpowers-for-the-testers-500cf6699b41

[15] Kinsbruner, E. Testing Conversational AI — Botium documentation. 2022. Available from: https://botium-docs.readthedocs.io/en/latest/03_testing/01_testing_conversational_ai.html

[16] Wilson, D. Express application generator. 2021. Available from: https://expressjs.com/en/starter/generator.html

[17] de Oliveira, D. Tokyo Free White Typescript Dashboard - BloomUI. 2022. Available from: https://bloomui.com/product/tokyo-free-white-react-typescript-material-ui-admin-dashboard/

[18] Garcia, B.; Gortazar, F.; Gallego, M.; et al. User Impersonation as a Service in End-to-End Testing. In *Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development*, edited by S. Hammoudi; L. Pires; B. Selic, Scitepress, 2018, ISBN 978-989-758-283-7, pp. 707–714, doi:10.5220/0006752207070714, WOS:000570978800073. Available from: https://www.webofscience.com/wos/woscc/full-record/WOS:000570978800073

[19] Jonathan Fingold. Debug your bot using transcript files - Bot Service. 2021. Available from: https://docs.microsoft.com/en-us/azure/bot-service/bot-builder-debug-transcript

Seznam zkratek

AI Umělá inteligence

API Application Programming Interface

B2B Business-to-Business

BE Back End

CD Continuous Deployment

CEE Central and Eastern Europe

CI Continuous Integration

CORS Cross-Origin Resource Sharing

CX Customer Experience

E2E end-to-end

FE Front End

FT Full Time

GDPR General Data Protection Regulation

HR Human Resources

HTTP Hypertext Transfer Protocol

IBM International Business Machines Corporation

ICT Information and Communications Technology

IDC International Data Corporation

IKEM Institut klinické a experimentální medicíny

IoT Internet of Things

JSON JavaScript Object Notation

LSTM Long Short Term Memory

LUIS Language Understanding

MSN Microsoft Network

NER Named entity recognition

NLP natural language processing

NPS Net Promoter Score

PPL Professional Parcel Logistic

RNN Recurrent Natural Networks

ROI Return on Investment

SaaS Software as a service

SMS Short message service

URL Uniform Resource Locator

VA Virtuální asistent

WHO World Health Organization

ČRA České radiokomunikace

Přílohy

Zabalený adresář v zip obsahuje zdrojové kódy celé aplikace. Aplikace je rozdělená na BE část a FE část, které běží jako samostatné aplikace. Struktura je následující:

- client - adresář s FE React aplikací
- client/src - adresář, který obsahuje zdrojové kódy klientské aplikace
- server - adresář s BE Express aplikací
- server/bin - obsahuje inicializační soubor aplikace pro její start
- server/routes - obsahuje soubory s routy, které je možné volat
- server/src - obsahuje zdrojové kódy, které se používají v routes

Kód je také dostupný na platformě GitHub.
`https://github.com/martinsmall12/e2eBotTester`

UNIVERZITA HRADEC KRÁLOVÉ
Fakulta informatiky a managementu
Akademický rok: 2021/2022

Studijní program: Aplikovaná informatika
Forma studia: Kombinovaná
Obor/kombinace: Aplikovaná informatika (ai2-k)

Podklad pro zadání DIPLOMOVÉ práce studenta

Jméno a příjmení: **Bc Martin Malý**
Osobní číslo: **I2100437**
Adresa: Nábřeží Závodu míru 1885, Pardubice – Zelené Předměstí, 53002 Pardubice 2, Česká republika

Téma práce: E2E Bot tester
Téma práce anglicky: E2E Bot tester

Vedoucí práce: Ing. Pavel Kříž, Ph.D.
Katedra informatiky a kvantitativních metod

Zásady pro vypracování:

Cíl: Navrhnout a implementovat nástroj pro automatizované end-to-end testování chatbotů. Nástroj bude schopen mj. zaznamenat konverzaci s chatbotem a vytvořit z jejího průběhu testovací scénář. Následně scénař otestuje vč. případného volání externích systémů.
Obsah

- 1 Úvod
- 2 Cíl práce
- 3 Chatboti
- 4 Společnost Feedyou
- 5 Implementace testovacího nástroje E2E bot tester
- 6 Souhrn výsledků
- 7 Závěry a doporučení
- Literatura
- Seznam zkratek
- Přílohy

Seznam doporučené literatury:

- https://botanalytics.co/
- https://chatbottest.com/
- http://dimon.co/
- Amir Shevat: Designing Bots: Creating Conversational Experiences

Podpis studenta: Datum:

Podpis vedoucího práce: Datum: