Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Jakub Adam, Monika FIlipčiková, Andrej Švec, Filip Vozár

moderateIT

Dokumentácia k inžinierskemu dielu z predmetu Tímový projekt

Študijný program: Informačné systémy / Softvérové inžinierstvo
Vedúci tímu: Ing. Jakub Šimko, PhD.
Zimný semester šk. r. 2016 / 2017

Obsah

Dokumentácia k inžinierskému dielu

Big picture

Úvod

S pribúdajúcim počtom používateľov webu, rastie každodenne aj počet komentárov, v rôznych diskusiách. Novinové portály majú veľký záujem o to, aby sa na ich stránkach vo veľkom diskutovalo. Avšak čím ďalej, tým viac je web zaplavovaný negativitou, nenávistnými komentármi a tzv. “trollmi”, ktorí sa snažia diskusie znehodnocovať a napádať ostatných návštevníkov.

Portály sú za diskusie zodpovedné z právneho hľadiska a taktiež je v ich záujme, aby sa návštevníci cítili na ich stránkach príjemne. Často však nie je v silách moderátorov novinových portálov odstrániť všetky komentáre nepatriace do diskusie. Práve preto sme sa rozhodli, že vytvoríme nástroj, ktorý bude slúžiť, ako poradný hlas pre moderátorov. Jeho účelom je ušetriť im čas a zároveň urýchliť ich prácu.

Služba, ktorú sme vytvorili je založená na strojovom učení. Z komentára sú abstrahované črty opisujúce komentár, autora a článok. Tieto črty sú následne spracovávané pomocou rozhodovacích stromov. Hodnota, ktorú získame zodpovedá kvalite komentára a slúži nám na zoraďovanie komentárov podľa kvality. Moderátor sa tak môže zamerať primárne na toxické komentáre a nemusí čítať úplne všetko.

Globálne ciele pre ZS/LS

Nasadenie do SME.sk

Naším prvým a najdôležitejším cieľom, ktorý je zároveň aj trošku biznis typu, je nasadenie našej služby pre nášho partnera SME.sk. S týmto je spojené množstvo práce inžinierskeho typu. V prvom kroku sme so SME.sk dohodli, ako budeme na ich systém napojení.

API. Dohodli sme, že nám budú posielať do nášho API vystaveného na URL *https://api.moderateit.tech* jednotlivé články, autorov článkov a komentáre. My budeme komentáre analyzovať a výsledky si ukladať do databázy. Nebudeme ich SME.sk nijako vracať, budú si ich musieť sami vypýtať.

iframe. Našou úlohou je navrhnúť a vytvoriť iframe, ktorý si SME.sk vloží na svoju stránku a vďaka tomu ich moderátori uvidia vo svojom moderačnom rozhraní výsledky našich analýz. Do iframe umiestnime graficky pekne a prehľadne spracovanú informáciu o celkovej kvalite daného príspevku ohodnotenej našou službou a o reputácii autora, tiež ohodnotenej našou službou.

Podrobné štatistiky. Po kliknutí do iframe sa v novom okne otvoria podrobné štatistiky komentára, alebo autora. Tieto štatistiky už máme pripravené, akurát SME.sk ich potrebuje v novom okne a nie vy vyskakovacom okne, čiže ich bude treba trošku upraviť

Model. Pre hodnotenie príspevkov SME.sk je potrebné na ich dátach natrénovať aj nový model. Náš model beží v prostredí Microsoft Azure Machine Learning Studio. Trénovanie v praxi prestavuje zobrať nový dataset a použiť hyperparametre modelu, ktoré už máme prednastavené a pohrať sa s nimi, aby sme dosiahli čo najlepšie výsledky.

Ďalšie dáta. Od SME.sk sme dostali aj ďalšie dáta, konkrétne všetky dáta za rok 2016. Tieto dáta by sme chceli analyzovať a potom trénovať model znova. Získaním väčšieho množstva trénovacích dát by sme sa mohli pokúsiť natrénovať modely, ktoré vyžadujú viac dát na trénovanie, napr. nejaký hĺbkovo sa učiaci model.

Optimalizácie databázy. V projekte nepoužívame relačné databázy. Všetky perzistentné dáta sú uložené v databáze Couchbase. Avšak spôsob akým využívame Couchbase z neho môže časom spraviť úzke hrdlo systému. Navyše tento spôsob nie je v súlade s odporúčaným používaním Couchbase. Ide konkrétne o to, že často využívame views aj na vyhľadanie nejakých konkrétnych záznamov, čo by sme nemali. Views by sme mali používať len pri hľadaní množstva príspevkov, ktoré spĺňajú nejaké požiadavky.

Preto budeme potrebovať zmeniť spôsob, akým Couchbase využívame. Pôjde najmä o systém generovania takých identifikátorov pre jednotlivé uložené entity, že namiesto hľadania entít vo views ich budeme vedieť následne nájsť iba pomocou kľúča.

Autentifikácia. Pre nasadenie do produkcie je veľmi dôležitá aj bezpečnosť. Doteraz sme vyvíjali len u seba na lokálnom počítači a teda sme neriešili produkciu. Avšak momentálne s príchodom nášho prvého používateľa je nutné vyvinúť systém autentifikácie. Autentifikáciu a autorizáciu prístupu k API implementujeme s možnosťou obmedzenia na čítanie výsledkov, vkladanie nových záznamov a administráciu.

Autentifikácia je založená na Json Web Tokenoch, a preto nám odpadá nutnosť mať akúkoľvek databázu pre autentifikáciu. Tento mechanizmus bude potrebné dôkladne otestovať a v prípade potreby implementovať dodatočnú funkcionalitu ktorá vyplynie z požiadaviek zákazníkov pri testovaní.

100 najhorších. Výhodou používania nášho systému je hlavne rýchle nájdenie zlých komentárov. Naším cieľom je teda ponúknuť SME.sk 100 najhorších príspevkov za posledných 24 až 72 hodín, podľa ich nastavenia. Týmto spôsobom nájdu zlé komentáre oveľa rýchlejšie. Pre túto funkcionalitu bude potrebné vystaviť v API prístupový bod, ktorého výsledkom bude zoznam identifikátorov 100 najhorších príspevkov, ktorý sa vyrenderuje v ich moderačnom rozhraní.

Refaktor javascriptu. SME.sk chce výsledky analýz zobrazovať priamo v našom portáli a aj kvôli nim, aj do budúcna bude pre nás dôležité urobiť refaktor javascriptu, ktorý zobrazuje grafy, nakoľko sa pri ňom využíva veľa princípov metaprogamovania a nie je v tom prehľad. Upravovať tento javascript je veľmi zdĺhavé a často to môže viesť k nečakaným dôsledkom. Preto je veľmi dôležité urýchlene tento javascript upraviť, aby bolo možné a jednoduchšie vyvíjať nad ním niečo.

Ďalší vývoj

Filtrovanie a zoraďovanie. Naše moderačné rozhranie (portál) je taktiež momentálne len prototypom a teda nie je pripravený na úplne reálnu prevádzku. Sú v ňom síce zobrazené komentáre, ale portál funguje svižne iba pre málo komentárov. Toto však nie je prípad

diskusných portálov. Na niektorých slovenských denníkoch pribudnú denne tisícky komentárov. Niektoré články obsahuju stovky komentárov a našou úlohou bude zachovať aj pri takýchto množstvách komentárov prehľadnosť a svižnosť portálu. Potrebné bude najmä zavedenie stránkovania, alebo tzv. nekonečného skrollovania s postupným načítavaním komentárov do zoznamu. Nad takýmto množstvom príspevkov budeme musieť umožniť aj efektívne filtrovať a zoradovať. Niektoré akcie a možnosti filtrovania a zoraďovania, ktoré by radi uvítali moderátori sme konzultovali aj s moderátorom z Denníku N. Išlo o filtrovanie podľa článku, autora článku, podľa viacerých článkov a zoraďovanie podľa času alebo kvality komentára odhadnutej našou službou.

Portál. Popri práci na integrácii našich výsledkov do už existujúcich moderačných rozhraní chceme pokračovať aj vo vývoji nášho používateľského rozhrania (portálu), ktoré chceme optimalizovať pre efektivitu moderovania. Naším cieľom je teda spraviť rozhranie v ktorom možno čo najefektívnejšie označovať príspevky za vhodné, alebo nevhodné do diskusie. Všetky rozhodnutia chceme zjednodušovať pomocou analýz z našej služby a takto moderátorom poskytovať nástroj pre podporu rozhodovania.

Webhooky. Veľkú prioritu má aj implementácia vrátenia výsledkov analýz pomocou webhookov. Toto bude uprednostňovaný spôsob vrátenia výsledkov kvôli jednoduchšiemu škálovaniu a nižšej záťaži na naše servery v porovnaní s dopytovaním sa na výsledky. Pri konzultácii so sli.do ohľadom prípadnej integrácie sme sa rozprávali o možnostiach použitia technológie websocketov na medziserverovú komunikáciu. Ak by aj spolupráca so sli.do nevyšla, tak websockety budeme naďalej vyhodnocovať ako jednu z reálnych možností na prepojenie serverov diskusných portálov s našimi API servermi.

Pravidlový klasifikátor. Naša služba je založená na klasifikátore, ktorému sú predložené črty a ktorý sa na základe už ohodnotených dát učí. Do toho by sme však radi zapracovali fuzzy vyhodnocovanie, kde by sme si dokázali určiť, pri akej hodnote, ktorej črty môžeme považovať komentár za zlý alebo naopak dobrý. Mohli by sme sa riadiť kódexmi jednotlivých denníkoch, ale vyžaduje si to aj ďalšie konzultácie s denníkmi, ktoré majú o týchto pravidlách portálov s našimi API servermi.

Celkový pohľad na systém

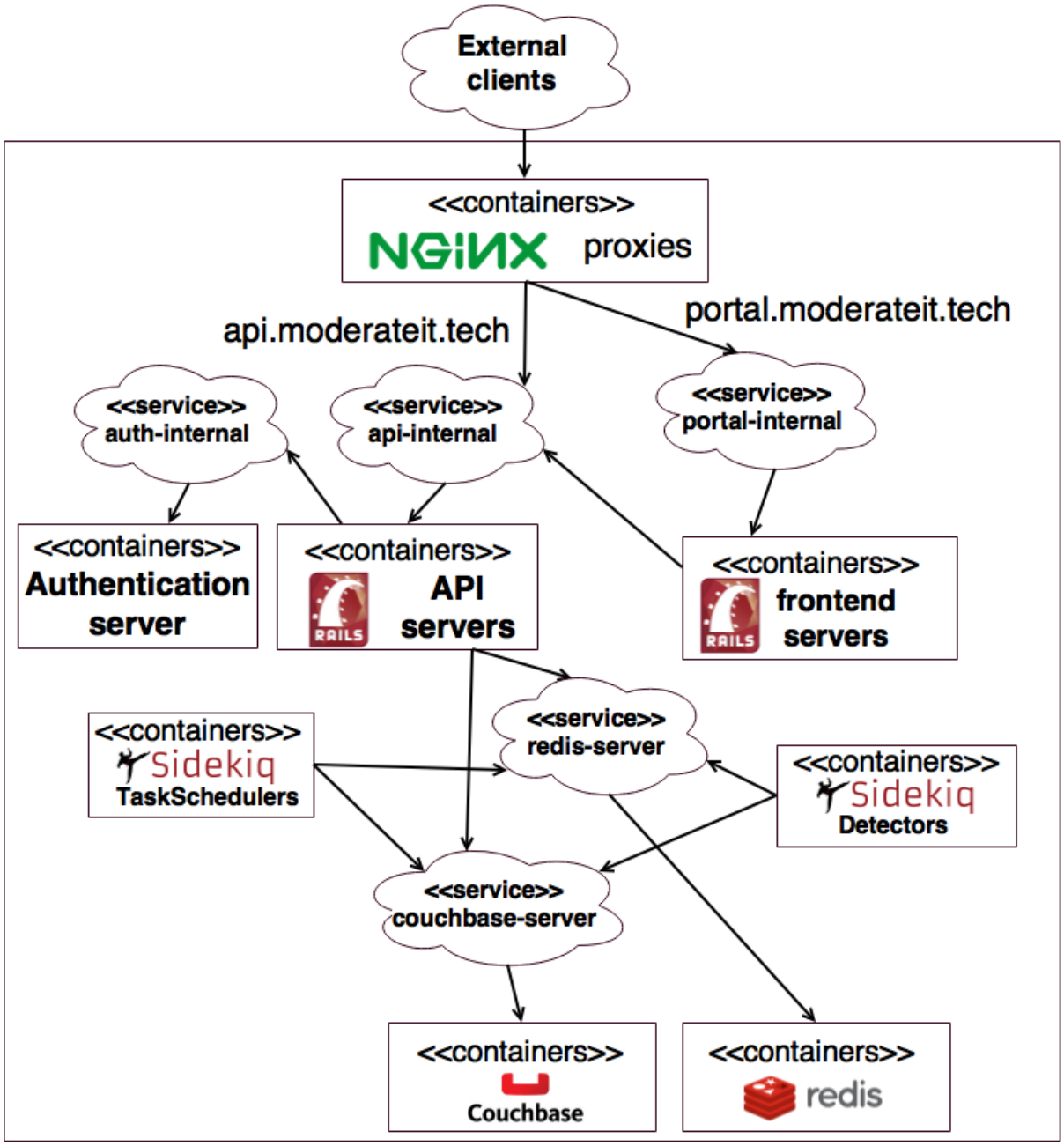
Úvod do architektúry

Jadrom služby je schopnosť prijať nový komentár, analyzovať ho a poskytnúť výsledky analýzy zákazníkovi. Keďže pri analýze využívame aj externé služby a celková doba analýzy môže narásť aj na niekoľko sekúnd, tak sme sa rozhodli, že analýzy sa budú vykonávať asynchrónne na pozadí. Požiadavka na analýzu nového komentára teda končí potvrdením o prijatí a nečaká sa na výsledky analýzy. Po prijatí komentára sa v odpovedi vráti aj URL odkazujúci na miesto kde budú dostupné výsledky po skončení analýzy. V ďalších iteráciách plánujeme implementovať vrátenie výsledkov pomocou Webhookov - teda preddefinovaných HTTP endpointov, na ktoré po skončení analýzy klientovi pošleme výsledok.

Rozmiestnenie a nasadenie komponentov

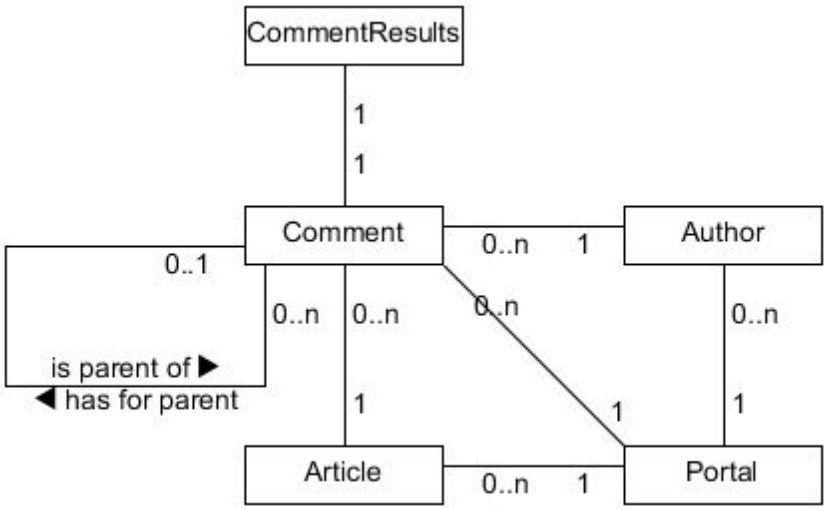
Služba moderateIT je nasadená v Microsoft Azure cloude, pretože na Azure máme k dispozícii kredit prostredníctvom programu Microsoft BizSpark. V Azure sme vytvorili cluster pozostávajúci z niekoľkých virtuálnych strojov. Služby nasádzame použitím Docker kontajnerov, o ich rozmiestnenie v clustri sa stará Openshift, ktorý nám výrazne zjednodušuje činnosti súvisiace s prevádzkou infraštruktúry (napríklad vnútorné DNS, rozmiestnenie kontajnerov, automatické zotavovanie po zlyhaní, load balancing, bezvýpadkové aktualizácie služieb).

Nasledujúci diagram zobrazuje vysokoúrovňové prepojenie medzi jednotlivými komponentami aplikácie.



Dátový model

Dátový model možno vidieť na nasledujúcom obrázku. Tento model sme vytvorili v databáze Couchbase. Couchbase je databáza, ktorá obsahuje dokumenty uložené v úložisku typu kľúč-hodnota. Nepodporuje žiadne cudzie kľúče, a preto si všetky validácie robíme ručne.



Moduly

- API
- Autentifikacny server
- Klasifikator
- Diakritikovac
- Portal

Moduly systému

Autentifikačný server

Analýza

Pri každom requeste do API potrebujeme overiť a zistiť o ktorého zákazníka ide. Preto potrebujeme efektívny spôsob ako tieto informácie z requestu vytiahnuť. Zároveň by sme chceli zachovať jednoduché a bezpečné používanie API z pohľadu zákazníka. Zvolili sme autentifikáciu pomocou JSON Web Tokenov (JWT). JWT je JSON, ktorý v sebe nesie akúkoľvek informáciu, ktorú potrebujeme, v našom prípade nás zaujíma predovšetkým zákazníkove ID a čas expirácie tokenu. JWT je bezpečne podpísaný našim súkromným kľúčom, ktorý sa používa aj pri overení tokenu, takže je zabezpečené, že token ktorý prejde validáciou sme vydali my a nevyrobil ho niekto iný. V neskorších fázach do JWT môžeme

uložiť aj informáciu o právach, ktoré je s ním možné vykonávať - zákazníci si tak budú môcť vygenerovať osobitné tokeny pre rôzne prípady použitia, napríklad moderátori by mohli mať právomoc čítať, schvaľovať a zamietať komentáre, avšak nemohli by meniť konfiguráciu systému.

Identifikovali sme 4 hlavné požiadavky na túto službu:

1. dokázať vytvoriť nový JWT s informáciami o zákazníkovi
2. dokázať validovať JWT - teda povedať, či bol JWT vytvorený našim systémom (podľa podpisu) a či už nie je expirovaný
3. vytiahnuť z JWT údaje o zákazníkovi
4. rýchla odozva - keďže každé jedno volanie do API musí prejsť autentifikačným serverom, tak je potrebné, aby tento server nespôsoboval ďalšie spomalenie alebo úzke hrdlo (bottleneck) pri vybavovaní requestov

Návrh

Autentifikačný mechanizmus sme sa rozhodli implementovať ako službu, pretože predpokladáme, že v budúcnosti to bude potrebovať viac aplikácií a nechceme mať viac implementácií tej istej logiky v rôznych aplikáciách. Jednotlivé časti systému by teda mali vedieť fungovať bez akejkoľvek znalosti o tom čo je to JWT a ako sa spracováva, pretože túto logiku za nich rieši autentifikačný server. V prvej fáze od služby požadujeme aby pomocou HTTP volania vedela overiť platnosť JWT a vytiahnuť z neho ID zákazníka. Pri generovaní nového JWT pomocou HTTP volania by bolo nutné zabezpečiť, aby toto volanie okrem nás nemohol urobiť nik iný. Táto operácia nazačiatku nebude častá a nebude potrebné ju automatizovať, preto stačí, aby sa nové JWT dali generovať ručne z príkazoveho riadku.

Implementácia

Na implementáciu sme zvolili jazyk Go, ktorého vlastnosti ho robia takmer ideálnym na tento typ služby - malá, jednoduchá, výkonná služba dostupná pomocou HTTP API. Jazyk Go má v sebe všetko potrebné na implementáciu tejto služby, ako jedinú externú knižnicu používame knižnicu go-jwx, ktorá obsahuje všetko potrebné na prácu s JWT.

Veľkou výhodou použitia JWT je to, že potrebné informácie o zákazníkovi dokážeme uložiť priamo do tokenu. Toto umožňuje aby služba bola tzv. bezstavová (stateless), čiže nepotrebuje načítavať a ukladať údaje do databázy alebo na disk.

Služba má 2 návratové kódy:

- 403 Forbidden - poskytnutý JWT je z nejakého dôvodu neplatný (môže byť zle naformátovaný, expirovaný, neplatný podpis,..).
- 200 OK - poskytnutý JWT je platný. V tomto prípade je v HTTP odpovedi nastavená hlavička X-Moderateit-Portal, ktorej hodnota je zákazníkove ID v našom systéme (vytiahnuté z JWT)

Testovanie

Keďže vytvorená služba poskytuje zatiaľ len jednoduchú funkcionalitu, tak sme ku nej nepísali automatické testy. Jej funkčnosť je možné overiť niekoľkými HTTP volaniami, ktoré sa dajú rozdeliť do 2 kategórií:

- poskytnutý platný JWT - návratový kód volania musí byť 200 a v HTTP odpovedi musí byť nastavená hlavička X-Moderateit-Portal s hodnotou zákazníkovho ID vytiahnutého z JWT
- čokoľvek iné musí vrátiť návratový kód 403 BEZ nastavenej HTTP hlavičky X-Moderateit-Portal. Čokoľvek iné zahŕňa expirované tokeny, tokeny vygenerované s použitím iného podpisu, náhodné a prázdne (resp. neposkytnuté) tokeny.

Jednou z požiadaviek na službu bola aj jej rýchla odozva, čo sme otestovali jednoduchým benchmarkom s použitím programu Apache Benchmark. Test pozostával zo 100 000 HTTP volaní na overenie poskytnutého tokenu, pričom bolo robených vždy 20 volaní súbežne.

Tu sú výsledky po niekoľkých testoch:

Requestov za sekundu: 5758.67

Priemerná doba odozvy: 3.473ms

Z testov tiež vyšlo, že 95% requestov malo dobu odozvy 10ms a menej, 99% malo dobu odozvy 19ms a menej, najhoršia doba odozvy bola 168ms. Tieto výsledky považujeme za dostatočne dobré pre naše použitie, obzvlášť keď vezmeme do úvahy, že testovaná záťaž bola približne 1000x vyššia, ako očakávame pri nasadení služby pre zákazníka veľkosti SME.sk.

Klasifikátor

Analýza

Pri analýze komentárov z príspevku extrahujeme množstvo znakov. Vstupom pre klasifikátor sú tieto jednotlivé znaky a výstupom musí byť jedno číslo, celková kvalita komentára. Klasifikátor bol vytvorený už minulý akademický rok v rámci bakalárskej práce. Trénovaný a testovaný je na dátach z portálu SME.sk, nakoľko je to náš prvý zákazník. Pre každého zákazníka budeme vytvárať vlastný klasifikátor.

Návrh

Klasifikátor sme tiež navrhovali ešte minulý rok v rámci bakalárskej práce. Je možnosť použiť viacero modelov. Ide najmä o rozhodovacie stromy, neurónové siete, alebo SVM. Tieto všetky sme vyskúšali a následne sme vybrali taký model, ktorý mal najväčšiu správnosť nad testovacím datasetom a zároveň mal čo najvyššie F1 skóre.

Implementácia

Náš klasifikátor beží v prostredí Microsoft Azure Machine Learning. V tomto prostredí sa dá jednoducho natrénovať model, ktorý je možné vystaviť vo forme HTTP REST API. Naše API sa následne pripája ku klasifikátoru prostredníctvom tohto API kedykoľvek je potrebné získať celkové hodnotenie komentáru na základe jeho opisu pomocou znakov, ktoré z neho extrahujeme.

Azure Machine Learning Studio používame kvôli tomu, že minulý rok sme sa zúčastnili technologickej súťaže Imagine Cup, kde bolo použitie Azure jednou z postačujúcich podmienok účasti.

Testovanie

Úspešnosť klasifikátora testujeme použitím testovacej množiny dát priamo v Azure Machine Learning Studio. To, či funguje vystavené API testujeme ukážkovým volaním, ktoré je možno spraviť opäť priamo v Azure Machine Learning Studio.

Diakritikovač

Analýza

Keďže veľká časť diskusných príspevkov je písaná bez použitia diakritiky, rozhodli sme sa na zvýšenie presnosti analýzy túto diakritiku do príspevkov doplniť. Existuje viacero verejne dostupných služieb, ktoré sú určené práve na túto úlohu.

Jednou z nich je napríklad štatistický diakritikovač vytvorený na Fakulte informatiky a informačných technológií, dostupný na adrese http://text.fiit.stuba.sk:8081/, ktorý dosahuje úspešnosť až 98%. Použitie tejto služby je pomerne jednoduché no rýchlosť jeho odozvy je niekedy nedostatočná.

Ďalšou možnosťou je služba poskytovaná Jazykovedným ústavom Ľudovíta Štúra, dostupná na adrese http://korpus.juls.savba.sk/diakritik.html . Táto služba v niektorých prípadoch dosahuje lepšie výsledky a má aj kratšiu dobu odozvy, ako vyššie spomínaná služba, no jej nevýhodou je, že nie je poskytovaná ako API, takže práca s ňou je zložitejšia.

Návrh

Rozhodli sme sa používať vyššie spomínaný štatistický diakritikovač, ktorý sme so súhlasom autora modifikovali a kvôli zlepšeniu odozvy nasadili na vlastný server, aby sme neboli závislý na službe poskytovanej treťou stranou.

Oproti pôvodnej službe došlo k niekoľkým zmenám. Prvou zmenou je zmena formátu správ ktorými toto API komunikuje. Ďalšou je zmena databázy, ktorú používa.

Implementácia

Keďže konkrétna implementácia tejto služby nie je dielom žiadneho z členov tímu, nebude opisovaná do detailov a vyjadríme sa len k zmenám, ktoré boli nami vykonané a všeobecnému opisu tejto služby.

Tento diakritikovač pri rekonštrukcií vychádza z kontextu. Využíva sa jazykový model slovenčiny, ktorý bol vytvorený na JÚĽŠ SAV v Bratislave. Úspešnosť rekonštrukcie sa pohybuje nad úrovňou 98%. Diakritika je doplnená len do gramaticky správne napísaných slov. Slová s preklepmi nie sú rekonštruované.

Je implementovaný v jazyku python a využíva hlavne opensource knižnice, ako sú kenlm (práca s jazykovým modelom) a web.py (spracovanie http requestov).

Ako bolo vyššie spomenuté, oproti pôvodnej implementácii sme urobili niekoľko zmien. Prvou zmenou je, že ako databázu sme namiesto MySQL použili redis, hlavne kvôli jeho rýchlosti. Druhou zmenou bola zmena API rozhrania. Namiesto povôdne použitých GET requestov, kedy sa text určený na diakritikovanie posielal ako parameter v url, sme sa rozhodli použiť POST requesty obsahujúce jednoduchý JSON. Taktiež bola zmenená forma odpovedí z API. Nemiesto pôvodných XML správ sa v súčasnosti taktiež posielaju jednoduché JSON-y.

Testovanie

Testovanie tejto služby je vykonané v rámci testovania triedy DiacriticDetector. Počas týchto testov sa porovnáva očakávaný textu s diakritikou a text, ktorý pre rovnaký text bez diakritiky vráti po zaslaní requestu diakritikovač. Taktiež je testovaný aj kód odpovede na http POST request.

Portál

Analýza

Primárnym účelom portálu bola priniesť verejnosti predstavu o práci moderátorov a taktiež priblížiť ako naša služba funguje. Či už grafickým znázornením jednotlivých znakov komentárov a autorov alebo možnosť napísať si vlastný komentár, ktorý bude do pár sekúnd vyhodnotený. Taktiež sa dali koemntáre filtrovať podľa článkov, aby si moderátor mohol vybrať pod, ktorým článkom chce moderovať.

Portál bol predstavený denniku, ktorý nám zadefinoval svoje požiadavky, tak aby sa mohli dopracovať k výsledkom kvality komentára a a reputácie autora. Taktiež si denník vyžiadal prístup k bližším informáciám o komentári a autorovi.

Návrh

Pri vytváraní grafického rozhrania sme si dali záležať na tom, aby bolo čo najlepšie prispôsobené pre používateľa. Snažili sme sa dodržiavať základné princípy UX dizajnu. Najskôr vznikol prvý prototyp na papieri, aby sme mali jasnú predstavu o tom kam umiesntime rôzne prvky.

Na to, aby sme denníku mohli zobraziť potrebné informácie sme museli nás portál upraviť, aby sa dal používať ako iframe. Moderátor bude mať k dispozícií vysledky komentára a autora v malom iframe. V závislosti od toho na aký graf klikne sa mu zobrazia podrobnejšie štatistiky autora alebo komentára. V neskorších fázach pribudne zoznam 100 najhorších príspevkov pre moderátora.

Implementácia

Pre používanie portálu denníkom je potrebné, aby si nastavili cookie moderateit_api_token, ktorá im zabezpečí, že sa nebudú musieť nikam prihlasovať na to, aby sa dostali k iframom a výsledkom, ktoré potrebujú. Do svojho moderačného rozhrania si implementujú iframy s odkazom na náš portál. Následne po kliknutí na výsledok komentára sa zobrazia v novom okne grafy, ktoré znázorňujú črty komentára. Napríklad počet nadávok, gramatická správnosť komentára či bohatosť slovnej zásoby v komentári. To isté platí pre autora. Po kliknutí na malý graf v iframe znázorňujúci reputáciu používateľa sa zobrazia črty opisujúce autora. Napriklad odkedy je autor komentára zaregistovaný alebo informícia o tom koľko príspevkov je zmazaných.

Zoznam článkov, zoznam komentárov a input pre zadanie nového komentára boli z portálu zmazané, pretože nie sú potrebné pre denník. Výsledky komentára a autora komentára a taktiež ostatné informácie sú dotiahnuté do portálu z api pomocou requestov.

Testovanie

Jedna z vecí, ktorú bolo pri portáli otestovať, bolo grafické rozhranie. Rozhranie sme dali vyskúšať známym, ale aj samotným predstaviteľom denníkov. Dostali sme spätnú väzbu o tom, že potál je veľmi ľahko použiteľný. Informácie sú zobrazené prehľadne.

Tiež sme testovali portál ručne či sa grafy správne zobrazujú aj v prípadoch keď výsledky nemáme k dispozícií. Na autentifikáciu cez cookie máme napísaný test.

API

Analýza

moderateIT API je hlavnou časťou nášho projektu. Do toto API sú zasielané dáta zákazníka, ktoré má služba ohodnotiť. Po prijatí potrebných dát a ich zvalidovaní je tento modul zodpovedný za spustenie potrebných analýz, uloženie ich výsledkov do databázy a ich následné zaslanie na vstup klasifikátora, ktorý bol opísaný vyššie.

S týmto API však nekomunikuje len strana zákazníka, ale rovnako aj nami vytvorený portál. Ten po prijatí požiadavky na zobrazenie výsledkov analýz urobí request do tohto api a zobrazí získané výsledky.

Návrh

Už od začiatku projektu sme sa rozhodli vytvoriť API, ktoré bude dodržiavať “best practices” pre tvorbu REST API. Ide hlavne o dodržiavanie správneho pomenovávania, využívania správnych návratových kódov pre jednotlivé volania a ďalších maličkostí, ktoré však v

konečnom dôsledku majú veľký prínos pre jednoduchšiu prácu s týmto API. Aplikovanie týchto princípov je možné vidieť napríklad v dokumentácii k tomuto API (http://docs.moderateitapi.apiary.io).

Implementácia

API je rovnako ako väčšina projektu implementovaná v jazyky RUBY, s použitím frameworku Rails. Celé toto API beží na webserveri puma (http://puma.io/), ktorý poskytuje lepšie možnosti ako defaultný rails server webrick.

Ako databáza na ukladanie informácií o článkoch, komentároch, autoroch a výsledkov analýz využívame dokumentovú NoSQL databázu Couchbase (http://www.couchbase.com/). Na ukladanie medzivýsledkov analýz a konštánt používaných pri analýze používame Redis (http://redis.io/).

Kedže jednotlivé detektory spracúvajú dáta na pozadí, na ich manažovanie využívame externú knižnicu Sidekiq (http://sidekiq.org/).

Testovanie

Keďže sa jedná o najdôležitejšiu časť projektu, aj jej testovaniu sa prikladá najväčší význam. Je našou snahou pokryť testami každú časť tohto API od reakcie na requesty, cez činnosť jednotlivých detektorov až po komunikáciu, či už s portálom alebo klasifikátorom.

Používateľské testovanie

Na konci prvého semestra sa nám podarilo dohodnúť si stretnutie u nášho zákazníka SME.sk. Počas tohto stretnutia bola odprezentovaná služba ako výsledok našej práce, ktorá dokáže hodnotiť kvalitu komentárov. Zákazníkovi sme ukázali zoznam komentárov, ktorý bol usporiadaný podľa kvality. Komentáre s najhoršou kvalitou si zákazník mohol pozrieť ako prvé. Ku každému komentáru prislúchal graf s podrobnými štatistikami autora a komentára. Reakcia zo strany zákazníka bola veľmi pozitívna.

Na základe tejto prezentácie sme si dohodli ďalšie kroky. Podarilo sa nám získať nové dáta od SME.sk za rok 2016, na to aby sme mohli našu službu vylepšovať. SME.sk od nás chcelo, aby sme im umožnili do ich stránky pridávať iframy z nášho portálu s výsledkami kvality a konkrétnymi štatistikami. Tieto požiadavky sme zapracovali, poslali sme im informácie s prístupom do našej služby a potrebnú dokumentáciu. Následne developer zo SME.sk pracoval na integrácií našej služby do ich systému a komunikácia prebiehala online.

Začiatkom roka nám SME oznámilo, že sa chystajú robiť väčšie zmeny v ich systéme a bude nutné pozastaviť integráciu našej služby. My sme pracovali na vylepšovaní výsledkov vyhodnocovania kvality, zatiaľ čo sme čakali na to kým nám dajú vedieť, že obnovili prácu na integrácií. Tento proces sa pomerne dosť natiahol a bolo by pre nás lepšie, ak by sme sa im ozvali skôr a veci urýchlili.

Po pár mesiacoch sme sa s nimi skontaktovali opätovne a dohodli sme si ďalšie stretnutie, ktoré bude prebiehať v nasledujúcich dňoch. Na tomto stretnutí sa budú diskutovať ďalšie kroky a možnosti aké máme, vzhľadom na to, že sme.sk sa rozhodlo, že začnú používať rozhranie pre diskusie od Coral Projectu. Tiež by sme im chceli odprezentovať zoznam najhorších komentárov za isté časové obdobie v dátach, ktoré nám poskytli. Komentáre je možné filtrovať podľa dátumu, ale defaultne sa zobrazí 50 najhorších komentárov za posledných 24 hodín. Cieľom tejto prezentácie je poukázať na to, ako rýchlo ich naša služba dokáže upozorniť na tie najhoršie komentáre spomedzi obrovského množstva, ktoré za deň pribudnú do diskusií. Čím sa rýchlo vedia dostať aj ku komenárom porušujúcim ich kódex, ktoré im predtým unikli.

Pri analyzovaní najhorších komentárov za daný deň sa nám podarilo odhaliť niekoľko nedostatkov vo vyhodnocovaní kvality, ktoré nám umožnili vylepšiť službu. Taktiež sme upravili dataset, na ktorom je náš algoritmus testovaný. Na testovacej vzorke sa momentálne
naša average precision pohybuje okolo 75-76% percent, avšak testovacia vzorka nie je ideálna a bude potrebné ju vylepšovať.

Trollhunt

Analýza

Keďže dát, ktoré nám boli poskytnuté z portálu SME.sk obsahovali len cca 45 000 komentárov, ktoré boli ich moderátormi vymazané, kvôli ich nevhodnosti, rozhodli sme sa obohatiť si náš dataset ďalšími dátami.
Tentokrát sme sa rozhodli ako zdroj dát použiť portály, v ktorých diskusiách sa nachádza viac „bahníčka“. Konkrétne sme stiahli diskusie z portálu topky.sk, ktorý sa často zameriava aj na kontroverznejšie témy a bulvár, takže je väčšia pravdepodobnosť, že sa v ich diskusiách objavia nevhodné, urážlivé komentáre, vďaka ktorým sa zvrhne diskusia zlým smerom.
Problémom bolo, že dáta získané z týchto diskusií nie sú žiadnym spôsobom označkované, takže pred samotným trénovaním nad týmito novo-získanými dátami potrebujeme ručne prejsť a označkovať, čo najväčšie množstvo týchto komentárov. Práve na tento účel sme vytvorili aplikáciu „Trollhunt“.

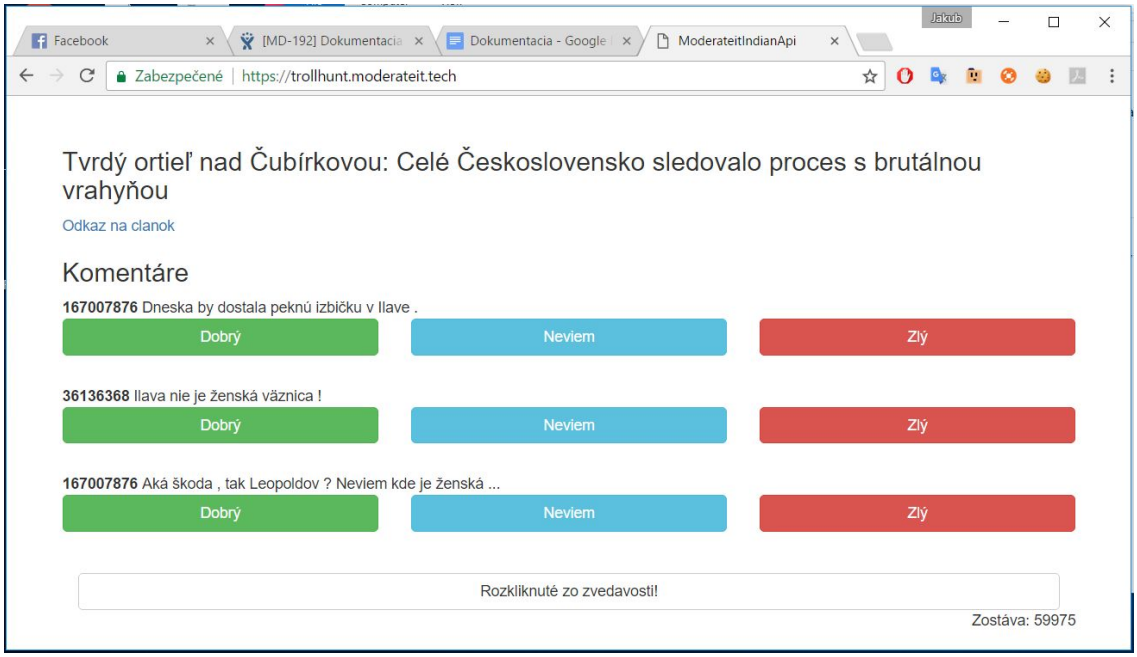
Návrh

Aplikácia je dostupná na adrese http://trollhunt.moderateit.tech . Po otvorení tejto adresy vo webovom prehliadači, je používateľovi zobrazený jeden náhodný komentár z doteraz neohodnotených, spolu s názvom článku a odkazom naň, aby si používateľ v prípade potreby mohol prečítať tento článok a zistiť, daný komentár nie je „offtopic“. Taktiež má používateľ možnosť zobraziť všetky predchádzajúce komentáre, na ktoré aktuálny komentár reagoval, a môže tak lepšie pochopiť, ako sa daná diskusia vyvíjala. V aktuálnom stave je možné označiť každý komentár len raz. Rozmýšľali sme nad možnosťou požadovať viacero

označení, aby sme sa vyhli možnosti zlého označkovania, ale keďže potrebujeme v čo najkratšom čase označkovať, čo najväčšie množstvo komentárov, rozhodli sme sa to nechať len v štádiu možného vylepšenia a neimplementovať to v aktuálnej verzii.
Pri každom so zobrazených komentárov sú zobrazené 3 tlačidla „Dobrý“, „Zlý“ a „Neviem“, po kliknutí na ktoré sa odošle požiadavka na zaradenie komentára do jednej z týchto kategórií. Do kategórie „Dobrý“ patria komentáre, ktoré podľa názoru používateľa neporušujú žiadne zásady diskusného kódexu. Do kategórie „Zlý“ patria naopak komentáre, ktoré používateľ nepovažuje za vhodné do diskusie. Kategória „Neviem“ je vyhradená pre komentáre, ktoré nie je možné jednoznačne zaradiť medzi „zlé“, no nepatria ani medzi „dobré“, v podstate sú to komentáre balansujúce na hrane medzi týmito dvoma kategóriami.

Implementácia

Aplikácia je vytvorená v Ruby on Rails. Ako databáza, v ktorej sú uložené komentáre sa rovnako ako aj v moderateIT API používa Couchbase. Keďže množstvo dát s ktorým aplikácia pracuje je pomerne veľké, pri práci s databázou sa kvôli efektívnosti používajú map-reduce funkcie. V aplikácií nie je implementovaný žiaden spôsob autentizácie používateľa, takže v aktuálnom stave môže dataset označkovať ktokoľvek, kto pozná adresu na ktorej sa aplikácia nachádza.
Pri tvorbe front-endu, ktorý je veľmi jednoduchý a bez zbytočných efektov, boli použité klasické webové technológie ako HTML, CSS, jQuery a Ajax. Aplikácia je responzívna, vďaka čomu je ju možné používať na akomkoľvek zariadení, ako sú napríklad počítač, tablet, mobilný telefón. Responzívnosť je dosiahnutá použitím frameworku Bootstrap. Ukážky aplikácie ako vo verzii pre počítače aj mobilné zariadenia sú na nasledujúcich obrázkoch.



Aplikácia Trollhunt zobrazená na PC

Aplikácia Trollhunt zobrazená na mobilnom telefóne

Príloha: A-1 Dokumentácia k API volaniam

moderateIT API

moderateIT JSON API for creating new comments, authors, articles. Authentication is done using tokens sent in 'Authorization' header with Bearer type (example: "Authorization: Bearer fkl23109ufrelkjsdfl2309ulkmnf2z").

Comments

SUBMITTING COMMENTS

POST `/comments` Create new comment

Submit new comment that you want to evaluate. After a successful response, the comment's quality will be evaluated asynchronously and the results can be retrieved at the URI received in 'Location' header.

Example URI

POST https://api.moderateit.tech//comments

Request `Create new comment` Hide

Headers

```
Content-Type: application/json
Authorization: Bearer TOKEN
```

Body

```
{
  "id": "uswn2s92smq1w",
  "content": "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididu
  "date_added": "2016-10-19T15:34:20Z",
  "parent_id": "Dj2W2kdaA24",
  "author_id": "dlkq20fm3213f",
  "article_id": "2md28dae210v8"
}
```

Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "id": {
      "type": "string",
      "description": "`Your internal ID of the comment. Must be unique.` (string, required)"
    },
    "content": {
      "type": "string",
      "description": "`Comment's body (plain text)` (string, required)"
    },
    "date_added": {
      "type": "string",
      "description": "`Timestamp of comment's creation time (ISO8601 format)` (string, required)"
    },
    "parent_id": {
      "type": "string",
      "description": "`Your internal ID of comment's parent comment (must exist in our database). Use
    },
    "author_id": {
      "type": "string",
      "description": "`Your internal ID of comment's author. If the author doesn't exist in our datab
    },
    "article_id": {
```

Back to top

```
      "type": "string",
      "description": "`Your internal ID of the article, which this comment belongs to. Article must e
    }
  }
}
```

Response `202` Hide

Comment successfuly created.

Headers

```
Location: /comments/{id}/results
```

Response `409` Hide

Comment ID not unique (comment with the same ID already exists in our database).

APPROVING COMMENTS

POST `/comments/{id}/approve` Approve comment

Mark comment as approved (by a moderator).

Example URI

POST https://api.moderateit.tech//comments/1dfbc1983/approve

URI Parameters Hide

id `string` (required) **Example:** 1dfbc1983
your internal comment ID

Request `Approve comment` Hide

Headers

```
Content-Type: application/json
Authorization: Bearer TOKEN
```

Response `200`

DELETE `/comments/{id}/approve` Undo approve comment

Marks the comment as not approved (neutral). This does NOT mean that the comment is inappropriate. It should be used when moderator marks the comment as approved, but then changes his mind or realizes that he made a mistake (undo action).

Example URI

DELETE https://api.moderateit.tech//comments/1dfbc1983/approve

URI Parameters Hide

id `string` (required) **Example:** 1dfbc1983
your internal comment ID

Request `Undo approve comment` Hide

Headers

```
Content-Type: application/json
Authorization: Bearer TOKEN
```

Response `200`

REJECTING COMMENTS

POST `/comments/{id}/reject` Reject comment

Marks comment as rejected (by a moderator).

Example URI

POST https://api.moderateit.tech//comments/1dfbc1983/reject

URI Parameters Hide

id `string` (required) **Example:** 1dfbc1983
your internal comment ID

Request `Reject comment` Hide

Headers

```
Content-Type: application/json
Authorization: Bearer TOKEN
```

Response `200`

DELETE `/comments/{id}/reject` Undo reject comment

Marsk the comment as not rejected (neutral). It should be used as UNDO action with the same meaning as 'Undo approve comment'.

Example URI

DELETE https://api.moderateit.tech//comments/1dfbc1983/reject

URI Parameters Hide

id `string` (required) **Example:** 1dfbc1983
your internal comment ID

Request `Undo reject comment` Hide

Headers

```
Content-Type: application/json
Authorization: Bearer TOKEN
```

Response `200`

REPORTING COMMENTS

POST `/comments/{id}/report` Report comment

Adds new report. You should call this endpoint when someone reports a comment to moderators (or marks it as potentially inappropriate).

Example URI

POST https://api.moderateit.tech//comments/dj2oid31/report

URI Parameters Hide

id `string` (required) **Example:** dj2oid31
Your internal comment ID

Request `Report comment` Hide

Headers

```
Content-Type: application/json
Authorization: Bearer TOKEN
```

Body

```
{
  "reporter_id": "d39qw2w9kmdi"
}
```

Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "reporter_id": {
      "type": "string",
      "description": "`Your internal ID of user who reported the comment` (string, required)"
    }
  }
}
```

Response `200`

Articles

Every comment must belong to an article. This article must be created BEFORE you submit comments related to this article.

SUBMITTING ARTICLES

POST `/articles` Create new article

Example URI

POST https://api.moderateit.tech//articles

Request `Create new article` Hide

Headers

```
Content-Type: application/json
Authorization: Bearer TOKEN
```

Body

```
{
  "id": "2md28dae210v8",
  "title": "The standard Lorem Ipsum passage, used since the 1500s",
  "content": "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididu
  "date_added": "2016-10-19T15:34:20Z"
}
```

Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
```

Back to top

```
    "id": {
      "type": "string",
      "description": "`Your internal ID of the article. Must be unique.` (string, required)"
    },
    "title": {
      "type": "string",
      "description": "`Article title.` (string, required)"
    },
    "content": {
      "type": "string",
      "description": "`Article's body (plain text).` (string, required)"
    },
    "date_added": {
      "type": "string",
      "description": "`Timestamp of article's creation time (ISO8601 format)` (string, required)"
    }
  }
}
```

Response `201`

Author

Every comment must have it's author (user participating in discussion). The best way to use the API is to submit new author whenever new user is registered. If you submit a new comment and it's author doesn't exist in our database, we will create the author with the same registration timestamp as the newly created comment, but this can result in lower quality evaluation.

CREATING AUTHORS

POST `/authors` Create new author

Example URI

POST https://api.moderateit.tech//authors

Request `Create new author` Hide

Headers

```
Content-Type: application/json
Authorization: Bearer TOKEN
```

Body

```
{
  "id": "dlkq20fm3213f",
  "nickname": "jozko87",
  "date_registered": "2016-10-19T15:34:20Z"
}
```

Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "id": {
      "type": "string",
      "description": "`Your internal ID of the author. Must be unique.` (string, required)"
    },
    "nickname": {
      "type": "string",
      "description": "`Authors nickname.` (string, optional)"
    },
```

```
    "date_registered": {
      "type": "string",
      "description": "`Authors registration date (ISO8601 format).` (string, required)"
    }
  }
}
```

Response `201`

BANNING AUTHORS

We can use this information to detect patterns in behavior of banned authors and improve quality of evaluation.

POST `/authors/{id}/ban` Ban author

Marks the author as banned.

Example URI

POST https://api.moderateit.tech//authors/a3caqm39/ban

URI Parameters Hide

id `string` (required) **Example:** a3caqm39
your internal author ID

Request `Ban author` Hide

Headers

```
Content-Type: application/json
Authorization: Bearer TOKEN
```

Response `200`

Generated by aglio on 14 May 2017

Back to top