Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Anotační aplikace pro úlohy z oblasti zpracování přirozeného jazyka

Plzeň 2022 David Bubik

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**David BUBIK**
Osobní číslo:	**A19B0015P**
Studijní program:	**B0613A140015 Informatika a výpočetní technika**
Specializace:	**Informatika**
Téma práce:	**Anotační aplikace pro úlohy z oblasti zpracování přirozeného jazyka**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se s vybranými úlohami z oblasti zpracování přirozeného jazyka a dostupnými nástroji pro anotaci dat pro tyto úlohy.
2. Navrhněte aplikaci umožňující anotování dat pro alespoň tři vybrané úlohy.
3. Implementujte navrženou aplikaci a zajistěte její snadnou rozšiřitelnost o další úlohy.
4. Aplikaci otestujte a ověřte její funkčnost anotací dat dodaných vedoucím práce.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Ing. Pavel Přibáň**
Nové technologie pro informační společnost

Datum zadání bakalářské práce: **4. října 2021**
Termín odevzdání bakalářské práce: **5. května 2022**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 14. října 2021

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 22. června 2022

David Bubik

Poděkování

Tímto bych rád poděkoval Ing. Pavlu Přibáňovi za odborné vedení, cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracovávání bakalářské práce.

Abstract

The goal of this work is to create an annotation application that allows annotation of at least three types of NLP tasks and is easily extensible to other tasks. An annotation application is a system that allows easy annotation of datasets. In the first part of this paper, I will describe selected NLP tasks. Next, I will review existing annotation applications and compare their features. The following part of the thesis describes the system developed by me that allows annotating NLP tasks (text classification, NER, BSA, text summarization a text translation) and exporting labeled datasets. The last part contains the implementation and testing of the developed annotation application.

Abstrakt

Cílem této práce je vytvořit anotační aplikaci, která umožňuje anotaci nejméně tří typů NLP úloh a je dále snadno rozšiřitelná o další úlohy. Anotační aplikace je systém, který umožňuje snadnou anotaci (označování) datových sad. V první části práce popíši vybrané NLP úlohy. Dále projdu již existující anotační aplikace a porovnám jejich vlastnosti. Následující část práce popisuje mnou vyvíjený systém, který umožnuje anotaci NLP úloh (klasifikace textu, NER, BSA, sumarizace textu a strojový překad) a exportování označených datových sad. Poslední část obsahuje implementaci a testování vytvořené anotační aplikace.

Obsah

1 Úvod

V dnešní době je běžné, že informace získáváme ze zpravodajských serverů a sociálních sítí a různých diskuzních fór. Tyto informace mohou pocházet z různých zdrojů a jsou převážně nestrukturované. Oblast zpracování přirozeného jazyka (Natural Language Processing, NLP) se zabývá zpracováním, reprezentací a analýzou velkých množství textových dat, které jsou právě v této formě.

V současnosti se pro řešení NLP úloh používá především strojové učení (Machine Learning, ML), které je na nejvyšší úrovni možné rozdělit na učení s učitelem (supervised) a učení bez učitele (unsupervised). Abychom mohli použít algoritmy strojového učení učeného s učitelem, je nutné vytvořit datové sady, které budou již anotované. Anotované datové sady, jsou takové datové sady, jejichž data mají přiřazený (označený) správný výsledek, který je výstupem dané NLP úlohy. Kupříkladu sada novinových článků může být rozdělena podle tématu textu (politika, sport, atd.) nebo je možné text rozdělit na části a jednotlivým částem přiřadit kategorii. K vytvoření těchto sad slouží anotační nástroj, ve kterém je možné neoznačená data upravovat a označovat (anotovat). Je samozřejmě možné data anotovat ručně, například v aplikaci Excel, ale to stojí velké množství úsilí a času. Dalším úskalím použití editorů, které nejsou zaměřeny na anotaci datových sad je to, že anotace některých úloh je nepraktická a v jiných případech (např. ABSA, NER) není možná. Výsledné datové sady nemusejí mít odpovídající hodnotu. Jinak řečeno, je to drahé.

Motivací pro tuto práci, je vytvořit snadno rozšiřitelný anotační nástroj, který by usnadnil vytváření datových sad použitelných pro anotaci vybraných NLP úloh.

1.1 Cíle práce

Prvním cílem této práce je seznámit se s vybranými úlohami z oblasti NLP a jejich využitím. Dalším cílem je prozkoumat již vytvořené anotační nástroje. Třetím cílem je návrh systému, který bude umožňovat anotaci. Zde navrhnu a popíši základní funkcionalitu mnou vyvíjeného systému.

Hlavním cílem této práce je návrh a implementace anotační aplikace. Aplikace musí umožňovat práci dvou typů uživatelů: administrátora a anotátora. Administrátor vytváří anotační scénáře a přiděluje je anotátorům k

vypracování. Anotační scénář je vytvořen pro jednu NLP úlohu a obsahuje datové sady určené k anotaci. Dále administrátor může vytvářet a exportovat jednotlivé scénáře do vybraného formátu (např. JSON, CSV). Anotátor vypracovává zadané anotační scénáře. Celý systém musí být snadno rozšiřitelný a umožňovat implementaci dalších úloh z oblasti NLP.

Cílem této práce je vytvořit vhodný anotační nástroj pro usnadnění a urychlení vytváření anotovaných datových sad pro úlohy NLP. V poslední části práce je popsáno ověření funkcionality aplikace a její otestování.

Po přečtení této práce by měl čtenář pochopit, jak vyvíjený systém funguje, jaké jsou jeho součásti a jak ho případně rozšířit. Dále by se měl dozvědět jaké funkcionality jsou již implementovány a jak je užívat.

2 Úlohy z oblasti zpracování přirozeného jazyka

Zpracování přirozeného jazyka (NLP) je soubor metod a technik pro zpřístupnění lidského jazyka počítačům [18, 19], dále spadá pod oblast umělé inteligence (Artifical Inteligence dále jen AI). NLP využívá strojové učení (Machine Learning, dále jen ML) a hluboké učení (Deep Learning, dále jen DL) (viz obr. 2.1). ML se obecně dělí na přístup učení s učitelem (supervised) a bez učitele (unsupervised) [17].

Při učení s učitelem (supervised learning) se k trénování klasifikátorů používají označené (anotované) datové sady. Data jsou tedy předem rozdělena do pevně daných kategorií.

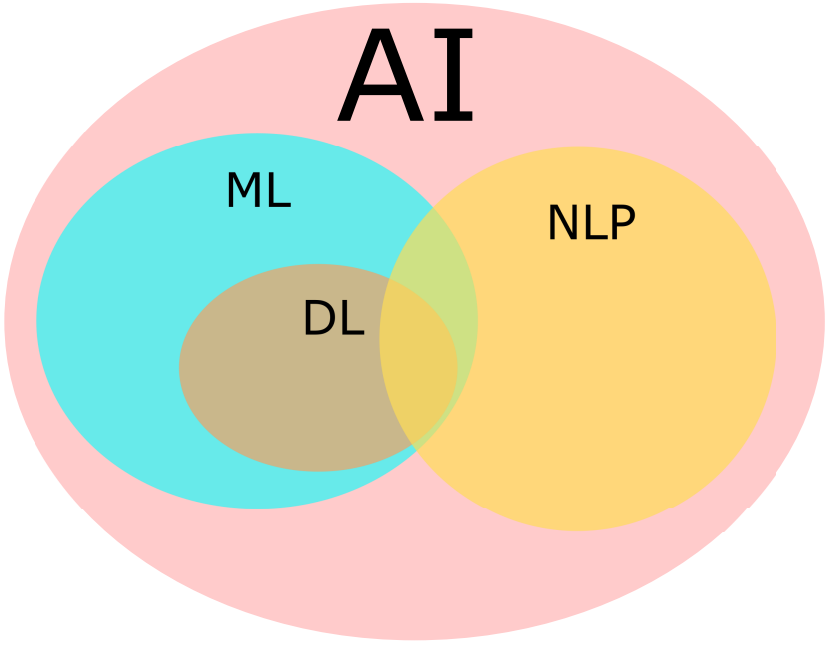
Učení bez učitele (unsupervised learning) anotované datové sady a předem dané kategorie nevyžaduje. Typickou úlohou učení bez učitele například je shlukování. Princip shlukování je rozdělování dat do clusetrů (skupin) na základě podobných vlastností.

NLP využívá oba přístupy učení s učitelem i bez učitele. Pro úlohy učení s učitelem je tedy nutné vytvářet označované datové sady. NLP obsahuje mnoho úloh pro zpracování dat, dále uvedu a stručně popíši některé nejznámější z nich.

2.1 Klasifikace textu

Klasifikace textu [23] je úloha z oblasti zpracování přirozeného jazyka. Cílem této úlohy je zadanému vstupnímu textu, dokumentu, odstavci či větě přiřadit odpovídající třídu nebo třídy. Danému textovému dokumentu se přiřadí množina označení y ϵ Y, kde Y je množina všech možných označení [19]. Klasifikace textu se dále dělí na klasifikaci právě jedné třídy (single label, viz obr. 2.2a) a klasifikaci více tříd (multi label, viz obr. 2.2b).

Tato úloha z NLP je například vhodná pro získávání názoru o zadaném textu, např. zda je poskytnutá recenze negativní, pozitivní, nebo neutrální. Další příklad je úloha detekce spamu, tzn. zda je email spam nebo není spam.



Obrázek 2.1: Znázornění vztahu mezi pojmy AI, ML, DL a NLP [1].

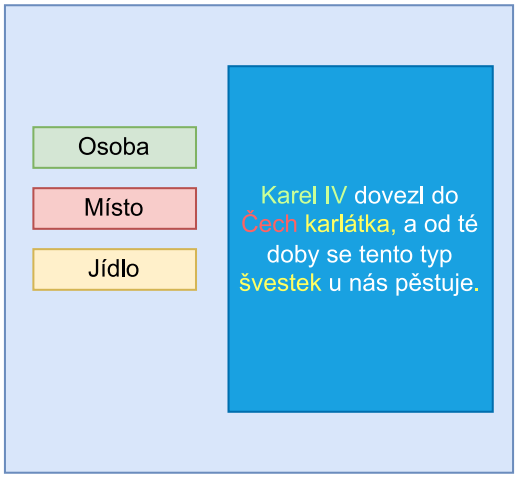
(a) Single label. **(b)** Multi label.

Obrázek 2.2: Klasifikace textu

2.2 Rozpoznání pojmenovaných entit

Rozpoznání pojmenovaných entit (Named entity recognition dále jen NER) [31] také patří mezi základní úlohy z oblasti NLP. Úkolem NER je najít části textu, které představují vlastní jména, a označit typ pojmenované entity (Named Entity dále jen NE) [21]. NE je cokoli, na co se můžeme odkazovat jasným jménem [21], např. město "New York City" nebo jméno "Marie Curie".

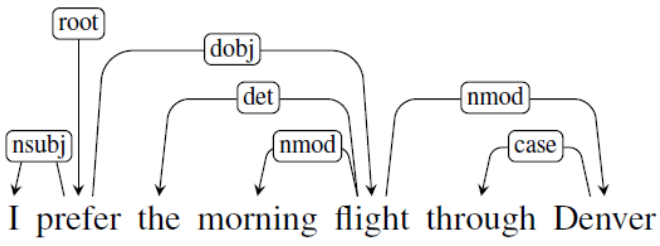
NER se využívá v personálním oddělení firem k urychlení náborového procesu shrnutím životopisů uchazečů, zlepšení interních pracovních postupů kategorizací stížností a dotazů zaměstnanců. Další využití je na zákaznické podpoře, kde snižuje čas zpracování dotazu pomocí kategorizace požadavků, stížností a dotazů uživatelů a filtrování podle prioritních klíčových slov [22].



Obrázek 2.3: NER výběr tří entit.

2.3 Parsování závislostí (Dependency parsing)

Parsování závislostí je jednou z pokročilejších úloh z NLP. Při vybírání závislostí jednotlivé fráze a slovní spojení nehrají přímou roli. Místo toho se pomocí řízené binární soustavy popisuje syntaktická struktura textu a vztahy mezi slovy [21]. Anotátor tedy vybere entity které spolu závisí a spojí je příslušnou vazbou. Cílem této metody je určit jak daná slova, nebo slovní spojení na sobě závisí.



Obrázek 2.4: Příklad Dependency parsing [21]

2.4 Sumarizace textu

Sumarizace textu je dalším typem NLP úlohy. Tato úloha je praktickou aplikací kontextového autoregresního generování. Úkolem je vzít plnohodnotný článek a vytvořit jeho efektivní shrnutí [21].

Použití úlohy Sumarizace textu je vhodné pro psaní automatických shrnutí. Například automatické napsání obsahu internetových článků.

Obrázek 2.5: Příklad Sumarizace textu.

2.5 Strojový překlad

Další zmíněnou úlohou je strojový překlad. Cílem této úlohy je využití stroje k přeložení textu z jednoho jazyka (např. čeština) do jiného jazyka (např. angličtina).

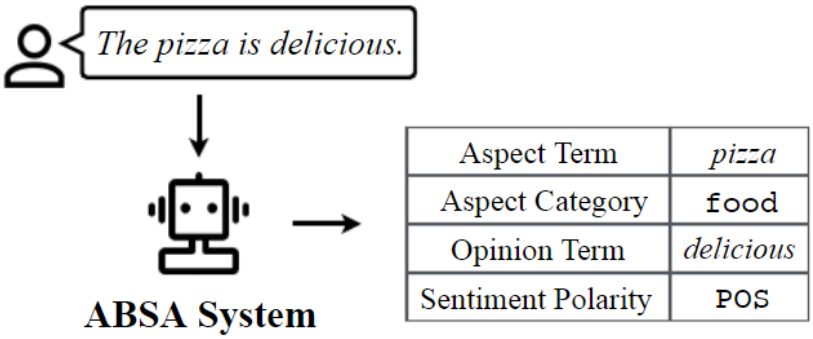
Strojový překlad v současné podobě se zaměřuje na řadu velmi praktických úkolů. Asi nejčastějším současným využitím strojového překladu je pro přístup k informacím [21], například přeložení instrukčních materiálů.

2.6 Aspektová analýza sentimentu

Při analýze sentimentu na základě aspektů (Aspect-Based Sentiment Analysis, dále jen ABSA) je cílem identifikovat vlastnosti entit a sentiment vyjádřený pro každý aspekt [25, 27]. Tuto NLP úlohu lze dále rozdělit na dvě podúlohy, klasifikaci na úrovni slov (termů) ABSA-Term a klasifikaci kategorií aspektů ABSA-Category .

Obrázek 2.6: Příklad strojového překladu.

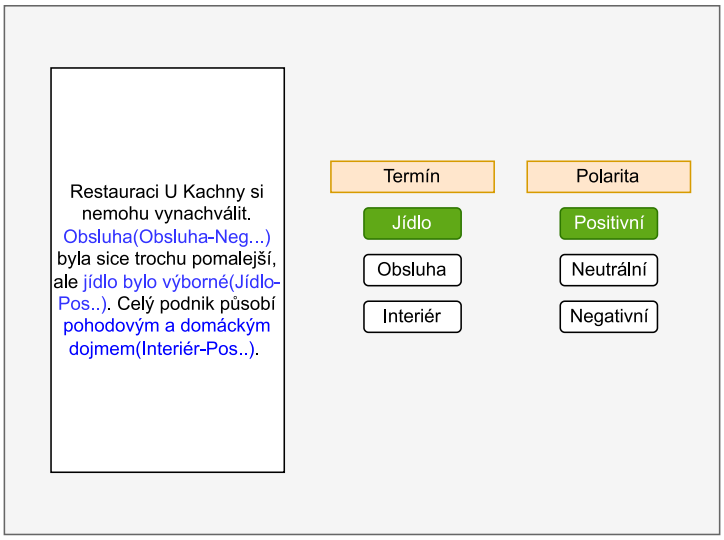
ABSA-Term se dále dělí na dvě části. První je extrakce termů (slov), zde vzhledem k sadě je úkolem identifikovat všechny aspekty na úrovni slov, které se v každé větě vyskytují (např. "víno", "číšník", "předkrm", "cena", "jídlo"). Druhou úlohou je určit polaritu vybraných aspektů (např. pozitivní, neutrální, negativní, atd.) [28] (viz obr. 2.7).



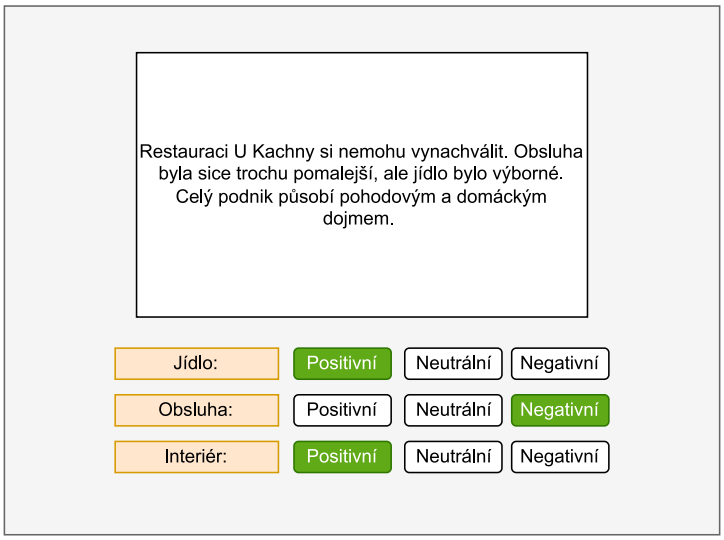
Obrázek 2.7: Ukázka fungování ABSA-Term [32]

ABSA-Category se také dělí na dvě podúlohy. V první je cílem detekovat kotegorie, které se v daném textu nacházejí a ve druhé se kategoriím přiděluje polarita [28].

V dnešní době je mimo jiné ABSA používána pro získávání dat z recenzí, kritik nebo jiných posudků. Například z dlouhé recenze na restauraci lze snadno získat informace o jídle, obsluze, prostředí, atd. [32].



Obrázek 2.8: ABSA anotace - Term



Obrázek 2.9: ABSA anotace - Categories

3 Vybrané anotační nástroje

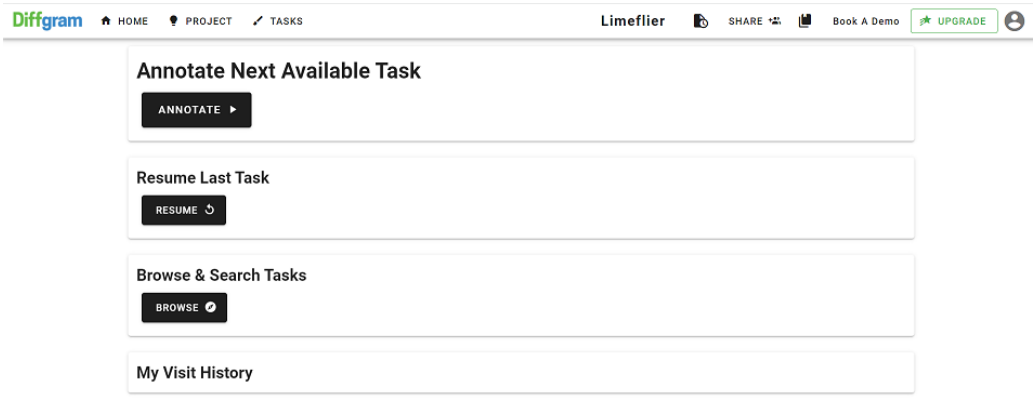
Motivací k prozkoumání již existujících anotačních aplikací, je získat přehled o těchto programech, jejich možnostech a funkcích, které mají k dispozici. Dále se budu při vývoji mého systému vybranými prozkoumanými aplikacemi inspirovat.

Představím tedy existující anotační nástroje, které jsou k dispozici. Zhodnotím jejich možnosti a porovnám je.

3.1 Diffigram

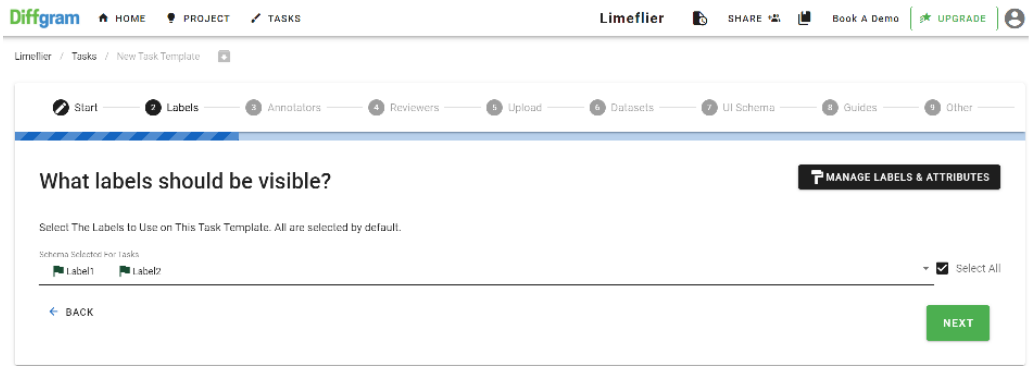
Difiigram [29] je zcela zdarma poskytovaný anotační nástroj, který umožňuje anotaci obrázků, videa a textu. Dále také umožňuje organizaci týmových projektů. Je to jeden software, který pokrývá a řeší velikou škálu různých problémů.

Z mého pohledu má příjemný design a vypadá velice přehledně. Snadné spuštění anotace a prohlížení vypracovaných scénářů.



Obrázek 3.1: Diffigram úvodní obrazovka.

Připravování nových anotačních scénářů se jeví jako poněkud zdlouhavé, ale přehledné a aplikace uživatele sama naviguje. Jedním z nedostatků je, že během vytváření scénáře musíte mít již připravené třídy, do kterých se bude poskytnutý text zařazovat. Dále import již připraveného scénáře není možný.



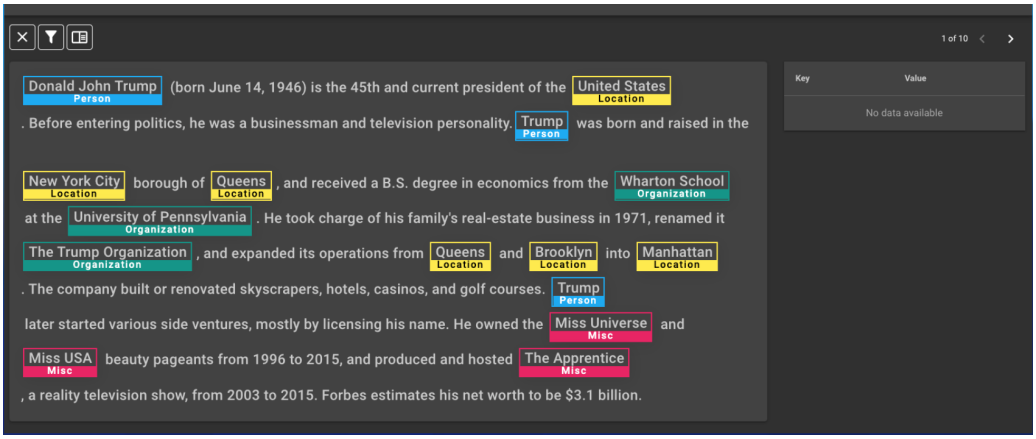
Obrázek 3.2: Diffigram zakládání scénáře.

Výhodou tohoto systému je, že je poskytnut zdarma. Je vyvinutý pro platformu linux, ale základní verze je implementovaná i jako webová aplikace. Je také pravidelně udržován a aktualizován vývojovým týmem.

Hlavní nevýhodou je, že Diffigram není dále rozšiřitelný. Dalším problémem je nepřítomnost funkcí Admin/Anotátor. Je také potřeba zmínit, že není možné importovat již připravené scénáře a že vytváření nových scénářů za použití Diffigramu je zdlouhavé.

3.2 Doccano

Doccano [26] je zdarma rozšiřitelný anotační systém, který je příjemný pro práci a přehledný pro uživatele. Další výhodou tohoto systému je to, že je poskytnut zdarma a má příjemné vizuální prostředí pro práci.



Obrázek 3.3: Anotace NER v aplikaci Doccano .

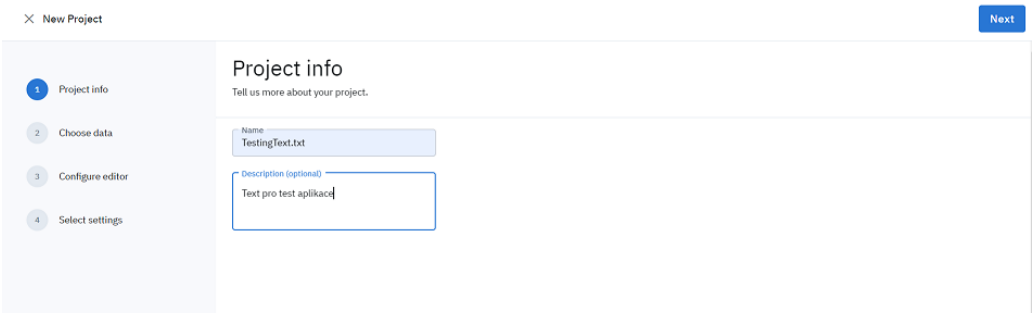
Tento systém je opensource, takže je možné implementovat úlohy z oblasti NLP, ale nevýhodu je, že není možné rozlišit mezi zadavatelem a ano-

tátorem. Jediná možnost jak oanotovat datovou sadu je, že ji uživatel sám nahraje, vybere typ anotace, zpracuje a exportuje. Každý uživatel je zde sám za sebe. Tento systém také obsahuje malé množství implementovaných úloh z oblasti NLP, dodatečné úlohy se tedy musí doprogramovat.

3.3 LabelBox

LabelBox [4] je anotační nástroj, který je vhodný pro anotaci videa, obrázků a textu. Obsahuje několik typů licencí jmenovitě common, pro a enterprise. Dále se separatně hradí každý anotovaný dokument a nový uživatel, který je nad počtem specifikovaným v licenci.

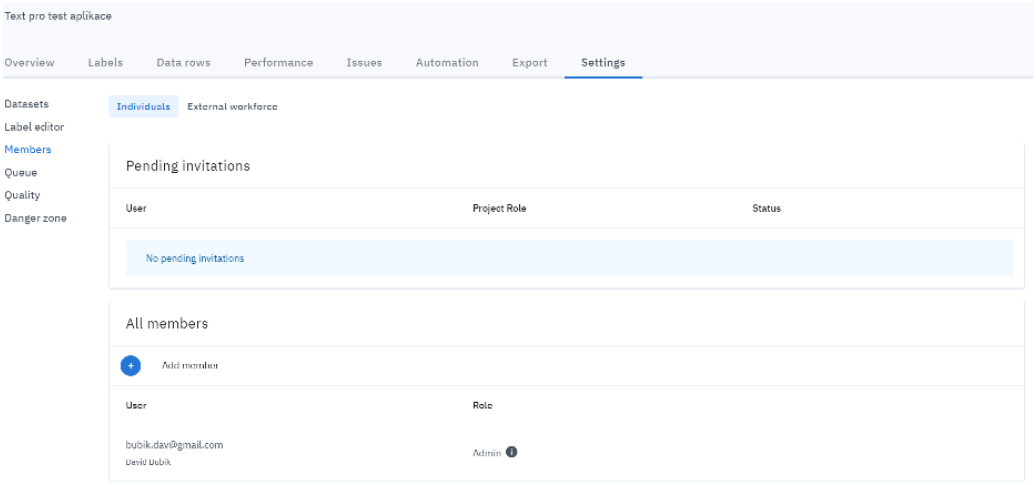
Systém samotný je přehledný rozvržený a snadný pro použití. Import a export datových sad je intuitivní. Vytváření nových scénářů je snadné.



Obrázek 3.4: Vytvoření nového scénáře v aplikaci LabelBox.

LabelBox rozlišuje roli anotátora a admina. Uživatel, který založí scénář se stává jeho adminem a může k němu přizvat další uživatele, které má ve skupině. Každý založený scénář lze sledovat, pozorovat jeho postup a získat statistické informace.

Jednotlivé licence se liší v počtu poskytovaných úloh z NLP. Dále má comon licence nastavený maximální počet anotovaných datových sad. Statistické informace, ke kterým lze přistoupit se také liší podle zakoupené licence.



Obrázek 3.5: LabelBox přidání anotátora.

3.4 Prodigy

Prodigy [8] je multiplatformní anotační nástroj, vhodný pro anotaci textu, obrazu, videa nebo audia.

Tento systém je velice přehledný a obsahuje velké množství typů úloh z oblasti NLP. Průběh samotné anotace je příjemný a snadný. Je zde možnost přehledně anotovat nejen text, ale také video a audio. Export a import datových sad je intuitivní. Prodigy má také poměrně velké zastoupení v komunitě a má plně funkční uživatelskou podporu.

Obrázek 3.6: Prodigy NER anotace.

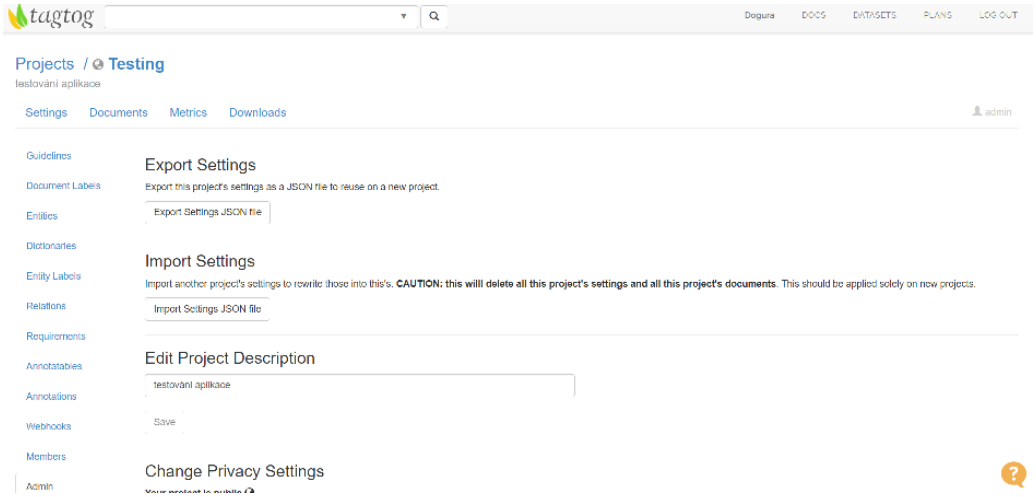
Prodigy obsahuje administrátorský model a anotátorský model. Design prostředí je nastavitelný. Z důvodu, že se nejedná o opensource, není možný

jakýkoli rozvoj aplikace ze strany uživatele.

3.5 TagTog

TagTog [10] je anotační nástroj, jehož základní verze je zdarma. Umožňuje anotaci textu obrazu.

V základní verzi je zde možnost se přihlásit jako admin nebo jako anotátor. Z mého pohledu je propracovanost vztahu mezi adminem a anotátorem velice přehledná. Dále je zde také k dispozici mnoho typů úloh z oblasti NLP.



Obrázek 3.7: TagTog import export JSON souborů.

TagTog má z mého pohledu nepřehledné GUI. Je velice těžké se zde orientovat bez tutoriálu a i když aplikace obsahuje mnoho typů úloh z oblasti NLP, samotný průběh anotace je nepříjemný a nepřehledný. Dále založené scénáře pod neplacenou licencí jsou typu public. Což znamená, že jsou viditelné pro kohokoli.

3.6 Porovnání existujících systémů

Z porovnávaných anotačních aplikací lze vybrat, jako nejpropracovanější aplikaci Prodigy, která nabízí možnost připojení pod rolí admina a anotátora, je přehledná a má již implementovanou velkou škálu úloh z NLP a nastavitelný design anotačního prostředí.

Aplikace LabelBox je také velice propracovaná, až na finanční cenu a uzavřenost systému ji není co vytknout. Je zde možnost přihlásit se pod rolí anotátora nebo admina, má také možnost dále prohlížet již rozpracované scénáře.

Vlastnosti	Diffigram	Doccano	LabelBox	Prodigy	TagTog
Zdarma	Ano	Ano	Ne	Ne	Ano
Rozšiřitelná	Ne	Ano	Ne	Ne	Ne
Role admin/anotátor	Ne	Ne	Ano	Ano	Ano
Implementovaná úloha NER	Ne	Ne	Ano	Ano	Ano
implementovaná úloha Klasifikace textu	Ano	Ano	Ano	Ano	Ano
implementovaná úloha Parsování závislostí	Ne	Ano	Ano	Ano	Ano

Tabulka 3.1: Tabulka vybraných anotačních aplikací.

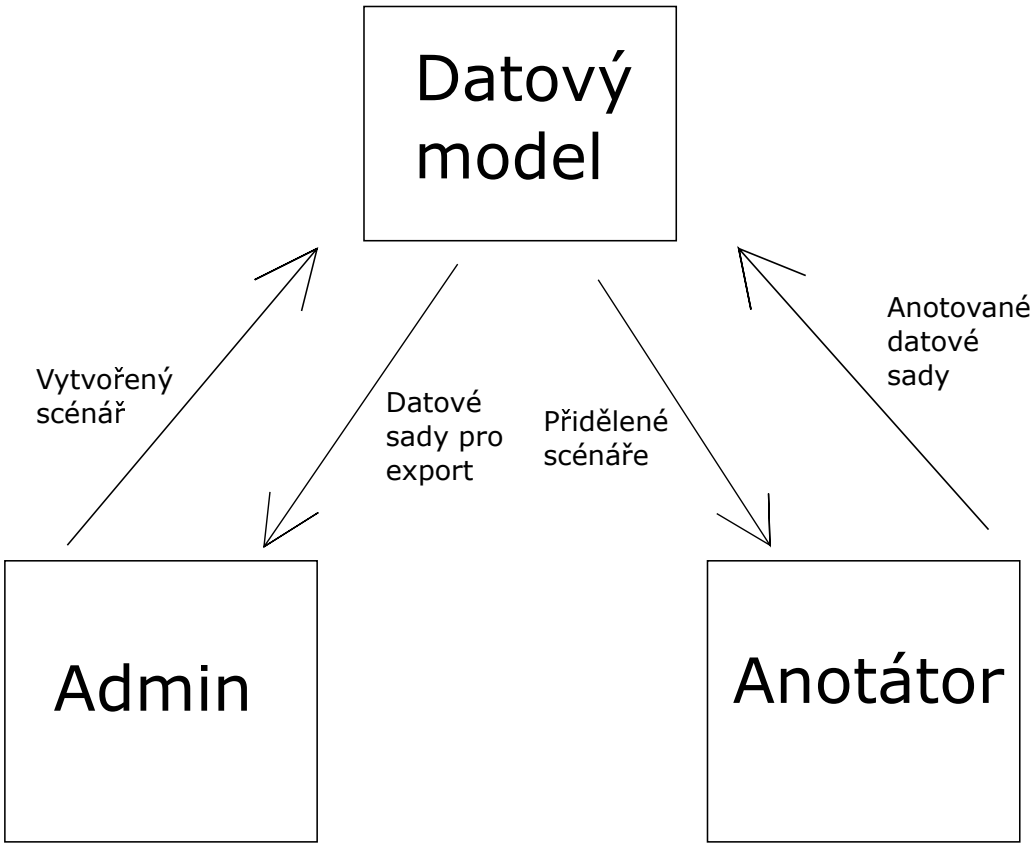
Doccano je opensource anotační nástroj. Jeho výhodou je možnost implementace dalších úloh z oblasti NLP. Nemá však roli admina. TagTog a Diffigram jsou poskytunty zdarma, ale nejsou dále rozšiřitelné a Diffigram neobsahuje možnost přihlášení pod administrátorem. Přednosti a nedostatky jednolitých anotačních aplikací jsou shrnuty v následující tabulce (viz Tabulka 3.1).

Na základě tabulky 3.1 je možné říci, že žádná z prozkoumaných anotačních aplikací nesplňuje všechny zadané požadavky. Vytvoření nového systému má tedy smysl.

4 Návrh anotační aplikace

V této kapitole popíši návrh mého systému z několika pohledů a zdůvodním výběr knihoven, které budu používat.

Mnou vyvíjený anotační systém (dále jen systém), musí umožňovat přihlášení jako administrátor a anotátor. Administrátor vytváří anotační scénáře a předává je k vypracování anotátorům. Anotační scénář obsahuje datové sady určené k označení a úlohy z NLP, za pomocí kterých bude datová sada označena. Dále administrátor může editovat a exportovat jednotlivé scénáře. Anotátor vypracovává zadané anotační scénáře. Jednotlivé datové sady a scénáře se budou ukládát do datového modelu. Z pohledu uživatele probíhá komunikace mezi administrátorem a anotátorem pouze přes datový model (viz obr. 4.1).



Obrázek 4.1: Návrh komunikace systému.

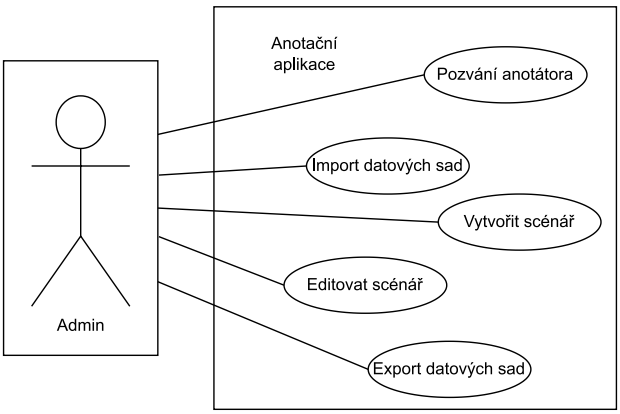
4.1 Návrh funkcionality

V této podkapitole popíši návrh požadované funkcionality. Návrh budu analyzovat z pohledu administrátora, anotátora a dále jej popíši pomocí UML diagramů případů užití. UML (Unified Modeling Language) je ”univerzální jazyk pro vizuální modelování systémů"[15], který se používá převážně v softwarovém inženýrství.

4.1.1 Návrh administrátora

Administrátor může vytvářet nové scénáře a exportovat výsledné datové sady z ukončených scénářů. Ke každému scénáři lze zobrazit postup anotace a statistické informace. Dále také admin může importovat datové sady k anotaci.

Na obrázku 4.2 je zobrazen UML diagram případů užití pro administrátora.



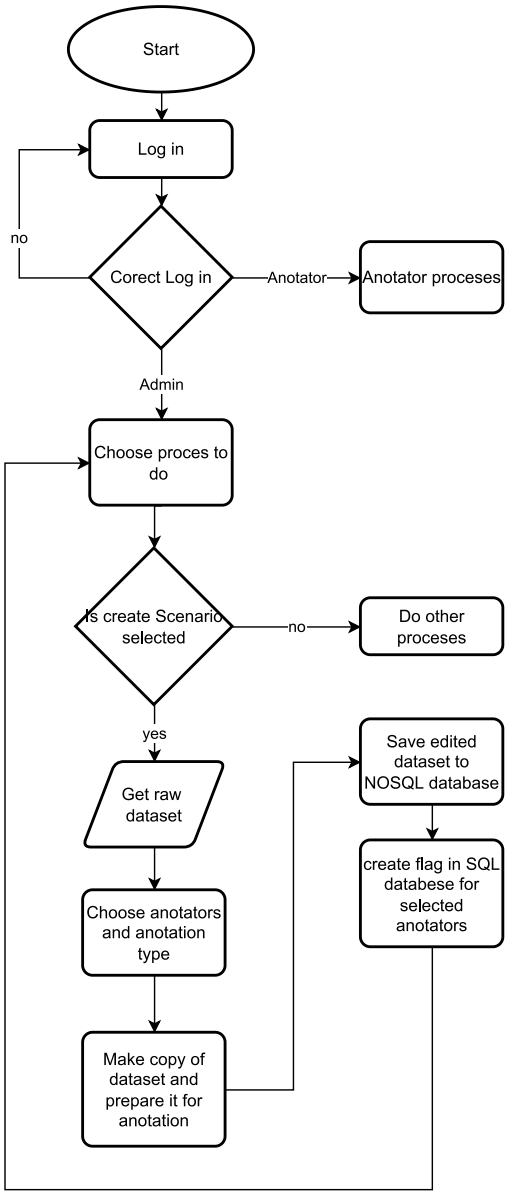
Obrázek 4.2: UML diagram případů užití znázorňující služby poskytnutých adminovi.

Administrátor také vybírá, které úlohy z NLP budou použity. Minimální počet implementovaných úloh v této práci jsou tři. Na základě kapitoly 2 byly vybrány takové NLP úlohy, které zastupují jednotlivé typy anotačních procesů. Například úloha klasifikace textu se anotuje pomocí výběru jedné nebo více tříd, úloha NER se anotuje pomocí zvýraznění různých částí textu. Jednotlivé NLP úlohy mají tedy rozdílný způsob anotace.

Návrh zadání scénáře

Při zakládání nového scénáře je potřeba zohlednit způsob uložení scénáře samotného a uložení datové sady. Je tedy zřejmé, že pro plnou funkcionalitu

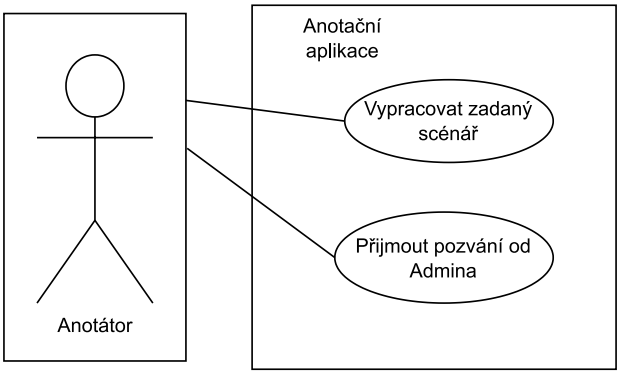
bude zapotřebí SQL databáze, pro přidělování scénářů uživatelům a NoSQL databáze, ve které budou datové sady. Při vytváření je tedy nutné vybrat neoanotovanou datovou sadu, vytvořit její kopii a připravit ji k anotaci a na závěr vložit informace o scénáři do SQL databáze pro anotátory (viz obr. 4.3).



Obrázek 4.3: Návrh zadání scénáře.

4.1.2 Návrh anotátora

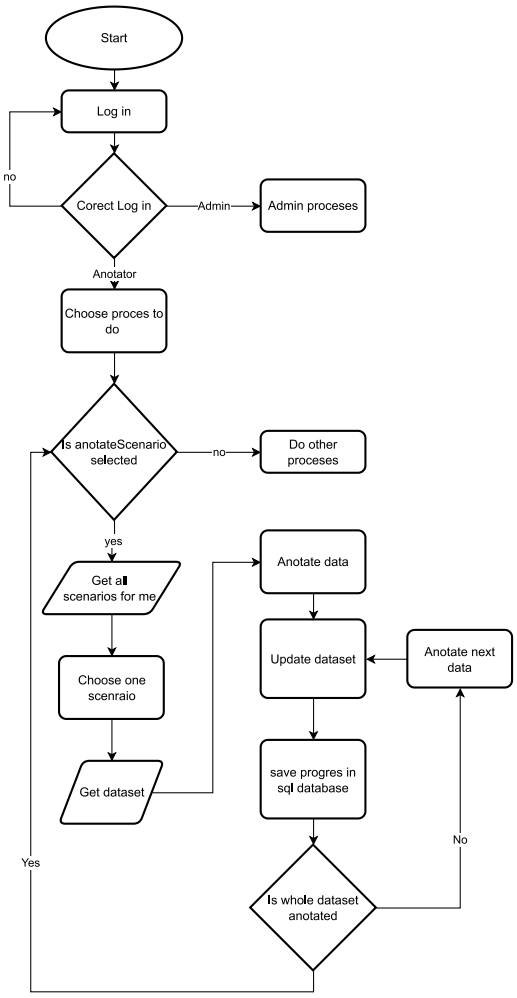
Anotátor si vybere přidělený scénář (nový nebo rozpracovaný) a začne anotovat přidělenou datovou sadu. Dále má anotátor k dispozici historii již ukončených přidělených scénářů. V historii je vidět práce, kterou uživatel vykonal (viz obr. 4.4).



Obrázek 4.4: UML diagram případů užití znázorňující služby poskytnuté anotátorovi.

Návrh vypracování scénáře

Anotátor zjistí, zda má zadanou práci a pokud ano, tak v ní najde identifikátor zadané datové sady pro anotaci. Anotátor průběžně ukládá informace o svém postupu do SQL databáze a oanotovaná data do NoSQL databáze (viz obr. 4.5).



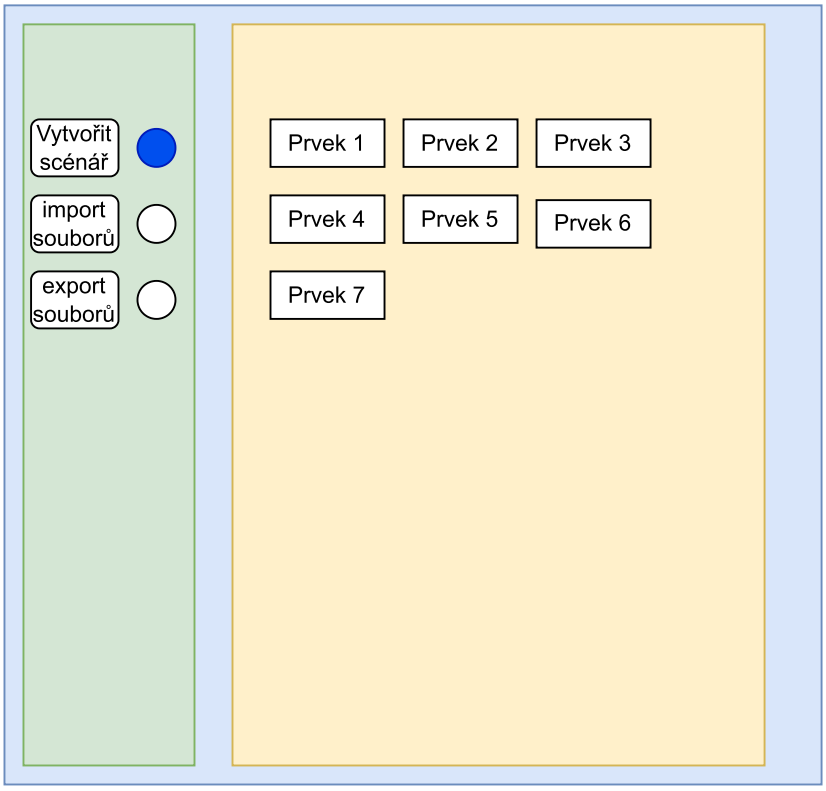
Obrázek 4.5: Návrh vypracování scénáře.

4.2 Návrh uživatelského rozhraní

Na základě návrhu funkcionality, v této kapitole popíši návrh grafického uživatelského rozhraní. Administrátor i anotátor mají několik separátních úkonů, které mohou provádět. Pro umožnění snadné orientace na levou stranu obrazovky umístím lištu, na které je možné se přepínat mezi úkony a na pravou stranu vložím již jednotlivé prvky specifické pro daný úkon (viz obr. 4.6). Prvky na liště jsou pro jednotlivé módy (administrátorský, anotátorský) odlišné.

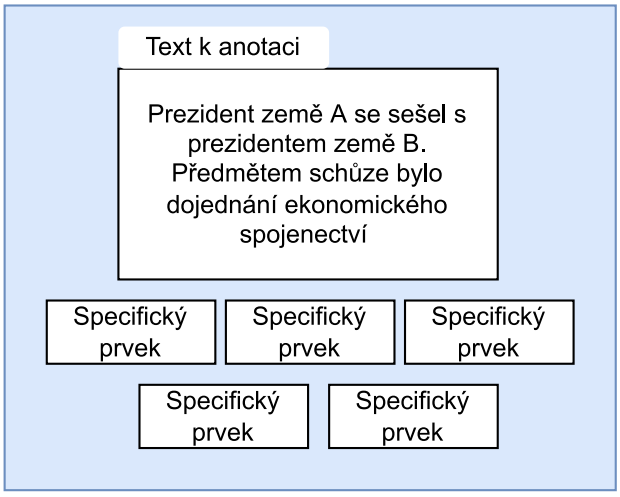
4.2.1 Uživatelské rozhraní pro anotaci NLP úloh

Anotace každé NLP úlohy vyžaduje specifické prvky pro svou realizaci. Výběrem těchto prvků se inspiruji z již existujících anotačních nástrojů popsa-



Obrázek 4.6: Návrh Základního GUI.

ných v kapitole 3. Základní layout pro anotaci jednotlivých úloh bude ale společný (viz obr. 4.7). Samotné rozložení prvků, které jsou unikátní pro svůj typ NLP úlohy, se může lišit.



Obrázek 4.7: Návrh GUI pro anotaci NLP úloh.

4.3 Návrh architektury aplikace

V této podkapitole popíši návrh mého systému z pohledu architektury a zdůvodním výběr knihoven, které budu používat.

4.3.1 Výběr frameworku

Pro implementaci webové aplikace v jazyce Java jsem se rozhodoval, zda použiji framework Spring-boot [14] nebo Vaadin [11].

Framework je již připravená sada knihoven, zdrojů a různých dokumentů, kterou programátor využívá jako základ při vytváření aplikace. Framework si lze představit jako šablonu funkčního programu, kterou je možné selektivně upravovat přidáváním kódu. Využívá sdílené zdroje - například knihovny, obrazové soubory a referenční dokumenty - a spojuje je do jednoho balíčku. Tento balíček lze upravit podle konkrétních potřeb projektu. Pomocí frameworku může vývojář přidávat nebo nahrazovat metody, aby aplikaci poskytl nové funkcionality [13].

Spring-boot

Spring-boot umožňuje rychlé a bezproblémové vytváření webových aplikací. Odstraněním většiny šablonovitého kódu a konfigurace spojené s vývojem webových aplikací poskytuje moderní model programování webových aplikací, který zjednodušuje vývoj aplikací HTML na straně serveru, rozhraní REST API a obousměrných systémů založených na událostech [9].

Vaadin

Vaadin je moderní framework pro vývoj webových aplikací v jazyce Java. Obsahuje většinu komponentů a nástrojů, které jsou potřebné k vytvoření spolehlivé a bezpečné aplikace. Komponenty, které chybí, je možné doprogramovat.

Volba frameworku

Zvolil jsem framework Vaadin. Důvodem je, že tento framework má implementováno mnoho komponent vhodných pro mou aplikaci (např. grid, button, text area) v několika grafických provedeních. Dále je plně kompatibilní s frameworkem Spring-Boot využitím add-onu *Vaadin with Spring Boot* [12]. Je tedy možné využít i vlastnosti frameworku Spring Boot.

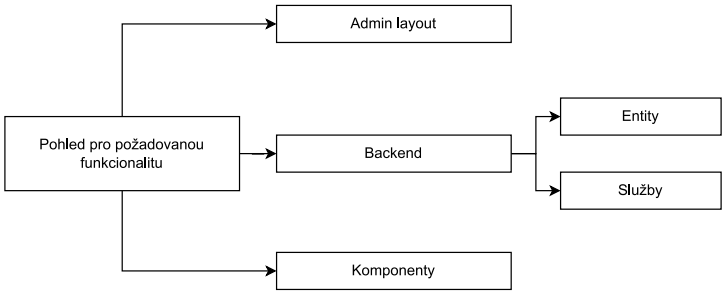
4.3.2 Obecný návrh architektury

Protože jsem zvolil framework Vaadin 4.3.1, který implementuje architekturu MVC (Model-view-controler), tak mou aplikaci lze rozdělit do tří částí: Pohledy, components a backend. Návrh mé aplikace přesně nesplňuje model MVC, ale vychází z něho. Pohledy obsahují šablony, které se zobrazí uživateli. V komponentech se nacházejí prvky, se kterými uživatel intraguje (např. tlačítka, textové plochy, atd.) a backend obsahuje entity, např. uživatel, článek; a služby, které slouží pro interakci s databázemi.

Pohledy se dále dělí na pohledy pro anotátora, pohledy pro admina a pohledy anotačních úloh.

4.3.3 Návrh architektury administrátorské části

Administrátor musí mít k dispozici služby založení/editování/export scénářů, import datové sady, přidání anotátora a porovnání anotovaných datových sad. Každá z těchto úloh musí mít vlastní pohled, komponenty a musí využívat backend aplikace. Všechny pohledy, které využívá Admin budou dědit jeden layout (viz obr. 4.8).



Obrázek 4.8: Návrh architektury administrátorské části.

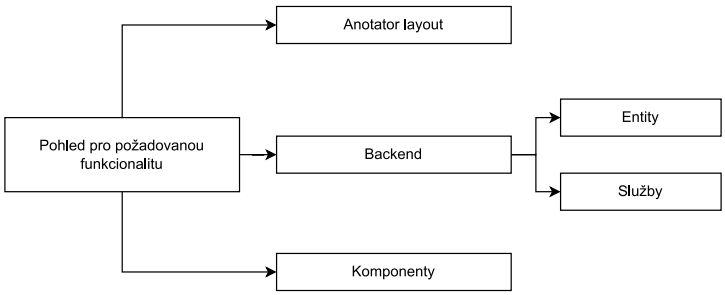
Administrátor - komunikace s databázemi

Administrátor musí ukládat tři typy dat: údaje o uživateli, údaje o scénáři a datové sady (prázdné a připravené k anotaci). Do relační databáze se budou ukládat strukturovaná data, tedy informace o uživatelích a scénářích. Každý uživatel bude mít v databázi uloženy své registrační údaje (heslo pouze v hash podobě) a ID pracovních skupin do kterých patří. Každý scénář bude mít název své datové sady (musí být unikátní), druh NLP úlohy a email anotátora (každý anotátor musí mít unikátní email). V případě, že má stejný scénář přiděleno více anotátorů bude v tabulce více záznamů, které se budou lišit pouze anotátorem.

Nestrukturovaná data budou uložena v nerelační databázi Elastisearch. Budou uložena ve formátu JSON a jsou dvojího typu: základní "čistá"datová sada a datová sada připravená k anotaci, která se od "čisté"liší tím, že má přidané informace nutné pro anotaci (např. typ anotace, co bylo anotováno, atd.). Při vytváření scénáře vybere anotátor "čistou"datovou sadu, ze které se vytvoří kopie, která se připraví pro anotaci.

4.3.4 Návrh architektury anotátorské části

Anotátor musí mít k dispozici služby vypracování scénáře a přijetí pozvání od admina. Každá z těchto úloh musí mít vlastní pohled, komponenty a využívat backend aplikace (viz obr. 4.9).



Obrázek 4.9: Návrh architektury anotátorské části.

4.4 Návrh uložení dat

Aplikace bude pracovat se dvěma typy dat. Prvním typem jsou datové sady, které se v aplikaci budou anotovat a druhým typem dat jsou data o jednotlivých scénářích, uživatelích a jiná strukturovaná data nutná pro běh aplikace. Je tedy nutné zvolit vhodné databáze pro perzistentní uložení dat.

Velké datové sady textových dokumentů nejsou vhodné pro uložení do relačních databází, ale informace o scénáři a uživatelích jsou nevhodné pro uložení do nerelačních databází. Data tedy budou uložena do dvou databází, jedné relační a druhé nerelační.

4.4.1 Relační databáze

SQL databáze je soubor datových položek uspořádaných do formálně popsaných tabulek, z nichž lze k datům přistupovat nebo s nimi manipulovat [20].

Tabulky mají pevně danou strukturu a jednotlivé sloupce mají pevně daný datový typ. S daty se manipuluje pomocí jazyka SQL. Jsou vhodné pro uchovávání strukturovaných dat.

Rozhodoval jsem se mezi Oracle-db [6], Postgres [7] a MySQL [5].

Oracle database

Databáze oracle je multiplatformní, spustitelná na Windows, Unix a na distribucích GNU/Linux. Jedná se o placenou distribuci, která je převážně zamýšlena pro správu velkého množství dat. Nejvíce je využívána v kombinaci s rozsáhlými aplikacemi.

Postgres

PostgreSQL je výkonný objektově-relační databázový systém s otevřeným zdrojovým kódem, který využívá a rozšiřuje jazyk SQL v kombinaci s mnoha funkcemi, které bezpečně ukládají a škálují nejsložitější datové úlohy. Počátky PostgreSQL sahají do roku 1986 jako součást projektu POSTGRES na Kalifornské univerzitě v Berkeley a má za sebou více než 30 let aktivního vývoje základní platformy [7].

MySQL

Databázový software MySQL je systém klient/server, který se skládá z vícevláknového serveru SQL podporujícího různé backendy, několika různých klientských programů a knihoven. Jedná se o opensource řešení, které je

vhodné pro správu malého až středně velkého množství dat. Je také možné pracovat s velkým množstvím dat.

Výběr relační databáze

Nepoužiji řešení Oracle, protože se jedná o velkou databázi vytvořenou se záměrem uchovávání velkého množství dat a je zpoplatněná. Postgres i MySQL jsou vhodné pro můj anotační systém, jedná se o stabilní opensource relační databáze, které jsou hojně používány v komunitě.

Zvolil jsem řešení MySQL, protože jsem s ním více obeznámen, používal jsem jej při vytvoření jiných projektů a znám její funkcionalitu.

4.5 Nerelační databáze

Nerelační databáze se liší od relační databáze v mnoha ohledech, hlavním z nich je, že nepoužívá relace (tabulky) jako strukturu pro ukládání dat [20]. Nerelační databáze pracuje s daty na úrovni dokumentů. Jsou vhodné pro ukládání velkého množství nestrukturovaných dat.

V mé aplikaci je nutné uložit datové sady do nerelační databáze. Při výběru jsem měl k dispozici databáze Apache Cassandra [24] a Elasticsearch [16].

4.5.1 Apache Cassandra

Apache Cassandra je open source distribuovaná databáze NoSQL, které důvěřují tisíce společností díky její škálovatelnosti a vysoké dostupnosti bez snížení výkonu [24]. Tato databáze je uzpůsobena ke zpracovávání velkých objemů rychle pohybujících se dat. Z tohoto důvodu ji pro kritické funkce využívají například firmy Netflix, Facebook nebo Instagram [2]. Manipulace s daty probíhá za pomocí Cassandra Query Language (CQL).

4.5.2 Elasticsearch

Elasticsearch je distribuovaný, bezplatný a otevřený vyhledávací a analytický nástroj pro všechny typy dat, včetně textových, číselných, geoprostorových, strukturovaných a nestrukturovaných. Elasticsearch je postaven na platformě Apache Lucene a poprvé byl vydán v roce 2010 společností Elasticsearch N.V. (nyní známou jako Elastic) [16]. Elasticksearch v zásadě pracuje s JSON soubory a má tři základní úrovně. První úrovní je dokument, jedna entita kterou lze vyhledat, např. jeden novinový článek. Druhou úrovní jsou

typy. Jednotlivé typy popisují jak budou vypadat dokumenty, které obsahuje. V našem případě se jedná o jednotlivé datové sady. Poslední úrovní je index který obsahuje kolekci typů. Například *datoveSady-06.22*. Tato databáze je vhodná pro ukládání nestrukturovaných textových souborů, jejich analýzu a prohledávání.

4.5.3 Výběr nerelační databáze

Apache Cassandra i Elasticsearch jsou vhodné pro mou aplikaci. V obou případech se jedná o open source řešení, ktré je vybavené pro zpracování nestrukturovaných dat z mé aplikace. Důvod proč jsem zvolil Elasticksearch je jeho kompatibilita s JSON soubory, možnost rychlého prohledávání, analýzy datových sad a do budoucna možnost implementace full textového vyhledávání. Dále mi bylo toto řešení doporučeno vedoucím práce.

5 Implementace anotační aplikace

V této kapitole bude popsána implementace vytvářené aplikace a způsob uložení dat. Aplikace byla programována v jazyce Java s využitím frameworku Vaadin. Jedná se o webovou aplikaci se servrovou částí vyvíjenou pro platformu Linux. Dále je k aplikaci přístupná uživatelská příručka, která je umístěna v příloze A. Aplikace samotná byla vytvořena ve vývojovém prostředí IntelliJ IDEA [3] a pro sestavení projektu byl využit nástroj Maven.

5.1 Struktura aplikace

V této části popíši strukturu aplikace a dat, které budou uloženy v databázi.

5.1.1 Struktura programu

Jednotlivé zdrojové kódy aplikace jsou rozděleny do balíčků, které jsou umístěny v adresáři */src*. Třídy jsou rozděleny podle své funkcionality a toho, kdo je využívá, např. pohled pro vytvoření scénáře je v balíčku *views.admin*. Toto rozdělení se při implementaci ukázalo jako velice praktické a přehledné.

V ardesáři *src/resources* se nachází soubor *application.properties*, který obsahuje klíčové konstanty aplikace. Mezi tyto konstanty patří port, který aplikace využívá, údaje pro připojení k databázím a jiné inicializační údaje.

Posledním souborem, který programátor upravuje, je soubor pom.xml (Project Object Model). V tomto souboru se nachází konfigurace projektu, závislosti a odkazy na knihovny a všechna potřebná data pro vytvoření projektu za pomocí nástroje Maven.

5.1.2 Strukutra dat v databázi

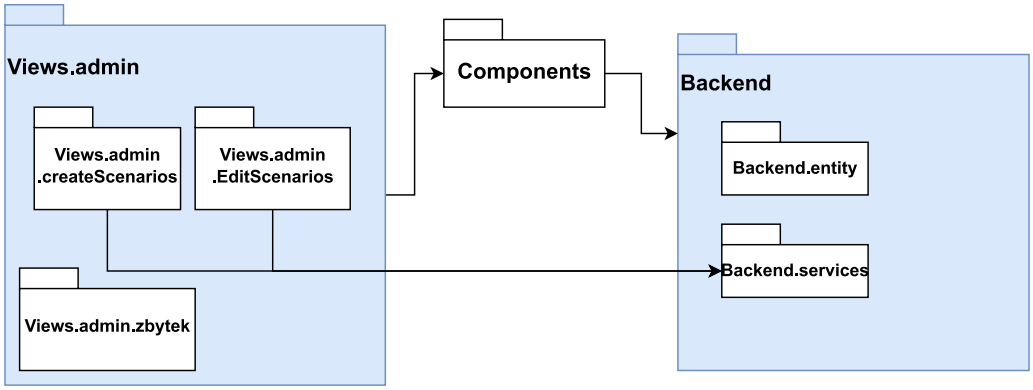
Do SQL databáze se ukládají dva typy dat: informace o uživateli a informace o vytvořeném scénáři. Každý typ dat má vlastní tabulku, která je uložená v databázi.

O uživateli se ukládají nasledující informace: ID, email, jméno, přijmení, heslo, pozice (administrátor nebo anotátor) a ID administrátorů, kteří vlastní pracovní skupiny ve kterých je uživatel přítomen. Heslo je zakódované pomocí hash funkce MD5 a uložené v databázi jako hash.

Informace o scénáři jsou následující: ID, administrátor (který založil scénář), anotátor, datum, datová sada, typ NLP úlohy, ID prvního článku, ID posledního článku, ID naposledy anotovaného článku. Z ID prvního, posledního a naposledy anotovaného článku se vypočítá procentuální postup v anotaci.

5.2 Implementace administrátorské části

V této podkapitole popíši implementaci administrátorské části aplikace. Podle cílů práce (viz 1.1) musí mít administrátor implementovány následující funkcionality: vytvářet scénáře, exportovat scénáře a přidělovat je anotátorům. Dále je vhodné, aby administrátor viděl postup v anotaci jednotlivých scénářů. UML diagram 5.1 znázorňuje relace mezi balíčky administrátorské části.



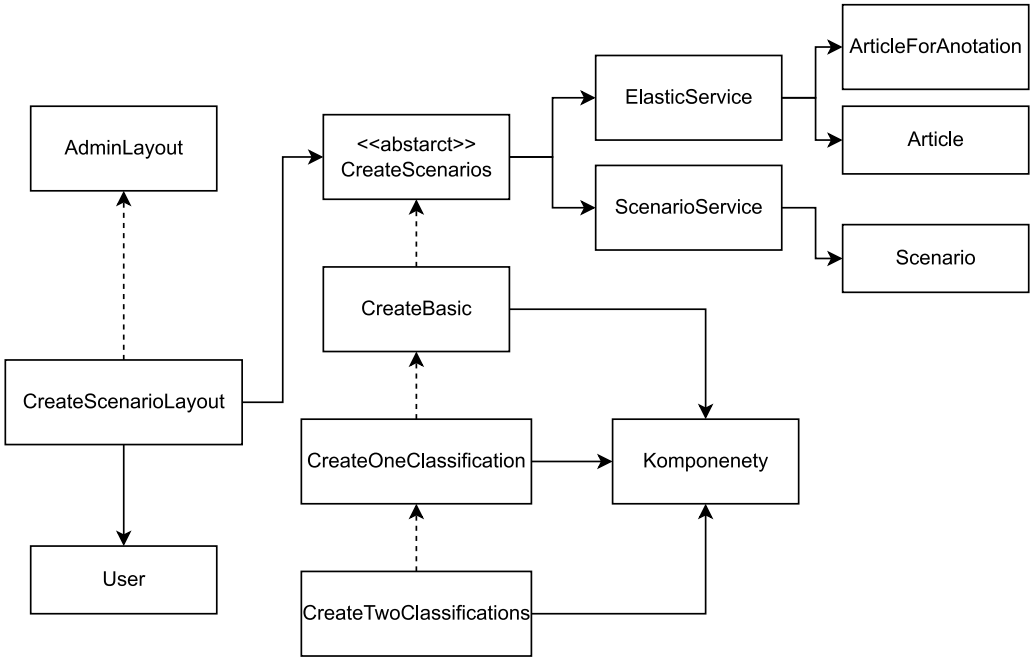
Obrázek 5.1: UML diagram balíčků administrátorské části.

Jednotlivé třídy, které jsou v balíčku *views.admin* jsou zodpovědné za zobrazení šablon pro jednotlivé administrátorské funkcionality. V těchto šablonách jsou zobrazeny komponenty z *components.admin*, kterými uživatel ovládá aplikaci. Šablony i komponenty využívají služby pro komunikaci s databázemi z balíčku *backend.service*.

5.2.1 Vytváření scénářů

Z cílů práce (viz 1.1) je zřejmé, že musí být možné založit scénář pro více NLP úloh. Podle návrhu anotátorské části (viz 4.3.3) budou všechny pohledy pro založení scénáře používat stejný layout. Při implementaci se ukázalo, že je nejpraktičtější postupně dědit jednotlivé sady komponent od "nejjednoduššího"po "nejsložitější". Například chceme založit scénář s typem anotace

"Klasifikace textu". Nejprve načteme společný layout, dále se rozhodneme do jakého typu "návrhu"patří klasifikace textu a podle toho zvolíme komponenty, které jsou nutné pro vykreslení. Jednotlivé sady komponentů dědí od abstraktní třídy, která obsahuje metody pro komunikaci s databázemi (viz obr. 5.2).

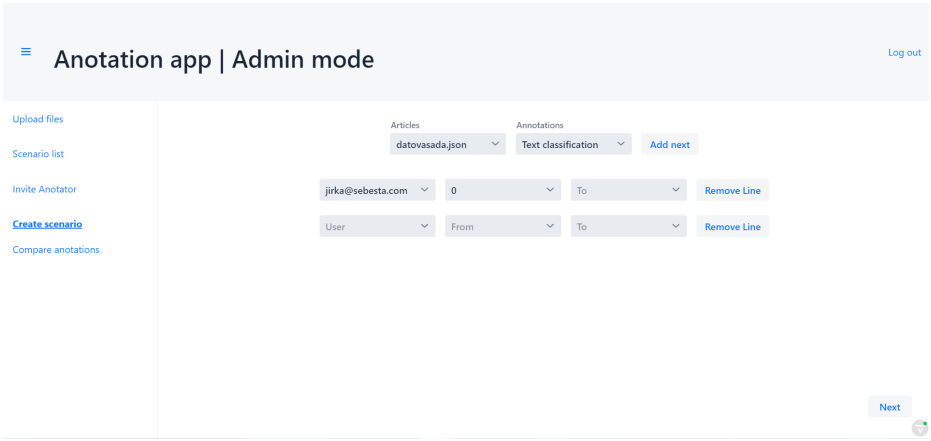


Obrázek 5.2: UML-diagram založení scénáře.

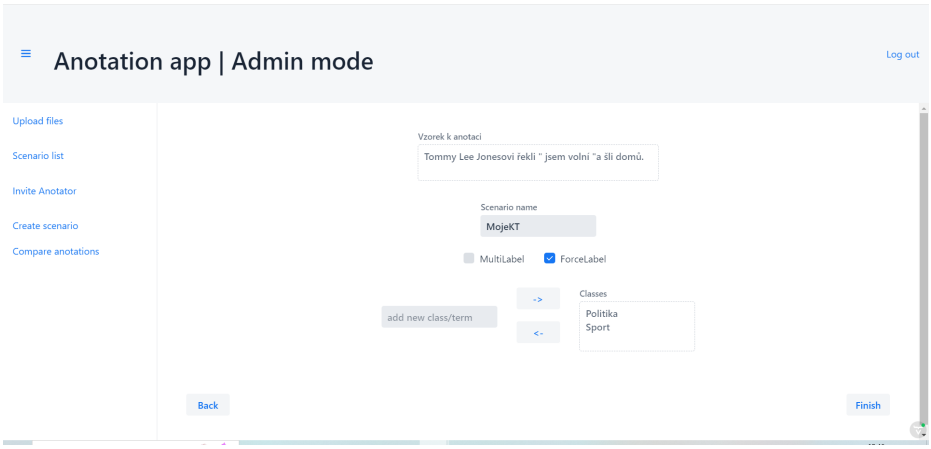
Pro zobrazení základní šablony je využita třída *views.admin.AdminView.java* (viz obr. 5.3). V této šabloně uživatel vybere datovou sadu pro anotaci, NLP úlohu pro anotaci a přidělí scénář anotátorům. Pro přidělení scénářů je využita třída WorkFrom. Po přidělení scénáře se přejde na jednu ze šablon v balíčku *views.admin.CreateScenario*, která je vybrána na základě NLP úlohy, která je anotována. V této šabloně se upřesňují prvky scénáře, které jsou specifické pro jednotlivé NLP úlohy (viz obr. 5.4).

5.2.2 Export scénáře

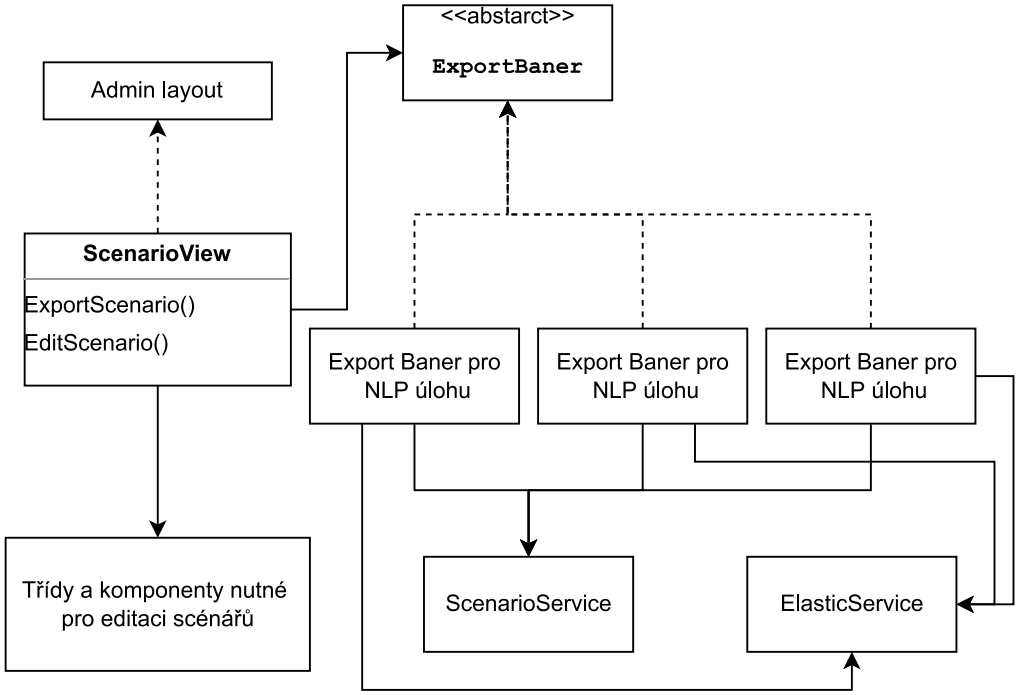
Export scénáře musí být opět možný pro všechny NLP úlohy. Každá NLP úloha může mít svůj specifický typ exportu (např. CSV, BIO Label 7 nebo JSON), proto použiji společný layout, ve kterém se nachází tlačítko *"export"*, po jehož stisknutí se zobrazí baner s možnostmi exportu. Baner bude odlišný mezi jednotlivými NLP úlohami. Všechny třídy reprezentující banery pro export, dědí od třídy *"ExportBaners"*, tímto způsobem je zajištěna přehlednost a funkčnost kódu (viz obr. 5.5).



Obrázek 5.3: Obrazovka základního nastavení, stejná pro všechny NLP úlohy



Obrázek 5.4: Obrazovka specifického nastavení pro dané NLP úlohy



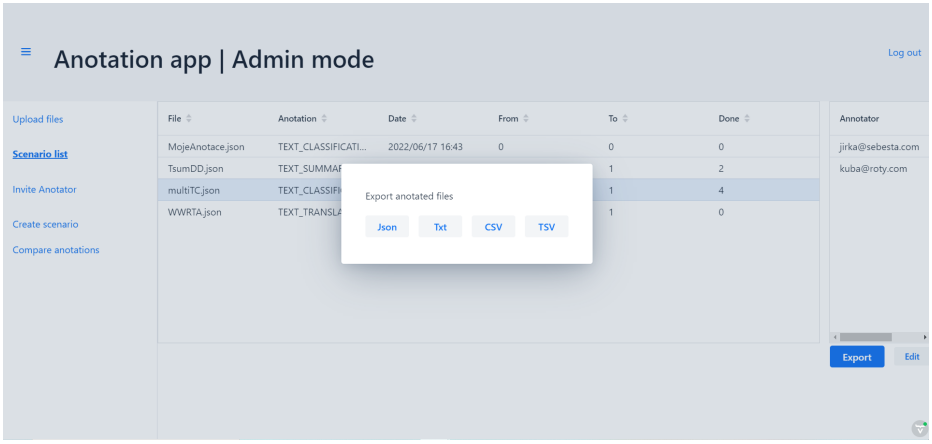
Obrázek 5.5: UML-diagram exportování scénáře.

Pro export scénáře je využita šablona *views.admin.ScenarioView.java*, ve které uživatel vybere scénář pro export. Po stisknutí tlačítka *export* se zobrazí baner specifický pro každy typ NLP úlohy. Tyto banery jsou reprezentovány třídami v balíčku *components.admin.exportBanners* (viz obr. 5.6).

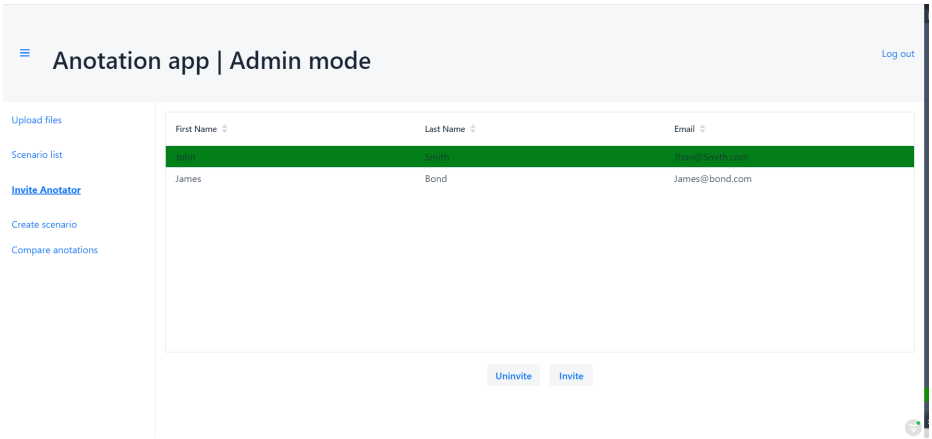
5.2.3 Pozvání anotátora

Aby admin mohl zadat anotátorovi scénář, musí být anotátor přítomen v jeho pracovní skupině. Každý anotátor může být součástí několika pracovních skupin. V šabloně *AddAnotatorView* jsou v tabulce zobrazeni všichni anotátóři, kteří nejsou přítomni v administrátorově pracovní skupině. Každého anotátora lze pozvat, nebo lze pozvání zrušit.

Pro pozvání anotátora do pracovní skupiny je využita šablona *views.admin.AddAnotatorView.java*. V této šabloně jsou použity základní komponenty, kterými uživatel ovládá aplikaci a může pozvat anotátora (viz obr. 5.9).



Obrázek 5.6: Obrazovka exportu datové sady



Obrázek 5.7: Obrazovka pozvání anotátora do skupiny

5.2.4 Importování datové sady

Administrátorovi musí být umožněno importovat čisté datové sady. Tato funkcionalita je mu zpřístupněna díky pohledu *views.admin.UploadView.java*. V tomto pohledu se nachází prvek pro import JSON souborů do aplikace a metoda pro následné zpracování JSON souboru a uložení jej do nerelační databáze.

5.2.5 Porovnání anotovaných datových sad

Administrátorovi je umožněno porovnat anotované datové sady a zjistit v jakých prvcích se anotátoři liší. Pohled pro tuto funkcionalitu je *views.admin.CompareView.java*. Zde se náchazí komponenta pro porovnání anotace datové sady. Tato komponenta ukáže odlišnosti v anotaci a vypočítá číslo odlišnosti κ [30] .

Porovnání je možné pouze pro NLP úlohu klasifikace textu ve verzi single select (klasifikace pouze jedné třídy). Důvodem je výpočet čísla shody mezi anotátory (inter-annotator agreement Cohen's Kappa), jehož výpočet je implementovaný pouze pro tento typ úloh. Vzorec pro výpočet čísla κ je

$$\kappa = \frac{(P_0 - P_e)}{(1 - P_e)}$$

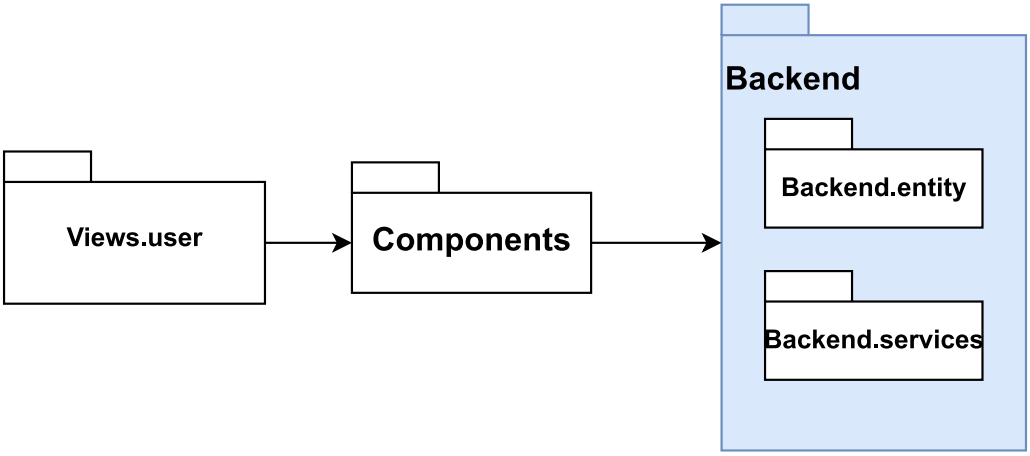
.

Kde P_0 je relativní pozorovaná shoda mezi hodnotiteli a P_e je hypotetická pravděpodobnost náhodné shody.

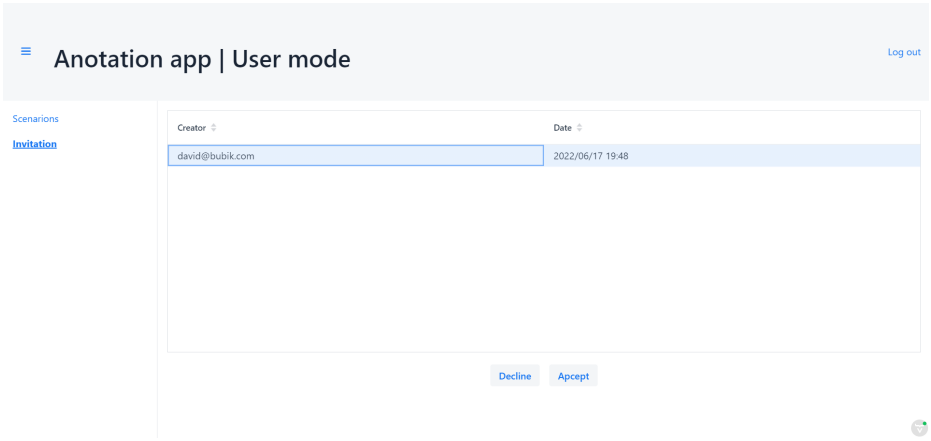
5.3 Implementace anotátorské části

V této podkapitole popíši implementaci anotátorské části aplikace. Podle cílů práce (viz 1.1) musí být anotátorovi umožněno vypracovat zadané scénáře a přijmout pozvání od administrátora do pracovní skupiny. UML diagram 5.10 znázorňuje implementaci anotátorské části na úrovni balíčků.

Jednotlivé třídy, které jsou v balíčku *views.user* jsou zodpovědné za zobrazení šablon pro jednotlivé anotátorské funkcionality. V těchto šablonách jsou zobrazeny komponenty z *components.user*, kterými uživatel ovládá aplikaci. Šablony i komponenty využívají služby pro komunikaci s databázemi z balíčku *backend.service*. Pro anotaci jednotlivých NLP úloh jsou využity šablony z balíčku *views.anotations*.



Obrázek 5.8: UML diagram balíčků anotátorské části.



Obrázek 5.9: Obrazovka přijmutí pozvání od anotátora

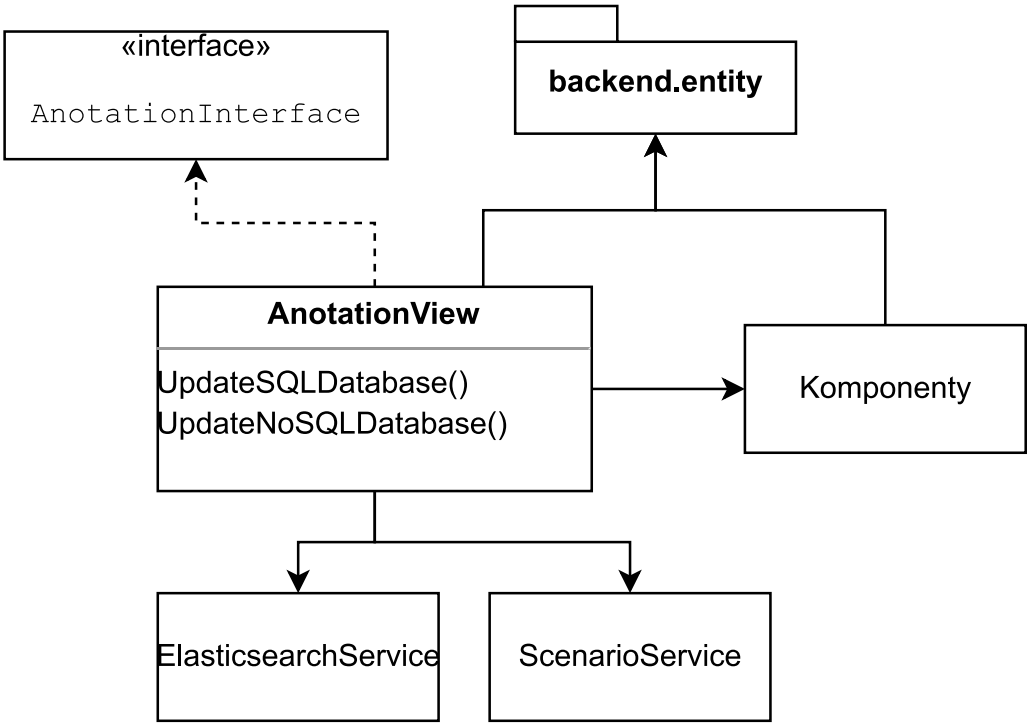
5.3.1 Přijmutí pozvání od admina

V pohledu *InvitationView* má administrátor k dispozici tabulku, ve které jsou zobrazeny pozvánky, které může akceptovat nebo odmítnout. *InvitationView* pro základní rozložení komponent implementuje *anotatorLayout* a jednotlivé komponenty za pomoci obsluhy události registrují uživatelské vstupy. Tyto vstupy se využitím tříd v balíčku *Backend.services* uloží do databází.

5.3.2 Vypracování scénáře

Každá NLP úloha má svůj vlastní pohled, který se nachází v balíčku *views.anotations*. Všechny tyto pohledy implementují společný interface. Pokud mají NLP úlohy stejný typ anotace (např. strojový překlad a sumarizace textu), je možné, aby jeden pohled dědil od druhého. Podle údajů vybraného

scénáře, na počátku načte aplikace z Elasticsearch databáze celou datovou sadu. Při každé anotaci jednotlivého prvku je změna uložena v Elasticsearch databázi, postup je zaznamenán do MySQL databáze a načtená data jsou také upravena. Obrázek 5.11 ukazuje průběh anotace NLP úlohy klasifikace textu.



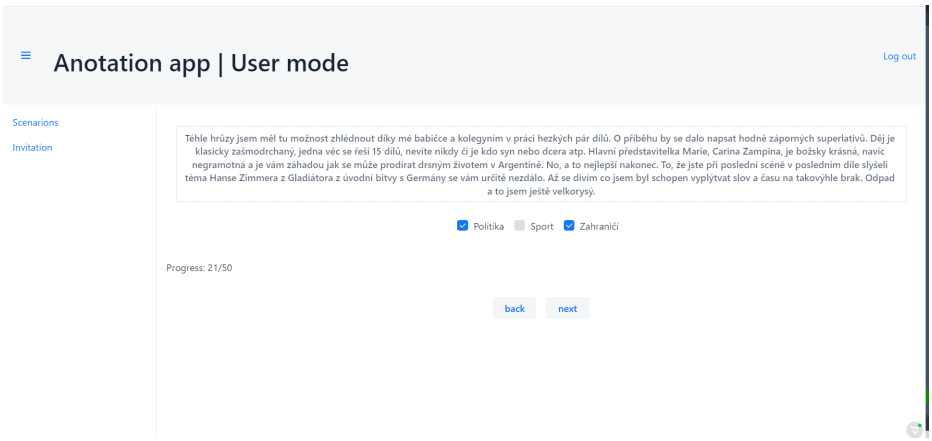
Obrázek 5.10: UML diagram vypracování scénáře.

5.4 Příklad implementace další NLP úlohy

Dalším z cílů práce (viz 1.1) je snadná rozšiřitelnost systému o další NLP úlohy. Jako vysvětlující příklad jsem vybral implementaci NLP úlohy sumarizace textu, za předpokladu, že máme již implementovanou úlohu strojového překladu. Celá implementace nové úlohy se skládá z pěti kroků.

5.4.1 Implementace sumarizace textu

Jak bylo výše uvedeno, implementace nové úlohy se skládá z pěti kroků. Prvním z nich je do souboru *anotApp.enumn.Annotations.java* přidat nový záznam a k němu alias, unikátní pro novou NLP úlohu. V tomto případě přidávám *TEXT_SUMMARIZATION* a alias *Text sumarization*.



Obrázek 5.11: Obrazovka anotace NLP úlohy klasifikace textu (multilabel)

Druhým krokem je vytvořit v balíčku *ui.views.anotations* nový pohled, specifický pro novou NLP úlohu. Tento pohled bude sloužit pro anotaci NLP úlohy a musí implementovat rozhraní *AnotationInterface*. Může samozřejmě dědit od kteréhokoli již vytvořeného anotačního pohledu. Pro implementaci sumarizace textu budu dědit od pohledu *TextTranslation*.

Třetím krokem je vytvoření pohledu pro založení scénáře. Tento krok je rozdělen na dva podúkoly. Nejprve je nutné v souboru *views.admin.CreateScenarios.AnnotView.java* upravit switch case větev tím, že do ní přidáme novou podmínku s názvem (enum) nové NLP úlohy. Dále je nutné si vybrat, jaký typ scénáře zakládáme. Jsou předpřipraveny tři třídy: Basic, který nedovoluje přidání žádné klasifikační sady, OneClass, který dovoluje přidání jedné klasifikační sady a TwoClass, který dovoluje přidat dvě klasifikační sady. Pokud již implementované třídy nejsou vyhovující, je možné vytvořit novou, je ale důležité, aby dědila od abstraktní třídy *ScenarioCreators*. Konstuktor vybrané třídy vložím do nově přidané podmínky ve switch case větvi. V případě implementace sumarizace textu použiji třídu Basic, protože při anotaci této NLP úlohy není vybrání třídy ze sady tříd používáno.

Začtvrté je potřeba umožnit editaci již vytvořeného scénáře s novým typem NLP úlohy. Nejdříve je důležité přidat do switch case větve v souboru *views.admin.EditScenarios.EditView.java* novou podmínku s názvem (enum) nové NLP úlohy. Dále se musí zvolit, jaký typ editace je vhodný pro úlohu, kterou implementuji. Jsou předpřipraveny tři třídy: BasicEdit, který nedovoluje editování žádné klasifikační sady, OneClassEdit, který dovoluje editování jedné klasifikační sady, a TwoClassEdit, který dovoluje editování dvou klasifikačních sad. Pokud již implementované třídy nejsou vyhovující je možné vytvořit novou, je ale nutné, aby dědila od abstraktní třídy *ScenarioEdit*.Konstuktor vybrané třídy vložím do nově přidané podmínky ve

switch case větvi. Při implementaci NLP úlohy sumarizace textu jsem pro reprezentaci editace zvolil třídu BasicEdit, protože při anotaci sumarizace textu se nepoužívají žádné klasifikační sady.

Poslední krokem je zpřístupnit exportování vypracovaného scénáře s novou NLP úlohou. Uživatel exportuje vypracovaný scénář za pomocí "baneru", na kterém vybere formát, v němž se oanotovaná datová sada vyexportuje. Veškeré editace kódu budou tedy probíhat v balíčku *components.admin*. Nejprve je nezbytné v souboru *EditScenarioComponent.java* upravit switch case větev, která se nachází v metodě *setUpDialog()*. Do této větve přidáme novou podmínku s názvem (enum) nové NLP úlohy. Dále je zapotřebí zvolit jaký baner pro export bude použit, protože každá NLP úloha může exportovat datové sady do jiných formátů. Je možné si vybrat jeden z pěti již implementovaných banerů, které se nacházejí v balíčku *components.admin.exportBaners*, nebo si vytvořit vlastní baner. Při vytvoření vlastního baneru je ale nutné, aby třída reprezentující tento baner dědila od abstraktní třídy *ExportBaner*. Konstruktor třídy reprezentující vybraný baner vložím do nově vytvořené podmínky v switch case větvi. Při implementaci NLP úlohy sumarizece textu jsem zvolil stejný baner pro export, jako pro NLP úlohu strojového překladu.

6 Testování anotační aplikace

V této kapitole popíši, jakým způsobem jsem testoval funkčnost mého systému. Testování probíhalo ve dvou částech, paralelně s vývojem byly použity jednotkové testy spolu s průběžným testováním funkcionality a finální testování testery (běžnými uživateli), kteří dostali k dispozici finální verzi aplikace a testovali její funkcionalitu. Mezi nejdůležitější části aplikace patří: import datové sady, vytvoření scénáře, anotace datové sady podle vytvořeného scénáře a export oanotované datové sady.

6.1 Průběh testování

Testování aplikace bylo prováděno převážně mnou a mým vedoucím práce. Zkušební datovou sadu pro anotaci poskytl vedoucí práce.

6.1.1 Jednotkové testování

Jednotkové testování bylo použito pro výpočetní části aplikace. Jedním z příkladů je testování třídy *CompareComponent.java*, která má za úkol vypočítat shodu mezi jednotlivými anotátory, nebo test správného řazení mnou vytvořených entit (např. scénářů). Jednotlivé testy jsou v adresáři *src.test.java.anotApp*.

6.1.2 Testování testery

Finální aplikace byla poskytnuta testerům, kteří ověřovali její funkcionalitu a robustnost. Někteří z testerů měli základní znalost z oblasti IT (informační technologie) a jiní byli zaměřeni na jiné obory. Byly otestovány i různé nestandardní události, např. přístup na jednotlivé části aplikace za pomocí URL adresy a nikoli za pomocí aplikace samotné.

K dipozici bylo třicet příkladů, které byly anotovány pro každou NLP úlohu. Jednotliví testeři měli za úkol vytvořit scénáře, upravit je a provést anotaci všech NLP úloh.

6.2 Výsledky testování

Během testování bylo zjištěno, že editace některých scénářů, není možná. Aplikace není schopná načíst obrazovku pro editaci a vypíše chybovou zprávu.

Problémem bylo to, že všechny typy scénářů se upravovaly pomocí jednoho komponentu. Tento problém se vyřešil implementací několika dalších komponent, kde každá z nich je zodpovědná za editaci jednoho typu scénáře.

Druhý zjištěný nedostatek se týkal opakované anotace jednoho scénáře. Pokud anotátor vypracoval celý scénář, nebylo možné ho znovu otevřít a upravit stávající anotaci. Tento problém byl způsoben tím, že v informacích o scénáři, které jsou uloženy v databázi, se nachází ID posledního anotovaného textu, v případě, že byla v půlce scénáře anotace přerušena, bylo možné se vrátit zpět na poslední anotované místo. V případě, že jste vypracoval celý scénář, bylo poslední zapamatované místo konec scénáře. Řešením bylo po vypracování scénáře ukazatel vynulovat.

Posledním zmíněným problémem, který byl odhalen během testování, byla přítomnost uvozovek v anotovaném textu. Tento problém byl zapříčiněn špatným parsováním (zpracováním) importovaného JSON souboru. Tento nedostatek byl vyřešen použitím speciálních znaků, které dané uvozovky oddělovaly od uvozovek, které separovaly jednotlivé texty.

7 Závěr

Hlavním cílem této bakalářské práce bylo navrhnout, vytvořit a ověřit funkčnost anotační aplikace, která implementuje nejméně tři anotační úlohy a je snadno rozšiřitelná. V rámci práce jsem naimplementoval anotační aplikaci a tento cíl splnil.

Anotační aplikace slouží ke snadnému označování (anotování) datových sad. Typy anotace se liší mezi jednotlivými NLP úlohami a různé aplikace mají implementovány jiné NLP úlohy. Výsledné datové sady se používají pro učení modelů strojového učení.

V této práci jsem nejdříve prozkoumal jednotlivé úlohy z oblasti NLP a vybral tři, které mají rozdílný typ anotace. Dále jsem prozkoumal již existující anotační aplikace, porovnal jejich vlastnosti a nenalezl jsem žádnou, která by vyhovovala všem požadavkům z cílů práce. V následujícím kroku jsem provedl návrh mé aplikace, její funkčnost, architekturu a vzhled. Mnou vyvinutá apliakce umožňuje anotaci více než tří typů NLP úloh, je snadno rozšiřitelná a obsahuje role anotátora a administrátora. Navržený systém jsem implementoval a otestoval jeho funkčnost.

Anotační aplikaci lze snadno rozšířit o další NLP úlohy (viz kap. 5.4) a byla otestována vybranými uživateli. Tento systém představuje vhodný způsob pro vytváření anotovaných datových sad. Všechny body zadání bakalářské práce byly splněny.

Seznam použitých zkratek a výrazů

Anotace Proces označování datových sad.

NLP Zpracování přirozeného jazyka (Natural language procesing). Odvětví informatiky zaměřené na zpracování přirozeného jazyka.

JSON JavaScript-Object-Notation je datový formát využitý pro přenos dat.

CSV Coma-seperated-vaules je datový formát využitý pro přenos dat.

BIO label Formát exportovaného textu využíván pro úlohu NER. Principem je, že pokud se entita skládá z více prvků (př. Marie Curie-Skłodowská) je první prvek entity označen *B* a zbytek *I*. Prvky, které nejsou součástí žádné entity jsou označeny *O*.

AI Umělá inteligence (Artifical Inteligence).

ML Strojové učení (Machine learning). Jedná se o jeden z podoborů AI.

Supervised ML Strojové učení s učitelem. Odvětví strojového učení, ve kterém se pro naučení umělé inteligence využívají anotované datové sady.

Unsupervised ML Strojové učení bez učitele. Odvětví ML, které nevyžaduje předem anotované datové sady.

DL hluboké učení (Artifical Inteligence). Jedná se o jeden z podoborů AI.

NER Named entity recognition je úloha z oblasti NLP.

NE Named entity jsou jedntolivé části textu, které představují vlastní jména, pojmeování, atd.

ABSA Aspect-Based Sentiment je úlohou z oblasti NLP.

UML Unified Modeling Language je jazyk pro vizuální modelování systémů.

SQL Structured Query Language je jazyk pro práci s relačními databázemi.

SQL databáze Databáze, které využívají jazyk SQL, také známé jako relační databáze.

NoSQL databáze Nerelační databáze.

MVC Model-view-controler je architektura pro vytváření webových aplikací.

IT Informační technologie.

Literatura

[1] AI, Machine Learning (ML) and Natural Language Processing (NLP). Dostupné z: https://athenatech.tech/f/ai-machine-learning-ml-and-natural-language-processing-nlp.

[2] Apache Cassandra usage. Dostupné z: https://ubuntu.com/blog/apache-cassandra-top-benefits.

[3] IntelliJ IDEA. Dostupné z: https://www.jetbrains.com/idea/.

[4] Labelbox. Dostupné z: https://labelbox.com.

[5] MySQL. Dostupné z: https://www.mysql.com/.

[6] Oracle database. Dostupné z: https://www.oracle.com/index.html.

[7] Postgres-about. Dostupné z: https://www.postgresql.org/about/.

[8] Prodigy. Dostupné z: https://prodi.gy.

[9] Spring web aplications. Dostupné z: https://spring.io/web-applications.

[10] Tagtog. Dostupné z: https://www.tagtog.net/.

[11] Vaadin, . Dostupné z: https://github.com/vaadin.

[12] VaadinWithSpring, . Dostupné z: https://vaadin.com/docs/v14/flow/integrations/spring/tutorial-spring-basic.

[13] Framework-Code Institute. Dostupné z: https://codeinstitute.net/global/blog/what-is-a-framework/.

[14] Spring. Dostupné z: https://github.com/spring-projects.

[15] Müller Miroslav, H. K. *UML srozumitelně*. 1. Brno: Computer Press, 2004.

[16] Banon, S. Elasticsearch, 2010. Dostupné z: https://www.elastic.co/.

[17] Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer, 2006.

[18] Christopher D. Manning, H. S. *Foundations of Statistical Natural Language Processing - Christopher D. Manning*. Massachusetts Institute of Technology, 1999.

[19] Eisenstein, J. Natural language processing, 2018.

[20] Jatana, N. et al. A survey and comparison of relational and non-relational database. *International Journal of Engineering Research & Technology.* 2012, 1, 6, s. 1–5.

[21] Jurafsky, D. – Martin, J. H. *Speech and Language Processing.* https://web.stanford.edu/ jurafsky/slp3/ed3book_jan122022.pdf, 2021.

[22] Karatas, G. Named Entity Recognition (NER): What It Is & How It Is Used. 2022. Dostupné z: `https://research.aimultiple.com/named-entity-recognition/`.

[23] Kowsari, K. et al. Text Classification Algorithms: A Survey. *Information.* 2019, 10, 4. ISSN 2078-2489. doi: 10.3390/info10040150. Dostupné z: `https://www.mdpi.com/2078-2489/10/4/150`.

[24] Lakshman, A. Apache Cassandra, 2008. Dostupné z: `https://cassandra.apache.org/_/index.html`.

[25] Liu, B. *Sentiment Analysis and Opinion Mining.* Morgan & Claypool Publishers, 2012.

[26] Nakayama, H. et al. doccano: Text Annotation Tool for Human, 2018. Dostupné z: `https://github.com/doccano/doccano`. Software available from https://github.com/doccano/doccano.

[27] Pannala, N. U. et al. Supervised Learning Based Approach to Aspect Based Sentiment Analysis. In *2016 IEEE International Conference on Computer and Information Technology (CIT)*, s. 662–666, 2016. doi: 10.1109/CIT.2016.107.

[28] Pontiki, M. et al. SemEval-2014 Task 4: Aspect Based Sentiment Analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, s. 27–35, Dublin, Ireland, August 2014. Association for Computational Linguistics. doi: 10.3115/v1/S14-2004. Dostupné z: `https://aclanthology.org/S14-2004`.

[29] Sarkis, A. diffgram: Text Annotation Tool for Human, 2021. Dostupné z: `https://github.com/diffgram/diffgram`. Software available from https://github.com/diffgram/diffgram.

[30] Warrens, M. Kappa coefficients for dichotomous-nominal classifications. *Advances in Data Analysis and Classification.* 04 2020, 15. doi: 10.1007/s11634-020-00394-8.

[31] Yadav, V. – Bethard, S. A Survey on Recent Advances in Named Entity Recognition from Deep Learning models. In *Proceedings of the 27th International Conference on Computational Linguistics*, s. 2145–2158, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. Dostupné z: `https://aclanthology.org/C18-1182`.

[32] Zhang, W. et al. A Survey on Aspect-Based Sentiment Analysis: Tasks, Methods, and Challenges. 03 2022.

Seznam příloh

A Uživatelská příručka

Uživatelská příručka obsahuje základní informace o aplikaci, tj. co aplikace umožňuje, popis jejího spuštění a ovládání.

A.1 Spuštění systému

Aplikace se spouští ve dvou krocích. Nejdříve je nutné spustit databáze, do kterých aplikace ukládá data, následně je možné spustit aplikaci samotnou.

A.1.1 Spuštění databází

Pro stažení správné verze a spuštění databází Elasticsearh a Mysql je použit software Docker, který je potřeba mít nainstalovaný. Databáze se spouští z příkazové řádky za pomocí scriptu *anotApp.yaml* příkazem *docker-compose -f anotApp.yaml up*. Tento scrip nastaví komunikaci s Mysql databází na portu 6200 a Elasticsearch na portech 6000 a 6100. Spuštění databází může trvat až několik minut v závislosti na výkonnosti použitého PC.

A.1.2 Spuštění aplikace

Aplikace je exportovaná ve formátu jar. Spouští se z příkazové řádky za pomocí příkazu *java -jar anotApp.jar*. Po spuštění komunikuje na portu 49155. Spuštění aplikace bylo testováno na platformě Windows 10 a Ubuntu, s verzí javy 16.0.2

A.2 Ovládání aplikace

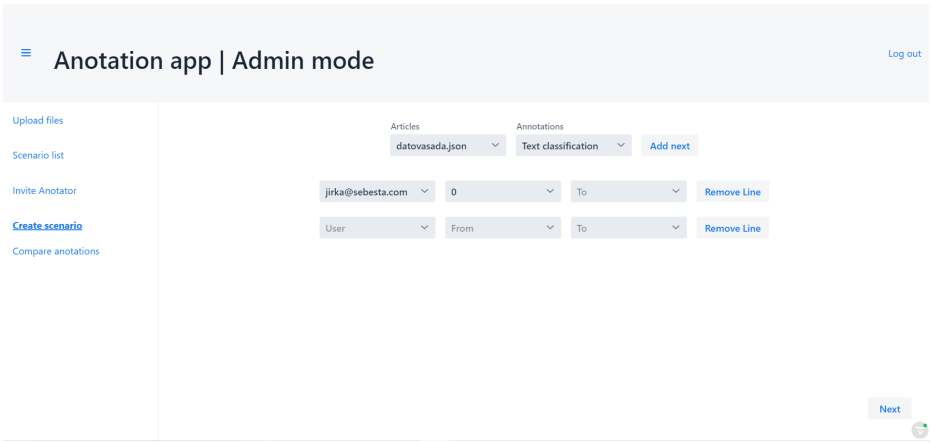
Zde popíši jak ovládat mnou vytvořenou aplikaci. Nejdříve ukáži ovládání administrátorské části a následně anotátorské.

A.2.1 Registrace/Přihlášení

Po spuštění aplikace je na adrese *http://localhost:49155/* (pokud jste na stejném zařízení, kde byla aplikace spuštěna) přístupná přihlašovací obrazovka. Zde si můžete vybrat, zda se chcete přihlásit nebo zaregistrovat jako nový

```
[
 {
    "id":0,
    "text":"Zde je text, který se bude anotovat",
    "source":"Zde je zdroj odkud je tento text čepán",
    "date":"Zde je datum kdy byl text získán"
  }
]
```

Obrázek A.1: Struktura importované datové sady.



Obrázek A.2: Obrazovka pro základní návrh scénáře.

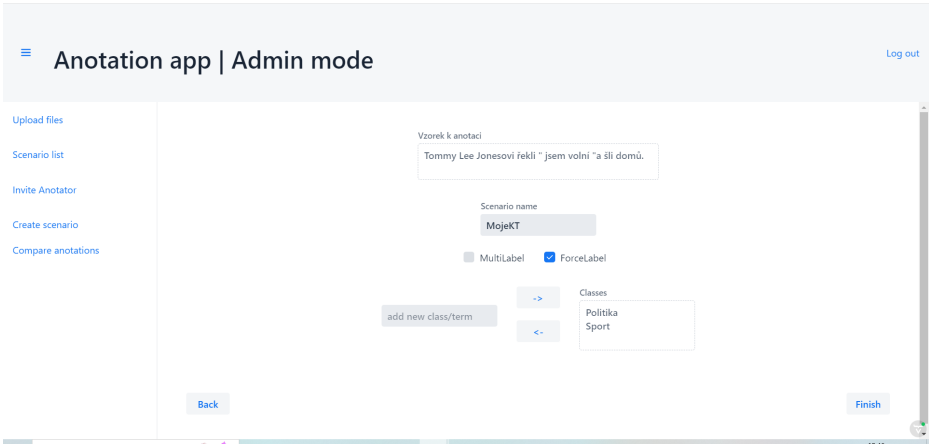
uživatel. Při registraci nového uživatele je nutné si vybrat, zda má uživatel být administrátor nebo anotátor. Předem připraveni jsou čtyři uživatelé, jejich přihlašovací údaje naleznete v souboru *Vstupni_data\uzivatele.txt*.

A.2.2 Upload datové sady

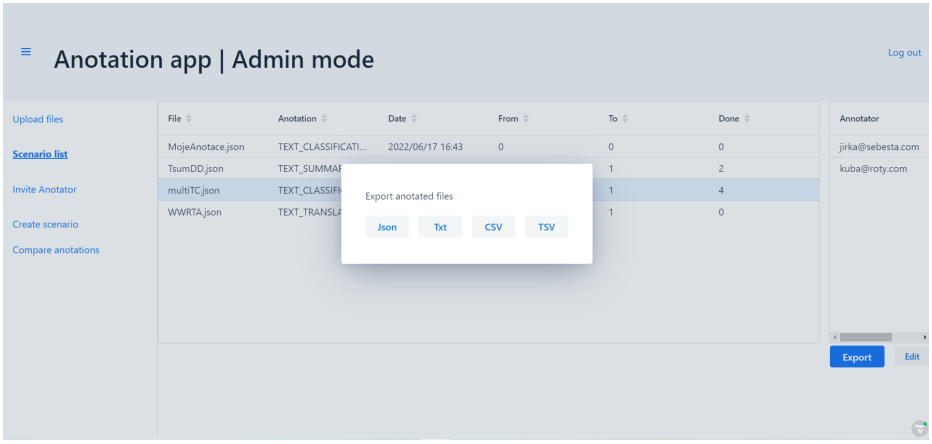
Formát datové sady pro nahrání musí být JSON se strukturou znázorněnou na obrázku A.1. Pro upload zvolte na levé liště možnost *Upload files* a pomocí komponentu pro nahrání souborů vyberte datovou sadu, kterou chcete nahrát.

A.2.3 Vytvoření scénáře

Pro založení scénáře je nutné se přihlásit jako administrátor. Po přihlášení je na levé straně lišta s jednotlivými dostupnými funkcionalitami, pro založení scénáře je nutné vybrat položku *Create scenario*. Po vybrání se objeví obrazovka pro základní návrh scénáře (viz obr. A.2), na které si zvolíte datovou sadu, NLP úlohu a anotátory pro scénář. Po stisknutí tlačítka *next* jste



Obrázek A.3: Obrazovka pro detailní návrh scénáře.



Obrázek A.4: Obrazovka pro export scénáře.

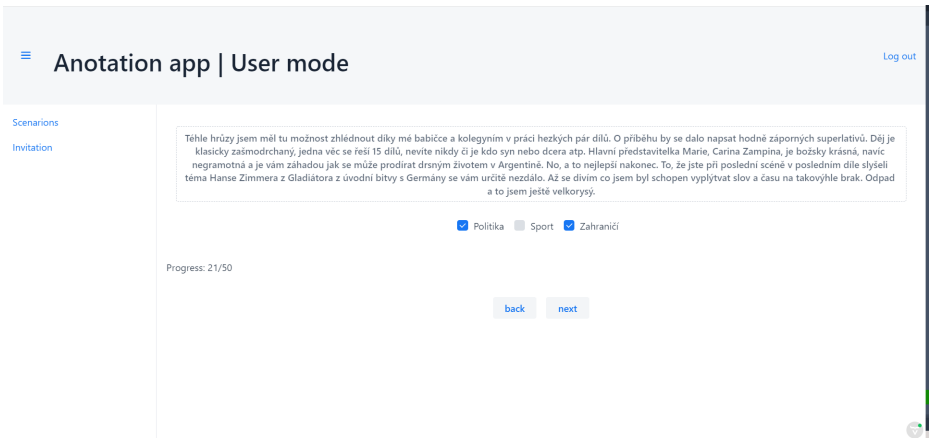
přesměrováni na obrazovku umožňující detailnější návrh scénáře (viz obr. A.3), kde se přidávají detaily scénáře specifické pro daný typ NLP úlohy. Po stisknutí tlačítka *finish*, je scénář úspěšně vytvořen.

A.2.4 Pozvání anotátora do skupiny

Pro pozvání anotátora do skupiny vyberte na levé liště možnost *Invite Anotator*, zvolte anotátora, kterého chcete pozvat a stiskněte tlačítko *invite*.

A.2.5 Export scénáře

Pro exportování scénáře na levé liště zvolte možnost *Scenario list*. Vyberte scénář, který chcete exportovat a stiskněte tlačítko export. Následně se objeví baner, na kterém vyberete formát exportu (viz obr. A.4).



Obrázek A.5: Obrazovka anotace úlohy klasifikace textu.

A.2.6 Porovnání anotací vypracovaného scénáře

Pro porovnání anotace jednotlivých anotátorů, musí vypracovaný scénář být: anotovaný alespoň dvěma anotátory, NLP úloha musí být *Klasifikace textu* a typ anotace musí být single label. Pokud scénář splňuje výše uvedené požadavky, lze srovnat anotaci mezi anotátory.

Pro srovnání anotace vyberte na levé liště možnost *Compare anotations*, zvolte scénáře anotátory a stiskně tlačítko *Compare*.

A.2.7 Vypracování scénáře

Pro vypracování scénáře je nutné se přihlásit jako anotátor. Po přihlášení na levé liště zvolit *Scenarios*, vybrat scénář a klinout na tlačítko *Start*. Následně se objeví obrazovka, na které probíhá anotace (viz obr. A.5).

A.2.8 Přijmutí pozvání do anotační skupiny

Pokud chcete přijmout pozvání od administrátora, na levé liště zvolte možnost *Invitation*, vyberte pozvánku a klikněte na tlačítko *Apcept*.

B Struktura přiloženého souboru

- `Aplikace_a_knihovny` Tento soubor obsahuje spouštěcí scripty aplikace, knihovny a zdrojové soubory
  - `frontend` ..... Tento adresář obsahuje css soubory
  - `javadoc` ..... Tento adresář obsahuje javadoc dokumentaci
  - `node_modules` ..... Tento adresář obsahuje externí knihovny
  - `src` ..... Tento adresář obsahuje zdrojové soubory
- `Text_prace` ..... Tento adresář text bakalářské práce
- `Vstupni_data` ..... Tento adresář vstupní datovou sadu a informace o předpřipravených uživatelích
- `readme.txt` ..... Stručný popis obsahu souboru