Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

# BAKALÁŘSKÁ PRÁCE

Chatbot pro FAQ

2020

Jan Nejezchleba

Vedoucí práce: Mgr. Jiří Zacpal, Ph.D.

Studijní obor: Aplikovaná informatika, prezenční forma

**Bibliografické údaje**

| | |
|---|---|
| Autor: | Jan Nejezchleba |
| Název práce: | Chatbot pro FAQ |
| Typ práce: | bakalářská práce |
| Pracoviště: | Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci |
| Rok obhajoby: | 2020 |
| Studijní obor: | Aplikovaná informatika, prezenční forma |
| Vedoucí práce: | Mgr. Jiří Zacpal, Ph.D. |
| Počet stran: | 57 |
| Přílohy: | 1 CD/DVD |
| Jazyk práce: | český |

**Bibliographic info**

| | |
|---|---|
| Author: | Jan Nejezchleba |
| Title: | Chatbot for FAQ |
| Thesis type: | bachelor thesis |
| Department: | Department of Computer Science, Faculty of Science, Palacký University Olomouc |
| Year of defense: | 2020 |
| Study field: | Applied Computer Science, full-time form |
| Supervisor: | Mgr. Jiří Zacpal, Ph.D. |
| Page count: | 57 |
| Supplements: | 1 CD/DVD |
| Thesis language: | Czech |

## Anotace

*Hlavním cílem této práce je tvorba chatbota na platformě Facebook Messenger komunikujícího se zájemci o studium na Katedře informatiky. Obsah práce se skládá z průzkumu existujících řešení, technické specifikace samotné implementace a její uživatelské dokumentace. Tato práce obsahuje také popis webového rozhraní pro správu chatbota doplněnou o vlastní technickou i uživatelskou dokumentaci.*

## Synopsis

*The main goal of this thesis is the creation of chatbot on the Facebook Messenger platform, which will communicate with those interested in studying at the Department of Computer Science. Contents of this thesis include survey of existing solutions, technical specifications of the implementation and its user documentation. The thesis also contains description of the web interface used to maintain the chatbot complete with its own technical and user documentation.*



**Klíčová slova:** chatbot; Node.js; Facebook Messenger; FAQ; webové rozhraní; Express.js; SQLite; Wit.ai

**Keywords:** chatbot; Node.js; Facebook Messenger; FAQ; web interface, Express.js; SQLite; Wit.ai

Chtěl bych poděkovat Mgr. Jiřímu Zacpalovi, Ph.D. za odborné vedení mé bakalářské práce. Dále mé poděkováni patří RNDr. Martinu Trnečkovi, Ph.D. za ochotu poskytnout informace z databáze katedry pro účel mé práce. V neposlední řadě děkuji i všem, kteří mě jakkoli podpořili při psaní této bakalářské práce.

*Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.*

datum odevzdání práce podpis autora

# Obsah

# Seznam obrázků



# Seznam tabulek

# 1 Úvod

Tvorba chatbota schopného konverzace se zájemci o studium byla představena vedoucím této bakalářské práce Mgr. Jiřím Zacpalem, Ph.D. jako způsob popularizace studia na Katedře informatiky, Univerzity Palackého.

Cílem chatbot aplikace je poskytovat uživatelům informace o různých často dotazovaných aspektech studia informatiky. To zahrnuje mimo jiné představení univerzity, fakulty a jejich specifik, se kterými se může student během svého studia setkat. Hlavním zaměřením ale zůstává katedra, která skrze tuto aplikaci může představit svou činnost, personál, obory, způsob přijímacího řízení a další informace, které přispějí k popularizaci a objasnění specifik studia informatiky.

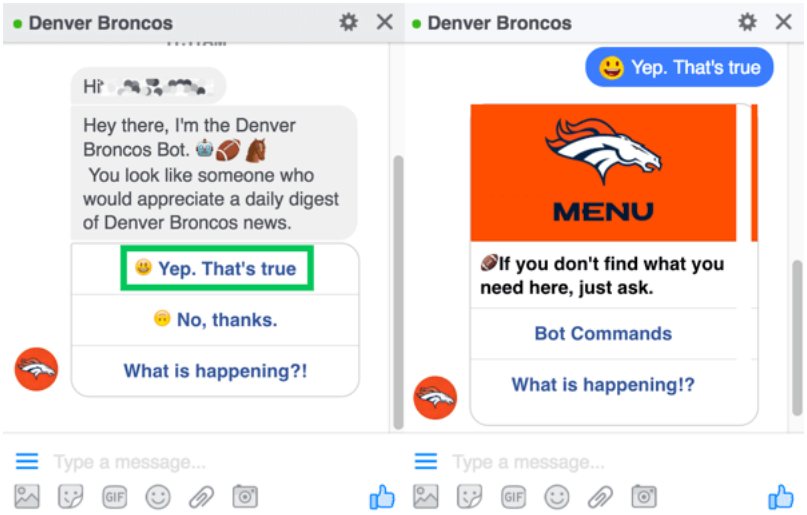
Zároveň je v této práci řešena i potřeba aktualizace vědomostní banky vyplývající z takto koncipované aplikace, a to konkrétně implementací webového rozhraní vyvíjeného spolu s chatbot aplikací, která ve výsledku umožní jednoduchou správu struktury a obsahu zpráv prezentovaných uživatelům.

# 2 Průzkum existujících řešení

Chatboti se v současnosti vyskytují na různorodých pozicích s mnohdy velmi odlišnými úkoly, pro které jsou vybaveni specifickou funkčností. Na kategorizaci řešení tvorby chatbotů lze nahlížet z mnoha úhlů. Pro tuto práci je zvolen pohled na to jakým způsobem jsou schopni vést konverzaci s koncovým uživatelem:

- Chatboti s předdefinovaným průběhem konverzace
- Chatboti využívající algoritmy zpracování přirozeného jazyka
- Chatboti rozpoznávající kontext
- Hlasoví chatboti

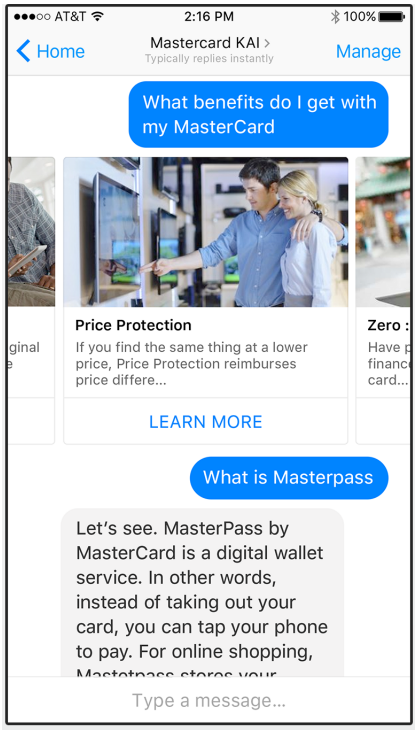
## 2.1 Chatboti s předdefinovaným průběhem konverzace




Obrázek 1: Chatbot pro Facebook stránku týmu Denver Broncos

Jedná se o nejjednodušší druh chatbotů, kteří uživatele vedou skrze konverzaci ohraničenou předem vytyčenými otázkami. Uživatel může většinou zvolit z několika možností skrze poskytnuté menu s odpověďmi, které posune konverzaci dále. Dalším lehce odlišným způsobem je využití příkazů od uživatele, které na základě vnitřních pravidel chatbota vyvolají určitou akci či odpověď.

## 2.2 Chatboti využívající algoritmy zpracování přirozeného jazyka




Obrázek 2: Chatbot firmy Mastercard obsahující NLP

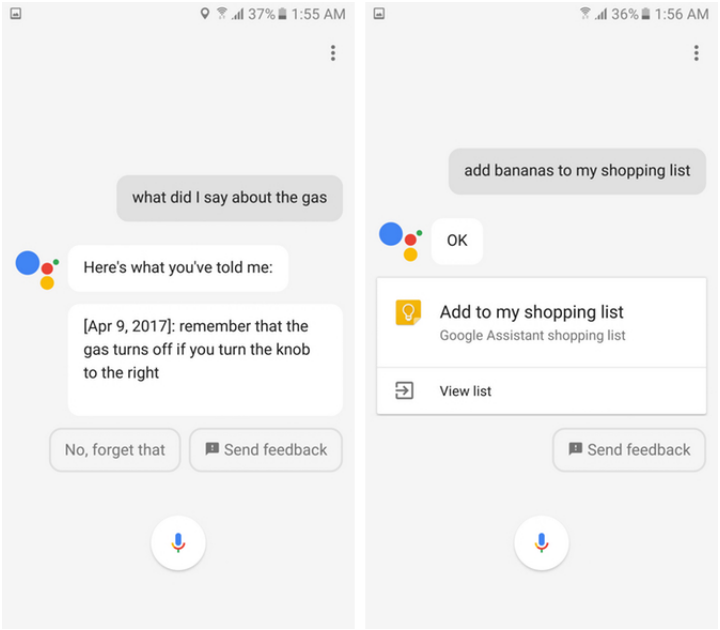
Tento druh chatbotů v sobě aplikuje algoritmy zpracování přirozeného jazyka, zkráceně NLP. Od předchozího druhu se liší svou schopností z textového vstupu uživatele vyvodit jeho záměr a adekvátně na něj reagovat. Uživatel tudíž není v konverzaci veden po předem dané cestě. Samotná interakce probíhá díky tomu pro člověka více přirozeně.

## 2.3 Chatboti rozpoznávající kontext

Tito chatboti v sobě využívají strojové učení a rozpoznávání konverzačního kontextu za účelem zjednodušení interakce s uživatelem. Jejich hlavní výsadou je schopnost pamatovat si předešlé konverzace s uživatelem, ze kterých se mohou učit a lépe se přizpůsobit potřebám uživatele. Zahrnují v sobě kombinaci i předem zmíněných způsobů komunikace s uživatelem, což je činí nejkomplikovanějšími druhy chatbotům. Řadí se mezi ně dnes populární Siri, Alexa nebo Google Assistant.

## 2.4 Hlasoví chatboti

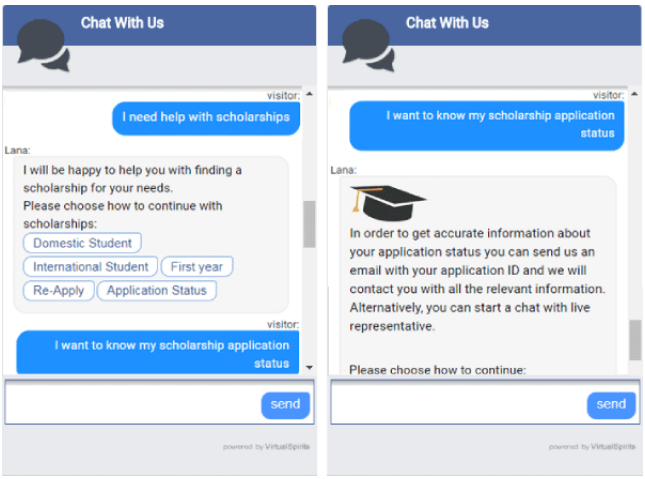
Tato skupina může být vnímána spíše jako podskupina kontextových chatbotů. Stále totiž obsahují vše z jejich komplexnosti, jen k ní navíc přidávají možnost




Obrázek 3: Příklad konverzace s Google Assistentem

přijímání hlasového vstupu od uživatele. Příklady chatbotů uvedené v předchozí skupině v sobě všechny obsahují schopnost rozpoznání hlasového vstupu i generování hlasové odpovědi.

# 3 Konkrétní případ řešení




Obrázek 4: University chatbot od firmy VirtualSpirits

Na obrázku 4 je uvedeno jedno z možných řešení chatbota pro studenty a potenciální zájemce o studium na univerzitě.

Lze vidět, že autoři využili kombinaci dvou způsobů interakce s uživatelem. Chatbot zřejmě využívá NLP algoritmus, díky kterému je schopen generovat odpovědi na základě analýzy textu. Zároveň je však uživateli poskytnuta možnost výběru z předem určených oblastí zájmu.

Výhoda takovéhoto přístupu je ve větší přímočarosti a schopnosti rychleji dosáhnout žádané odpovědi, kterou lze ještě více specifikovat textovým vstupem.

Další pozitivum vyplývá z možného zrychlení nalezení odpovědi samotným chatbotem. Ve struktuře dat s odpověďmi se totiž ve výsledku můžeme pohybovat jen na vybraném úseku týkajícího se právě zvoleného předmětu konverzace.

Nevýhoda daného řešení se může ukázat v případě, kdy jsou oblasti zájmu zvoleny příliš obecně či zmatečně, což může vést k odrazení uživatele od jejich použití.

# 4 Použité technologie

## 4.1 JavaScript

### 4.1.1 Základní informace

JavaScript je multiplatformní programovací jazyk, který v sobě snoubí objektové, imperativní a událostmi řízené paradigma. Byl poprvé představen v roce 1995 Brendanem Eichem.[1]

### 4.1.2 Vlastnosti JavaScriptu

JavaScript mnoho ze své syntaktické programové struktury jako if/else větvení nebo cykly přejímá z jazyka C.

Je dynamicky typovaný, disponuje prototypováním, skrze které implementuje tzv. prototypovou dědičnost na rozdíl od typické dědičnosti s třídami. V závislosti na tom není striktní rozdíl mezi metodou a funkcí, rozdíl se projeví až při jejich zavolání, kde funkce zavolána jako metoda obsahuje lokální *this* objekt obsahující odkaz na objekt, který ji vyvolal.

Specifikou JavaScriptu býval funkční rozsah platnosti proměnných, který zajišťoval typ *var*. Dnes jsou součástí jazyka i typy *const* a *let* dodávající blokový rozsah platnosti.

Zásadním prvkem především v serverových implementacích JavaScriptu je použití příslibů[1], které slouží k obsluze asynchronních operací skrze tzv. *Promise* objekty. [1]

### 4.1.3 Využití JavaScriptu

Využití Javascriptu se soustředí na webové aplikace a webové stránky, kterým dodává interaktivitu. U webových stránek se o interpretaci kódu stará přímo

[1]Příslib reprezentuje eventuální dokončení (nebo selhání) asynchronní operace a její návratovou hodnotu.

prohlížeč. U serverových aplikací pracuje JavaScript skrze událostmi řízenou architekturu interpretovanou na straně serveru s využitím běhového prostředí např. Node.js. To ho činí užitečným u aplikací pracujících v reálném čase např. chatbotů.[1]

### 4.1.4 ECMAScript

U JavaScriptu stojí za to zmínit ECMAScript. Jedná se o specifikaci skriptovacího jazyka standardizovanou organizací Ecma International, která byla vytvořena právě pro standardizaci jazyka JavaScript. Zpravidla je standard využíván pro implementaci dynamických webových stránek pro službu World Wide Web. V dnešní době jsou v něm ale stále častěji tvořeny i serverové aplikace a služby s pomocí Node.js.[2]

## 4.2 Node.js

### 4.2.1 Základní informace

Node.js byl vyvinut Ryanem Dahlem v roce 2009 a dnes je podporován na všech nejvyužívanějších operačních systémech. Jedná se o běhové prostředí umožňující spuštění JavaScript kódu mimo prohlížeč. Aplikace využívající Node.js ale nemusí být nutně napsány přímo v jazyce JavaScript, lze použít i jazyky s možnou kompilací do JavaScriptu jako CoffeeScript, Dart nebo TypeScript. [3]

### 4.2.2 Využití Node.js

Využívá se zejména pro svoji škálovatelnost, a to hlavně u webových serverů a jiných internetových aplikací. Používá model událostí a asynchronní I/O operace, které zaručují menší režii procesoru a v důsledku lepší výkon. [4]

### 4.2.3 Správce balíčků

Specifický pro Node.js je taktéž **npm** (Node.js package manager). Jedná se o výchozího správce balíčků. Skládá se z klienta ve formě příkazové řádky a on-line databáze veřejných/bezplatně publikovaných a placených soukromých balíčků, nazvaný npm registry. [5]

## 4.3 Facebook Messenger

Messenger od firmy Facebook je zvolená platforma, na které poběží vytvořený chatbot. Stará se o správu komunikace mezi klientem a chatbotem.

Pokaždé když uživatel zašle zprávu na Facebookovou stránku, která z části nebo zcela implementuje aplikaci pro automatizaci konverzace, stane se následující. Facebook server zašle webhook[2] na URL adresu serveru, na kterém běží

[2]Webhook je uživatelsky definované HTTP volání, které na základě zvolených událostí na webu provede vývojářsky definovanou akci např. po kliknutí na tlačítko se odešle email.

konverzační aplikace. Pomocí Send API[3] může aplikace následně reagovat na zprávu od uživatele na Messengeru. Tímto způsobem je vývojářům umožněna tvorba strukturovaných konverzací s uživateli.

Pro využití platformy je nutné zaregistrovat aplikaci u Facebooku a vyplnit všechny náležitosti, po kterých bude hotová aplikace podrobena evaluaci ze strany Facebooku a v případě splnění všech požadavků jí bude povolen přístup k Send API. [6]

## 4.4 Wit.ai

Bezplatná aplikace Wit.ai pod záštitou firmy Facebook poskytuje aplikační rozhraní pro rozpoznávání přirozeného jazyka a jeho přeměnu na strukturovaná data.

Pro její použití je nutné založení nové aplikace na oficiálních stránkách. Tyto stránky fungují zároveň jako webové rozhraní pro správu vytvořených aplikací, avšak zároveň nabízí i možnost vzdálené správy pomocí HTTP API. [7]

Pokud vývojář chce svou Wit.ai aplikaci propojit s aplikací využívající platformu Messenger stačí do nastavení aplikace na stránkách Facebooku vložit klíč vygenerovaný v rámci Wit.ai aplikace. Messenger bude následkem toho jako součást zpráv zasílat i data vygenerovaná Wit.ai aplikací.

Mezi data, která Wit.ai poskytuje, se řadí informace o rozpoznaném jazyce a informace o rozpoznané uživatelsky definované entitě. Například pokud uživatel zašle zprávu *"Ahoj."* může zpráva obsahovat informace, o tom že byla rozpoznána vývojářem vytvořená entita *pozdrav* v českém jazyce s číselnou mírou jistoty rozpoznání.

## 4.5 SQLite

### 4.5.1 Základní informace

SQLite projekt započal v roce 2000 a je vyvíjen D. Richardem Hippem. Jedná se o relační databázový systém, obsažený v relativně malé knihovně napsané v jazyce C. Systém je především specifický tím, že nevyužívá architekturu klient-server. Samotné databáze místo toho uchovává jako soubory, ke kterým se přistupuje pomocí jednoduchého rozhraní. [8]

### 4.5.2 Vlastnosti SQLite

Mezi jeho vlastnosti se řadí přenositelnost a přímočará nenáročná správa systému, oproti nutnosti konfigurace a správy samostatného serveru. Podporuje transakční přístup k databázím a tzv. PRAGMA příkazy, které slouží pro základní konfiguraci databáze.

[3] Jedná se o hlavní aplikační rozhraní pro zasílání zpráv a dalších událostí uživatelům. Během vývoje je tato API zpřístupněna pouze Facebook účtům vývojářů a testerů.

S absencí serveru je ale zároveň ztracena možnost autorizace v rámci databáze, která lze částečně nahradit nastavením práv v rámci souborového systému.

Dále, jelikož jsou databáze pouhé soubory, je simultánní zápis více vláken procesu najednou problematičtější než u klient-server databází. Omezení spočívá v tom, že při zápisu se soubor zamkne a vlákna musí čekat na jeho odemčení, což při hodně zápisově náročných použitích může značně snižovat efektivitu databáze. SQLite ale nabízí možnosti, jak případné dopady zmírnit pomocí povolení tzv. WAL[4] (Write Ahead Log) módu. [8]

### 4.5.3 Využití SQLite

Jedná se o jednu z nejoblíbenějších verzí tzv. embedded databáze, která je úzce provázána s aplikací, se kterou spolupracuje. Funguje na všech operačních systémech a je díky své univerzálnosti součástí např. mnoha populárních webových prohlížečů.

## 4.6 Express.js

### 4.6.1 Základní informace

Jedná se o populární webový aplikační rámec pro běhové prostředí Node.js. Autoři ho prezentují jako rychlý a minimalistický. Byl představen roku 2010 a od té doby se stal de facto standardem pro tvorbu webových aplikací v prostředí Node.js. [10]

### 4.6.2 Vlastnosti Express.js

Umožňuje tvorbu obsluhy žádostí na základně HTTP metod pro různé URL adresy skrze objekt nazvaný *router* neboli směrovač.

Podporuje integraci s tzv. vykreslovacími jádry, které dokáží generovat webové stránky s pomocí dat vložených do šablon HTML dokumentů.

Implementuje podporu tzv. middlewaru, který se vkládá do procesu obsluhy požadavků a přidává další funkčnost jako například správu přihlašování nebo komunikaci s chatbotem.

Jelikož je Express.js sám o sobě minimalistický, jsou to právě balíčky výše zmíněných middleware funkcí, které do něj přidávají další nástroje pro řešení téměř všech problému naskýtajících se při vývoji webový stránek a aplikací. [11]

[4] WAL mód invertuje použití klasického žurnálu, a to tak, že změny jsou postupně zapisovány do separátním souboru a jsou do databáze transakčně vloženy po naplnění daného souboru nad určitou mez. To umožňuje rychlejší současné čtení a zápis, ale databáze je v důsledku zranitelnější vůči selhání systému. [9]

## 4.7 Pug

### 4.7.1 Základní informace

Pug (dříve Jade) implementuje šablonovací jádro využívané v kombinaci s Express web serverem. Ve zkratce tato technologie slouží k vytváření šablon webových stránek, které server zpracuje a zašle uživateli. [12]

### 4.7.2 Vlastnosti Pug

Specifikem Pug šablon je, že nemají podobu typického HTML souboru. Využívá se zde ve velké míře odsazení jako hlavní způsob určení hierarchie kódu. Samotný kód je pak kombinací nepárově zapisovaných HTML značek a javascriptovým kódem se syntaxí specifickou právě pro Pug. [13]

Příkladem zmíněného specifického kódu mohou být cykly, či ternární operátory, které na základě argumentů předaných šabloně, například dat ze serverové databáze, mění výslednou podobu HTML stránky.

Tyto předem vytvořené šablony mohou být navíc do sebe vzájemně vkládány. Server jejich finální podobu před odesláním vždy zpracuje a převede na výsledný HTML soubor zobrazený uživateli.

Šablony ve výsledku tudíž umožňují zkrácení HTML kódu a snížení jeho duplicity v rámci jednotlivých stránek spolu s možností dynamického způsobu jejich tvorby na základě předaných argumentů.

## 4.8 Ngrok

Nástroj pro zviditelnění portů lokálně běžícího webového serveru v rámci internetu, díky náhodně vygenerované webové adrese na doméně ngrok.io.

Umožňuje vytvoření i více souběžných HTTP/HTTPS spojení pro aplikace naslouchajících na různých portech. Poskytuje zároveň možnost monitorování HTTP požadavků a celkovou diagnostiku proběhlé komunikace na zadaném portu.

Využívá se zejména během vývoje pro testování zahrnující komunikaci s webhooky na lokálních serverech, které tímto způsobem mohou komunikovat se vzdálenými aplikačními rozhraními přes internet.

## 4.9 REST

### 4.9.1 Základní informace

REST (Representational State Transfer) je architektura rozhraní, navržená pro distribuované prostředí neboli pro programy, jejichž části běží na různých strojích a ke komunikaci využívají počítačovou síť. [14]

### 4.9.2 Popis architektury REST

Vymezení, zda daný systém splňuje podmínky REST architektury, je dáno dodržením těchto podmínek:

- **Klient-server architektura** - Zajistí oddělení uživatelského rozhraní od úkonů spojených s uchováváním dat, což umožňuje nezávislý vývoj těchto komponentů.
- **Bezstavovost** - Komunikace mezi klientem a serverem není určena žádným kontextem uloženým na serveru během zasílání požadavků. Každá žádost od klienta v sobě drží všechny potřebné informace k její obsluze a stav sezení je držen v klientovi.
- **Využití mezipaměti** - Klienti mohou ukládat odpovědi do mezipaměti. Odpovědi musí explicitně nebo implicitně definovat, zda je lze nebo nelze ukládat do mezipaměti.
- **Vrstvený systém** - Klient nemůže obyčejně poznat, zda je připojen přímo ke koncovému serveru nebo prostředníkovi.
- **Kód na vyžádání** - Servery mohou dočasně rozšiřovat či upravovat funkcionalitu klienta zasláním spustitelného kódu např. ve formě klientského JavaScriptu.
- **Uniformní rozhraní** - Je fundamentálním prvek pro nezávislý vývoj jednotlivých částí REST architektury. Je určen následujícími prvky:
  - Stav aplikace a chování je vyjádřen takzvaným resourcem neboli zdrojem (klíčová abstrakce), každý zdroj musí mít unikátní identifikátor (URL, URN).
  - HATEOAS (Hypermedia as the Engine of Application State, v překladu Hypermedia jako aplikační stav) – stav aplikace je určen pomocí URL. Další možné stavy můžeme získat pomocí odkazů, které klient dostane v odpovědi od serveru.
  - Je definován jednotný přístup pro získání a manipulaci se zdrojem v podobě čtyř operací CRUD (Create, Read, Update, Delete).
  - Zdroj může mít různé reprezentace (XML, HTML, JSON, SVG, PDF), klient nepracuje přímo se zdrojem, ale s jeho reprezentací.

[14]

### 4.9.3 Popis využívaných HTTP metod

REST využívá HTTP metod k vytvoření, čtení, editaci nebo smazání dat:

- GET - Je základní metodou pro přístup k datům.

- POST - Je základní metodou pro tvorbu dat.
- PUT - Je základní metodou pro přepsání existujících dat. Pokud nejsou data nalezena, jsou vytvořena.
- PATCH - Je základní metodou pro aktualizaci existujících dat.
- DELETE - Je základní metodou pro odstranění dat.

# 5 Specifikace řešení

## 5.1 Chatbot pro FAQ

Samotný bot bude umět vést konverzaci s uživatelem s pomocí rozpoznávání přirozeného jazyka poskytnutého asociovanou Wit.ai aplikací.

Pro snadnější komunikaci bude přítomen i seznam přichystaných témat konverzace, kterých bude moci uživatel využít tím, že klikne na některou zobrazenou položku.

Aplikace se bude skládat ze tří základních částí:

- **Chatbot** - Část aplikace starající se o přijetí, zpracování a vygenerování zpráv v reakci na vstup uživatele.
- **Databáze** - Část aplikace, kde budou uloženy informace, potřebné pro tvoření odpovědí a správné vyhodnocení kontextu konverzace s jednotlivými uživateli.
- **Webové rozhraní** - Část aplikace zodpovědná za editaci informací obsažených v databázích, úprav Wit.ai aplikace a řízení chodu chatbota.

## 5.2 Implementace chatbota

Základem práce je tvorba aplikace fungující jako chatbot. V rámci tohoto problému existuje mnoho různých způsobů řešení od hotových balíčků funkcí, které jsou uživatelsky nenáročné a stačí pouze naplnit možné konverzační stromy daty, až po vývojářské nástroje pro tvorbu chatbota od úplných základů. Tato práce se zabývá druhou uvedenou možností, kdy je jádro chatbota vystavěno od základů s pomocí různých externích modulů.

Pro základ implementace je zvolena kombinace prostředí Node.js s jazykem JavaScript, což umožní dále i jednodušší tvorbu webového rozhraní pro správu chatbota, jelikož bude vycházet ze stejných technologií. Mimo jiné je dokumentace pro Facebook Messenger s tímto řešením zcela kompatibilní.

Facebook dále požaduje pro aplikace na platformě Messenger tvorbu webhooku, který s Facebook serverem komunikuje přes HTTPS protokol dle architektury REST.

Webhook bude implementován skrze Express.js web server, který bude obsahovat i samotnou logiku chatbota.

Skrze Facebook bude asociována i Wit.ai aplikace, jejíž data budou zasílána jako součást zprávy uživatele Messengeru a zpracována uvnitř chatbota.

Pro ostrý provoz bude chatbot umístěn na server s platným SSL certifikátem pro zajištění HTTPS spojení s Facebookem a během vývoje bude testován s pomocí nástroje ngrok.

## 5.3 Implementace databáze

Databáze bude sloužit chatbotovi jako vědomostní banka a zároveň i jako úschovna aktuálního kontextu konverzace pro jednotlivé uživatele.

K implementaci poslouží relační databáze SQLite, která se hodí pro aplikace menších měřítek jako je chatbot a poslouží k jeho jednodušší správě, která nebude závislá na konfiguraci databázového serveru. Navíc učiní celou aplikaci snadněji přenositelnou.

Pokud by z nějakého důvodu bylo během ostrého provozu zatížení databází příliš vysoké, je možné databáze přepnout do WAL módu, což by mělo optimalizovat jejich výkon.

Ke komunikaci s webovým rozhraním i chatbotem budou využity moduly pro Node.js implementující potřebné funkce pro databáze typu SQLite.

## 5.4 Implementace webového rozhraní

Webové rozhraní bude administrátorovi chatbota sloužit jako nástroj pro aktualizaci vědomostní databáze a správu chodu.

Pro jeho implementaci bude opět využit Express.js web server pro Node.js a jednotlivé stránky budou vytvořeny s pomocí Pug šablon.

Rozhraní by mělo běžet stejně jako chatbot na serveru s HTTPS a pouze naslouchat na odlišném portu než samotný chatbot. Bude obsahovat vlastní databázi uchovávající přihlašovací údaje uživatelů rozhraní a informace o aktivních sezeních na stránce.

Obecná funkcionalita rozhraní bude zahrnovat tyto body:

- Přihlášení
- Přidání/odebrání uživatelského účtu
- Správu přihlašovacích údajů
- Správu FAQ otázek
- Správu konverzačního stromu chatbota
- Možnost trénování chatbota pomocí komunikace s Wit.ai
- Ovládání chodu chatbota (vypnutí, zapnutí, restart)

# 6 Programátorská dokumentace

## 6.1 Použité externí komunitní moduly

Následující použité moduly jsou vedeny pod open-source licencí:

- **express** - Modul implementující aplikační rámec Express.js.
- **pug** - Modul dodávající do Node.js Pug šablonování.
- **dotenv** - Modul umožnující oddělení konstant specifických pro prostředí aplikace v samostatném souboru *.env*. Konstanty jsou při spuštění aplikace nahrány do objektu *process.env* odkud mohou být volány. Je využit např. při uchovávání přístupových klíčů do externích API.
- **bcrypt** - Modul zpřístupňující funkce pro hašování hesel pomocí hašovací funkce bcrypt[5].
- **body-parser** - Middlewarový modul pro Express.js využívaný pro analyzování těl příchozích HTTP požadavků a jejich zpřístupnění v *req.body* objektu.
- **helmet** - Middlewarový modul pro Express.js dodává ochranu proti častým zranitelnostem na webových stránkách, mimo jiné jsou zahrnuta opatření proti clickjackingu[6], odstranění X-Powered-By[7] hlavičky a přidání HSTS hlavičky[8]
- **express-session** - Middlewarový modul pro Express.js implementující funkčnost cookies spolu se správou sezení mezi serverem a klientem.
- **connect-sqlite3** - Modul umožnující ukládání informací o sezeních do SQLite databáze.

[5] Bcrypt je hašovací funkce pro odvození klíče navržená Nielsem Provosem a Davidem Mazièresem založena na šifře Blowfish

[6] Clickjacking je způsob útoku na uživatele webových stránek, kdy se po např. kliknutí spustí nečekaná akce.

[7] Hlavička obsahující informace o serveru.

[8] HSTS hlavička říká prohlížečům, že by se mělo na stránku přistupovat výhradně přes HTTPS.

- **express-validator** - Middlewarový modul pro Express.js implementující serverovou validaci POST požadavků.

```
1  //Přidání modulu
2  const { check, validationResult } = require('express-validator');
3
4  app.post('/user', [
5    //Uživatelské jméno musí být email
6    check('username').isEmail(),
7    //Heslo musí mít aspoň 5 znaků
8    check('password').isLength({ min: 5 })
9  ], (req, res) => {
10   //Nalezne validační errory v tomto požadavku a zabalí je do
       objektu
11   let errors = validationResult(req);
12   if (!errors.isEmpty()) {
13     return res.status(422).json({ errors: errors.array() });
14   }
15   User.create({
16     username: req.body.username,
17     password: req.body.password
18   }).then(user => res.json(user));
19 });
```

Zdrojový kód 1: Příklad využití modulu express-validator

- **sqlite3** - Jedná se o balíček funkcí umožnující komunikaci s SQLite databázemi z prostředí Node.js. Pro pohodlnější a přehlednější implementaci asynchronních volání s await/async syntaxí byly využívané funkce z modulu přeměněny na přísliby. Tyto pozměněné funkce se nachází v souboru *sqlite-async.js*.

```
1  //Přidání upraveného modulu
2  const sqlite3 = require('./sqlite-async.js');
3
4  //Funkce tvořící databázové spojení
5  function setupWorkerConn() {
6    return new sqlite3.Database('myDB.db', sqlite3.OPEN_READWRITE, (
       err) => {
7      if (err) {
8        throw new Error('Invalid database connection.')
9      }
10   });
11 }
12
13 async function exampleFun() {
14   let results = null;
15   let db = setupWorkerConn();
16   try {
17     //Vymazání z tabulky pomocí parametrického dotazu
18     await db.runAsync(`DELETE FROM ipsum WHERE lorem = ?`, ['param'];
19     //Vrácení prvního nalezeného prvku z dotazu
20     results = await db.getAsync(`SELECT lorem FROM ipsum`);
21     //Vrácení všech nalezených prvků z dotazu
22     results = await db.allAsync(`SELECT lorem FROM ipsum`);
23     return results;
24   } catch (err) {
25     return null;
26   } finally {
27     db.close();
28   }
29 }
```

Zdrojový kód 2: Příklad základního použití asynchronně upravené sqlite3

- **node-fetch** - Jedná se o malý modul umožnující snadnou implementaci manipulace se zdroji na serveru využitím HTTP metod.

```
1  //Přidání modulu
2  const fetch = require("node-fetch");
3
4  async function sendRequest(requestBody) {
5    //Nastavení spojení
6    let connection = {
7      url: new URL('https://graph.facebook.com/v6.0/me/messages?
           access_token=' + process.env.FACEBOOK_ACCESS_TOKEN),
8      headers: {'Content-Type': 'application/json'},
9    };
10
11   try {
12     //Odeslání požadavku
13     await fetch(connection.url, {
14       method: 'POST',
15       headers: connection.headers,
16       body: JSON.stringify(requestBody)
17     });
18     return true;
19   } catch(err) {
20     console.log("ERROR: Could not send request to Messenger." + err);
21     return false;
22   }
23 }
```

Zdrojový kód 3: Využití node-fetch k odeslání zprávy na Messenger

## 6.2 Pomocné databázové moduly

Moduly obsažené ve složce *database_util_modules* upravují určitým způsobem přístup k SQLite databázím, a to jak u chatbot aplikace, tak u webového rozhraní. Složka obsahuje následující soubory:

- **sqlite-async.js** - Upravuje vybrané výchozí funkce pomocného sqlite3 modulu pro jednodušší implementaci async/await syntaxe s využitím příslibů.
- **database-class.js** - Definuje třídu *Database*, která obsahuje upravené metody z *sqlite-async.js*. S její pomocí jsou vytvořeny databázové moduly chatbota i webové aplikace. Její využití zpřehledňuje a zkracuje syntaxi, jelikož třída při zavolání jedné z definovaných metod sama otevře i ukončí spojení s databází, která je zadána v jejím konstruktoru.
- **chatbot-db-updater.js** - Tento modul se stará o aktualizaci databáze *chatbot_answers* z webového rozhraní. Obsahuje množství definovaných

transakcí a pomocných funkcí, díky kterým je možná asynchronní manipulace s daty v databázi se zachováním konzistentního stavu při případné chybě.

- **resource-db-updater.js** - Tento modul se stará o aktualizaci databáze *chatbot_resources* z webového rozhraní. Analogicky jako výše zmíněný chatbot-db-updater využívá transakce k zajištění konzistence stavu během asynchronních volání.

## 6.3 Spuštění aplikace

Ke spuštění za účelem lokálního vývoje a testování obou aplikací (chatbota i webového rozhraní) slouží konzolový příkaz **node server**, který spustí v prostředí Node.js soubor *server.js*, v rámci jehož inicializace je spuštěna i aplikace chatbot. Tato varianta spuštění vyžaduje spolupráci s nástrojem ngrok pro následné vytvoření HTTPS spojení pomocí příkazů **ngrok http 3000** pro chatbota a případně **ngrok http 8080** pro rozhraní. Pro funkčnost chatbota je dále potřeba vložit vygenerovanou doménu na portu 3000 podporující HTTPS do nastavení Messengeru např. **https://ff319648.ngrok.io/webhook** a ověřit ji. Tímto získáme provizorní spojení s Messengerem.

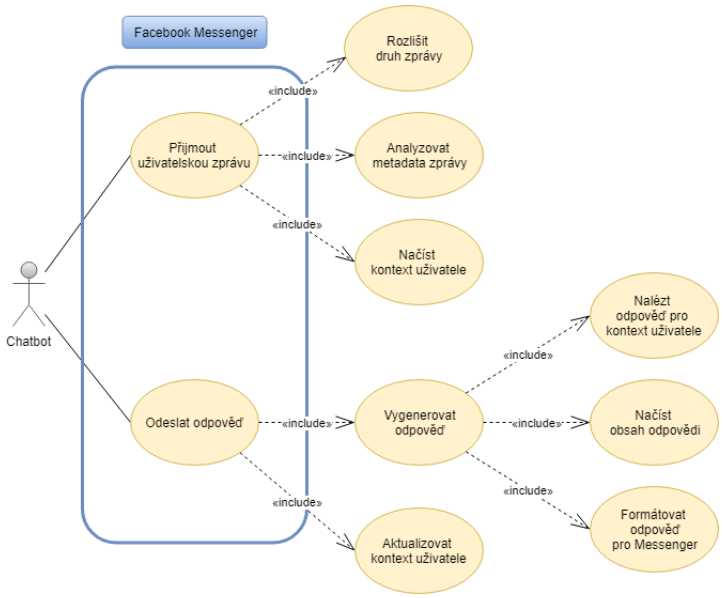
Pro konverzaci s botem na facebookové stránce je nutné během vývoje mít zřízený facebookový účet, který má na vývojové platformě přístupová práva k aplikaci.

Během ostrého provozu by měla být aplikace nasazena na veřejně přístupném webovém serveru s HTTPS a zároveň ověřena Facebookem skrze vývojovou platformu.

## 6.4 Chatbot

Následující sekce se zabývají popisem technické implementace a struktury chatbota. Jsou v nich popsány konkrétní funkce a datové struktury využité pro jeho tvorbu spolu se schématy využívaných databází. Vysvětlení jednotlivých prvků je vázáno na logicky seskupené celky operací prováděné chatbotem.

### 6.4.1 Diagram použití




Obrázek 5: Diagram použití chatbota

### 6.4.2 Struktura chatbota

Části chatbot aplikace jsou rozděleny do složek s JavaScript soubory obsahujícími Node.js moduly, kde každý z modulů sdružuje logicky související části funkcionality.

#### 6.4.2.1 chatbot.js

Tento soubor je základním stavebním kamenem aplikace. Je zde inicializován Express.js web server, který slouží jako webhook. Dále obsahuje funkce pro změnu chodu chatbota, pro ověření validity webhooku Facebookem a funkci reagující na přijaté události.

Spuštění chatbota je vázáno na spuštění souboru *server.js*, který využívá chatbot aplikaci jako middleware.

#### 6.4.2.2 message_handling

Moduly v této složce se starají o hlavní aspekty konverzace s uživatelem. Tyto aspekty se skládají z analýzy přijaté zprávy, vyhodnocení kontextu zprávy, vygenerování odpovědi a jejího odeslání uživateli. Složka obsahuje tyto soubory:

- **conversation-handler.js** - Implementuje logiku pro průběh konverzace s uživatelem.
- **user-message-handler.js** - Klasifikuje typ a obsah uživatelské zprávy.
- **context-handler.js** - Zjišťuje a aktualizuje kontext pro uživatele.
- **message-creation-handler.js** - Stará se o generování zpráv z konverzačního stromu a databází v závislosti na daném kontextu.
- **messenger-api.js** - Stará se o formátování dat do podoby akceptované platformou Messenger a jejich následné odeslání.

#### 6.4.2.3 database_modules

Databázové moduly spravují komunikaci mezi jednotlivými databázemi, které chatbot využívá. Všechny využívají třídu definovanou v souboru *database-class.js*, která zpřehledňuje používání dotazů. Složka obsahuje tyto soubory:

- **chatbot-DB.js** - Implementuje dotazy do databáze týkající se konverzačního stromu a FAQ.
- **context-DB.js** - Implementuje dotazy do databáze s kontexty uživatelů konverzujících s chatbotem.
- **resource-DB.js** - Implementuje dotazy do databáze se zdrojovými tabulkami zaměstnanců, testů a oborů.

#### 6.4.2.4 tree_modules

Stromové moduly se starají o definici, sestavení a procházení konverzačního stromu na základě dat získaných z asociovaných databází. Složka obsahuje tyto soubory:

- **conversation-tree.js** - Obsahuje definici struktury konverzačního stromu a funkce využité pro jeho naplnění při spuštění aplikace.
- **tree-traverser.js** - Modul se skládá z funkcí majících za úkol procházení konverzačního stromu a nalezení vyžadovaných dat dle kontextu konverzace.

#### 6.4.2.5 auxiliary_modules

Pomocné moduly obsahují definice funkcí implementujících generování specifických typů zpráv. Složka obsahuje tyto soubory:

- **carousel-handler.js** - Stará se o generování elementu specifického pro Messenger zvaného carousel v překladu kolotoč[9], který je použit při zobrazování zaměstnanců a studijních oborů.
- **FAQ-handler.js** - Modul se stará o generování odpovědí na otázky uživatelů, které jsou obsažené v FAQ.
- **fields-handler.js** - Modul obsluhuje generování odpovědí s informacemi o jednotlivých studijních oborech nalezených ve zdrojové databázi.
- **tests-handler.js** - Modul zprostředkovává funkce využité při realizaci možnosti vyplnění zkušebních přijímacích testů přímo v chatbotovi a stažení jejich pdf verzí.

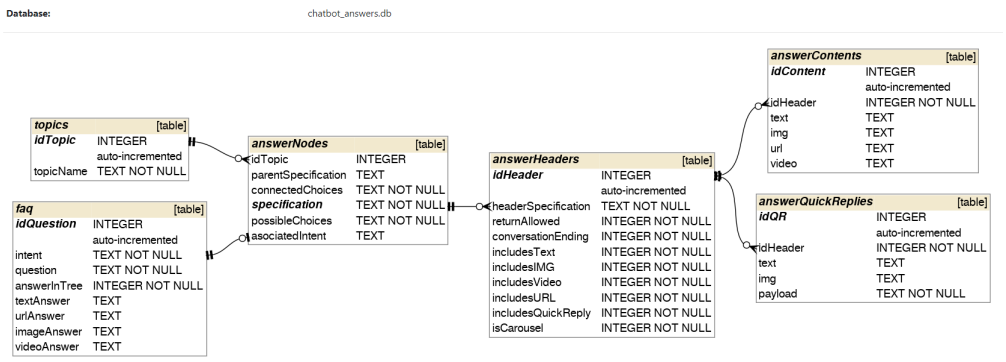
[9]Kolotoč je kolekcí elementů skládajících se z obrázku rozměrů 500x500, titulku, podtitulku a maximálně tří tlačítek. Kolotočem je nazván kvůli seřazení prvků v Messengeru do posuvné fronty.

### 6.4.3 Databáze

#### 6.4.3.1 Chatbot databáze

Název souboru databáze: *chatbot_answers.db*
Popis: *Databáze obsahuje součásti pro tvorbu jednotlivých uzlů konverzačního stromu doplněných o tabulku často kladených otázek.*

- **topics** - Tabulka obsahující možná témata konverzace včetně pomocných k zobrazování specifických informačních zpráv.
- **answerNodes** - Tabulka základních informací o uzlech konverzačního stromu.
- **answerHeaders** - Tabulka obsahující pravdivostní hodnoty[10] použité při generování obsahu uzlu.
- **answerQuickReplies** - Tabulka obsahuje data o rychlých odpovědích pro jednotlivé uzly.
- **answerContents** - Tabulka uchovává možný obsah zprávy pro každý uzel stromu.
- **faq** - Tabulka často kladených otázek obsahuje otázku a její odpověď, nebo příznak přítomnosti odpovědi v konverzačním stromě, sloupec intent slouží pro její klasifikaci v rámci Wit.ai aplikace.



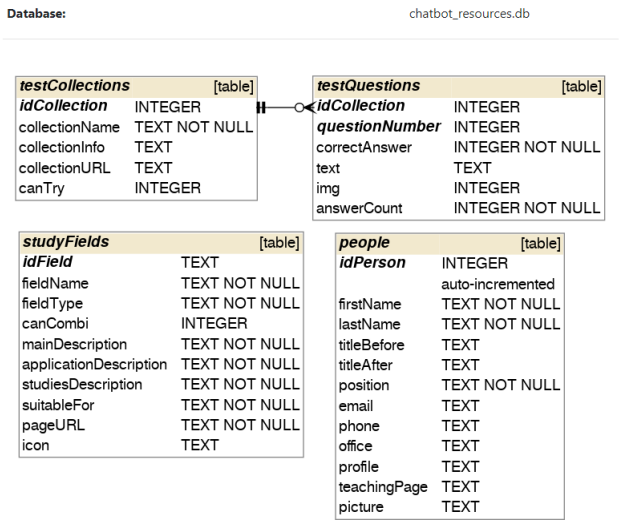

Obrázek 6: Diagram databáze chatbot_answers

[10] V SQLite se pravdivostní hodnoty ukládají jako celá čísla konkrétně 0 nebo 1.

#### 6.4.3.2 Databáze zdrojů

Název souboru databáze: *chatbot_resources.db*
Popis: *Databáze obsahuje data vázaná ke studijním oborům, přijímacím testům a zaměstnancům katedry.*

- **testCollections** - Tabulka popisující kolekce testů.
- **testQuestions** - Tabulka obsahující otázky pro kolekce testů.
- **people** - Tabulka obsahující veřejně přístupné informace o zaměstnancích Katedry informatiky.
- **studyFields** - Tabulka obsahuje informace o všech oborech v rámci Katedry informatiky.

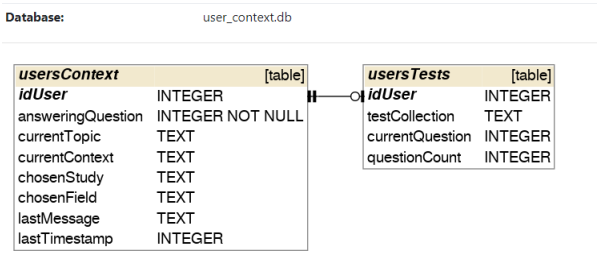



Obrázek 7: Diagram databáze chatbot_resources

#### 6.4.3.3 Kontextová databáze

Název souboru databáze: *user_context.db*
Popis: *Databáze obsahuje informace o uživatelích nutné pro správné určení aktuálního místa ve vedené konverzaci.*

- **userContext** - Tabulka obsahuje informace potřebné pro zjištění současného stavu konverzace s daným uživatelem.
- **userTests** - Tabulka obsahující stav daného uživatele během vyplňování testů.




Obrázek 8: Diagram databáze user_context

### 6.4.4 Komunikace s platformou Messenger

Pokud má aplikace komunikovat s Messengerem je nutné nastavit několik věcí na stránkách aplikace, která musí být vytvořena na vývojářské platformě Facebooku.

1. Vygenerovat unikátní přístupový klíč aplikace, který umožní komunikaci mezi Facebook stránkou a chatbot aplikací.
2. Nastavit a ověřit webhook, na který má Facebook zasílat vývojářem zvolená data.
3. Nepovinným krokem je přidružení Wit.ai aplikace, která následně ke každé zprávě z Messengeru přidá svoje metadata, se kterými může aplikace pracovat.
4. Pro ostrý provoz je nutné od Facebooku vyžádat schválení aplikace pro umožnění přístupu k jejich Send API, potřebné k získávání a odesílání zpráv z Messengeru, čímž Facebook zároveň ověřuje validitu a nezávadnost celé aplikace.

### 6.4.5 Tvorba konverzačního stromu

Vytvoření samotných stromů pro témata konverzace se volá při spuštění chatbot aplikace a to pouze jednou, což napomáhá zachování konzistence dat ve stromech při změnách provedených v databázích za chodu aplikace. Pro propagaci změn je tudíž vyžadován restart aplikace z webového rozhraní. Tvorba stromu probíhá v těchto krocích:

1. Načtení všech uzlů a jejich přidružených dat z databáze.
2. Rozdělení uzlů podle témat spojené s nalezením kořenových uzlů pro jednotlivá témata.
3. Vytvoření vlastního stromu konverzace pro každé téma, které je následně rekurzivně plněno z pole uzlů daného tématu na základě vztahu rodič-potomek. Každý uzel je v tomto kroku navíc formátován do podoby uzlu datové struktury stromu. [15]

```
1  class Node {
2     constructor(dataObject) {
3       //Název kontextu
4         this.context = dataObject.context;
5         /* V konverzaci skrytý obsah možných rychlých odpovědí
6         předcházející tomuto uzlu */
7         this.lastChoices = dataObject.lastChoices;
8         /* V konverzaci skrytý obsah možných rychlých odpovědí, které
              tento uzel obsahuje */
9         this.choices = dataObject.choices;
10        this.header = dataObject.header;
11        this.quickReplies = dataObject.quickReplies;
12        //Data s možnými obsahy zprávy
13        this.contents = dataObject.contents;
14        this.parent = null;
15        this.children = [];
16    }
17 }
```

Zdrojový kód 4: Podoba uzlu konverzačního stromu

### 6.4.6 Přijmutí uživatelské zprávy

Když uživatel zašle zprávu, ať už textovou či skrze kliknutí na rychlou odpověď nebo tlačítko, Messenger zašle tuto událost chatbotovi. Ten zprávu přijme a pokud je spuštěn, pošle ji dále ke zpracování.

O tuto funkčnost se stará funkce *handleEvents* v souboru *chatbot.js*.

```
function handleEvents(req, res) {
  if (!CHATBOT_IS_RUNNING) {
    // Pokud chatbot neběží pouze potvrdí přijetí zprávy
    res.status(200).send('EVENT_RECEIVED');
    return;
  } else {
    let body = req.body;
    // Kontrola zda přišla zpráva ze stránky
    if (body.object === 'page') {
      body.entry.forEach(function (entry) {
        // Získá informace o~události
        let webhookEvent = entry.messaging[0];
        let senderPSID = webhookEvent.sender.id;
        let messageTime = entry.time;
        // Zašle data ke zpracování vnitřní logikou chatbota
        //(ignoruje zpětné zprávy o doručení)
        if (webhookEvent.message || webhookEvent.postback) {
          conversationHandler.handleReceivedEvent(senderPSID,
              webhookEvent, messageTime);
        }
      });
      // Vrací '200 OK' odpověď na všechny zprávy ze stránky
      res.status(200).send('EVENT_RECEIVED');
    } else {
      res.sendStatus(404);
    }
  }
}
```

Zdrojový kód 5: Funkce handleEvents

### 6.4.7 Zpracování zprávy

Zpráva je postupně zpracována funkcí *handleConversation*, kde zpráva prochází těmito kroky:

1. Extrahují se data s obsahem a typem zprávy.
2. Načte se aktuální kontext uživatele, který zprávu zaslal.
    - V případě, kdy uživatel zahájí s chatbotem konverzaci poprvé, je pro uživatele vytvořen nový kontext a zaslána informační zpráva s představením aplikace uživateli. Načež je přesměrován do hlavního menu témat.
3. Prioritně se zkontroluje, zda uživatel nezadal jeden z možných příkazů definovaných uvnitř chatbota, ať už pomocí rychlé odpovědi nebo textově.
4. Chatbot zkontroluje jestli má pouze vygenerovat zprávu pro aktuální kontext, nebo reagovat na zprávu uživatele a posunout konverzaci dále.

- Mimo jiné je přidána kontrola, zda od poslední zprávy uživatele neuplynul více než den. Pokud ano, je mu zaslána zpráva vítající ho zpět, po které konverzace pokračuje tam, kde dříve skončila.

5. Pokud aplikace reaguje na uživatelovu odpověď, je volána funkce *handleUserAnswer*, která zjistí validitu odpovědi v aktuálním kontextu. Mohou nastat tyto případy:

   (a) Uživatel poslal zprávu s obsahem, na který chatbot neumí reagovat např. video. Reakcí je informační zpráva o nepochopení odpovědi.

   (b) Odpověď uživatele je validní v aktuálním kontextu. Reakcí je posunutí konverzace dále dle konverzačního stromu a aktualizace kontextu.

   (c) Odpověď uživatele není validní v aktuálním kontextu. V návaznosti na to se otestuje, jestli byla, díky Wit.ai, zachycena otázka z FAQ. Pokud byla, je kontext změněn na *odpovedFAQ* a poslán dále. V druhém případě je zaslána informační zpráva s nepochopením odpovědi a dochází k zopakování poslední zprávy.

6. Dojde k vygenerování odpovědi pro zvolený kontext, která se odešle uživateli na Messenger.

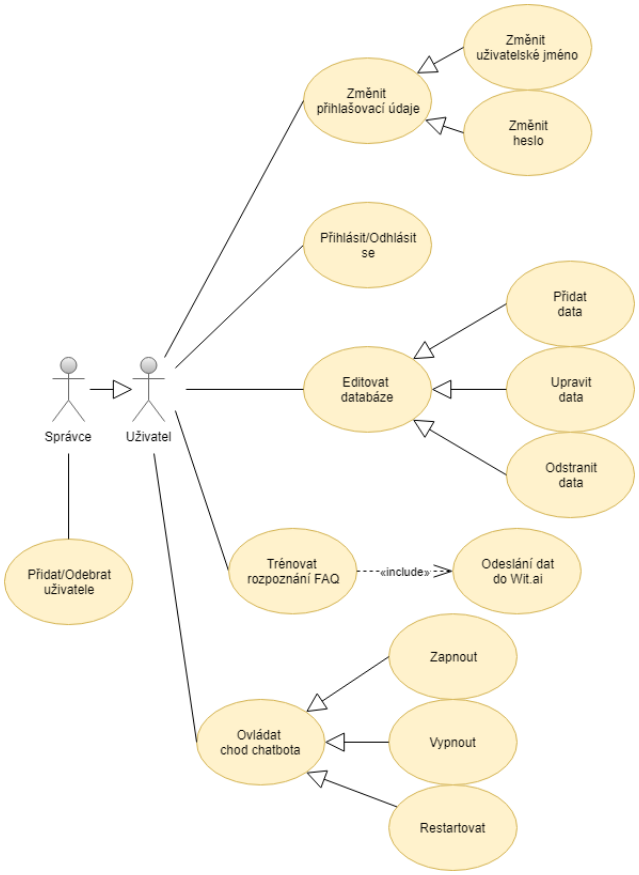
### 6.4.8 Vygenerování odpovědi

Po zpracování zprávy se přechází ke generování odpovědi na základě kontextu. Chatbot nejdříve kontroluje, zda se nenachází v jednom z tzv. dynamických kontextů, které vlastním způsobem implementují tvorbu zpráv. Dochází k tomu například u FAQ, testových otázek nebo u informací o vybraném studijním oboru. O tyto kontexty se starají odpovídající pomocné moduly.

Zbylé odpovědi se generují způsobem, kdy se pomocí informací o podobě a obsahu odpovědi z konverzačního stromu postupně naformátují do JSON objektu se strukturou vyžadovanou platformou Messenger.

## 6.5 Webové rozhraní

### 6.5.1 Diagram použití




Obrázek 9: Diagram použití webového rozhraní

### 6.5.2 Struktura webového rozhraní

Hlavním prvkem aplikace je soubor *server.js*, který obsahuje inicializaci Express.js serveru spolu se základním nastavením externích middlewarů a definicí reakce na HTTP požadavky vyhodnocené se statusem 404 a 500.

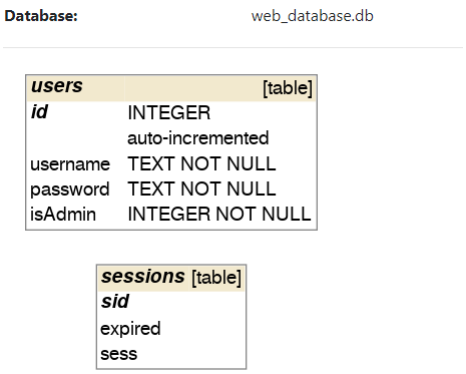
Webové rozhraní se dále skládá ze souborů s middleware funkcemi, souborů Pug šablon pro dynamické generování stránek a směrovacího souboru spojujícího předchozí dvě položky ve fungující celek.

Rozhraní má svou vlastní SQLite databázi pro administrativu webu. Složka *models* obsahuje modul *users.js* implementující komunikaci právě s touto databází.

### 6.5.3 Webová databáze

Název souboru databáze: *web_database.db*
Popis: *Databáze obsahuje údaje o uživatelích a právě aktivních webových sezeních.*

- **users** - Tabulka obsahuje uživatelská jména a hašovaná hesla uživatelů doplněné o příznak statutu administrátora webu.
- **sessions** - Tabulka uchovávající identifikátor sezení, dobu vypršení jeho platnosti a JSON objekt s cookie daného sezení. Každá cookie obsahuje následující:
  - uživatelovo id
  - příznak správcovského statutu
  - příznak, zda je uživatel přihlášen
  - objekt se zprávami pro dané sezení
  - objekt s chybovými hlášeními pro dané sezení




Obrázek 10: Diagram databáze web_database

### 6.5.4 Složka public

Složka public obsahuje statické soubory stránky viditelné koncovým uživatelem.

Obsahuje kaskádové styly pro stránku obsažené ve *style.css* a styly pro využité ikony v *icons.css* spolu se vzhledem ve složce *webfonts*. Všechny použité ikony jsou vybrány z bezplatné distribuce Font Awesome 5 Free.

Dále se zde nachází vytvořené logo chatbota a vygenerované favicon zmenšeniny.

Nakonec je zde umístěno i několik JavaScript souborů:

- **answers-accordion.js** - Implementuje interaktivitu pro element využitý při rozdělení odpovědí dle témat.

- **disable-unused-inputs.js** - Stará se o znepřístupnění vstupů ve formulářích v závislosti na vybraném druhu odpovědi.
- **FAQ-form-toggles.js** - Stará se o přepínání zobrazení výběru FAQ odpovědi mezi existující a nově definovanou.
- **selects-toggles.js** - Zajišťuje přehlednější zobrazování vybraných možností v elementech typu select.
- **test-questions-toggles.js** - Stará se o přepínání mezi textovým a obrázkovým zadáním testové otázky a změnu možných správných odpovědí v závislosti na celkovém počtu odpovědí.

### 6.5.5 Šablony webových stránek

Soubory šablon s příponou *.pug* jsou umístěny ve složce *views* a rozděleny do podsložek kopírující strukturu webové stránky.

```
1  //Obsah tohoto souboru je přidán jakou součást šablony níže
2  extends ../components/main-layout
3
4  //Block určuje místo v nadřazené šabloně, kam se má kód přidat
5  block content
6    //V šabloně lze deklarovat vlastní proměnné
7      - var chosenTopic = (topic === undefined ? -1 : topic.idTopic);
8      section
9          h1(class="header-text--centered") Upravit téma
10
11         //Do šablon lze vkládat jiné šablony
12         include ../components/errors
13
14         form(action="/topic-choose", method="post" name="chooseTopic")
15           label(for="topicSelect") Vyberte téma:
16           select(name="topicSelect" id="topicSelect")
17
18               //Lze využívat cyklů pro procházení předaných argumentů
19               each topic in topicsSelect
20                //Atributům HTML značek lze předávat proměnné
21                  option(value= topic.idTopic) #{topic.topicName}
22
23     //Přidání obsahu na stránku může být podmíněno hodnotou proměnné
24         if(chosenTopic !== -1)
25            h2(class="header-text--centered") Detaily o~tématu
```

Zdrojový kód 6: Příklad šablony Pug

### 6.5.6 Middleware

Jedná se o funkce navázané na volání HTTP metod ve směrovacím souboru.

Poté co se daná metoda provede jsou na výsledné datové objekty[11] postupně aplikovány middleware funkce, které na data nějakým způsobem reagují nebo je upravují. Po skončení své činnosti mají vždy moc zavolat další middleware funkci ve frontě.

V rámci webového rozhraní jsou nově definované middleware moduly umístěny ve složce middleware.

- **autentification.js** - Soubor obsahuje funkce, které reagují na přihlášení a odhlášení uživatele. Konkrétně mají za úkol pro uživatele vytvořit sezení a zkontrolovat platnost zadaných přihlašovacích údajů.

- **edit-answers.js** - Soubor obsahuje funkce pro správu odpovědí chatbota. Stará se o aktualizaci databází skrze modul *chatbot-db-updater.js*, získání informací o jednotlivých odpovědích, formátování získaných dat z požadavku a o část validace zaslaných formulářů.

- **edit-credentials.js** - Soubor obsahuje funkce pro změnu přihlašovacích údajů, včetně zjištění těch stávajících a kontroly dostupnosti nově zadaných.

- **edit-FAQ.js** - Soubor obsahuje funkce potřebné k aktualizaci Wit.ai aplikace spojené s aktualizací chatbot databáze skrze pomocný modul *chatbot-db-updater.js*.

- **edit-resources.js** - Soubor se skládá z funkcí spravujících zdrojové tabulky obsahující data o zaměstnancích, studijních oborech a přijímacích testech. Mimo jiné obsahují i funkce pro získání aktuálních hodnot a formátování získaných dat. Jsou zde využívány moduly pro aktualizování databází *chatbot-db-update.js* a *resource-db-updater.js*.

- **edit-topics.js** - Soubor obsahuje funkce pro správu témat a formátování získaných dat. Při odstraňování témat je využit *edit-FAQ.js* modul pro smazání FAQ odpovědí asociovaných s daným tématem.

- **edit-user.js** - Soubor obsahuje funkce pro správu uživatelů, ke které má přístup hlavní správce. Realizuje možnosti přidání a odebrání uživatelů.

- **error-handling.js** - Soubor obsahuje dvě pomocné funkce spolupracující s externím modulem *express-validator* využívané při nacházení chybových hlášení ve validovaných formulářích.

- **utils.js** - Soubor obsahuje funkci pro získávání parametrů ze zadané adresy a funkci spravující informační zprávy zasílané uživateli během sezení.

- **wit-ai.js** - Soubor se stará o formátování a zasílání požadavků na vzdálený server Wit.ai aplikace.

[11] Přesněji se jedná v Express.js o objekty req (obdržená data) a res (odesílaná data)

### 6.5.7 Směrování

Popis směrování v rámci webového rozhraní je obsažen ve složce *routes* v souboru *index.js*.

Uvedený soubor definuje s pomocí *Router* objektu, převzatého z Express.js, HTTP metody typu GET a POST pro všechny validní URL definované ve webovém rozhraní. Tyto definice jsou často doplněny o aplikaci middleware funkcí a u metod GET o příkazy vykreslující Pug šablony s možnou pomocí předaných argumentů.

POST metody jsou nejčastěji doplněné validačním middlewarem, kterým server dodatečně ověřuje platnost informací zadaných uživatelem.

```
1  //GET požadavek na zobrazení stránky s přihlášením
2  router.get('/prihlaseni', function (req, res) {
3    if (req.session && req.session.loggedIn) {
4    //Uživatel je již přihlášen, přesměruj ho do /menu
5      res.redirect('/menu');
6    }
7    else {
8    //Uživatel není přihlášen, vykresli šablonu s případnými errory
9      let error = errorHandler.checkForErrorMsg(req);
10     res.render('OTHERS/prihlaseni', { errors: error });
11   }
12 });
13
14 //POST požadavek na přihlášení
15 router.post('/login', [
16   //Validace údajů pomocí express-validator
17   check('password')
18     .custom(async (value, { req }) => {
19       let result = await auth.checkLoginInfo(req);
20       if (!result) throw new Error('Špatné uživatelské jméno nebo
             heslo.');
21     })
22 ], errorHandler.checkForErrors, async function (req, res) {
23   await auth.logIn(req);
24   res.redirect('/menu')
25 });
```

Zdrojový kód 7: Příklad GET a POST v index.js

### 6.5.8 Editace databází

Editace databází se děje v reakci na obdržení validovaného POST požadavku na server odeslaného jedním z formulářů přítomných na webové stránce.

Pro nalezení editované položky jsou získány parametry z URL přijatého požadavku, které jsou poté spolu s formulářovými daty přeposlány jednomu z middlewarů obhospodařujících editaci databází. Ten data zformátuje a pomocí jednoho

z pomocných databázových modulů je skrze transakci zapíše do databáze. Následně se ověří, jestli byl zápis úspěšný. Poté je uživateli aktualizována stránka, na které je mu zobrazena buď zpráva o úspěšném vykonání akce, nebo chybová zpráva obsahující důvod chyby.

### 6.5.9 Editace Wit.ai

Editace se děje skrze přidružený middleware, který odesílá požadavky skrze HTTP API definované Wit.ai pro editaci zdrojů přítomných v jejich aplikacích. Po vytvoření aplikace k ní má uživatel možnost získat přístup skrze vygenerovaný unikátní klíč, který vloží do hlavičky HTTP požadavku.

Je nutné zmínit existenci alternativního způsobu editace asociované Wit.ai aplikace, a to skrze přihlášení do webového rozhraní dostupného na stránce wit.ai.

Webové rozhraní chatbota v této práci implementuje pouze část obsáhlejší funkcionality výše zmíněného nástroje nezbytnou pro správný běh chatbota a základní editaci Wit.ai aplikace.

# 7 Uživatelská dokumentace

Následující sekce popisují uživatelská nastavení a akce, kterými disponuje chatbot a webové rozhraní.

## 7.1 Chatbot

Jelikož o správu samotného chatbota se stará webové rozhraní budou v následující části práce představena pouze nastavení spjatá se založením facebookové aplikace spolu s výčtem možností uživatelské interakce, které chatbot skrze platformu Messenger momentálně nabízí.

### 7.1.1 Vývojová platforma Facebooku

Pro vývoj aplikace využívající služeb poskytovaných firmou Facebook je nutné si vytvořit Facebook účet, který dá vývojáři přístup do platformy pro vývoj aplikací na stránce developers.facebook.com.

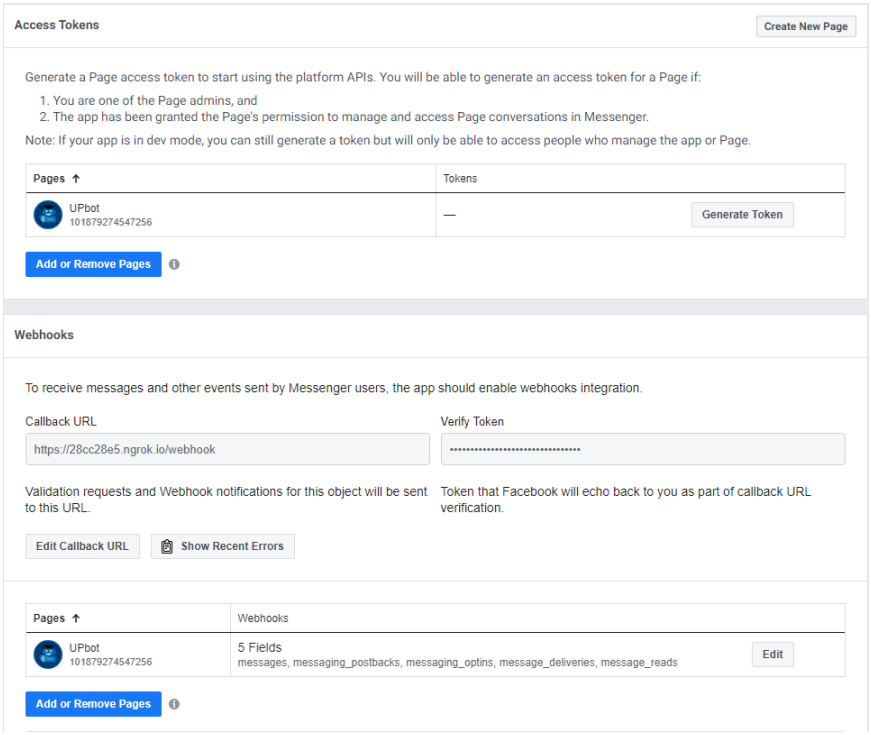
Vývojářská platforma poskytuje možnost spravovat uživatele, kteří mají k aplikaci přístup, ať už pro testování, vývoj nebo analytiku. Dále nabízí přehled základních analytických dat, které si Facebook o aplikaci uchovává a seznam oprávnění a služeb, které byly pro aplikaci zvoleny.

Chatbot konkrétně využívá služeb Messenger a Webhook, které Facebook nabízí, a má zároveň oprávnění týkající se zpracování událostí obdržených asociovanou Facebook stránkou.

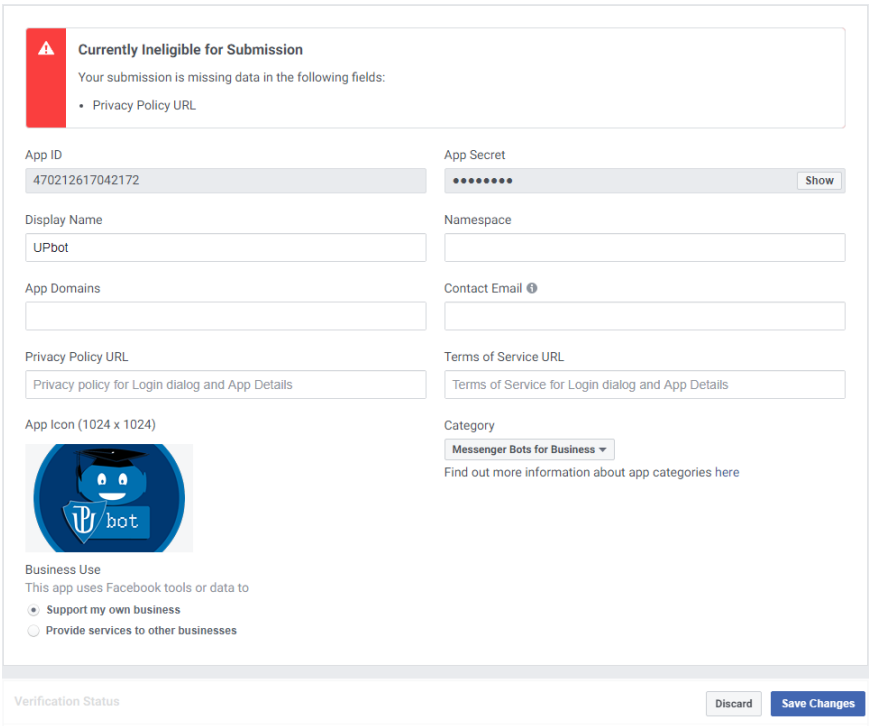
Na obrázku 11 je zobrazena část stránky pro nastavení aplikace v platformě Messenger. Tato stránka umožňuje následující akce:

- Generování klíče, který spojí Facebook stránku s vytvořenou aplikací.
- Změnu adresy webhooku, na který odesílá platforma Messenger přijaté události, spolu s uživatelsky definovaným klíčem, kterým bude webhook ověřen.
- Změnu oprávnění, která má zadaný webhook vůči asociovaným stránkám.
- Možnost propojení s Wit.ai aplikací.
- Odeslání na posouzení zaměstnanci Facebooku, po kterém aplikace získá povolení k využívání zvolených služeb během ostrého provozu.

Co se týká ostrého provozu aplikace, je nutné, aby byly vyplněny všechny náležité informace uvedené na obrázku 12 spjaté s aplikací, aby mohla být odeslána k testování, po kterém obdrží práva na komunikaci s uživateli Messengeru. K tomuto dojde po vyžádáním přístupu k tzv. **pages_messaging**, které umožní přijímání a zasílání zpráv všem uživatelům Messengeru.




Obrázek 11: Stránka s nastavením aplikace




Obrázek 12: Stránka s informacemi o aplikaci

### 7.1.2 Ukázka konverzačních elementů

Messenger nabízí kromě prostého zasílání zpráv i několik různý prvků a šablon elementů, s kterými mohou uživatelé interagovat. Níže jsou zmíněny prvky, které chatbot momentálně využívá. Zbylé možnosti interakce s uživateli lze nalézt v dokumentaci platformy Messenger.

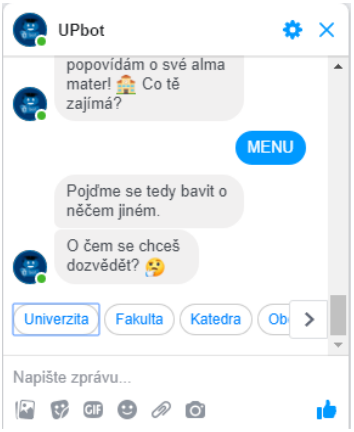
Je nutné podotknout, že níže zmíněné elementy lehce mění svůj vzhled v závislosti na zařízení, kde jsou zobrazovány. Mimo jiné v mobilní verzi Messenger aplikace se mění v závislosti na tom, zda je zvolený tzv. tmavý mód aplikace.[12]

#### 7.1.2.1 Rychlé odpovědi

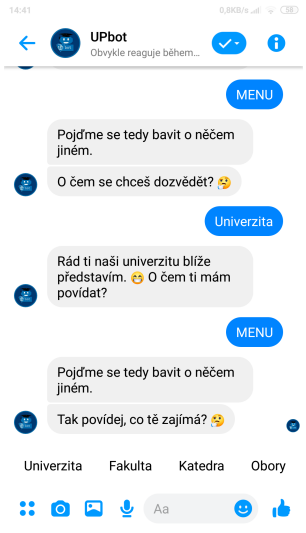
Rychlé odpovědi jsou jedním z hlavních prvků konverzace s aplikací chatbot. Uživatel je po kliknutí na jednu z možností posunut v konverzaci dále dle konverzačního stromu. Jejich maximální počet dle dokumentace [16] je 13 u jedné zprávy.

V aplikaci se opakovaně objevují i dvě rychlé odpovědi se specifickou funkčností:

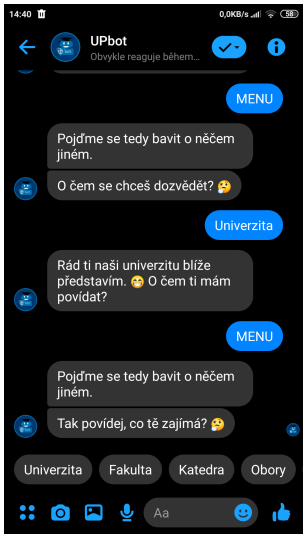
- **MENU** - Možnost vrátí uživatele k výběru z možných témat a je přítomná jako poslední možnost u všech seznamů rychlých odpovědí
- **ZPĚT** - Možnost vrátí uživatele zpátky v konverzaci o jeden krok. Tlačítko je přítomné téměř u všech odpovědí kromě těch dynamicky generovaných.




Obrázek 13: Rychlé odpovědi na počítači




Obrázek 14: Rychlé odpovědi na mobilu

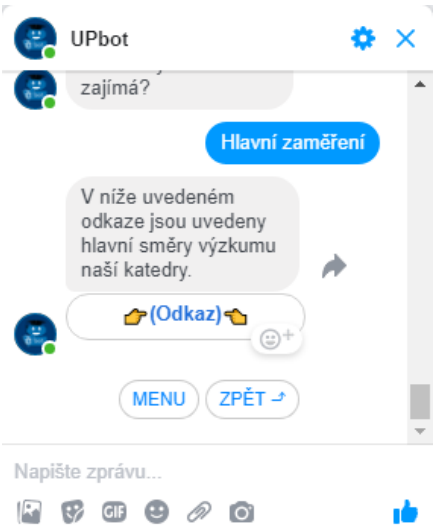



Obrázek 15: Rychlé odpovědi na mobilu tmavě

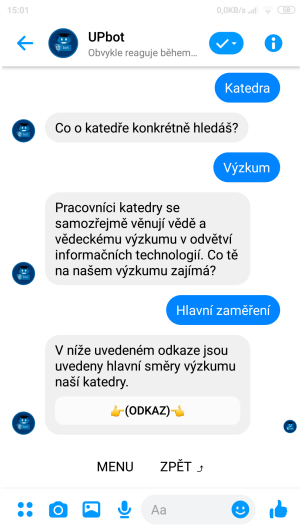
[12] U příkladů rychlých odpovědí jsou ukázána všechna možná zobrazení, dále však už budou využívány pouze výchozí mobilní verze bez tmavého módu, jelikož funkcionalita ve všech variantách zůstává zachována.

#### 7.1.2.2 Tlačítko s odkazem

Dalším prvkem konverzace je využití tlačítkové šablony Messengeru. Ta umožňuje zobrazit u zprávy maximálně 3 tlačítka. Chatbot využívá její lehkou úpravu, kterou je pouze jedno tlačítko obsahující odkaz na zdroje podrobnějších informací, které by nebylo praktické v konverzaci s chatbotem zobrazovat.




Obrázek 16: Tlačítko na PC




Obrázek 17: Tlačítko na mobilu

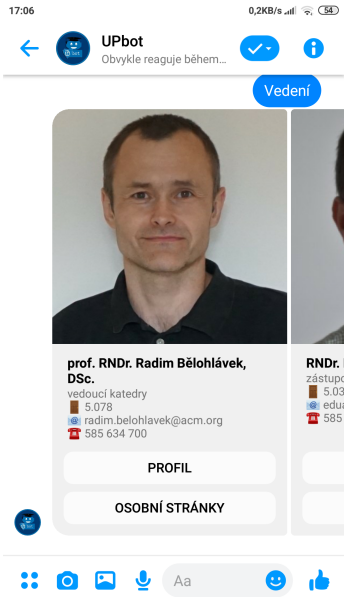
#### 7.1.2.3 Element kolotoč

Element kolotoč má formu obrázku o rozměrech 500x500 pixelů doplněném o titulek, podtitulek a maximálně 3 tlačítka. Těchto elementů může vedle sebe být až 10 v jedné zprávě a lze je prohlížet posunem do strany.

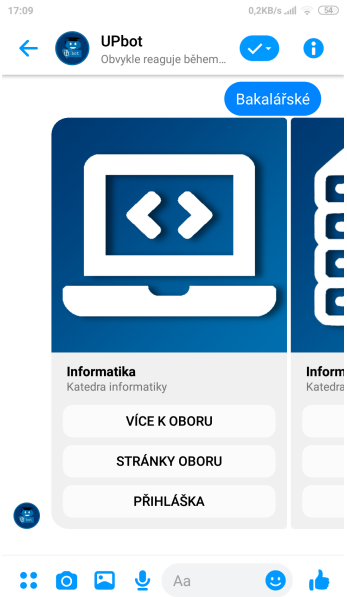
Chatbot tento element momentálně využívá ve dvou případech:

- **Informace o lidech** - Element je v tomto případě naformátovaný jako medailonek s profilovým obrázkem, jménem, pozicí a kontaktními informacemi zaměstnanců katedry. Zobrazovaná tlačítka v sobě nesou odkazy na profil na stránkách katedry a případně na osobní stránky uvedené na webu katedry.

- **Informace o oborech** - Element je se v tomto případě skládá z profilového obrázku oboru vytvořeného z ikon přítomných na stránkách katedry, jména oboru, názvu katedry a odkazu na stránky katedry. Tři tlačítka obsahují možnosti dozvědět se více o oboru v rámci konverzace s chatbotem, odkaz na stránku s oborem a odkaz k přihlášce.




Obrázek 18: Element kolotoč se zaměstnanci




Obrázek 19: Element kolotoč se studijními obory

### 7.1.3 Vyzkoušení přijímacího testu

Součástí konverzace je také možnost vyzkoušet si několik ukázkových přijímacích testů.

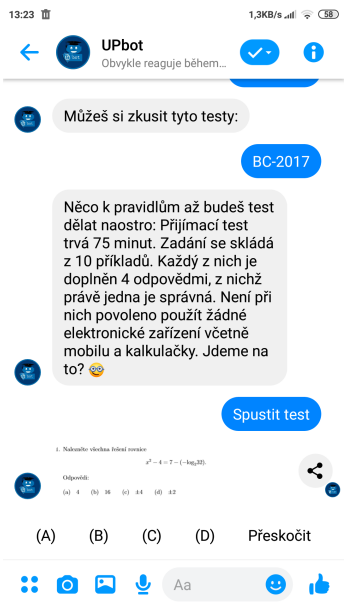
Při zahájení testu je uživateli poslána zpráva, za jakých podmínek a pravidel se test vypracovává během přijímací zkoušky. Poté jsou uživateli postupně zasílány textovou, nebo obrázkovou formou zadání jednotlivých otázek. Uživatel může zvolit z poskytnutých odpovědí, nebo otázku přeskočit. Po zvolení jedné z odpovědí na testovou otázku má uživatel několik možností. Může test ukončit, může pokračovat na další otázku, nebo se při nesprávné odpovědi může pokusit otázku zodpovědět znova.

Jak vypadá začátek testu je ilustrováno na obrázku 20.

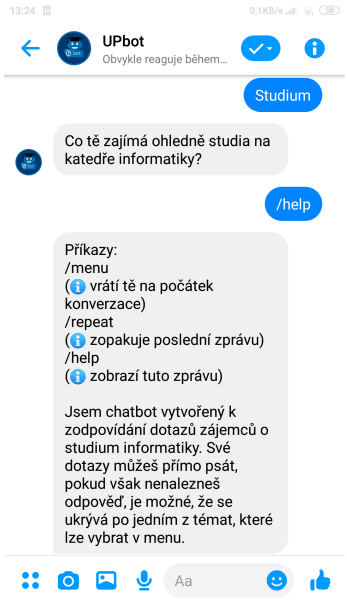
### 7.1.4 Příkazy

Příkazy jsou v chatbotovi zprávy začínající znakem "/". Tyto speciální zprávy umožňují vyvolat kdykoliv během konverzace určité akce. Momentálně jsou podporovány 3 příkazy:

- **/menu** - Tento příkaz vrátí uživatele zpět do menu témat.
- **/repeat** - Tento příkaz zopakuje uživateli poslední zprávu, kterou od chatbota obdržel.
- **/help** - Tento příkaz, zobrazen na obrázku 21, zašle uživateli zprávu s výpisem všech příkazů, které chatbot obsahuje doplněnou o základní popis chatbota.




Obrázek 20: Ukázka testu

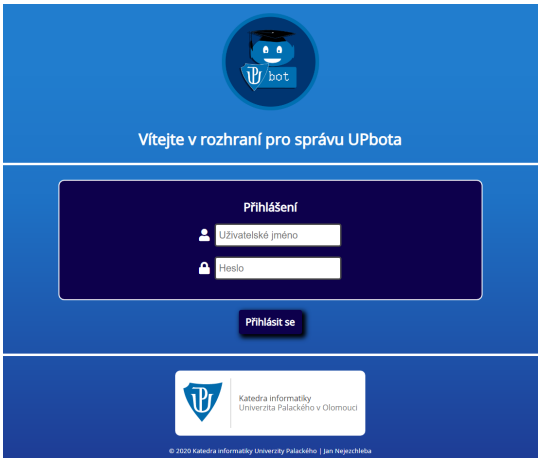



Obrázek 21: Ukázka příkazu /help

## 7.2 Webové rozhraní

Tato část popisuje jednotlivé stránky webového rozhraní, se kterými se může uživatel během správy chatbota setkat. Všechny webové stránky rozhraní se přizpůsobují velikosti zařízení, na kterém jsou zobrazovány. Uvedené obrázky jsou ukázkou zobrazení na monitoru.

### 7.2.1 Přihlášení




Obrázek 22: Přihlašovací stránka

Každý nepřihlášený uživatel se nejdříve ocitne na stránce, kde jsou od něj

vyžadovány přihlašovací údaje. Po úspěšném přihlášení je přesunut do hlavního menu, odkud má přístup ke všem funkčnostem rozhraní.

### 7.2.2 Hlavní menu


Obrázek 23: Stránka s hlavním menu

Hlavní menu se skládá ze záhlaví a zápatí stránky a jejího obsahu ve formě tlačítek pro zvolení prvků chatbota, které chce uživatel spravovat. Prvky hlavního menu nabízí tyto akce:

- **Záhlaví** - Je přítomno na všech stránkách rozhraní a obsahuje tyto prvky:
  - **Uživatel** - Přesměruje uživatele do uživatelského nastavení.
  - **Logo UPbota** - Přesměruje uživatele zpět do hlavního menu z jakékoliv stránky.
  - **Odhlásit** - Odhlásí uživatele a přesměruje ho na stránku s přihlášením.
- **Volby správy**
  - **Spravovat FAQ** - Přesměruje uživatele do správy FAQ otázek obsahující možnosti editace otázek a trénování jejich rozpoznávání.
  - **Spravovat konverzační strom** - Přesměruje uživatele do správy konverzačního stromu obsahující možnosti editace témat a odpovědí.
  - **Spravovat zdrojové tabulky** - Přesměruje uživatele do správy zdrojových tabulek obsahující možnosti editace informací o zaměstnancích, studijních oborech a ukázkových přijímacích testech.

  - **Ovládání** - Přesměruje uživatele na stránku s možnostmi ovládání chatbota, které obsahuje tři možnosti, a to spustit, vypnout a restartovat.

- **Zápatí** - Je přítomno na všech stránkách rozhraní a obsahuje rámeček s odkazem na stránky Katedry informatiky.

### 7.2.3 Změna přihlašovacích údajů a editace uživatelů

Přihlašovací jméno a heslo mohou uživatelé změnit v nabídce ukryté pod tlačítkem *Uživatel* v záhlaví stránky.

Pokud má přihlášený uživatel práva administrátora, v nabídce se objeví i správy uživatelů webového rozhraní. Pro přidání stačí stanovit výchozí jméno a heslo. Pro odebrání stačí najít daného uživatele a potvrdit akci.


Obrázek 24: Stránka s uživatelským nastavením

### 7.2.4 Správa FAQ

Správa nabízí možnosti přidání, úpravy, odebrání a trénování otázek. Přidání je možné tlačítkem *Přidat*. Všechny zbylé akce jsou přístupné vedle názvů jednotlivých otázek. Po zvolení jedné z možnosti je poté uživateli vždy zobrazen formulář.

Pakliže se uživatel rozhodne například přidat otázku, je nutné vyplnit následující informace:

- **Evidenční zkratku** - Krátké označení otázky pro interní evidenci ve Wit.ai.

- **Výchozí znění otázky** - Jakým způsobem se uživatel Messengeru na danou otázku může zeptat.

- **Odkaz na existující odpověď** - Pokud je odpověď na takovou otázku již přítomná v konverzačním stromě stačí ji vybrat ze zobrazené nabídky odpovědí. V druhém případě je potřeba vyplnit údaje o nové odpovědi.
- **Nová odpověď na otázku** - Zde se zvolí typ zobrazované zprávy a vloží její obsah.

Možnost trénování chatbota, taktéž přítomná pod správou FAQ, zahrnuje přidání varianty uživatelského vstupu pro zvolenou otázku pro přesnější rozpoznávání dané otázky aplikací.

