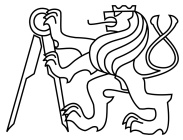
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Open-source crowdsourcingová aplikace
Student:	Bc. Tomáš Marek
Vedoucí:	Ing. Milan Dojčinovski
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Crowdsourcing, jako model pro řešení složitých úkolů, se v posledních letech hodně rozšířil. Možnost použití open-source crowdsourcing nástrojů je však limitováno. Hlavním cílem diplomové práce je vytvořit open-source crowdsourcing framework. Nástroj bude realizován jako webová aplikace v programovacím jazyce Node.js. Aplikace umožní uživatelům: vytvořit úkol, sdílet ho s jinými uživateli, sledovat průběh zpracování úkolu, sesbírat a agregovat řešení.
Pokyny:
- Seznamte se s hlavními principy crowdsourcingu.
- Seznamte se s aktuálním stavem a proveďte rešerši.
- Na základě analýzy navrhněte architekturu crowdsourcing frameworku.
- Implementujte a otestujte nově navržený framework.
- Ověřte funkčnost aplikace na několika reálných scénářích.
- Vyhodnoťte výsledky.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 12. prosince 2015

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Diplomová práce

Open-source crowdsourcingová aplikace

Bc. Tomáš Marek

Vedoucí práce: Ing. Milan Dojčinovski

4. května 2016

Poděkování

Chtěl bych poděkovat panu Ing. Milanovi Dojčinovskému za návrh velice zajímavého tématu a za rady a připomínky, které mně poskytoval během vypracování této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 4. května 2016 .....

Odkaz na tuto práci

Marek, Tomáš. *Open-source crowdsourcingová aplikace.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Tato práce se zabývá možnostmi použití techniky zvané crowdsourcing pro získávání informací a dat. Po analýze této metody jsou zkoumány některé již existující webové nástroje. Na základě výhod a nevýhod jednotlivých řešení je navržena a implementována vlastní aplikace. Možnosti výsledného systému jsou pak ukázány na několika konkrétních případech.

Klíčová slova Crowdsourcing, Webová aplikace, Node.js, MongoDB, API

Abstract

This diploma thesis deals with the use of crowdsourcing for retrieving information and data. After studying this method, some existing tools are analyzed. On the basis of the pros and cons of these solutions, a new application is designed and implemented. The possible results are demonstrated in few concrete use cases.

Keywords Crowdsourcing, Web application, Node.js, MongoDB, API

Obsah

Seznam obrázků

Úvod

V současné době jsme každý den obklopeni novými informacemi, které se k nám dostanou, ať chceme nebo ne. Díky relativně novým prostředkům, jako je internet, je množství dat, na které narazíme, větší než kdykoli předtím. Některé získané údaje jsou pro nás poměrně nepodstatné a bez větších starostí je můžeme zapomenout. Na druhou stranu existují informace, které mají obrovskou hodnotu. V době počítačů jsme si zvykli, že tyto stroje za nás provádějí mnoho práce v širokém spektru činností a manipulace s daty není výjimkou. Díky výkonnějšímu hardwaru a chytřejším algoritmům lze dnes automatizovaně provádět úkony, které před několika lety nebyly možné. Kromě klasické správy informací, jako je jejich ukládání, vyhledávání, řazení, mazání a další, dokážou počítače z uložených záznamů získat mnoho dalších údajů. Existují aplikace, které dokáží detekovat obličej ve fotografii, přečíst text ve videu nebo obrázku, vyhledávat podobně znějící hudbu, dokonce je možné z příspěvku na sociální síti analyzovat, zda byl míněn spíše pozitivně nebo negativně. Při získávání informací ovšem stále existují úkoly, které počítače nemohou splnit, popřípadě to dokáží pouze s omezenou přesností. V takových případech musí požadované údaje poskytnout člověk. Příkladem může být překlad textu do jiného jazyka, který při provádění strojem prakticky nikdy není dokonalý a obsahuje gramatické chyby. Pokud se jedná o malé množství dat, může daný úkol splnit sám majitel informací, popřípadě může tuto úlohu delegovat na své podřízené či kolegy. Při velkém množství záznamů však tato práce bude trvat dlouho a může být finančně poměrně nákladná. Řešením může být crowdsourcing, kdy se na plnění úkolu podílí velké množství dobrovolníků, z nichž každý přispěje relativně malým dílem, ale ve výsledku získáme dostatečné množství údajů.

Cíle práce

Při práci s daty různého typu můžeme chtít získat informace, které nelze extrahovat automatizovaným způsobem. Jedná se například o definici klíčových slov k obrázku, přepis audio nahrávky do textové podoby, překlad dokumentů do jiného jazyka a podobně. Takových úkolů, pro které doposud nebyly implementovány spolehlivé algoritmy, existuje velké množství. Získané údaje obohatí data, což může být potřebné nejenom ve vědecké a vzdělávací sféře, ale také v oblasti komerční. Internetové obchody mohou chtít získat doplňující fakta o svých produktech a zákaznících, lokalizovat aplikaci pro jinou zemi nebo napsat popis produktu tak, aby byl pro zákazníka čtivější, a tím pádem koupě produktu atraktivnější. Firmy také mohou pomocí sentimentální analýzy zjišťovat, jaké názory na jejich služby a produkty kolují po sociálních sítích a názorových fórech. V akademické sféře, nejenom že můžeme získávat zcela nová data, ale dobrovolníci také mohou ohodnotit výsledky algoritmů, a tím přispět k jejich zlepšení. Jednou z velkých oblastí, kterými se mnoho lidí v současné době zabývá, je text mining, což je proces získávání rozsáhlých údajů o textech a jeho částech. Může se jednat o gramatické vlastnosti výrazu, nebo také o význam slov či celých vět. Mnoho úkolů, které se řadí do text miningu, využívá právě taková data, která nelze získat strojovou analýzou. Jestliže zkoumáme nějaký text, často se využívá různých slovníků, které obsahují potřebné informace o daných entitách v konkrétním jazyce. Použité záznamy ovšem musely vznikat dlouhodobou lidskou činností. Pro rychlejší a efektivnější získávání potřebných informací lze použít crowdsourcing, který využívá velkého množství lidí k provedení daného úkolu.

Cíl práce je analyzovat možnosti použití crowdsourcingu při získávání informací. Nejdříve je třeba vyhodnotit, co přesně tato technika obnáší, jaké jsou její výhody a nevýhody a v jakých případech se vyplatí její nasazení. Jelikož by mělo řešení sloužit především k akademickým účelům, jeho použití by mělo být bezplatné a kvůli velké rozmanitosti požadovaných úloh by měla být aplikace snadno rozšiřitelná. V dalším kroku se pokusíme analyzovat několik existujících řešení, jaké nabízejí možnosti a zda vyhovují našim požadavkům. Dále bude navržena vlastní aplikace a implementována pomocí moderních vývojových prostředků, které zaručí její rychlost a spolehlivost. Výsledná aplikace bude následně otestována na různých úrovních a nakonec budou její možnosti demonstrovány na několika konkrétních případech.

Struktura práce

Celý text je členěn do několika kapitol, které jsou za sebou seřazeny tak, aby byl jasný postup vytvoření crowdsourcingové aplikace. V části „O crowdsourcingu“ se zaměříme na popis této techniky sběru informací, její výhody a nevýhody. V sekci „Analýza existujících řešení“ se podíváme na několik již

existujících aplikací, které by mohly být použitelné pro data mining, a opět zhodnotíme klady a zápory jednotlivých systémů. Kromě toho také získáme lepší povědomí o tom, jak aplikace určené pro crowdsourcing vypadají, a na základě těchto poznatků můžeme přistoupit k návrhu vlastního řešení. Kapitola „Analýza a návrh“ se bude zabývat právě analýzou a návrhem nové aplikace. Zde bychom měli získat lepší teoretickou představu o tom, co všechno systém bude a nebude vykonávat. V podsekci „Návrh“ poté bude proveden základní popis informací a datových struktur, kterými se budeme muset v systému zabývat. Nakonec zde budou také popsány použité nástroje. V kapitole „Implementace“ se přechází od teoretického základu k praktické realizaci aplikace. Budou zde popsány důležité soubory, třídy a metody, které zajišťují funkčnost celého řešení. Sekce „Experimenty“ se zabývá použitím výsledné aplikace. Nejdříve budou provedeny testy funkčnosti jednotlivých komponent a následně si ukážeme několik příkladů, jak lze systém použít pro sběr dat. V závěru práce se zaměříme na zhodnocení výsledného systému a na možnosti další práce s ním.

Kapitola 1

O crowdsourcingu

Crowdsourcing je proces vykonání práce nebo získání financí, většinou online, od davu lidí. Alespoň tak je definován ve článku s názvem „What is Crowdsourcing?“ na internetovém portálu Daily Crowdsource[1], který se touto problematikou zabývá. „Jde zkrátka o společné úsilí, spojení sil i nápadů jednotlivců, které umožňuje dosáhnout kýženého cíle mnohem efektivněji. Říká se, že víc hlav víc ví, a crowdsourcing na tomto přesně stojí. I proto je často využíván ve vědeckých či technologických komunitách.“[1] Celosvětově známým projektem postaveným na příspěvcích uživatelů je encyklopedie wikipedia.org, jejíž obsah je tvořen dobrovolníky ze všech koutů země.[1][2]

Crowdsourcing může mít několik podob. První z nich je Crowd design. Pokud například firma potřebuje logo, místo toho, aby najala jednoho návrháře, zadá tento úkolu na internet, kde uvede svoje požadavky, kolik je ochotna za práci zaplatit a do kdy je potřeba úkol splnit. Do speciální aplikace pak může kdokoli nahrát svůj návrh. Zadavatel má možnost vybrat si výtvor, který se mu nejvíce líbí, přičemž pokud použije některý ze známých crowdsourcingových serverů, může získat klidně stovky obrázků. Díky tomu je veliká šance, že vybrané logo bude skutečně kvalitní a také za jeho vytvoření může firma zaplatit méně než při použití jiných způsobů zadávání práce.[2]

Typ crowdsourcingu, kdy zadavatel získává pro uskutečnění projektu od veřejnosti finanční prostředky se nazývá crowdfunding. Uplatnění lze najít u realizování nápadů začínajících podnikatelů, ale také pokud náklady na výrobu nového produktu není firma schopna pokrýt. Příkladem může být vydávání nových alb interprety, kteří mohou ještě před započetím nahrávání zjistit, zda o jejich tvorbu bude skutečně zájem. Na vybrání prostředků je většinou omezená doba, za kterou se musí získat 100 % požadované částky. Jestliže se na tuto sumu nepodaří dosáhnout, záleží na zvážení interpreta, zda chce do alba i přesto investovat. V opačném případě musí umělec nabízený produkt vytvořit. Uživatelům je také přislíbena odměna za poskytnutí příspěvku. Ta

[1] http://dailycrowdsource.com/

je většinou odvozena od výše částky, kterou zájemce poskytne. Například za 200 Kč má jistotu, že získá nově vydané CD, za větší obnos může obdržet něco navíc, například reklamní a dárkové předměty. Pro získávání prostředků slouží specializované servery, v České republice jsou to například hithit.com nebo startovac.cz. Mezi nejznámější aplikace pro výběr financí ve světě patří kickstarter.com.[1][2]

Třetím typem použití crowdsourcingu jsou takzvané Microtasks, neboli mikroúkoly. Podstatou je, že se velká úloha, která vyžaduje mnoho práce, rozdělí na malé podúkoly, které jsou opět delegované na velké množství řešitelů. Pokud například budeme mít tisíce fotografií a budeme chtít zjistit, co se na daných snímcích nachází, necháme uživatele internetu, aby popsali tolik obrázků, kolik sami chtějí. Za přidání jednoho popisu pak může být přispěvatel odměněn malou finanční částkou v řádech korun nebo desetihaléřů. Systém odměn však oproti crowdfundingu nemusí existovat, popřípadě ocenění za provedenou práci může mít jinou než finanční podobu. Pokud například okomentované fotografie budou použity v nějaké webové aplikaci, jejíž používání bude zpoplatněno, přispěvatel ji může používat zdarma. Samozřejmě se na projektech mohou podílet i dobrovolníci, ačkoli za svoji pomoc žádnou odměnu nedostanou. Příkladem jsou open-source softwarové produkty, k jejichž vývoji se může kdokoli připojit a do výsledné aplikace přidat novou funkcionalitu nebo opravit existující chyby. Výhodou mikroúkolů je především to, že práce je hotová rychleji a získaná data nejsou pouze subjektivním ohodnocením jednoho člověka, ale poskytnuté informace jsou mnohem objektivnější.[2]

Pokud si uživatel není jistý s nápadem pro podnikání, svým produktem, obchodní strategií nebo reklamní kampaní, může využít opět crowdsourcing. Tento způsob použití se označuje jako Open Innovation a umožňuje všem zainteresovaným osobám v podnikání, například investorům, designerům, inovátorům nebo obchodníkům spolupracovat na vytvoření prosperující společnosti. Tento způsob spojuje různé lidi z odlišných oborů a částí světa a umožňuje jim pracovat na společných projektech. Díky tomu vznikají skupiny lidí se širokým spektrem znalostí a dovedností, které mezi sebou dokáží sdílet a nabízet produkty a služby, které by samostatně nikdo z nich nemohl vytvářet.[2]

Mezi největší výhody patří bezpochyby možnost získání lepších výsledků než v případě, kdy pouze několik jedinců nabídne své služby. Zadavatel může vybrat nejlepší řešení z velkého množství příspěvků uživatelů. Potřebná data jsou také získána velice rychle především díky způsobu komunikace přes internet. Nové logo můžeme mít navrženo během týdne a některé úlohy vyhodnoceny za pár minut. V mnoha případech může být tento způsob využívání lidských zdrojů levnější, než zadání zakázky jedné firmě.[2][3]

Ačkoli by se mohlo zdát, že crowdsourcing přináší lidem samé výhody, existují zde i jistá omezení a negativní aspekty. Při definování úkolu je nezbytné přesně vyjádřit naše požadavky. Pokud by došlo k tomu, že přispěvatel zcela neporozumí zadání, můžeme získat ne příliš kvalitní výsledky. V případě Crowd designu se zadavateli můžou sejít tisíce návrhů a vybrat z nich

ten nejlepší bude samo o sobě velice časově náročné, z tohoto důvodu se mohou dodatečně zvýšit náklady na dotažení projektu do konce. Některé úkoly vyžadují od přispěvatelů dobré znalosti, dovednosti nebo zkušenosti v daném oboru, což není vždy zaručeno. Tím pádem i získané údaje nebudou dosahovat takových kvalit, jaké bychom potřebovali. Další nevýhoda crowdsourcingu je ta, že mezi zadavatelem a přispěvatelem neexistuje žádný pevný vztah. Pokud by jedna z těchto stran chtěla dodatečně změnit vykonanou práci, ve většině případů to bude jen těžko realizovatelné.[2][3]

Ačkoli crowdsourcing zahrnuje několik odvětví, v této práci se zaměříme pouze na mikroúkoly, kdy je práce rozdělena na malé části a ty mohou být provedeny různými uživateli. Tento způsob vykonávání úloh je použitelný ke komerčním účelům, ale také jako způsob získávání dat při zkoumání na akademické půdě. Firmy mohou pomocí mikroprací nechat analyzovat data o sobě nebo o svých produktech a službách. Přispěvatelé například mohou navrhnout co nejpoutavější text prezentující společnost nebo nabízené zboží. Mohou přispět svými znalostmi k rozdělení nabízených produktů do kategorií, vyhledávat o nich dodatečné informace na webu nebo naopak označit některé údaje za zbytečné, protože nejsou pro uživatele důležité při rozhodování o nákupu. Samozřejmě je možné využít crowdsourcing pro získání informací z vnitřních dokumentů firmy, například překlady textů a přepis naskenovaných dokumentů.

Ve vědecké sféře má crowdsourcing také mnoho uplatnění. Kromě získávání nových informací se například jedná o případy, kdy uživatelé řeší stejný úkol jako nějaký algoritmus a na základě výsledků se autor může rozhodnout, zda program pracuje správně, popřípadě lze získaná data použít pro vylepšení učících se algoritmů. Velkou oblastí, kde se uplatňuje získávání dat od uživatelů je data mining nebo text mining. Mikroúlohy se pak hodí například pro extrakci informací z textu, nalezení klíčových slov nebo označení pojmenovaných entit. Samozřejmě přispěvatelé mohou rozhodnout o významu homonym v daném kontextu, nebo u podstatných jmen zadají tvary slova ve všech pádech. Crowdsourcing zde má velikou výhodu, jelikož potřebné informace získáme poměrně rychle a díky tomu, že více lidí splní ten samý úkol, budeme mít také přesnější a objektivnější údaje.

Kapitola 2

Analýza existujících řešení

V této kapitole se podíváme na několik existujících webových řešení pro crowdsourcing. Ačkoli podobných aplikací existuje jistě více, zaměříme se pouze na ty, které by mohly nejlépe splňovat předpoklady pro použití v data miningu. Každý nástroj bude nejdříve popsán a následně budou zhodnoceny jeho výhody a nevýhody. Na konci kapitoly se na základě získaných poznatků pokusíme rozhodnout, zda by nějaké řešení bylo vhodné pro naše požadované účely, popřípadě zda je lepší navrhnout a implementovat nový systém.

2.1 CrowdFlower

2.1.1 Popis

CrowdFlower.com je webová aplikace, která nabízí svým zákazníkům získání užitečných informací od dalších uživatelů. Dokonce o sobě tvrdí, že je nezbytnou platformou pro obohacování dat. Autoři aplikace se mohou pochlubit, že jejich produkt využívají významné společnosti jako například Google, Facebook, Twitter, Cisco, Microsoft, GitHub, Ebay a mnoho dalších. Zákazníci mohou nad svými daty spustit různé úlohy, díky kterým získají užitečné informace pro další zpracování nebo pro vylepšení výsledků strojového učení. Mezi poskytované nástroje patří například sentimentální analýza (uživatel na určité škále hodnotí, zda na něj text působí pozitivně, negativně, neutrálně a podobně), search relevance (uživatel vybere, jaké výsledky vyhledávání nejlépe odpovídají zadaným kritériím), content moderation (podstatou je okomentování obsahu textů, obrázků, internetových stránek a tak dále), data collection (uživatelé vyhledávají a sbírají různé informace, například kontaktní údaje zadaných firem), data categorization (přispěvatelé zařazují data do různých kategorií), transcription (jedná se například o popis obrázku nebo přepis audio záznamu).[4]

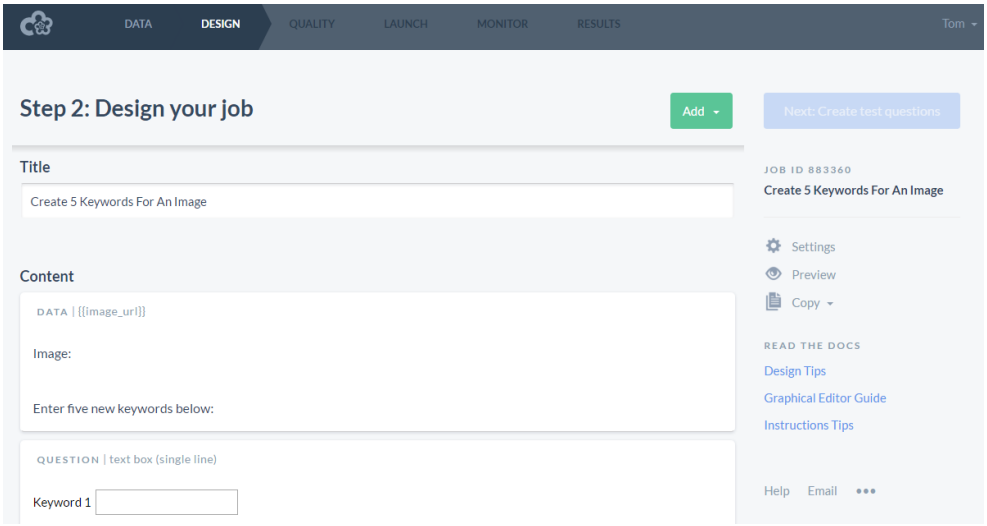
Zákazník má možnost vybrat si jeden ze dvou typů členství. První typ se nazývá Data for Everyone, který je určen pro studenty, akademické pra-

covníky a vědce a získané informace nesmí být použity ke komerčním účelům. Nahraná data mohou být zveřejněna, což je pro firemní účely nevhodné. Takto vytvořené členství je zdarma, platí se však uživatelům, kteří data ohodnotí, plus malá provize. Druhý typ registrace se nazývá CrowdFlower Pro a je určen pro firmy s velkým množstvím dat. Toto členství je placené, konkrétní cena na internetových stránkách však uvedena není. Zákazníci získají možnost nahrávat neomezené množství údajů k ohodnocení, jejich data zůstanou soukromá a mohou využít zákaznickou podporu včetně on-line tréninků.[4]

Po registraci a přihlášení může zákazník začít vytvářet první úkoly, které je možné pro větší přehlednost dělit do různých projektů. Jestliže chce uživatel vytvořit nový úkol, musí nejdříve vybrat, o jaký typ se jedná. Každý typ je navíc rozdělen na několik podtypů, které se většinou liší v drobnostech. Například u sentimentální analýzy lze zvolit, zda se bude odpověď vybírat na škále very positive, slightly positive, neutral, slightly negative, very negative, not relevant, nebo je mu nejdříve zobrazena na výběr možnost, zda je text relevantní k danému tématu a pokud ano, pak jsou teprve zobrazeny možnosti, popřípadě může text hodnotit na číselné stupnici od 1 do 10. U každého podtypu je přímo popsána struktura vstupního souboru, ze kterého se budou data načítat. Ještě předtím, než si uživatel konkrétní typ vybere, může se podívat na demo ukázku, jak bude daný úkol vypadat a jak se bude hodnotit.[4]

Poté, co uživatel zvolí typ úlohy, je dále směrován průvodcem pro její nastavení a spuštění. Nejdříve zákazník nahrává data, která mohou být v několika formátech, například CSV, TSV, XLS, XLSX a ODS. V dalším kroku je možnost nastavení úkolu, konkrétně se jedná mimo jiné o název, popisky jednotlivých možností a návod, jak správně úlohu vyplňovat. V následující fázi je možné specifikovat kontrolní otázky, které slouží k ověření, zda má uživatel potřebné znalosti k vyplnění úlohy. Cílem těchto otázek je zajistit, aby získané informace byly co možná nejkvalitnější. Dále zákazník zadá informace o cenách za poskytnutí potřebných údajů k jednotlivým datovým položkám. Tímto krokem je nastavení kompletní a úkol může být spuštěn. Během toho, kdy vyplňování běží, má uživatel informace o tom, kolik ohodnocení již bylo vyplněno a také se na poskytnuté informace může podívat. CrowdFlower pro správu úloh a dat poskytuje také přístup pomocí API. V uživatelské příručce je velice detailně a srozumitelně popsáno, jak se s API pracuje, jaké jsou internetové adresy metod, jaký vstup server očekává a jaký výstup je vrácen klientské aplikaci.[4]

Pro uživatele, kteří chtějí pomoci ohodnotit nějaká data je připraveno přehledné uživatelské rozhraní, kde si mohou zobrazit seznam úkolů, které je možné řešit. Úlohy je možné řadit podle identifikátoru, názvu, velikosti odměny za vyplnění a počtu datových položek. Jeden ze sloupců se nazývá requirements, obsahuje obrázky s čísly jedna, dva, tři a udává, jací přispěvatelé se mohou práce zúčastnit. U všech těchto úrovní musí mít uživatel za sebou vyplnění několika set testových otázek. U „levelu“ tři je nutné, aby uživatel v těchto testových otázkách dosáhl přesnosti alespoň 85 %, u úrovně jedna je



Obrázek 2.1: Ukázka nastavení úkolu v aplikaci CrowdFlower

to pak 70 %. Vyplácení odměn je prováděno metodou PayPal, to znamená, že přispěvatel musí mít vytvořený účet, který zadá do nastavení svého účtu na crowdsourcingovém portálu.[4]

2.1.2 Zhodnocení

CrowdFlower je velice povedený nástroj s hezkým, příjemným a přehledným uživatelským rozhraním. Nabízí možnost vytvoření a správu mnoha různých typů úloh s texty, obrázky a také pár úkolů s audio soubory. Systém pracuje na základě provizí, které jsou poměrně malé (v řádu setin dolaru za ohodnocení jedné datové položky), což je příjemné pro zákazníky, kteří potřebují svá data obohatit. Aby možnost získání provize nebyla zneužívána automatizovaným vyplňováním, je sekce přispěvatelů chráněná pomocí CAPTCHA kódu a také je potřeba vyplňovat kontrolní otázky, aby uživatel prokázal svou způsobilost ke zvládnutí daného úkolu. Na druhou stranu mohou být tyto úkoly trochu obtěžující.

Jako velká nevýhoda se jeví formát nahrávaných dat. Veškerý import probíhá v „tabulkových“ formátech jako CSV, TSV, XLS, XLSX, kde je každý řádek brán jako jedná datová položka. Není tedy možné například nahrát soubor a ten brát jako jeden záznam pro ohodnocení. Další nevýhoda, která z tohoto způsobu uploadu dat plyne, je ta, že obrázky a audio soubory nelze nahrávat přímo, ale v nahraném souboru jsou zadány internetové adresy položek. To znamená, že uživatel musí svá data nejdříve publikovat na veřejně přístupné místo na internetu a až pak je možné s nimi pracovat v aplikaci.

Ačkoli aplikace nabízí velké množství typů úloh, může se stát, že zákazník potřebuje zcela jiný způsob ohodnocení dat. Například když je potřeba

v daném textu ohodnotit několik částí zvlášť – může se jednat o nalezení významu slov v daném kontextu. V takovém případě neexistuje způsob, jak takovou úlohu vytvořit, nicméně je možnost kontaktovat společnost CrowdFlower. Otázka ovšem je, zda bude přání uživatele vyhověno či nikoli.

Poslední nevýhoda, která stojí za zmínku, je poměrně velký rozdíl vzhledu aplikace mezi sekcí pro zákazníky a přispěvatele. Část aplikace pro uživatele, kteří chtějí přispět k obohacení dat, se liší nejenom od klientské části aplikace, ale také od informačních stránek. To může být pro přispěvatele velice matoucí. Navíc nebylo možné začít s ohodnocením nějakých dat. Předpokládám, že je to kvůli tomu, že nemám vyplněný dostatečný počet otázek s určitou přesností, nicméně je to jenom domněnka.

2.2 PyBossa

2.2.1 Popis

Projekt PyBossa se na svých internetových stránkách pybossa.com představuje jako framework pro analýzu a obohacení dat, která nemohou být zpracována automatizovaně. Tentokrát se nejedná o aplikaci, kterou může uživatel okamžitě začít používat přes internet, ale jedná se o nástroj, který je třeba nainstalovat na vlastní virtuální stroj. PyBossa je zdarma a open-source projekt, je naprogramován v jazyce Python a jako databázi používá PostgreSQL. Podle dokumentace je třeba jej spustit v operačním systému Ubuntu verze 14.04 LTS. Po instalaci jej musí uživatel nakonfigurovat a následně může začít vytvářet a spravovat své projekty a úlohy.[5]

Autoři ve webové prezentaci uvádějí několik klíčových vlastností tohoto projektu. Jelikož se framework instaluje na vlastní server, záleží čistě na uživateli, zda použije privátní či veřejný strpj a od toho se také odvíjí, kdo bude mít k datům přístup. Aplikaci lze přizpůsobit různým požadavkům a je možné ji rozšířit pomocí pluginů. Aby nebylo nutné začínat zcela od nuly, jsou k dispozici šablony několika typů úloh. Jednou z dalších vlastností je integrace s mnoha webovými aplikacemi například Amazon, Flickr, Dropbox, Google, SoundCloud, Twitter a další. Sociální servery lze také využít pro přihlašování do aplikace.[5]

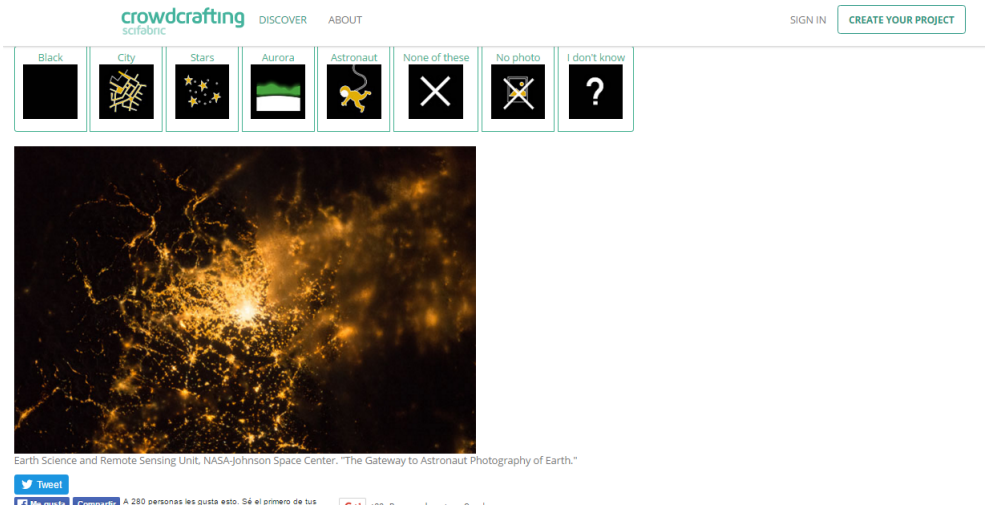
Mezi základní případy použití je zařazen například „PDF data-mining“, kdy uživatelé mohou získávat různé informace z PDF souborů. Zdrojová data jsou v tomto případě načítána z Dropbox nebo z vlastního webového serveru. Dalším příkladem použití je analýza video souborů, které jsou umístěny v aplikaci Vimeo nebo Dropbox (další by měly být postupně přidávány). U videa pak mohou přispěvatelé přidávat informace například o obsahu videa, kde se děj odehrává, jaké zde vystupují postavy nebo jaké emoce snímek v člověku vyvolává. Velice specifickým příkladem je analýza fotografií ze zařízení Rapsberry Pi. V tomto případě je možné použít konzolový nástroj pro vytváření fotografií v určitých intervalech, jejich nahrání na Flickr a následné ohodnocení

pomocí aplikace vytvořené v PyBossa. Další příklad se týká také fotografií, konkrétně jde o poskytnutí meta informací o snímku. Obrázky vytvořené staršími aparáty neobsahují například informace týkající se lokace, kde byl snímek pořízen. Opět je tedy možné uploadovat fotografie na Flickr nebo Dropbox a nechat uživatele, aby zadali místo, odkud snímek pochází. Framework podporuje také práci s audio záznamy, u kterých chceme například vědět, co je v nahrávce slyšet, popřípadě potřebujeme přepsat hlasy do textové podoby. Opět se záznamy nacházejí v Dropboxu nebo ve službě SoundCloud.[5]

Dokumentace na internetových stránkách je velice rozsáhlá a přehledná. Obsahuje návody, jak framework nainstalovat na vlastní server a jak jej nakonfigurovat. Aby mohla aplikace úspěšně běžet, je třeba nastavit několik údajů například připojení do databáze, API klíče pro přihlašování přes sociální média, nastavení kontaktních údajů na provozovatele, nastavení cache a mnoho dalších. Po instalaci a nastavení aplikace může správce přidávat a odebírat další administrátory, spravovat kategorie úloh, zobrazovat logy a statistické údaje o používání. V příručce pro administrátory se uživatel také může dočíst, jak přeložit aplikaci do požadovaného jazyka a jakým způsobem se vytvářejí nové pluginy. V další části dokumentace je pak uvedeno, jak ovládat aplikaci, konkrétně přidávání projektů a úloh, načítání dat ze serverů třetích stran a orientace v získaných výsledcích. Jelikož PyBossa podporuje přístup pomocí API, je zde také rozsáhlý návod, jak k aplikaci přes toto rozhraní přistupovat.[5]

Na internetových stránkách PyBossa je jako produkt uveden také projekt CrowdCrafting, což je již webová aplikace sloužící k obohacení dat a využívající samotný framework. Nástroj je prezentován jako pomocník pro práci s daty, které slouží k vědeckým a edukačním účelům. Díky tomu je použití zcela zdarma a také přispěvatelé nejsou za svoji práci finančně odměněni. Kvůli zaměření projektu je při vytváření nové úlohy vyžadováno několik povinných informací, například k čemu budou data sloužit, pro koho jsou určeny a podobně. Dále následuje nastavení projektu, to znamená, s jakým typem dat budeme pracovat, co s nimi budeme dělat, jaké jsou konkrétní otázky na data a podobně. Přispět k ohodnocení dat může kdokoli, není třeba žádné přihlášení. V době psaní této práce se mezi úkoly nacházelo například hodnocení obrázků pořízených z Mezinárodní vesmírné stanice (ISS). Cílem je rozhodnout, zda je na fotografiích zachyceno město, hvězdy, polární záře nebo například astronaut. Tento úkol čítá přes 190 000 obrázků, z nichž 165 000 bylo ohodnoceno necelými 19 000 přispěvateli. Další úlohou bylo opět ohodnocení obrázků z ISS, tentokrát se měli uživatelé pokusit rozhodnout, jaké město bylo z vesmírné stanice vyfoceno. Velice zajímavý úkol byl přepis ručně psaných textů z obrázků. Konkrétně se jednalo o zápisky v deníku Winstona Churchilla z období druhé světové války.[5][6]

Kromě aplikace CrowdCrafting, která byla vytvořena stejnými tvůrci jako samotný framework, jsou ve webové prezentaci uvedeny další projekty, které nástroj PyBossa využívají. Prvním uvedeným projektem je micropasts.org,



Obrázek 2.2: Ukázka ohodnocení úkolu v aplikaci CrowdCrafting

který provozuje British Museum and University College London. Pomocí aplikace jsou získávána data od vědců, studentů a nadšenců, kteří se zabývají historií. Mezi projekty patří například vytváření trojrozměrných modelů historických artefaktů, rozšiřování archivů se starými fotografiemi nebo přepis dávných rukopisů. Dalšími organizacemi, které využily projekt PyBossa jsou The British Library a The National Library of Israel. Obě tyto knihovny používají koncept crowdsourcingu k získávaní dat o knihách ať už současných či historických. Jedná se například o přepis informací ze zastaralých, papírových katalogových lístků do elektronické podoby nebo o získávání metadat o různých dokumentech. Noviny The Guardian vytvořily na základě frameworku projekt, který slouží k přepisu PDF dokumentů do strojově čitelné podoby. Organizace se tak snaží veřejně šířit informace o investicích, darech a dalším hospodaření politiků.[5]

2.2.2 Zhodnocení

Projekt PyBossa působí jako velice užitečný nástroj. Jeho uživatelské rozhraní je srozumitelné a vzhledově velmi hezké. Jeho velikou výhodou je otevřený zdrojový kód, který si kdokoli může stáhnout a upravit podle své libosti. Díky tomu mohou být realizovány i velice specifické úlohy. Upravená verze se pak nahraje na server a následně ji mohou začít používat lidé, kteří mají k serveru přístup. To znamená, že data mohou například zůstat jenom ve firemní síti, nebo mohou být dostupná přes internet odkudkoli.

Nevýhoda tohoto řešení je, že uživatel si musí zprovoznit vlastní server, ať už přístupný z internetu, nebo umístěný ve firemní síti. Výpočetní uzel je také potřeba nakonfigurovat, nainstalovat Python, databázi a případně další

potřebné nástroje. Tento úkol může být pro nezkušeného uživatele překážkou. Běžnějším způsobem zprovoznění bude nejspíš pronájem prostoru na severu poskytovatele webhostingu, za tyto služby ovšem zákazník musí většinou zaplatit. Pokud navíc potřebujeme ohodnotit data od velkého množství přispěvatelů, bude problém tolik lidí sehnat, jelikož půjde o neznámý server s neznámou doménou. Řešení se tedy hodí skutečně spíše do vědecké a vzdělávací sféry, kdy je možné šířit odkazy na naši vytvořenou aplikaci prostřednictvím akademické obce.

Další nevýhodou je práce s mediálními soubory, které není možné nahrávat přímo do aplikace, ale data se načítají z jiných internetových portálů, jako například GitHub, Flickr, Vimeo a podobně. Tento způsob je nevhodný především z toho důvodu, že uživatel je nucen vytvářet si účty na těchto službách. Po nahrání dat na jakýkoli z těchto portálů nad nimi zcela ztrácí kontrolu a nemůže ovlivnit, kde a jak dlouho budou data uložena, popřípadě, zda se nebudou dále šířit internetem. Co se týče aplikace CrowdCrafting, nebylo dokonce ani možné nahrávat textová data, šlo pouze pracovat s příspěvky na sociální síti Twitter.

2.3 CrowdSource

2.3.1 Popis

Aplikace CrowdSource ve své internetové prezentaci uvádí, že poskytuje firmám „pracovní sílu jako službu“ (workforce as a service). Toto řešení je cíleno především na společnosti, které mohou využít kvalifikovaných dobrovolníků k rychlejšímu a efektivnějšímu plnění úkolů. Mezi typické zájemce o využití produktu mohou patřit internetové obchody, které pomocí konkrétních úkolů zpřehlední svoji nabídku, rozšíří informace o produktech, pročistí nedůležité údaje a poskytnou kvalitnější služby nakupujícím. Další využití lze jednoznačně najít u firem, které poskytují různé on-line služby. Komunita přispěvatelů pak dokáže vytvořit poutavé články, příspěvky v blogu a analyzovat klíčová slova a pojmy v dokumentech, což je zásadní prvek v procesu optimalizace webové prezentace pro vyhledávací stroje. Samozřejmě službu CrowdSource může použít kdokoli, kdo potřebuje zpracovat určitý úkol a nechce kvůli tomu zaměstnat nové pracovníky, popřípadě chce získat mnohem objektivnější data než při vypracování jediným člověkem.[7]

Způsob fungování se od předchozích dvou řešení mírně liší z toho důvodu, že se jedná o produkt určený především společnostem ke komerčním účelům. Produkt CrowdSource není pouze aplikací, ale zahrnuje v sobě další služby, například kontrolu výsledků důvěryhodnými osobami. Při zadání nového úkolu jsou všichni přispěvatelé instruováni o způsobu zodpovídání otázek, popřípadě jsou zasvěceni do problematiky, které se úloha týká. Tím je zajištěno, že získané informace budou spolehlivé a skutečně přínosné. Následně si mohou pracovníci vybrat úkol, jež odpovídá jejich kvalifikaci. Při zodpovídání

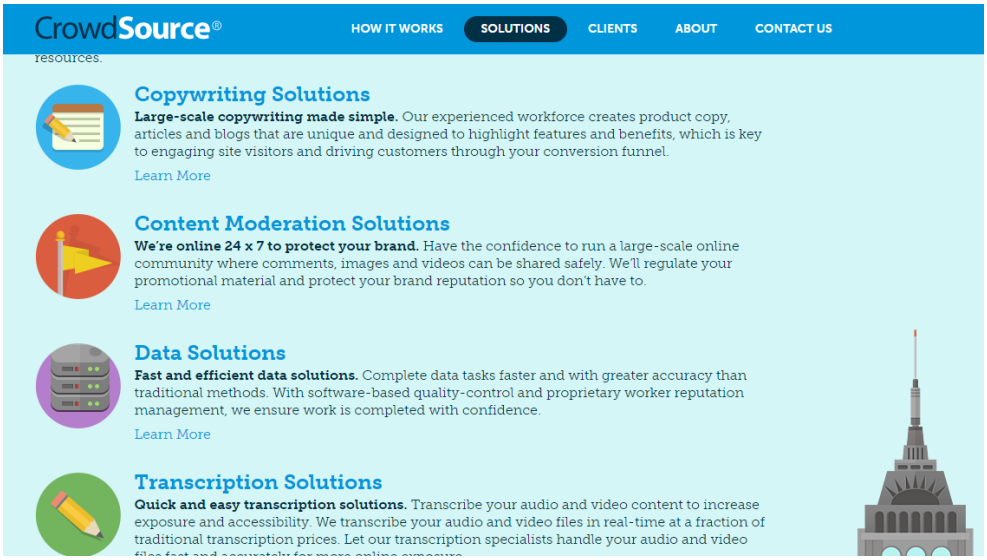
položených otázek jsou jejich odpovědi a celý průběh kontrolovány týmem odborníků, kteří dohlížejí na kvalitu výsledku. Přispěvatelé jsou hodnoceni na základě toho, jak dobré jsou jejich poskytnuté údaje. Hodnocení pracovníků není pouze pozitivní, ale i negativní. Čím lepší data uživatel poskytne, tím se dostává na vyšší úrovně a může se podílet na širším spektru úloh. Naopak méně úspěšní přispěvatelé mohou získávat záporná hodnocení a jsou nuceni znovu absolvovat školení.[7]

Seznam služeb, které produkt nabízí, je opět velice široký, tentokrát je tvořen především úkoly, které vytváří přidanou hodnotu pro podnikání a byznys. Úlohy jsou rozděleny do čtyř kategorií a první z nich je Copywriting. V tomto balíčku služeb se nachází například psaní článků, tvorba efektivních prodejních materiálů, popis produktů a služeb nebo tvorba článků do blogů. Jako důvod, proč využít právě těchto služeb, je uvedeno široké spektrum přispěvatelů s různými dovednostmi. Tím pádem vytvořený obsah není dílem jediného autora, ale výsledek je vybrán z několika návrhů. Vybraný text pak může být ještě zkontrolován a modifikován dalšími dobrovolníky. Výsledek je dílem několika lidí a tím pádem by měl být objektivnější a čtivější. Zadavatel má samozřejmě možnost sledovat stav vývoje úkolu.[7]

Druhá skupina úkolů je pojmenována Content Moderation, která v sobě zahrnuje úkoly týkající se komentování obsahu. Jedná se o přidávání informací k obrázkům, videům, komentářům uživatelů nebo o sentimentální analýzu. Poslední zmíněný úkol se hodí k získávání a ohodnocení názorů veřejnosti na společnost prostřednictvím sociálních sítí. Třetí kategorii úloh autoři pojmenovali Data. Jedná se konkrétně o úkoly Search Relevance, které slouží k zlepšení strategie vyhledávání obsahu na webu, Data Categorization, jenž umožňuje rozdělit záznamy do kategorií podle různých kritérií, Data Tagging, sloužící k lepší orientaci v dokumentech, Data Deduplication, v němž mají uživatelé rozhodnout o existenci duplikací mezi záznamy, Data Enrichment, který slouží k získání nových údajů o datových položkách, a Data Cleaning, kdy přispěvatelé mají označit skutečně zajímavé a relevantní informace, aby zákazníci společnosti nebyli zatěžováni těmi nepodstatnými. Veškeré úkoly v této skupině slouží k získání lepších a hodnotnějších podnikových dat, a firma tak může svým klientům a společníkům poskytovat kvalitnější informace.[7]

Poslední kategorií úkolů je transkripce, která se může týkat audio nebo video nahrávek. Díky těmto úlohám je možné získat nejenom kompletní přepis záznamu, ale také například klíčová slova, což jsou údaje potřebné k tomu, aby vyhledávače dané soubory lépe indexovaly. CrowdSource nabízí zpracování multimediálních dokumentů z různých odvětví od firemních prezentací přes meetingy a konference až po akademické záznamy. Podle informací je přepis velice kvalitní, konkrétně je přesnost lepší než 98 %, ať už se jedná o doslovný přepis nebo o přeformulování slovního projevu, popřípadě výtah důležitých částí.[7]

Mezi další výhody použití CrowdSource patří možnost přistupovat k aplikaci přes rozhraní API, díky kterému má uživatel vždy aktuální údaje. Dále



Obrázek 2.3: Popis použití produktu CrowdSource ve webové prezentaci

produkt nabízí možnosti pokročilého monitorování úloh a pravidelné poskytování informací o průběhu vyplňování. Kromě nasazení již existujících případů použití nabízí tým stojící za tímto projektem implementaci nových typů úkolů šitých na míru potřebám zákazníka. Při zadávání úloh, které od přispěvatelů vyžadují vytvoření vlastních textů, jsou nahrané příspěvky automatizovaně kontrolovány proti plagiátorství a také je ověřována jejich kvalita ještě před tím, než jsou doručeny zadavateli.[7]

2.3.2 Zhodnocení

Produkt CrowdSource působí dojmem, který je pro firmy skutečně přitažlivý, protože dokáže pomoci se ziskem důležitých informací, které budou kvalitní. Zaměření na byznys je znát nejenom z webové prezentace, ale také z nabízených úkolů. Ty jsou uzpůsobeny především firemním datům, ale určitě by se daly použít i na některé úkoly ve vědecké nebo vzdělávací oblasti. Informace o cenových podmínkách ovšem nejsou ve webové prezentaci uvedeny, otázkou tedy zůstává, zda by se použití mimo komerční sféru vyplatilo.

Na rozdíl od předchozích dvou řešení je největší nedostatek to, že není možné používat aplikaci zdarma, dokonce ani v demo verzi po určitou dobu. Kvůli tomu nebylo možné vyzkoušet práci s projekty, úkoly ani daty. Na stránkách navíc neexistuje žádná dokumentace, ze které by se dalo vypozorovat, jak se aplikace používá. Ačkoli produkt podporuje přístup přes aplikační rozhraní, není k dispozici žádná dokumentace týkající se této problematiky. Absence kvalitních materiálů o použití aplikace může potencionální zákazníky odradit.

Další nevýhoda je zřejmá v aplikační části pro přispěvatele. Jestliže se

chce uživatel zaregistrovat, využije odkaz, který není příliš dobře viditelný. Po přechodu na novou stránku je informován, že projekt CrowdSource nyní spadá pod OneSpace a následně je přesměrován na prezentaci této služby, aniž by přesně věděl, o co se jedná. OneSpace.com je aplikace, která shromažďuje nezávisle pracovníky (freelancers), jejichž služby mohou společnosti využít. Zdá se tedy, že produkt CrowdSource byl přejmenován nebo odkoupen jinou společností, ačkoli ve většině internetové prezentace je představován pod svým původním názvem. To může být velice matoucí nejen pro uživatele, kteří se chtějí připojit ke komunitě hodnotitelů, ale také pro osoby, které pouze chtějí získat informace o nabízených produktech a službách.[8]

Na stránkách OneSpace.com se ani nedočteme, jaké přesné služby aplikace nabízí, dokonce neexistuje žádný odkaz, který by vedl zpět na prezentaci produktu CrowdSource. Podle obrázků to vypadá, že se skutečně jedná o řešení pro crowdsourcing. Ačkoli jsou na stránkách uvedeny různé číselné údaje, které mají dokazovat prospěšnost využití služby, a jsou zde vyjmenováni významní klienti jako Facebook, Ebay, Microsoft, Hallmark a další, informace o aplikaci mohou být skutečně spíše matoucí.[8]

2.4 Shrnutí

Kromě těchto tří nástrojů existují samozřejmě další, které jsou více či méně povedené. Co se týče hotových aplikací, pracují na podobných principech jako CrowdFlower a CrowdSource, nebo byly velice úzce specializované. Rozsáhlý seznam crowdsourcingových webových aplikací lze najít například ve článku „35 Places to Find Micro Jobs“[2]. Tento dokument obsahuje seznam nástrojů, které se zabývají mikroúkoly, a zkoumá jejich potenciál pro získání peněz za obohacování dat. Další výpis crowdsourcingových aplikací obsahuje článek s názvem „8 Crowdsourcing apps we love“[3], který se ovšem zabývá specializovanými nástroji, jako například Waze od společnosti Google ke sledování dopravních informací nebo OpenStreetMap, což je projekt spravující open-source mapové podklady. Podobné seznamy jsou i na mnoha jiných stránkách, navíc často jsou v článcích zahrnuty také crowdfundingové nástroje. Z tohoto důvodu CrowdFlower a CrowdSource vypadaly nejslibněji, ovšem stále se jedná o kompletní řešení především pro komerční účely. Z tohoto důvodu je zřejmé, že nenabízejí otevřený zdrojový kód a nesplňují tak jeden z našich klíčových požadavků.[9][10]

Co se týče open-source nástrojů, jsou možnosti opravdu velice omezené. Kromě výše zmíněného frameworku PyBossa lze najít projekt Hive, který je postavený na technologiích Go a Elasticsearch. Avšak informace nejsou příliš

[2] http://workathomemoms.about.com/od/Micro-Jobs-Crowdsourcing/ss/Micro-Jobs.htm

[3] https://opensignal.com/blog/2015/07/09/8-crowdsourcing-apps-besides-opensignals-love/

rozsáhlé, existuje stránka na portálu GitHub, která obsahuje API dokumentaci, ale nepopisuje možnosti frameworku. Proto není ani zcela jasné, zda obsahuje například grafické rozhraní či nikoli. Nejlepším existujícím nástrojem se tedy jeví PyBossa, ovšem ani ten není zcela ideální. Například nahrávání fotografií, audio a video záznamů neprobíhá z lokálního počítače, ale z jiných internetových služeb jako je YouTube, Flickr, Twitter, Dropbox a podobně. Také použití API je poměrně omezené, například není možné najít funkce pro export dat do souboru nebo pro přidávání příspěvků, což zcela znemožňuje připojit se k aplikaci z jiného nástroje z pohledu pracovníka. Jelikož má PyBossa otevřený zdrojový kód, zcela jistě by bylo možné chybějící funkcionality naprogramovat, nicméně úprava cizího kódu může být velice nepříjemná, zdlouhavá a náročná. Z tohoto důvodu se jako nejlepší řešení jeví implementace vlastního nástroje, čímž se budeme zabývat v dalším textu.[11][5]

Kapitola 3

Analýza a návrh

3.1 Analýza

3.1.1 Popis aplikace

Aplikace bude sloužit ke správě mikroúloh a jejich plnění. V systému budou existovat tři uživatelské role – nepřihlášený uživatel, zadavatel úkolu a přispěvatel. Zadavatel úkolu se musí nejdříve registrovat, poté je potřeba, aby se do aplikace přihlásil. Úlohy jsou pro větší přehlednost členěny do projektů, které je možné vytvářet, editovat a mazat. Do existujícího projektu lze přidat nový úkol, nejdříve se vybere jeho typ a následně jsou doplněny informace jako název a popis. Jednotlivé druhy zadání se budou vztahovat pouze na konkrétní datové typy, to znamená texty, obrázky, videa a audio nahrávky. Úloha bude obsahovat data, která uživatel nahraje prostřednictvím formuláře s tím, že jeden nahraný soubor bude obsahovat jednu datovou položku k hodnocení. Výjimkou je import ve formátu NIF, kde jsou data a jejich informace reprezentovány jako RDF trojice a jeden soubor může obsahovat několik prvků k obohacení. Texty mohou obsahovat HTML tagy, které se interpretují, a zobrazí se tedy ve formátované podobě. Jednotlivé záznamy k ohodnocení je možné odstranit, stejně tak půjde smazat celý úkol. U textů se příspěvky mohou vztahovat také pouze k jeho částem, například slovům a větám. Tyto fragmenty bude zadavatel specifikovat v souborech ve formátu CSV.

Přispěvatel může, po registraci a přihlášení, poskytovat svá hodnocení k datovým položkám. Úkoly lze seřadit podle různých atributů, například podle jejich názvu, datového typu nebo počtu vyplnění. Způsob zadávání informací není blíže specifikován, bude záležet na programátorovi, který vytvořil daný typ úkolu a společně s ním také šablony pro vyplňování údajů. Zadavatel práce má možnost sledovat stav zpracování a také lze prohlížet jednotlivé příspěvky, které je možné exportovat ve formátu CSV nebo NIF, záleží opět na konfiguraci úkolu.

Všechny funkce budou dostupné přes API. Jelikož veškerá činnost je možná

až po přihlášení, ověření identity bude probíhat některým běžně používaným způsobem. Aplikace by měla být snadno rozšiřitelná o další funkce, druhy úloh a způsoby hodnocení.

3.1.2 Funkční požadavky

3.1.2.1 Registrace uživatele

Pro používání systému je nutné se přihlásit, pokud osoba nemá doposud vytvořen účet, může se zaregistrovat. Uživatel zadá emailovou adresu a heslo a vybere, zda se chce registrovat jako zadavatel nebo přispěvatel, popřípadě obojí. Po odeslání formuláře se zkontroluje, zda email již v systému neexistuje. Nebude se provádět kontrola platnosti zadané adresy odesláním kontrolní zprávy. Kvůli vyhnutí se překlepům při volbě hesla bude tato hodnota vyžadována dvakrát.

3.1.2.2 Přihlášení uživatele

Jestliže osoba má vytvořen v systému účet, je potřeba, aby svoji totožnost prokázala přihlášením. Ověření identity bude probíhat pomocí emailové adresy a hesla, které si uživatel zvolil při registraci. Jestliže autentizace proběhne úspěšně, dojde k přesměrování na stránku, kde bude možné pokračovat v práci podle toho, jakou roli uživatel v systému plní (zadavatel, přispěvatel).

3.1.2.3 Správa projektů

Každý úkol musí být zařazen do nějakého projektu, které je možné vytvářet, editovat a mazat. Seskupování slouží k přehlednější správě úloh a také informují přispěvatele, k čemu jsou nasbíraná data potřeba. U projektu je evidován název a popis, kde by mělo být uvedeno, jaká data a z jakého důvodu zadavatel potřebuje.

3.1.2.4 Správa úkolů

Úkoly je možné vytvářet, modifikovat a mazat. Každý úkol má přesně jeden typ, který udává, jakým způsobem se budou data hodnotit. Jeden typ se také vztahuje k jednomu typu dat. Příkladem může být sentimentální analýza u textů, popis obrázku, transkripce audio záznamu a podobně. Úlohy mají název a popis, který může obsahovat informace o tom, jaká data jsou přispěvateli předložena a k jakému účelu budou použita, popřípadě jak co nejlépe provést jejich hodnocení. Tyto údaje je možné později měnit, nepůjde však změnit typ úkolu. U mikroprací je možné specifikovat cíl, to znamená, kolik minimálně potřebujeme příspěvků k datům. Pokud například bude hodnota rovna číslu tři, znamená to, že ke každé položce chceme mít alespoň tři ohodnocení. Dalším atributem úkolu bude počet zobrazených datových položek při

jednom vyplnění. Přispěvateli bude vždy zobrazen pouze tento počet položek k ohodnocení.

Úkoly se budou nacházet vždy v nějakém stavu. Stav může být buď zastavený, probíhající nebo dokončený. Zastavený úkol znamená, že ještě nebyl dokončen a není možné k němu přidávat ohodnocení. Každý úkol bude po vytvoření v tomto stavu, aby uživatelé nemohli přidávat příspěvky k úlohám, které nebyly dostatečně nakonfigurovány a nebyla do nich vložena všechna potřebná data. Po kliknutí na tlačítko bude možné přejít do stavu probíhající, kdy je možné k úkolu přidávat ohodnocení. Bude také možné probíhající úlohu pozastavit. Do posledního stavu (dokončený) přechází úkol automaticky, jestliže se dosáhne požadovaného počtu ohodnocení u všech datových položek.

3.1.2.5 Správa dat

V úkolech se bude obohacovat různý počet datových položek. Záznamy se budou nahrávat ze souborů a budou podporovány texty, obrázky, videa a audio nahrávky. Import bude probíhat tak, že každý nahraný soubor bude představovat jednu položku. Nebude možné používat data, která budu umístěna jinde na internetu, například specifikováním adresy v CSV souboru. V textu mohou být obsaženy HTML značky a obsah tím pádem bude zobrazen ve formátované podobě. Jednotlivé datové záznamy bude možné odstranit.

Při manipulaci s texty není nutné pracovat s celým obsahem, ale pouze s jeho částmi. Tyto části budou specifikovány v souboru CSV, který se do aplikace uloží. Import dat bude možné provést také ve formátu NIF. Jeden soubor může obsahovat více datových položek a také mít přímo specifikovány fragmenty textu. V úkolech, kde bude povinností přispěvatele vybrat jednu nebo více možností z mnoha, může mít každá datová položka zadány jiné volby. Ty se budou opět nahrávat v konfiguračním souboru ve formátu CSV.

3.1.2.6 Přidávání příspěvků

Uživatel s rolí přispěvatel může pomoci s obohacením dat. Lze zobrazit seznam úkolů v aplikaci, vyhledávat v nich a řadit je podle různých kritérií jako název, typ úkolu, typ dat nebo počet ohodnocení. K úlohám může uživatel může přispívat. Způsob, jakým bude poskytování informací provedeno záleží na autorovi daného typu práce. Může se jednat o vyplňování formulářů, klikání na odkazy a tlačítka nebo o jiný druh interakce uživatele se systémem. Ohodnocení je možné přidávat pouze k úkolům, které se vyskytují ve stavu probíhající. Pokud uživatel chce přispět, je mu zobrazen pouze omezený počet datových položek, který specifikuje zadavatel. Po ohodnocení jedné sady však může ten samý přispěvatel pracovat na další sadě položek z úkolu. Jeden uživatel může každou datovou položku zpracovat maximálně jednou. Pokud tedy ohodnotí všechna data v úkolu, nebude pro něj úloha v seznamu nadále zobrazena.

3.1.2.7 Správa příspěvků

Zadavatel práce bude mít možnost zobrazit průběh plnění úkolu, kolik přispěvatelé poskytli ohodnocení a zda se podařilo dosáhnout požadovaného cíle. Nebude možnost zjistit, jaký uživatel poskytl jaké údaje, ale u každé datové položky bude seznam hodnot, které byly do systému zadány. Konkrétní vzhled správy příspěvku bude mít na starost autor typu úkolu. Ohodnocení je možné exportovat ve formátech CSV a v určitých případech také v NIF. O tom, zda lze provést stažení konkrétního souboru, rozhodne v nastavení opět autor typu úlohy.

3.1.2.8 Sdílení úkolů

Aplikace bude poskytovat funkcionalitu pro snadné sdílení úkolů mezi přispěvateli. Bude se jednat o zobrazení URL adresy pro vyplnění úlohy ke snadnému zkopírování a odeslání jinému uživateli. Aplikace samotná nebude zabezpečovat zasílání zpráv mezi uživateli, to znamená, že o přeposlání odkazu se musí postarat sám přispěvatel nebo autor úkolu prostřednictvím jiného komunikačního nástroje. Také nebude možné sdílet vlastnictví projektů a úkolů, to znamená, že každý úkol a projekt bude mít vždy pouze jediného vlastníka.

3.1.3 Nefunkční požadavky

3.1.3.1 RESTful

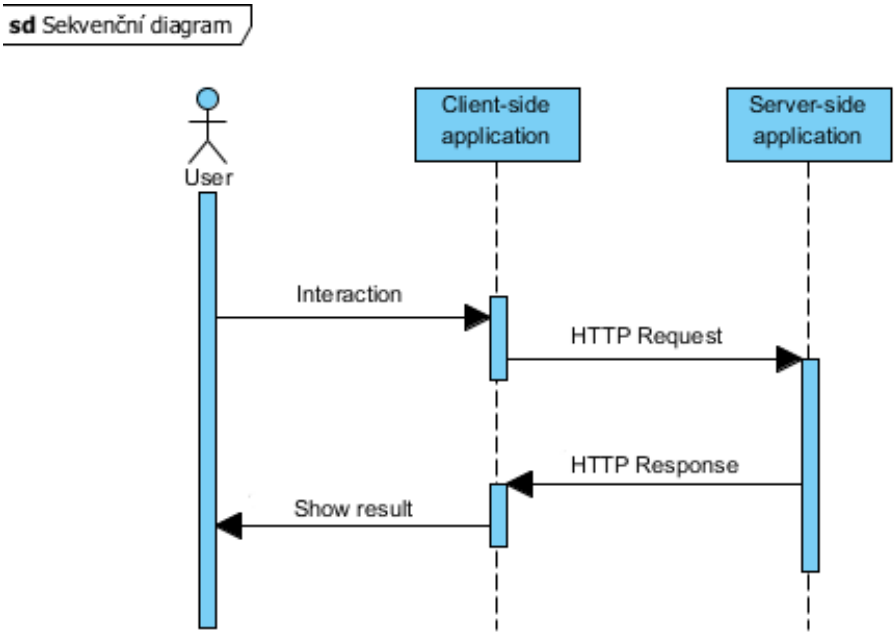
Systém bude poskytovat aplikační programové rozhraní pro automatizovaný přístup jiných aplikací. HTTP metody a URI zdrojů budou odpovídat standardům REST. Jelikož všechny nabízené funkce systému, kromě registrace a přihlášení, jsou dostupné pouze pro autorizované uživatele, bude přístup zabezpečen některou z běžně používaných metod.

3.1.3.2 Webová aplikace

Systém bude mít podobu webové aplikace, která bude dostupná přes internet. Její součástí bude uživatelské rozhraní, které umožní práci v nejčastěji používaných webových prohlížečích (Google Chrome, Mozilla Firefox, Internet Explorer, Opera, Safari). Podpora pro zobrazení v mobilních zařízeních není vyžadována. API bude také dostupné přes web.

3.1.3.3 Open-source a přidávání typů úkolů

Aplikace bude mít otevřený zdrojový kód, který bude dostupný na internetu. Kdokoli bude moct systém modifikovat a dále šířit i s provedenými změnami. Jelikož přidávání typů úloh bude vyžadovat definici funkcí pro zpracování načítaných nebo odesílaných dat, nebude možné přidávat typy úkolů přes uživatelské rozhraní, ale bude potřeba přidávat nový zdrojový kód aplikace. Přidávání



Obrázek 3.1: Ukázka komunikace mezi uživatelem, client-side a server-side aplikací

by mělo být co nejpřehlednější a mělo by vyžadovat pouze přidávání nových zdrojových kódů, nikoli změnu stávajících.

3.2 Návrh

3.2.1 Architektura

Na základě požadavků obsažených v analýze bude aplikace rozdělena na dvě části – serverová a klientská část. Serverová část bude sloužit k ukládání dat do databáze a bude zajišťovat aplikační logiku. Tato komponenta nebude mít grafické uživatelské rozhraní, ale bude k ní přistupováno přes API. Klientská část bude webová aplikace s grafickým UI, bude posílat požadavky na serverovou část přes API a získaná data bude zpracovávat a zobrazovat uživateli.

3.2.2 Serverová část

3.2.2.1 Typy úloh

Serverová část bude rozlišovat čtyři typy úloh, na které se budou mapovat úkoly z klientské části aplikace. Tyto typy jsou zcela abstraktní a liší se způsobem, jakým budou přidávány příspěvky. Prvním typem úkolu jsou takové,

kdy uživatel bude hodnotit vybráním jedné nebo více možností z nabídky. Volby v seznamu budou definovány pro celý úkol, to znamená, že pro všechny datové položky budou možnosti stejné. Tento způsob se hodí například pro sentimentální analýzu, odpovědi typu Ano/Ne nebo výběr objektů, které lze vidět na obrázku. Druhý typ je podobný jako první s tím rozdílem, že každá datová položka bude mít jiné možnosti. Volby jsou konfigurovány v souboru CSV, který se nahraje na server. Mezi takové úlohy se může řadit například rozhodování, zda je nějaké slovo pro daný text klíčové nebo ne. Pokud by nabídka podstatných slov vznikala na základě automatizované analýzy textu, je zřejmé, že seznam klíčových výrazů bude pro každý soubor jiný. Ve třetím typu úlohy je úkolem přispěvatele definovat klíčová slova. Jedná se ale o abstraktní typ, to znamená, že údaj nemusí být pouze zadaný text uživatelem, ale může obsahovat další informace, například délku výrazu a jeho pozici v textu. To, jaké údaje budou uloženy, závisí na klientské části a serverová část se o to nestará, pracuje s danou hodnotou jako s prostým textem. Posledním případem jsou odpovědi ve tvaru rozsáhlých textů, které se hodí například pro překlady a přepisy dat.

3.2.2.2 Konfigurační soubory

V některých případech je pro práci potřeba datové položky dále nakonfigurovat. Jedná se o situace, kdy chceme, aby přispěvatel poskytoval informace pouze k částem textu. Druhým případem jsou úkoly, kdy uživatel má vybrat jednu nebo několik z mnoha možností a tyto možnosti mají být pro datové položky různé. V obou případech jsou požadované informace získány z konfiguračních souborů ve formátu CSV.

Při definici částí textu bude každý řádek obsahovat jeden fragment jednoho záznamu. Na řádku budou čtyři položky odděleny čárkami:

fileName Název souboru, který je v rámci dané úlohy jedinečný, identifikuje datovou položku, k níž se vztahuje specifikovaný záznam.

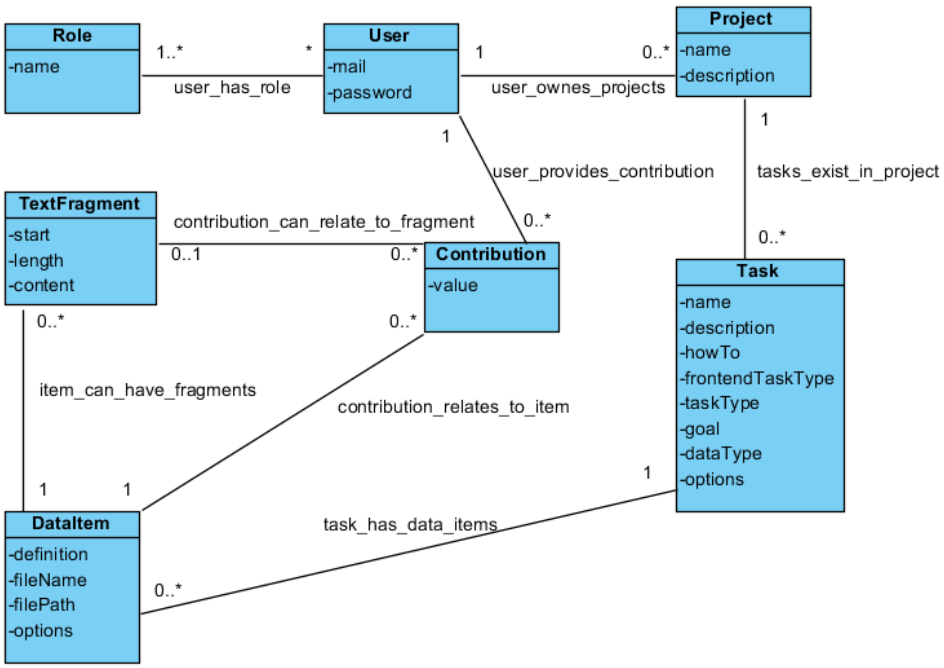
start Číselná hodnota značící počátek výrazu v textu.

length Číslo, které udává délku fragmentu.

content Samotný obsah části textu.

Jestliže v sobě datová položka bude obsahovat HTML tagy, pozice v textu a délka fragmentu bude udávána tak, jakoby tam značky nebyly. Pokud například bude obsah souboru priklad.txt Praha je hlavním městem České republiky. a zkoumanou částí bude slovo Praha, pak bude záznam v konfiguračním souboru vypadat: „priklad.txt,0,5,Praha“.

V situacích, kdy chceme k jednotlivým datovým položkám zadat možnosti k výběru, budeme opět používat soubor ve formátu CSV. Jeden řádek bude



Obrázek 3.2: Doménový model serverové části

definovat jednu možnost k jednomu záznamu. Řádek bude obsahovat dva údaje oddělené čárkou, kde

fileName Název souboru, který je v rámci dané úlohy jedinečný, identifikuje datovou položku, k níž se vztahuje specifikovaný záznam.

opt Jedna možnost k dané položce.

Důvod, proč není více možností na jedné řádce je ten, že by toto řešení selhávalo v případech, kdy by měla jedna možnost obsahovat čárku. V takovém případě by došlo k nesprávnému rozdělení na více voleb.

3.2.2.3 Doménový model

Pro snazší implementaci byl navržen doménový model, který obsahuje základní informace o entitách, které se budou v systému evidovat, jejich atributy a vztahy mezi nimi. Třídy v implementaci se od návrhu mohou lišit především z důvodu jednoduší manipulace s daty.

Role Tato entita ukládá informaci o uživatelských rolích v systému.

- Atributy:
  - name – název role

- Vztahy:
  - s entitou User – každý uživatel bude mít jednu nebo více rolí, jednu roli bude mít libovolný počet uživatelů

User Entita ukládající informace o uživatelích.

- Atributy
  - mail – emailová adresa
  - password – zašifrovaná podoba hesla
- Vztahy
  - s entitou Role – viz vztahy entity Role
  - s entitou Project – každý projekt bude vlastněn právě jedním uživatelem, uživatel může mít libovolné množství projektů
  - s entitou Contribution – každý uživatel může být autorem libovolného množství příspěvků, jeden příspěvek má vždy právě jednoho autora

Project Entita ukládající informace o projektech.

- Atributy
  - name – název projektu
  - description – popis projektu
- Vztahy
  - s entitou User – viz vztahy entity User
  - s entitou Task – každý projekt může obsahovat libovolný počet úkolů a jeden úkol náleží vždy přesně do jednoho projektu

Task Entita ukládající jednotlivé úkoly.

- Atributy
  - name – název úkolu
  - description – popis úkolu
  - howTo – návod, jak postupovat při vyplňování úkolu
  - state – stav úkolu, ve kterém se nachází: pozastavený, probíhající, dokončený
  - dataType – typ dat: text, image, audio, video
  - taskType – identifikátor typu úkolu na serverové části, číslo 0 – 4, viz Typy úloh
  - frontendTaskType – identifikátor typů úkolu v klientské části
  - goal – požadovaný počet vyplnění úkolu

  - options – pole možností pro výběr v případech, kdy uživatel volí jednu nebo více možností z nabídky a ta je stejná pro všechny datové položky

- Vztahy
  - s entitou Project - viz vztahy entity Project
  - s entitou DataItem – jeden úkol má libovolný počet datových položek, každá datová položká náleží přesně do jednoho úkolu

DataItem Entita, která reprezentuje jednu datovou položku v úkolu.

- Atributy
  - definition – definice datové položky, text u textových datových typů nebo cesta k souboru u multimediálních typů
  - fileName – název souboru, jedinečný v daném úkolu
  - filePath – cesta k souboru, jedinečná v celém systému
  - options – pole možností pro výběr v případech, kdy uživatel volí jednu nebo více možností z nabídky a ta je různá pro jednotlivé datové položky
- Vztahy
  - s entitou Task - viz vztahy entity Task
  - s entitou TextFragment – textová datová položka může být dělena na libovolný počet textových fragmentů, jeden fragment náleží přesně k jedné datové položce
  - s entitou Contribution – jeden příspěvek se vždy týká právě jedné datové položky, jedna položka může mít libovolný počet příspěvků

TextFragment Entita reprezentující část textové datové položky, lze k ní přidávat příspěvky.

- Atributy
  - start – pozice prvního znaku fragmentu v textu
  - length – délka textového fragmentu
  - content – obsah fragmentu
- Vztahy
  - s entitou DataItem - viz vztahy entity DataItem
  - s entitou Contribution – jeden příspěvek se může týkat jednoho nebo žádného textového fragmentu, jeden fragment může mít libovolné množství příspěvků

Contribution Entita reprezentující příspěvek uživatele.

- Atributy
  - value – hodnota zadaná uživatelem
- Vztahy
  - s entitou DataItem - viz vztahy entity DataItem
  - s entitou TextFragment - viz vztahy entity TextFragment
  - s entitou User - viz vztahy entity User

3.2.2.4 Návrh API

API je navrženo tak, aby odpovídalo principům REST (Representational State Transfer). Funkce API lze rozdělit do čtyř skupin podle povolení přístupu. V první skupině je jediný požadavek a to registrace nového uživatele. Tento úkol může provádět jakýkoli uživatel, to znamená, že v požadavku nemusí být definována hlavička Authenticate. V druhé skupině jsou URL pro přístup k entitám uživatelů. Adresa je ve tvaru /api/users/user_id a je možné zde získat informace o uživateli, změnit heslo nebo záznam odstranit z databáze. Tyto URL mohou využívat všechny registrované osoby bez ohledu na roli v systému, každý uživatel ovšem může přistupovat pouze ke svým údajům. Třetí skupinou jsou metody povolené pouze uživatelům, kteří chtějí obohatit svá data. Tyto požadavky začínají vždy řetězcem /api/users/user_id/*, kde user_id je identifikátor uživatele, k jehož datům přistupujeme. Funkce může využívat pouze majitel daného identifikátoru. Poslední skupinou jsou routy pro přispěvatele, které slouží především ke čtení informací o projektech, úlohách a datových položkách. Jedinou výjimkou je vytváření nových údajů o datech v aplikaci.

- /api/authenticate
  - POST - ověření, zda v systému existuje uživatel se zadaným emailem a má dané heslo
    - Odpovědi - 200 OK, 400 Bad request, 401 Unauthorized
- /api/users
  - GET - seznam všech uživatelů
    - Odpovědi - 200 OK, 401 Unauthorized, 403 Forbidden
  - POST - vytvoření nového uživatele
    - Odpovědi - 201 Created, 400 Bad request 409 Conflict
- /api/users/user_id
  - GET - výpis informací o uživateli
    - Odpovědi - 200 OK, 401 Unauthorized, 403 Forbidden, 404 Not found

  - PUT - editace informací o uživateli
    - Odpovědi - 200 OK, 400 Bad request, 401 Unauthorized, 403 Forbidden, 404 Not found
  - DELETE - odstranění záznamu uživatele
    - Odpovědi - 204 No content, 401 Unauthorized, 403 Forbidden, 404 Not found

- /api/users/user_id/projects
  - GET - výpis projektů uživatele s daným identifikátorem
    - Odpovědi - Odpovědi - 200 OK, 401 Unauthorized, 403 Forbidden, 404 Not found
  - POST - vytvoření nového projektu uživatele
    - Odpovědi - Odpovědi - 201 Created, 400 Bad request, 401 Unauthorized, 403 Forbidden, 404 Not found

- /api/users/user_id/projects/project_id
  - GET - výpis informací o projektu
    - Odpovědi - 200 OK, 401 Unauthorized, 403 Forbidden, 404 Not found
  - PUT - editace informací o projektu
    - Odpovědi - 200 OK, 400 Bad request, 401 Unauthorized, 403 Forbidden, 404 Not found
  - DELETE - odstranění projektu
    - Odpovědi - 204 No content, 401 Unauthorized, 403 Forbidden, 404 Not found

- /api/users/user_id/projects/project_id/tasks
  - GET - výpis úkolů v projektu
    - Odpovědi - 200 OK, 401 Unauthorized, 403 Forbidden, 404 Not found
  - POST - vytvoření nového úkolu v projektu
    - Odpovědi - 201 Created, 400 Bad request, 401 Unauthorized, 403 Forbidden, 404 Not found

- /api/users/user_id/projects/project_id/tasks/task_id
  - GET - zobrazení detailu úkolu
    - Odpovědi - 200 OK, 401 Unauthorized, 403 Forbidden, 404 Not found

  - PUT- editace informací o úkolu
    - Odpovědi - 200 OK, 400 Bad request, 401 Unauthorized, 403 Forbidden, 404 Not found
  - DELETE - odstranění úkolu
    - Odpovědi - 204 No content, 401 Unauthorized, 403 Forbidden, 404 Not found

- /api/users/user_id/projects/project_id/tasks/task_id/export
  - GET - export výsledků obohacení dat
    - Odpovědi - 200 OK, 401 Unauthorized, 403 Forbidden, 404 Not found

- /api/users/user_id/projects/project_id/tasks/task_id/data
  - GET - výpis datových položek v úkolu
    - Odpovědi - 200 OK, 401 Unauthorized, 403 Forbidden, 404 Not found
  - POST - upload datových položek
    - Odpovědi - 201 Created, 400 Bad request, 401 Unauthorized, 403 Forbidden, 404 Not found
  - PUT - konfigurace datových položek
    - Odpovědi - 200 OK, 400 Bad request, 401 Unauthorized, 403 Forbidden

- /api/users/user_id/projects/project_id/tasks/task_id/data/data_id
  - GET - zobrazení konkrétní datové položky
    - Odpovědi - 200 OK, 401 Unauthorized, 403 Forbidden, 404 Not found
  - DELETE - odstranění datové položky
    - Odpovědi - 204 No content, 401 Unauthorized, 403 Forbidden, 404 Not found

- /api/../projects/project_id/tasks/task_id/data/data_id/contributions
  - GET - zobrazení obohacení konkrétní datové položky
    - Odpovědi - 200 OK, 401 Unauthorized, 403 Forbidden, 404 Not found

- /api/projects
  - GET - zobrazení všech projektů v systému

        * Odpovědi - 200 OK, 401 Unauthorized, 403 Forbidden

- /api/projects/project_id
  - GET - zobrazení konkrétního projektu
    * Odpovědi - 200 OK, 401 Unauthorized, 403 Forbidden, 404 Not found
- /api/projects/project_id/tasks
  - GET - vypsání úkolů v projektu
    * Odpovědi - 200 OK, 401 Unauthorized, 403 Forbidden, 404 Not found
- /api/projects/project_id/tasks/task_id
  - GET - získání konkrétního úkolu
    * Odpovědi - 200 OK, 401 Unauthorized, 403 Forbidden, 404 Not found
  - POST - přidání ohodnocení celého úkolu
    * Odpovědi - 201 Created, 400 Bad request, 401 Unauthorized, 403 Forbidden, 404 Not found
- /api/projects/project_id/tasks/task_id/data
  - GET - zobrazení datových položek v úkolu
    * Odpovědi - 200 OK, 401 Unauthorized, 403 Forbidden, 404 Not found
- /api/projects/project_id/tasks/task_id/datadata_id
  - GET - zobrazení konkrétní datové položky
    * Odpovědi - 200 OK, 401 Unauthorized, 403 Forbidden, 404 Not found
- /api/projects/project_id/tasks/task_id/data/data_id/contributions
  - POST - přidání nového příspěvku k datové položce
    * Odpovědi - 201 Created, 400 Bad request, 401 Unauthorized, 403 Forbidden, 404 Not found
- /api/tasks
  - GET - zobrazení všech úkolů v systému
    * Odpovědi - 200 OK, 401 Unauthorized, 403 Forbidden

3.2.3 Klientská část

3.2.3.1 Typy úloh

Zatímco serverová část definuje kvůli ukládání velice abstraktní typy úloh, klientská aplikace bude svým uživatelům nabízet již konkrétní úkoly vztahující se k jednomu typu dat. To znamená například popis obrázků, přepis audio souborů, hledání klíčových slov v textu a tak dále. Přidání nového typu by mělo být co nejsnazší, aby programátor mohl v případě potřeby vytvořit požadovanou funkcionalitu.

3.2.3.2 Rozdělení souborů

V klientské části aplikace se bude nacházet několik typů zdrojových souborů. Prvním z nich jsou šablony, které mají za úkol zobrazit data uživateli a získávat od něj potřebný vstup. Šablony budou tvořeny klasickými HTML stránkami a formátované CSS styly. Způsob, jakým se budou data předávat z modelu do šablon, závisí na konkrétní použité technologii.

Druhým typem souborů budou takové, které zajišťují logiku aplikace – model. Jejich úkolem bude především komunikace se serverovou částí aplikace přes API. To samozřejmě zahrnuje také posílání dat ve správném tvaru a případné změny dat tak, aby s nimi mohly pracovat obě části aplikace. Pokud bude třeba, modelové třídy budou zajišťovat další logiku, která se netýká komunikace se server-side aplikací.

Posledním typem souborů jsou kontrolery, které budou mít za úkol správně reagovat na požadavky uživatele. Po přístupu na danou stránku budou vědět, jaká se má použít šablona a jaká data získaná modelem by do ní měla být vložena. Jestliže uživatel přistoupí na stránku, která neexistuje, kontrolery zobrazí příslušné chybové hlášení.

Bude snaha v aplikaci co nejvíce oddělit soubory, které souvisí s vnitřní implementací aplikace od těch, které poskytují implementaci typů úloh. Cílem tohoto snažení bude, aby se programátor, který bude chtít přidat nebo změnit typ úkolu, nemusel zabývat zdrojovými kódy, které jsou pro něj nedůležité.

3.3 Analýza vývojových nástrojů a technologií

3.3.1 REST

Representational state transfer je způsob identifikace zdrojů a metod přístupu k nim. Konkrétně to znamená, že jednotlivé datové zdroje jsou identifikovány URI adresami a operace, která se má s entitou vykonat, je určena HTTP metodou. Metoda GET znamená čtení, ať už se jedná o seznam položek nebo o jednu, která je jednoznačně identifikována. Metoda POST vytváří nový zdroj. Identifikátor je vždy automaticky generovány, to znamená, že při vkládání záznamu do databáze nebude URL obsahovat ID nového záznamu. Me-

toda PUT je použita k editaci entity s konkrétním identifikátorem a DELETE slouží k odstranění konkrétního zdroje.[12]

Požadavek je po odeslání zpracován serverem a na základě výsledku operace je vrácena klientovi odpověď s kódem, který indikuje úspěch nebo druh chyby, která nastala. V aplikaci je použito několik základních kódů odpovědí:

200 OK Tento kód indikuje, že požadavek byl proveden bez chyb. Typicky je vracen při získávání nějakých informací o zdroji, popřípadě při jeho modifikaci. Tělo odpovědi pak obsahuje další data, která chtěl klient získat.[12]

201 Created Při úspěšném vytvoření nové entity obsahuje odpověď tento stavový kód. V hlavičce Location je pak uvedeno URI nově vytvořeného záznamu. Tělo odpovědi může obsahovat další informace.[12]

204 No Content Tento kód znamená, že požadavek byl úspěšně vykonán, ovšem neexistují žádná data, která by měla být vrácena klientovi. Taková situace nastává například při úspěšném odstranění záznamu z databáze.[12]

400 Bad request Tento chybový kód značí, že požadavek nebyl nějakým způsobem správný a druh chyby není blíže specifikován. V aplikaci je použit v případech, kdy klient vytváří nebo modifikuje nějaký záznam, ovšem v požadavku chybí nebo nejsou validní nějaká data. Pokud například při registraci nového uživatele není uvedena emailová adresa, je odeslán právě tento kód s popisem chyby.[12]

401 Unauthorized Chyba, která nastává v případech, kdy se nepodařilo ověřit identitu uživatele. Jedná se o případy, kdy požadavek neobsahuje hlavičku Authenticate nebo její hodnota není správná, to znamená, že uživatel zadal nesprávně přihlašovací údaje.[12]

403 Forbidden Uživatel se snaží vykonat operaci, ke které nemá oprávnění. Zatímco u chyby 401 se vůbec nepodařilo identifikovat uživatele, v tomto případě je autentizační hlavička v pořádku, ovšem osoba není oprávněna danou operaci vykonat. Ve většině případů nemá uživatel požadovanou roli, popřípadě se snaží přistupovat k údajům jiného uživatele.[12]

404 Not found Klient se snaží přistupovat ke zdroji, který se nepodařilo najít. Může se jednat o případy, kdy je přistupováno na nespecifikovanou adresu URI, tedy na takovou, která není v systému definována. Druhou situací, která může nastat, je ta, že uživatel chce přistoupit k entitě přes její identifikátor, ovšem zdroj s daným ID neexistuje.[12]

406 Not Acceptable Tato chyba nastává v případě, kdy uživatel v hlavičce Accept požaduje vrácení odpovědi ve formátu, který aplikace nepodporuje. Tato chyba může nastat při zaslání jakéhokoli požadavku, proto ji v návrhu API neuvádím.[12]

409 Conflict Tento stavový kód je odeslán v případech, kdy klient chce vytvořit entitu s údajem, který by měl být unikátní, v systému už ale zdroj s uvedenou hodnotou existuje. V aplikaci je tento kód vracen pouze při vytváření uživatele, kdy zadaná emailová adresa je v databázi již evidována.[12]

415 Unsuported Media Type Tento chybový kód je odeslán v případě, že uživatel odeslal požadavek, který má v hlavičce Content-Type uveden formát, který není aplikace schopna zpracovat. Tato chyba může také nastat při jakémkoli požadavku, proto není v seznamu API uvedena.[12]

500 Internal Server Error Tato chyba nastává při vzniku vnitřní výjimky na serveru. Například pokud selže pokus o připojení k databázi, nebo je provedena nepřípustná operace kvůli programátorské chybě.[12]

3.3.2 Node.js

Node.js je velice výkonné JavaScriptové prostředí postavené na interpretu V8 z Google Chrome, které umožňuje používat tento populární jazyk na serverové části. To přináší některé výhody, například to, že pro celý projekt využíváme jednu technologii a nemusíme jich kombinovat více (PHP na serveru a JavaScript v klientské části).[13][14]

Obsluha požadavků probíhá v jednom vlákně, což má za následek rychlejší zpracování. „Node.js pracuje pouze v jednom vlákně, což je rozdíl oproti častějším případům, kdy webový server odpověď na každý požadavek zpracovává v jiném vlákně. Známe to všichni například z kombinace PHP + Apache, kdy pro každý požadavek Apache alokuje určité množství paměti, ve které PHP zpracuje daný HTTP požadavek. I v případě, že je odpověď záležitostí jednoduchého skriptu, musí dojít např. k načtení konfigurace projektu, inicializaci spojení s databází, načtení celého frameworku, routeru atd., což bývá často zbytečné a neefektivní. Protože Node.js pracuje pouze v jednom vlákně, při spuštění aplikace (před zpracováním prvního HTTP požadavku) dojde k načtení a inicializaci všeho potřebného a jakmile je vše načteno, Node.js poslouchá příchozí požadavky a ty pak směřuje na konkrétní controllery a akce (v případě MVC architektury).“[14]

Další vlastností, kterou Node.js disponuje, je asynchronní zpracování. „Dále se můžete o Node.js dočíst, že používá událostmi řízený (event-driven) neblokující I/O model. Co to přesně znamená, si opět vysvětlíme na příkladu s PHP. V PHP píšete obvykle jeden příkaz na každý řádek. Jakmile skript zpracuje jeden příkaz, pokračuje dále. Podstatné je, že dokud není dokončen předchozí příkaz, není možné zpracovat ten další, protože jeden příkaz blokuje ostatní. Takže např. po dokončení objednávky v e-shopu nejprve vložíte objednávku do databáze, čekáte, dokud nepřijde odpověď od databáze, dále odešlete e-mail správci, počkáte, dokud neodejde, pak e-mail zákazníkovi atd. V Node.js

můžete psát také tímto způsobem, ale pouze tam, kde nedochází k zpracování požadavků uživatele (např. při zmíněném načítání konfigurace, lokalizaci, inicializaci frameworku ap.). V případě zpracování konkrétního HTTP požadavku už vždy musíte používat asynchronní metody a funkce, protože pracujete v jednom vlákně a synchronní metody zablokují celý server, dokud nejsou zpracovány.“[14]

Výpis kódu 3.1: Příklad událostmi řízeného chování

```
task.save(function (err) {
  if (err) {
    return callback(err);
  }
  return callback();
});
```

Díky svým vlastnostem se Node.js uplatní v některých případech více a v některých méně. „Felix Geisendörfer sepsal základní případy užití, kde Node.js najde své uplatnění, a také případy, kde byste měli použít jinou technologii. Node.js je velmi vhodné použít, pokud chcete vytvářet tyto typy projektů:

1. Aplikace s RESTful/JSON API

2. Single-page aplikace (např. ve spolupráci s vynikajícím frameworkem AngularJS)

3. Real-time aplikace (Socket.io, frameworky Derby.js či Meteor)

Naopak se Node.js nehodí pro projekty, které jsou velmi náročné na CPU. Jako příklad Felix uvádí software pro kódování videa.“[14]

Základní funkce zabudované v Node.js lze rozšířit pomocí modulů, nebo také balíčků. Balíčky lze rozdělit do dvou skupin. V prvním jsou takové, které jsou již součástí Node.js a není nutné je instalovat, například modul HTTP pro pracování z požadavky a odpověďmi, FS pro manipulaci se souborovým systémem nebo URL pro zpracování internetových adres. Ve druhé skupině jsou moduly, které vytvořila komunita programátorů z celého světa. Těchto balíčků je ohromné množství, slouží k různým účelům a samozřejmě dosahují různých kvalit. „Všechny moduly je možné procházet přes webové rozhraní na npmjs.org. Kromě hlavního webu je dobré vědět o Node Toolbox, kde jsou moduly řazeny do mnoha kategorií, a také o nástroji Nipster, kde je možné moduly procházet např. podle toho, jak jsou oblíbené na Githubu. Vedle zmíněných nástrojů můžete moduly hledat na wiki projektu (zde však ani zdaleka nejsou všechny moduly).“[14]

Pro zprovoznění komunitních balíčků je potřeba jejich instalace. Toho lze nejsnáze dosáhnout pomocí konzolového nástroje npm. Existují i další, ovšem tento je přímo součástí Node.js. Základní použití je zadání příkazu npm install

nazev_balicku, kdy je modul se zadaným názvem stažen z internetu a připraven k použití. Požadované součásti je možné také specifikovat v souboru package.json, který obsahuje informace o projektu, například autor, název, verze, odkaz na repozitář a podobně. Kromě těchto údajů je zde sekce dependencies, kde lze definovat moduly potřebné pro běh aplikace. Instalace proběhne zadáním příkazu npm install ve stejném adresáři, kde se soubor package.json nachází. Pokud bychom chtěli některé balíčky instalovat do centrálního adresáře a zpřístupnit je tak všem projektům, které na serveru poběží, příkaz zadáme s přepínačem -g.[14]

3.3.3 Node.js balíčky

Do aplikace je doinstalováno několik modulů, které nejsou součástí Node.js. Nejdůležitější z nich jsou Express, což je framework pro usnadnění práce, a Mongoose, který slouží k manipulaci s databází MongoDB v prostředí Node.js. Oba dva nástroje nabízí mnoho výhod, a proto se jim budeme podrobně věnovat později. Pro testování jsou použity nástroje Mocha a Chai, kterým bude také věnována větší pozornost.

Pro šifrování hesel uživatelů v databázi bude použit algoritmus bcrypt, který je realizovaný v knihovně bcrypt-nodejs. Body-parser je nástroj, který pomáhá lépe zpracovávat požadavky, to znamená, že například atributy v requestu POST jsou převedeny do tvaru JSON. Pro přihlašování uživatelů a práci s autorizačními hlavičkami požadavků a odpovědí jsou použity balíčky passport a passport-http. Kvůli uploadu dat je nainstalován modul multer, který zpracovává nahraný soubor a informace o něm opět převádí do notace JSON. Jelikož v aplikaci dochází k otevírání a čtení údajů ve formátu CSV, je použita knihovna fast-csv a pro parsování RDF dokumentů je použit nástroj n3. Oba dva balíčky slouží nejenom ke čtení, ale také zápisu, což je potřeba při exportu dat. Aby mohl server posílat také HTML stránky při přístupu na klientskou část aplikace, je použit šablonovací nástroj ejs. Posledním použitým modulem je morgan, který slouží k logování dění na serveru do konzole v hezké, formátované podobě.[14][15][16]

3.3.3.1 Express

Express je minimalistický framework nad Node.js sloužící k vytváření webových aplikací. Tento nástroj nepředstavuje obrovské „řešení pro všechno“, ale snaží se zjednodušit základní práci. Příkladem může být routing, kdy chceme volat dané funkce podle toho, jaký přijde request na server. Jestliže pro příchozí požadavek není routovací pravidlo specifikováno, automaticky se odešle odpověď s kódem 404.[18]

Silným nástrojem ve frameworku Express je tak zvaný middleware, což je funkce, která má k dispozici objekty request, response a odkaz na další metodu. Zpracování požadavků tak pracuje na bázi pipeliningu, kdy každý

middleware nějakým způsobem zpracuje požadavek a pak jej předá ke zpracování dál. Příkladem využití může být například logování. Nejdříve bude request předán metodě, která provede zápis informací o požadavku, a následně bude pokračovat k dalšímu zpracování. Druhým příkladem může být kontrola oprávnění uživatele. Jestliže by naše aplikace obsahovala pouze chráněné adresy, předá se požadavek nejdříve middlewaru, který zkontroluje autorizační hlavičky a buď pošle odpověď, že uživateli není povolen přístup, nebo je požadavek předán k dalšímu zpracování. Middleware funkcí lze za sebe řetězit libovolné množství.[18]

Výpis kódu 3.2: Příklad definice middleware v Express

```
isAuthorized: function (req, res, next) {
  if (req.user._id.toString() !== req.params.uid.toString()) {
    return next(errors.Forbidden());
  }
  return next();
},
```

Další užitečnou vlastností, kterou Express nabízí, je zpracování chyb. Pokud v aplikaci nastane výjimka, například nelze vložit do databáze uživatele kvůli existujícímu uživatelskému jménu, je možné poslat hned odpověď s informací o nastalé situaci, protože každý middleware má referenci na objekt respose. V takových situacích ovšem většinou chceme vykonat další akce, například chybu zalogovat. To by ale znamenalo, že budou v mnoha zdrojových souborech opakující se instrukce pro nakládání s výjimečnými případy. Tento problém lze vyřešit díky řízení chyb, kdy informace o výjimce předáváme dále do další funkce. Každá metoda, která pak bude spuštěna, nejdříve kontroluje, zda od předchozího middlewaru nepřišlo hlášení o chybě, a pokud ano, jenom údaje předává opět dál. O zpracování chybových odpovědí se pak na konci stará jediná funkce, která může provést zalogování a podle povahy chyby může rozhodnout, jaký kód a hlášku odešle klientovi.[18]

Výpis kódu 3.3: Příklad zpracování chyb v Express

```
var express = require('express');
var app = express();

app.use(function(err, req, res, next){
  console.error(err.message);
  if (err.status) {
    res.status(err.status).json(err.message);
  } else {
    res.status(500).json(err.message);
  }
});
```

3.3.3.2 Mongoose

Mongoose je modul v Node.js, který slouží pro přístup k databázi MongoDB. Poskytuje podobné funkcionality jako různé ORM frameworky, například Doctrine v PHP. „Mongoose nabízí nástroje pro CRUD operace (vytváření, čtení, editaci a mazání) nebo třeba umožňuje provádět validaci před vložením dokumentu do kolekce (např. máte jistotu, že bude v databázi vždy e-mail nebo URL).“[14] Prostřednictvím tohoto balíčku se nejdříve nastaví spojení s databází a definují se schémata a dokumenty, které se budou v úložišti skladovat. Kromě již zmíněných create, read, update a delete funkcí lze definovat mnoho dalších operací. Například vykonání nějakého příkazu před uložením entity, vytváření asociací přes více kolekcí nebo úprava dat po načtení z databáze před tím, než s nimi budeme dále pracovat. Samozřejmě s načtenými záznamy se dá různě manipulovat, například řadit podle určitých kritérií, limitovat zobrazený počet výsledků nebo omezit vypisované atributy. Vlastní dotazy do databáze se dají definovat jako funkce daného schématu a tím pádem jsou vždy dostupné a nemusí se vytvářet stále znovu.[14][19]

Výpis kódu 3.4: Ukázka vytvoření struktury kolekce v Mongoose

```
var mongoose = require('mongoose');

var projectSchema = mongoose.Schema({
  name: { type: String, required: true },
  description: { type: String },
  userId: { type: String, required: true }
});
```

Výpis kódu 3.5: Ukázka načtení dat z MongoDB pomocí Mongoose

```
var Project = require('../model/schemas/Project.js');

Project.find({}, function (err, projects) {
  if (err)
    return next(err);
  successes.Success(res, projects);
});
```

3.3.3.3 Mocha a Chai

K testování tříd i API jsou použity Node.js moduly Mocha a Chai. Chai je knihovna výrazů, které se používají pro testování. Díky struktuře tohoto nástroje je možné vytvářet rozmanité testy v různých případech. Můžeme vytvořit nějakou třídu, v ní definovat funkci a v Chai potom vývojář řekne, jaké výstupy se očekávají při zadání různých parametrů, popřípadě, jaké situace by neměly nastat. Je možné kontrolovat konkrétní hodnoty, typy tříd, zda nějaký objekt není null nebo jestli nemá undefined proměnnou. Samozřejmostí jsou také testy, zda se během výpočtu nevyskytla chyba.[16][20]

Nástroj Mocha je pak určen k řízení spouštěných testů. Jedná se o konzolovou aplikaci, která je spuštěna s parametrem definujícím umístění souborů s testy v adresářové struktuře. Díky tomuto nástroji lze kontrolovat synchronní i asynchronní chování kódu. Ke každému testu můžeme definovat informace, k jaké třídě se vztahují, jaká bude volaná funkce, a také popis, co chceme daným testem zjistit. V jednom případě se například chceme ujistit, zda funkce neskončí chybou, a následně nás zajímá, zda je vrácen výsledek, který očekáváme. Mocha také rozumí dalším funkcím, které jsou zapsané přímo v testovacím nebo konfiguračním souboru. Lze tak například vykonávat nějakou akci před spuštěním každého testu, nebo jenom v některých případech. Nějakou akci lze samozřejmě vykonávat i po dokončení kontroly. Tyto metody mohou opět řídit synchronní i asynchronní události. Příkladem může být navázání spojení s databází před každým testem, popřípadě vymazání obsahu databáze a odpojení od serveru po skončení.[16][21]

Výpis kódu 3.6: Příklad testu asynchronní funkce

```
var chai = require('chai');
var expect = chai.expect;

describe('UserSchema', function () {
  describe('#save()', function () {
    it('After saving User, mail should be lowercased and password
        encrypted',
    function (done) {
      var user = new User();
      user.mail = "Tester@tesTing.com";
      user.password = "testpassword";
      user.save(function (err) {
        if (err)
          return done(err);
        expect(user.mail).to.equal("tester@testing.com");
        bcrypt.compare("testpassword", user.password, function (
            err, isMatch) {
          if (err)
            return done(err);
          expect(isMatch).to.equal(true);
          done();
        });
      });
    });
  });
});
```

3.3.4 MongoDB

MongoDB patří mezi takzvané NoSQL databáze. Místo tradičních relačních tabulek používá pro ukládání dat formát BSON (binary JSON), který je klasickému JSON velice podobný. Tento zápis dat však umožňuje například specifikovat datové typy položek.[14]

Velikou výhodou MongoDB oproti relačním databázím je ta, že je velice jednoduchá na používání. Při použití nějaké SQL databáze je nutné orientovat se v návrhu relačních databází, jejich struktury, normalizace a podobně. Tabulky mají pevně stanovené názvy a datový typ sloupečků a jejich změna v případě, kdy databáze již obsahuje nějaká data, může být velice komplikovaná. V MongoDB jsou data rozdělena do kolekcí a ty obsahují libovolný počet dokumentů, což jsou objekty ve formátu BSON. Každý záznam v dané kolekci může vypadat prakticky jinak, může obsahovat jiné názvy a typy hodnot. Strukturu kolekce totiž nevytváříme nijak dopředu, dokonce není ani potřeba dopředu danou kolekci vytvářet, ta vznikne až při prvním vložení dokumentu do databáze. Položky záznamu nemusí mít elementární tvar, naopak je možné vytvářet v jednom dokumentu libovolně složitou strukturu. Například v internetovém obchodě můžeme definovat kategorii zboží a jednotlivé položky mohou být součástí toho samého záznamu. Tím nám odpadá mnoho úkolů a starostí, které souvisí s relačními databázemi.[14][17]

Pro dotazování v SQL databázích je potřeba se nejdříve naučit alespoň základní příkazy pro hledání, vkládání, úpravu a mazání záznamů. Při větších databázích s mnoha tabulkami je pak také třeba znát instrukce obsahující konstrukce JOIN a podobně. V MongoDB lze všechna data získat jednoduchými dotazy.

MongoDB lze použít jako součást projektů realizovaných v mnoha programovacích jazycích. Oficiální stránka udává Python, Java, Node.js, C++, C# a Shell, lze však najít návody pro zprovoznění v dalších vývojových nástrojích, například PHP. Vše záleží na knihovnách, které pro danou platformu vznikly, pro Node.js je to například Mongoose, ale samozřejmě jich existuje daleko více.[17]

Z velké volnosti, kterou MongoDB přináší, ovšem plynou i určité nástrahy. Tím, že data nemají pevnou strukturu, programátor nemá zaručeno, že záznam bude obsahovat ty informace, které potřebuje, popřípadě že je najde pod názvem, který by očekával. Speciálně u JavaScriptu je práce s objekty velice volná, na rozdíl od jiných programovacích jazyků není podoba třídy předem definována, ale jakoukoli hodnotu s daným názvem přidáme, ta bude v objektu uložena. Kolekce v MongoDB předpokládá, že záznam o uživateli má atribut password, ale objekt vytvořený v JavaScriptu místo něj ukládá hesla do proměnné passwd. V takovém případě bude složité odhalit zdroj datových nekonzistentností, protože ani databáze ani skriptovací jazyk nebudou uživateli hlásit chybu. Další nevýhodou je, že databáze nabízí pouze omezenou velikost úložiště v 32 bitových systémech.

3.3.5 RDF a NIF

Resource Description Framework je standardní model pro zápis a výměnu dat na webu. Díky tomuto frameworku je možné ukládat k jednotlivým hodnotám také jejich sémantický význam. Například lze říci, že slovo jazyk je myšleno

jako část lidského těla a ne jako sníh nafoukaný na silnici. Bez použití RDF by toto nebylo možné a ačkoli by člověk významu slova nejspíše rozuměl díky kontextu, počítače takovou schopnost nemají. Pokud se bavíme o sémantickém webu, existuje mnoho slovníků, které nám dovolují snadno specifikovat nějaká data. Například zboží v internetovém obchodě, informace o osobách a adresách nebo vztahy mezi jednotlivými entitami. Jedním ze slovníku je také NLP Interchange Format (NIF), který se snaží zajistit spolupráci mezi nástroji pro NLP (Natural Language Processing – zpracování přirozeného jazyka), texty a anotacemi. Tento slovník je použit pro export a import dat v systému. Lze totiž specifikovat textové datové položky, části textů a vztahy mezi nimi. Také se u anotací mohou ukládat další informace, jedná se například o přesnost algoritmu, v našem případě se budou ukládat pouze informace o tom, kolikrát a jací uživatelé danou anotaci označili. RDF lze zapisovat v různých syntaxích, jednou z nich je také Turtle, která je velice přehledná, a proto bude použita v systému.[22][23][24]

V následující ukázce je vidět použití slovníku NIF v Turtle syntaxi pro specifikaci textů a jejich částí. Celý text, který je analyzován, je typu nif:Context a má vlastnosti nif:beginIndex a nif:endIndex, které určují začátek a konec pozice v textu. V tomto případě uvažujeme celý řetězec, proto je beginIndex roven nule a endIndex je jeho délka. Stejné vlastnosti mají také fragmenty textu, v tomto případě už říkají, kde se jednotlivé části nacházejí. Každý kousek textu je typu nif:Phrase. Predikát nif:referenceContext ukazuje na text, jehož součástí je slovní spojení. Další vlastností u nif:Phrase je nif-ann:AnnotationUnit ukazující na blank node, který obsahuje informace o anotaci. Atribut itsrdf:taIdentRef prázdného uzlu odkazuje na zdroj informace, v tomto případě je to URI uživatele, který provedl ohodnocení. Vlastnosti nif:String a nif:anchorOf pak obsahují samotnou textovou informací u celého řetězce, popřípadě jeho části.[23][24]

Výpis kódu 3.7: Příklad RDF dokumentu a slovníku NIF v syntaxi Turtle

```
@prefix nif: <http://persistence.uni-leipzig.org/nlp2rdf/
    ontologies/nif-core#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix nif-ann: <http://persistence.uni-leipzig.org/nlp2rdf/
    ontologies/nif-annotation#>.
@prefix itsrdf: <http://www.w3.org/2005/11/its/rdf#>.

<http://example.org/#doc=1.txt> a nif:String, nif:Context;
    nif:isString "Praha je hlavni a soucasne nejvetsi mesto Ceske
        republiky."^^xsd:string;
    nif:beginIndex "0"^^xsd:int;
    nif:endIndex "58"^^xsd:int.

<http://example.org/#doc=1.txt?char=0,5> a nif:String, nif:Phrase;
    nif:referenceContext <http://example.org/#doc=1.txt>;
    nif:beginIndex "0"^^xsd:int;
    nif:endIndex "5"^^xsd:int;
```

```
    nif:anchorOf "Praha"^^xsd:string;
    nif-ann:AnnotationUnit [ itsrdf:taIdentRef <http://example.org
        /#user=56f4fc34b5d614781e7ec470> ].

<http://example.org/#doc=1.txt?char=42,15> a nif:String, nif:
    Phrase;
    nif:referenceContext <http://example.org/#doc=1.txt>;
    nif:beginIndex "42"^^xsd:int;
    nif:endIndex "57"^^xsd:int;
    nif:anchorOf "Ceske republiky"^^xsd:string;
    nif-ann:AnnotationUnit [ itsrdf:taIdentRef <http://example.org
        /#user=56f4fc34b5d614781e7ec470> ].
```

3.3.6 AngularJS

Webové aplikace, které komunikují se serverem přes API, většinou vyžadují hodně JavaScriptu. Pro lepší přehlednost a strukturu takové prezentace je vhodné použít nějaký framework, jedním z nejznámějších je AngularJS. Tento nástroj je určen především pro single page stránky, to znamená, že se nenačítá vždy kompletně nový obsah, ale je změněna pouze část pomocí JavaScriptu. AngularJS je postaven na architektuře MVC, která dělí aplikaci na tři části – model, view, controller.

Začneme-li u zpracování požadavků, tak je třeba nejdříve definovat routovací pravidla. Ta říkají, jaká se má použít šablona a jaký kontroler do ní bude předávat data, jestliže přijde uživatel na stránku s danou adresou. Popřípadě je možné specifikovat, co se má vykonat, když požadovaná stránka neexistuje. Samozřejmě, pokud se informace na stránce nenačítají dynamicky, není třeba kontroler používat, stejně tak je možné, aby byl jeden použit pro více views, nebo jich jedna stránka měla několik. V kontroleru se předávají data a funkce do šablony. Pokud je například potřeba odesílat formulář na nějaké stránce, příslušná třída obsahuje definici funkce, která se o zpracování zadaných údajů postará. Data do stránky pak mohou být nastavena přímo v kontroleru, nebo mohou být získána třeba přes API z jiné aplikace.[14][25]

Výpis kódu 3.8: Příklad definice routovacích pravidel v Angular.js

```
var app = angular.module("Routing");

app.config(['$routeProvider', function ($routeProvider) {

    $routeProvider
        .when('/login', {
          controller: 'LoginController',
          templateUrl: '/templates/base/login.html'
        })
        .when('/change-password', {
          controller: 'UserController',
          templateUrl: '/templates/base/changePassword.html'
        })
```

```
        .otherwise({redirectTo: '/'});
  }]);
```

Views jsou v případě AngularJS HTML šablony, které je možné do sebe různě vnořovat a jsou pak vykresleny jako celek. Jestliže je potřeba vypsat obsah proměnné nebo vykonat a následně zobrazit výsledek nějaké funkce, zapisuje se to do dvojitých složených závorek – {{nazev_promenne}}. Při potřebě užití podmínek, cyklů a podobně se používají direktivy jako atributy elementů. Kromě velice široké nabídky již existujících konstruktů je možné definovat svoje vlastní. Poslední částí aplikace je model, jehož třídy v tomto případě budou pouze získávat data přes API ze serverové části a podle potřeby je modifikovat. Angular má pro komunikaci přes aplikační rozhraní zabudované silné nástroje, se kterými se pracuje opravdu snadno.[14][25]

Výpis kódu 3.9: Získávání dat přes API v AngularJS

```
var app = angular.module("Application");

app.controller('ContributionIndexController', ['$scope', '$http',
  function ($scope, $http) {

    $http.get('/api/tasks').success(function (data) {
      $scope.tasks = data;
    });
  }]);
```

Knihoven, které řeší architekturu aplikace v JavaScript je větší množství. Angular, který je momentálně vyvíjen a spravován společností Google, má několik dalších užitečných nástrojů, které z něj dělají velice oblíbenou platformu. „AngularJS obsahuje řadu zajímavých myšlenek, přičemž mezi nejužitečnější koncepty frameworku se obvykle uvádí Two Way Data-Binding, implementace Dependency Injection, testovatelnost, direktivy a znovupoužitelnost komponent.“[14]

První koncept, tedy Two Way Data-Binding, řeší synchronizaci stavu mezi modelem a view. Příkladem může být případ, kdy má uživatel zadat do formulářového políčka nějakou hodnotu a aplikace na ni má nějakým způsobem zareagovat. Pokud bychom takovou situaci řešili v prostém JavaScriptu, popřípadě pomocí jQuery, museli bychom v kódu vybrat pomocí selektorů příslušné elementy a pomocí funkce onchange kontrolovat, zda se zadání vstupu změní a na případnou změnu reagovat. Angular ovšem obsahuje direktivy, které umožní problém řešit mnohem efektivněji. Vstupnímu poli bychom přidali atribut ng-model="nazev", kde nazev představuje jméno proměnné v modelu. U výstupního elementu, ať už je to odstavec, nadpis nebo jiné formulářové pole, přidáme vlastnost ng-bind="nazev"a tím tyto dva elementy provážeme, jelikož pracují se stejnou proměnnou. Direktiva bind potom říká, že text v prvku bude stejný jako v proměnné „nazev“, to znamená stejná jako ve vstupním formulářovém poli. Samozřejmě hodnotu proměnné lze změnit i v modelu, takže pak

se do elementu promítnou data získaná například voláním funkce na nějaké API.[14][25]

Dalším konceptem je řešení Dependency Injection (DI), což je návrhový vzor, který řeší závislosti mezi částmi aplikace. „AngularJS obsahuje zabudovaný subsystém, který řeší DI napříč celou vyvíjenou aplikací. Umožňuje vždy přesně specifikovat, které jiné komponenty jsou uvnitř konkrétní komponenty používané.“[14] V definici třídy, například kontroleru, uvedeme služby, které zde budou použity. Služby mohou být interní, tedy součástí Angularu, ale také mohou být definovány vývojářem. Zabudované se poznají podle znaku dolaru na začátku názvu a patří mezi ně třeba $scope, která je velice důležitá pro předávání dat mezi view a modelem. Pokud například v šabloně máme formulářové pole s atributem ng-model="name", hodnotu, kterou uživatel zadal, najdeme v kontroleru jako atribut třídy $scope, tedy $scope.name. Samozřejmě to platí obráceně, ve $scope definujeme nějaká data, která se pak objeví v HTML stránce, popřípadě definujeme funkci, kterou lze v šabloně zavolat. Mezi další nabízené služby patří $routeParams, která umožňuje práci s parametry v URL adrese, $location sloužící mimo jiné k přesměrování, nebo $http, která dovoluje posílat požadavky do serverové části aplikace.[14][25]

3.3.7 Bootstrap

Bootstrap je podle svojí internetové prezentace nejpopulárnější framework pro vývoj client-side aplikací. Jedná se o hotové CSS a JavaScriptové soubory, které usnadňují tvorbu uživatelského rozhraní. Po integraci stažených zdrojových kódů do HTML šablon, získá uživatel dobře vypadající internetové stránky a o jejich další přizpůsobení pro uživatele se nemusí starat. První sadou souborů jsou kaskádové styly, které tak přinášejí kompletní řešení pro vzhled webu. Všechny prvky jsou stylizovány do nejmenších detailů a výsledek vypadá velice příjemně a jednotně. Framework dále nabízí různé vizuální komponenty od širokého výběru ikon, přes tlačítka a rozšířená formulářová pole, až po přehledně formátované tabulky. Díky JavaScipru má kodér k dispozici také interaktivní prvky, například modální okna, rozbalovací menu, nebo slideshow obrázků. Další velikou výhodou Bootstrapu je to, že návrh je již uzpůsoben pro responzivní design, to znamená stránky vytvořené pomocí tohoto frameworku budou správně zobrazeny na obrazovkách různých velikostí.[26]

Kapitola 4

Implementace

V této kapitole si popíšeme základní principy, jak byla aplikace implementována. Jedná se především o popis nejdůležitějších tříd a metod, za co jsou zodpovědné a za co naopak ne. Zdrojové kódy jsou děleny do různých skupin podle způsobu jejich použití, a proto i jejich popis je členěn do více podsekcí.

4.1 Adresářová struktura

- app – obsahuje zdrojové kódy serverové části aplikace
  - config – konfigurační soubory
  - controllers - kontrolery
  - components - další součásti aplikace
  - model – model aplikace
  - responses – definice zasílaných odpovědí
  - routing - routovací pravidla
- data – data, která uživatel nahraje na server
- node_modules – stažené balíčky Node.js
- public – zdrojové kódy klientské části aplikace
  - angular – zdrojové kódy AngularJS
  - css – kaskádové styly
  - fonts – složka s písmem obsažená ve frameworku Bootstrap
  - img – složka s obrázky
  - js – JavaScriptové kódy
  - templates – HTML šablony

  - index.html – layout, základní rozvržení stránky
- test – zdrojové kódy pro testování
  - api – testování API
  - data – pomocná data pro testování
  - schemas – testování schémat
- main.js – hlavní soubor aplikace
- package.json – informace o projektu a závislostech

4.2 Server-side aplikace

4.2.1 Routování

Routovací pravidla se nacházejí v souboru app/routing/routes.js. Hlavní třída, která se zde využívá, je Router, což je součást modulu Express. U této třídy se nejdříve zavolá funkce route s parametrem, která udává adresu dotazu. Po zavolání metody jsou zde další funkce, které pro jednotlivé metody požadavků získávají názvy akcí, které se mají zavolat, když přijde daný request. Kromě HTTP metod lze u Route třídy použít i funkci all, která je vykonána pro všechny požadavky na dané adrese. Díky systému middlewaru je možné nechat vykonat při příjmu požadavku libovolné množství operací. Následující kód ukazuje způsob použití, kdy při přístupu na všechny adresy začínající /api/users/:uid/ (správcovská část aplikace) se nejdříve kontrolují autentizační hlavičky, následně zda uživatel přistupuje do svého prostoru a zda má oprávnění administrátora.

Výpis kódu 4.1: Nastavení routování v klientské části aplikace

```
router.route('/api/users/:uid/*')
      .all(Auth.isAuthenticated, Auth.isAuthorized, Auth.isAdmin);
```

Pomocí routování je nastaveno, že adresy funkcí dostupných přes API budou začínat /api. Naopak pokud budou začínat jiným způsobem, bude tento požadavek předáván do klientské části aplikace. Konkrétně se říká, že se má vykreslit šablona index.html. Umístění šablon je pak definováno v hlavním souboru aplikace main.js.

4.2.2 Kontrolery

Třídy z kategorie kontrolery mají na starost manipulaci s entitami a schématy v modelu aplikace. Nabízejí funkce pro získání všech záznamů, nalezení položky s daným identifikátorem, vytvoření nových dat v databázi nebo jejich úpravu a mazání. Různé kontrolery pak mohou nabízet speciální funkcionality, například nalezení uživatele podle emailové adresy, nebo provádí různé

pomocné výpočty. Kontrolery jsou do aplikace zasazeny jako middleware, to znamená, že mají přístup k objektům request a response. Z informací o požadavku si získají údaje potřebné k provedení úkolu. Pokud se vyskytne nějaká chyba, je předána pomocí callbacku do dalšího middleware. V opačném případě je odeslána odpovídající odpověď.

V aplikaci existují tyto kontrolery

AuthController Tento kontroler zajišťuje autentizaci pomocí metody Basic. Vyhledá uživatele podle emailové adresy a vytvoří hash z uživatelského jména a hesla podle standardu tohoto způsobu ověření. Vypočtenou hodnotu pak zkontroluje proti poslanému textu v hlavičce Authorization v požadavku. Jestliže se oba řetězce shodují, je požadavek předán dalšímu middlewaru. Další funkce kontrolují, zda je uživatel oprávněn přistoupit na dané adresy, to znamená, zda má potřebné oprávnění a jestli přistupuje do svého prostoru a nesnaží se manipulovat s daty někoho jiného. Poslední funkce s názvem authorize slouží pro přihlašování. Zatímco všechny předchozí metody sloužily pouze jako ověřovací bezpečnostní mechanizmy pro přístup k jiným funkcionalitám aplikace, authorize zjišťuje, zda byly poslány správné přihlašovací údaje a podle toho hned posílá odpověď, nevykonávají se další funkce.

ContributionController Tato třída má na starost správu příspěvků uživatelů. Je možné nechat vrátit příspěvky k dané datové položce a vytvořit nový příspěvek. Neobsahuje funkce pro editaci a mazání, tyto dvě možnosti nebudou přispěvatelům k dispozici.

DataController Třída DataController slouží pro práci s datovými položkami. Jelikož nejsou záznamy vytvářeny přímo, ale jsou generovány na základě obsahu nějakého souboru, funkce newData předpokládá, že požadavek bude obsahovat uploadované soubory. Ty se předají funkci newDataFromDataFiles ve třídě ImportComponent, která se postará o parsing vstupu a vytvoření položek v databázi.

ProjectController Kontroler, který slouží pro práci s projekty. Vrací seznam všech projektů v systému, projekty patřící jednomu uživateli, nebo záznam s konkrétním ID. Samozřejmě obsahuje také funkce k vytváření, editaci a mazání projektů.

TaskController Tato třída je zodpovědná za správu úloh. Umožňuje vytváření, editaci, mazání. Pokud chceme zobrazit úlohy v databázi, je možné nechat vypsat všechny patřící do jednoho projektu nebo k danému uživateli, popřípadě lze získat detail úkolu s konkrétním identifikátorem.

UserController Kontroler slouží ke správě záznamů v kolekci uživatelů. Obsahuje funkce pro registraci, vypsání uživatele s konkrétním identifikáto-

rem a odstranění uživatele. Není možné zobrazit všechny záznamy, tato funkcionalita by nebyla ve výsledné aplikaci nikde využita.

ExportController Třída, která zajišťuje export dat a ohodnocení do souborů. Hlavními metodami jsou contributionAsCSV, která provádí zápis do souboru ve formátu CSV, a contributionAsNIF, která slouží pro vytvoření souboru s trojicemi ve formátu turtle se slovníkem NIF.

4.2.3 Schémata

Schémata jsou třídy, které představují entity v aplikaci. Díky mongoose jsou pak mapovány do databáze tak, že jedno schéma odpovídá jedné kolekci v MongoDB. Aby to tak mohlo být, všechny třídy jsou vytvářeny příkazem například var userSchema = mongoose.Schema(), kde parametr funkce je objekt definující danou kolekci. Pro položky přidané v dané kolekci se automaticky generuje identifikátor, jehož hodnota je dostupná pod atributem _id. V aplikaci existuje pět schémat - pro příspěvky, datové položky, úkoly, projekty a uživatele.

UserSchema Entita představující uživatele. Mezi atributy patří mail, password a roles, který představuje pole uživatelských rolí. V schématu je definovaných několik funkcí, které zjednodušují práci s entitami. První z nich nám zabezpečí, že před uložením nového záznamu je emailová adresa převedena na malá písmena. Díky tomu v databázi nebude docházet ke konfliktu adres, kdy jednou by stejná hodnota byla zadána různě. Další funkce zajišťuje zašifrování hesla před jeho uložením do databáze v případech, kdy bylo změněno. Další metody pracují s rolemi, jedna k entitě přidává oprávnění a druhá kontroluje, zda uživatel má potřebnou roli.

ProjectSchema Schéma představující projekt nemá žádné speciální atributy ani metody. Obsahuje pouze definici kolekce, to znamená, má specifikovány název projektu, jeho popis a identifikátor uživatele, který je jeho správcem.

TaskSchema TaskSchema představuje entitu reprezentující úkol. Tato třída má několik atributů, mezi něž patří název úkolu, popis, identifikátor projektu, do kterého patří, a ID správce úlohy. K dalším parametrům patří taskType, který říká, o jaký typ úkolu na serverové straně se jedná. Oproti tomu frontendTaskType je hodnota identifikátoru typu úlohy na klientské straně aplikace. Dalšími parametry jsou goal, což je počet ohodnocení, kterého chceme dosáhnout u jednotlivých datových položek. Stav úkolu je evidován v proměnné state, která může být buď stopped pro pozastavené úkoly, running pro probíhající a finished pro dokončené. Jelikož se jeden úkol vztahuje pouze k jednomu typu dat, je tato informace samozřejmě také evidována. Další proměnná nese název options a jedná

se o seznam možností v případech, kdy jich má uživatel několik vybrat z nabídky a tento seznam je stejný pro všechny datové položky v úkolu. Poslední údaj, který je evidován slouží k nastavení počtu zobrazených datových položek při jednom vyplnění. Pokud tedy úkol bude mít tisíce fotografií k ohodnocení, uživateli se jich zobrazí například jenom 10. Ten stejný uživatel ovšem může k jednomu úkolu přispívat vícekrát, ale ke každé datové položce maximálně jednou.

DataSchema Entita Data opět obsahuje několik atributů. Prvními z nich jsou fileName a filePath, což jsou název a cesta datového souboru, buď multimediálního, který se bude přímo zobrazovat, nebo textového, ze kterého byla data získána. Dalšími jsou dvě pole fragments a options. První z nich uchovává informace o částech textových dat. U každého fragmentu se uchovávají informace, kde začíná v textu, jakou má délku a jeho textový obsah. Druhé pole ukládá informace o možnostech v typech úkolů, kdy uživatel má jeden nebo více z nich vybrat. V případech, kdy jsou možnosti stejné pro všechny datové položky, jsou volby získávány z příslušného úkolu. V opačném případě jsou uloženy přímo u datového záznamu. Dalšími atributy jsou taskId, což je identifikátor úkolu, ke kterému datová položka spadá, a definition, který obsahuje text u textových dat nebo url souboru u multimediálních dat. Nakonec je třeba zmínit také pole contributors, které uchovává seznam uživatelů, kteří k položce přispěli. Na základě tohoto pole se uchovává počet ohodnocení datové položky a také je zajištěno, aby jeden uživatel nemohl ke stejné položce přispět vícekrát.

ContributionSchema Poslední schéma se týká příspěvků a obsahuje několik potřebných informací. První z nich je identifikátor dat, kterých se hodnocení týká, value zaznamenává hodnotu zadanou nebo vybranou uživatelem, identifikátor přispěvatele, který hodnocení poskytl a také číslo fragmentu v případě, že předmětem obohacení je pouze část textových dat.

4.2.4 Další komponenty

FSComponent Tato třída zjednodušuje práci se souborovým systémem. Konkrétně se jedná o vytváření cesty adresářů. V modulu fs, který se stará o práci s file systémem a je součástí Node.js, existuje pro vytváření složek pouze asynchronní funkce. Pokud navíc potřebujeme nové adresáře, které jsou různě vnořené, existující funkce si s tím nedokáže poradit. Pokud bychom navíc jednotlivé fragmenty psali za sebe, díky asynchronnímu zpracování snadno dojde k chybě, protože se systém může pokoušet o vytvoření složky v adresáři, který doposud neexistuje. Proto jsem implementoval vlastní metodu, která rozdělí cestu adresářů a pomocí callbacků vytvoří jednotlivé složky ve struktuře.

ImportComponent Tato třída slouží k importu dat ze souborů různých typů. Jestliže se jedná o textový dokument, dojde k jeho otevření, přečtení textu a uložení do datové entity. V případech multimediálních souborů, to znamená obrázky, audio a video záznamy, je do datové třídy uložen název a cesta k souboru, která se používá při jeho vykreslení. Speciálně jsou zpracovávány vstupy ve formátu RDF, kdy jeden soubor může obsahovat více datových záznamů a také může definovat jejich fragmenty. V takovém případě dojde ke zpracování pomocí knihovny n3.

Datové záznamy je možné konfigurovat, což se provádí také v této třídě. Prvním typem rozšíření je definování fragmentů, ke kterým je možné přidávat hodnocení u textových dat. V takovém případě je zpracováván soubor ve formátu CSV, kdy jsou čteny jednotlivé řádky, podle názvu souboru je vyhledán příslušný záznam a následně je vytvořena a uložena požadovaná část textu. Druhým způsobem konfigurace je specifikace možností při úkolech, kde cílem je zvolit jeden nebo více údajů a tato nabídka je pro každou datovou položku různá. Možnosti jsou opět definovány v CSV souboru, který je parsován, podle názvu je nalezena datová položka a k ní jsou přidávány jednotlivé možnosti.

4.3 Client-side aplikace

4.3.1 Šablony

Angular.js nabízí pokročilé možnosti vykreslování HTML stránek. Mezi obrovskou výhodu patří možnost libovolně do sebe šablony vnořovat. Základní šablonou je soubor index.html ve složce public. V ní jsou definované prvky, které mají všechny stránky společné, to znamená nadpis a hlavní menu. V těle obsahuje blokový element div s atributem ng-view. Díky této direktivě se uvnitř elementu bude vykreslovat obsah jednotlivých podstránek.

4.3.2 Routování

Stejně jako na straně serveru i zde je potřeba definovat routovací pravidla, aby aplikace věděla, jakou šablonu a kontroler má při přístupu na stránku použít. Pravidla jsou definována v souboru public/js/routing.js. Zde se pracuje s interním objektem Angularu $routeProvider. Pravidlo je pak definováno zavoláním funkce when tohoto objektu se dvěma parametry. První parametr udává adresu, pro kterou je dané pravidlo použito. Druhý parametr je objekt, který má vždy dva atributy. Controller říká, jaký kontroler se v daném případě má použít a proměnná templateUrl udává, jaký soubor obsahuje šablonu, která se bude vykreslovat uvnitř HTML stránky.

4.3.3 Kontrolery

O předávání dat do šablon se starají kontrolery, které jsou dvojího typu. Do první skupiny patří takové, které jsou nezbytné pro běh jednotlivých součástí aplikace, jako například přihlášení uživatele, vytvoření projektu nebo výpis všech úkolů v systému. Ty jsou umístěny v souboru public/js/controllers.js. Druhou skupinou kontrolerů jsou takové, které se týkají konkrétních typů úloh. Těm se budu věnovat podrobně později.

Všechny kontrolery mají za úkol převážně získávat data odesíláním požadavků na API serverové části aplikace. Některé z nich potom samozřejmě odesílají informace na server, popřípadě provádějí validaci údajů. Jedna šablona může mít více kontrolerů, to je využito tak, že existuje jeden hlavní s názvem MainController a ten obsahuje funkce, které jsou využívané na všech stránkách. Jedná se například o zjištění uživatelských rolí. Stejně tak je možné, aby jeden kontroler byl použit více šablonami. Pro jednu entitu, se kterou se pracuje, tedy existují většinou dva kontrolery. Jeden do šablony předává seznam záznamů, například všechny projekty daného uživatele, a druhý, který pracuje s jednotlivými entitami, například jeden konkrétní projekt.

4.3.4 Služby

Služby představují funkce, které jsou přístupné a znovu použitelné všemi kontrolery. Hodí se tedy v případech, kdy je potřeba nějakou akci opakovat na více místech, ať už se jedná o získávání dat přes API nebo jiné funkce. V aplikaci je definována služba TaskTypeService, která má na starost správu typů úloh na klientské straně. Kromě toho, že obsahuje celou specifikaci úkolů, nabízí metody pro získání všech typů úloh nebo pouze jednoho s konkrétním identifikátorem.

4.3.5 Úkoly

Úkoly jsou definovány jako služba, kterou potom mohou využívat controllery. Jeden úkol je definovaný jako objekt a musí mít nastaveny tyto atributy

id Jedná se o jednoznačný identifikátor typu úkolu.

name Název typu úkolu.

description Popis typu úkolu, co je cílem a podobně.

backendTaskType Identifikátor serverového druhu úkolu, na který se tento typ mapuje.

dataType Typ dat, se kterými se v úkolu pracuje.

template Cesta k šabloně, která bude použita při vykreslení úkolů tohoto typu pro přispěvatele.

adminTemplate Cesta k šabloně, která bude sloužit pro zobrazení ohodnocení dat v úkolu tohoto typu.

export Pole formátů souborů, které je možné exportovat.

options Tento atribut má smysl pouze u úkolů, kdy se vybírá jedna nebo více možností z nabídky, která je stejná pro všechny datové položky. Toto pole obsahuje právě jednotlivé nabízené volby k výběru.

parseContributions Funkce, která má za úkol zpracovat příspěvky k jednotlivým datům. U některých úkolů je vyžadováno, že se například budou slučovat stejná ohodnocení a podobně. Funkce získá pole příspěvků tak, jak jsou uloženy v serverové části a vrací pole, se kterým umí pracovat šablona pro výpis příspěvků v administrátorské sekci.

4.3.6 Vytvoření úkolu

Při vytvoření úkolu je potřeba provést několik kroků. Prvním z nich je vytvoření šablon. Jedna je použita pro přispěvatele a druhá pro správce úkolů. Jejich umístění v adresářové struktuře záleží čistě na programátorovi, protože nejsou hledány nijak automaticky, ale cesta k nim je zadána explicitně. Existující typy mají šablony ve složce templates/taskTypes, kde pro každý typ úkolu existuje adresář s právě dvěma soubory. V šabloně určené pro odesílání příspěvků uživateli je potřeba pomocí direktivy ng-controller specifikovat, jaký kontroler bude použit pro načítání dat. Dále je vytvořen HTML kód pro zobrazení datových položek a formulář, tlačítka nebo jiný způsob pro odeslání informací. Vhodné také je, aby šablona obsahovala instrukce, jakým způsobem by mělo dojít k správnému plnění úkolu. Kontroler musí obsahovat kód pro načítání datových položek a informací o úkolu ze serveru a také funkci, popřípadě více funkcí pro odesílání příspěvků. Umístění zdrojového kódu kontroleru je opět na zvážení programátora, protože po registraci do hlavní aplikace bude dostupný odkudkoli. Pokud se vývojář rozhodne uložit kód v novém souboru, je pak nutné jej načíst v hlavní šabloně, aby mohl být použit.

Šablona, která slouží pro zobrazení a vyhodnocení příspěvků, nemusí mít specifikovaný kontroler. Předpokládá se, že data budou získávána ze serveru stále stejným způsobem. Rozdíl bude ovšem ve vizualizaci výsledků. Například při výběru hodnot se budou stejné výsledky nejspíše spojovat a bude zobrazen počet, kolikrát byla daná možnost vybrána. Naopak pokud bude přispěvatel poskytovat textový popis, zřejmě bude každý příspěvek zobrazen zvlášť.

Posledním krokem vytvoření typu úkolu je přidání jeho záznamu do služby TaskTypeService. Je důležité zadat identifikátor, který ještě nebyl použit a také je potřeba se rozhodnout, jaký serverový typ úkolu použijeme. Kromě toho si můžeme vybrat, zda bude možné exportovat výsledky do souboru a popřípadě v jakém formátu. Jestliže bude vše správně nastaveno, je možné vytvářet úkoly nového druhu.

Kapitola 5

Experimenty

Tato kapitola obsahuje podsekce „Testování“ a „Případy použití“. V první z nich si ukážeme způsob testování, jakým byla ověřena funkčnost jednotlivých komponent a celého systému. Pomocí testů lze definovat požadované chování zdrojových kódů a díky tomu lze ověřit funkčnost jednotlivých součástí aplikace. K zápisu testovacího kódu byly použity nástroje Chai a Supertest tak, jak bylo uvedeno v kapitole Analýza nástrojů. Ke spouštění testů je použita aplikace Mocha, což je konzolová aplikace, a lze pomocí ní definovat širokou škálu testů. V druhé části se podíváme na několik případů, jak lze aplikaci využít. Zaměříme se na situace, které systém dokáže řešit na rozdíl od produktů uvedených v kapitole Analýza existujících řešení. Kromě níže popsaných příkladů lze aplikaci samozřejmě využít mnoha jinými způsoby. Navíc je možné díky navržené struktuře zdrojových kódů snadno přidat úkoly nové.

5.1 Testování

5.1.1 Testování schémat

Schémata jsou hlavním stavebním kamenem aplikace, pracuje se s nimi všude v serverové části aplikace a zajišťují komunikaci systému s databází. Z tohoto důvodu je zcela nezbytné, aby byly implementovány pokud možno bez chyb, a proto je potřeba vytvořit pro ně sadu testů. Některé testy se týkají funkcí, které nepotřebují komunikovat s úložištěm, naopak schémata obsahují také metody, které se mají zavolat před uložením nebo načtením dat. Testování proběhlo pouze u schémat User, Task a Data, protože Project a Contribution neobsahují žádné speciální funkce. Seznam všech testů schémat je následující:

UserSchema

- Funkce save()
  - Před uložením do databáze by se zadaný email měl převést na malá písmena a heslo by mělo být zahešované.

  - Jesltliže existuje v databázi uživatel se zadanou emailovou adresou, nový uživatel se do databáze neuloží.
- Funkce addRole()
  - Při vytvoření uživatele musí být seznam rolí prázdný, po zavolání funkce addRole() musí mít uživatel právě jednu roli.
- Funkce hasRole()
  - Při volání funkce hasRole() musí být výsledek false, pokud danou roli uživatel nemá, a true, pokud uživatel danou roli má.

TaskSchema

- Funkce addOption()
  - Při vytvoření je seznam možností prázdný, po přidání jedné možnosti, musí pole options obsahovat přesně jednu položku.
- Funkce addCompletedData()
  - Po zavolání této funkce se musí počet dokončených datových položek zvýšit o jedna. Jestliže počet dokončených položek je stejný jako počet všech položek, stav úkolu se změní na finished.

DataSchema

- Funkce existsFile()
  - Funkce musí vrátit false v případě, že v systému neexistuje datová položka se zadanou cestou k souboru, v opačném případě musí vrátit true.
- Funkce addOption()
  - Při vytvoření je seznam možností prázdný, po přidání jedné možnosti, musí pole options obsahovat přesně jednu položku.
- Funkce addFragment()
  - Při vytvoření je seznam částí dat prázdný, po přidání jednoho záznamu obsahuje přesně jednu položku.
  - Do seznamu fragmentů nelze vložit dva stejné záznamy, to znamená takové, které mají stejnou pozici v textu a délku.
- Funkce save()
  - Jestliže je typ úkolu roven 0 (výběr z několika možností, které jsou stejné pro všechny datové položky), před uložením jsou datové položce přiděleny stejné možnost, jaké jsou definované u úkolu.
- Funkce addContribution()

  - Při přidání nového příspěvku se počet příspěvků u datové položky zvýší o jedna.
  - Jestliže je u všech datových položek dosaženo požadovaného počtu příspěvků, stav úkolu se změní na finished.

5.1.2 Testování API

Cílem tohoto testování je zjistit, zda serverová část aplikace pracuje správně a vykonává to, co se od ní očekává. Znamená to tedy, jestli dochází ke správnému ukládání a načítání dat. Mimo to se tyto testy zaměřují na to, jestli API rozhraní odpovídá principům REST, to znamená, jestli se v konkrétních situacích vrací správné kódy, hlavička a data odpovědí.

Testy jsou rozděleny na dvě skupiny podle toho, jací uživatelé mohou k adresám přistupovat. Soubory začínající admin testují chování aplikace z pohledu správce projektů a úkolů. V těchto testech se vždy zkoumá načtení entit daného uživatele, vytvoření nové entity, dotaz na existující a neexistující záznam, modifikace dat a odstranění informací z databáze. Entitami jsou myšleny projekty, úkoly a data. Dále bude detailně popsáno pouze testování API pro práci s projekty, protože práce s úkoly a datovými položkami je velice obdobná. Co se týče manipulace s příspěvky z pohledu správce, je možné pouze prohlížet údaje k dané datové položce. Proto ani tento soubor nebude detailně rozebrán. Podrobně si ukážeme soubor admin-export-test.js, který testuje exportování dat a příspěvků do souborů.

Druhá skupina testů simuluje práci přispěvatelů. Opět je u všech entit testovací kód velice podobný, tentokrát ještě jednodušší než u správce úloh, protože přispěvatel může data ze serveru pouze číst. Výjimkou je práce s příspěvky, kde je možné pouze vytváření nových dat. Konkrétní testy tedy nebudou detailně popsány.

Posledními dvěma testy jsou takové, které nezapadají ani do jedné výše uvedené skupiny. Buď je možné vykonávat dané operace jak správcům, tak přispěvatelům, nebo dokonce nepřihlášeným uživatelům. Tyto testy jsou authentication.js a user.js a u obou si ukážeme, jaké konkrétní funkce se testují.

5.1.2.1 Vybrané konkrétní testy

admin-project-test.js Testy se týkají práce s projekty z pohledu jejich správce.

- Jestliže se uživatel dotazuje na všechny svoje projekty, kód odpovědi by měl být 200 a data by měla být ve tvaru pole. Pokud žádné projekty neexistují, pole by mělo být prázdné.
- Při vytvoření projektu by měl být kód odpovědi 201 a měla by být nastavena hlavička Location.
- Při přístupu na detail existujícího projektu by měl být kód odpovědi 200. V těle by měly být informace o daném projektu.

- Při přístupu na detail neexistujícího projektu by měl být kód odpovědi 404.
- Po dotazu na úpravu objektu by měl být kód 200 a data by měla obsahovat nové informace o projektu.
- Po úspěšném smazání projektu by měl být kód odpovědi roven 204.

admin-export-test.js Tyto testy se týkají exportu dat v různých formátech.

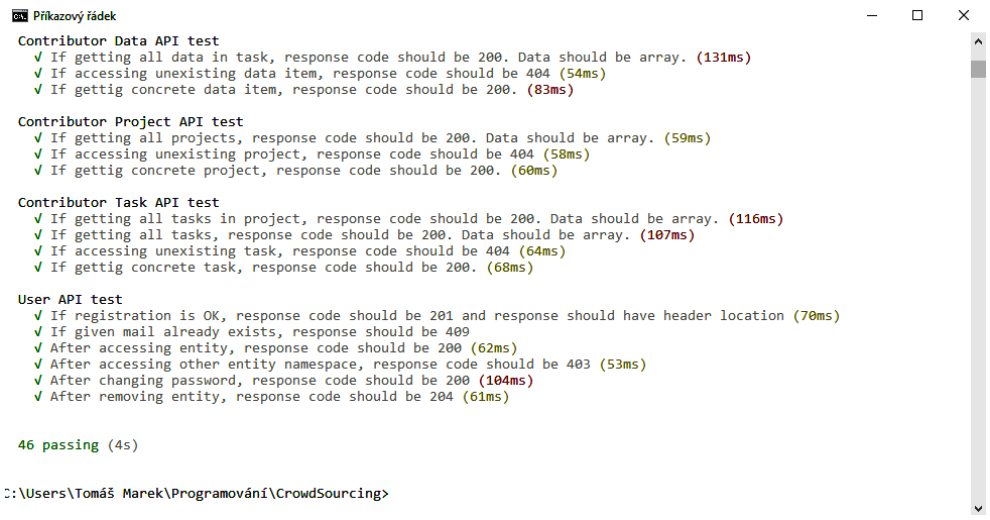
- Jestliže přijde požadavek s hlavičkou Accept s hodnotou text/csv, odpověď by měla mít kód 200 a hlavičku Content-Type nastavenou na text/csv; charset=utf-8.
- Jestliže přijde požadavek s hlavičkou Accept s hodnotou text/turtle, odpověď by měla mít kód 200 a hlavičku Content-Type nastavenou na text/turtle; charset=utf-8.
- Jestliže na API adresu pro export dorazí požadavek, který má hlavičku Accept nastavenou na jinou hodnotu, měla by se vrátit odpověď s kódem 415.

authentication.js Tento test se týká ověřování identity uživatelů.

- Jestliže při přístupu do zabezpečené části webu chybí hlavička Authorization, odpověď by měla mít kód 401.
- Jestliže při přístupu do zabezpečené části aplikace je uvedena autorizační hlavička, ale údaj v ní je nesprávný, kód odpovědi by měl být 401.
- Jestliže je autorizační hlavička v pořádku, ale uživatel nemá potřebná oprávnění, odpověď by měla mít kód 403.
- Jestliže je autorizační hlavička v pořádku, kód odpovědi by měl být 200.
- Jestliže je při přístupu na adresu /api/authenticate zadaný špatný email nebo heslo, odpověď by měla být 401.
- Jestliže je při přístupu na adresu /api/authenticate zadaný správný email i heslo, odpověď by měla být 200.

user.js Tyto testy se týkají API pro práci se záznamy uživatelů.

- Jestliže při registraci je zadán již evidovaný email, kód odpovědi by měl být 409.
- Jestliže registrace proběhne v pořádku, kód odpovědi by měl být 201 a hlavička Location by měla obsahovat URI nově vytvořeného uživatele.
- Při přístupu na detail neexistujícího uživatele (neexistující ID) by měl být kód odpovědi 404.



Obrázek 5.1: Ukázka výsledku testování API pomocí nástroje Mocha

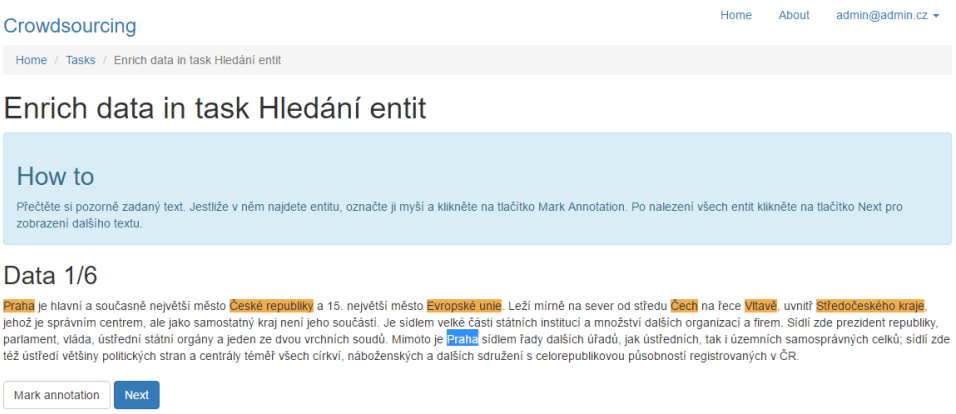
- Při přístupu na detail existujícího uživatele (existující ID) by měl být kód odpovědi 200.
- Při úspěšné změně hesla by odpověď měla mít kód 200.
- Při úspěšném zrušení účtu uživatele by měl být kód odpovědi 204.

5.2 Příklady použití

5.2.1 Hledání entit v textu

První úloha se týká textů a dalo by se říct, že se jedná o jeden ze základních úkolů v text miningu. Jedná se o hledání entit, to znamená názvů států, měst, osob, společností, značek a podobně. Tento úkol by se dal částečně algoritmizovat pomocí učících se algoritmů. Například by program měl seznam výrazů, podle kterých lze jednotlivé entity identifikovat. Bohužel tato databáze by byla velice rozsáhlá a ne vždy by zcela zafungovala. Další možností, jak tento úkol automatizovat, je udržovat seznam samotných jmen a názvů. Takový algoritmus by v textech ovšem nacházel již známé entity a nikoli nové. Z tohoto důvodu byl implementován tento úkol do nové aplikace a s řešením tak mohou pomoci různí uživatelé.

Hlavním principem by bylo zobrazení textu uživateli, ten by si jej přečetl a vybral všechny entity, na které narazí. Bylo by nevhodné po přispěvateli chtít, aby entity vypisoval, jednak jich může být mnoho a práce by byla hodně zdlouhavá a za druhé by mohly vznikat překlepy, které by pak výsledky znehodnocovaly. Navíc by bylo dobré udržovat informace o tom, na jaké pozici v textu se entita nachází a jaká je délka vybrané fráze. To je důležité kvůli



Obrázek 5.2: Ukázka aplikace - hledání entit v textu

stejným slovům, která mají různý význam (například Most a most). Bylo by tedy dobré, kdyby uživatel označil entitu myší a kliknutím na tlačítko svůj příspěvek odeslal. Po nalezení všech požadovaných položek by mohl přejít na další datový záznam. Ačkoli by se mohlo na první pohled zdát, že takový úkol není možné v aplikaci realizovat, je implementace poměrně jednoduchá. Dá se říci, že je to prakticky hledání klíčových slov v textu, ovšem hodnota nebude zadána ručně a uložíme další informace, nejenom obsah slova. Z tohoto důvodu je na serverové straně použit typ úkolu číslo tři, tedy zadávání klíčových slov. Hodnota atributu value uložená v databázi u schématu contribution pak bude mít textovou podobu. Bude obsahovat nejdříve číslo označující pozici entity v textu, následně délku výrazu a nakonec samotný obsah. Tyto hodnoty pak budou odděleny čárkou.

Největší nástrahou je již zmíněné označování dat, kdy na základě výběru části textu myší je třeba získat pozici a délku selekce v daném elementu. JavaScript naštěstí má funkce, které s označeným textem dokáží pracovat, dokonce i když je formátován pomocí HTML značek. API funkce pro přidání nového ohodnocení je v tomto případě volána pro každý vybraný úsek textu a nejsou všechny příspěvky zpracovány zároveň. Naopak při přechodu na další dokument se žádná data na server neposílají, pouze se získá obsah další textové položky. V administrátorské části je možné sledovat již přijaté příspěvky. V tomto případě jsou jednotlivé vybrané výrazy seskupovány a je k nim připsán počet, kolikrát byly vybrány. Lze to samozřejmě udělat i jinak, záleží čistě na autorovi daného typu úkolu. Data je možné exportovat ve formátu CSV i NIF.

5.2.2 Přidání dalších informací k entitám

Na tomto příkladě si ukážeme přidávání příspěvků k částem textu. V návaznosti na předchozí úkol si můžeme představit, že jsme pomocí přispěvatelů extrahovali z textu entity a nyní bychom k nim chtěli získat další informace. Například potřebujeme zjistit, o jaký typ objektu se jedná, to znamená osoba, město, společnost a podobně. Takové údaje bychom mohli získávat už v rámci prvního příkladu. Po označení části textu by uživatel vybral z nabídky druh entity a všechna tato data by se poslala na server. Opět by v nastavení úlohy bylo potřeba nějakým způsobem spojit všechny informace, aby se daly uložit v textové podobě.

V některých případech ovšem máme části textu a k nim potřebujeme získat další údaje dodatečně. Může jít o případy, kdy o entitách chceme vědět pouze to, zda byly vybrány správně nebo ne. Při ručním ohodnocení dat se může díky lidské chybě snadno stát, že označené slovo entitou není. To samé může nastat při zpracování dat automatizovaně, kdy u algoritmu bude vždy existovat určitá pravděpodobnost chyby. Algoritmus navíc může nějakou entitu celkem snadno opomenout, naopak při ručním ohodnocení většinou žádáme o pomoc více uživatelů, to znamená, že se riziko vynechání nějaké entity snižuje.

Ať už budeme chtít o entitách získávat jakékoli informace, je zřejmé, že potřebujeme znát celý text, ve kterém se vyskytují. Příkladem opět mohou být homonyma most a Most. Na rozdíl od jiných existujících řešení tato možnost v naší aplikaci existuje. Při uploadu dat má uživatel na výběr dvě možnosti. První z nich je formát NIF, kde jsou definované texty i jejich části v jednom souboru. Druhý způsob je nahrát klasicky dokumenty a poté přidat části textů pomocí souboru ve formátu CSV. Při přidávání hodnocení pak přispěvatel vyplňuje u každé datové položky formulář, konkrétně vybírá jednu možnost z několika pomocí zaškrtnutí políčka. Odeslání probíhá vždy pro jednu datovou položku a pro všechny části textu zároveň. Ze způsobu hodnocení vyplývá také typ úkolu na serverové straně, jelikož se u všech položek vybírá právě jedna možnost ze seznamu, který je pro všechny fragmenty stejný. Jednotlivé možnosti jsou nastaveny v definici typu úkolu na klientské straně. Při zobrazení ohodnocení jsou odpovědi pro jednotlivé fragmenty opět shlukovány. Export je v tomto případě možný pouze ve formátu CSV, jelikož slovník NIF ve své specifikaci neobsahuje potřebné entity a vlastnosti.

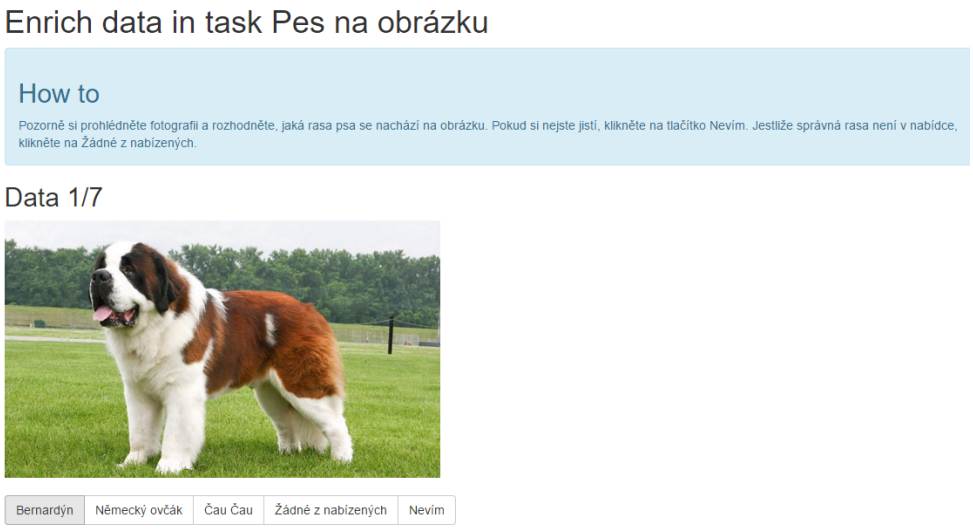
5.2.3 Potvrzení nalezených objektů v obrázku

Následující příklad by měl ukázat, jak lze pracovat s obrázky a také jak se může hodit typ úkolu, kdy uživatel u každé datové položky vybírá jednu možnost a nabídka možností je u každé položky jiná. Představme si, že máme algoritmus, který dokáže detekovat rasu psa na obrázku. Taková aplikace by zřejmě zkoumala tělesné proporce, délku ocasu a nohou, barvu srsti a další vlastnosti. Nicméně se dá předpokládat, že ani ten sebelepší algoritmus by

Obrázek 5.3: Ukázka aplikace - kontrola správnosti entit

nedokázal rasu určit zcela stoprocentně. Proto by ke každé fotografii vybral několik ras, které jsou nejpravděpodobnější. U nějakých fotografií by to byly pouze dvě možnosti, u jiných by mohla být nabídka rozsáhlejší. Pro vybrání co nejpřesnější možnosti lze využít crowdsourcingovou aplikaci, kde budou uživatelé o správné rase na obrázku „hlasovat“.

Jelikož se jedná o obrázky, je potřeba tomu přizpůsobit šablony v části aplikace pro správce úlohy i pro přispěvatele. V nastavení úkolu je samozřejmě změněn typ dat a na serveru je použit druh úkolu číslo dvě. Při vytváření úkolu musí uživatel nejdříve nahrát obrázkové soubory a následně je nutné přidat k nim odpovídající položky. Aby mohly volby obsahovat čárky, je každá položka na novém řádku. K jaké fotografii náleží, je pak dáno názvem souboru, který musí být v úloze vždy jedinečný. Nijak se nekontroluje počet možností u jednotlivých dat, to znamená, že jich může být veliké množství, ale také nějaké záznamy nemusí nabízet žádné položky. Je tedy třeba dát pozor, aby nedocházelo k nějakým chybám nebo omylům. Další nástrahou je situace, kdy ani jeden návrh neodpovídá skutečnosti. To znamená, že algoritmus nedokázal navrhnout rasu psa, která na obrázku skutečně je. Existují dvě možností řešení. První z nich představuje přidání položky typu „Žádné z nabízených“ do souboru ke všem datům. To je samozřejmě pro uživatele pracné a nevýhodné. Druhou možností, která byla v aplikaci realizována, je, že se tato dodatečná položka bude přidávat automaticky v rámci nastavení typu úkolu. To znamená, že ve formulářích pro odeslání ohodnocení bude automaticky existovat jedna možnost navíc. S tímto chováním aplikace musí uživatel počítat, a proto je dobré zmínit tuto vlastnost v popisu úkolu. Úplně stejná situace může nastat v případech, kdy si uživatel není zcela jistý. Je diskutabilní, zda je lepší



Obrázek 5.4: Ukázka aplikace - výběr objektu na obrázku

po uživateli chtít, aby správnou hodnotu tipl, nebo aby zvolil možnost, že si není jistý. Opět byla vybrána druhá možnost, to znamená, že k formulářům bude přidáno další tlačítko, které umožní uživateli přejít na další obrázek, aniž by odeslal ohodnocení. Samozřejmě by se mohla i tato možnost odesílat na server a ukládat. Tím by správce úkolu získal zpětnou vazbu o tom, že na obrázku se vyskytuje rasa psa, která zřejmě není příliš obvyklá. Mohlo by se ovšem stát, že by k jedné fotografii byly všechny příspěvku typu „Nevím“, a ačkoli bychom nezískali žádné konkrétní ohodnocení, cíl vyplnění otázky by byl dosažen.

V tomto případě, kdy uživatelé mají zvolit pouze jednu možnost, je odesílání ohodnocení realizováno pomocí tlačítek. Po kliknutí se data odešlou a není možné tuto volbu měnit. Samozřejmě by bylo možné posílat údaje pomocí formuláře s radio buttony nebo check boxy v případech, kdy by na fotografiích bylo více psů. Jestliže ani jedna možnost z výběru neodpovídá realitě, je po kliknutí na tlačítko „Žádné z nabízených“ odeslána hodnota null. Jestliže si uživatel není jistý, klikne na tlačítko „Nevím“, kdy se žádná data na server neodesílají a uživatel pokračuje k dalším obrázkům. Při zobrazení výsledků jsou jednotlivé hodnoty opět slučovány a je zobrazen počet, kolikrát byla položka vybrána.

5.2.4 Nalezení refrénu v hudební skladbě

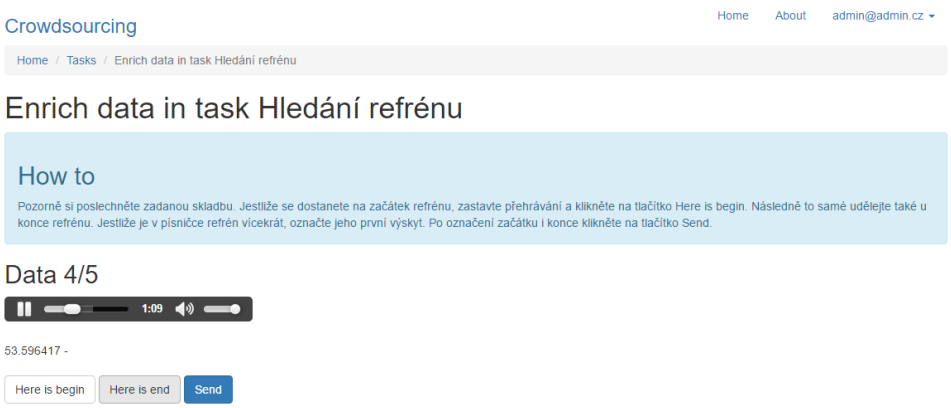
Posledním případem použití, který si ukážeme, je práce s audio soubory, která by šla ovšem provádět také s videi. Většina konkurenčních aplikací nabízí víceméně pouze překlad, přepis nebo popis obsahu. V aplikaci je definován

úkol, který slouží k označení částí v těchto typech médií. Může se jednat například o označení reklamy, nevhodného obsahu pro děti a mládež nebo označení refrénu v písničce. Poslední zmíněná možnost je implementována v systému. Právě refrén je klíčovou částí drtivé většiny hudebních skladeb a na jeho základě lze písničku popsat nebo identifikovat. To se pak hodí například k rozpoznání hudebního žánru nebo k hledání plagiátů. Problém s refrénem je, že ačkoli se dají definovat jeho vlastnosti, které lze najít ve většině případů, zdaleka neplatí ve všech skladbách. Tato část skladby se mnohdy vyznačuje tím, že se ve skladbě vyskytuje vícekrát, má vždy stejnou melodii a slova a od ostatních částí skladby se liší. Ovšem jak již bylo zmíněno, ne vždy jsou tyto rysy zachovány, naopak jsou často porušovány a označit část skladby jako refrén je značně obtížné. Pro počítače je to pak zcela nesplnitelný úkol, a proto je dobré požádat o pomoc uživatele internetu.

Ve své podstatě se jedná o hledání klíčových slov, ta ovšem zase nejsou definovaná přímo uživatelem, ale jsou automaticky získaná po kliknutí na tlačítko. Přesněji se tedy jedná o dvě tlačítka, kdy prvním označíme začátek a druhým konec části skladby. V typické skladbě se refrén vyskytuje zpravidla vícekrát, po uživateli ovšem budeme žádat označení pouze prvního výskytu. Pokud totiž budeme mít jednu pozici v písničce, automatizovaně by pak neměl být problém najít i další výskyty na základě podobnosti jednotlivých úseků.

Část pro poskytování příspěvků tedy zobrazuje HTML5 audio přehrávač, kde si uživatel samozřejmě může danou skladbu pustit. Jestliže se při přehrávání dostane na začátek refrénu, pozastaví skladbu a klikne na tlačítko „Zde je začátek“. Před kliknutím může pozici přehrávání upravit, aby byla co nejpřesnější. Stejně funguje zaznamenání konce refrénu. Po kliknutí na tlačítko „Odeslat“ je záznam uložen v databázi a uživatel může pokračovat další skladbou. Opět je zde možnost zvolit „Nevím“ pro případy, kdy je určení refrénu obtížné a uživatel si není jistý.

Administrátor nahrává skladby ze svého počítače, není možné hodnotit skladby, které jsou umístěny jinde na internetu. Tato možnost v aplikaci sice také existuje, ale uživatel by musel úkol vytvořit jako textový a obsah souborů by pak obsahovat HTML kód pro vložení skladeb umístěných jinde na webu. Při zobrazení příspěvků má pak správce možnost vidět nejenom vybrané časy začátků a konců, ale může také jednotlivé části přehrát pomocí tlačítka. Export je možný pouze do souboru ve formátu CSV, protože NIF se užívá výhradně pro textová data. Jeden řádek ve výstupním souboru bude vždy obsahovat nejdříve název skladby, které se hodnocení týká, a následně čas začátku a konce refrénu.



Obrázek 5.5: Ukázka aplikace - výběr části audio záznamu

Závěr

Cílem práce bylo prozkoumat a vyhodnotit možnosti použití crowdsourcingu při získávání užitečných informací v případech, kdy tato data nelze získat automatizovaným způsobem. Tato metoda může být v mnoha situacích rovnocennou, nebo dokonce výhodnější alternativou ke klasickému zaměstnání lidí. Největšími přínosy jsou rychlost získání informací, objektivita a také může být tento způsob dělby práce levnější. Nevýhodou může být nekvalifikovanost přispěvatelů nebo nutnost propagovat své úlohy, aby se dostaly k potřebnému počtu uživatelů. Je ovšem zřejmé, že crowdsourcing v mnoha případech představuje efektivní řešení, a také proto existují různé nástroje pro zadávání mikroúloh a jejich vyplňování. Všechny tyto aplikace mají své výhody a nevýhody a na základě toho bylo jedním z cílů navrhnout a implementovat vlastní crowdsourcingový systém. Aplikace byla úspěšně naprogramována a otestována. Stejně jako u jiný nástrojů i zde najdeme různé plusy a mínusy. Velikou výhodu je to, že je aplikace snadno rozšiřitelná. Existuje široká škála jejího použití od běžných úkolů až po velice specifické. Některé z těch méně častých, které se ovšem mohou hodit, jsme si ukázali v závěru práce. Další pozitiva jsou přehledné uživatelské rozhraní, jednoduchost použití a možnost pracovat s daty přes API. Mezi nevýhodami stojí za zmínku především to, že pro použití aplikace veřejně je třeba zřídit hosting, který pro Node.js není ještě tak běžný. Dalším faktem je, že pro rozšíření aplikace a přidání nových typů úloh je třeba alespoň základní znalost programování v JavaScriptu. I přes tyto nedostatky se ovšem podařilo vytvořit plně funkční a využitelný systém, který by se dal využít v mnoha různých případech.

Budoucí práce

Při testování aplikace byly odhaleny některé vlastnosti systému, které by bylo možné změnit nebo vylepšit. První z nich je například rozdělení odpovědnosti za definici úkolu mezi zadavatele a autora úlohy. Jedná se například o zadávání možností v případech, kdy má uživatel vybrat jednu nebo více z nabízených po-

ložek. V aktuálním stavu tyto údaje specifikuje autor typu úkolu, to znamená, že úlohy jsou velice specifické a konkrétní. Bylo by možná lepší, kdyby zrovna tyto informace zadával uživatel, který potřebuje data obohatit. Další prostor pro vylepšení je jednoznačně ve vizualizaci výsledků. Určitě by se hodilo zobrazit získané informace v různých tabulkách či grafech. Prostor pro zlepšení je určitě také v exportu dat, kdy současná podpora jednoho formátu, v některých případech dvou, pro mnoho uživatelů nemusí být dostatečná. Oproti analyzovaným systémům naše aplikace nenabízí možnost importu dat z jiných zdrojů, například aplikací Facebook, Twitter, YouTube a podobně, což může některé uživatele od použití výsledného systému odradit.

Literatura

[1] CZ.NIC: Crowdsourcing. [online], 2012-2014 [cit. 2016-02-12]. Dostupné z: http://www.jaknainternet.cz/page/1665/crowdsourcing/

[2] Bratvold, D.: What is Crowdsourcing? [online], 2014 [cit. 2016-02-12]. Dostupné z: http://dailycrowdsource.com/training/crowdsourcing/what-is-crowdsourcing

[3] Crowdsourcing. [online], 2001-2016 [cit. 2016-02-12]. Dostupné z: https://en.wikipedia.org/wiki/Crowdsourcing

[4] CrowdFlower: CrowdFlower. [online], 2015 [cit. 2016-02-15]. Dostupné z: http://www.crowdflower.com

[5] Scifabric: The ultimate crowdsourcing framework - PyBossa. [online], 2011 [cit. 2016-02-15]. Dostupné z: http://pybossa.com/

[6] Scifabric: Crowdcrafting. [online], 2011 [cit. 2016-02-15]. Dostupné z: http://crowdcrafting.org/

[7] CrowdSource: CrowdSource.com - Managed Crowdsourcing Solutions. [online], 2016 [cit. 2016-02-18]. Dostupné z: http://www.crowdsource.com/

[8] OneSpace.com: OneSpace. [online], 2016 [cit. 2016-02-18]. Dostupné z: http://www.onespace.com/

[9] Brunelli, L.: Where to Find Real Micro Jobs. [online], 2016 [cit. 2016-02-18]. Dostupné z: http://workathomemoms.about.com/od/Micro-Jobs-Crowdsourcing/ss/Micro-Jobs.htm

[10] Fitchard, K.: 8 Crowdsourcing apps (besides OpenSignal) we love. [online], 2015 [cit. 2016-02-18]. Dostupné z: https://opensignal.com/blog/2015/07/09/8-crowdsourcing-apps-besides-opensignals-love/

[11] Maher, J.: Hive. [online], 2014 [cit. 2016-02-18]. Dostupné z: https://github.com/nytlabs/hive

[12] RestApiTutorial.com: Learn REST: A RESTful Tutorial. [online], 2015 [cit. 2016-01-20]. Dostupné z: http://www.restapitutorial.com/

[13] Node.js Foundation: Node.js. [online], 2016 [cit. 2016-03-11]. Dostupné z: https://nodejs.org/en/

[14] Mrozek, J., Nešetřil, J.: Node.js - s JavaScriptem na serveru. [online], 2012 [cit. 2016-03-01]. Dostupné z: https://www.zdrojak.cz/serialy/node-js-s-javascriptem-na-serveru/

[15] Sevilleja, C.: Easy Node Authentication: Setup and Local. [online], 2013 [cit. 2016-03-01]. Dostupné z: https://scotch.io/tutorials/easy-node-authentication-setup-and-local

[16] Manricks, G.: Testing in Node.js. [online], 2014 [cit. 2016-03-01]. Dostupné z: http://code.tutsplus.com/tutorials/testing-in-nodejs--net-35018

[17] MongoDB: MongoDB for GIANT Ideas. [online], 2016 [cit. 2016-03-02]. Dostupné z: https://www.mongodb.org/

[18] Node.js Foundation: Express - Node.js web application framework. [online], 2016 [cit. 2016-03-02]. Dostupné z: http://expressjs.com/

[19] LearnBoost: Mongoose. [online], 2011 [cit. 2016-03-02]. Dostupné z: http://mongoosejs.com/

[20] Luer, J.: Chai Assertion Library. [online], 2011-2015 [cit. 2016-03-07]. Dostupné z: http://chaijs.com/

[21] Holowaychuk, T.: Mocha - the fun, simple, flexible JavaScript test framework. [online], 2011 [cit. 2016-03-07]. Dostupné z: https://mochajs.org/

[22] RDF Working Group: Resource Description Framework (RDF). [online], 2014 [cit. 2016-03-08]. Dostupné z: https://www.w3.org/RDF/

[23] Hellmann, S.: NLP Interchange Format (NIF) 2.0 - Overview and Documentation. [online], 2013 [cit. 2016-03-08]. Dostupné z: http://persistence.uni-leipzig.org/nlp2rdf/

[24] Hellmann, S., Brümmer, M., Ackermann, M.: NLP Interchange Format. [online], 2015 [cit. 2016-03-08]. Dostupné z: http://nif.readthedocs.org/en/2.1-rc/index.html

[25] Google: AngularJS - Superheroic JavaScript MVW Framework. [online], 2010-2016 [cit. 2016-03-10]. Dostupné z: `https://angularjs.org/`

[26] Otto, M., Thornton, J.: Bootstrap. [online], 2010-2016 [cit. 2016-03-13]. Dostupné z: `http://getbootstrap.com/`

Příloha **A**

Seznam použitých zkratek

API Application Programming Interface

CSS Cascading Style Sheet

CSV Comma-Separated Values

DI Dependency Injection

HTML Hyper Text Markup Language

JSON JavaScript Object Notation

MVC Model-View-Controller

NIF NLP Interchange Format

NLP Natural Language Processing

ORM Object-Relational Mapping

PHP PHP: Hypertext Preprocessor

RDF Resource Description Framework

REST Representational State Transfer

TSV Tab-Separated values

UI User Interface

URI Uniform Resource Identifier

URL Uniform Resource Locator

Příloha B

Návod k instalaci

Tato kapitola ukazuje, jakým způsobem je možné snadno nainstalovat a spustit výslednou aplikaci. Návod je určen především pro zprovoznění na vlastním počítači nebo serveru. Při použití hostingu třetích stran je nutné postupovat podle pokynů poskytovatele, nastavení aplikace se ovšem nijak neliší. Instalace je popsána pro operační systémy Windows, u jiných platforem by byl postup obdobný.

B.1 Stažení a instalace potřebných nástrojů

Na oficiálních stránkách[4] je nutné nejdříve stáhnout Node.js, který lze snadno nainstalovat pomocí průvodce. To stejné je potřeba udělat s databází MongoDB[5]. Podle pokynů je poté nutné vytvořit složku, kam se budou data ukládat. Pro systémy Windows je to C:/data/db.

B.2 Nakopírování souborů a stažení balíčků

Zdrojové kódy aplikace je nutné nakopírovat na pevný disk počítače, popřípadě přenést na server přes internet. Stažení balíčků se provádí pomocí nástroje npm, který je součástí instalace Node.js. V příkazovém řádku je třeba přejít do adresáře, kam byly nakopírovány zdrojové soubory. Na tomto místě se musí spustit příkaz npm install, který provede instalaci. V adresáři node_modules by se měly objevit složky stažených modulů. Jestliže systém nezná příkaz npm, je potřeba zadat celou adresu umístění nástroje. Během této procedury je nezbytné mít počítač připojený k internetu.

[4] https://nodejs.org/en/
[5] https://www.mongodb.org/

B.3 Konfigurace aplikace

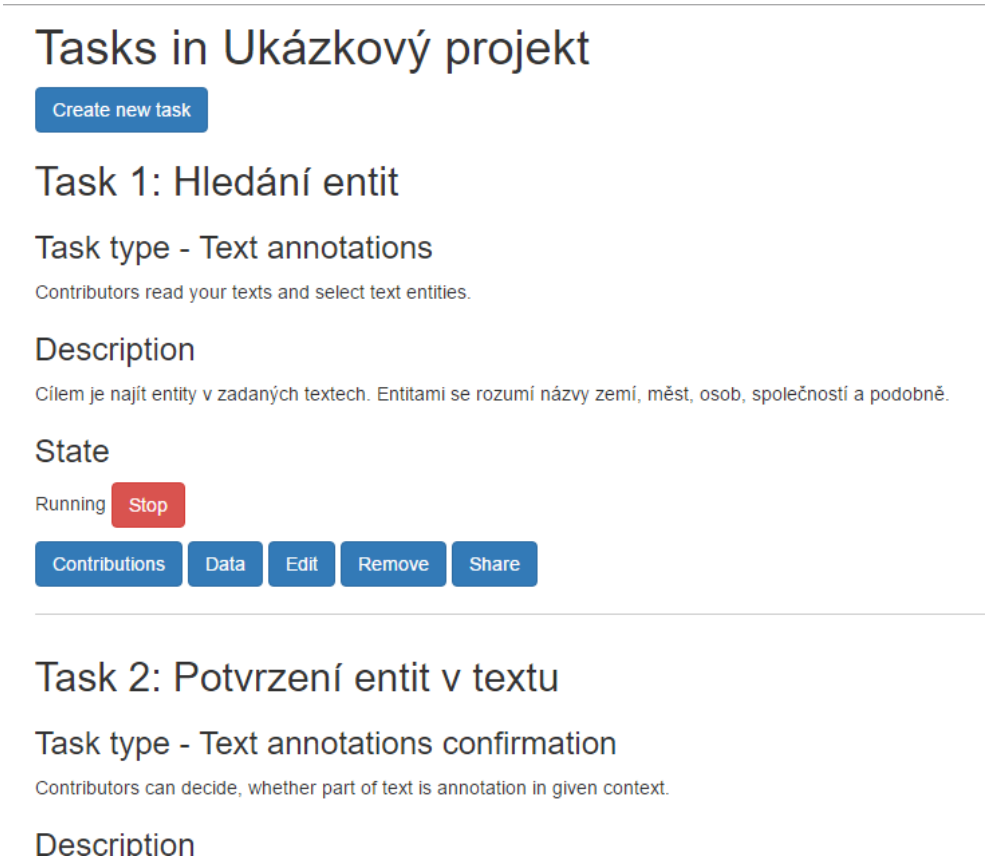
V souboru /app/config/config.js je nutné změnit hodnotu url na adresu databázového serveru. Tato adresa má tvar mongodb://server:port/database, kde hodnoty server, port a database je třeba uzpůsobit dané situaci. Při využití hostingu třetí strany uživatel obdrží potřebné údaje od poskytovatele. Jestliže má aplikace běžet na počítači, je třeba správné hodnoty zjistit. Ve výchozím nastavení má server hodnotu localhost a port je 27017. Název databáze je možné zvolit libovolný. Volitelně je možné nastavit také port, na kterém bude aplikace k dispozici. Ve výchozím stavu je hodnota portu 8080, ovšem lze ji změnit v souboru /main.js nastavením proměnné port na řádku 4.

B.4 Spuštění aplikace

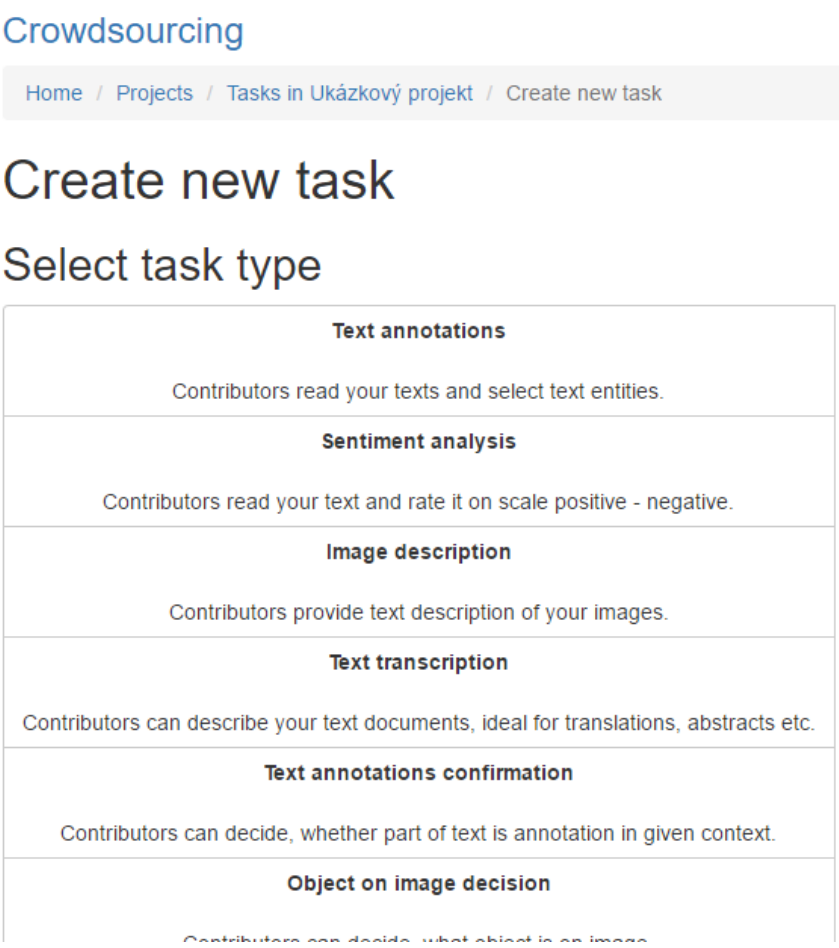
Nyní je nezbytné aplikaci spustit. Nejdříve je potřeba spustit databázi, čehož lze dosáhnout spuštěním programu mongod.exe v adresáři, kam byl nástroj MongoDB nainstalován. Konzolová aplikace by měla zobrazit informace pro připojení k databázi. Samotná aplikace se pak spustí v příkazovém řádku pomocí node.exe main.js, kde node.exe je název včetně cesty k binárním souborům Node.js, a main.js je cesta k hlavnímu souboru aplikace. Aby skutečně šlo systém spustit, je nutné mít k dispozici připojení k internetu.

Příloha C

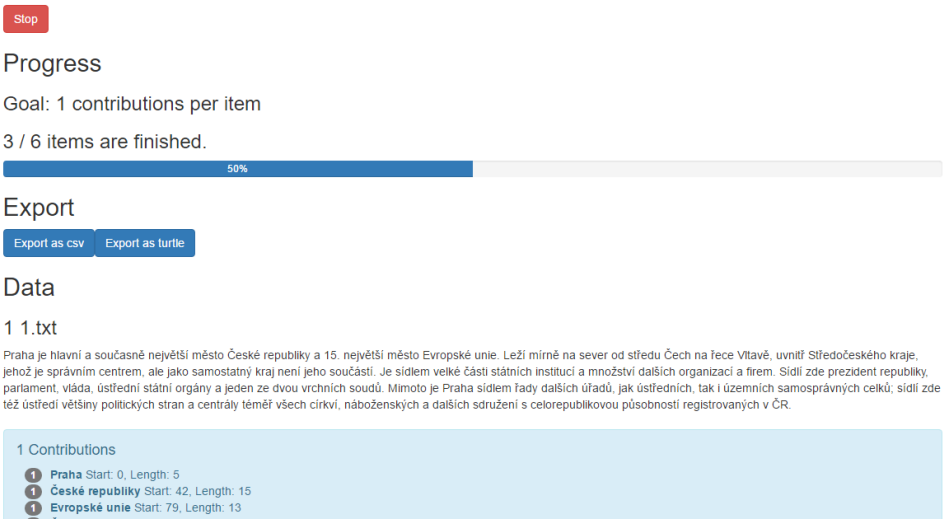
Grafická příloha



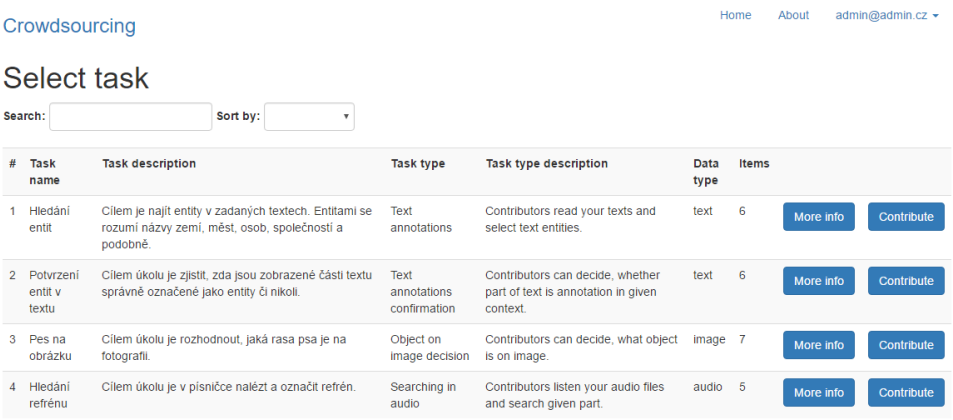
Obrázek C.1: Seznam úkolů v části aprávce



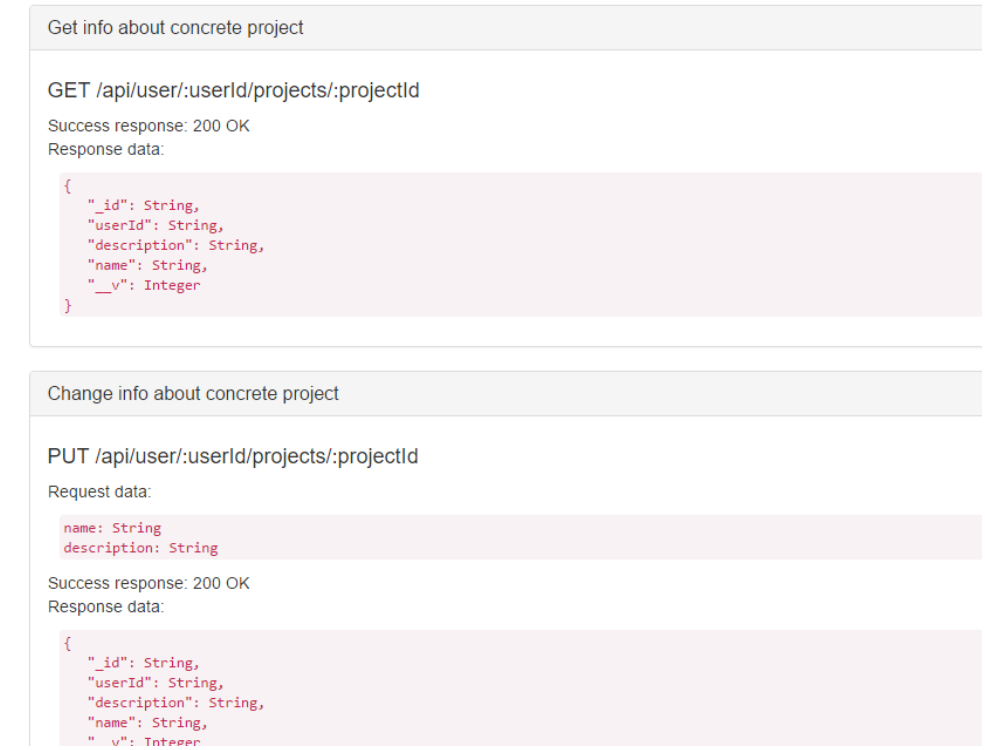
Obrázek C.2: Výběr typu úkolu při vytváření nové úlohy



Obrázek C.3: Stav vyplnění úkolu a seznam příspěvků k jednotlivým položkám



Obrázek C.4: Výběr úkolu, který může přispěvatel splnit



Obrázek C.5: API dokumentace aplikace

Příloha D

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF