VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

REZERVACE VSTUPENEK POMOCÍ BOTŮ NA CHATOVACÍCH PLATFORMÁCH

CHAT BOT BASED TICKET RESERVATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **TOMÁŠ CHOVANEC**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JAN PLUSKAL**
SUPERVISOR

BRNO 2017

Abstrakt

V súčasnosti sa čoraz viac do povedomia odbornej, ale aj laickej verejnosti dostáva pojem chat bot. Táto práca má za cieľ definovať daný výraz, preskúmať dostupné platformy, ich vývojové nástroje a následne sa pokúsiť implementovať prvky pokročilej funkcionality akými sú napríklad asistenčné služby alebo platobné brány a to v kontexte procesu rezervácie vstupeniek.

Abstract

Today, people are starting to get familiar with the term of chat bots and it makes no difference if they are specialists or general public. The aim of this thesis is to define chat bots as a web service, describe platforms which use this service and try to develop a connection for advanced functions, for example online payment gateways or virtual assistants, to extend current ticket booking chat bot services.

Kľúčové slová

chat bot, Microsoft Bot Framework, proces rezervácie vstupeniek, inteligentný asistent, Cortana, platobná brána, PayPal, internetový kalendár, Google Calendar, kognitívne rozpoznávanie

Keywords

chat bot, Microsoft Bot Framework, ticket reservation, virtual assistant, Cortana, online payment solutions, PayPal, calendar services, Google Calendar, cognitive recognition

Citácia

CHOVANEC, Tomáš. *Rezervace vstupenek pomocí botů na chatovacích platformách.* Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Pluskal Jan.

Rezervace vstupenek pomocí botů na chatovacích platformách

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Jana Pluskala. Uviedol som všetky literárne zdroje a publikácie, z ktorých som čerpal.

.....
Tomáš Chovanec
18. mája 2017

Poďakovanie

Týmto by som sa chcel poďakovať špecialne svojmu vedúcemu Ing. Janovi Pluskalovi, externým konzultantom Romanovi Jašekovi, Miroslavovi Kaděrovi a tiež všetkým programátorom z firmy RIGANTI, s ktorými som prisiel do kontaktu pri konzultáciách.

Obsah

Kapitola 1

Úvod

Ak chce v dnešnej dobe človek používať ktorýkoľvek dostupný program, aplikáciu alebo aj operačný systém, musí sa s ním najprv naučiť pracovať a oboznámiť sa so všetkými funkciami, ktoré poskytuje. Nadnesene povedané používateľ sa musí prispôsobiť komunikačným schopnostiam programu a nie naopak. Je však možné tento proces otočiť a nechať program naučiť sa komunikačným schopnostiam človeka? Nejde tu pritom iba o spôsob akým sa počítač naučí reč alebo konkrétny jazyk. Takto navrhnutý program by musel vedieť komunikovať tak, aby to pre človeka bolo prirodzené ako rozprávať sa s inou osobou.

Problematikou programov, ktoré vedia simulovať človeka ako partnera v konverzácii, sa ako prvý začal zaoberať Alan Turing. V roku 1950 popísal kritériá takzvaným Turingovým testom [23]. Ten hovorí o testovacom prostredí v podobe anonymnej komunikácie kde na jednom konci skúšajúci pokladá otázky dvom oponentom z ktorých je jednen človek a druhý program simulujúci človeka. Test je úspešny vtedy keď skúšajúci nevie určiť rozdiel medzi človekom a programom.

Na základe Turingovho testu by sa mohlo zdať, že program, ktorý vie komunikovať ako inteligentná bytosť, spĺňa kritériá umelej inteligencie. V roku 1966 sa však ukázalo, že program nazvaný ELIZA [27], ktorý odpovedal iba predpripravenými frázami dokázal oklamať skúšajúceho a ten nevedel určiť, ktorý z oponentov je stroj a ktorý človek.

Turingov test tak bol podrobený kritike, kedže mal v prvom rade otestovať zložitosť inteligencie v algoritme. Táto udalosť však nepriamo potvrdila, že program nepotrebuje nutne obsahovať prvky umelej inteligencie aby dokázal komunikovať s používateľom v rámci istých pravidiel. Dnes je táto logika implementovaná v programovaní veľkého počtu komunikačných programov, ktoré nemusia vždy poskytovať otázku na všetky odpovede. Označujú sa názvom *chating robot* alebo skrátene *chat bot*.

1.1 Chat bot

Označenie *Chat bot* nie je oficiálnym. Bolo použité na základe výskytu v kľúčových materiáloch tejto práce [15]. Niektoré iné použité zdroje uvádzajú pojem *Chatterbot*, ktorý je použitý hlavne v súvislosti moderátora fór alebo nezávislého programu. V tejto práci však bude reč o chat botoch v IM (Instatnt messaging) službách a klientoch.

Instant messaging je služba primárne určená pre posielanie správ ale aj posielanie iného, napríklad multimediálneho obsahu v reálnom čase. To znamená, že odosielané správy sa okamžite zobrazujú adresátovi. Používatelia tiež môžu zistiť aktuálnu dostupnosť svo-

jich kontaktov, prípadne vyhľadávať iných používateľov na základe identifikátoru predom stanovených parametrov.

Na rozdiel od programu ELIZA sa môžu dnešné boty spoliehať na rôzne vývojové nástroje a strojové učenie. Vďaka tomu ich už v súčasnosti nemožno nazývať len ako jednoduchú simuláciu konverzácie na oklamanie Turigovho testu. Dnešné chat boty dokážu rozlišovať objekty na obrázkoch alebo porozumieť kontextu ľudského jazyka.

1.2 Motivácia

Motiváciou pre vytvorenie práce však nebol iba záujem o skúmanie noviniek v tejto oblasti ale najmä zistenie, že väčšína botov v dnešnej dobe nevyužíva naplno svoj potenciál. Komerčné riešenia sú väčšinou iba funkčne obmedzeným dopnkom existujúcich štruktúr zväčša vo forme internetových portálov, akými sú napríklad internetové obchody alebo konkrétnejšie pre túto prácu - systémy rezervácií vstupeniek.

Firma RIGANTI, ktorá sa podieľala na tvorbe zadania bakalárskej práce, vyvíja chat bota, ktorý by mal byť rozšírením k existujúcej webovej službe pre rezerváciu vstupeniek. Vývojári sa pritom zamerali na implementáciu základnej konverzačnej štruktúry a odladenie krajných prípadov použitia, pričom riešenie problematických častí a prípadných rozšírení bolo ponechané na túto prácu.

1.3 Cieľ práce

Práca má za cieľ preskúmať aktuálne možnosti v programovaní chat botov, predstaviť vývojové nástroje a platformy. Popísať ich možnosti, výhody aj nevýhody oproti iným. Konkrétny rozbor dostupných platforiem a ich vývojových nástrojov pre základné komunikačné prostriedky je popísaný v kapitole 2.

Okrem vstavaných nástrojov je možné implementovať aj rozšírenia tretích strán a funkcionalitu nad rámec základnej komunikácie. V kapitole 3 je preto popísaný prehľad o týchto rozšíreniach, či už formou oficiálnej podpory vo vývojových nástrojoch platformy alebo skrz aplikačné rozhrania iných služieb. Tým sa myslí aj hľadanie možností využitia vývojových nástrojov tretích strán pri ich spájaní s rozhraním chat bota.

Na základe výskumu sa vyberú najlepšie možnosti rozšírení a bude podrobne navrhnuté ich napojenie na najuniverzálnejšiu platformu alebo takú, ktorá tieto rozšírenia bude najviac podporovať. Tento postup bude odôvodnený v kapitole 5. Nakoniec budú implementované ukážkové príklady prezentujúce jednotlivé pokročilé možnosti v komunikácii chat botov.

Kapitola 2

Chatbot platformy

Ako už bolo spomenuté v prechádzajúcej kapitole súčasná situácia okolo chatovacích programov, chatbotov, je dynamicky sa vyvíjajúca oblasť informatiky. Okrem pokusov, ktoré sú určené pre vývoj a na ktorých si spoločnosti testujú pokroky v oblasti umelej inteligencie, existuje aj snaha o komerčné využitie. Aby bol vývoj čo najjednoduchší pre programátorov, jednotlivé spoločnosti vytvorili súbory vývojových nástrojov, frameworky alebo api. Tie sú väčšinou závislé na príslušnej platforme.

Používatelia k botom naprogramovaným touto cestou môžu pristupovať iba cez IM službu spoločnosti, ktorá nástroje vytvorila. Chat bot je tak obmedzený iba na jednu platformu, v ktorej je k nemu možné pristupovať tak ako, k ostatným registrovaným kontaktom, prípadne cez špeciálne rozhranie.

Medzi instant messaging služby, ktoré umožňujú komunikovať s chat botmi patria: Facebook Messenger (miliarda mesačne aktívnych používateľov), Skype (300 miliónov používateľov), Slack, Kik Messenger (300 miliónov používateľov), Telegram (viac ako 100 miliónov používateľov), WeChat (viac ako 800 miliónov používateľov), Line (200 miliónov používateľov[1]).

Ďalšie populárne platformy ako Whats App alebo Snapchat chat botov nepoužívajú buď kvôli inej aplikácii z firemného ekosystému alebo kvôli celkom odlišnej architektúre služby. BlackBerry Messenger by mal začať podporovať botov v priebehu roku 2017.

Nakoľko sú chat boti nová a rýchlo sa rozvíjajúca technológia, platí to isté aj o platformách pre ich programovanie. Priekopníkmi v tejto oblasti sú spoločnosti stojace za službami ako Telegram alebo Slack, ktoré sa v silnej konkurencii snažili prilákať k svojej službe používateľov funkciami navyše. V roku 2016 sa však o slovo prihlásili aj technologickí giganti ako Microsoft a Facebook, ktorí predstavili frameworky pre ich IM platformy.

Svoje pôsobenie v novej oblasti obhajujú pomocou počtu zákazníkov využívajúcich ich služby, ktoré nie sú závislé na žiadnom operačnom systéme. Napríklad Facebook Messenger dosiahol približne v polovici roku 2016 jednu miliardu mesačne aktívnych používateľov, za pomoci najmä mobilných aplikácií a webového klienta. Tieto čísla majú motivovať aj vývojárov aby práve na ich platformu tvorili čo najviac kvalitného obsahu.

Škála využitia jednotlivých platforiem je vysoká. Od primárne experimentálneho zamerania, kde sa využíva strojové učenie na určenie informácie z obrázkov alebo videa až po komerčnú oblasť, kde chat bot slúži ako „predajca“, ktorý zobrazuje a následne predáva konkrétny tovar alebo službu. Škála závisí od počtu možností, ktoré daná platforma ponúka.

[1] Všetky údaje o mesačne aktívnych používateľoch boli získané v januári 2017 zo zdroja: https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/

Možnosťami sú v tomto prípade myslené pokročilé formy komunikácie využívané v rámci platformy alebo iba špeciálne určené pre bota. Ako pokročilé formy komunikácie sú pritom brané aj prvky ako obrázky, webové odkazy, emotikony, stiahnuteľné súbory, ktoré sú dnes bežne využívané aj v komunikácii človek – človek v rámci komunikačných platforiem.

Pokročilejšími prvkami určenými iba pre chat bota môžu byť karty zobrazujúce štruktúrovaný obsah, ukladanie informácií o používateľoch, vytváranie platobných príkazov alebo napojenie na systémových asistentov. Detailný popis a možnosti pokročilejších prvkov komunikácie je spracovaný v kapitole 3. V ďalších častiach tejto kapitoly sú popísané frameworky jednotlivých platforiem, ich možnosti, obmedzenia.

2.1 Facebook Messenger

Služba Facebook Messenger bola pôvodne vstavanou súčasťou sociálnej siete Facebook. Postupom času sa však čoraz viac osamostatňuje v rámci produktového portfólia spoločnosti. Messenger podporuje okrem klasickej textovej komunikácie posielanie webových odkazov, emotikonov, obrázkov z lokálneho úložiska alebo fotiek vyhotovených aplikáciou, gify, hlasové záznamy, súbory a polohu.

Aplikácia tiež obsahuje pluginy pre iné služby schopné pridávať ich obsah. Sú označované ako App for Messenger. Medzi ďalšiu funkcionalitu patria telefónne hovory, video konferencie, komunikácia v skupinách, šifrovaná peer-to-peer komunikácia a iné.

Messenger Platform

V apríli 2016 pribudli do služby chat boti. Sú programovateľné pomocou vývojových nástrojov pod označením Messenger Platform [10]. Konfigurácia a registrácia nového chat bota pozostáva z vytvorenia alebo nastavenia troch častí.

- Facebook aplikácia je základná entita, ktorá definuje každého bota. Konfiguračné rozhranie obsahuje nastavenia takzvaných webhook adries, umožňuje generovanie unikátneho prístupového kódu či proces schvaľovania pred vydaním.
- Ako identita bota sa využíva profil Facebook stránky. Určuje názov, popis, profilový obrázok a iné informácie, ktorými budú používatelia program identifikovať.
- Webhook URL adresa umožňuje vytvárať bezpečné volania pre posielanie správ do rozhrania aplikácie a naopak. Volanie (anglicky callback) je vyvolané nejakou akciou. Napríklad prijatím správy od klienta.

Úspešná registrácia nového účtu vznikne po korektnom nastavení zmienených súčastí a po prejdení overovacím procesom. Na adrese webhooku následne môže začať vznikať aplikácia bota, komunikujúca skrz HTTPS protokol správami v JSON formáte.

Každá nová konverzácia s používateľom je označovaná ako vlákno, má svoj vlastný identifikátor. Na začiatku je možné nastaviť jednoduchú uvítaciu obrazovku v rámci grafického rozhrania aplikácie, prípadne po začatí konverzácie zobraziť úvodnú správu s navádzacími informáciami.

Používateľ môže odpovedať pomocou vlastných správ alebo dostane na výber z možností prezentovaných tlačidlami. Komunikácia je riadená pomocou webhook referencií. V prípade prijatia správy to môžu byť napríklad volania o prijatí fulltextovej odpovede alebo vybranej možnosti, o prečítaní správy, autentizačné volania, zmeny v objednávke a iné.

Existuje niekoľko typov správ, ktoré môže bot odosielať. Okrem obyčajného textu to môžu byť obrázky, videá, zvukové záznamy, súbory, štruktúrované správy a niekoľko typov tlačidiel. Štruktúrované karty obsahujú kombináciu obrázkov, formátovaného textu a tlačidiel. Môžu sa zobrazovať aj vo väčšom počte naraz v takzvaných carouseloch. Existujú aj špeciálne typy kariet pre zobrazenie informácií o letoch.

Tlačidlá sú rozdelené podľa typu akcie, ktorú spúšťajú. Sú to zobrazenie webového odkazu, možnosť jednoduchej odpovede, zahájenie hovoru, zdieľanie obsahu alebo aj potvrdenie platby. Špeciálnym zobrazovacím prostriedkom je takzvaný webview. Ide o okno zobrazujúce obsah formátovaný pomocou HTLM v rámci aplikácie. Element sa zobrazí keď tlačidlo webového odkazu dostane parameter `webview_height_ratio`. Stránka, ktorá sa vyvolá z odkazu je schopná komunikovať s dátami v aplikácii pomocou vývojových nástrojov Messenger Extensions Javascript SDK.

2.2 Microsoft Bot Framework

Je komplexný súbor nástrojov na vytvorenie a nasadenie chatovacích botov, ktorí sú narozdiel od konkurenčných riešení prístupní aj z viacerých služieb [16]. Medzi tieto patria napríklad Microsoftom vlastnené aplikácie Skype a Teams, konkurenčné služby Facebook Messenger, Slack, Telegram, Kik, GroupMe či SMS službu Twillio, ale aj univerzálne kanály ako e-mail alebo webový klient.

Konverzácia môže byť vo forme voľnej prirodzenej komunikácie alebo nalinkovaná skrz tlačidlá pre výber odpovedí. Komunikácia môže prebiehať pomocou textu alebo multimédií v podobe obrázkov, videí a dedikovaných prvkov ako sú napríklad karty s voliteľným obsahom.

Hosťovanie botov je možné na akejkoľvek dostupnej webovej službe alebo serveri. Microsoft odporúča využívanie vlastnej služby Azure. Okrem toho je možné pri vývoji používať emulátor, ktorý je v najnovšej verzii open source a dostupný na portáli správy verzií, Github.

Vyššie popísaná registrácia prepája chat bota a službu Bot Connector, čo je časť frameworku umožňujúca prepojenie na podporované koncové kanály. Pri preposielaní správy z bota na konkrétnu službu príde k normalizácii správy tak, aby bola rozumne interpretovaná používateľovi na danej platforme.

Nejde však iba o jednoduchý preklad formátu správy ale aj o interpretovanie elementov, ktoré konkrétna platforma nepodporuje. Práve to umožňuje, aby bol bot kompatibilný s viacerými službami.

Programovanie bota uľahčuje súbor vývojových nástrojov Bot Buider SDK pre C# [1] a Node.js. Ten okrem abstrakcie procesov pri tvorení komunikácie popísaných nižšie umožňuje aj automatickú autentizáciu s konektorom. Ak bot nevyužíva SDK ale REST API je potrebné implementovať štvorbodové overenie vo forme:

- SSL/TSL pre všetky service-to-service spojenia;
- OAuth 2.0 prihlásenie do Microsoft účtu pre vygenerovanie tokenu, ktorý bot použije na posielanie správ;
- JSON webových tokenov (JWT) pre dekódovanie posielaných správ;
- OpenId metadáta na koncových bodoch pre overovanie podpísaných JWT tokenov.

Po úspešnej autentizácii oboch strán je možné posielať REST (Representational state transfer) správy cez protokol HTTPS. Môžu to byť správy typu GET pre získanie informácií z konverzácie, alebo o používateľoch, POST na vytvorenie konverzácie alebo posielanie správ, DELETE pre zmazanie či PUT pre aktualizáciu konverzácie.

Bot Connector REST API doplňuje Bot State REST API pre ukladanie, editáciu a vymazanie používateľských dát. Pre registráciu nového chat bota je nutné vyplniť formulár na webovom portáli Bot Frameworku. Pri registrácii sú doležité najmä tri položky:

- Bot handle – unikátny názov bota pozostávajúci alfanumerických znakov;
- Messaging endpoint – url adresa, na ktorej je služba dostupná vždy vo forme: https://adresa/api/messages;
- Microsoft app id – unikátny identifikačný kód spoločný pre všetky Microsoft aplikácie, pri jeho generovaní sa generuje aj overovací kľúč, ktorý je v dokumentácii označovaný ako Microssoft app password.

Bot handle, Microsoft app id a Microssoft app password slúžia ako overovacie hodnoty medzi botom a konektorom pri využívaní Bot Buider SDK. Na portáli je ďalej možné registrovaného bota editovať, prípadne pridávať dostupné kanály. V základe je vždy nastavený iba kanál webového chatu, ktorým je možné otestovať komunikáciu priamo z tejto stránky a kanál Skypu. Nachádza sa tu aj jednoduché overenie dostupnosti pripojenia.

Pridávanie kanálu závisí od jednotlivých platforiem. V prípade každej je dostupný návod ako sa na službu napojiť. Získaním kľúčov a iných informácií a následným zadaním do formuláru v sprievodcovi prepojenia si Bot Connector automaticky nakonfiguruje prístup k danému kanálu.

V zozname pripojených služieb sa nachádzajú hlásenia o prípadných chybách a informácie či je bot pre platformu povolený alebo publikovaný. Chat boti sú potom dostupní skrz katalóg Bot Directory. Bot Builder pre Microsoft Bot Framework existuje v dvoch verziách, ktoré sa od seba líšia najmä použitou technológiou.

Skype

Skype je služba, ktorá bola primárne navrhnutá ako peer-to-peer klient poskytujúci telefónne a video hovory za využitia proprietárneho Skype protokolu. Neskôr pribudla podpora pre instant messaging. Službu v roku 2011 kúpil Microsoft. Dnes Skype využíva Microsoft Notification Protokol 24 (MSNP24), ktorý lepšie synchronizuje komunikáciu medzi zariadeniami.

Používateľ sa registruje pomocou e-mailu alebo telefónneho čísla. Skype umožňuje vytvárať skupinové konverzácie, hovory a videokonferencie, volať na čísla v telefónnej sieti (po zakúpení kreditu), v rámci videa vzdialene zdieľať pracovnú plochu, spravovať kontakty a komunikovať s chat botmi.

Klienti Skypu sú dostupní pre všetky mobilné aj stolné platformy. Webová aplikácia je vstavanou súčasťou portálu Outlook.com. Klient pre Windows 10 ponúka aj sekciu pre vyhľadávanie botov. Skype bot umožňuje posielanie textových správ. Tiež môže odosielať a prijímať obrázky, videá (iba odosielať) a karty špeciálnych typov. Zobrazuje aj tlačidlá pre usmernenie komunikácie.

Skype podporuje niekoľko druhov kariet. Tie zobrazujú text, obrázky, zvukovú stopu, video, tlačidlá, prípadne ovládacie prvky. Je možné ich zobraziť viac naraz v zozname

označovanom ako carousel. Dostupné sú aj takzvané potvrdenky (anglicky receipt card), zobrazujúce napríklad potvrdenie objednávky.

Bot dokáže pracovať aj v rámci skupinových konverzácií. Nedokáže však fungovať ako moderátor skupiny, čiže pracovať s používateľmi. Prijíma iba správy, ktoré sú adresované priamo preň. Tie sú vždy vo formáte `@<názov bota> správa`. Výhodou oproti iným platformám je možnosť využívať hovory. Bot dokáže odpovedať, pustiť nahrávku, zaznamenávať hovor, prekladať reč na text, púšťať vyzváňacie tóny a ukončiť hovor. Ak používateľ zavolá bota, platforma notifikuje server pomocou špeciálneho webhooku nastaveného v službe Bot Connector.

Microsoft Teams

Je komunikačný nástroj pre pracovné tímy, súčasť platformy Office 365 a odpoveď Microsoftu na úspešný Slack. Služba bola spustená v Novembri 2016. Umožňuje komunikáciu medzi členmi tímu pracujúcich v prostredí Office 365, čo je služba združujúca podnikové aplikácie ponúkané spoločnosťou Microsoft. Medzi tieto programy patrí balíček Office, Skype for bussiness, či úložisko OneDrive.

Teams umožňuje zdieľať dáta z týchto aplikácií, prípadne s nimi priamo pracovať. Platforma využíva chat botov naprogramovaných pomocou Microsoft Bot Framework. Keďže prostredie vychádza zo Skypu, sú funkcionalita a možnosti botov rovnaké. Keďže je Team celkom nová služba, je možné, že v budúcnosti rozšíri podporu pre botov aj o prácu s dokumentami. Pri vytváraní prepojenia v Bot Connectore nie je potrebné dodatočné overovanie. Podobne je riešený aj Skype, ktorý je navyše predinštalovaný.

2.3 Slack

Je komunikačný software určený pre tímovú spoluprácu. Využíva funkcie na základe Internet relay chat (IRC) protokolu, ktorý bol v začiatkoch vývoja súčasťou služby. Vzhľadom na vydanie prvej verejnej verzie v roku 2013 je to celkom nová služba. Po čiastočnom úniku dát v 2015 používa Slack dvojfázové overovanie.

Služba implementuje komunikačné miestnosti, verejné alebo súkromné kanály organizované podľa témy. Taktiež sa využíva aj priama komunikácia. Základom je však tím, ku ktorému sa dá prihlásiť po zadaní správnej domény, mena a hesla. Do tímu sa však používateľ nedostane, pokým mu admin tímu nepošle pozvánku a on nepotvrdí prihlasovacie údaje.

Slack obsahuje VoIP služby a videokonferencie v priamej komunikácii alebo skupinách. Umožňuje jednoduché zdieľanie dokumentov, ich ukladanie priamo v službe a vyhľadávanie v konverzáciách pomocou jednoduchých príkazov. Je rozšíriteľný pomocou vstavaných pluginov, ktoré integrujú prepojenie na služby ako napríklad Dropbox, GitHub, Google, JIRA a mnoho ďalších.

Slack apps

Je vývojová platforma aplikačných rozšírení Slacku so zameraním na prepojenie na inú webovú službu alebo chat bota [21]. Tí sa v tomto prípade označujú ako Bot Users. Majú svoj vlastný profil a dokážu využiť všetky komunikačné prostriedky ako používatelia.

Je možné naimplementovať dva druhy takýchto botov. Custom bot slúži pre interné potreby v tíme, nie je možné ho distribuovať ostatným používateľom mimo danej skupiny.

Tento typ je primárne určený pre úlohy moderátora či správcu skupiny alebo komunikačného kanálu. App bot je napojený na Slack aplikáciu, vďaka čomu je dostupný pre všetkých.

Ako komunikačný prostriedok je využívané aplikačné rozhranie Real Time Messaging API (RTM API) založené na posielaní webových socketov. Autentizáciu a začatie konverzácie zabezpečuje metóda `rtm.start` posielaním úvodných metadát na určený server. Alternatívnym spôsobom posielania správ je Events API.

Základom komunikačných schopností bota je reagovať na ostatných používateľov posielaním textových správ, súborov alebo multimédií, hodnotiť a zobrazovať štruktúrovaný obsah. Spôsobov ako programovať Slack botov je viac. Okrem RTM API je tiež možné využiť framework Botkit napísaný v Node.js a podobne ako Facebook Messenger, aj Slack umožňuje jednoduché pridávanie komerčne zameraných botov pomocou tlačidla Slack button.

2.4 Telegram

Telegram je ďalšou službou na posielanie instantných správ s dôrazom na vytváranie skupín a šifrovanie dát. Je to pomerne mladá služba vytvorená v roku 2013. Je založená na cloudových riešeniach, ktoré umožňujú synchronizáciu naprieč platformami. Projekt je licencovaný ako open-source a vyvíja ho firma Telegram Messenger LLP. Ako základná požiadavka registrácie je overenie skrz telefónne číslo.

Základom služby sú servery spoločnosti, klientské aplikácie a protokol MTProto umožňujúci ich prepojenie. Je to komunikačný protokol vyvíjaný priamo Telegramom a je prístupný pre vývojárov tretích strán, ktorí ho môžu využívať zadarmo pre svoje aplikácie. Dôraz je kladený na kódovanie správ pomocou autorizačných kľúčov.

Telegram umožňuje komunikovať s registrovaným kontaktom pomocou klasických správ aj multimédií, vytvárať skupiny až do počtu 5000 členov či zakladať privátne alebo verejné kanály pre posielanie správ odberateľom bez limitu počtu. Ďalej je možné posielať šifrované správy v komunikácii klient – klient s možnosťou automatického zmazania po určitom čase. Služba poskytuje nástroje na vývoj a registráciu chat botov.

Pre potreby chat botov umožňuje Telegram pripojenie špeciálneho účtu, ktorý nevyžaduje overenie telefónnym číslom a slúži ako prostredie pre aplikáciu bežiacu na ich serveroch. Komunikácia prebieha pomocou Bot API, čo je zjednodušené rozhranie založené na protokole HTTP [22].

Registrácia nových účtov pre jednotlivé chat boty je riešená skrz špeciálny bot. Ten sa nazýva Botfather čím vtipne odkazuje na svetoznámu knihu a film Gotfather. Zadaním príkazu /newbot sa vytvorí nový bot, ktorému stačí zadať názov a meno kontaktu. Pomocou /mybots je možné zobraziť všetky priradené účty chat botov, prípadne ich editovať skrz príkazy ako /setname /setuserpic /deletebot a iné.

Bot API

Každý bot účet získa pri vytvorení unikátny autorizačný kľúč, ktorý slúži na overenie komunikujúceho programu. Komunikácia prebieha cez protokol HTTPS. Správy sú posielané vo formáte `https://api.telegram.org/bot<token>/METHOD_NAME`, kde token je unikátny autorizačný kľúč a METHOD_NAME názov podporovanej metódy.

Odpoveďou je vždy JSON objekt s boolean položkou „ok“ a ďalšími voliteľnými. Všetky názvy metód nie sú case sensitive a všetky odosielané správy musia mať kódovanie UTF-8.

Existuje niekoľko podporovaných názvov metód a ich parametrov popísaných v dokumentácii na webových stránkach Telegramu. Napríklad metóda sendMessage pre odosielanie správ, ktorá má povinné parametre chat_id identifikátor konverzácie a text správy.

Všetky typy používané v Bot API správach sú reprezentované JSON objektami. V dokumentácii sú uvedené napríklad typy: User, Chat, Message, Audio, Voice, Contact, KeyboardButton, ChatMemeber, InputFile a mnoho iných. Väčšina má ako jeden z parametrov identifikačné číslo, ostatné parametre sa líšia v závislosti od typu.

Chat bot pre Telegram dokáže komunikovať skrz klasický text, pomocou multimédií ako obrázky, zvuk a video, posielať dokumenty, pracovať s preddefinovanými možnosťami zobrazujúcimi tlačidlá pod textovým polom. Tiež dokáže pracovať s viacerými používateľmi, pridávať kontakty, prípadne ich blokovať alebo vyhodiť z konverzácie, takže môže plniť úlohu moderátora skupiny.

Telegram podporuje aj špeciálny typ bota, ktorý pracuje pre používateľa na úrovni vkladacieho riadku. Tento typ bota podporuje príkazy vo formáte `@<command>`, kde command je názov podporovaného príkazu. Po zadaní príkazu v prostredí aplikácie sa rozpozná daný bot, ak je v priradený v kontaktoch používateľa a zobrazia sa možnosti v súvislosti s príkazom.

2.5 Kik Messenger

Kik je kontroverzná IM komunikačná služba založená v roku 2009 a obľúbená najmä u mladších používateľov. Aplikácia podporuje klasickú konverzáciu v dvojici alebo v skupine, okrem textu aj komunikačné nástroje ako obrázky, fotografie, nálepky a emotikony. Zaujímavosťou sú webové rozšírenia určené pre vkladanie multimediálneho obsahu či podobne ako vo Facebook Messengeri, pridávanie kontaktu pomocou unikátneho grafického kódu.

Novinkou sú verejné skupiny, definované podľa fráz vo formáte `#<názov skupiny>`. Aplikácia umožňuje aj komunikáciu s chat botmi. Vyhľadávanie prebieha klasicky ako u normálneho kontaktu zadaním hľadaného mena alebo jeho časti, ale aj pomocou kompletného bot katalógu. Zoznam je rozdelený do kategórií podľa určenia.

Boti dokážu komunikovať všetkými dostupnými prostriedkami, ktoré sú spomenuté vyššie. Rozšírená je aj nalinkovaná konverzácia kde používateľ kliká na vybrané odpovede. Tú je možné kombinovať s klasickým zadávaním textu pre zrýchlenie komunikácie. Platforma podporuje aj inline botov. Vyvolávajú sa podobne ako v Telegrame, čiže vo formáte: @<názov bota> parameter.

Vytvorenie bota

Pre registrovanie nového bota je potrebné, podobne ako u Telegramu použiť konverzáciu so špeciálnym botom. Najľahší spôsob ako ju vyvolať, je načítať kruhový kód na webových stránkach dokumentácie. V aplikácii sa načíta konverzácia, kde po zadaní unikátneho mena vznikne nový bot. Konfiguračný bot dokáže ešte nastaviť ikonu, verejné meno či pridanie iných administrátorov.

Vývojové nástroje sú dostupné ako voľne dostupné knižnice v jazykoch python a node.js [13]. Podporované sú aj správy cez HTTP protokol. Autorizácia prebieha pomocou klasického 64 bitového kódovania. Informácie sú posielané ako JSON objekty cez správy typu POST a GET.

Po úvodnom vytvorení je potrebné bota autorizovať a nastaviť pomocou konfiguračnej POST správy. Autorizačný kľúč je dostupný v developerskom profile bota. API ďalej

obsahujú formátovanie správ, rozpoznávanie používateľov, vytváranie rýchlych odpovedí namiesto klávesnice a iné.

2.6 GroupMe

Instant messaging služba založená v roku 2011. Dnes ju vlastní Microsoft, ktorý ju získal pri akvizícii Skypu. Zameriava sa hlavne na skupinové konverzácie. Okrem nich však umožňuje aj tie klasické. Služba je dostupná na väčšine mobilných aj stolových systémoch. Okrem nich je možné dostávať správy aj skrz SMS. Funkcionalita je však obmedzená iba pre americký trh.

V konverzácii ide okrem posielania správ zdieľať obrázky, videá, polohu alebo dokumenty. Tiež je možné z aplikácie vytvárať nové udalosti a zdieľať ich s ostatnými členmi skupiny. Špecialitou je označovať jednotlivé správy alebo odkazy ako obľúbené.

GroupMe používa jednoduché REST API [11], pomocou ktorého je možné implementovať aj chat bota. Pomocou JSON správ je možné editovať skupiny, pridávať nové alebo ich mazať. Tiež je možné pridať sa do existujúcej, editovať členov skupín, prípadne ich blokovať.

Ďalej je možné pomocou API posielať všetky podporované typy správ. Formát správy poslanej v skupinovej konverzácii sa líši od priamej v tom, že sa navyše uvádza aj identifikátor používateľa. Podporované je aj pridávanie správ do obľúbených.

Pre vytvorenie nového bota je nutné sa najprv zaregistrovať na portále pre vývojárov. Botov je možné registrovať iba v skupinách v ktorých je vývojár členom. Tým pádom nie sú verejne prístupní ako na ostatných platformách. Pre úspešnú registráciu je nutné okrem mena zadať aj callback URL, čiže prístupovú adresu na vzdialený server. Pri registrácii sa následne vygeneruje URL obsahujúce identifikátor bota pre posielanie správ opačným smerom.

2.7 Twillio

Ide o súbor nástrojov, ktorý uľahčuje programovanie komunikačných aplikácií pomocou abstrahovania architektúry komunikácie [24]. Tým pádom sa vývojár nemusí zaoberať protokolmi potrebnými pre daný typ komunikácie. Pomocou týchto nástrojov je možné riešiť hovory, správy ale aj autentizáciu.

Nástroje pre hlasové alebo video hovory podporujú rozhranie pre komunikáciu skrz telekomunikačné kanály, prípadne hovor z aplikácie cez internet, ktorý služby Twillio následne prepoja na telefónnu linku. Podporovaná je aj peer-to-peer komunikácia, hromadné hovory a VoIP s plnou podporou SIP protokolu.

Podpora textovej komunikácie spočíva v implementácii SMS komunikácie dostupnej celosvetovo a MMS komunikácie obmedzenej na severoamerické štáty. Ide o komunikáciu z aplikácie cez internet na vonkajšiu telefónu linku. Okrem toho sú podporované aj nástroje pre vývoj vstavaných chatov v aplikáciách a napríklad aj push notifikácie.

Nástroje pre abstrakciu autentizačnej logiky implementujú dvojfázové overovanie, ktoré zabezpečuje služba AUTHY. Ide o vysoko spoľahlivé zabezpečenie jednoduché na implementáciu. Všetky nástroje sú dostupné v širokej palete programovacích jazykov ako napríklad PHP, Python, Javascript, C# a iné. Služba Twillio figuruje ako kanál pre posielanie SMS správ v Microsoft Bot Connectore.

2.8 Line

Pôvodom japonská služba, ktorá vznikla v roku 2011. Na domácom trhu sa krátko po vypustení stala najpoužívanejšou IM službou. Je obľúbená aj v ostatných východoázijských krajinách. Dostupná je pre väčšinu mobilných a stolných systémov.

Aplikácia podporuje klasickú textovú komunikáciu ako aj hlasové a video hovory, vytváranie skupinových konverzácií či posielanie multimédií a súborov. Line podporuje aj posielanie príspevkov, ktoré sa zobrazujú v príslušnej sekcii aplikácie. Zobrazujú sa iba príspevky od kontaktov ktoré si používateľ pridal ako priateľov.

Platforma podporuje aj mnoho rozšírení v podobe prídavných modulov alebo aplikácií. Ide o funkčné rozšírenia umožňujúce platby, zdieľanie polohy či úložisko súborov, ale aj o hry využívajúce prepojenie s účtom.

Ďalšou pridanou hodnotou sú takzvané „Official accounts“ (oficiálne účty), ktoré slúžia ako komunikačné kanály spoločností alebo implementujú pomocníkov vo forme chat botov. Vyhľadávanie oficiálnych účtov je špecifikované podľa krajiny, tieto účty môžu využívať klasickú konverzáciu alebo tiež pridávať príspevky.

Chat bot môže na platforme fungovať iba pod oficiálnym účtom, ktorý je možné voľne vytvoriť a využívať, pričom je potrebné nastaviť povolenia vývojových nástrojov.

Messaging API

Po registrácii účtu, skrz ktorý bude komunikovať bot, je potrebné naprogramovať logiku prijímania a odosielania správ. K tomuto účelu slúži vývojové rozhranie Messaging API [14]. Pomocou týchto vývojových nástrojov je možná nasledújúca funkcionalita:

- vytváranie konštantných spojení, takzvaných webhookov, a následné prijímanie správ;
- posielanie správ používateľom okamžite alebo v stanovenom čase ako notifikácia;
- posielanie správ niekoľkým používateľom (v skupine);
- sťahovanie obsahu ako obrázky, zvuk a iné dáta;
- získavanie profilových údajov o používateľoch;
- či opustenie konverzácie alebo skupiny.

Architektonicky je komunikácia realizovaná pomocou klasických HTTPS správ, kde overenie serveru zastrešuje úvodná POST správa, pričom sa vytvorí spojenie. Odosielanie správ z klienta na server prebieha pomocou JSON objektov. Vývoj je uľahčený aj pomocou nástrojov abstrahujúcich spojenie a jednotlivé metódy. Sú dostupné ako LINE Bot SDK v jazykoch Java, PHP, Go, Perl, Ruby a Python. Line ako jedna z dvoch platforiem v tomto porovnaní neumožňuje prepojenie s konektorom Microsoft Bot Frameworku.

2.9 WeChat

Komunikačná služba populárna najmä v Číne, prevádzkovaná spoločnosťou Tencent. Z aktuálne odhadovaného počtu 800 miliónov aktívnych používateľov približne iba 10 percent nepochádza z domáceho trhu. Spustená bola v roku 2011. Dnes podporuje klientov na všetkých stolných aj mobilných platformách.

Služba funguje najmä ako IM komunikačný prostriedok, ale obsahuje aj súčasti ako zdieľanie príspevkov pre odberateľov, vyhľadávanie nových kontaktov vo svojom okolí či podporu platobných služieb priamo v aplikácii. V konverzácii je možné využiť všetky zaužívané prostriedky ako sú obrázky, videá, súbory, zvukové nahrávk,y ale napríklad aj zdieľanie svojej aktuálnej lokalizácie. Podporované sú tiež skupinové konverzácie.

Vývoj botov

WeChat ponúka rozsiahle možnosti interakcie s ich IM službou [26]. V tejto práci bude spomenutá iba na časť venovaná chat botom. Ich identita je viazaná na takzvané oficiálne účty. Tie fungujú aj ako kanály pre šírenie príspevkov širšiemu obecenstvu a pomocou natívnych vývojových nástrojov ich je možné automatizovať. Bot však môže fungovať aj v režime klasickej komunikácie.

Verifikácia prebieha pomocou kombinácie podpisov, časovej značky posielanej v úvodnej správe. Služba pracuje s protokolom HTTPS a správami vo formáte JSON. Prístup k službe skrz API umožňuje unikátny kľúč.

Vývojové prostredie ponúka možnosti pre prijímanie a odosielanie správ. Tiež je možné zabezpečiť komunikáciu pomocou kódovania komunikácie. Odosielanie multimediálneho obsahu je riešené skrz manažment príloh. Posielať je možné obrázky, hlasové záznamy a videá obmedzené na maximálnu veľkosť.

Z ďalšej funkcionality je nutné zmieniť podporu pre zoznam možností. Využíva sa hlavne na poskytnutie rýchlych odpovedí. Oficiálne účty je možné propagovať pomocou QR kódov. Vývojové prostredie ponúka možnosť ich generovania. Okrem toho môže byť pre vývoj bota dôležitý nástroj pod názvom WeChat Pay umožňujúci platby v prostredí aplikácie.

Kapitola 3

Pokročilé komunikačné postriedky

V predchádzajúcej kapitole boli predstavené platformy podporujúce vytváranie chat botov. Každá platforma pritom implementovala špecifické rozšírenia komunikácie ako posielanie obrázkov, audia alebo aj špeciálnych kariet s rozličným obsahom.

Všetky tieto pomôcky v komunikácii môžeme označiť ako pokročilé prostriedky komunikácie. Avšak okrem prostriedkov, ktoré obohacujú komunikáciu, existujú aj rozšírenia funkčnosti. Tieto funkčné rozšírenia môžu byť implementované v rámci platformy alebo poskytované treťou stranou.

Príkladom však môže byť aj jednoduché rozšírenie o posielanie notifikácií. Keďže každá z platforiem umožnuje posielanie správ. Stačí vymyslieť automatizovaného odosielania a používateľ môže dostávať upozornenia v stanovenom čase. Napojenie na notifikačný systém zariadenia pritom zastrešuje aplikácia klienta danej služby.

Ako ale bolo spomenuté väčšina rozšírení potrebuje napojenie na konkrétne služby mimo platformu a to buď oficiálnou cestou skrz podporované nástroje alebo využitím toho, že väčšina botov je online služba podporujúca iné aj iné komunikačné kanály. V tejto kapitole bude zhrnutých niekoľko najdôležitejších služieb a popis ich vývojových rozhraní.

3.1 Hlasový asistenti

Inteligentný osobný asistent (anglicky Inteligent personal assistent, skrátene IPA) je program schopný plniť úlohy zadávané používateľom v rámci agendy. Tým je myslené posielanie e-mailov, vytváranie udalostí a úloh väčšinou na základe hlasových povelov, následná správa informácií a posielanie upozornení alebo aj ovládanie podporovaných zariadení z kategórie IoT či v neposlednom rade vyhľadávanie informácií na internete.

Narozdiel od chat bota, asistent je komplexnejšia aplikácia v oblasti funkčnosti aj zavádzania princípov umelej inteligencie. Väčšinou je viazaný na určitý ekosystém alebo zariadenie, nie je multiplatformálny a podporuje istú formu vývojových nástrojov.

Z hľadiska architektúry je služba v zariadeniach prezentovaná ako systémová aplikácia, veľká časť réžie a dát je však posielaná na príslušné servery. Medzi dnes najznámejších asistentov patria napríklad Microsoft Cortana, Apple Siri, Google Assistant, Amazon Alexa, Samsung Bixby, Black Berry Assistant a iné.

3.1.1 Cortana

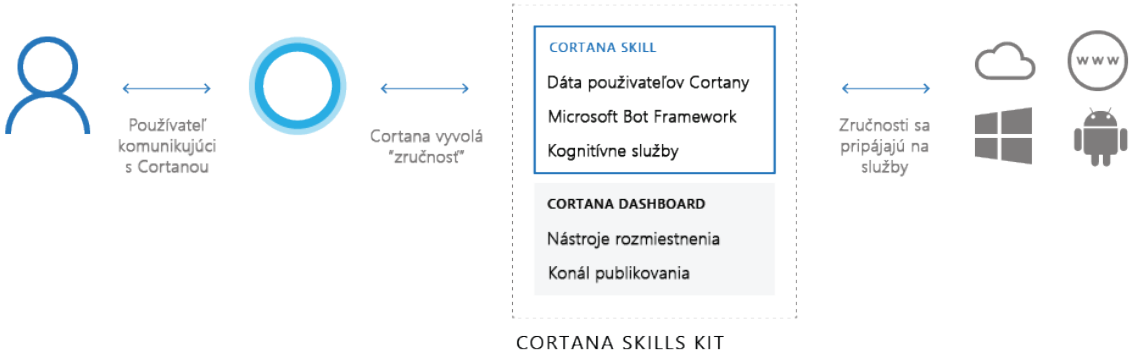
Inteligentný asistent vyvíjaný firmou Microsoft nielen pre zariadenia so systémom Windows. Aplikácia Cortany je dostupná v operačných systémoch Windows 10 pre počítače, mobilné

telefóny ale aj pre konzolu Xbox One, v systéme Android a s obmedzenou funkcionalitou aj v zariadeniach s iOS.

Cortana dokáže posielať upozornenia, rozpoznáva príkazy zadávané hlasom alebo pomocou klávesnice a pracuje s osobnými dátami používateľa. Vyhľadávanie informácií je realizované skrz Bing. Dokáže komunikovať v angličtine, nemčine, francúzštine, španielčine, portugalčine, taliančine, čínštine a japončine. Ako zaujímavosť možno spomenúť, že meno Cortana bolo vybrané na základe ženskej AI postavy z hernej série Halo.

Pre zobrazovanie osobného obsahu Cortana obsahuje nastavenia (takzvaný notes), kde používateľ môže vyplniť osobné informácie a okruhy záujmu. Používateľ tu môže vyplniť informácie ako adresu domácnosti a práce, body záujmu, hoby, obľúbené športy a prípadne aj jednotlivé tímy, filmy, hudbu a iné. Okrem toho je možné povoliť sledovanie aktuálnej pozície či telesnú aktivitu skrz aplikáciu Health. Na základe zozbieraných informácií dokáže Cortana podávať informácie o počasí, premávke na ceste do práce, o konaní dôležitých zápasov obľúbeného športového tímu alebo filmových premiér v najbližšom kine, zobrazovať reštaurácie v okolí a mnoho iného.

V neposlednom rade dokáže Cortana komunikovať s aplikáciami a pomocou príkazov od používateľa v nich vykonávať akcie. Napríklad prehrať určitú skladbu v prehrávači hudby. Takáto funkcionalita musí byť podporovaná hlavne na strane programu. V rámci počítačov s Widnows 10 je jej súčasťou aj prehľadávanie súborového systému a zoznamu nainštalovaných programov, ktoré je tiež možné spustiť na hlasový príkaz. Službu dnes využíva približne 145 miliónov používateľov. Schéma je popísaná aj pomocou obrázka 3.1



Obr. 3.1: Graf popisujúci závislosti systému zručností v prostredí Cortana Skills Kit. Obrázok je prevzatý z oficiálnej dokumentácie [17].

Vývojové prostriedky

Vývojové nástroje, ktoré umožňujú vyvíjať pre Cortanu prepojenia do aplikácií sa nazývajú Cortana Skills Kit [17]. Pomocou nich sú vytvárané takzvané zručnosti (z anglického slova skills). Pomocou nich je možné rozšíriť funkčnosti Cortany a integrovať do nej webovú službu, bota, univerzálne Windows aplikácie (Windows 10), Android aplikácie alebo internetovú stránku. Zručnosť tiež môže modifikovať používateľské rozhranie samotnej asistentky.

Nasadenie jednotlivých zručností nepotrebuje inštaláciu, Cortana ich podporuje natívne a dôležitým aspektom je iba správna fráza, ktorá vyvolá danú zručnosť. Dnes existujú tri základné typy fráz: otázku (ask), oznámenie (tell), príkaz (use), pričom je potrebné

vždy kľúčové slovo, napríklad názov danej služby, a niekoľko určujúcich, ktoré fráze pridajú konkrétne parametre.

Najjednoduchší spôsob ako zaregistrovať novú zručnosť je vytvoriť ju pomocou vývojového portálu. Ako základ každej novej zručnosti je vždy zadávaný názov, meno vlastníka alebo spoločnosti, ikona, text zobrazujúci sa pri použití zručnosti, adresa internetovej stránky, popis a trh, pre ktorý je použiteľná. Ďalej sa nastavia pravidlá podľa ktorých sa vyberú kontextovo dôležité slová a dátové štruktúry. Nakoniec sa nastaví cieľ v ktorom je implementovaná logika danej zručnosti. Tento cieľ môže byť buď webová služba (aj chat bot) alebo aplikácia pre Windows, či Android. Služba je prepojená pomocou REST API pomocou JSON správ cez HTTP protokol.

Ďalším spôsobom ako vytvoriť prepojenie medzi aplikáciou a asistentkou je pomocou vývojových nástrojov pre hlasové povely. Podporované sú aj štruktúrované dáta vo forme predpísanou slovníkom Schema.org[1]. Tieto dáta môžu byť prezentované vo formátoch Microdata a JSON-LD.

3.1.2 Siri

Inteligentný asistent od firmy Apple. Predstavený bol v roku 2011 pre iPhone 4S v rámci mobilné ho operačného systému iOS 5 a neskôr aj pre tablety iPad tretej generácie. V roku 2016 bola pridaná podpora pre počítače s operačným systémom macOS Sierra. Okrem toho Siri podporuje aj prouktovú líniu Apple Watch a Apple TV. Služba je schopná komunikovať v 21 jazykoch s lokalizáciou v 36 krajinách (stav v roku 2017).

Siri podporuje najmä hlasové príkazy. Tie sa môžu vzťahovať na súkromnú agendu, čiže príkazy pre poslanie emailu, SMS správy alebo vytvorenie pripomienky či udalosti v kalendári. Ďalej to môžu byť všeobecné otázky, ktoré služba čerpá z vlastnej databáze ale najmä zo služieb ako Wikipedia, Shazam alebo Rotten Tomatoes. Asistent tiež dokáže komunikovať so zariadeniami IoT, ktoré podporujú technológiu HomeKit.

Počínajúc iOS vo verzii 10 je možné vytvárať prepojenie vlastných aplikácií pomocou platformy SiriKit [5]. Apple vytvoril sadu nástrojov, ktoré umožňujú prístup k Siri tak aby používateľ mohol zadávať príkazy typu: „Pošli správu Petrovi Novotnému cez Moju textovú aplikáciu.“ Možnosti využitia sú však limitované na typy príkazov zahŕňajúce VoiP hovory, textové správy, platby, prístup k fotkám, aktivity pri cvičení, rezerváciu leteniek alebo miesta v reštaurácii, či ovládanie automobilu skrz CarPlay platformu. Z implementačného hľadiska sa SiriKit skladá z nasledujúcich dvoch frameworkov.

- Intents extension – prijíma objekty zo systému obsahujúce používateľský príkaz a spracováva ich podľa priradených funkcií.
- Intents UI extension – umožňuje prispôsobenie používateľského rozhrania po úspešnom spracovaní príkazu. Napríklad pre zadanie dodatočných informácií alebo zobrazení štruktúrovanej odpovede.

Pri vytváraní rozšírení je najprv nutná konfigurácia projektu. Nastavenie povolení je nutné pre autorizáciu používania nástrojov SiriKit a je nutné ju inicializovať chode aplikácie pomocou metódy requestSiriAuthorization. Následne je potrebné vyriešiť obsluhu prichádzajúcich správ, ktoré sú reprezentované takzvaným intent objektom. Po získaním parametrov tohto objektu sa vykoná metóda confirmStartWorkout, pomocou ktorej aplikácia

[1] Schema.org je iniciatíva za vytvorenie jednotnej schémy pre štruktúrované dáta na webových stránkach. Bola založená v roku 2011 spoločnosťami Google, Microsoft a Yahoo!. Viac informácií na oficiálnych stránkach https://schema.org/.

potvrdzuje úspešné prezvanie objektu a jeho parametrov a oznamuje začatie vykonávania úkonov v nadväznosti na príkaze, a metóda handleStartWorkout, ktorá riadi vykonávanie priradených funkcií. Obe metódy sú asynchrónne.

Ak vývojárom nestačí základná slovná zásoba asistenta môžu si definovať vlastné frázy alebo slová pomocou objektu INVocabulary ak ide o výraz pre konkrétneho používateľa alebo pridaním súbory globálnej slovnej zásoby pre zaregistrovanie výrazu v rámci aplikácie. Môže ísť o názvy spoločností, mená kontaktov, prípadne názvy zložiek alebo albumov a iné.

Po spracovaní požiadavku zobrazí Siri obsah odpovede pomocou štandardného rozhrania. Okrem toho je možné vytvárať aj vlastné a to za pomoci už spomínaného Intents UI rozšírenia. Samozrejme je opäť nutné najprv nakonfigurovať projekt. Pri implementácii grafických rozšírení sa následne vytvárajú takzvané pohľady (anglicky view), ktoré musia mať naprogramované rozloženie prvkov pomocou klasických iOS zobrazovacích metód. Napríklad viewWillAppear, viewDidAppear, viewWillDisapear atď.

3.1.3 Google Assistant

Google Assistant je služba inteligentného asistenta nadväzujúca na predošlú, ešte stále existujúcu službu Google Now, ktorá bola svojho času odpoveďou na Apple Siri avšak ponúkala iné koncepčné riešenie. To nebolo primárne založené iba na hlasovom ovládaní a zadávaní príkazov ale aj na zobrazovaní obsahu v kartách. Tie zobrazovali informácie o aktuálnom počasí, udalostiach, doprave a osobných záujmoch. Karty bolo možné nastaviť podľa osobných preferencií.

V roku 2016 predstavil Google novú, už spomenutú, asistenčnú službu Google Assistant, ktorá sa od Google Now líši hlavne v schopnosti viesť s používateľom komunikáciu pričom si pamätá kontext. Pôvodne bol asistent integrovaný iba do aplikácie Allo a hlasom ovládaného reproduktoru Google Home. Neskôr nahradil službu Now exkluzívne v prvej generácii Pixel smartfónov a od februára 2017 bol uvoľnený aj pre ostatné zariadenia s operačným systémom Android ako telefóny, tablety, nositeľná elektronika, inteligentné televízie a palubné systémy v automobiloch s Android Auto technológoiu.

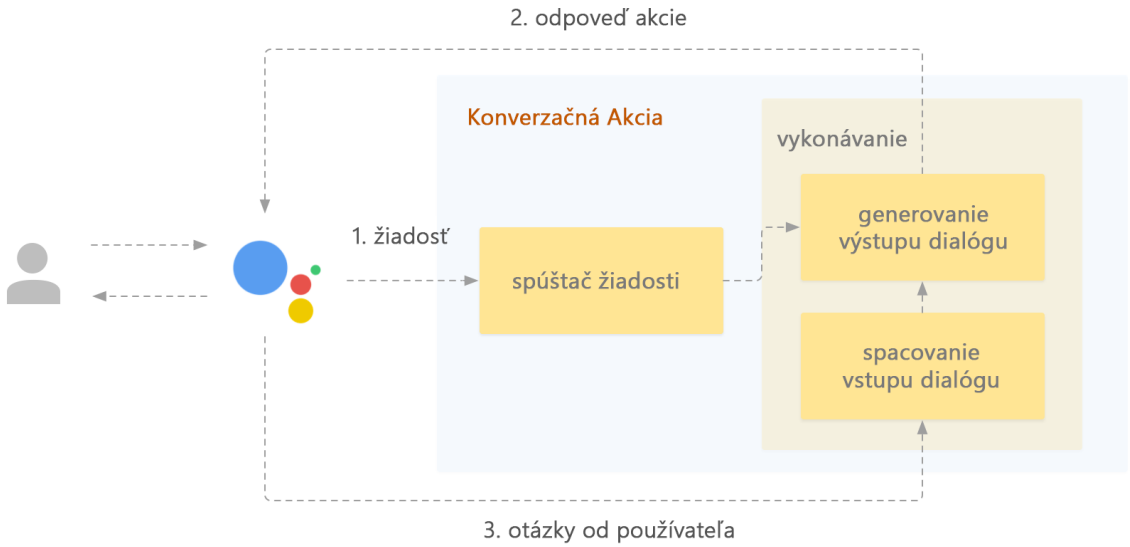
Okrem zariadení je možné asistenta integrovať aj do aplikácií tretích strán. Špeciálne využitie poskytuje asistent v IM službe Google Allo, ktorá slúži ako prostriedok pre komunikáciu s asistentom pomocou textu. Formát konverzácie je podobný ako ten pri písaní si botom na iných platformách. Využívajú sa všetky pokročilé komunikačné prostriedky ako multimédia, karty, prípadne vedenie konverzácie pomocou výberu z možností. Rozdiel je však v inteligencii a všestrannosti asistenta na rozdiel od chat botov. Asistenta je tiež možné prizvať do konverzácie s inými používateľmi a to pomocou príkazu @google.

Programovanie prístupov k službe Google Assistant je umožnené pomocou nástroju Actions on Google [9], ktorá implementuje dva základné koncepty akcií.

- Konverzačné akcie – umožňujú vytvorenie kompletnej konverzácie, jednotlivých akcií a spôsoby reagovania.
- Priame akcie – prednastavené akcie kde vývojár nerieši interakciu s používateľom a implementuje iba vykonanie akcie.

Zatiaľ čo priame akcie existujú hlavne kvôli zjednodušeniu a sú obmedzené na niekoľko základných situácií, ako napríklad objednanie jedla, automatizáciu zariadení v domácnosti alebo ovládanie multimédií, pomocou konverzačných akcií je možné vytvoriť kompletne nový use case. Pri vytváraní akcií je nutné definovať tri základné komponenty popísané na obrázku 3.2.

- Spúšťačom žiadosti (invocation) sa definuje ako môžu používatelia nájsť a spustiť akciu. Po spustení sa začína konverzácia.
- Ako prvá sa vygeneruje sa úvodná odpoveď, ktorá zahájí dialóg medzi akciou a používateľom. Dialógy definujú spôsob komunikácie. Dialóg je viazaný na vykonávanie procesov na pozadí.
- Vykonávanie procesu je kód, ktorý spracováva otázky, na základe ktorých vykoná akcie a generuje náležité odpovede. Je to koncový bod v REST komunikácii.



Obr. 3.2: Graf popisujúci fungovanie konverzačnej akcie. Čísla indikujú postupnosť krokov. Obrázok je prevzatý z oficiálnej dokumentácie [9].

Všetky spomenuté komponenty sú povinnou súčasťou komunikácie s Google Assistantom pomocou aplikačného rozhrania Conversation API. V ňom je definovaný formát žiadostí a odpovedí. Žiadosť obsahuje zadaný vstup, informácie o používateľovi a zariadení. Sú posielané pri vyvolaní akcie alebo pri jej vykonávaní ako dodatočné informácie na spracovanie. Odpovede sú posielané naspäť na asistenta za sprostredkovať informácie. Pri odoslaní odpovede je možné nastaviť či sa má očakávať aj odpoveď. Komunikácia je sprostredkovaná pomocou HTTP žiadostí a jednotlivé správy sa posielajú ako JSON objekty.

Vývoj jednotlivých akcií je možné realizovať pomocou vývojových nástrojov na webovej platforme Node.js. Konfigurácia jednotlivých skupín akcií sú definované v špeciálnych JSON súboroch. Medzi asistenta a logiku akcie je tiež možné dosadiť vstavaný systém pre rozpoznávanie reči, api.ai, o ktorom je viac napísané v kapitole 3.4.2. Na testovanie v prostredí príkazového riadku sa využíva nástroj gactions CLI a simulácia pre zariadenie Google Home je riešená skrz nástroj Web Simulator.

3.1.4 Amazon Alexa

Amazon Alexa je inteligentný osobný asistent vytvorený v roku 2014 spoločnosťou Amazon. Podobne ako iné aj tento asistent umožňuje základné úkony ako napríklad vyhľadávať informácie, prehrávať hudbu, vytvárať upozornenia a zoznam úloh, oznámiť predpoveď počasia

alebo manažovať zariadenia IoT v domácnosti. Okrem aplikácií pre Android a iOS je Alexa dostupná aj v hlasom ovládanom reproduktore Amazon Echo.

Alexa podporuje aj aplikácie tretích strán a to pomocou nástroju Alexa Skills Kit [2], ktorý umožňuje nové zručnosti. Každú takúto zručnosť je možné spustiť na základe vety, ktorá sa vždy skladá z pravidiel, kde na začiatku musí byť oslovenie asistenta, ďalej nasleduje identifikácia danej zručnosti podľa názvu a podporovaná fráza s prípadnou konkrétnou žiadosťou. Ak používateľ zadá najprv otvárací príkaz, ďalšie už môže zadávať bez konkretizovania danej zručnosti.

Pri vývoji je potrebné vytvoriť profil zručnosti s unikátnym menom. Amazon preferuje umiestnenie logickej časti programu v prostredí ich cloudového riešenia Amazov Web Services (AWS). Vývojové nástroje existujú hlavne pre platformu Node.js. Z architektonické hľadiska však ide opäť od HTTP komunikáciu. Viac špecifický nástroj je Smart Home Skill Kit, ktorý sa zameriava priamo na vytváranie akcií pre IoT.

3.2 Platobné brány

Pri tvorbe chat botov pre internetové obchody alebo služby, ktoré istým spôsobom ponúkajú spoplatnený obsah či tovar sa naskytáva možnosť zaniesť priamo do prostredia chatu rozhranie platobných brán, jednotlivých služieb a systémov. Medzi takéto služby nepatria iba online prístupy k účtom jednotlivých bánk ale aj poskytovatelia kreditných kariet či systémy nebankových spoločností umožňujúce internetové platby. Celosvetovo najrozšírenejšou takouto firmou je PayPal.

3.2.1 PayPal

Firma založená v roku 1998 prevádzkuje rovnomennú internetovú službu poskytujúcu online platby a prevody peňazí. Dnes je spoločnosť vo vlastníctve eBayu. Služba umožňuje odosielanie a prijímanie financií v online prostredí bez vystavenia svojej kreditnej karty a informácií o bankovom účte potenciálnym hrozbám. Využíva pritom existujúcu infraštruktúru platobných kariet a bankových účtov, ktoré si používateľ prepojí so svojím účtom. Okrem ochrany je PayPal aj vhodnou platformou pre prepojenie viacerých bankových účtov a platobných kariet na jednom mieste. Ako zaujímavosť je možné uviesť, že jedným zo zakladajúcich členov bol aj Elon Musk.

Dôležitou súčasťou ekosystému je predovšetkým napojenie na jednotlivé služby, ktoré istým spôsobom pracujú s financiami. Za týmto účelom existuje veľké variabilita vývojových možností [20]. Podporované je ako základné aplikačné rozhranie tak aj vývojové nástroje pre niekoľko platforiem a programovacích jazykov. Základné vývojové prostriedky nepodporujú všetky svetové meny.

Aplikačné rozhranie využíva metódy HTTP komunikácie cez JSON správy a štruktúru koncových bodov pomocou technológie REST. Autorizačným frameworkom je OAuth 2.0. Volanie v sebe kombinuje HTTP metódu, URI zdroja a ak sú požadované tak aj HTTP hlavičky a JSON formát, prípadne stránkovanie ak je návratovou hodnotou zoznam objektov. Pri každom volaní je potrebné posielať prístupovú reťazec. Tá má obmedzenú dobu platnosti a získava sa pomocou špeciálneho volania, pri ktorom je posielané identifikačné číslo účtu a súkromného kľúča.

Pomocou volaní je možné zobrazovať dáta a na ich základe vykonávať operácie. Oblasti ku ktorým je možné pristupovať sú fakturačné zmluvy a plány, používateľské informácie, faktúry, platby, výplaty, pokladničné systémy a ďalšie. Napríklad pri zaplatení niečoho

v aplikácii využívajúcej aplikačné rozhranie PayPalu je nutné vykonať hneď niekoľko krokov. Okrem autorizácie je nutné vytvoriť novú unikátnu PayPal platbu, získať pre ňu povolenie od používateľa a nakoniec pomocou iného príkazu vykonať. Rozhranie pre platby taktiež podporuje aj jednoduché platby pomocou kreditnej karty kde stačí iba získať prístupový kľúč a vytvoriť platbu so zadanou kartou.

Pre ľahšiu implementáciu a abstrakciu niektorých volaní či overovania sú dostupné vývojové nástroje. Tie možno rozčleniť na nástroje pre mobilné zariadenia s operačným systémom Android a iOS, platformu integrácie pokladničných systémov, PayPal Here, a potom nástroje postavené nad základným aplikačným rozhraním využívajúcim REST technológiu a nad rozhraním využívajúcim NVP a SOAP. Nástroje postavené nad REST aplikačným rozhraním sú dostupné v programovacích jazykoch Java, PHP, Python, Ruby a platformách .NET a Node.js.

Aplikačné rozhrania pre NVP (Name Value Pair) a SOAP (Simple Object Access Protocol), využívajú model klient-server na odosielanie reťazcov alebo dátových štruktúr v závislosti na technológii. Podobne ako REST majú aj NVP a SOAP vlastné koncové body. Pre tieto rozhrania je dostupných niekoľko vývojových nástrojov cez ktoré je možné pristupovať k adaptívnym účtov, rýchlemu vybaveniu platby, faktúram, hromadným platbám, povoleniam a iným. Rozhrania NVP a SOAP nie sú na rozdiel od REST rozhraní obmedzené na zopár peňažných mien.

3.2.2 Mastercard Masterpass

Mastercard je nadnárodná spoločnosť, ktorá sa zaoberá predovšetkým spracovávaním platieb vykonaných ich platobnými kartami medzi obchodníkmi a bankami. Okrem toho však podniká kroky aj v online platbách, či už vo forme poskytovania vývojových nástrojov pre online platby alebo pomocou platformy Masterpass.

Masterpass je platobná služba, ktorá uľahčuje proces nákupu tým, že zjednodušuje zadávanie osobných informácií. Zákazník používa jeden účet, v ktorom má bezpečne uložené platobné karty a vždy k nim pristupuje až pri platbe a nemusí tak vždy zadávať svoje kontaktné a platobné informácie. Službu je však možné využívať iba vtedy ak ju daný obchod podporuje. Existuje aj mobilná aplikácia, ktorá umožňuje platbu namiesto karty vo vybraných kamenných obchodoch.

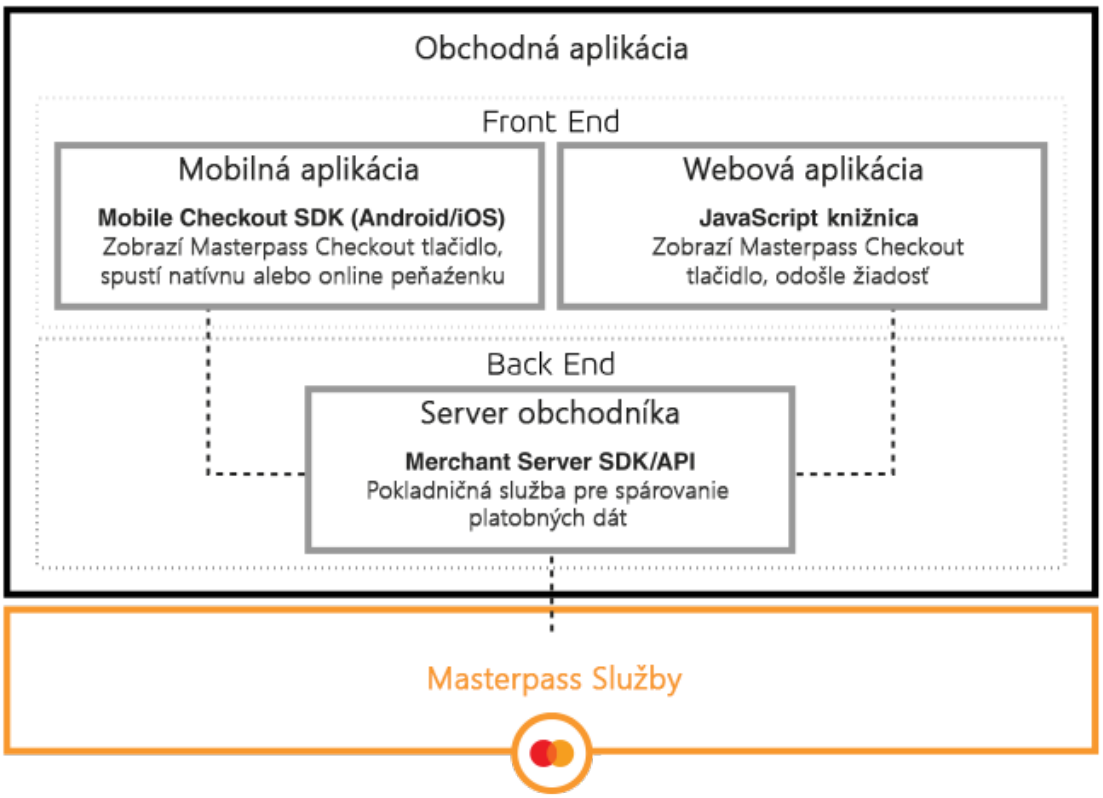
Implementáciu služby Masterpass v obchodných aplikáciách sprostredkováva hneď niekoľko vývojových nástrojov [6]. Rozdelené sú na front end a back end nástroje (Obrázok 3.3). Front end nástroje májú na starosti spôsob zobrazenia interaktívnych prvkov v aplikácii alebo na webe. Sú to nástroje Mobile Checkout SDK pre mobilné systémy Android a iOS a JavaScript knižnica pre webovú aplikáciu. Back end pozostáva z Merchant Server SDK alebo API, ktoré má na starosti pripojenie platobných dát a komunikáciu so službami Masterpass.

Vývojový balíček nástrojov Merchant Server je dostupný v niekoľkých verziách, konkrétne pre jazyky Java, C#, PHP a Ruby. Pre ostatné jazyky je dostupné aplikačné rozhranie. Pri inicializácii komunikácie je použitý autorizačný kľúč OAuth špecifikácie[2]. Masterpass podporuje tri základné typy platieb.

- Štandardná platba (standard checkout), kde sa používateľ prihlási k svojmu účtu a zadá zvolené informácie.

[2]Vo verzii 1.0a. (https://oauth.net)

- Zrýchlená platba (expess checkout), kde aplikácia vyplní predvolené informácie namiesto používateľa. Tento typ je zatiaľ obmedzený iba na aplikácie, ktoré prešli špeciálnym schvaľovacím procesom.
- Mobilná platba (mobile checkout), špeciálne určená pre zariadenia Andorid, kde je pomocou vývojových nástrojov sprostredkované vlastné rozhranie, ktoré od používateľa získa dáta a vykoná platbu.



Obr. 3.3: Grafické znázornenie usporiadania jednotlivých vývojových nástrojov v rámci štruktúry obchodnej aplikácie. Obrázok je prevzatý z oficiálnej dokumentácie [8].

3.3 Kalendárne služby

Online kalendár je internetová služba poskytujúca používateľom vytváranie pripomienok či udalostí pre konkrétny dátum a čas. Udalosti môžu byť v dlhšom časovom rozpätí alebo sa periodicky opakovať. Okrem toho umožňuje kalendár vytvárať zdielané udalosti či editovať už vytvovorené udalosti a prípadne ich mazať. V súčastnosti populárne kalendárne služby sú Google Calendar, Microsoft Outlook Calendar, Apple Calendar, Mozilla Sunbird, Yahoo! Calendar a iné. Pokročilejšia funkcionalita jednotlivých služieb sa líši, prípadne sa prenáša na klientov.

Klienti nielen týchto služieb existujú ako webové aplikácie dostupné online alebo ako programy v operačnom systéme, ktoré dáta ukladajú lokálne a priebežne synchronizujú so serverom. Môžu to byť aplikácie pre stolné aj mobilné systémy. Niektoré služby môžu mať zároveň viac typov klientov. Aplikácie klintov môžu synchronizovať jeden a viac zdrojových systémov, môžu byť súčasťou komplexnejšej aplikácie, ktorej súčasťou často býva aj e-mailový klient či adresár kontaktov.

Pre napojenie na viac klientov (aj univerzálnych vyvíjaných treťou stranou) musí mať kalendárna služba určitý typ aplikačného rozhrania, ktoré poskytuje prístupové funkcie k vytváraniu, editácii a iným úkonom nad daným kalendárom. Niektoré služby ako napríklad Yahoo! kalendár ale aj mnohé iné výrobcami vstavané systémy dnes najmä v mobilných systémoch, nemajú žiadne možnosti vývoja.

3.3.1 Google Calendar

Je platforma online kalendárnej služby vyvýjaná firmou Aphabet (Google), spustená v roku 2006. K službe sa je možné pripojiť pomocou oficiálneho webového klienta, či aplikácií pre platformy Android a iOS.

Google kalendár podporuje okrem základného vytvárania a editácie, pridávať čas začiatku a konca udalosti dokonca aj v špecifickom časovom pásme, prípadne nastaviť konanie na celý deň. Ďalej je možné opakovať udalosť v stanovených intervaloch, nastavovať farbu pre rozlíšenie v rámci skupiny, posielať notifikácie a skrz e-mail, možnosť pridať polohu udalosti a v neposlednom rade posielať iným používateľom žiadosť o účasti. Udalosť je možné vygenerovať aj automaticky na základe prijatia e-mailu v službe Gmail s podporovaným formátom.

Pre potreby vývojárov aplikácií tretích strán existuje vývojové aplikačné rozhranie, Calendar API [8], ktoré integruje služby kalendára do webovej či mobilnej aplikácie. Na základe toho je umožnené prehliadať verejne dostupné udalosti alebo po autentizácii aj tie súkromné. Aplikáciám je umožnená synchronizácia s konkrétnym kalendárom. Najväčšou výhodou je však automatizované pridávanie nových udalostí na základe akcií v rozhraní aplikácie. Aplikačné rozhranie je dostupné pre platformy .NET, Android, iOS, Google Apps Script, Node.js, prípadne pre jazyky Go, Java, Javascript, PHP, Python či Ruby. Popri klasickom rozhraní je dostupné aj REST API pre prístup skrz webové volania.

Základnými prostriedkami rozhrania, ku ktorým je možné pristupovať, sú: udalosti, kalendáre, zoznamy kalendárov, celkové nastavenia systému, pravidlá kontroly prístupu do kalendára, farby, reprezentujúce kategorizáciu udalostí alebo kalendárov, a indikátor stavu zaneprázdnenosti, ktorá je nastavená v prípade ak práve prebieha udalosť s príznakom „busy“ (nedostupný).

3.3.2 Outlook Calendar

Je súčasťou služby správcu osobných informácií Outlook. Tá okrem toho obsahuje e-mailového klienta, správu kontaktov, úloh, značenie poznámok a iné. Korene tohto systému siahajú až do 90. rokov kedy bol súčasťou Exchange serverov a od roku 1997 ako samostatný program v rámci balíku Office 97. Dnes je Outlook stále súčasťou balíku Office, či už klasickej verzie alebo predplatenej Office365, pre Windows aj Mac, tiež existuje v zjednodušenej forme ako aplikácie pošta a kalendár pre Windows 10, vo webovej podobe ako portál Outlook.com a tiež v mobilných systémoch Windows, Android, iOS a ďalších.

Aplikácia kalendára obsahuje základnú funkcionalitu ako vytváranie, listovanie a správa udalostí. Tie obsahujú okrem základného nastavenia nadpisu, popisu a času aj pridávanie

ďalších kontaktov alebo miesta udalosti. Nedostatkom niektorých oficiálnych klientov je nemožnosť nastaviť opakovanie udalosti avšak služba ako celok túto funkcionalitu podporuje. Zaujímavosťou je vytvorenie skupinovej konverzácie v Skype. Možnosť sa zobrazí v rozhraní prípadne je odkaz na pripojenie umiestnený do detailu udalosti.

Prístup k operáciám v prostredí Outlook kalendára umožňuje vývojové prostredie Calendar API [19], ktoré môže byť implementované ako dedikovaná knižnica alebo ako REST. Pri využívaní týchto operácií nad konkrétnym kalendárom je potrebné autorizácia. Používa sa spoločná autorizácia pre všetky aplikačné rozhrania spadajúce pod Outlook API. Ak sa nevyužije vstavanej autorizácia skrz Office 365, je potrebné registrovať aplikácia za využitia protokolov Azure AD v2.

Aplikačné prostredie podporuje napríklad metódy: Get events, Sync events, Find meeting times, Create events, Update events, Delete events, pre editáciu udalostí. Ďalej Get attachments, Create attachments, Delete attachments pre pridávanie príloh k jednotlivým udalostiam a príkazy správy upozornení: Get reminders, Snooze reminders a Dismiss reminders ktoré existujú iba vo forme REST API. Súčasťou vývojového rozhrania sú aj metódy správy jednotlivých kalendárov a skupín: Get calendars, Create calendars, Update calendars, Delete calendars.Je možné editovať všetky kalendárne skupiny okrem základnej.

3.3.3 Calendar od Apple a iné

iCalendar alebo jednoducho Calendar je služba vytvorená spoločnosťou Apple a je prednastavená ako základná kalendárna služba v systémoch macOS a iOS. Podporuje previazanie so servermi od Googlu a Microsoftu.

Z hľadiska podpory aplikácií tretích strán je služba obmedzená v rámci vlastného ekosystému a dostupná iba pre vlastné operačné systémy spoločnosti. V aplikáciách je možné implementovať komunikáciu so serverom kalendára pomocou vývojových nástrojov EventKit Framework [4]. Z architektonického hľadiska je aplikácia pomocou nástrojov previazaná na databázu kalendára v chránenom súborovom systéme. Komunikácia prebieha pomocou proprietárneho protokolu.

Možnosti frameworku sú limitované. Je možné vytváranie či editácia udalostí, ktoré majú klasické parametre ako názov, interval platnosti alebo opakovanie sa. Možnosti nastavenia pripomienok alebo nastavovanie alarmov pre jednotlivé udalosti.

Facebook udalosti Ďalšou možnou službou ktorá do istej miery implementuje udalosti je sociálna sieť Facebook. Na rozdiel od predchádzajúcich systémov nemožno hovoriť o entite kalendára iba o udalostiach, pričom dôraz je kladený na ich zdieľanie s inými používateľmi Facebooku. Čiže systém nie je využívaný na zostavenie súkromnej agendy.

Okrem vytvorenia udalosti v stanovenom čase a intervale nie je možné nastaviť opakovanie alebo vlastnú formu notifikovania. Naopak každá udalosť obsahuje príspevky od používateľov, ktorí majú záujem sa zúčastniť na udalosti alebo priamo od administrátora, ktorý je väčšinou zakladateľom alebo ho zakladateľ určil.

Okrem toho je možné ich vytvárať aj automatizovane a to pomocou vývojového rozhrania nielen pre tvorbu udalostí, Graph API [7]. Rozhranie založené na posielaní GET správ pomocou HTTP protokolu. Tiež je možné využiť vývojové nástroje pre PHP, JavaScript, Android a iOS. Okrem udalostí je zastrešené veľké množstvo funkcionality naprieč celým ekosystémom Facebooku a jeho služieb. Napríklad pridávanie príspevkov, komentárov, správa používateľského účtu ale aj posielanie správ cez Messenger.

Ako parametrami pri vytváraní či editácii udalosti sú napríklad identifikačné číslo, názov, popis, limit maximálneho počtu zúčastnených, obrázok, miesto, čas začiatku a konca a iné. Ďalšími údajmi s ktorými je možné pracovať sú napríklad zoznamy zúčastnených, fotiek alebo komentárov.

3.4 Kognitívne rozpoznávanie

Pojem „kognitívny“ pochádza z latinského cognoscere, čo v preklade znamená poznávať. Kognitívne vedy sa zaoberajú ľudskou myslou, a to najmä inteligenciou, pamäťou, vnímaním, vedomosťami či jazykom. Pod kategóriu kognitívnych vied spadá hneď niekoľko odvetví ako filozofia, psychológia, lingvistika, neurológia, antropológia a čiastočne aj informatika, konkrétne už v úvode spomínaný výskum umelej inteligencie. Proces poznávania je možné rozčleniť do niekoľkých kategórií, konkrétne existuje šesť základných kognitívnych oblastí [25].

- Znalosti – fakty, výrazy, teória a podobne (povedať, vymenovať, označiť).
- Porozumenie – pochopenie významu znalostí (vysvetliť, odlíšiť, dať príklad).
- Uplatnenie – využitie znalostí s porozumením v nových situáciách (vykonať, vyriešiť, použiť).
- Analýza – rozčleniť informácie na elementárne časti a rozpoznať vzťahy medzi nimi.
- Syntéza – usporiadať informácie do nových celkov (navrhnúť, vytvoriť).
- Hodnotenie – posúdiť hodnotu informácií podľa vlastných kritérií (zhodnotiť, zdôvodniť).

Informatika pri výskume umelej inteligencie berie do úvahy všetky spomenuté oblasti [12]. Dnes existuje niekoľko služieb, ktoré vývojárom umožňujú vyvíjať programy s prvkami umelej inteligencie. Ide väčšinou o komplexné riešenia, sady nástrojov alebo frameworkov, prípadne celé systémy, pomocou ktorých je možné naučiť službu napríklad rozumieť ľudskej reči alebo rozpoznávať objekty z obrazov. Aplikácia sa väčšinou zdokonaľuje s časom a množstvom dát.

Všetky takto fungujúce systémy väčšinou využívajú model neurónovej siete. Ten je postavený na základe abstrakcie biologických nervových systémov a využíva schopnosti abstrakcie pravidiel medzi vstupnými a výstupnými hodnotami, čiže takýto model sa zdokonaľuje procesom učenia buď vlastným alebo za pomoci cvičiaceho, ktorý predkladá dáta a označuje ktoré sú správne, a ktoré naopak nie.

V tejto práci sú popísané hlavne systémy, ktoré sa zaoberajú rozpoznávaním hovoreného slova a jazyka. V procese učenia sú im predkladané vety v ktorých sú väčšinou vyznačené kľúčové slová a význam danej vety. Kľúčovým slovom vo vete: „Čo robí Peter?“ je meno Peter, viažuce sa na konkrétnu osobu. Meno sa nahradí parametrom a následne sa celej vete priradí význam otázky na prácu. Systém by tak iba po jednej vete mal rozpoznať, že veta „Čo robí Jana?“ je otázka na prácu ktorú vykonáva osoba Anna.

3.4.1 LUIS

Microsoft Cognitive Services[3] je súbor vývojových nástrojov pre spracovávanie reči, jazyka, obrazu a vedomostí za pomoci počítačovej inteligencie a algoritmov umelej inteligencie. Aplikácie implementujúce metódy týchto nástrojov dokážu spracovávať hovorené slovo, vyhodnocovanie akcií na základe prirodzeného jazyka, rozpoznávanie objektov z obrázkov a fotografií, vyhľadávacie algoritmy či mapovanie dát za účelom vyriešenia úloha sémantických vyhľadávaní. Takýmto nástrojom je aj služba LUIS.

LUIS je skratka pre Language Understanding Intelligent Service [18] čo v preklade znamená inteligentná služba pre porozumenie jazyka. Pomocou vytvárania intentov (v preklade úmysel myslí sa význam vetu) a objektov, ktoré sú vyčlenené a typovo zaradené v procese učenia, je možné vytvárať jednoduché modeli úloh. Tie je následne možné napájať na koncové body cez protokol http a v závislosti na využití vývojových nástrojov, webová služba získava schopnosť vykonať akciu na základe rozkazu zadaného v prirodzenom ľudskom jazyku. Vývojové nástroje pre koncové body sú dostupné pre platformy C#, Node JS, Android a Python.

Okrem vývojových nástrojov je dostupné aj aplikačné rozhranie v REST podobe. Ide o JSON štruktúrované správy. Pomocou nich sa dá odkazovať na jednotlivé koncové body, prípadne autorizovať vlastnú aplikáciu. Medzi koncové body patria jednotlivé úkony ako manažment aplikácie, správa kľúčov, vytváranie nových fráz a vzorov, ich správa a iné. LUIS má v súčasnosti schopnosť rozlišovať anglický, francúzsky, taliansky, nemecký, španielsky, portugalský, kórejský, čínsky, a japonský jazyk.

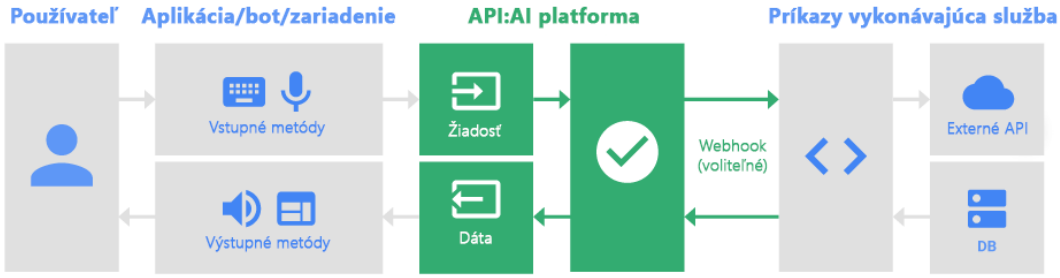
3.4.2 API.AI

Api.ai [3] je platforma pre rozpoznávanie jazyka, umožňujúca jednoduchú integráciu do používateľských konverzačných rozhraní pre mobilné, webové a aplikácie stolných systémov a botov. Systém prijíma otázky (anglicky queries), ktoré sa následne vyhodnocujú a prepájanú na intent, ktorý sa najviac zhoduje so štruktúrou a dátami otázky. Systém buď reaguje priamo odpoveďou alebo ak je na konkrétny intent nastavený vzdialený odkaz, webhook, systém vzdialene vyvolá akciu na serveri. Rozhranie sa tak nachádza medzi frontend aplikáciou alebo botom a backendom v podobe serveru alebo webovej aplikácie. Popis architektúry je znázornený na obrázku 3.4.

Napojenie na systém je opäť možné pomocou GET a POST HTTP správ, prípadne pomocou vývojových nástrojov (SDK) pre Android, iOS, JavaScript, Node.js, .NET, Xamarin, C++, Python, Ruby, PHP, Java, Cordova, Unity a BotKit (Slack). V niekoľkých systémoch, zahŕňajúcich IM, inteligentných asistentov a IoT, je api.ai priamo integrované pre zjednodušené vytváranie chat botov alebo takzvaných zručností pre asistentov.

Pre chod aplikácie využívajúcej api.ai je potrebné vytvoriť nový profil – agenta. V rámci webového rozhrania je potom možné vykonávať nastavenia, prípadne priamo vytvárať úmysly (intenty). Aplikačné rozhranie api.ai komunikuje v URL formáte so základnou adresou ku ktorej sa pripájajú jednotlivé súčasti a parametre. Takto je možné odovzdávať textové otázky (queries), prevod textu na reč, manažment entít, úmyslov a kontextu konverzácie. Prístup do systému je ošetrený pomocou kľúčov, ktoré je možné vygenerovať v nastaveniach agenta. Existujú vždy dva kľúče, klientský umožňuje iba posielanie otázok, vývojársky aj manažment systému.

[3] https://www.microsoft.com/cognitive-services/



Obr. 3.4: Diagram napojenia systému api.ai medzi aplikáciu v koncovom zriadení a server pracujúci v pozadí. Systém rozpozná informácie z normálneho textu a spracované ich prepošle na server ktorý nad nimi vykoná ďalšie operácie. Obrázok je prevzatý z oficiálnej dokumentácie [3].

Systém dokáže rozpoznávať angličtinu, nemčinu, španielčinu, portugalčinu, taliančinu, holandčinu, čínštinu, japončinu, kórejčinu, ruštinu a ukrajinčinu. Niektoré jazyky však systém nedokáže stopercentne rozpoznať.

Kapitola 4

Popis rezervácie vstupeniek

Proces rezervácie vstupeniek je možné vnímať ako celkový postup, ktorého súčasťou sú akcie potrebné vykonať pred aj po samotnom akte rezervácie. Proces je súčasťou celkového systému, ktorý môže poskytovať aj vedľajšie avšak tie by mali po správnosti súvisieť s hlavnou úlohou. Výstupom rezervácie je platná vstupenka.

Vstupenka je doklad o umožnení vstupu do oblasti alebo budovy za účelom účasti na nejakej expozícii alebo udalosti. Môže byť získaná po zaplatení poplatku alebo aj zdarma a nadobúda buď fyzickú alebo digitálnu podobu s špecifickými poznávacími znakmi, prípadne je identifikovateľná pomocou kódu.

Keďže vstupenky je možné vnímať ako niečo čo si dopredu zákazník vyberie a následne zaplatí je možné systém rezervácie vnímať ako špecifický druh internetového obchodu, za účelom predávať vstupenky na akcie a udalosti v rámci jednej alebo aj viacerých organizácií. Ako špeciálne požiadavky na produkt môžu byť určenie konkrétneho sedadla alebo dátumu konania ak ide o udalosť ktorá sa opakuje. Systém však s týmito parametrami môže pracovať podobne ako u iného tovaru.

Z pohľadu postupnosti úkonov, potrebných na vykonanie rezervácie by sa mal systém skladať z týchto dôležitých častí:

- možnosť výberu zo zoznamu produktov (vstupeniek);
- zadanie príslušných parametrov (miesto a čas) a uloženie produktu do entity košíku;
- dokončenie objednávky a platba.

Pri vyberaní produktov, v tomto prípade vstupeniek, je najlepšie zobrazovať ich v určitej forme zoznamu a prípadne ich roztriediť podľa parametrov alebo kategórií. V prvom rade je nutné si uvedomiť, že zákazník si nevyberá vstupenku ale udalosť alebo predstavenie, ktoré chce ísť navštíviť. Preto je ponuka zobrazovaná ako predstavenie, so všetkými údajmi o ňom. Posun v procese rezervácie je obvykle riadený tlačidlom „zarezervovať vstupenku“, či v prípade výberu parametrov objednávky tlačidlom v zmysle „zobraziť termíny“ alebo „náhľad na voľné sedadlá“.

Po zadaní všetkých parametrov je možné vložiť ponuku do virtuálneho nákupného košíku. Ten je možné ľubovoľne editovať pridávaním alebo odoberaním položiek alebo celý obsah zrušiť. V niektorých prípadoch je možné po výbere prejsť rovno k dokončeniu objednávky. Oproti košíku sa zvolený tovar neukladá do žiadnej entity a pri prerušení objednávky sa priebeh môže stratiť. Z architektonického hľadiska je preto dobré vždy implementovať istú entitu, ktorá buď ukladá stav objednávky alebo je to košík s obmedzenou kapacitou.

Dokončenie objednávky sprevádza u klasických internetových obchodov okrem výberu spôsobu platby aj výber spôsobu doručenia tovaru. Keďže lísty objednávané po internete je dnes možné predložiť v digitálnej forme alebo vytlačené doma na tlačiarni nie je potrebné implementovať spôsob dopravy. Pri dokončení objednávky sa zobrazuje zoznam objednaného tovaru a celková suma. Až na konci sa objednávka potvrdí a používateľ je vyzvaný k uskutočneniu platby formou, ktorú si zvolil.

Ako dodatočná funkčná časť aplikácie môže byť prihlásenie používateľa do systému a správa osobných dát, ktoré uľahčuje proces rezervácie tým, že systém nepotrebuje pri každej objednávke požiadať o fakturované dáta.

Pri programovaní webových služieb za účelom predaja lístkov alebo tovaru obecne sa dnes využíva veľké množstvo vývojových nástrojov, programovacích jazykov, frameworkov, aplikačných rozhraní ako pre frontend tak aj backend aplikácie.

V súčasnosti existuje niekoľko chat botov, ktoré umožňujú prehliadanie tovaru na základe kategorizácie alebo parametrického vyhľadávania. Väčšinou však nie je možné tovar pridávať do košíka a dokončiť objednávku priamo v prostredí konverzácie. Pomocou odkazu alebo tlačidla pod konkrétnym tovarom je používateľ nasmerovaný na internetovú stránku obchodu.

Kapitola 5

Návrh riešenia

Pri návrhu riešenia je dôležité vytvoriť niečo, čo nebude kopírovať hotové riešenia ale poskytne náhľad na pokročilú funkcionalitu týchto riešení. Tým sa myslí, že v oblasti botov, ktoré istým spôsobom vykonávajú rezervácie predstavení alebo filmov, existujú v kapitole 4 popísané riešenia. Výstupom tak nebude aplikácia bota, ktorý dokáže komunikovať a zobrazovať dáta ohľadom predstavení. Úlohou je vytvoriť riešenie, ktoré poskytne náhľad na možnosť integrácie platobných brán, kalendárneho systému či osobného asistenta.

Výstupné riešenie si možno predstaviť ako demo ukážky týchto funkcionalít. Navyše bude zohľadňované aj použitie pokročilých zobrazovacích elementov, ktoré umožnia lepšiu interakciu v danej problematike a uľahčia používateľom prechod konverzáciou. Pre každé zo skúmaných rozšírení bude existovať práve jedno riešenie, ktoré môže byť implementované separátne alebo ak to bude možné, tak aj v rámci jednej aplikácie.

Z pohľadu architektúry výsledného riešenia je potrebné najprv zadefinovať, s ktorou vývojovou platformou sa bude pracovať. Dôležitosť vykonania tohto kroku pred samotným návrhom spočíva v tom, aby boli pri jeho vytváraní zohľadnené všetky možnosti, ktoré poskytuje platforma a jednotlivé vývojové nástroje. Je dobré vyvarovať sa návrhu častí, ktoré neskôr nebudú môcť byť implementované skrz technologickú bariéru. V horšom prípade by mohol byť nepoužiteľný celý návrh.

5.1 Výber vhodnej platformy

Dôležitým bodom je výber vhodnej vývojovej platformy, keďže od nej sa budú odvíjať požiadavky na vývojové nástroje tretích strán ale aj základ samotného návrhu. V kapitole 2 bolo predstavených hneď niekoľko platforiem, pre ktoré je možné vytvárať chat botov. Pre vytvorenie čo najuniverzálnejšieho rešenia bude nutné vybrať platformu tak, aby pokryla čo najviac komunikačných kanálov. Zároveň však musia byť implementované pokročilé elementy, ktoré budú uľahčovať vizualizáciu dát alebo prechod konverzáciou.

Podľa popísaných parametrov je v tomto smere najvhodnejšou a zároveň aj jedinou správnou voľbou *Microsoft Bot Framework*. Vďaka službe *Bot Connector* sa chat bot môže napojiť až na 8 komunikačných služieb (*Facebook Messenger*, *Skype*, *Microsoft Teams*, *Slack*, *Telegram*, *Kik Messenger*, *GroupMe*, *Twilio*). Tým pádom nebudú podporené služby *Line* a *WeChat*. Nezahrnutie týchto služieb nebude prekážkou, keďže aj skrz svoju popularitu a rozšírenosť cielia najmä na ázijský trh. Ostatné vývojové nástroje a platformy boli vždy viazané na konkrétnu službu a preto nemohla byť ani jedna z nich uvážená ako možné riešenie.

5.1.1 Výber kalendárnej služby

Pri výbere vhodných kalendárnych služieb bolo braných do úvahy hneď niekoľko faktorov, ale konkrétne najmä dva najdôležitejšie, a to rozšírenosť danej služby a možnosti vývojových nástrojov. Ďalej sa prihliadalo aj na zamerania jednotlivých kalendárnych služieb, a síce či odpovedajú zameraniu aplikácie.

Ak sa nebude prihliadať na rozšírenosť, hlavne kvôli skutočnosti, že sa v tejto práci vyskytujú iba takto definované služby, jediným parametrom sú možnosti jednotlivých vývojových nástrojov a rozhraní. Keďže chat bot je webová aplikácia, je pochopiteľné, že ak niektorá zo služieb tento formát nepodporuje, nemôže byť súčasťou návrhu. Práve tomuto kritériu nevyhovuje služba *Apple Calendar*, keďže tá je obmedzená iba na aplikačné rozhranie pre systémy *iOS* a *macOS*.

Zo zadania vyplýva, že chat bot sa pri vytvorení udalosti orientuje najmä na vytvorenie udalosti v súkromnej agende, nie na vytvorenie verejného eventu, a aj keď udalosti vytvorené v systéme *Facebooku* môžu byť aj súkromné, služba primárne neplní úlohu agendy. Preto bolo rozhodnuté sa na túto platformu neorientovať. Tým pádom jediné spomenuté služby, ktoré splňujú požiadavky sú kalendárne služby od Microsoftu a Google. Keďže služba *Outlook Calendar* podporuje iba základné aplikačné rozhranie. Bude pri implementácii uprednostnený kalendár od Google. Ten podporuje pokročilejšiu sadu vývojových nástrojov pre jazyk C#.

5.1.2 Výber vhodnej platobnej brány

Hlavnou podmienkou pri výbere vhodnej platobnej brány je čo najväčšia univerzálnosť rozhrania. Riešením tak bude použitie niektorej z nezávislých služieb *Masterpass* alebo *Paypal*. Obidve služby majú okrem základného aplikačného rozhrania aj vývojové nástroje dostupné pre jazyk C#. *Masterpass* je však na rozdiel od *Paypalu* viac viazaný na webové rozhranie, a aj keď implementácia umožňuje programovanie v jazyku C#, pri potvrdení platby je potrebné vytvoriť špecifické tlačidlo v kóde stránky a k nemu následne importovať príslušný Javascript súbor. Ten zabezpečuje presmerovanie na interné stránky Masterpassu, kde používateľ potvrdí svoju platbu. Problémom je tiež callback adresa, čiže odkaz presmerovania po vykonaní platby.

Toto je problém aj *Paypalu*, ktorý síce nie je nutne viazaný na rozhranie html stránok ale aj tak je pri vytváraní platby nutné zadať callback adresu. Bohužiaľ neexistuje spôsob, ako rozumne vyriešiť presmerovanie naspäť do konverzácie s botom. V návrhu a implementácii však bola aspoň snaha o minimalizáciu tohto problému.

5.1.3 Výber asistenčnej služby

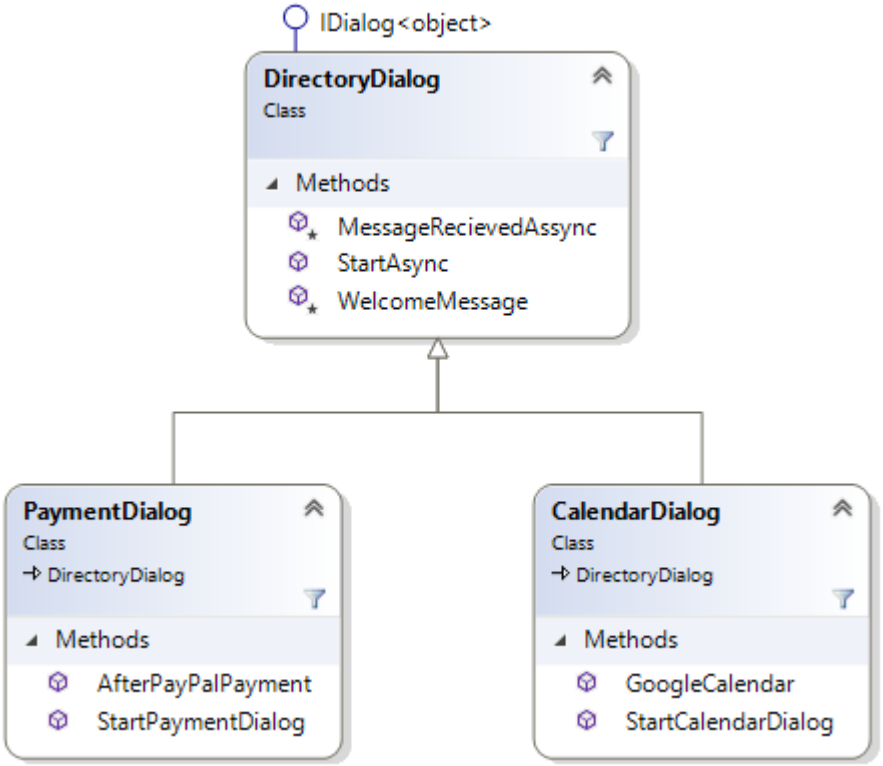
Výber vhodnej asistenčnej služby bol vcelku jednoduchý. Najdôležitejším faktorom, ktorý musela platforma splniť, bola existencia aplikačného rozhrania alebo vývojových nástrojov, ktoré umožňujú priamu komunikáciu s chat botom. Bohužiaľ, ani jedna zo služieb, ktoré boli spomenuté v kapitole 3.1, túto požiadavku zatiaľ nespĺňa.

Apple Siri je podobne ako Apple Calendar obmedzená iba pre aplikácie naprogramované skrz vývojové nástroje pre iOS a macOS. Tiež podporuje iba niekoľko základných spôsobov interakcie s aplikáciami tretích strán. *Google Assistant* podporuje „akcie“, ktoré rozširujú funkcionalitu asistenta o nové možnosti. Podobne fungujú aj „zručnosti“ (anglicky „skills“) pre *Amazon Alexa* a *Microsoft Cortana*. Riešenia od Google a Amazonu pritom fungujú výhradne ako prepojenia na webové služby.

Aplikačné rozhranie *Cortany* umožňuje okrem prepojenia na webovú službu aj napojenie Windows 10 a Android aplikácií a podľa náznakov v dokumentácii aj na Chat Botov, avšak po hlbšom skúmaní sa zistilo, že táto časť rozhrania ešte nie je dostupná skrz ohlásenie tejto funkcionality a spustení prihlášok do uzatvoreného testovania doposiaľ nebolo umožnené pracovať s danými nástrojmi.

5.2 Návrh architektúry

Odhliadnuc od základnej triedy kontroly správ sa bot po aplikačnej stránke skladá z jednotlivých tried dialógov. V úvode kapitoly bolo spomenuté, že pri implementácii jednotlivých častí sa bude postupovať buď cestou vytvorenia samostatných programov pre každú rozširujúcu funkcionalitu alebo budú všetky implementované v rámci jedného projektu. Práve vďaka triedam dialógov je možné prikloniť sa k druhej možnosti. Základom tak bude jedna rodičovská trieda dialógu, z ktorej sa budú odvíjať ďalšie triedy, vždy jedna pre konkrétnu implementáciu rozšírenia. Hierarchia tried je znázornená na obrázku 5.1).



Obr. 5.1: Návrh rozdelenia programu na tri základné časti reprezentované triedami, kde rodičovská trieda *DirectoryDialog* dedí priamo z triedy dialógu *IDialog*. Podtriedy sú rozdelené kvôli implementácii odlišných rozširujúcich fukcionalít.

Z pohľadu prechodu konverzáciou bude rodičovská trieda reprezentovaná rozcestníkom, ktorý sa bude zobrazovať vždy na začiatku konverzácie a bude mať podobu karty s textom a tlačidlami. Pomocou nich sa budú inicializovať triedy jednotlivých rozšírení. V rámci rodičovskej triedy budú úvodné inicializácie týchto ukážok v podobe karty s textom a tlačidlami.

Zahrnutie vykreslenia úvodných hlášok do rodičovskej triedy je spôsobené architektúrou samotného frameworku, kedy je možné prepnúť kontext konverzácie do inej triedy avšak ten už následne čaká na ďalšiu akciu zo strany používateľa. Zobrazenie možností v danej podtriede tak musí byť vykonané ešte predtým ako je na ňu kontext presmerovaný, čiže v triede rodičovskej.

5.2.1 Návrh štruktúry platobnej brány

V existujúcom webovom riešení je platobná brána súčasťou vykonania platby za objednaný tovar, v tomto prípade za lístky na predstavenie. Vykonaniu platby predchádza niekoľko úkonov a ak zjednodušíme vyberanie predstavenia, termínu a obsadenia určitého počtu voľných miest na jednoduchý výber položky (tovaru) bez akejkoľvek varianty, ostanú tri úkony, ktoré pri návrhu musia byť pokryté.

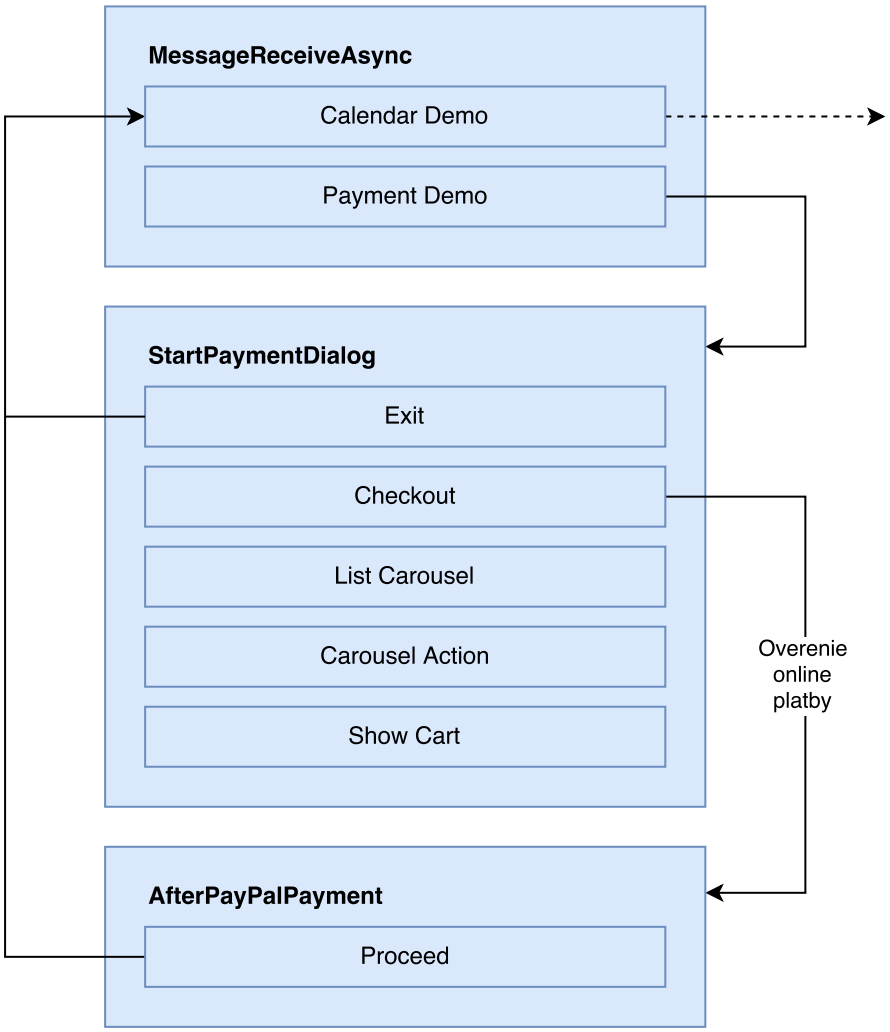
- Pridávanie položiek do košíka.
- Zobrazenie položiek s možnosťou vytvorenia platby.
- Autorizácia a dokončenie platby.

Tak ako je zobrazené na obrázku 5.2, sú po stránke prechodu aplikáciou navrhnuté jednotlivé moduly a v rámci nich možnosti, ktoré môžu, ale aj nemusia meniť kontext. Výber položiek je možné realizovať pomocou zoznamu kariet, ktorý sa v oficiálnych dokumentoch označuje ako *„carousel“* (v preklade kolotoč). Medzi kartami je možné navigovať sa posunom v horizontálnej rovine. Pri každej položke bude okrem obrázku a štrukturovaného textu aj tlačidlo, ktoré po kliknutí vykoná pridanie konkrétneho tovaru do košíka v počte 1 kus. Pridávanie položiek do košíka je teoreticky možné opakovať nekonečne veľa krát.

Všetky doteraz spomenuté úkony je možné ľubovoľne kombinovať, keďže sú navrhnuté v rámci jednej funkcie. Každá takáto funkcia je asynchrónna. Dôležitý je pritom kontext dialógu, ktorý nielenže umožňuje posielať asynchrónne správy, ale aj opakovane čakať na ďalší podnet od používateľa pomocou cyklického odkazovania sa na seba. Presun medzi jednotlivými metódami je možný porušením cyklu. Základná logika všetkých doterajších úkonov bola vykonávaná iba v rámci jednej metódy s názvom *StartPaymentDialog*.

Pri stlačení tlačidla „potvrdiť objednávku“ sa v karte košíka vykoná vytvorenie objednávky, pričom nastane presun kontextu medzi metódami a ten momentálne čaká na odpoveď vo funkcii *AfterPayPalPayment*. Okrem toho sa vygeneruje *Paypal* platba na základe dát korešpondujúcich s položkami vloženými do košíka. Vytvorenie platby sa skaladá z niekoľkých po sebe nasledujúcich úkonov.

- Vytvorenie zoznamu jednotlivých položiek. Každá položka reprezentuje jeden typ objednaného tovaru a obsahuje názov, menu, cenu a kvantitu.
- Vytvorenie objektu osoby platiteľa.
- Zadefinovanie návratových adries pre úspešné aj neúspešné overenie platby.
- Vytvoriť celkovú čiastku, ktorá musí byť rovnaká ako súčet cien všetkých položiek obsiahnutých v platbe.
- Vytvorenie zoznamu transakcií obsahujúcich popis, zoznam položiek a celkovú čiastku.
- Nakoniec vytvoriť objekt platby obsahujúci zámer platby (napríklad predaj), platiteľa, návratové adresy a zoznam transakcií. Následne sa pomocou metódy objektu platba vytvorí.



Obr. 5.2: Prechod aplikáciou pre úkážku platieb implementuje tri základné metódy, v ktorých sú pomocou rozhodovacej logiky jednotlivé akcie, ktoré môžu meniť kontext konverzácie na inú metódu.

Po vytvorení *PayPal* platby je na používateľovi, aby ju autorizoval, a tým aj dokončil. Tým že sa pri vytvorení presmeroval kontext na novú metódu, nie je možné pristupovať k zoznamu predstavení, pridávať ich do košíka, či košík zobrazovať. Používateľ môže overiť platbu po kliknutí na tlačidlo, ktoré je navrhnuté v rámci karty vygenerovanej pri vytvorení platby. Používateľ je presmerovaný na webovú stránku, ktorej adresa je vrátená vždy pri vytvorení platby.

Adresa odkazuje na webovú stránku *PayPalu*, kde sa používateľ prihlási k svojmu účtu a potvrdí objednávku. Následne je presmerovaný na návratovú adresu, ktorá bola zadaná pri vytváraní platby. Ako už bolo spomenuté pri výbere vhodnej platformy, *PayPal* neumožňuje inú logiku ako je presmerovanie po overení platby na návratovú adresu a zatiaľ ani neexistuje možnosť presmerovať sa pomocou url adresy priamo do konverzácie s daným chat botom.

Preto bol navrhnutý kompromis v podobe vytvorenia návratovej adresy za využitia faktu, že bot je v podstate webová služba. V rámci projektu tak bola vytvorená jednoduchá stránka s logikou automatického dokončenia platby na pozadí. V konverzácii sa následne vygeneruje text o potvrdení platby navádzajúc používateľa naspäť do prostredia konverzácie s botom. V nej sa ukončí konverzácia a tým aj demo ukážka platby a kontext sa vráti na úvod do rodičovského dialógu.

5.2.2 Návrh štruktúry manažmentu kalendára

Keďže toto rozšírenie je aj nad rámec funkcionality webovej stránky, prvé čo je vhodné spraviť, je zadefinovať časť v pôvodnom systéme v ktorej by sa mohlo nachádzať. V skratke by mala navrhovaná funkcionalita zo zoznamu sledovaných alebo už objednaných predstavení vytvoriť udalosť v prostredí kalendárnej služby s dátumom a časovým rozpätím daného predstavenia. Pre potreby návrhu názornej ukážky tejto funkcionality je možné zjednodušiť výber predstavení na jeden zoznam.

Prv než príde rad na návrh vybratia daného predstavenia, bude potrebné nastaviť účet a prepojenie na aplikačné rozhranie kalendárneho systému, ktorým bol v podkapitole 5.1.1 zvolený *Google Calendar*. Autorizácia aplikácie voči nástrojom Googlu tak môže prebehnúť počas úvodnej fázy ukážky. Po presmerovaní z inštancie hlavnej rodičovskej triedy sa zobrazí ešte v jej rámci informatívna karta s možnosťou prihlásenia sa do danej kalendárnej služby. Po stránke kontextu sa však už čaká na akciu v novo vytvorenej inštancii triedy CalendarDialog.

Po stlačení tlačidla sa aktivuje v metóde *StartCalendarDialog* prístupový bod a používateľ sa bude môcť prihlásiť k svojej službe. Tento úkon prebieha presmerovaním do prehliadača, kde používateľ zadá prihlasovacie údaje k svojmu Google účtu. Následne je po overení opäť presmerovaný na informačnú stránku, ktorá mu oznámi, že sa môže vrátiť do chatu. Opäť sa musel zvoliť systém nabádania používateľa pre návrat do konverzácie. Je zrejmé že toto riešenie nie je optimálne, ale momentálne je najlepšie aké možno navrhnúť.

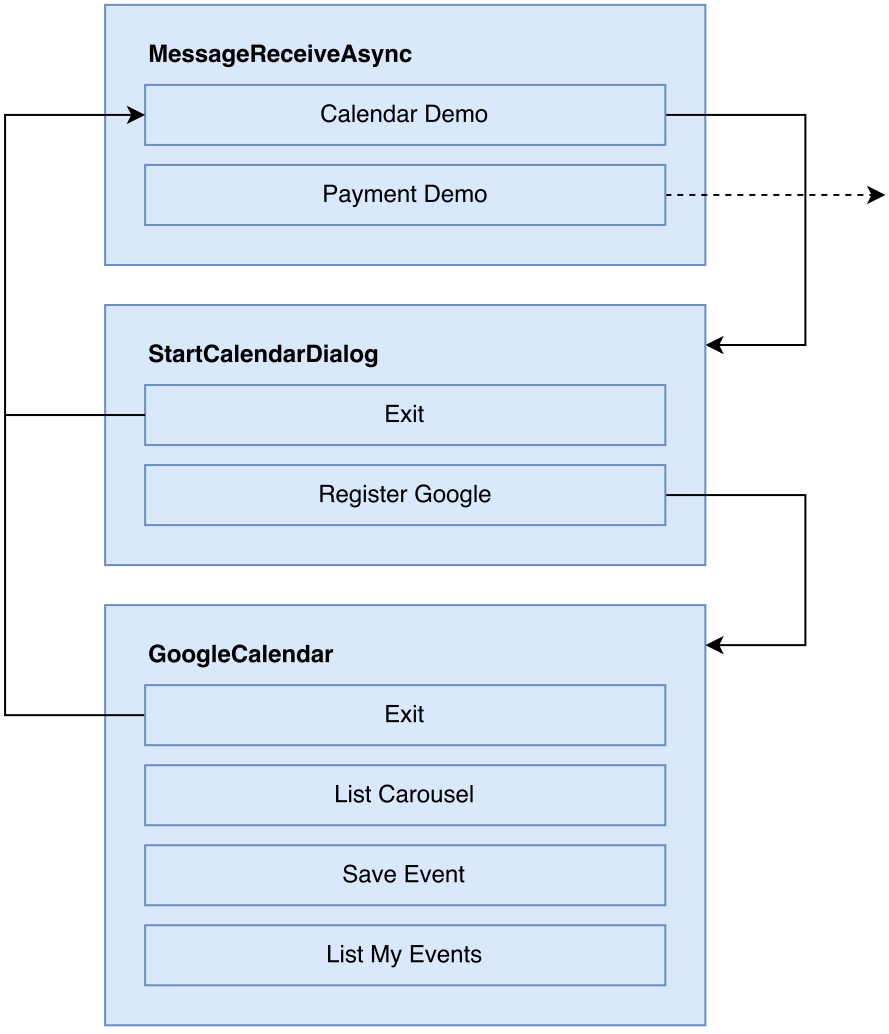
Po návrate do konverzácie sa vygenerovala jednak nová karta a kontext sa opäť presunul a momentálne čaká na odozvu v metóde *GoogleCalendar*. Odozva, ktorú používateľ zadá skrz jednu z možností zobrazenej vo forme tlačidiel, môže byť okrem ukončenia danej ukážky zobrazenie predstavení s možnosťou pridania do kalendára alebo zobrazenie nadchádzajúcich predstavení uložených v kalendári.

Pri prihlásení do služby sa vytvorí nový kalendár v rámci používateľovho účtu. Zámerom k jeho vytvoreniu bola skutočnosť, aby sa v rámci ukážky neukladali dáta do reálne používaných kalendárov. Kalendár má unikátny názov, podľa ktorého je rozpoznateľný v danom systéme. Po korektnom skončení dema sa kalendár vždy automaticky vymaže. Ak by na-

stal problém a ukážka alebo celý bot by sa neukončil správne, kalendár zostane v systéme. Ak však používateľ opäť spustí danú ukážku a bude stále prihlásený pod rovnakým Google účtom, nevytvorí sa nový kalendár, ale použije ten sa stávajúci s daným unikátnym názvom.

Zobrazenie predstavení je podobne ako pri deme platieb realizované pomocou zoznamu kariet, ktoré dokopy tvoria už spomínaný carousel. V tomto prípade sa však pomocou tlačidla vytvorí nová udalosť. Následne sú spracované údaje o predstavení ako čas začiatku, čas konca a názov predstavenia. Tie sú vložené pripraveného objektu.

Druhou možnosťou bolo zobrazenie nadchádzajúcich udalostí. Vytvorí sa žiadosť na zobrazenie zoznamu udalostí, ktorá môže mať dodatočné parametre ako minimálny dátum konania, maximálny počet vrátených odpovedí, spôsob zoradenia a iné. Žiadosť je nastavená tak, aby vrátila 10 najbližšie sa konajúcich akcií od aktuálneho času. Ak nenájde žiadny, vráti tomu príslušnú odpoveď. Po ukončení ukážky sa kontext vráti do rodičovského objektu *DirectoryDialog*. Podobne ako pri návrhu ukážky platieb je aj vyššie popísaná logika zhrnutá na obrázku 5.3.



Obr. 5.3: Podobne ako u ukážky platieb sú aj tu navrhnuté metódy a akcie, ktoré môžu meniť kontext. Návrhy sa prekrývajú v metóde MessageReceiveAsync.

Kapitola 6

Implementácia riešenia

Z návrhov popísaných v predchádzajúcej kapitole vyplýva, že je možné naprogramovať chat bota, ktorý bude do istej miery implementovať pokročilé funkcie platobnej brány a kalendárneho systému. Pred začatím implementácie je však nutné okrem porozumenia jednotlivým vývojovým nástrojom a inštalácie potrebných súčastí a programov, ktoré budú použité pri vývoji, vykonať aj nastavenie profilov a prístupových kľúčov k jednotlivým aplikačným rozhraniam.

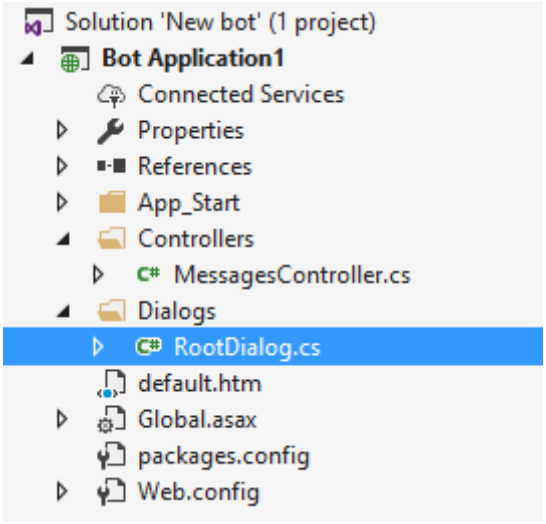
- Pre *Paypal* je potrebné najprv na developerskom portáli vytvoriť novú REST API aplikáciu, ktorá po vytvorení vygeneruje unikátny identifikátor Client ID a tajný prístupový kód Secret. Tieto údaje sa následne skopírujú do konfigurácie aplikácie. Vhodné je vytvoriť aj testovací účet v rámci vývojového režimu.
- Po prihlásení sa do vývojárskeho rozhrania, ktoré *Google* označuje ako konzolu, je potrebné v ponuke vytvoriť nový overený prístupový bod, takzvané *credentials* (v preklade poverovací list). Pri vytváraní sa vyberie spôsob overenia cez *OAuth 2.0* a adresa presmerovania, na ktorej príde k overeniu účtu. Následne sa stiahne konfiguračný súbor vo formáte JSON, ktorý sa vloží do projektu. Pri inicializácii rozhrania sa ďalej pracuje s týmto súborom.

Pri vytváraní nového projektu vo Visual Studiu bude ako šablóna použitá predom stiahnutá Bot Application. Po vytvorení je možné začať s implementáciou. Novo vytvorený projekt obsahuje niekoľko súborov a zložiek. Najdôležitejšia z pohľadu ďalšej implementácie bude zložka *Dialogs* kde budú vytvorené všetky súbory potrebné pre chod bota. Celkový pohľad na základnú štruktúru poskytuje obrázok 6.1.

6.1 Vytvorenie základnej štruktúry

Ako už bolo popísané v návrhu, aplikácia bota sa skladá z troch hlavných častí, ktoré sa budú sa nachádzať v zložke *Dialogs*, pričom každá z nich bude implementovaná ako trieda zadefinovaná vo vlastnom súbore. Triedy potomkov sú zatriedené do vlastného adresára. Súbor *DirectoryDialog.cs*, v ktorom sa nachádza rodičovská trieda *DirectoryDialog*, je lokalizovaný priamo v priečinku *Dialogs*. Súbory *CalendarDialog.cs* a *PaymentDialog.cs* sa nachádzajú v podzložkách *CalendarDemo* a *PaymentDemo*.

Po spustení bota v konkrétnej konverzácii sa z programového hľadiska ako prvá funkcia spustí asynchrónna Post metóda triedy *MessagesController*. Keďže táto trieda aj metóda



Obr. 6.1: Ukážka štruktúry súborov po vytvorení projektu zo stiahnutej šablóny. Väčšina architektonických záležitostí ako základný MessagesController a konfiguračné nástroje sú už vytvorené. Najviac vlastnej implementácie bude zložke Dialogs.

boli pripravené popredu v rámci šablóny, jedinou zmenou bude prepísanie *IDialog* triedy z preddefinovanej na *DirectoryDialog*.

```
await Conversation.SendAsync(activity, () => new DirectoryDialog());
```

Pri implementácii triedy *DirectoryDialog* je dobré sa inšpirovať v už existujúcej predpripravenej dialógovej triede. Tá obsahuje dve základné asynchrónne metódy typu Task. Základom je inicializačná metóda *StartAsync*, ktorú je vždy nutné implementovať ak má byť trieda dialógu inicializovaná skrz *MessagesController*. Jediným parametrom je objekt triedy *IDialogContext*, ktorý zabezpečuje proces konverzácie. Dôležitou je metóda *Wait*.

```
context.Wait(NextTask);
```

Metóda má za úlohu uviesť aplikáciu do stavu čakania, pokým nepríde nová správa od používateľa. V tom prípade sa vykoná príslušná akcia, ale už vo funkcii určenej parametrom. Keďže reakcie používateľa sú asynchrónneho charakteru, musia byť aj všetky *Task* funkcie asynchrónne.

Ďalšie funkcie obsahujú okrem kontextu aj parameter správy, ktorá je definovaná ako IMessageActivity a obsahuje informácie o správe a aj o odosielateľovi. Na základe týchto poznatkov je možné utvoriť si vlastnú štruktúru základného dialógu obsahujúcu funkcie: *StartAsync*, *WelcomeMessage* a *DirectoryLogic*.

Za účelom naviesť konverzáciu smerom k spusteniu danej ukážky vytvára metóda WelcomeMessage kartu typu „Hero“ s textom a tlačidlami. V priebehu implementácie bolo zistené, že tieto karty sú použité naprieč celou aplikáciou a kód sa stal na určitých miestach redundantný. Preto bola vytvorená nová trieda v rámci nového súboru *GlobalDialogFunctions.cs* obsahujúca generovanie kariet, tlačidiel a iných grafických komponentov.

Karty typu „hero“ sú reprezentované ako inštancie triedy *HeroCard* implementovanej v rámci frameworku. Objekt karty sa do odosielanej správy priraďuje cez zoznam príloh. Trieda obsahuje niekoľko členov, ktoré súvisia priamo s výsledným vykresleným obsahom a to konkrétne:

- *Title, Subtitle, Text* – (názov, podnázov, popis) textové polia odlíšené iným formátovaním;
- *Images* – zoznam objektov triedy CardImage, čiže obrázkov;
- *Buttons* – zoznam objektov triedy CardAction reprezentujúcej tlačidlá.

6.2 Platobná brána

Prvou funkciou triedy *PaymentDialog*, na ktorú sa presmeruje kontext konverzácie, je *StartPaymentDialog*. Obsahuje rozhodovaciu logiku ukážky platby, ktorá prebieha na dvoch úrovniach. Rozdiel v úrovniach tvorí hlavne presúvanie kontextu. Kým na prvej úrovni sú zadefinované tri hlavné možnosti spojené zároveň s metódou Wait, ktorá v každom z prípadov používa ako parameter inú funkciu, v druhej úrovni sa rozlišuje iba typ príkazu a akcia, ktorá sa má vykonať. Štruktúra akcií je zrejmá z návrhového obrázka 5.2.

Zaujímavou entitou v rámci implantácie je košík, do ktorého sa vkladajú údaje o kúpených lístkoch pred tým ako sa vytvorí platba. Ten nie je implementovaný ako objekt, ale iba ako jednoduché pole čísiel typu integer. Dôvodom minimalizácie košíka na pole je ten, že pri výbere lístkov je možné vyberať iba z jedného zoznamu, ktorého položky sú spracovávané z konfiguračného súboru *biddings_list.json*. Pozícia konkrétnej položky v JSON súbore reprezentuje index poľa a hodnota na danom indexe počet kúpených kusov. Pri vytváraní košíka sú dáta čerpané opäť zo súboru a k nim sa pripočíta počet kusov z poľa.

Konfiguračný súbor vo formáte JSON sa nachádza v koreňovom adresári projektu ako dodatočný zdroj. Skladá sa z poľa objektov pod identifikátorom „movies“. Pre priblíženie je štruktúra JSON objektu znázornená ako kódu v tejto ukážke 6.1.

```
{
    "title": "",
    "subtitle": "",
    "text": ""
    "image_url": "",
    "button": {
        "title": "",
        "type": "",
        "value": ""
    },
    "time_start": "dd-MM-yyyy/HH:mm",
    "time_end": "dd-MM-yyyy/HH:mm",
    "event_button": {
         "title": "",
         "type": "",
         "value": ""
    }
}
```

Listing 6.1: Štruktúra JSON objektu obsahujúceho dáta predstavení, ktoré sú dôležité pre chod oboch častí aplikácie.

Objekt tak obsahuje informácie potrebné pre vykreslenie karty predstavenia ako názov, podnázov, popis, adresu obrázka. Ďalej entitu tlačidla s údajmi pre naplnenie parametrov

a informácie, ktoré sa budú zobrazovať pri vykreslení v ukážke správy kalendára. Tu sú obsiahnuté časy začiatku a konca udalosti a tiež aj parametre tlačidla pre ukladanie udalosti do kalendára.

Zobrazenie položiek košíka bude realizované pomocou inštancie triedy *ReceiptCard*, čo je iný typ karty ako už spomínaná Hero karta. Okrem už známych členov ako Title alebo Buttons obsahuje aj ďalšie. Člen *Total* značí konečný súčet všetkých položiek. Má vyhradené špeciálne formátovanie, podobne ako *Tax*, značiaci výšku dane, ktorý však v tomto prípade nie je použitý. Člen *Items* obsahuje zoznam položiek košíku a je reprezentovaný triedou *ReceiptItem*. Jednotlivé objekty obsahujú titul, podtitul, obrázok, cenu za jeden kus a kvantitu.

Metóda *AfterPayPalPayment* sa používa po vytvorení platby a následnom vygenerovaní karty s tlačidlom presmerovania na PayPal overovaciu adresu. Tá je mimochodom získaná z objektu *createdPayment*, ktorý vznikol ako návratová hodnota vytvorenia platby. Proces platby prebieha ešte v rámci funkcie *StartPaymentDialog*, z ktorej sa však referuje na špeciálnu funkciu určenú pre obsluhu vytvorenia PayPal platby. Funkcia *PayPalPayment* má návratovú hodnotu typu string a je ňou už spomínaná adresa.

```
var createdPayment = payment.Create(apiContext);
return createdPayment.GetApprovalUrl();
```

Po autorizovaní platby je používateľ presmerovaný pomocou callback odkazu naspať do systému, avšak ako bolo vysvetlené v návrhu, nie priamo do konverzácie. Presmerovanie je nastavené na webovú stránku napísanú pomocou vývojových nástrojov *ASP.NET*, čo umožňuje predovšetkým využitie rovnakého programovacieho jazyka pri dokončení objednávky. Vstavaná metóda *Page_Load* umožňuje pri načítaní stránky na pozadí spustiť skript, ktorý v tomto prípade dokončí platbu Paypal.

```
payment.Execute(apiContext, paymentExecution);
```

Potrebné je tiež ošetriť, aby používateľ, v prípade že ešte neautorizoval a nedokončil platbu, nemohol pokračovať v konverzácii. Za týmto účelom bola implementovaná trieda *GlobalVariables* a v nej statická premenná typu integer, `payment_done`, ktorá má implicitne hodnotu nastavenú na 0. Po úspešnom ukončení sa však jej hodnota nastaví na 1. Zároveň platí pravidlo, že pokiaľ nie je táto premenná rovná 1, tak nie je možné v konverzácii dokončiť objednávku.

6.3 Manažment kalendára

Úvodnou funkciou triedy *CalendarDialog* z pohľadu prochodu aplikáciou je *StartCalendarDialog*, implementujúca prihlásenie do Google účtu. Pri vytváraní entity služby, ktorá v rámci *Google Calendar API* reprezentuje implementovanú aplikáciu, je potrebné určiť jej názov a poverenia. Tie sú získané z údajov uložených v súbore *calendar_sectret.json*. Následne je vytvorený objekt triedy *CalendarService*. Prebeh je zobrazený v náhľade 6.2.

```
1 static CalendarService service;
2 service = new CalendarService(new BaseClientService.Initializer()
3 {
4     HttpClientInitializer = credential,
5     ApplicationName = ApplicationName,
6 });
```

Listing 6.2: Vytvorenie inštancie triedy *CalendarService*, reprezentujúcej kalendárnu službu.

Po vytvorení objektu služby je nutné vygenerovať aj testovací kalendár. Vychádzajúc z náhľadu 6.3, premenná `calendar_existence` určuje počet už existujúcich kalendárov s menom *„bot_demo_calendar“*, ktoré bolo predom vybrané ako unikátny identifikátor. Nový kalendár je tak vytvorený iba v prípade, že je premenná nulová. Okrem mena je vhodné nastaviť aj časovú zónu.

```
1  if (calendar_existence < 1)
2  {
3      Google.Apis.Calendar.v3.Data.Calendar new_calendar_enty = new
           Google.Apis.Calendar.v3.Data.Calendar
4      {
5          Summary = "bot_demo_calendar",
6          TimeZone = "Europe/Bratislava"
7      };
8      Google.Apis.Calendar.v3.Data.Calendar createdCalendar =
           service.Calendars.Insert(new_calendar_enty).Execute();
9      calendarId = createdCalendar.Id;
10 }
```

Listing 6.3: Kód popisujúci vytvorenie nového kalendára v rámci používateľovho účtu.

Po prihlásení je kontext presmerovaný na metódu *GoogleCalendar*. Tá má implementovanú dvojúrovňovú rozhodovaciu logiku podobne ako metóda *StartPaymentDialog* popísaná v kapitole 6.2 a podobne ako v prípade platieb, aj tu je štruktúra akcií zrejmá z návrhového obrázku 5.3. Zdroj dát potrebných pre zobrazenie udalostí, ktoré je možné pridať do súkromnej agendy, je opäť lokalizovaný v konfiguračnom súbore *biddings_list.json*, pričom sú využívané aj parametre `time_start`, `time_end` a `event_button`. Graficky je zoznam udalostí, z ktorých je na výber, realizovaný opäť ako carousel kariet triedy *HeroCard*.

Pri vytáraní novej udalosti sú vo funkcii *SetEvent* na základe indexu spracované korešpondujúce dáta a následne vložené do pripraveného objektu triedy *Event*, ktorej parametre budú obsahovať názov udalosti, začiatok a koniec udalosti, pričom tie sú ukladané ako *EventDateTime* s parametrami dátumu a časovej zóny. Dátum je pritom potrebné spracovať na základe prednastaveného vzoru `dd-MM-yyyy/HH:mm`. Následne je nová udalosť pridaná do kalendára.

```
service.Events.Insert(new_event, calendarId).Execute();
```

Kapitola 7

Spôsoby testovania

Testovanie riešenia prebieha hneď v niekoľkých fázach vývoja a aj po jeho skočení kedy prišlo na rad záverečné ladenie a používateľské testy. Pokrytá bola vlastná implementácia ale aj komunikácia a prepojenie jednotlivých služieb. Dôraz pri testovaní bol kladený hlavne na jednoduchosť a možnosť riešiť všetky testy v rámci vývojového stroja, čiže zbytočne nepresúvať riešenie napríklad do cloudu. Za týmto účelom bolo použitých hneď niekoľko podporných programov.

Bot Framework Emulator[1] je program, ktorý umožňuje vývojárom testovať bota lokálne na zariadení počas programovania. Aj keď program dovoľuje aj testovanie zo vzdialeného zdroja, najviac osohu však prináša práve pri lokálnom vývoji. Programátorom je umožnené pripojiť sa na webovú aplikáciu spustenú z vývojového prostredia pomocou služby *IIS_Express*[2] a následne testovať komunikáciu priamo v konverzácii s chat botom. Emulátor navyše zobrazuje aj informácie o úspešnosti jednotlivých dotazov a štruktúre prijímaných správ v JSON formáte.

Pre otestovanie komunikácie bota s jednotlivými platformami prepojenými cez *Bot Connector* je možné využiť priamo vstavané riešenie v rámci webového rozhrania. Stačí iba nastaviť príslušnú adresu a využiť na to určené textové pole. Systém indikuje, či je aplikácia bota dostupná a komunikuje s konektorom. Pre otestovanie jednotlivých platforiem je možné pomocou používateľského testovania kontaktovať bota v rámci komunikačnej aplikácie. Konektor následne ukladá prípadné chybové hlášky a oznamuje dostupnosť danej služby.

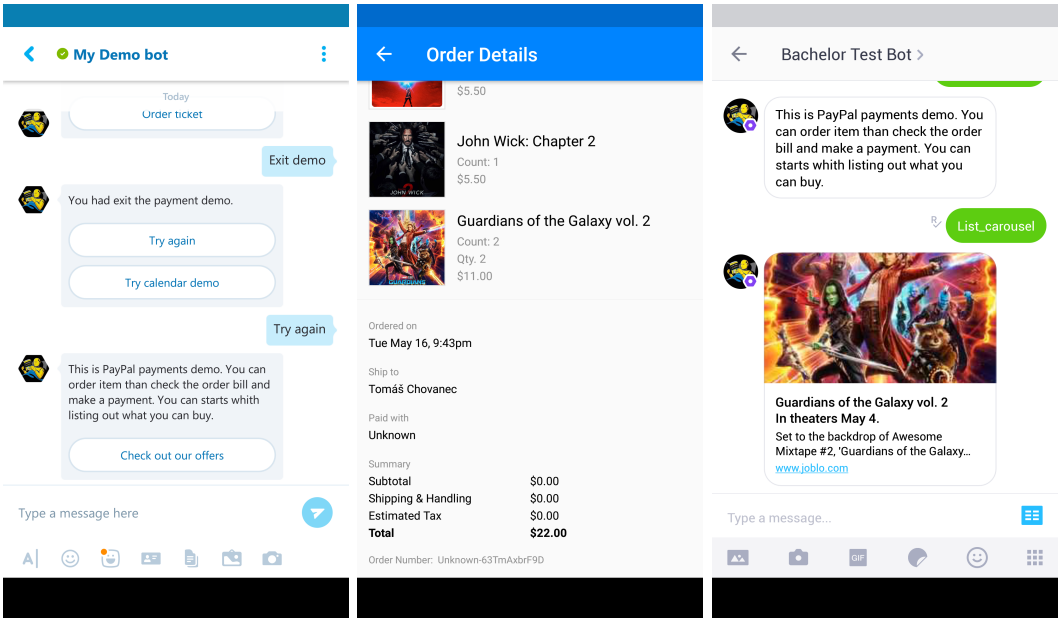
Testovanie v koncových aplikáciách alebo iba v rámci Bot Connectora je podmienené zadaním IP adresy alebo url na ktorom je bot dostupný. Preto, ak je zámerom túto funkcionalitu otestovať ešte pred nasadením na web je nutné použiť program *Ngrok* [3], ktorý umožňuje vystaviť do verejnej siete server bežiaci na lokálnej sieti za firewallom pomocou zabezpečeného tunelovania.

Záverečné testovanie bota prebiehalo pomocou nástrojov *Ngrok* a *Bot Connector* pripojeného ku kanálom aplikácií *Skype*, *Facebook Messenger* a *Kik Messenger*, ktorých náhľady sú dostupné na obrázku 7.1. Bot tak vystupoval pod verejným účtom schopný konverzovať s popredu vybranými osobami z oblasti IT ale aj laickou verejnosťou. Ich úlohou bolo vyskúšať alebo odborne otestovať funkcionalitu a následne odoslať spätnú väzbu.

Na jej základe boli vyhodnotené výsledky testovania, ktoré boli uspokojujúce s nízkou chybovosťou. Tento záver bol dosiahnutý vďaka priebežnému testovaniu počas vývoja

[1] https://docs.microsoft.com/en-us/bot-framework/debug-bots-emulator
[2] Internet Information Services Express je zjednodušená verzia IIS (https://www.iis.net) pre vývojárov.
[3] https://ngrok.com

a dodržiavaním obecných programovacích a aj testovacích zásad. Vyskytli sa však aj chyby a nedostatky spôsobené rozdielnosťou konverzačných platforiem, na ktorých bolo testovanie uskutočnené. Tieto chyby boli následne odladené.



Obr. 7.1: Náhľad na modelovú konverzáciu používateľa s vytvoreným chat botom v mobilných aplikáciách *Skype*, *Facebook Messenger* a *Kik Messenger*.

Kapitola 8

Záver

Práca mala za cieľ priniesť prehľad o aktuálnej situácii v oblasti chat botov a možností prepojenia na rozhrania tretích strán s ohľadom na systém rezervácie vstupeniek. Tento cieľ bol z veľkej časti naplnený najmä po teoretickej časti práce, kde je popísaná široká škála technológií a vývojových nástrojov pre tvorbu chat botov. Z tejto skupiny vyčnieva najmä *Microsoft Bot Framework* popísaný v kapitole 2.2, ktorý bol v procese rozhodovania použitý ako najsprávnejšia platforma pre tvorbu výstupného riešenia.

Po teoretickej stránke boli spracované aj vývojové nástroje inteligentných asistenčných služieb, platobných brán, kalendárnych systémov a služieb umožňujúcich kognitívne rozpoznávanie. Práca sa zamerala aj na rozbor procesu rezervácie vstupeniek, ktorý bol klasifikovaný ako istá forma objednávky a systém, v rámci ktorého tento proces existuje ako webová služba s prvkami internetového obchodu. Táto skutočnosť bola zohľadnená pri návrhu výstupného riešenia popísaného v kapitole 5.

Po implementačnej stránke je výstupom aplikácia chat bota, ktorý je schopný vykonať ukážkovú transakciu v platobnej službe PayPal a manažovať udalosti v testovacej agende služby Google Calendar. Kvôli nedostupnosti potrebných vývojových nástrojov nebolo možné prepojiť bota s inteligentnou asistenčnou službou.

Pri výbere platforiem záviselo najmä na rozšírení služby medzi používateľmi, kompatibilite vývojových nástrojov ale aj vhodnosti použitia pre danú problematiku. Tieto kritériá a dôvody výberu jednotlivých služieb boli detailnejšie rozobrané v kapitole 5.1. V konečnom dôsledku však kritériami prešlo iba niekoľko nástrojov a aj tie mali určité obmedzenia, čo možno prisúdiť relatívne krátkej existencii chat botov ako vývojových platforiem.

Návrh architektúry bol kľúčovou časťou práce, kde sa získané poznatky pretavili do vytvorenia *abstraktného prototypu*, prihliadajúc na všetky dostupné funkcie a obmedzenia vybraných platforiem. V implementácii popísanej kapitolou 6, bolo potrebné konkretizovať abstraktné triedy a iné entity architektonického návrhu, prípadne vykonať zmeny a optimalizáciu častí, ktoré pri neboli zohľadnené ako napríklad redundancia kódu. Následne tak vzniklo výsledné riešenie.

Ďalšie chat bot platformy, ktorých vývojové nástroje nevyhovovali zvoleným kritériám, bolo možné napojiť na Bot Framework aspoň vo forme jednotlivých kanálov v Bot Connectore. Keďže zadaním práce bolo hľadať čo najuniverzálnejšie riešenie museli byť jednotlivé kanály naležíte otestované čo popisuje kapitola 7.

V ďalšej fáze budú nadobudnuté znalosti a vytvorené moduly použité ako rozšírenie pre aplikáciu chat bota vyvíjaného firmou RIGANTI za účelom doplnenia existujúceho systému rezervácie vstupeniek. Z pohľadu práce bude najzaujímavejšie prepojenie vytvorených komponentov s existujúcim rezervačným botom, ktorý využíva nástroje na rozpoznávanie

jazyka. Na základe toho bol potrebný rozbor týchto nástrojov, ktorý sa vykonal v rámci kapitoly 3.4. Získané poznatky by mali uľahčiť implementáciu vytvorených modulov.

Tak ako bolo predostreté v úvode práce, vývojové nástroje okolo chat botov sa budú v nadchádzajúcej dobe naďalej meniť. Vhodným príkladom môžu byť inovácie oznámené na akcii *Build 2017*[1], ktorá sa konala len niekoľko dní pred termínom odovzdania. Boli predstavené vylepšenia pre *Microsoft Bot Framework*, či sprístupnené prepojenie chat botov a vývojových nástrojov inteligentnej asistentky *Cortana*.

[1]Konferencia Build je každoročne organizovaná udalosť, na ktorej Microsoft predstavuje novinky pre vývojárov. (https://build.microsoft.com)

Literatúra

[1] ALBAHARI, J.; ALBAHARI, B.: *C# 6.0 in a Nutshell: The Definitive Reference*. O'Reilly Media, Inc., 6 vydání, 2015, ISBN 1491927062, 9781491927069.

[2] Amazon: *Alexa Skills Kit Documentation*. [Online; navštívené 6.3 2017]. URL https://developer.amazon.com/alexa-skills-kit

[3] API.AI: *API.AI's developer documentation*. [Online; navštívené 31.3 2017]. URL https://docs.api.ai/docs

[4] Apple: *Calendar and Reminders Programming Guide*. [Online; navštívené 15.3 2017]. URL https://developer.apple.com/library/content/documentation/DataManagement/Conceptual/EventKitProgGuide/Introduction/Introduction.html

[5] Apple: *SiriKit Programming Guide*. [Online; navštívené 21.2 2017]. URL https://developer.apple.com/library/content/documentation/Intents/Conceptual/SiriIntegrationGuide/index.html#//apple_ref/doc/uid/TP40016875-CH11-SW1

[6] Card, M.: *Masterpass Merchant Integration*. [Online; navštívené 4.4 2017]. URL https://developer.mastercard.com/documentation/masterpass-merchant-integration#api-integration

[7] Facebook: *Facebook Graph API Documentation*. [Online; navštívené 15.3 2017]. URL https://developers.facebook.com/docs/graph-api/overview

[8] Google: *Calendar API Documentation*. [Online; navštívené 12.3 2017]. URL https://developers.google.com/google-apps/calendar/

[9] Google: *Google Actions Developer Guide*. [Online; navštívené 1.3 2017]. URL https://developers.google.com/actions/develop/conversation

[10] Google: *Messenger Platform Documentation*. [Online; navštívené 6.1 2017]. URL https://developers.facebook.com/docs/messenger-platform

[11] GroupMe: *GroupMe Public API Documentation*. [Online; navštívené 23.1 2017]. URL https://dev.groupme.com/docs/v3

[12] JOHNSON-LAIRD, P. N.: *The Computer and the Mind-An Introduction to Cognitive Science*. Harvard University Press, 1988, ISBN 94705-1310, 444 s.

[13] Kik: *Kik Developer Documentation*. [Online; navštívené 22.1 2017]. URL https://dev.kik.com/#/home

[14] Line: *Line Developers Documentation*. [Online; navštívené 29.1 2017].
URL https://developers.line.me

[15] MAULDIN, M. L.: ChatterBots, TinyMuds, and the Turing Test: Entering the Loebner Prize Competition. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (Vol. 1)*, AAAI '94, Menlo Park, CA, USA: American Association for Artificial Intelligence, 1994, ISBN 0-262-61102-3, s. 16–21.

[16] Microsoft: *Bot Framework Documentation*. [Online; navštívené 16.12 2016].
URL https://docs.microsoft.com/en-us/bot-framework/

[17] Microsoft: *Cortana Skills Kit Documentation*. [Online; navštívené 20.2 2017].
URL https://docs.microsoft.com/en-us/cortana/getstarted

[18] Microsoft: *Language Understanding Intelligent Service Documentation*. [Online; navštívené 30.3 2017].
URL https://docs.microsoft.com/en-us/azure/cognitive-services/LUIS/Home

[19] Microsoft: *Outlook Calendar REST API reference*. [Online; navštívené 14.3 2017].
URL https://msdn.microsoft.com/en-us/office/office365/api/calendar-rest-operations

[20] PayPal: *PayPal Developer Documentation*. [Online; navštívené 21.3 2017].
URL https://developer.paypal.com/docs/directory/

[21] Slack: *Slack API Documentation*. [Online; navštívené 17.1 2017].
URL https://api.slack.com

[22] Telegram: *Telegram API Documentation*. [Online; navštívené 17.1 2017].
URL https://core.telegram.org

[23] TURING, A. M.: I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, ročník LIX, č. 236, 1950: str. 433, doi:10.1093/mind/LIX.236.433.

[24] Twillio: *Twillio Documentation*. [Online; navštívené 13.2 2017].
URL https://www.twilio.com/docs/

[25] VERNON, D.: *Artificial Cognitive Systems: A Primer*. The MIT Press, první vydání, October 17, 2014, ISBN 0262028387, 978-0262028387.

[26] WeChat: *WeChat Open Platform Documentation*. [Online; navštívené 30.1 2017].
URL
http://open.wechat.com/cgi-bin/newreadtemplate?t=overseas_open/index

[27] WEIZENBAUM, J.: ELIZA&Mdash;a Computer Program for the Study of Natural Language Communication Between Man and Machine. *Commun. ACM*, ročník 9, č. 1, Leden 1966: s. 36–45, ISSN 0001-0782, doi:10.1145/365153.365168.

Prílohy

Príloha A

Obsah CD

V tejto prílohe je zobrazená štruktúra adresára CD, ktoré je pribalené k originálu bakalárskej práce. Obsahuje elektronickú verziu práce vo formáte PDF, originál vo forme zdrojových súborov LaTeX a adresár obsahujúci projekt Visual Studia so zdrojovými súbormi potrebnými pre spustenie aplikácie. Štruktúra z pohľadu koreňového adresára disku vyzerá nasledovne:

- latex_source\ - zdrojové súbory a obrázky potrebné k vytvoreniu dokumentu bakalárskej práce;
- bot_aplication\ - projekt Visual Studia obsahujúci zdrojové kódy, knižnice a iné súbory vytvorené počas implementácie;
- bakalarska_praca.pdf - dokument vo formáte PDF obsahujúci bakalársku prácu rovnakej verzie aká bola odovzdaná v informačnom systéme.