**Vyšší odborná škola**

**a Střední průmyslová škola elektrotechnická**

**Plzeň, Koterovská 85**

**Dlouhodobá maturitní práce s obhajobou**

Téma:**Vytvoření Chatbota s orientací FAQ ohledně SPŠE Plzeň**

**Autor práce: Duc Anh CAO**

**Třída: 4.F**

**Vedoucí práce: Mgr. Alan KOUKOL**

**Dne: 22. 4. 2022**

**Hodnocení:**

**Vyšší odborná škola**

**a Střední průmyslová škola elektrotechnická**

**Plzeň, Koterovská 85**

**Zadání dlouhodobé maturitní práce**

**Žáci: Richard BASTL, Duc Anh CAO, Lukáš FEJTEK**

**Třída: 4.F**

**Studijní obor: 18-20-M/01 Informační technologie**

**Zaměření: Vývoj aplikací, Správa počítačových sítí**

**Školní rok:** **2021 – 2022**

*Téma práce:* **Vytvoření Chatbota s orientací FAQ ohledně SPŠE Plzeň**

***Pokyny k obsahu a rozsahu práce:***

* Provést rešerši v oblasti Chatbotů a popsat, jaké druhy Chatbotů jsou dnes k dispozici.
* Provést rešerši existujících platforem, které lze využít při implementaci Chatbota.
* Provést vhodný výběr platformy (vycházející z bodu 2.) a implementovat FAQ Chatbota pro potřeby SPŠE, což znamená:
  + Vytvořit mapy oblastí zájmů koncových uživatelů a sestavit nejčastější dotazy, na které bude Chatbot reagovat v kontextu informací o SPŠE.
  + Aplikovat metodu Webscraping pro získávání dat na otázky přímo ze stránek SPŠE.
  + Aplikovat metody strojového učení (NLP – Natural Language Processing) a připravit databázi nejčastějších dotazů, která bude napojena na Chatbot.
  + Navrhnout a implementovat REST API, které propojí frontend Chatbota s backendem (databází a porovnávacím algoritmem). REST API musí obsahovat i autorizační mechanismus pro zabezpečený přístup ke službě Chatbota.
  + Navrhnout a implementovat frontend Chatbota, tedy dialogové okno, které bude sloužit pro komunikaci mezi Chatbotem a koncovým uživatelem.
* Provést testování Chatbota s ohledem na rychlost odezvy (otázka – odpověď) a porozumění Chatbota otázkám (zda je Chatbot schopen správně odpovídat na položené otázky).
* Celý projekt bude realizován pomocí technologií Python, MySQL či PostgreSQL, GitLab a prostředků webového vývoje

**Určení částí tématu zpracovávaných jednotlivými žáky:**

* **Richard Bastl:**
  + Provést rešerši Python knihoven, které budou využívány při vývoji Chatbota.
  + Aplikovat metodu Webscraping pro získávání dat na otázky přímo ze stránek SPŠE Plzeň.
  + Data z webscrapingu poslat do databáze
  + Navrhnout a implementovat frontend Chatbota, teda dialogové okno, které bude sloužit pro komunikaci mezi Chatbotem a koncovým uživatelem.
  + Vytvořit autorizační mechanismus pro REST API pro zabezpečený přístup ke službě Chatbota.
* **Duc Anh Cao:**
  + Připravit databázi nejčastějších dotazů, která bude napojena na Chatbota.
  + Navrhnout a implementovat REST API, které propojí frontend Chatbota s backendem.
  + Připravit skripty pro Rasa Actions Server (custom actions).
  + Provést testování Chatbota s ohledem na rychlost odezvy (otázka – odpověď) a porozumění Chatbota otázkám (zda je Chatbot schopen správně odpovídat na položené otázky).
* **Lukáš Fejtek:**
  + Provést rešerši v oblasti Chatbotů a popsat, jaké druhy Chatbotů jsou dnes k dispozici.
  + Vytvořit mapy oblastí zájmů koncových uživatelů a sestavit nejčastější dotazy, na které bude Chatbot reagovat v kontextu informací na SPŠE.
  + Aplikovat metody strojového učení (NLP – Natural Language Processing)

***Požadavek na počet vyhotovení maturitní práce:*** *2 výtisky*

*Termín odevzdání:* ***22. dubna 2022***

*Čas obhajoby:* ***15 minut***

Vedoucí práce: **Mgr. Alan KOUKOL**

Projednáno v **katedře VTT** a schváleno ředitelkou školy.

V Plzni dne: 30. října 2021 Ing. Naděžda Mauleová, MBA, v.r.

*ředitelka školy*

**Anotace**

Cílem této maturitní práce je vytvořit Chatbota, který dokáže odpovídat na často kladené otázky z pohledu uchazeče o studium nebo zákonných zástupců, které své svěřence hlásí na SPŠE Plzeň. Chatbot by se měl naučit porozumět dané otázce, kterou uživatel odešle přes chatovací okénko, a dokázat v kontextu na ní odpovědět. Při vývoji budeme používat technologie Python, SQLite, GitLab, HTML, CSS a Javascript.

„Prohlašuji, že jsem tuto práci vypracoval samostatně a použil(a) literárních pramenů a informací, které cituji a uvádím v seznamu použité literatury a zdrojů informací.“

„Souhlasím s využitím mé práce učiteli VOŠ a SPŠE Plzeň k výuce.“

V Plzni dne: …..................... Podpis: ….....................................

**Obsah**

[Úvod 5](#_Toc101497834)

[Návrh celé práce 6](#_Toc101497835)

[1 Rešerše 7](#_Toc101497836)

[1.1 Druhy Chatbotů 7](#_Toc101497837)

[1.2 Python knihovny 9](#_Toc101497838)

[2 Otázky, odpovědi a oblasti zájmů 10](#_Toc101497839)

[3 Příprava databáze a její vytvoření 12](#_Toc101497840)

[4 Vývoj Chatbota 15](#_Toc101497841)

[4.1 Připravení struktury adresáře MP 15](#_Toc101497842)

[4.2 Výběr platformy a její následná instalace 15](#_Toc101497843)

[4.3 Budování struktury Rasy 16](#_Toc101497844)

[4.3.1 NLU data 16](#_Toc101497845)

[4.3.2 Stories 16](#_Toc101497846)

[4.3.3 Rules 17](#_Toc101497847)

[4.3.4 Domain 18](#_Toc101497848)

[4.3.5 Config 20](#_Toc101497849)

[4.3.6 Endpoints 22](#_Toc101497850)

[4.3.7 Natrénování Rasy a Action Server 23](#_Toc101497851)

[4.3.8 Custom Actions 23](#_Toc101497852)

[4.4 REST API 25](#_Toc101497853)

[4.4.1 Frontend Chatbota 26](#_Toc101497854)

[4.4.2 HTTP Request 28](#_Toc101497855)

[4.5 Testování 30](#_Toc101497856)

[Závěr 32](#_Toc101497857)

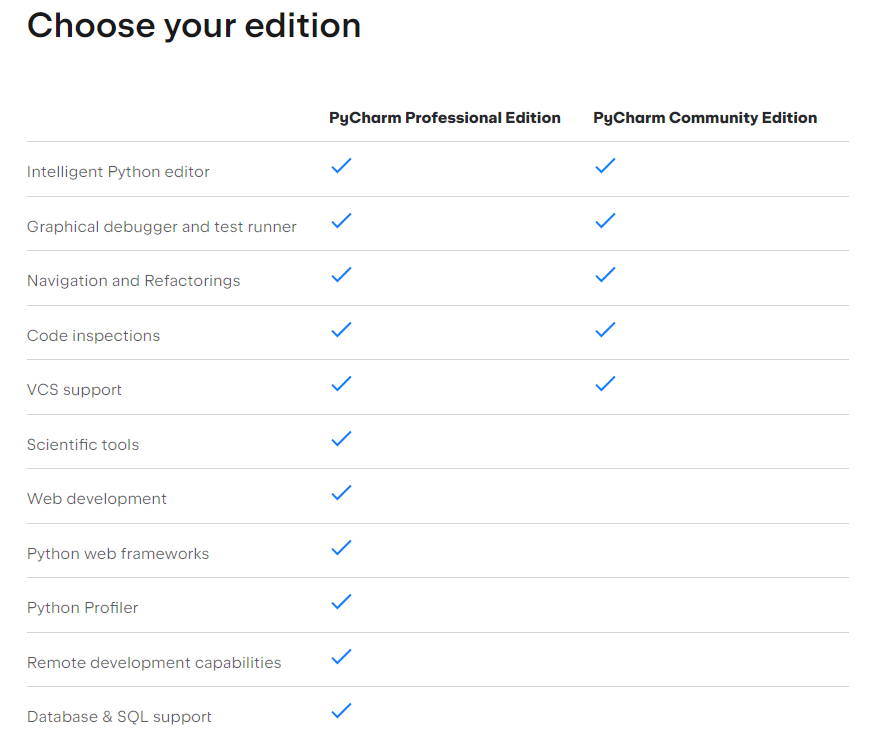
[Použitá Literatura 33](#_Toc101497858)

[Seznam obrázků a tabulek 35](#_Toc101497859)

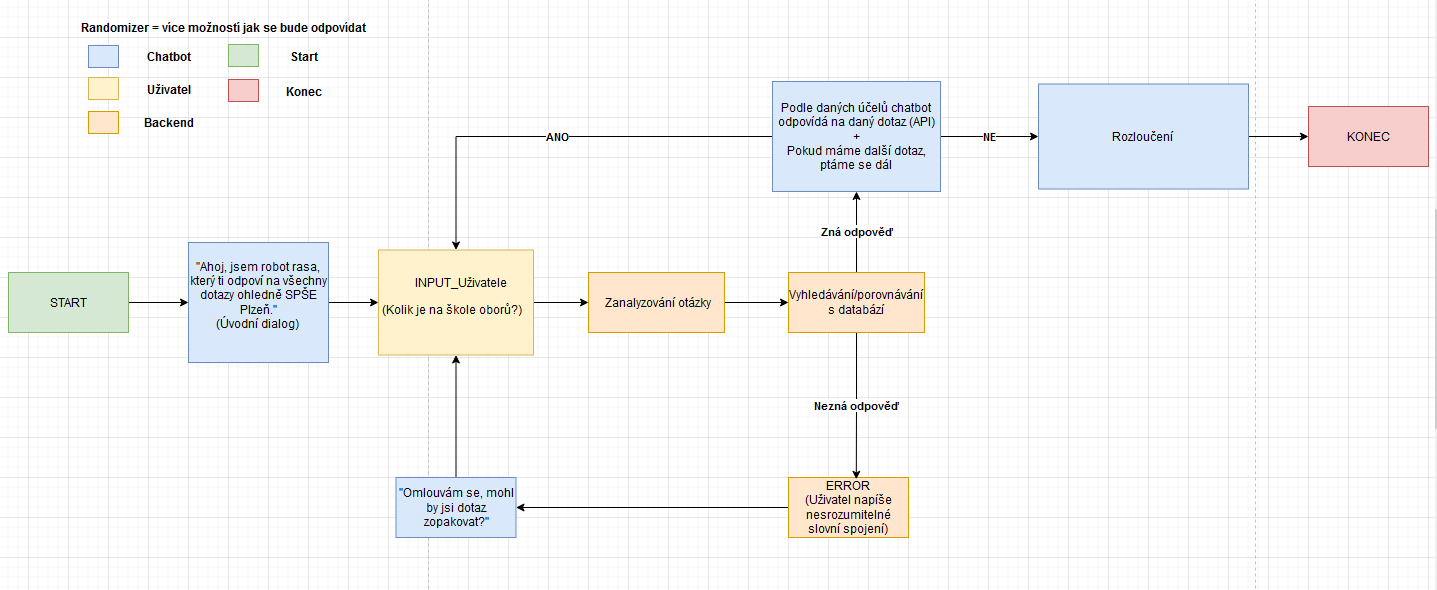
# Úvod

Výběrem tohoto tématu pro maturitní práci (dále jen MP) jsme již začali v roce 2021 ve skupině tří kluků. Vybrali jsme si toto téma, protože nás zaujal název práce, a protože ve druhém ročníku nás velice zaujal předmět IKT, přesněji Business Intelligence. Celá práce se dělí na základní části a to: rešerše Chatbotů a python knihoven, určit si jaké jsou nejvíce tázané otázky a na ně odpovědi, výběr platformy a poté návrh celé MP.

Pro vývoj jsme využívali metodu agilního řízení neboli scrum, což je rozdělení projektu na jednotlivé části, které se řeší poté samostatně. Práce na každé časti probíhají v krátkých intervalech, tzv. sprintech. Na konci každého sprintu se vyhodnotí výsledek, kterou jsme každý ze členů vytvořili. Tím jsme také dokázali rychle odhalit potíže a reagovat na ně včas. Pokud někdo ze členů měl nějaký problém, jako tým jsme daný problém řešili spolu. Obvykle jsme práci dělali současně neboli při vývoji jsme často byly spolu na video hovorech, při kterých každý z nás sdílel obrazovku. K uskutečnění práce jsme museli nainstalovat PyCharm Professional Edition, vývojové prostředí pro Python, kvůli větší podpoře nástrojů k realizaci MP na rozdíl od Community Edition. K získání Professional Edition jsme využili našich ISIC karet. Poté Framework Rasa, ve které jsou obsaženy všechny důležité knihovny, SQLiteStudio a platformu Postman pro testování našich API requestů.

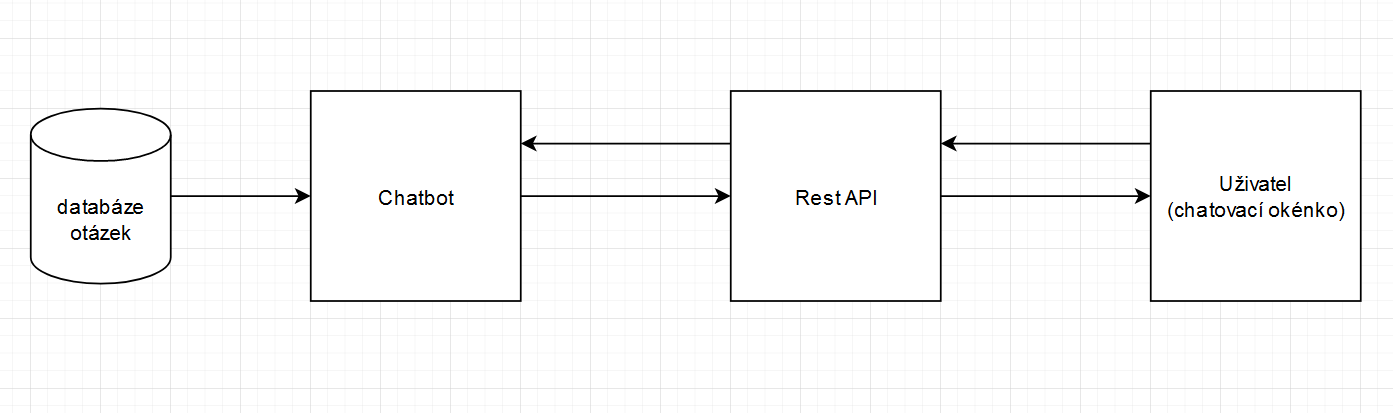
 Obrázek 1 – výstřižek z oficiálních stránek (viz. https://www.jetbrains.com/pycharm/features/)

## Návrh celé práce

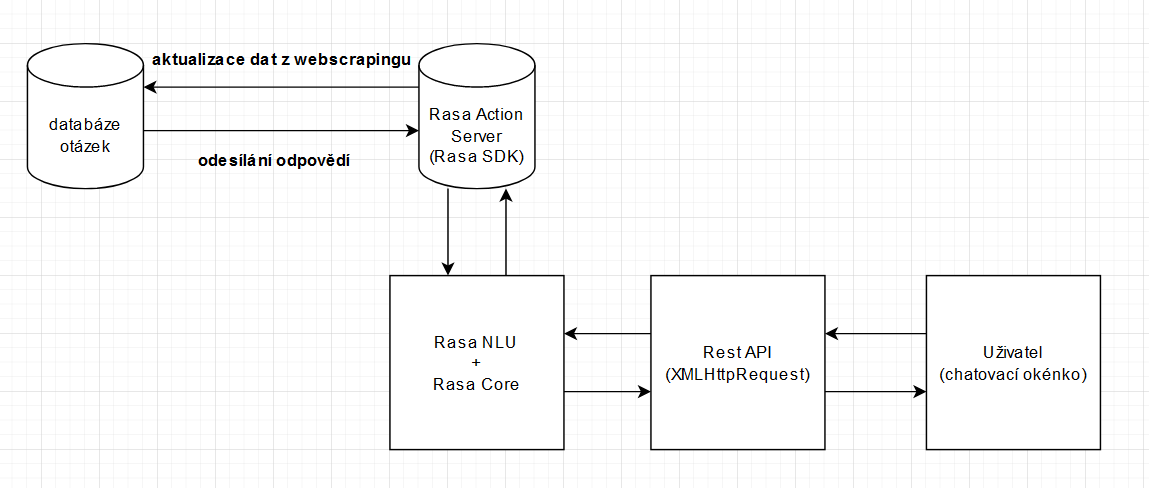


Obrázek 2 – jak by měl vypadat workflow

Návrh práce je důležitým aspektem při vývoji MP. Díky němu si dokážeme představit, jak by naše výsledná práce mohla vypadat. Také nám pomůže lépe určit každému členu práci při sprintech.



Obrázek 3 – původní návrh MP



Obrázek 4 – finální návrh MP

# Rešerše

## Druhy Chatbotů

Nejdříve si musíme vysvětlit, co to vůbec chatbot je. Chatbot je zkratkou pro „chat robot“. Jde o počítačový program, který simuluje lidskou konverzaci nebo chatování skrze umělou inteligenci. Typický chatbot komunikuje s člověkem, ale může jít také o komunikaci dvou chatbotů mezi sebou. Využití chatbotů najdeme v aplikacích, call centrech, různých internetových hrách, ale i v zákaznických podporách e-shopů. Chat bot se obvykle používá pro konkrétní účel, nikoliv pro celý rozsah lidské komunikace.

Typy chatbotů:

**Rule-based chatbot (decision-tree bots)**

Chatboti na základě pravidel, označováni také jako roboti rozhodovacího stromu. Jsou jazykově založeni. Musí mít předem naprogramované otázky a odpovědi (jelikož nejsou poháněni umělou inteligencí).

Výhody:

Jsou skvělí pro menší čísla a jednoduché dotazy, jako je rezervace stolu v restauraci nebo při požadavku o otevírací době. Nepotřebují rozsáhlé školení, které zrychluje a zkomplikuje proces implementace. Čím lépe bude předdefinována struktura a odpovědi tím snadněji budou schopni odpovídat a lépe se chovat.

Nevýhody:

Nemohou zachytit překlepy, což znamená, že v některých případech nepochopí uživatele, a to může způsobit frustraci. Interakce s nimi se zdá spíše robotická než konverzační. Nemohou se učit sami, takže kterékoliv zlepšení je třeba udělat ručně.

**AI Chatbot s využitím NLP**

AI chatbot je chatbot využívající zpracování přirozeného jazyka (NLP). Takže na rozdíl od chatbota založeného na pravidlech nebude k odpovědi používat klíčová slova, ale pokusí se pochopit záměr hosta, což znamená, co host chce. Čím více komunikuje s hosty, tím lépe bude chápat záměr a tím lépe bude odpovídat na požadavky hostů.

Výhody:

Chatbot rozumí překlepům a gramatickým chybám, takže je stále schopen odpovědět na danou otázku. Neustále se zlepšuje bez další pomoci. Rozhovor s AI se zdá mnohem přirozenější a lidštější.

Nevýhody:

Chatboti s AI procházejí procesem učení, což ztěžuje a prodlužuje jejich proces implementace. Když se chatbot cokoliv naučí špatně, chvíli trvá, než se odnaučí a poté znovu naučí správné chování. Bez předem definovaných struktur jsou konverzace vedené chatbotem AI méně předvídatelné.

**Hybrid**

Jedná se o spojení AI a Rule-based chatbotů. Je poháněn umělou inteligence a zároveň používá strukturu založenou na pravidlech. To znamená, že hostovi položí následné otázky, ale pomocí AI pochopí záměr hosta, aby mohl přeskočit nadbytečné otázky.

Výhody:

Implementace není tak dlouhá a komplikovaná jako u AI, protože používá předem definované struktury a odpovědi. Protože je však poháněn umělou inteligence, chatbot se neustále zlepšuje, aby pochopil záměr hosta. Rozhovory s hybridním modelem budou stále cítit konverzační a přirozené.

Nevýhody:

Jsou stejné jako u samotných typů chatbotů.

**Voice bots**

Hlasem aktivované chatboty jsou ti, kteří mohou komunikovat a komunikovat prostřednictvím hlasu. Jsou schopni přijmout příkaz ústní nebo písemnou formou. Naprogramovány jsou tak, aby odpověděly hlasem. To lze dále rozdělit do dvou typů – jeden, který odezvuje prostřednictvím textu a hlasu dohromady. A ten druhý, který odpovědí pouze hlasem.

Typy hlasových chatbotů:

**Hlasoví + textoví roboti (hybridní model podpory hlasu + chatu):**

Jedná se o textové roboty s vrstvou hlasu. Vstupním formulářem je řeč i text.

**Roboti pouze pro hlas (hlasem ovládaná zařízení)**

Tito roboti zvládnou vše od jednoduchých úkolů, jako je nastavení budíků, přehrávání hudby až po složitější úkoly, jako je ovládání miniaplikací a proměna vašeho domu v chytrou domácnost.

**NLP** je nástroj pro počítače analyzovat, pochopit a odvozovat význam z přirozeného jazyka inteligentním a užitečným způsobem. To jde daleko za nejnovější vyvinuté chatboty a chytré virtuální asistenty. Algoritmy zpracování přirozeného jazyka jsou ve skutečnosti všude od vyhledávání, online překladu, filtrů spamu a kontroly pravopisu.

## Python knihovny

**Rasa**

Rasa je open source framework strojového učení pro automatizované textové a hlasové konverzace. Porozumí zprávám, udržuje konverzace a dokáže se připojit ke kanálům pro zasílání zpráv a rozhraním API. Dále umožňuje natrénování modelu či přidání vlastních akcí.

**spaCy**

Knihovna vytvořená přímo pro vývojáře NLP (Natural language processing), která umí pojmout a rozumět. Může být využitá k NLU (Natural language understanding), kde používá deep learning. Knihovna používá systém tokenizace, kde rozdělí odpovědi uživatelů do tokenů (převážně jedno slovo), a na základě těchto tokenů vyhodnocuje odpovědi a hledá mezi tokeny případnou spojitost.

**ChatterBot**

ChatterBot je knihovna Pythonu vytvořená na jednoduché vyrábění softwarů, které mohou vést konverzaci. Používá výběr machine learning algoritmů ke generování různých typů odpovědí. To vede k vytvoření chatbotu a vedení konverzace s uživateli.

**NLKT**

Natural Language Tool Kit je sada knihoven a programů k tvoření programů v Pythonu. Toolkit je vybaven programy na podporu rozpoznávání jazyka jako celku spolu s programy sloužícími k tokenizaci a rozpoznávání odpovědí po částech, hledáním spojitosti a poté propojením s odpověďmi.

**TextBlob**

Knihovna ke zpracování textových dat napsaná v Pythonu. Knihovna poskytuje jednoduché API s NLP úkoly, jako je označování části řeči, extrakce podstatných jmen, analýza sentimentu a další. Knihovna se zaměřuje na poskytování přístupu k běžným operacím zpracování textu prostřednictvím známého rozhraní.

**DeepPavlov**

DeepPavlov je open-source knihovna AI konverzace zakládající na TensorFlow a Keras. Má komplexní a flexibilní nástroje, které umožňují vývojářům a výzkumníkům NLP vytvářet konverzační dovednosti připravené pro produkci a komplexní konverzační asistenty s více dovednostmi.

**PyNLPl**

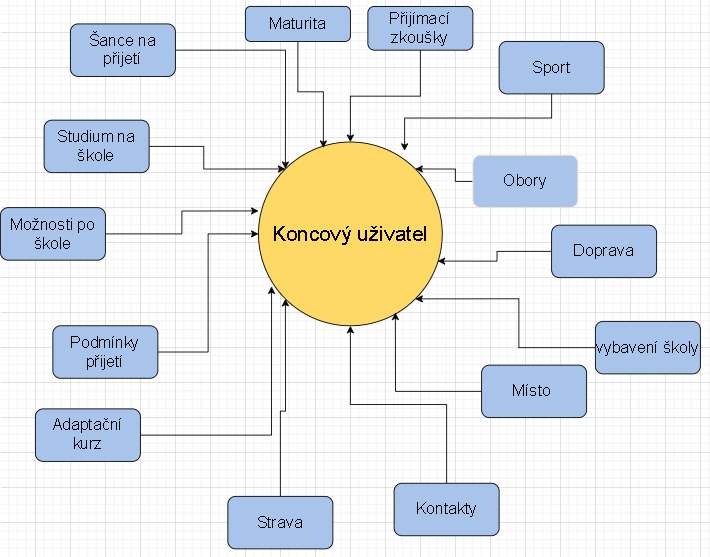
Knihovna, vyslovována jako ‘pineapple‘, slouží k NLP. Lze ji využít pro základní úkoly, jako je extrakce n-gramů a seznamů frekvencí, a pro vytvoření jednoduchého jazykového modelu. Knihovna je rozdělena do několika balíčků a modulů.

# Otázky, odpovědi a oblasti zájmů

**Oblasti zájmů koncových uživatelů:**

První věc, kterou jsme si určili je, kdo je vůbec koncový uživatel pro našeho chatbota. Zvolili jsme, že to bude student, který vychází základní školu nebo už vyšel a zajímá se o studium na škole SPŠE Plzeň. Po případě rodič nebo zákonný zástupce dítěte (žáka). Abychom byli schopni poté sestavit otázky, musíme se zaměřit na zájmy žáků a jejich rodičů ohledně školy.

Zde je diagram oblastí zájmů koncových uživatelů:



Obrázek 5 – diagram oblasti zájmů

Zájmů je určitě více, ale tyto jsou nejčastější, které jsme vyhodnotili.

**FAQ:**

Frequuently Asked Questions neboli často kladené dotazy. Díky určení zájmů je lehčí si sestavit otázky. Jelikož jsme nedávno byli ve stejné situaci jako naši koncový uživatelé, sestavili jsme si počet dotazů, se kterýma nám pomohli na sekretariátu školy, kamarádi nebo příbuzní.

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek 6 – ukázka otázek

**Odpovědi:**

Zodpovědět spoustu otázek není jednoduché, protože na ně musí být co nejlépe a správně odpovězeno. Abychom tedy měli oficiální odpovědi od školy, poprosili jsme na sekretariátu o zodpovězení všech otázek.

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek 7 – ukázka odpovědí

# Příprava databáze a její vytvoření

Při navrhování databáze jsme využili nástroje Visual Paradigm Community Edition, v něm jsme si vybrali diagram vztahů entit. Entity zde představují tabulky, v nichž jsou Columms, což jsou sloupce, a ty mají nějaké specifikace. V těchto specifikací poté určujeme např. datový typ sloupce, délku slova/čísla, zda je sloupec primární klíč nebo nenulový.

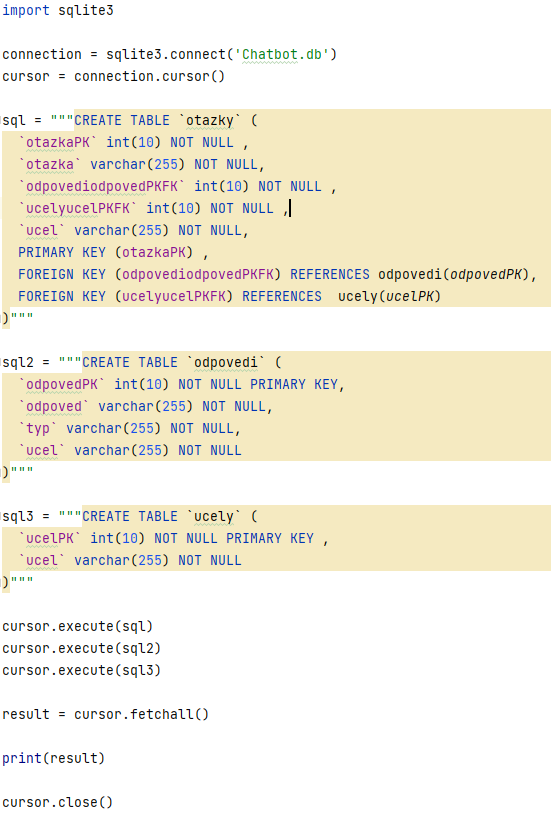
Na výběr z databází jsme měli možnost mezi MySQL, MariaDB nebo SQLite. Pro vývoj naší MP jsme si vybrali SQLite, protože by bylo pro nás jednodušší posílat mezi námi databázi přes GitLab, neboť SQLite je jen pouze soubor, a ne celá aplikace/software. To znamená, že nemusíme zbytečně stahovat a instalovat celý software.

Pro vytvoření samotné databáze byl vytvořen skript v pythonu pomocí knihovny sqlite3 a programovacího jazyka SQL, anglicky Structured Query Language. SQL je tedy strukturovaný dotazovací jazyk, který je používán pro práci s daty, které jsou v relačních databázích. Pro práci s daty máme na mysli uchovávání, manipulaci nebo získávání dat.

Použité příkazy pro práci s databází

|  |  |
| --- | --- |
| CREATE | Vytvoření např. databáze nebo tabulek. |
| DELETE | Smaže řádku v tabulce. |
| FOREIGN KEY | Klíč, podle kterého můžeme spojit dvě tabulky spolu. |
| FROM | Specifikujeme, zda vybíráme nebo mažeme data z … |
| INSERT INTO | Vložení řádky do tabulky. |
| NOT NULL | Určujeme, že sloupec nemůže obsahovat null hodnotu. |
| PRIMARY KEY | Určujeme, že sloupec má unikátní identifikátor (např. celé číslo). |
| SELECT | Vybereme data z databáze. |
| UPDATE | Aktualizuje existující řádky v databázi. |
| WHERE | Díky ní můžeme filtrovat dané záznamy, podle zadané podmínky. |

Tabulka 1 – příkazy SQL



Obrázek 8 – skript pro vytvoření databáze

# Vývoj Chatbota

## Připravení struktury adresáře MP

Předtím než vůbec spustíme PyCharm, musíme se nejdříve ujistit, zda vůbec máme stáhnutý Python. Pro MP jsme si vybrali verzi 3.8.8. Pokud se ptáte, proč jsme jí vybrali, tak je to kvůli tomu, že někde na fóru psal jeden z vývojářů, že je to nejlepší verze pro vyvíjení chatbota v Rase.

Po instalaci Pythonu můžeme spustit PyCharm. Po spuštění by se nám mělo objevit malé okénko, v ní můžeme vidět na levém sloupci seznam projektů, nastavení a pluginy. Nahoře jsou tlačítka pro vytvoření nového projektu, otevření projektu a získání projektu ze vzdáleného repositáře (např. GitHub). Pro nás ale bude zajímat tlačítko vytvoření nového projektu.

Po spuštění tlačítka bychom měli vidět výběr v čem bude projekt realizován. Samozřejmě vybereme Pure Python. Poté už stačí jen určit místo na disku, kde se bude projekt ukládat, nastavit si environment a stačí zmáčknout tlačítko Create, pro vytvoření projektu. Environment můžeme chápat nějaké virtuální prostředí, kde se uchovávají naše zdrojové kódy, nainstalované knihovny a skripty, které jsou odděleny od ostatních virtuálních prostředí (zjednodušeně jiné projekty).

První, co jsme udělali, bylo vytvoření adresáře docs, ve kterém se uchovávají dokumentace a rešerše. Tuto celou strukturu budeme nadále používat i pro odesílání zdrojových souborů na školní repositář GitLab.

## Výběr platformy a její následná instalace

Po vytvoření databáze a vytvoření projektu v PyCharmu, jsme museli vybrat jakou platformu budeme využívat pro našeho chatbota. Po neustálých zkoušení knihoven z rešerší nám pomohl Marek Lukšík. Doporučil nám Rasu. Po přečtení dokumentace a recenze, jsme zjistili, že je to jakýsi framework, který využívá více knihoven, jako např. spaCy. Nás tento framework velmi nadchl, a proto jsme si jí vybraly.

Abychom mohli nainstalovat Rasu a vyzkoušet si jí, musíme v PyCharmu do terminálu napsat následné příkazy:

*pip install --upgrade pip*

*pip install rasa=2.8.25*

První příkaz znamená aktualizace správce balíčků pip, přes který instalujeme knihovny.

Druhý příkazem instalujeme Rasu se specifickou verzí 2.8.25. V době první instalace Rasy již existovala verze 3, ale ta měla potíže, a proto jsme si vybrali naší verzi 2. Pro kontrolu, zda se nám opravdu nainstalovala přesná verze Rasy, stačí do terminálu napsat:

*rasa --version*

Měl by se nám objevit výpis do konzole verze Rasy (samotná Rasa, SDK a X).

## Budování struktury Rasy

### NLU data

NLU Training data (Natural Language Understanding) neboli v překladu se to nazývá porozumění přirozenému jazyku. Cílem NLU je extrahovat strukturované informace z uživatelských zpráv. Trénovací data NLU se skládají z ukázkových uživatelských promluv kategorizovaných podle intentů. Soubor, který jsme tedy vytvořili, se jmenuje nlu.yml.

Trénovací data NLU jsme definovali v klíči nlu. Každý intent se rovná jedné otázce. Každý intent jsme si pojmenovali bez háčků, čárek, mezer, diakritiky. Ke každé otázce jsme přidali více způsobů formulace otázky a vypsali je do examples (příkladů). NLU jsou seskupeny podle intentu a uvedeny v klíči examples. Uvedli jsme jeden příklad na řádek následujícím způsobem:

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek 9 - ukázka nlu.yml

### Stories

V překladu příběhy jsou typem trénovacích dat, která se používají k trénování modelu správy dialogů asistenta. Příběhy lze použít k trénování modelů, které jsou schopné zobecnit na neviditelné cesty konverzace. Vytvořili jsme si tedy soubor z názvem stories.yml. Stories jsou konverzace mezi uživatelem a asistentem AI, převedená do specifického formátu, kde jsou vstupy uživatele vyjádřeny jako intenty, zatímco odpovědi a akce asistenta jsou vyjádřeny jako názvy akcí.

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek 10 – ukázka stories.yml

**Uživatelské zprávy:**

Při psaní příběhů jsme se nemuseli zabývat konkrétním obsahem zpráv, které uživatelé odesílají. Místo toho jsme využili výstup z kanálu NLU, který umožňuje použít pouze kombinaci intentů k odkazování na všechny možné zprávy, které mohou uživatelé odeslat, aby znamenaly totéž.

### Rules

V překladu pravidla jsou typem trénovacích dat, která se používají k trénování modelu správy dialogů asistenta. Pravidla popisují krátké konverzace, které by měly vždy sledovat stejnou cestu. Pravidla jsou skvělá pro zpracování malých specifických konverzačních vzorců, ale na rozdíl od příběhů nemají pravidla sílu zobecnit na neviditelné konverzační cesty. Vytvořili jsme si tedy soubor s názvem rules.yml. Nechtěli jsme vytvořit hodně pravidel, protože by nám pak mohl blbnout chatbot. První věc, co musíte do souboru přidat jsou policies a to konkrétně RulePolicy. Bez nich by nám žádná pravidla nefungovali. Rules jsme definovali v klíči rules. První pravidlo, co jsme tedy vytvořili je pro pozdrav, aby kdykoliv, co je začátek konverzace, odpovídalo pozdravem a nepoužilo se např: Intent rozloučení. Můžeme si pravidlo pojmenovat, jak chceme, ale v našem případě Pozdrav.

A to dělá conversation\_start: true. Poté jsou jen steps pro dané pravidlo, a to funguje stejně jako ve stories. Jelikož jsou pozdravy a rozloučen v češtině podobné, je tohle pravidlo velice důležité.

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek 11 – ukázka rules.yml

Další pravidlo jsme vytvořili pro rozloučení na stejný způsob, jen pokaždý,

co je konverzace na konci. Což způsobuje conversation\_start:false.

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek 12 – ukázka rules.yml

### Domain

Doména definuje vesmír, ve kterém váš asistent působí. Určuje v našem případě intenty, odpovědi a akce, o kterých by měl váš robot vědět. Definuje také konfiguraci pro konverzační relace. Vytvořili jsme si tedy soubor s názvem domain.yml. Klíče intentů jsme vypsali v souboru domény, takže obsahuje seznam všech intentů použitých v datech NLU a trénovacích datech konverzací.



Obrázek 13 – ukázka souboru domain.yml (intenty)

Hned po seznamu intentů jsme přidali odpovědi na všechny z nich.

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek 14 – ukázka souboru domain.yml (odpovědi)

Konfigurace relace#

Konverzační relace představuje dialog mezi asistentem a uživatelem.

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek 15 – ukázka souboru domain (konfigurace relace)

Znamená to, že pokud uživatel odešle svou první zprávu po 60 minutách nečinnosti, spustí se nová konverzační relace.

### Config

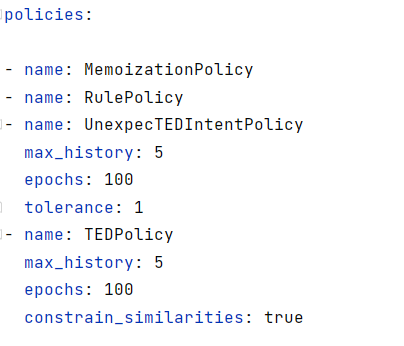
**Konfigurace modelu:**

Konfigurační soubor definuje komponenty a zásady, které bude váš model používat k vytváření předpovědí na základě vstupu uživatele. Na každém kroku je každá zásada definovaná ve konfiguraci předpovídat další akci s určitou úrovní spolehlivosti.Jde tedy o konfiguraci trénovaného modelu. Policies, která předpovídá s nejvyšší jistotou, rozhoduje o dalším postupu asistenta.



Obrázek 16 – nastavení config souboru do anglického jazyka

Používáme anglický jazyk, protože čeština na konfigurace není podporována.



Obrázek 17 – ukázka používaných Policies

**Policies:**

Asistent se pomocí zásad rozhoduje, kterou akci provést v každém kroku konverzace. Existují zásady strojového učení a zásady založené na pravidlech, které náš asistent používá v tandemu.

**Priorita zásad:**

V případě, že dvě zásady předpovídají se stejnou jistotou (například zásady zapamatování a pravidla mohou předpovídat s jistotou 1), je zvažována priorita zásad. Zásady Rasa Open Source mají výchozí priority, které jsou nastaveny tak, aby zajistily očekávaný výsledek v případě remízy. Vypadají takto, kde vyšší čísla mají vyšší prioritu:

6 - RulePolicy

3 - MemoizationPolicy nebo AugmentedMemoizationPolicy

2 - UnexpecTEDIntentPolicy

1 - TEDPolicy

Tohle jsou všechna Policies, které využíváme.

**Policies TED**

Policies TED (Transformer Embedding Dialogue) jsou víceúčelová architektura pro predikci dalších akcí a rozpoznávání entit. Architektura se skládá z několika transformátorových kodérů, které jsou sdíleny pro obě úlohy. Pro další predikci akce je výstup kodéru dialogu a popisky akcí systému vloženy do jednoho sémantického vektorového prostoru. Používáme bodovou ztrátu produktu, abychom maximalizovali podobnost s cílovým štítkem a minimalizovali podobnosti s negativními vzorky.

**UnexpecTEDIntentPolicy:**

Nám pomáhá kontrolovat konverzace a také umožňuje robotovi reagovat na nepravděpodobné obraty uživatelů. Jedná se o pomocnou policies, která se používá pouze ve spojení s alespoň jednou další politikou, protože jediná akce, kterou může spustit, je zvláštní action\_unlikely\_intent akce.

**RulePolicy:**

Je zásada, která zpracovává části konverzace, které sledují pevné chování (např. obchodní logika). Vytváří předpovědi na základě všech pravidel, která máte ve svých trénovacích datech.

**MemoizationPolicy:**

Pamatuje si stories z tréninkových dat. Zkontroluje, jestli aktuální konverzace odpovídá článkům v souboru stories.yml. Pokud ano, předpoví další akci z odpovídajících příběhů s jistotou 1,0. Pokud není nalezena žádná odpovídající konverzace, zásady předpovídají Žádná s jistotou 0,0.

**Epoch:**

Jedna epocha se rovná jednomu přihrávce dopředu a jednomu zpětnému průchodu ze všech tréninkových příkladů. Někdy model potřebuje více epoch, aby se správně naučil. V našem případě stačí 100.

### Endpoints

Vytvořili jsme si soubor endpoints.yml. Je tam adresa, která slouží jako endpoint, při zapínání severu. Hodí se to na spoustu dalších věcí pro práci.

Tracker store:

Použili jsme tracker store pro ukládání naší konverzace s chatbotem po zapnutí serveru.

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek 18 – ukázka endpoints.yml

### Natrénování Rasy a Action Server

Rasa používá YAML soubory jako způsob pro správu. Trénujeme příkazem do konzole *rasa train.* Trénují se takzvané modely, které se uchovávají do adresáře models. Můžeme si všimnout, že každý soubor má ve jméně přesný datum a čas v kolik byl model natrénován. (přesně na minutu a sekundu).

Abychom modli zapnout server musíme napsat do konzole *rasa run*, ale protože v budoucnu budeme používat REST API a jiné nástroje pro vyzkoušení API, budeme používat příkaz:

*rasa run -m models --enable-api --cors "\*" –debug*

-m models znamená vybraný model, enable-api je pro povolení rozhraní API, cors "\*" je vyžadován kvůli bezpečnému přenosu dat a k zabránění křížené šíření zdrojů.

### Custom Actions

#### Web Scraping

Web scraping je proces sběru strukturovaných webových dat. Nejčastěji se web scraping používá pro monitorování cen, sledování zpráv, generování leadů (lákání potenciálních zákazníků) či průzkumu trhu. Hlavní využití spočívá v tom, že dokážeme brát data ze specifické stránky v reálném čase. Nemusíme tedy nějakou případnou hodnotu ručně přepisovat. K tomu nám mohou práci ulehčit web scraping nástroje. Pro naši práci jsme vybrali nástroj Selenium, který lze použít s programovacím jazykem Python. Selenium API využívá protokol WebDriver pro přístup k webovému prohlížeči, jako je například Chrome, Firefox nebo Safari.

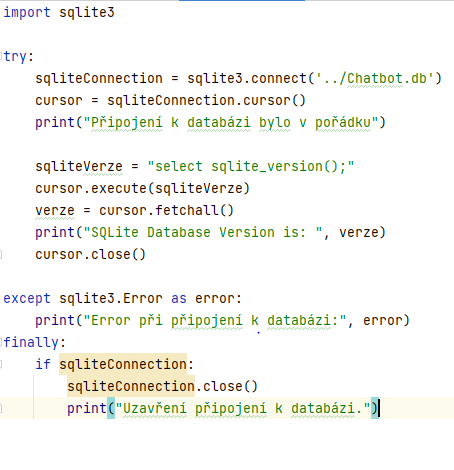
Po importu Selenia do našeho chatbota bylo potřeba definovat cestu k webdriveru. Při práci s Chromem, který jsme použili se jedná specificky o Chromedriver. Verzi Chromedriveru bylo třeba stáhnout podle verze Google Chromu, kterou máme nainstalovanou. Pokud nevíme, jakou verzi má náš Chrome, stačí jít do *Nápovědy* a poté do sekce *O aplikaci Chrome*. Dále bylo třeba definovat spuštění prohlížeče, k čemuž jsme použili cestu k Chromedriveru. Prohlížeči po spuštění musí být přidělena webová stránka na které bude prováděn web scraping, a to pomocí URL adresy. K vybrání specifických dat z webu jsme k orientaci použili XPATH, kterou jsme pro specifické elementy nalezli v debugovém nástroji Chromu. Jako odpověď se vrátilo pole, které bylo třeba projít pomocí dvojitého cyklu FOR, abychom dostali požadovanou odpověď do konzole. Níže můžeme vidět ukázku:



Obrázek 19 – ukázka web scrapingu

#### Připojení k databázi

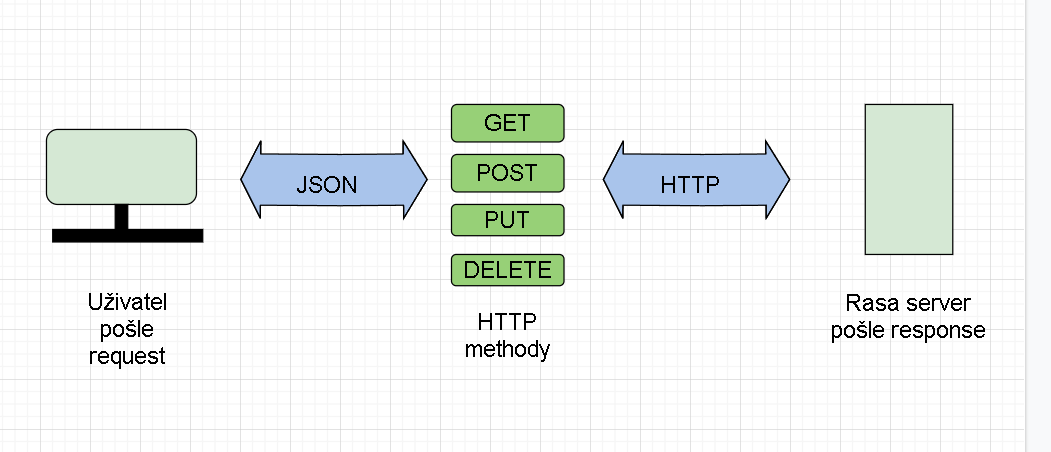
Pro připojení k databáze jsme využili metodu connect() z knihovny sqlite3. Je to vlastně jakýsi connector. Dovnitř metody se píše umístění a název databaze. Níže můžeme vidět ukázku našeho connectoru:



Obrázek 20 – connector pro databázi

## REST API

REST API je rozhraní pro programování aplikací (API nebo webové rozhraní API), které odpovídá omezením architektonického stylu REST a umožňuje interakci s webovými službami RESTful. Rozhraní REST API komunikují prostřednictvím požadavků HTTP k provádění standardních databázových funkcí, jako je vytváření, čtení, aktualizace a odstraňování záznamů (označovaných také jako CRUD) v rámci prostředku. Například rozhraní REST API by použilo požadavek GET k načtení záznamu, požadavek POST k jeho vytvoření, požadavek PUT k aktualizaci záznamu a požadavek DELETE k jeho odstranění. Všechny metody HTTP lze použít ve volání rozhraní API. Dobře navržené rozhraní REST API je podobné webu spuštěnému ve webovém prohlížeči s integrovanou funkcí HTTP. Rozhraní HTTP (REST) API se může použít k interakci se spuštěným serverem Rasa Open Source. Pomocí rozhraní API můžete trénovat modely, odesílat zprávy, spouštět testy a další.



Obrázek 21 – schéma pro REST API

### Frontend Chatbota

Přístup k chatbotu je přes label v levé spodní části webové stránky. Otevírání chatovacího okénka je zajištěno pomocí checkboxu, který je pro běžné uživatele neviditelný. Když je checkbox zaškrtnutý, okénko se otevře. Otevřené chatovací okno je rozdělené na tři části. Hlavička, kde se nachází jméno chatbota. Prostřední část, která slouží k chatování. A také spodní část, která slouží k psaní dotazů a odesílání zpráv pomocí tlačítka. K uzavření chatovacího okénka opět stačí kliknout na label, čímž se zruší zaškrtnutí checkboxu.

Využité tagy a deklarace při vývoji frontendu chatbota:

|  |  |
| --- | --- |
| <!DOCTYPE html> | Deklarace dokumentu HTML verze 5 |
| <html> </html> | Párový tag označující začátek a konec HTML souboru |
| <head> </head> | Párový tag označující začátek a konec hlavičky |
| <title> </title> | Párový tag pojmenovávající stránku |
| <style> </style> | Párový tag sloužící k zápisu CSS přímo do HTML dokumentu |
| <body </body> | Párový tag sloužící k zápisu hlavního obsahu stránky |
| <div> </div> | Párový tag sloužící k oddělení určité části k použití CSS či JavaScriptu |
| <b> </b> | Párový tag vytvářející tlustý text |
| <label> </label> | Párový tag sloužící k přidání textu např. k checkboxu a zároveň zvětšení hitboxu |
| <input> | Tag vytvářející vstupní pole kde uživatel může zadat data podle typu |
| <id> | Atribut specifikující unikátní HTML element |
| <class> | Atribut specifikující třídu pro HTML elementy |
| <button> </button> | Párový tag vytvářející stlačitelné tlačítko |
| {background: } | Změna pozadí specifickou deklarací  {…} - CSS |
| {position: } | Vlastnost specifikující typ poziční metody použité na element |
| {right/left/bottom/top: } | Vzdálenost od použité strany |
| {cursor: } | Změna kurzoru myši podle typu |
| {font-size: } | Změna velikosti písma |
| {color: } | Změna barvy textu |
| {width: } | Šířka určité části |
| {height: } | Výška určité části |
| {display: } | Vlastnost určující zobrazení (typ vykreslovacího rámečku) |
| {justify-content: } | Zarovná obsah v kontejneru horizontálně |
| {align-items: } | Zarovná obsah v kontejneru vertikálně |
| {border: } | Vytvoří ohraničení určité části |
| {border-radius: } | Vytvoří zaoblený okraj |
| {opacity: } | Vytvoří průhlednost určité části |
| {transition: } | Změna při hoveru nad elementem |
| {overflow: } | Vytvoří posuvník pro moc velké části |
| {min-height: } | Nastavuje minimální výšku |
| {margin: } | Posunutí od okraje stránky ze všech stran (lze vybrat specifickou) |
| {box-shadow: } | Přidá stín k určité části |
| {resize: } | Umožnuje dynamickou změnu velikosti určité části |
| {text-align: } | Zarovnání textu v určité části |
| {padding: } | Posunutí od okraje určité hranice ze všech stran (lze vybrat specifickou) |
| {transform: } | Umožňuje otočit, zkosit a změnit měřítko určité části |

Tabulka 2 – použité tagy a deklarace



Obrázek 22 - ukázka zdrojového kódu frontendu chatbota

### HTTP Request

Pro zhotovení našeho chatovacího okénka nám už jen stačí zhotovit backend, Javascript. Abychom mohli vůbec úspěšně odesílat nějaká data musíme udělat funkci pro tlačítko, onbtnClick(). Text, který píše uživatel, by měla být zpravidla nějaká otázka.

Nejprve ve funkci si uděláme podmínku, zda okénko, do kterého píše uživatel, není prázdné. Pokud není, text se napíše do nového odstavce v chatovacím okénku, a další otázky od uživatele by se měli psát pod ním.



Obrázek 23 – ukázka Javascriptu pro odesílání dat z chatovacího okénka

Podle vzoru v JSONu si dosadíme za *Messsage* hodnotu z pole, kam uživatel psal svůj dotaz/otázku. Celý tento JSON formát se poté pošle přes request metodu POST na link http://localhost:5005/webhooks/rest/webhook/. Neboť datový typ pro proměnnou Val je objekt, a ne text, musíme využít metodu JSON.stringify(). Pokud se úspěšně odešle request (200), měli bychom přes *xhr.responseText* získat zpět odpověď.

*console.log('Time elapsed:', (new Date().getTime() - start\_time) / 1000);*

Tento kus kódu nám vypíše do konzole v prohlížeči odezvu (Odpověď-Otázka)

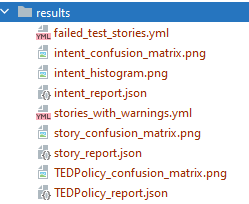
|  |  |
| --- | --- |
| 200 OK | Úspěšný požadavek |
| 201 Created | Vytvoření nového zdroje |
| 204 No Content | Úspěšný požadavek, kde nebyl předán žádný obsah |
| 304 No Modified | Nedošlo ke změně požadavku od posledního pokusu |
| 400 Bad Request | Chybně zapsaný požadavek |
| 401 Unauthorized | Zatím neprovedená autorizace |
| 403 Forbidden | Server odmítl odpovědět |
| 405 Method Not Allowed | Požadavek byl položen s nepodporovanou metodou |
| 410 Gone | Obsah již nadále není dostupný |

Tabulka 3 – stavové kódy http

## Testování

Pro testování chatbota jsme využili přikazů do konzole.To jsou *rasa data validate* a *rasa test*. Abychom zjistili, zda chatbot rozezná daný intent, jsme využili TrackerStoru, který si musíme nakonfigurovat v souboru endpoints.yml. Abychom vyzkoušeli, zda vůbec Rasa Server funguje, stáhnuli jsme si platformu Postman, která dokáže odesílat requesty.