Všechny akce spojené s editací a trénováním otázek zasílají data do externí aplikace Wit.ai závislé na vytíženosti serverů Facebooku, proto může trvat i několik minut, než je aplikace zpracuje a začne korektně reagovat na otázky, které byly změněny.


Obrázek 25: Stránka s editací FAQ

### 7.2.5 Editace témat

Tato možnost přítomná v nabídce správy konverzačního stromu obsahuje volbu přidání, úpravy a odebrání témat, pod kterými jsou odpovědi chatbota uspořádány do konverzačních stromů.

Během přidání je kromě názvu tématu a názvu jeho rychlé odpovědi, nutné zvolit i název kořenového kontextu a jeho obsahu. Ten slouží zejména jako úvod a hlavní rozcestník pro informace obsažené v daném tématu. Tento kontext je poté editovatelný přes editaci odpovědí.

Odebrání tématu způsobí, že všechny jeho součásti, jako odpovědi a FAQ otázky na něj odkazující, budou nenávratně smazány z důvodu zachování konzistence dat.

### 7.2.6 Editace odpovědí

Nabízí možnosti přidání, úpravy a odebrání jednotlivých odpovědí chatbota obsažených v konverzačním stromě některého z témat.

Položky jsou rozděleny do seznamu pro uživatelsky definovaná témata, do kterých lze libovolně přidávat a odebírat odpovědi, a na základní témata, která jsou esenciální pro chod chatbota a jeho funkcí a kterým lze pouze upravovat obsah.


Obrázek 26: Stránka s editací odpovědí

#### 7.2.6.1 Přidání

Během přidání je nutné zvolit umístění nové odpovědi v konverzačním stromě skrze volbu tématu a nadřazeného kontextu, pod kterým se bude nová odpověď zobrazovat.

Dále je stanoveno omezení, kvůli kterému se může pod každým kontextem nacházet maximálně 10 jeho potomků. Omezení plyne z dokumentace platformy Messenger a vyhrazeným využitím několika potomků pro speciální odpovědi. Stav potomků u zvoleného kontextu je zobrazen v tabulce, která je součástí formuláře.

V další části formuláře je potřeba vyplnit název kontextu odpovědi a znění její rychlé odpovědi.

Nakonec je zvolen typ odpovědi a vyplněn její obsah. Typy obsahu provází určitá omezení, a to taková, že obrázky i videa lze momentálně přidávat pouze pomocí přímých URL odkazů a jejich velikost nesmí překročit 25 MB, kterou povoluje Messenger.

#### 7.2.6.2 Úprava

Je možné upravit znění rychlé odpovědi spolu s obsahem odpovědi a lze tvořit více variant obsahu, ze kterých si chatbot při odesílání odpovědi vždy vybere

jednu náhodnou. To může posloužit jako vylepšení dynamičnosti konverzace.

#### 7.2.6.3 Odebrání

Odebrání odpovědi je nevratné a dochází při něm zároveň k odstranění všech potomků vybrané odpovědi spolu s odstraněním asociovaných FAQ otázek odkazujících na odebrané odpovědi.

### 7.2.7 Editace zdrojových tabulek

Mezi editovatelné zdrojové tabulky patří zaměstnanci, studijních obory a přijímací testy. Obsah ve zdrojových tabulkách lze přidávat, upravovat a mazat.


Obrázek 27: Stránka se správou zdrojových tabulek

#### 7.2.7.1 Editace zaměstnanců

Informace obsažené u jednotlivých zaměstnanců vychází z informací dostupných na webu katedry. Informace o kontaktu a odkazech jsou zcela nepovinné a stačí vždy pouze vyplnit základní informace o zaměstnanci.

#### 7.2.7.2 Editace oborů

Struktura informací o oborech je převzata ze stránek Katedry informatiky, kde jsou uvedeny vždy základní informace a popisy jednotlivých aspektů oboru.

#### 7.2.7.3 Editace testů

Přidání testů umožňuje přidat základní informace k testu doprovázené odkazem na soubor se zadáním.

Úpravy testů nabízí povolení vyzkoušení testu v chatbotovi. Po zaškrtnutí této možnosti je nutné přidat do testu otázky.

Formuláře pro editaci jednotlivých otázek se skládají z udání formy jejich zadání, a to buď textové, nebo písemné. Dále uživatel vybírá kolik otázka obsahuje odpovědí s maximálním počtem až 7 odpovědí. Nakonec uživatel vybere pod jakým písmenem je ukryta správná odpověď. Možný počet přidaných otázek není nijak omezen.

Odebrání testu současně smaže i všechny asociované otázky.

# Závěr

Cílem práce byla tvorba chatbota schopného komunikovat se zájemci o studium na Katedře informatiky. Tento cíl byl realizován skrze vytvoření aplikace využívající prostředí Node.js a platformu Messenger. Dále bylo vytvořeno webové rozhraní, taktéž v prostředí Node.js, zprostředkovávající správu chatbota spolu s možnou editací informací, které jsou uživatelům skrze něj poskytovány. Obě zmíněné aplikace mohou být dále vyvíjeny a vylepšovány.

U chatbotové aplikace se jako cíl dalšího vývoje nabízí implementace větší provázanosti s platformou Messenger skrze využití dalších nabízených interaktivních elementů konverzace, představení nové funkčnosti například na základě příkazů zadaných uživatelem a také další obohacování databází o informace, které chatbot uživatelům může poskytnout.

Pro webové rozhraní se do budoucna naskýtá možnost těsnější spolupráce se systémem STAG a interní databází katedry pro snadnější editovatelnost poskytovaných informací.

Osobně jsem se díky této práci hlouběji seznámil s vývojem aplikací pro Facebook a specifiky vývoje v běhovém prostředí Node.js, které mi zároveň poskytlo možnost lépe porozumět programovacímu jazyku JavaScript využívaném na serverové straně aplikace. Tvorba jednotlivých stránek pro webové rozhraní mi pak umožnila náhled do problematiky tvorby webu řešené s pomocí technologií HTML, CSS a JavaScript.

# Conclusions

The goal of this thesis was the creation of a chatbot that will be able to communicate with those interested in studying at the Department of Computer Science. This goal was realised through the creation of application using Node.js and Messenger platform. In addition to that a web interface was developed using Node.js to enable administration of chatbot application with possible editing of information, which will be provided to users through chatbot. Both aforementioned applications can be further developed and improved.

The next target of development for chatbot application can be a closer integration with Messenger platform through use of other possible conversation elements, introduction of new fuctionality e.g. based on user commands and also further addition of new information into the database that can be then provided to the users.

The future development of web interface provides the possibility of integration with STAG system and department's database to make the process of editing the information easier.

Personally, thanks to this thesis, I became more deeply acquainted with the development of applications for Facebook and Node.js, which at the same time offered me the chance to better understand the backend use of JavaScript programming language. Moreover, the creation of the web pages for the web interface made me research the issue of web development using HTML, CSS and JavaScript.

# A Obsah přiloženého CD/DVD

**bin/**
: Kompletní adresářová struktura webové aplikace UPbot (v ZIP archivu) pro zkopírování na webový server. Adresář obsahuje i instalační soubor Node.js.

**doc/**
: Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech příloh, a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu (v ZIP archivu), tj. zdrojový text textu, vložené obrázky, apod.

**src/**
: Kompletní zdrojové texty webové aplikace UPbot se všemi vytvořenými (příp. převzatými) zdrojovými texty a knihovnami.

**README.txt**
: Instrukce pro instalaci, testování a spuštění webové aplikace UPbot, včetně všech požadavků pro její bezproblémový provoz za účelem testování při tvorbě posudků práce a pro účel obhajoby práce.

Navíc CD/DVD obsahuje:

**assets/**
: Složky obsahující vytvořené obrázky a soubory převzaté z webu katedry, které chatbot využívá při generování odpovědí. Tyto soubory jsou pro účely testování aplikace nahrány na webová úložiště a jejich odkazy uloženy v databázích chatbota.

U veškerých cizích převzatých materiálů obsažených na CD/DVD jejich zahrnutí dovolují podmínky pro jejich šíření nebo přiložený souhlas držitele copyrightu. Pro všechny použité (a citované) materiály, u kterých toto není splněno a nejsou tak obsaženy na CD/DVD, je uveden jejich zdroj (např. webová adresa) v bibliografii nebo textu práce nebo v souboru `README.txt`.

# Literatura


[1] Wikipedia contributors. *JavaScript — Wikipedia, The Free Encyclopedia* [online]. 2020 [cit. 2020-5-9]. Dostupný z: ⟨https://en.wikipedia.org/w/index.php?title=JavaScript&oldid=955507334⟩.

[2] Wikipedia contributors. *ECMAScript — Wikipedia, The Free Encyclopedia* [online]. 2020 [cit. 2020-5-9]. Dostupný z: ⟨https://en.wikipedia.org/w/index.php?title=ECMAScript&oldid=954355333⟩.

[3] Wikipedia contributors. *Node.js — Wikipedia, The Free Encyclopedia* [online]. 2020 [cit. 2020-5-9]. Dostupný z: ⟨https://en.wikipedia.org/w/index.php?title=Node.js&oldid=955475730⟩.

[4] OpenJS Foundation. *About Node.js* [online]. 2020-01-01 [cit. 2020-5-9]. Dostupný z: ⟨https://nodejs.org/en/about/⟩.

[5] Přispěvatelé Wikipedie. *Npm — Wikipedie: Otevřená encyklopedie* [online]. 2019 [cit. 2020-5-9]. Dostupný z: ⟨https://cs.wikipedia.org/wiki/Npm⟩.

[6] Facebook. *Introduction to the Messenger Platform* [online]. 2020 [cit. 2020-5-9]. Dostupný z: ⟨https://developers.facebook.com/docs/messenger-platform/introduction⟩.

[7] Wit.ai, Inc. *Build your first app* [online]. 2020 [cit. 2020-5-9]. Dostupný z: ⟨https://wit.ai/docs/quickstart⟩.

[8] SQLite team. *About SQLite* [online]. 2020 [cit. 2020-5-9]. Dostupný z: ⟨https://www.sqlite.org/about.html⟩.

[9] SQLite team. *Write-Ahead Logging* [online]. 2020 [cit. 2020-5-9]. Dostupný z: ⟨https://www.sqlite.org/wal.html⟩.

[10] Wikipedia contributors. *Express.js — Wikipedia, The Free Encyclopedia* [online]. 2020 [cit. 2020-5-9]. Dostupný z: ⟨https://en.wikipedia.org/w/index.php?title=Express.js&oldid=944443884⟩.

[11] MDN contributors. *Express web framework (Node.js/JavaScript)* [online]. 2020 [cit. 2020-5-9]. Dostupný z: ⟨https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs⟩.

[12] Pug team. *Getting Started* [online]. 2020 [cit. 2020-5-9]. Dostupný z: ⟨https://pugjs.org/api/getting-started.html⟩.

[13] Morelli, Brandon. *What is Pug.js (Jade) and How can we use it within a Node.js Web Application?* [online]. 2017 [cit. 2020-5-9]. Dostupný z: ⟨https://codeburst.io/what-is-pug-js-jade-and-how-can-we-use-it-within-a-node-js-web-application-69a092d388eb⟩.

[14] Wikipedia contributors. *Representational state transfer — Wikipedia, The Free Encyclopedia* [online]. 2020 [cit. 2020-5-9]. Dostupný z: ⟨https://en.wikipedia.org/w/index.php?title=Representational_state_transfer&oldid=955607602⟩.

[15] Kim, Cho S. *Data Structures With JavaScript: Tree* [online]. 2015 [cit. 2020-5-9]. Dostupný z: ⟨https://code.tutsplus.com/articles/data-structures-with-javascript-tree--cms-23393⟩.

[16] Facebook. *Quick Replies* [online]. 2020 [cit. 2020-5-9]. Dostupný z: ⟨https://developers.facebook.com/docs/messenger-platform/send-messages/quick-replies⟩.

[17] Wit.ai, Inc. *HTTP API Reference* [online]. 2020 [cit. 2020-5-9]. Dostupný z: ⟨https://wit.ai/docs/http/20170307⟩.

[18] Facebook. *Quick Start Tutorial* [online]. 2020 [cit. 2020-5-9]. Dostupný z: ⟨https://developers.facebook.com/docs/messenger-platform/getting-started/quick-start⟩.

[19] Facebook. *Messenger Platform* [online]. 2020 [cit. 2020-5-9]. Dostupný z: ⟨https://developers.facebook.com/docs/messenger-platform⟩.

[20] Freitas, Eduardo; Bhintade, Madan. *Building Bots with Node.js: Automate workflow and internal communication processes and provide customer service without apps using messaging and interactive bots.* First. Birmingham: Packt Publishing Ltd., 2017. 266 s. ISBN 978-1-78646-545-0.