*Rasa data validate* validuje data, kontroluje naše zdrojové soubory Rasy (data, config, domain, endpoints). *Rasa test* spustí všechny soubory, které začínají „test\_“, a vytvoří adresář results, ve kterých jsou výsledky zpráv



Obrázek 24 – soubory s výsledky rasa test



Obrázek 25 – soubor endpoints.yml

# Závěr

Při vývoji této MP jsme byli ze začátku úplně ztraceni, neboť jsme nikdy nesetkali s programovacím jazykem Python a NLP. Byla to pro nás velká výzva, Ale i přesto jsme dokázali přes tuto fázi dostat. Velice často jsme ztráceli čas nad vybíráním platformy Chatbota a jeho struktury, poté jsme měli velké potíže zprovoznit GitLab. To se vyřešilo, ale bohužel docela pozdě. Naštěstí jsme se s klukama včas vzpamatovali a začali na MP pracovat. Jelikož cílem této práce bylo zprovoznění chatbota, aby mohl porozumět otázkám a dokázat na ně odpovědět. Proto si myslíme, že jsme tento cíl splnili. Samozřejmě nám chybělo dodělat autorizaci pro REST API a lepší testování při vývoji.

# Použitá Literatura

[online]. Copyright © [cit. 20.04.2022]. Dostupné z: <https://www.besteto.cz/chatboty-podivejte-se-co-ne-umi>

[online]. Dostupné z: <https://dev.to/petr7555/rasa-creating-a-chatbot-3kla>

13 Best Open Source Chatbot Platforms to Use in 2022 | Botpress. Build Chatbots | Chatbot for Developers [online]. Dostupné z: <https://botpress.com/blog/open-source-chatbots>

3 Types of chatbots you wish you knew sooner | Freshworks. Freshworks | Modern and Easy Customer and Employee Experience Software [online]. Copyright © Freshworks Inc. All Rights Reserved [cit. 20.04.2022]. Dostupné z: <https://www.freshworks.com/live-chat-software/chatbots/three-types-of-chatbots/>

A Rule-Based or AI Chatbot? Here's the Difference | ReviewPro. ReviewPro - Upgrade the Guest Experience [online]. Copyright © Shiji Information Technology Spain, S.A [cit. 20.04.2022]. Dostupné z: <https://www.reviewpro.com/blog/should-i-choose-a-rule-based-or-an-ai-chatbot/>

Attention Required! | Cloudflare. Attention Required! | Cloudflare [online]. Dostupné z: <https://geekflare.com/chatbot-development-frameworks/>

Facts & Figures · spaCy Usage Documentation. spaCy · Industrial-strength Natural Language Processing in Python [online]. Copyright © 2016 [cit. 20.04.2022]. Dostupné z: <https://spacy.io/usage/facts-figures>

Chat Bot in Python with ChatterBot Module - GeeksforGeeks. GeeksforGeeks | A computer science portal for geeks [online]. Dostupné z: <https://www.geeksforgeeks.org/chat-bot-in-python-with-chatterbot-module/>

Chatbots Using Python and Rasa - GeeksforGeeks. GeeksforGeeks | A computer science portal for geeks [online]. Dostupné z: <https://www.geeksforgeeks.org/chatbots-using-python-and-rasa/>

Introduction to Rasa Open Source. Open source conversational AI | Rasa [online]. Copyright © 2022 Rasa Technologies GmbH [cit. 20.04.2022]. Dostupné z: <https://rasa.com/docs/rasa/>

Natural Language Processing With Python's NLTK Package – Real Python. Python Tutorials – Real Python [online]. Copyright © 2012 [cit. 20.04.2022]. Dostupné z: <https://realpython.com/nltk-nlp-python/>

Open source conversational AI | Rasa [online]. Dostupné z: <https://rasa.com/docs/rasa/training-data-format>

Stavové kódy HTTP. Rychlý a spolehlivý webhosting ➡ ️Savana.cz [online]. Copyright © 2014 [cit. 20.04.2022]. Dostupné z: <https://www.savana.cz/napoveda/php-a-html/stavove-kody-http-2/stavove-kody-http>

Střední škola, kde může začít i tvá budoucnost - SPŠE Plzeň. Střední škola, kde může začít i tvá budoucnost - SPŠE Plzeň [online]. Copyright © VOŠ a SPŠE Plzeň 2018 [cit. 20.04.2022]. Dostupné z: <https://www.spseplzen.cz/>

TextBlob: Simplified Text Processing — TextBlob 0.16.0 documentation. [online]. Copyright © Copyright 2020 [cit. 20.04.2022]. Dostupné z: <https://textblob.readthedocs.io/en/dev/>

Types of Chatbots Relevant for 2022|Rule-Based Chatbots vs AI Chatbots. MindTitan - Build AI-driven organization with machine learning solutions [online]. Dostupné z: <https://mindtitan.com/resources/guides/chatbot/types-of-chatbots/#how-ai-chatbot-is-different>

W3Schools Online Web Tutorials. W3Schools Online Web Tutorials [online]. Dostupné z: <https://www.w3schools.com/>

Web Scraping using Selenium and Python | ScrapingBee . ScrapingBee, the best web scraping API. [online]. Copyright © 2022 [cit. 20.04.2022]. Dostupné z: <https://www.scrapingbee.com/blog/selenium-python/>

Web Scraping: What It Is & How It Works - Zyte. Zyte (formerly Scrapinghub) #1 Web Scraping Service [online]. Copyright © Zyte Limited [cit. 20.04.2022]. Dostupné z: <https://www.zyte.com/learn/what-is-web-scraping/#What-is-web-scraping>

What is a REST API? | IBM. [online]. Copyright © Copyright IBM Corporation 2022 [cit. 20.04.2022]. Dostupné z: <https://www.ibm.com/cloud/learn/rest-apis>

# Seznam obrázků a tabulek

Obrázek 1 výstřižek z oficiálních stránek (viz. https://www.jetbrains.com/pycharm/features/)

Obrázek 2 – jak by měl vypadat workflow

Obrázek 3 – původní návrh MP

Obrázek 4 – finální návrh MP

Obrázek 5 – diagram oblasti zájmů

Obrázek 6 – ukázka otázek

Obrázek 7 – ukázka odpovědí

Obrázek 8 – skript pro vytvoření databáze

Obrázek 9 - ukázka nlu.yml

Obrázek 10 – ukázka stories.yml

Obrázek 11 – ukázka rules.yml

Obrázek 12 – ukázka rules.yml

Obrázek 13 – ukázka souboru domain.yml (intenty)

Obrázek 14 – ukázka souboru domain.yml (odpovědi)

Obrázek 15 – ukázka souboru domain (konfigurace relace)

Obrázek 16 – nastavení config souboru do anglického jazyka

Obrázek 17 – ukázka používaných Policies

Obrázek 18 – ukázka endpoints.yml

Obrázek 19 – ukázka web scrapingu

Obrázek 20 – connector pro databázi

Obrázek 21 – schéma pro REST API

Obrázek 22 - ukázka zdrojového kódu frontendu chatbota

Obrázek 23 – ukázka Javascriptu pro odesílání dat z chatovacího okénka

Obrázek 24 – soubory s výsledky rasa test

Obrázek 25 – soubor endpoints.yml

Tabulka 1 – příkazy SQL

Tabulka 2 – použité tagy a deklarace

Tabulka 3 – stavové kódy http