SLOVENSKÁ TECHNICKÁ UNIVERZITA
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Aplikačné Architektúry Softvérových Systémov

Systém na skorú detekciu prírodných katastrof zo sociálnych sietí

Cvičenie: Ing. Eugen Molnár, PhD., štvrtok 16:00
Semester: letný semester, 2023/2024
Autori: Martin Melišek
Martin Nemec

Obsah

1. Prehľad a Analýza

V dnešnej dobe sú sociálne média významnou súčasťou globálneho zdieľania informácií. V rôznych ohladoch medzi médiá patria aj sociálne siete, ktoré napomáhajú k šíreniu rôznorodých správ. Milióny používateľov zdieľajú denne veľké množstvo príspevkov, ktoré nadobúdajú rozmanité formy. Vzhľadom na typ sociálnej siete, je jedným z dôležitých prostriedkov komunikácie práve text. Táto zložka komunikácie môže obsahovať skryté cenné informácie, ktoré má význam štruktúrovane spracovávať. V reálnom čase si ich môžeme predstaviť ako určitý typ senzorov, ktorý sa aktivuje ak sa zaznamená významná udalosť či už v prostredí internetu alebo aj sveta okolo nás.

Udalosti tohto typu si svojou podstatou vyžadujú rýchlu reakciu záchranárskych jednotiek, ktorým je možné pomôcť z hľadiska analýzy dostupných správ šíriacich sa sociálnymi médiami. Lokalita, rozsah, či kategória pohromy sú základnými druhmi informácií, ktoré má význam extrahovať. V prípade príspevkov získaných od internetových užívateľov je potrebné dokázať odfiltrovať veľké kvantum irelevantných dát a zamerať sa len na tie, ktoré sa týkajú danej katastrofickej udalosti.

S pomocou metód umelej inteligencie implementujeme systém na skorú detekciu katastrofických udalostí zo sociálnych sietí. Platforma bude používať natrénovaný model umelej inteligencie, ktorý na vstupe dostane text sociálneho príspevku, klasifikuje ho a ak je informatívny extrahuje z neho pomenované entity. Pokiaľ obsahuje text URL adresu, webové sídlo navštívime a doplníme získané entity aj z neho. Používateľ bude mať možnosť vytvoriť novú predikciu, získať entity, či zobraziť historické predikcie a relevantné dáta k danému príspevku.

2. Návrh a biznis scenár

2.1. Identifikácia zainteresovaných strán

Našimi kľúčovými stakeholdermi sú:

1. **Občan** - Obyvatelia zasiahnutého miesta katastrofy sú kľúčovými aktérmi z hľadiska informovania verejnosti. Dodatočné informácie, ktoré systém zozbiera, im môžu pomôcť pri evakuácií ak sa blíži prírodná katastrofa.

2. **Záchranár** - Záchranárskym jednotkám je možné poskytnúť dodatočné údaje, ktoré môžu zlepšiť ich pripravenosť na zásah a urýchliť či zlepšiť reakciu na prírodnú katastrofu

3. **Dispečer** - Vyslanie záchranárskych jednotiek na správne miesto katastrofy spolu s dodatočnými informáciami o tom, kedy bola katastrofa spozorovaná dopomáha vyriešiť problém v predstihu

4. **Koordinátor krízového štábu** - Pri vydávaní varovania občanom, koordinátor krízového štábu účinkuje ako dôležitá súčasť, ktorá poskytne občanom informácie o tom, že sa niečo deje.

5. **Administrátor IS** - Vzhľadom na pokrytie a získanie väčšieho množstva dát, administrátor informačného systému zadáva a konfiguruje zoznam sociálnych sietí. Je v jeho záujme rozširovať a dynamicky ho meniť na to aby bolo možné extrahovať čo najviac dát.

6. **Výskumník** - Modely umelej inteligencie je nutné priebežne aktualizovať a na základe získaných dát ich aj pretrénovávať. Výskumník dodáva model hlobkého učenia, ktorý vykonáva predikcie a má záujem ho vylepšovať aj na základe metodológie Human In the Loop, kedy používateľ (napr. občan) dokáže označiť správnu či nesprávnu predikciu modelom.

Každý stakeholder poskytuje komplexný pohľad na potreby a príležitosti pre služby systému na skorú detekciu katastrofických udalostí. Zohľadnenie potrieb je dôležité pre efektívne riadenie operácií, ale aj pre všeobecné informovanie občanov.

2.2. Motivation View

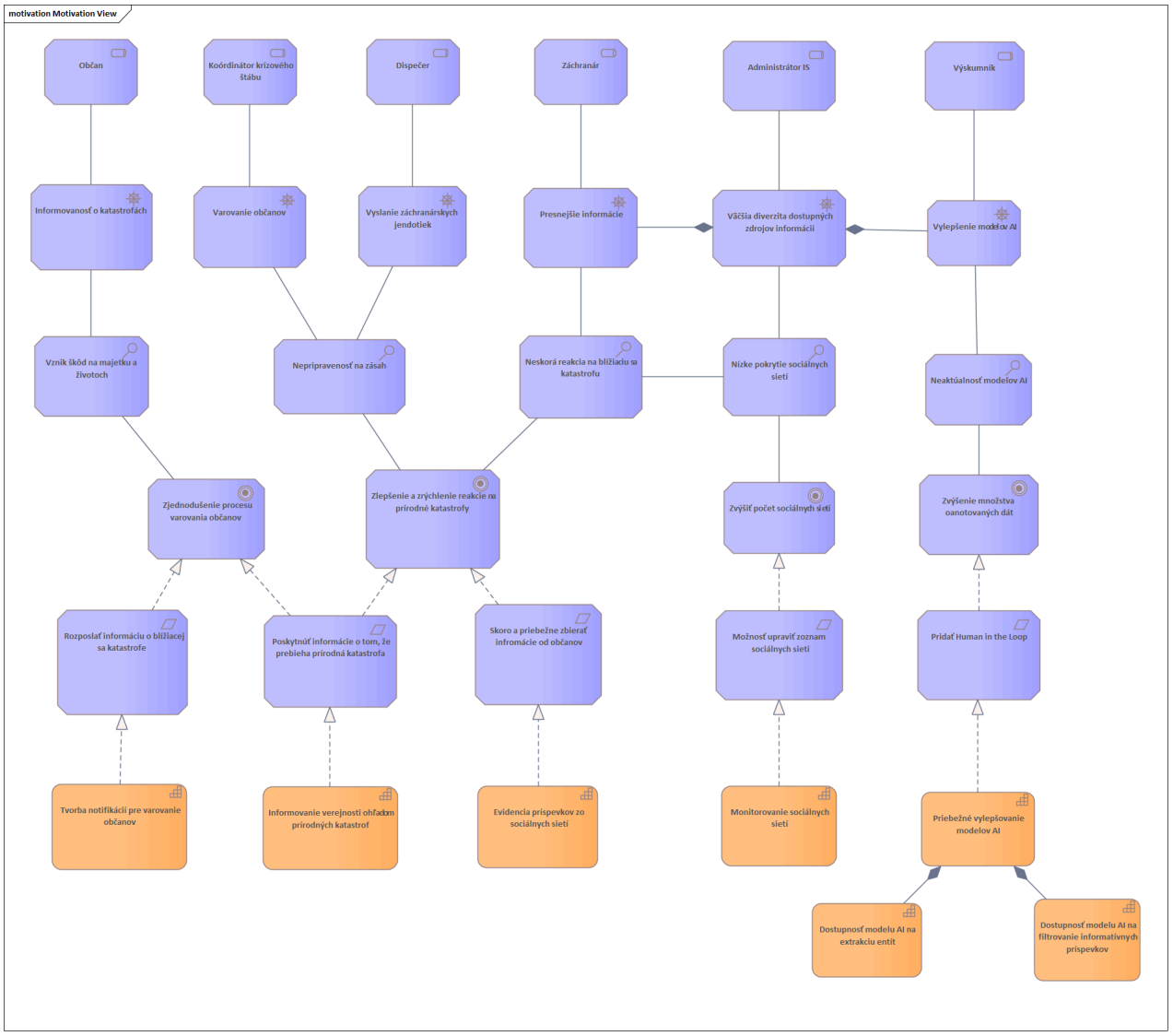
Našimi hlavnými cieľmi sú:

- **Zjednodušenie procesu varovania občanov.** Pokiaľ sa nám na základe získaných dát a predikcie modelom podarí lokalizovať a detegovať začínajúcu katastrofu, môžeme v predstihu varovať občanov, ktorí sa zdržiavajú na zasiahnutom mieste katastrofy. Získané informácie rozšíria všeobecné povedomie o tom, že sa niečo deje ako aj občanom tak aj záchranárom.
  Realizácia tohto cieľu spočíva v zavedení požiadavky na rozposlanie informácie o blížiacej sa katastrofy. Občania sa lepšie pripravia a predídeme tým škodám na majetku a stratených životoch.

- **Zlepšenie a zrýchlenie reakcie na prírodné katastrofy.** Dodatočné informácie extrahované z príspevkov zo sociálnych sietí pomôžu záchranárskym jednotkám konkrétnejšie identifikovať vznik problému, či prísť s riešením alebo v krajných prípadoch aj nájsť osoby priamo zastihnuté katastrofou. Zrýchlenie reakcie môže benefitovať z možnosti zberu informácií priamo od občanov. Pokiaľ koordinátorom a dispečerom budú dodané presnejšie informácie, môžu vyslať jednotky a varovať občanov, pričom sa zlepší ich celková pripravenosť na zásah.

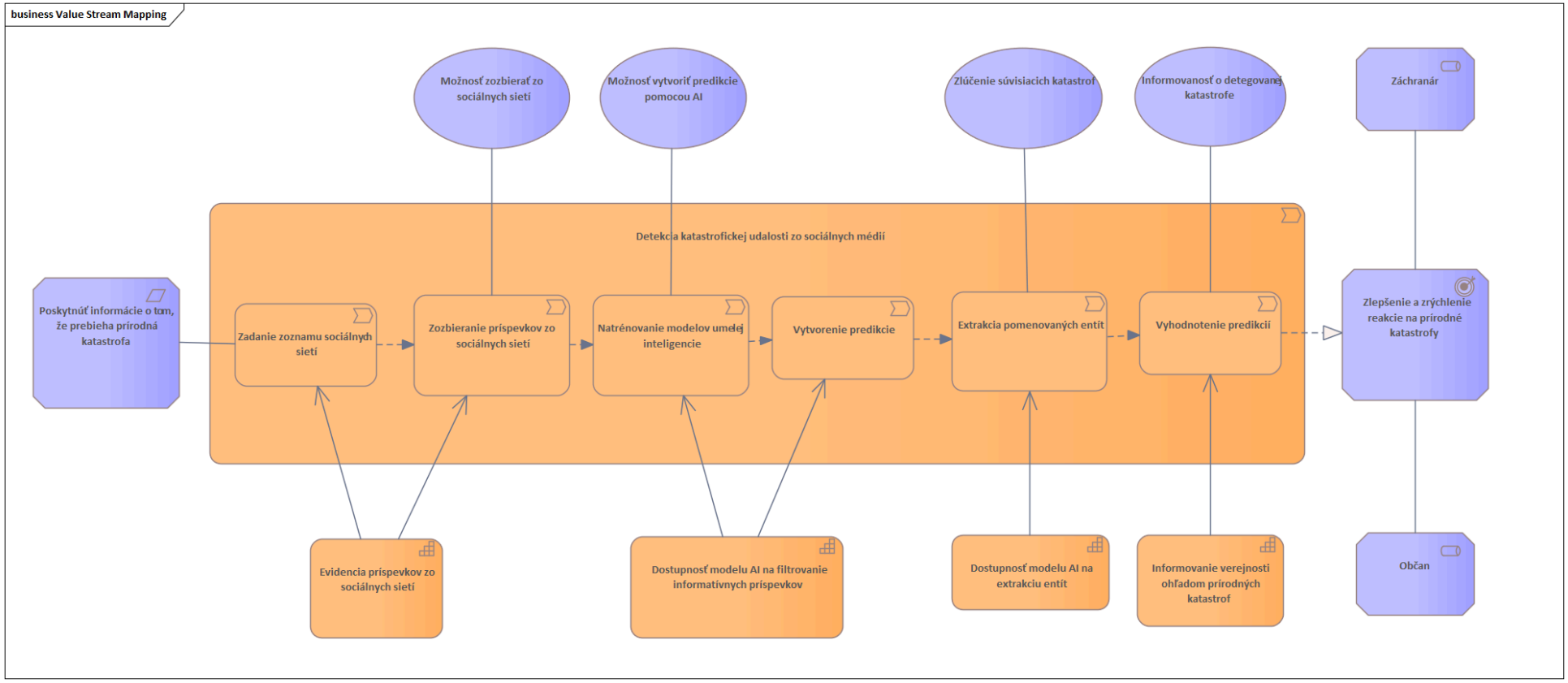
- **Zvýšiť počet pokrytých sociálnych sietí.** Systém svojou podstatou musí byť pripravený pracovať s veľkým množstvom sociálnych sietí a využívať model umelej inteligencie na filtrovanie neinformatívnych príspevkov. Zoznam týchto sietí je potrebné meniť v reálnom čase a dodatočné dáta zbierať na základe potreby. Pokiaľ bude zoznam pokrytých sociálnych sietí dostatočne vysoký, zvýši sa celkové pokrytie lokácií a typov katastrof, čím sa môže zlepšiť reakcia aj na netradičné katastrofy.

- **Zvýšenie množstva oanotovaných dát.** Na zlepšenie modelov umelej inteligencie, ktoré vykonávajú príslušné predikcie je nutné zaviesť proces, ktorý v systéme bude slúžiť na anotovanie stiahnutých dát používateľom, čím bude možné vylepšiť aktuálnosť modelov a zlepšiť ich presnosť. Pridanie tzv. Human In the Loop zabezpečí, že sa zvýši počet oanotovaných dát a modely umelej inteligencie zostanú aktuálne. Sú to práve tieto modely AI, ktoré ťažia z diverzity dát. Tým, že počas trénovania videli dáta, ktoré napísalo väčšie spektrum používateľov môžu byť lepšie pripravené na netradičné situácie a presnejšie filtrovať a získať metádata z príspevkov na mnohých sociálnych sieťach.



2.3. Value Stream Mapping

Nasledujúci diagram zobrazuje pohľad na sekvenciu krokov v rámci detekcie katastrofickej udalosti zo sociálnych sietí. V predchádzajúcej sekcii sme opísali dôvody jednotlivých požiadaviek a cieľov a na diagrame môžeme vidieť, že ak zrealizujeme systém nadobudne rôzne kapability, ktoré umožnia dosiahnuť stanovený hlavný cieľ - zlepšenie a zrýchlenie reakcie na prírodné katastrofy. Jednotlivé spôsobilosti sú úzko prepojené s hodnotami, ktoré vyplývajú z krokov v tomto procese. Inak povedané, ak napríklad zabezpečíme dostupnosť modelov AI na filtrovanie príspevkov, obdržíme možnosť vytvorenia predikcií v reálnom čase. Rovnako je na tom aj zbieranie príspevkov, extrakcia pomenovaných entít a na záver aj samotné vyhodnotenie predikcií. Iba v tomto bode môžeme vytvoriť analytickú správu, ktorá na základe zozbieraných dát a ich predikcií hovorí o tom, že práve prebieha prírodná katastrofa a má zmysel informovať o tom verejnosť.


business Value Stream Mapping
Možnosť zozbierať zo sociálnych sietí
Možnosť vytvoriť predikcie pomocou AI
Zlúčenie súvisiacich katastrof
Informovanosť o detegovanej katastrofe
Záchranár
Detekcia katastrofickej udalosti zo sociálnych médií
Poskytnúť informácie o tom, že prebieha prírodná katastrofa
Zadanie zoznamu sociálnych sietí
Zozbieranie príspevkov zo sociálnych sietí
Natrénovanie modelov umelej inteligencie
Vytvorenie predikcie
Extrakcia pomenovaných entít
Vyhodnotenie predikcií
Zlepšenie a zrýchlenie reakcie na prírodné katastrofy
Evidencia príspevkov zo sociálnych sietí
Dostupnosť modelu AI na filtrovanie informatívnych príspevkov
Dostupnosť modelu AI na extrakciu entít
Informovanie verejnosti ohľadom prírodných katastrof
Občan

2.4. Funkčné požiadavky

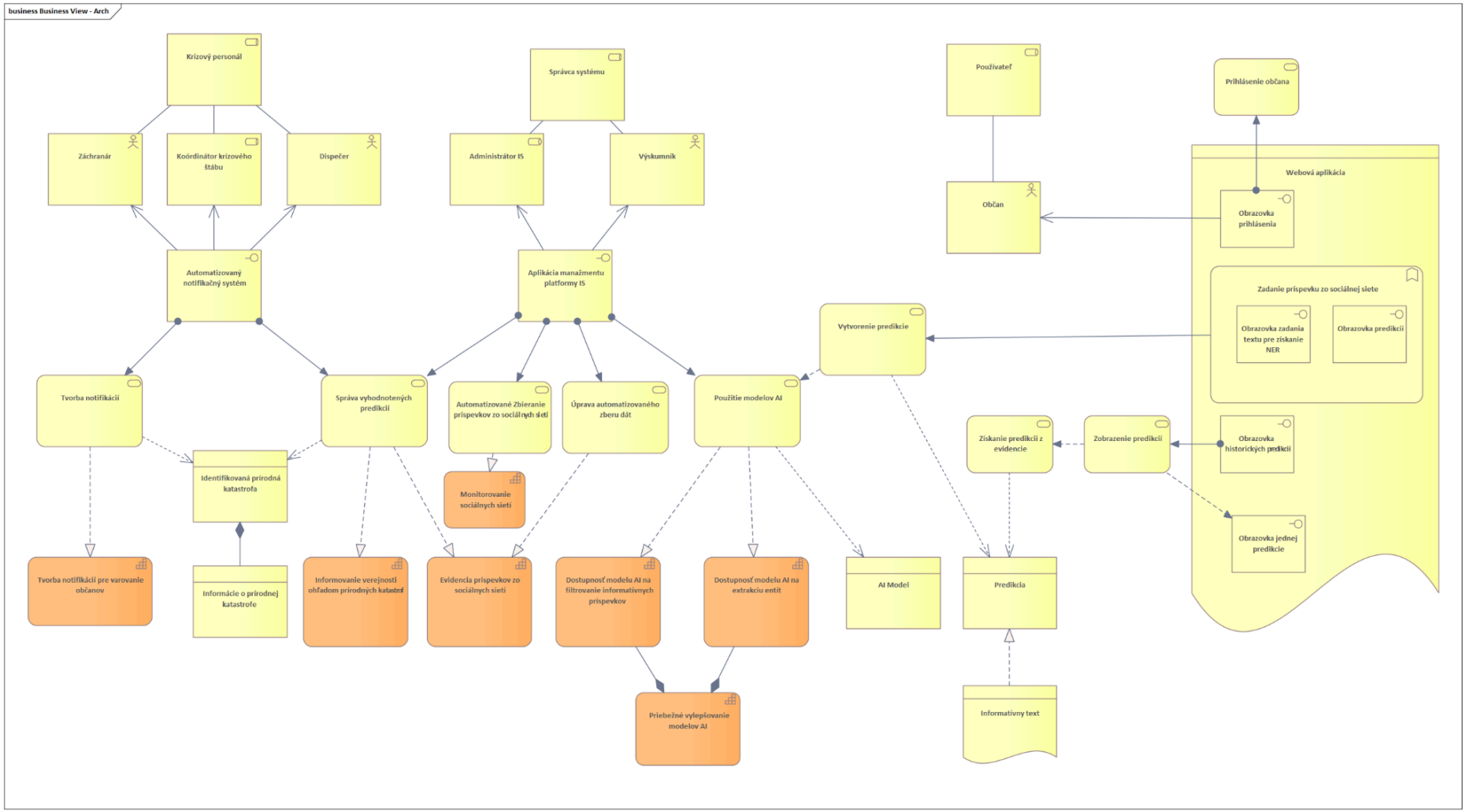
1. **Registrácia a prihlásenie používateľa**
   - 1.1. Do systému sa môže zaregistrovať len používateľ s doménou, ktorá je na zozname povolených
   - 1.2. Používateľ na prácu s verejnou webovou aplikáciou sa musí autentifikovať
2. **Zadanie príspevku zo sociálnych sietí a vytvorenie predikcie**
   - 2.1. Služba musí umožňovať vytvoriť novú predikciu na zadaný text
   - 2.2. Zadaný text musí byť vyhodnotení na základe dvoch prípadov: Informatívny text, ktorý sa vzťahuje a má informatívnu hodnotu k prírodnej katastrofe a bude ďalej analyzovaný. Neinformatívny text, s ktorým sa už nebude ďalej pracovať
3. **Extrakcia pomenovaných entít z informatívneho príspevku**
   - 3.1. Ak je príspekov informatívny extrahujú sa z neho pomenované entity ako lokalita, čas, dátum, miesto, organizácie, názvy udalostí a ostatné entity
4. **Obohatenie krátkych textov v podobe extrakcie a navštívenia URL adries z informatívneho príspevku**
   - 4.1. Pokiaľ príspevok obsahuje URL adresy, je ich nutné navštíviť a získať hlavnú časť textu, z ktorého budú získané ďalšie entity rovnako ako v požiadavke 3.
5. **Uloženie predikcií do perzistentného úložiska**
   - 5.1. Vytvorené predikcie musia byť priebežne ukladané do dostupného úložiska
6. **Zobrazenie historických predikcií na vyžiadanie**
   - 6.1. Vytvorené predikcie musí webová aplikácia vedieť spätne zobraziť spolu s entitami a aj stiahnutými webovými lokalitami
7. **Automatické získavanie spätnej väzby od používateľa**
   - 7.1. Po vytvorení predikcie webová aplikácia musí umožňovať ohodnotenie predikcie používateľom.
   - 7.2. Možnosti sú dve: Pozitívna reakcia ak systém vykonal správnu predikciu a negatívna ak bola predikcia nesprávna
8. **Automatické sťahovanie príspevkov zo sociálnych sietí a vytváranie predikcií**
   - 8.1. Systém musí umožňovať zadanie zoznamu sociálnych sietí, kanálov a stránok z ktorých sa priebežne budú sťahovať a klasifikovať príspevky
9. **Identifikácia prírodných katastrof zo sociálnych príspevkov**
   - 9.1. Systém musí dokázať vyhodnotiť a zlúčiť identifikované prírodné katastrofy.
   - 9.2. Po zlúčení katastrof systém odošle notifikáciu a získané informácie dostupným jednotkám krízového manažmentu

3. Architektúra

3.1. Business View

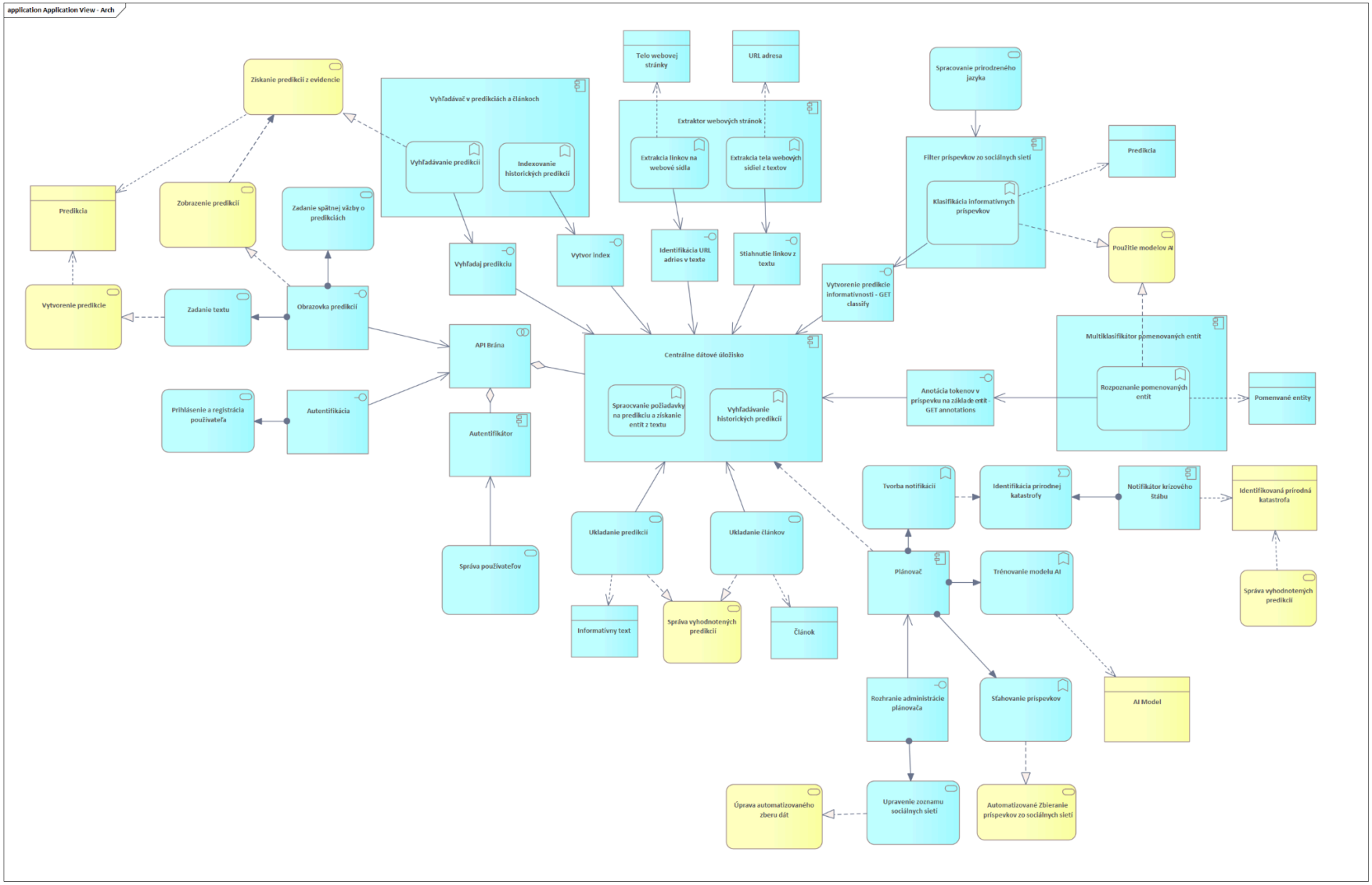
Biznis architektúra je previazaná s tokom hodnôt, ktorý sme opísali vyššie. Vidíme naviazanie jednotlivých biznis služieb na danú kroky v sekvencií a k nim pridružených aktorov a stakeholderov. Diagram je rozdelený do 3 nosných podcelkov systému. Rovnako ako to bolo pri motivácií zameriavame sa na tri roly, ktoré majú vplyv na úspešnosť systému.

Ide o: 1. Krízový personál - záchranárske jednotky a koordinátori, ktorým sú priamo poskytované informácie a následne robia kritické rozhodnutia, z ktorých ďalej občania môžu mať úžitok. 2. Správcovia systému, títo aktori nie sú priamo zainteresovaný do prebiehajúcej katastrofy, ale záleží im na tom aby mohli manažovať navrhnutý systém a priebežne ho vylepšovať. Systém manažmentu svojou podstatou definuje správanie výslednej aplikácie, ktorá posiela notifikácie krízovému personálu a pracuje s automatizovane získanými dátami. Záverečná tretia skupina je používateľ, ktorý používa a testuje výsledný systém prostredníctvom webovej aplikácie. V tomto ponímaní ide o bežného občana, ktorý chce pridať a ohodnotiť im v reálnom čase napísaný príspevkov. Tu je kritické aby systém ho mohol prihlásiť, a následne umožnoval zadanie textu. Tento text sa ďalej spracuje a vytvorí sa predikcia, na ktorú sa aplikuje model umelej inteligencie.
Reprezentáciou tejto predikcie je napríklad informatívny text, ktorý môže byť ďalej poslaný na analýzu a v prípade, že sa deteguje katastrofa jeho metadáta (lokalita, čas a podobne) vyslané ako notifikáciu a teda informáciu o prebiehajúcej katastrofe krízovému štábu.


business Business View - Arch
Krízový personál
Záchranár
Koordinátor krízového štábu
Dispečer
Správca systému
Administrátor IS
Výskumník
Používateľ
Občan
Prihlásenie občana
Webová aplikácia
Obrazovka prihlásenia
Automatizovaný notifikačný systém
Aplikácia manažmentu platformy IS
Zadanie príspevku zo sociálnej siete
Obrazovka zadania textu pre získanie NER
Obrazovka predikcií
Vytvorenie predikcie
Tvorba notifikácií
Správa vyhodnotených predikcií
Automatizované Zbieranie príspevkov zo sociálnych sietí
Úprava automatizovaného zberu dát
Použitie modelov AI
Získanie predikcií z evidencie
Zobrazenie predikcií
Obrazovka historických predikcií
Identifikovaná prírodná katastrofa
Monitorovanie sociálnych sietí
Obrazovka jednej predikcie
Tvorba notifikácií pre varovanie občanov
Informácie o prírodnej katastrofe
Informovanie verejnosti ohľadom prírodných katastrof
Evidencia príspevkov zo sociálnych sietí
Dostupnosť modelu AI na filtrovanie informatívnych príspevkov
Dostupnosť modelu AI na extrakciu entít
AI Model
Predikcia
Priebežné vylepšovanie modelov AI
Informatívny text

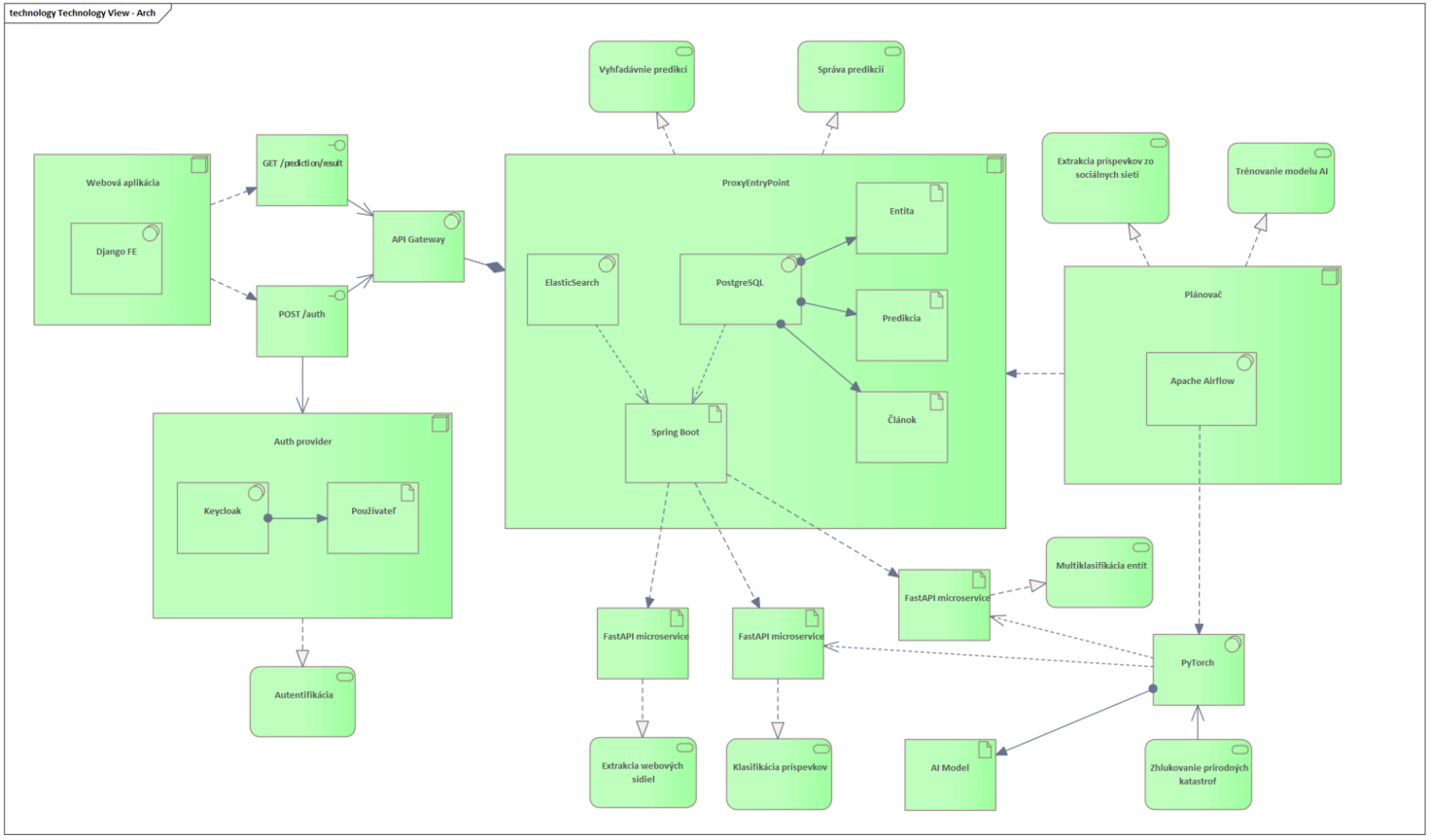
3.2. Application View

Aplikačná architektúra v našom projekte definuje kľúčové komponenty použité na splnenie vyššie definovaných požiadaviek. Jej naviazanie s biznis službami špecifikujú ich kooperáciu a nútne funkcie, ktoré je potrebné zabezpečiť aby boli realizované ciele z motivačného diagramu.
V navrhnutom systéme sa nachádza 9 aplikačných komponentov, ktoré zabezpečujú tok dát. Pomyselným vstupom je obrazovka prihlásenia. Komponent autentifikácie spravuje používateľov a je oddelenou jednotkou v rámci zvyšku systému. Po prihlásení používateľ má možnosť zadať text predikcie. Prvým vstupným bodom je API Gateway, ktorá je je kompozitným prvkom v rámci komponentu centrálneho dátového úložiska. Webová aplikácia, s ktorou používateľ interaguje pomocou jednotlivých obrazoviek nevie o tom, že za API gateway je ďalších 5 služieb, ktoré spolupracujú na získanie finálneho výsledku. Centrálne dátové úložisko plní dve funkcie a to ukladanie doručených dátových objektov od ďalších komponentov, ale aj vyhľadávanie už vytvorených predikcií a ďalších asociovaných objektov. Na vyhľadávanie využíva špecializovaný komponent, ktorý predikcie indexuje a zrýchľuje vyhľadávanie.
Po splnení týchto funkcií a sprostredkovaní daných aplikačných služieb cez rozhrania môžeme vyhlásiť že plní biznis službu správy vyhodnotených predikcií a môže byť použité komponentami zameranými na spracovanie existujúcich predikcií. Tieto automatizované analytické komponenty sa nepodielajú na doručení výsledku používateľovi na webovej stránke v reálnom čase, ale ich cieľom je spracovať vytvorené dátové objekty. Pokiaľ je nových predikcií dostatok, môžu sa buď oanotovať manuálne alebo využiť službu zadania spätnej väzby vo webovej aplikácií. Ak by sme mali už aj dostatok označených nových predikcií je možné pretrénovať modely AI, ktoré robia predikcie a zlepšiť tým efektívnosť systému. V prípade, že model označí viacero príspevkov ako informatívnych, tento komponent taktiež vykoná kontrolu pomocou tzv. klastrovania, identifikácie a verifikácie potenciálne prebiehajúcich katastrof a odošle identifikovanú prírodnú katastrofu notifikačnému komponentu.
Ak sa vrátime k tomu ako predikcie vznikajú môžeme si všimnúť tri dôležité komponenty v systéme. Sú to súčasti, ktoré pridávajú všetky metadáta k predikcií. Bez samotného vyhodnotenia či text je informatívny nerobíme žiadne ďalšie kroky. Ak nie je končíme s analýzou a ak je posúvame text ďalej. Túto predikciu zabezpečuje “Filter príspevkov zo sociálnych sietí”, ktorý obsahuje prvý AI model - klasifikátor. Následne sa z informatívneho príspevku pomocou “multiklasifikátora pomenovaných entít” vyextrahujú pomenované entity (lokalita, čas, miesto dátum a ostatné názvy). Toto je prvé obohatenie samotného textu. Posledným je kontrola či obsahuje URL adresy. URL adresy a webové sídla zahrnuté v tele príspevku obohacujú svojim rozsiahlym obsahom krátky text, ktorý používateľ môže na sociálnu sieť napísať. Ak príspevok obsahuje aspoň jednu adresu “extraktor webových stránok” ju navštívi a nájde na nej najväčšie telo textu, ktorý obsahuje najviac stopslov (slová, ktorá spájajú vety (spojky, predložky a pod). Keďže predpokladáme, že aj tento článok môže obsahovať ďalšie pomenované entity, tok dát pokračuje a pre každú extrahovanú URL adresu sa vytvorí ďalšia požiadavka na rozpoznávanie entít (NER). Priebežne sa tieto metadáta ukladajú a používateľ si ich môže zobraziť a ak je spokojný ohodnotiť kladne, alebo negatívne.


application Application View - Arch
Získanie predikcií z evidencie
Vyhľadávač v predikciách a článkoch
Vyhľadávanie predikcií
Indexovanie historických predikcií
Telo webovej stránky
URL adresa
Extraktor webových stránok
Extrakcia linkov na webové sídla
Extrakcia tela webových sídiel z textov
Spracovanie prirodzeného jazyka
Filter príspevkov zo sociálnych sietí
Klasifikácia informatívnych príspevkov
Predikcia
Použitie modelov AI
Predikcia
Zobrazenie predikcií
Zadanie spätnej väzby o predikciách
Vyhľadaj predikciu
Vytvor index
Identifikácia URL adries v texte
Stiahnutie linkov z textu
Vytvorenie predikcie informatívnosti - GET classify
Vytvorenie predikcie
Zadanie textu
Obrazovka predikcií
API Brána
Centrálne dátové úložisko
Spracovanie požiadavky na predikciu a získanie entít z textu
Vyhľadávanie historických predikcií
Anotácia tokenov v príspevku na základe entít - GET annotations
Multiklasifikátor pomenovaných entít
Rozpoznanie pomenovaných entít
Pomenované entity
Prihlásenie a registrácia používateľa
Autentifikácia
Autentifikátor
Tvorba notifikácií
Identifikácia prírodnej katastrofy
Notifikátor krízového štábu
Identifikovaná prírodná katastrofa
Ukladanie predikcií
Ukladanie článkov
Správa používateľov
Plánovač
Trénovanie modelu AI
Správa vyhodnotených predikcií
Informatívny text
Správa vyhodnotených predikcií
Článok
Rozhranie administrácie plánovača
Sťahovanie príspevkov
AI Model
Úprava automatizovaného zberu dát
Upravenie zoznamu sociálnych sietí
Automatizované Zbieranie príspevkov zo sociálnych sietí

3.3. Technology View

V rámci architektúry ponúkame aj pohľad na samostatný technologický diagram. Ako bolo spomenuté, vidíme jednotlivé komponenty realizované pomocou služieb a artifaktov. Webová aplikácia komunikuje s API Gateway, centrálne úložisko využíva ElasticSearch na vyhľadávanie a PostgreSQL na ukladanie dátových objektov. Jednotlivé komponenty na obohatenie predikcií sú implementované v jazyku Python a ich najdôležitejšou kapabilitou je komunikácia s AI modelmi pomocou frameworku PyTorch. Automatizované funkcie, ako detekcia, či notifikovanie sú riadené pomocou orechstrátora Apache Airflow.



4. Implementácia

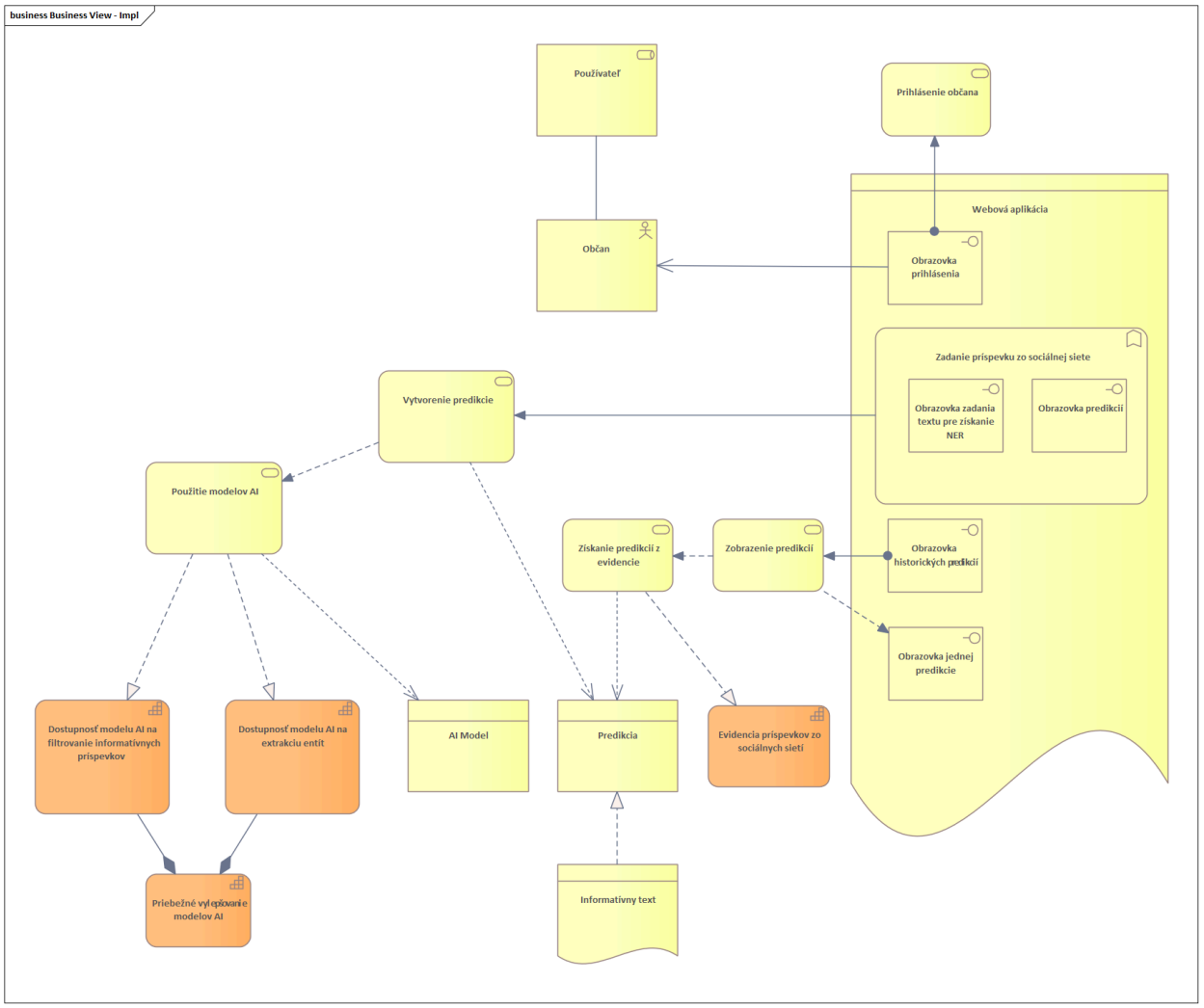
V ďalšej časti projektu, môžeme prejsť k implementácií konceptu riešenia. Na rozdiel od návrhu architektúry budeme sa zaoberať len používateľskou časťou, ktorá obsahuje webovú aplikáciu. Od používateľa zozbierame predikcie, extrahujeme entity a navštívime prípadné URL adresy. Predikcie uložíme a na požiadanie používateľovi zobrazíme historické predikcie.
Implementáciu rozdelíme do troch kategórií:

1. Mikroslužby
2. Business Process Model and Notation (BPMN) - Camunda
3. Event-Driven Architecture (EDA - Kafka)

Tok hodnôt z návrhu ostane bez zmeny. Naše spôsobilosti zostávajú rovnaké a proces krokov v sekvencií sa tiež nemení. Môžeme začať s pohľadom na biznis v Bussiness View.

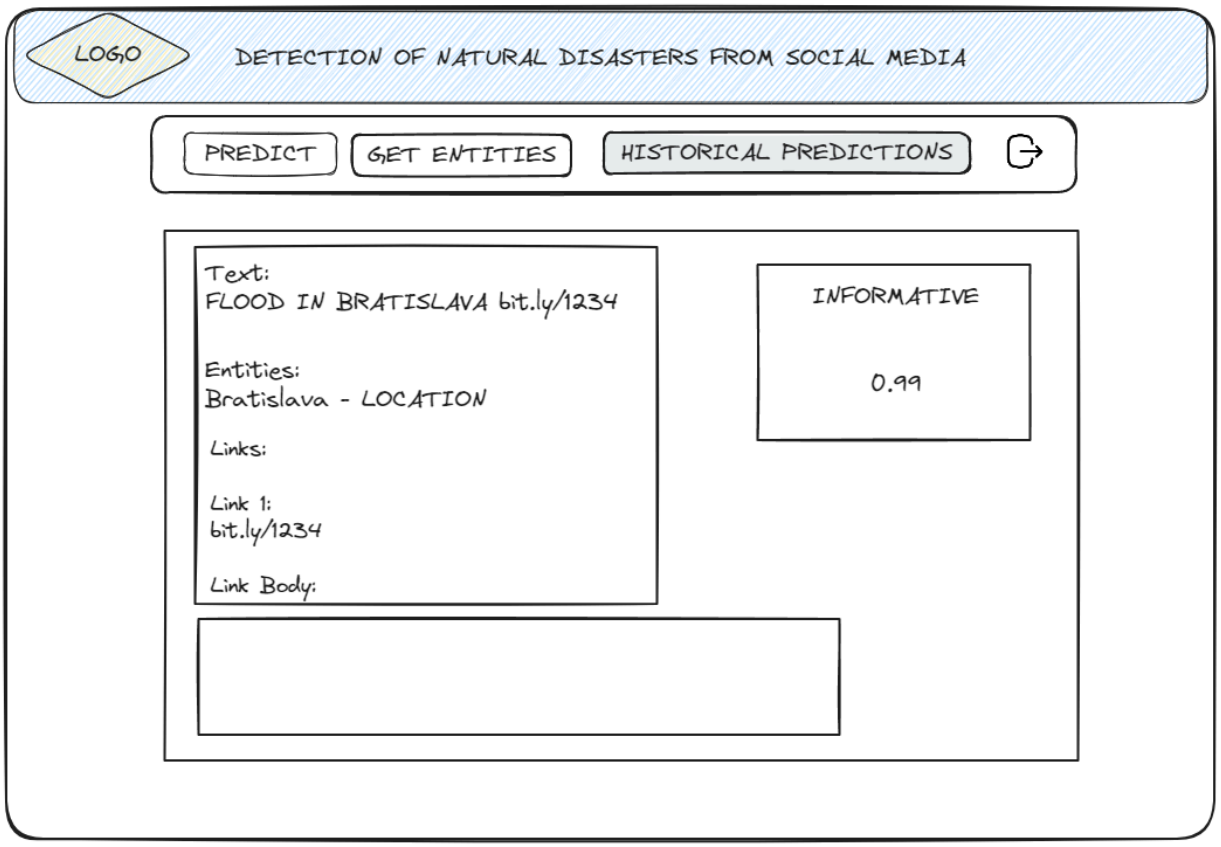
4.1. Business View

Biznis pohľad je pre všetky tri podkategórie implementácie rovnaký. Všimnime si, že v porovnaní s návrhom nám zostal už iba pravý podstrom. Samotná webová aplikácia reprezentuje biznis objekt, ktorá má v sebe obrazovky. Tieto obrazovky si bližšie predstavíme v sekcii 4.2, alna teraz ide o vstupné body interakcie s používateľom - občanom. Vidíme, že obrazovky predikcií sú zahrnuté vo funkcii zadania príspevku zo sociálnej siete, ktorá slúži na splnenie služby vytvorenia predikcie. Ostatné obrazovky majú podobný význam a pristupujú k biznis objektom Predikcia s jej ostatnými metadátami. Vytvorenie predikcie je prepojené so službou použitia modelov AI, ktorá pristupuje k samotným natrénovaný modelom umelej inteligencie, tiež biznis objektu. Ako sme spomínali už pri návrhu tieto služby realizujú spôsobilosti nášho systému. Na pripomenutie, ide o vytvorenie predikcií, vylepšovanie modelov a spravovanie príspevkov zo sociálnych sietí.

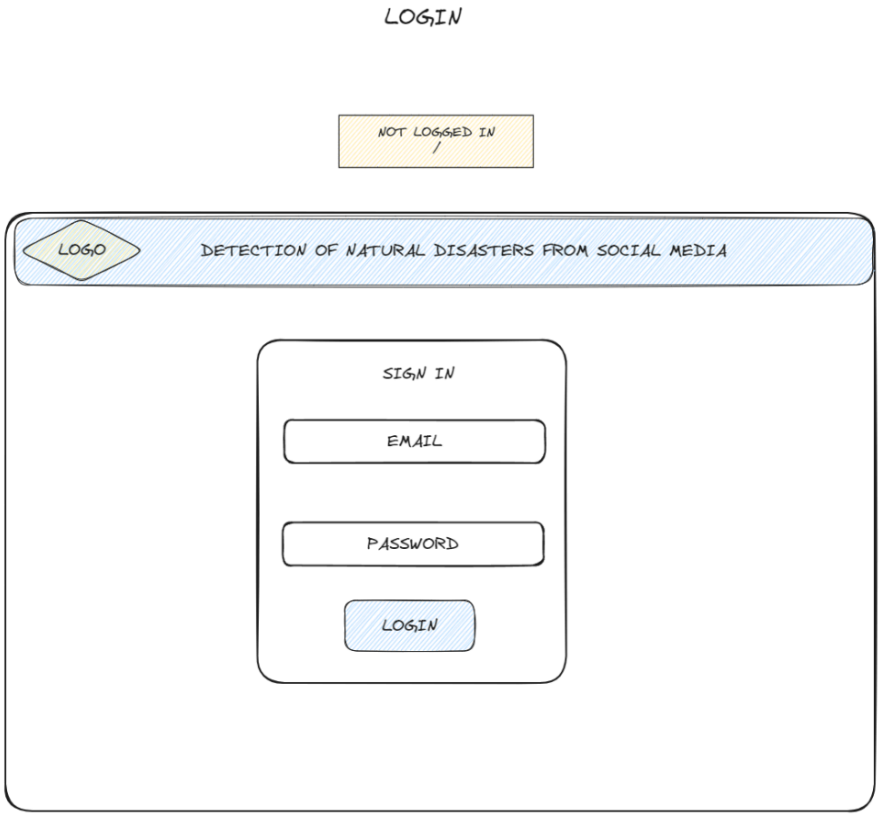


4.2. Návrh obrazoviek - Wireframe

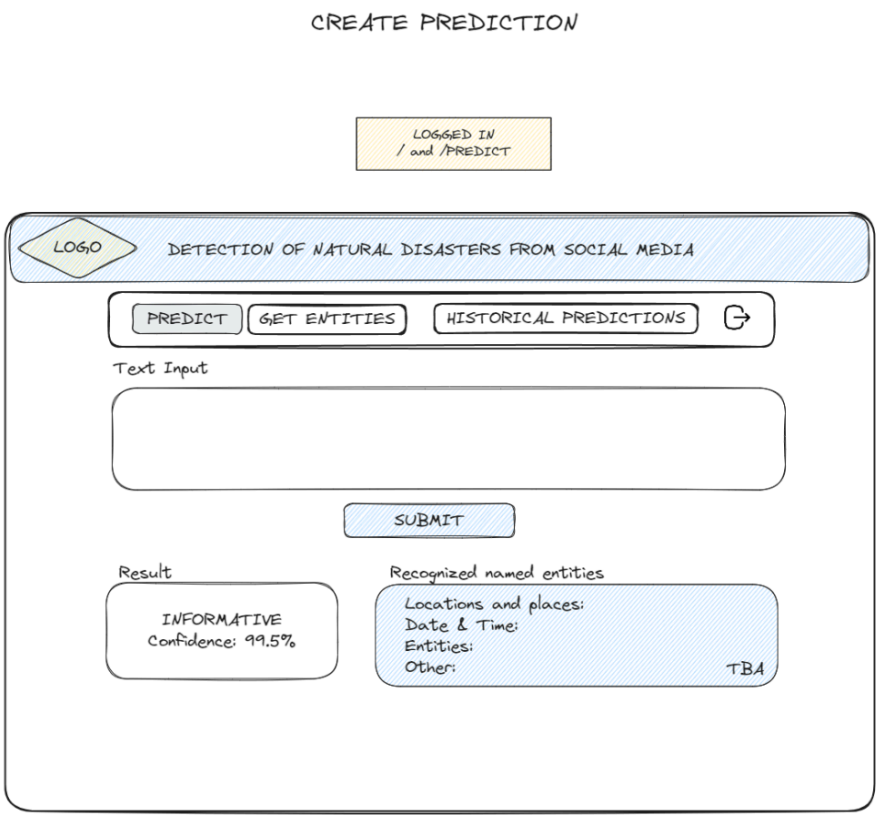
V momente keď máme predstavené všetky súčasti, ktoré budeme implementovať, môžeme navrhnúť obrazovky vo forme wireframe pre webovú aplikáciu, s ktorou používateľ bude interagovať na zadanie a prehliadanie vytvorených predikcií.
Celkovo ide o 5 obrazoviek. Prvá obrazovka opisuje stav prihlásenia do systému. Použitie AI modelov si vyžaduje pridelenie určitých zdrojov prostriedkov a preto chceme obmedziť to komu dovolíme ich vytvárať. Po prihlásení je používateľ následne presmerovaný na hlavnú obrazovku zadania predikcií, ktorá obsahuje textové pole a tlačidlo odoslať. Na rovnakej stránke sa po odoslaní zobrazí výsledná predikcia a môže si ju otvoriť do detailu spolu s vytvorenými entitami a potenciálne stiahnutými článkami, ktoré sú k nej priradené. Na zobrazenie predchádzajúcich vytvorených predikcií existuje samostatná obrazovka s prehľadnou tabuľkou, ktorá umožňuje zoradenie a vyhľadávanie. Táto obrazovka priamo neobsahuje entity, alebo články ale iba ich počet aby nebol používateľ ohromený počtom informácií na stránke. V neposlednom rade môže prechádzať stránky (pomocou paginácie) a zadefinovať počet zobrazených predikcií na jednu stranu.



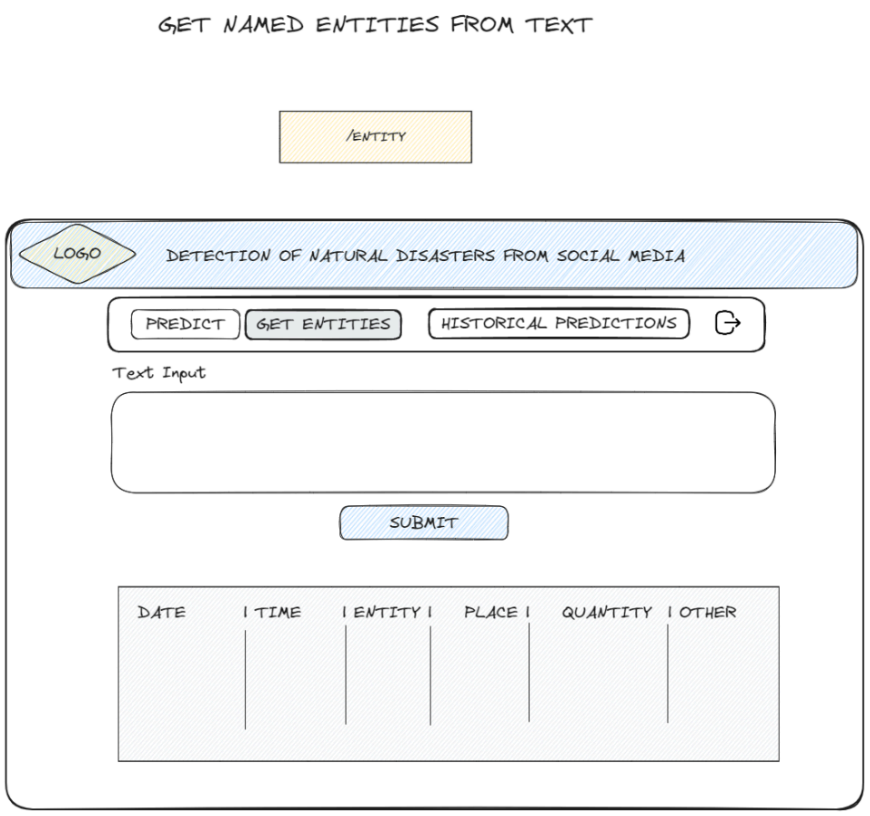
Obrazovka detailneho zobrazenia predikcie



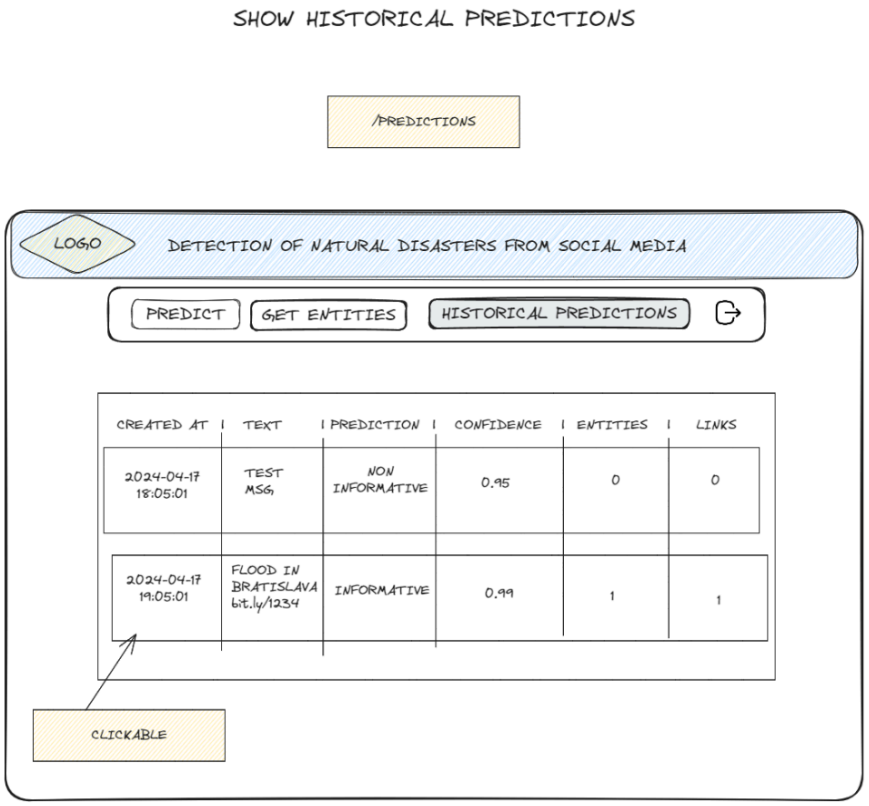
Obrazovka prihlásenia



Obrazovka tvorby predikcií



Obrazovka zobrazenia pomenovaných entít



Obrazovka zobrazenia historických predikcií

4.3. Fyzický DB model

Na interakciu s databázou a jednotlivými dátovými objektmi využijeme vo všetkých troch kategóriách implementácie relačnú databázu. Je vhodné si na úvod zadefinovať závislosti a atribúty dátových objektov, ktoré sme si predstavili v aplikačnej architektúre. V našom prípade opäť ide len o podmnožinu a zameriame sa na Predikcie, entity, linky (stiahnuté články) a používateľov. Vzťahy a cudzie kľúče vidíme na fyzickom diagrame nižšie. Pre pochopenie uvádzame krátke vysvetlenie jednotlivých stĺpcov. Tabuľky majú spoločné hodnoty pre dátum a posledné úpravu. Poslednou tabuľkou sú používatelia, ktorých len preberáme z KeyCloak SSO softvéru (viac v kapitole o mikroslužbách).

predikcie:

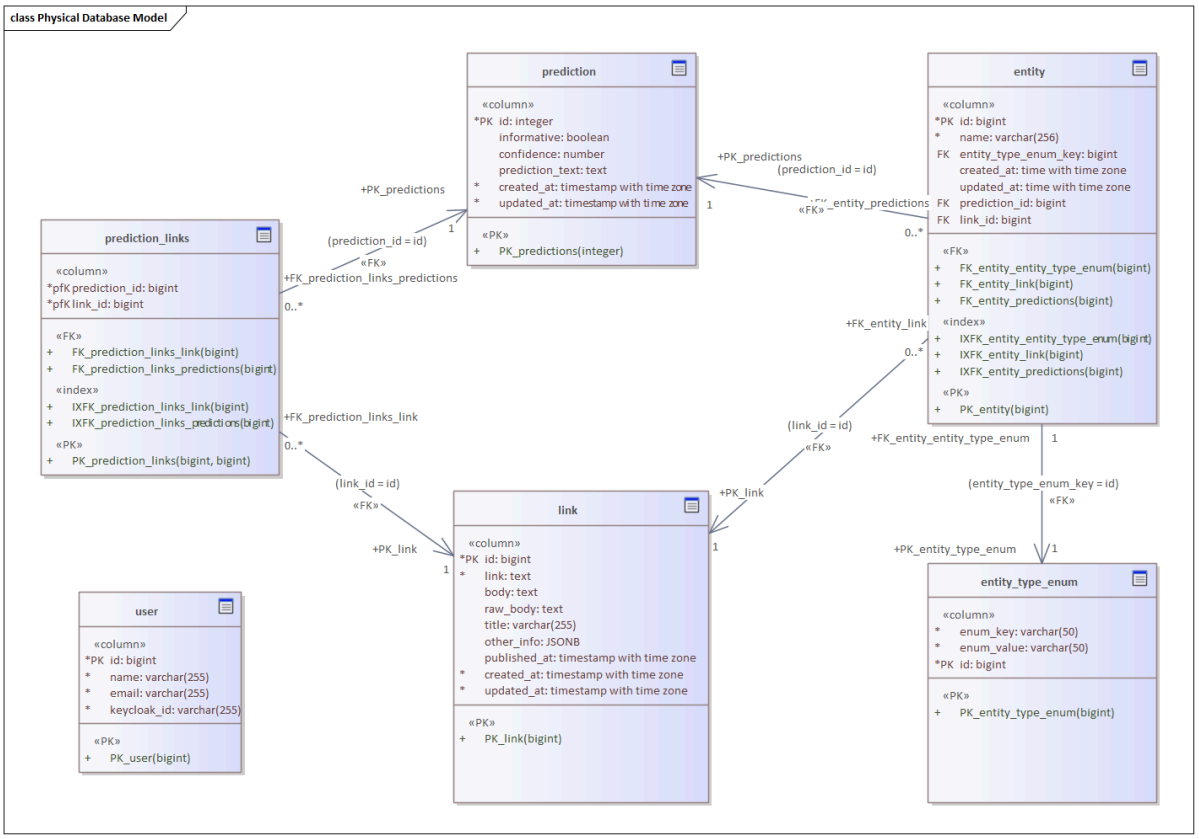
- informative - boolean hodnota - definuje či je text predikcie informatívny - vzťahuje sa na prírodnú katastrofu
- confidence - float hodnota od [0-1] - istota z akou model vytvovril predikciu
- prediction_text - text patriaci sociálnemu príspevku

entity

- id predikcie, ku ktorej patrí a enum hodnota pomenovanej entity - jedno z DÁTUM, LOKALITA, ČAS, ENTITA (osoba), INÉ

link

- detaily o článku ako dátum publikácie, raw HTML text, vyčistený text, odkaz naň a všetky ostatné detaily v podobe JSON.



4.4. Mikroslužby

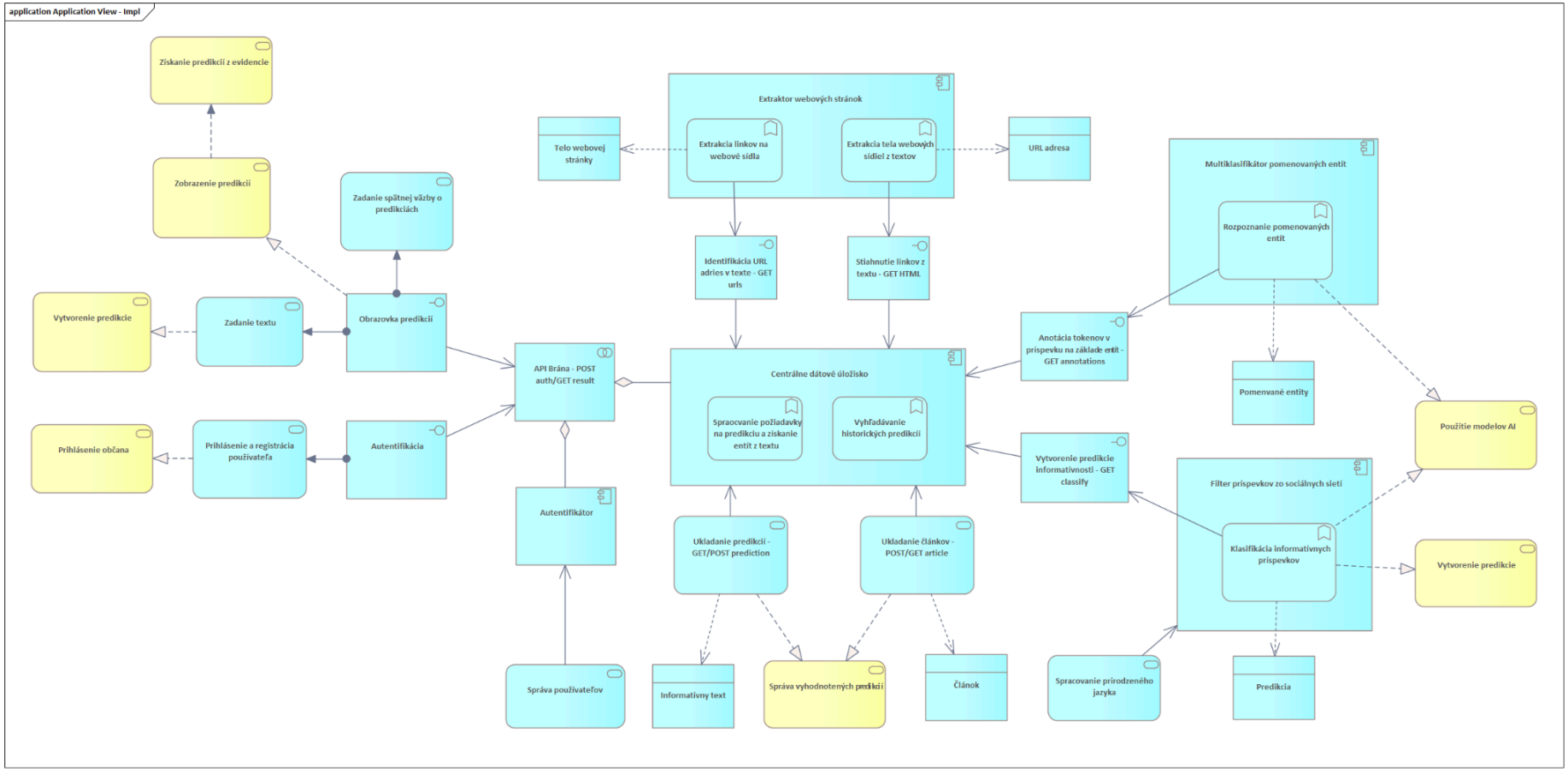
Prvým typom, ktorý sme implementovali sú mikroslužby. Tento prístup prináša rad výhod vrátane väčšej flexibilnosti, škálovateľnosti a jednoduchšej údržby systému. V kontexte systému na detekciu prírodných katastrof umožňujú mikroslužby efektívne spracovávať a analyzovať obrovské objemy dát získaných zo sociálnych sietí v reálnom čase.
V našej aplikácií budeme uvažovať o nasledujúcich mikroslužbách:

1. **Autentifikačný komponent - Keycloak - COTS riešenie - (Java)**
   Namiesto implementácie vlastnej registrácie a riešenia problému s citlivými používateľskými dátami, prenecháme riadenie aplikácií vytvorenej priamo na to. Keycloak taktiež podporuje mnoho identity providerov ako Google, GitHub, Microsoft a podobne. Keby chceme v budúcnosti povoliť registráciu cez jedného z týchto IDP bolo by to oveľa jednoduchšie ako implementovať všetko od začiatku. Po prihlásení pomocou používateľského mena a hesla, keycloak vygeneruje JWT token, tento JWT token sa použije pri požiadavkách na ďalšie služby
2. **Webová aplikácia - Frontend - Django Fullstack - Python**
   Zobrazuje interaktívne grafické používateľské rozhranie. Komunikuje s API Gateway a prostredníctvom požiadaviek zadáva dotazy na ďalšie služby - prihlásenie, vytvorenie novej predikcie, zobrazenie existujúcej, alebo zoznam historických predikcií.
3. **Centrálne dátové úložisko - Spring Boot - Java**
   Slúži na orchestráciu požiadaviek z webovej aplikácie. Obstaráva hlavný tok informácií. Po prijatí požiadavky z FE vytvorí predikciu a rozpošle dotazy o získanie metadát od ďalších mikroslužieb. Ukladá predikcie a všetky ostatné dátové objekty do PostgreSQL databázy a taktiež obsahuje upravenú implementáciu API Gateway.
4. **Relačná databáza - PostgreSQL - COTS riešenie - (C)**
   Zodpovedná za perzistenciu údajov v systéme. Obsahuje tabuľky a indexy na základe sekcie “fyzický DB model”.
5. **Filter príspevkov zo sociálnych sietí - FastAPI - Python**
   Prvá sady python mikroslužieb. Vystavuje 3 endpointy na spracovanie prirodzeného textu, tokenizáciu a vytvorenie samotnej predikcie. Obsahuje AI model na klasifikáciu textu. AI model vracia hodnotu istoty pre každú predikciu a binárnu hodnotu či je príspevok informatívny
6. **Multiklasifikátor pomenovaných entít - FastAPI - Python**
   Druhá Python služba, zodpovedná za extrakciu entít z textu. Všetky slová zo sociálneho príspevku sú klasifikované a ak sa v nich nájde entita, je pridaná do odpovede. Disponuje možnosťou vrátiť HTML text, ktorý označí entity v zadanom texte.
7. **Extraktor webových stránok - FastAPI - Python**
   Obsahuje 2 endpointy, prvý na získanie URL adries z textu a druhý na stiahnutie zadaného webového sídla z internetu. Používa knižnicu newspaper3k, ktorá získa telo článku a pridá k nemu dodatočné metadáta.

Všetky mikroslužby a asociované systémy sú implementované pomocou Docker Image a lokálne orechstrované systémom docker-compose na jednoduchú komunikáciu cez lokálnu sieť a prípadné perzistovanie dát.

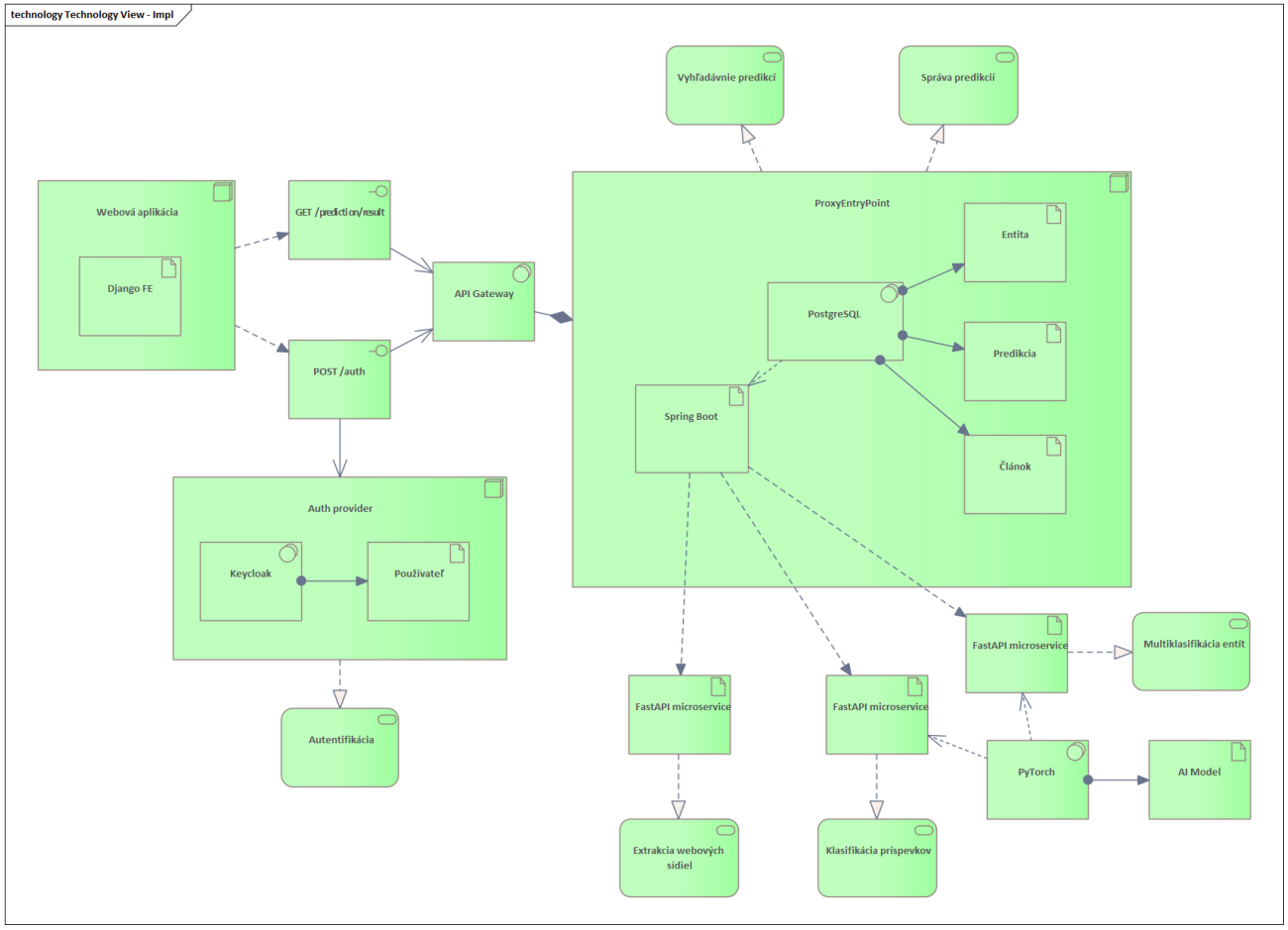
4.4.1. Application view

V aplikačnom pohľade sme oporti pôvodnému návrhu odstránili, súčasti o automatizácií pretrénovania, notifikačnom komponente a komplexnom nástroji na indexovanie a vyhľadávanie v predikciách. Podstata toku dát, funkcie komponentov a služieb zostávajú rovnaké. Vo všetkých ďalších aplikačných diagramoch, keď budeme hovoriť o niektorom z týchto komponentov, vždy uvažujeme o celom komponente spolu s jeho funkciami, rozhraním a službami. Pre prehľadnosť to nemusí byť zobrazené v ďalších pohľadoch.



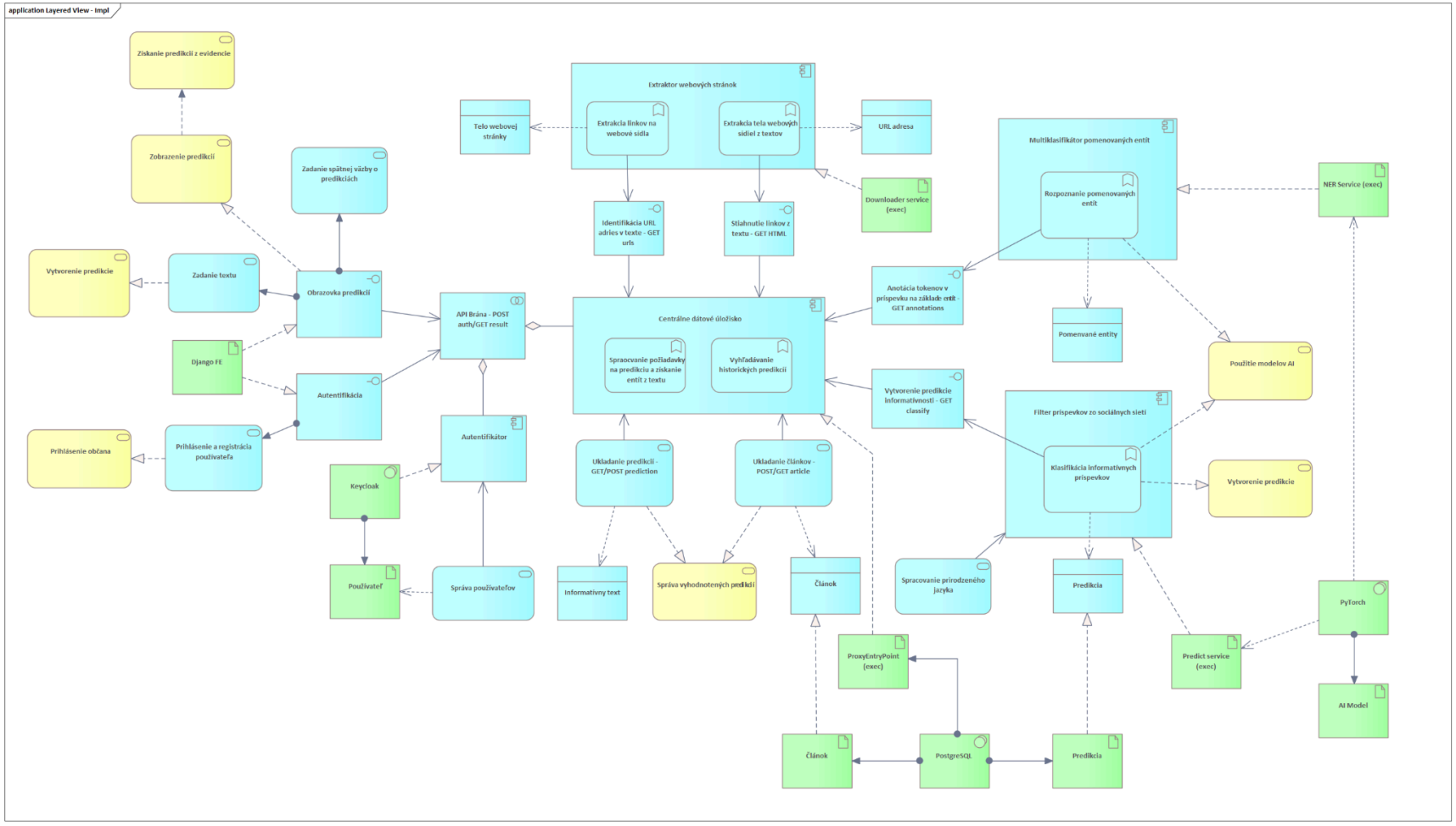
4.4.2. Technology view

V rámci technologického pohľadu vidíme konkrétne využitie softvérových služieb a artifakty mikroslužieb, ktoré sme implementovali. Odstránili sme ElasticSearch na vyhľadávanie a indexovanie v predikciách a orchestrátor Apache Airflow. Framework na integráciu AI modelov v jazyku Python, ktorý py-mikroslužby používajú je PyTorch. Webová aplikácia pristupuje k centrálnemu dátovému úložisku v tomto diagrame navzané “proxyentrypoint” (orchestrátor) cez API gateway. V požiadavkách zahŕňa JWT token, čím autentifikuje používateľa.



4.4.3. Layered View

Pohľad na vrstvy zjednocuje predchádzajúce dva pohľady a prepája jednotlivé komponenty pomocou kooperácie medzi biznis, aplikačnými a technologickými službami. Vidíme previazanie a od abstraktnej ku konkrétnejšie realizácií. Biznis scenár vytvorenia predikcie je realizovaný pomocou viacerých aplikačných komponentov, ktoré slúžia na splnenie cieľov kladených v motivačnom diagrame.



4.5. BPMN - Camunda

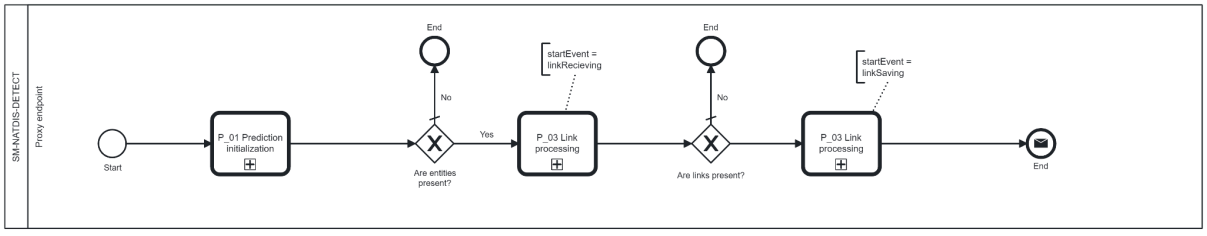
Druhý typ implementácie je realizovaný pomocou jazyka na modelovanie biznis procesov. Umožňuje jasné a jednoznačné vyjadrenie tokov, úloh a rozhodnutí, ktoré sa v systéme vykonávajú. Na spustenie BPMN procesov použijeme platformu Camunda, ktorá bude spravovať a riadiť jednotlivé procesy.
Camunda nahradí orchestrátor, ktorý sme v mikroslužbách museli samostatne implementovať. Procesy sme vytvorili v nástroji Camunda Modeler a to nasledovne:
Webová aplikácia zavolá jednoduché rozhranie na strane centrálneho dátového úložiska, ktoré iniciuje všetky procesy potrebné na dovŕšenie rovnakého cieľa a zozbieranie predikcií, metadát z príspevku na sociálnej sieti. Vystavili sme dodatočné endpointy na komunikáciu s podprocesmi v hlavnom BPMN procese s názvom “Main prediction flow”. Tieto endpointy slúžia na zapisovanie dát, ktoré sú získané z rovnakých mikroslužieb ako bolo opísané v predchádzajúcej sekcii. Namiesto priameho uloženia do databázy ich Camunda pošle na centrálne dátové úložisko, kde sa potom uložia a sú asynchrónne dostupné pre používateľa na zobrazenie. Taktiež si musíme uvedomiť, že bolo potrebné vykonať malé zmeny aj vo webovej aplikácií, keďže doteraz mikroslužby vždy nasledovali po sebe v sekvencií volanie bolo synchrónne. Používateľ počká chvíľu a zobrazil sa mu výsledok, no v tomto prípade sa procesy zapínajú asynchrónne a preto sme používateľovi oznámili pomocou notifikácie, čo sa práve deje a presmerovali ho na stránku s historickými predikciami, kde sa mu po neskôr výsledok zobrazí.
Toto riešenie je iné aj z toho hľadiska, že sú služby viac nezávislé na sebe. Doteraz existujúci orchestrátor implementovaný v Java Spring Boot, bol odľahčený a už neriadi celý proces komunikácie s py-mikroslužbami. Funguje už len na to, na čo bol navrhnutý, na ukladanie dát. Pre zjednodušenie a zachovanie určitého rozhrania ako to bolo aj pri mikroslužbách, bolo ponechané aby inicializoval procesy a sprostredkoval výstupy py-mikroslužieb pre Camundu. Webová aplikácia stále nevie o tom, že v pozadí je Camunda alebo niečo iné, posiela rovnaké dotazy len v tomto prípade na iný endpoint.

4.5.1. Camunda procesy

Implementovali sme 5 BPMN procesov spolu s jedným user taskom - ohodnotenie vytvorenej predikcie. Toto je záverečný proces, ktorý sa spustí, ked je predikcia uložená do databázy a obsahuje svoj vlastný formulár.
Procesy sú nasadené do “camunda engine”, kedy je ich možné spustiť a monitorovať.

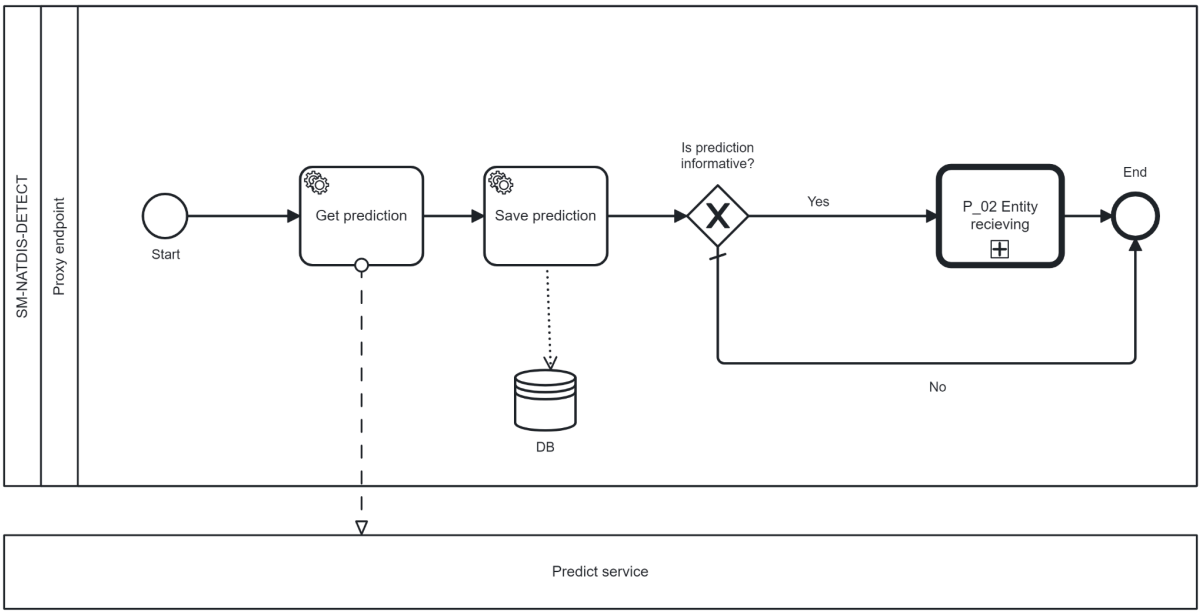
1. **Inicializácia hlavného toku**

Prvým zo spomenutých procesov je hlavný tok údajov, ktorý zahŕňa v sebe skupinu ostatných procesov. Môžeme vidieť, že po ukončení procesu inicializácie predikcie s označením 01, sa pokračuje ďalej. a ak je príspevok informatívny pokračuje sa so získaním počtu URL adries v texte a ak tam nejaké sú, ich stiahnutím



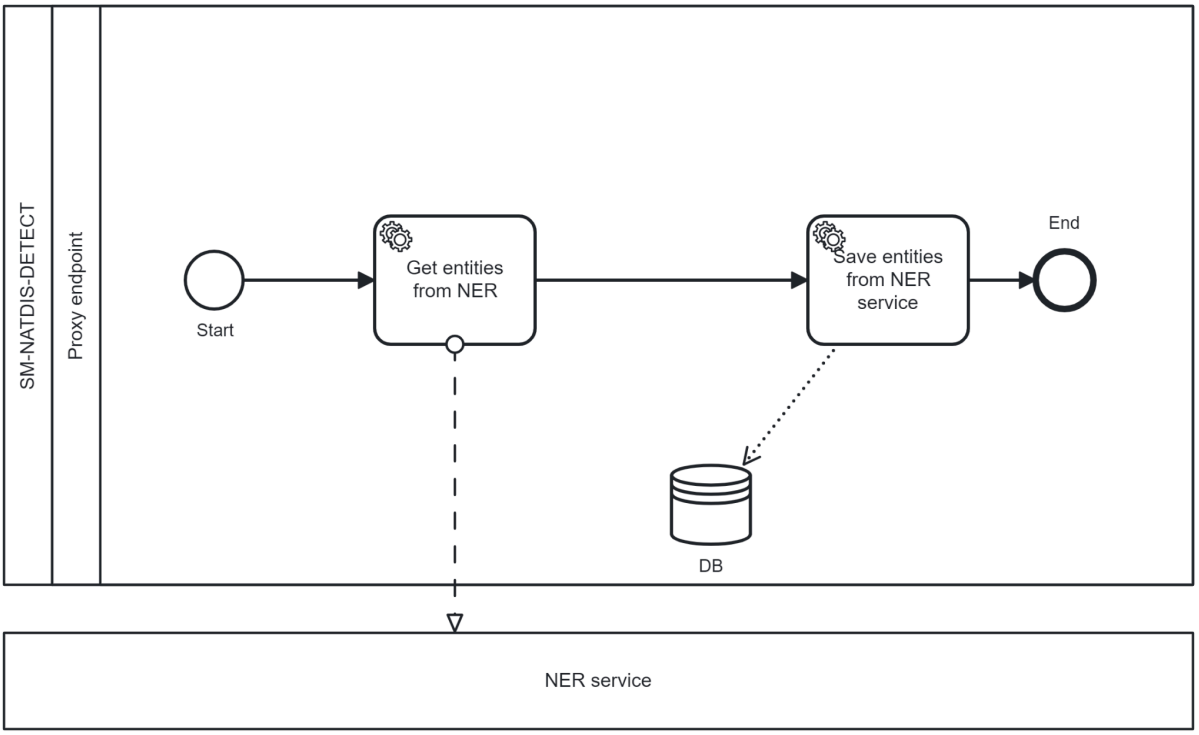
2. **Vytvorenie predikcie**

Inicializácie predikcie v sebe zahŕňa dopyt na predikčnú službu a uloženie predikcie do databázy. Ako sme už opísali vyššie Camunda komunikuje so službami a riadi tok dát. V tomto procese je taktiež ešte jeden vnorený proces extrakcie entít z textu sociálneho príspevku (id 0_2).



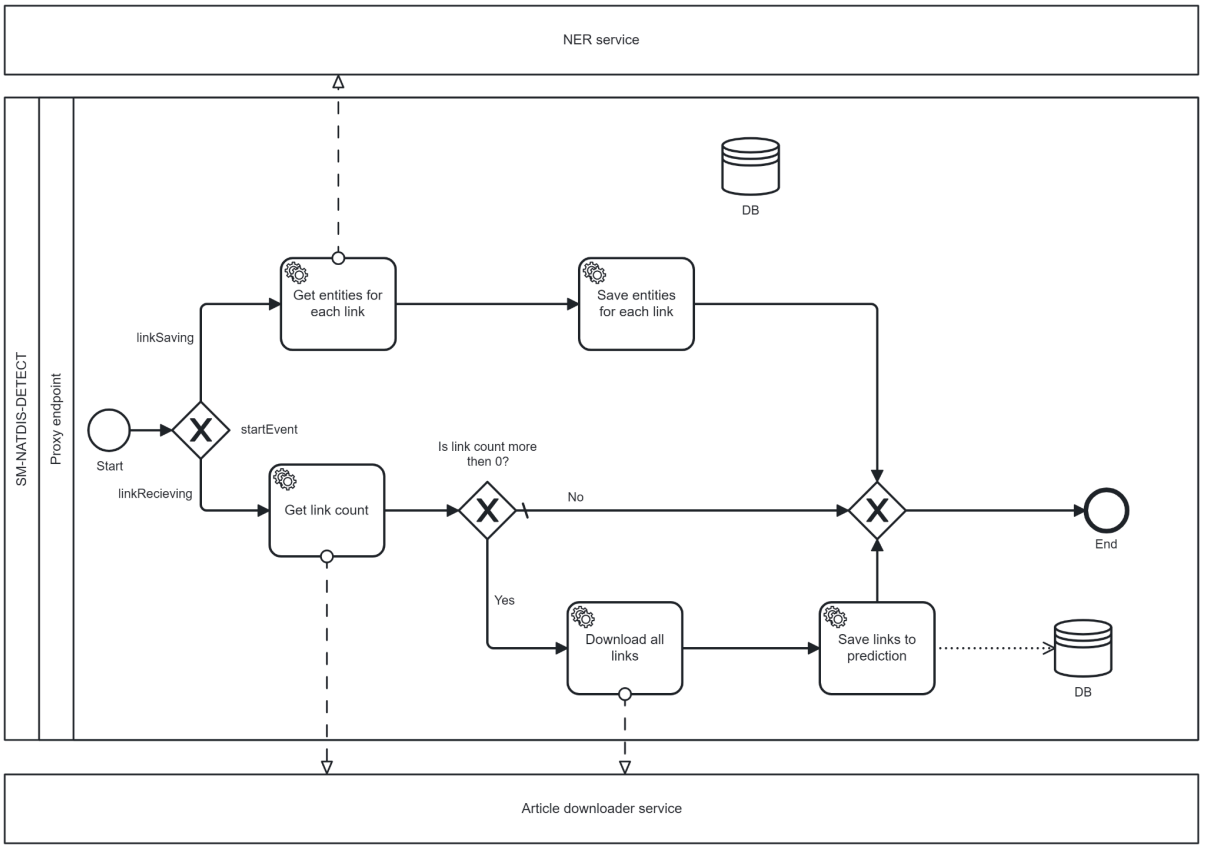
3. **Extrakcia entít z predikcie**

Proces extrakcie entít sme videli byť volaný procesom 0_1 a postup je veľmi podobný ako pre predchádzajúci proces.



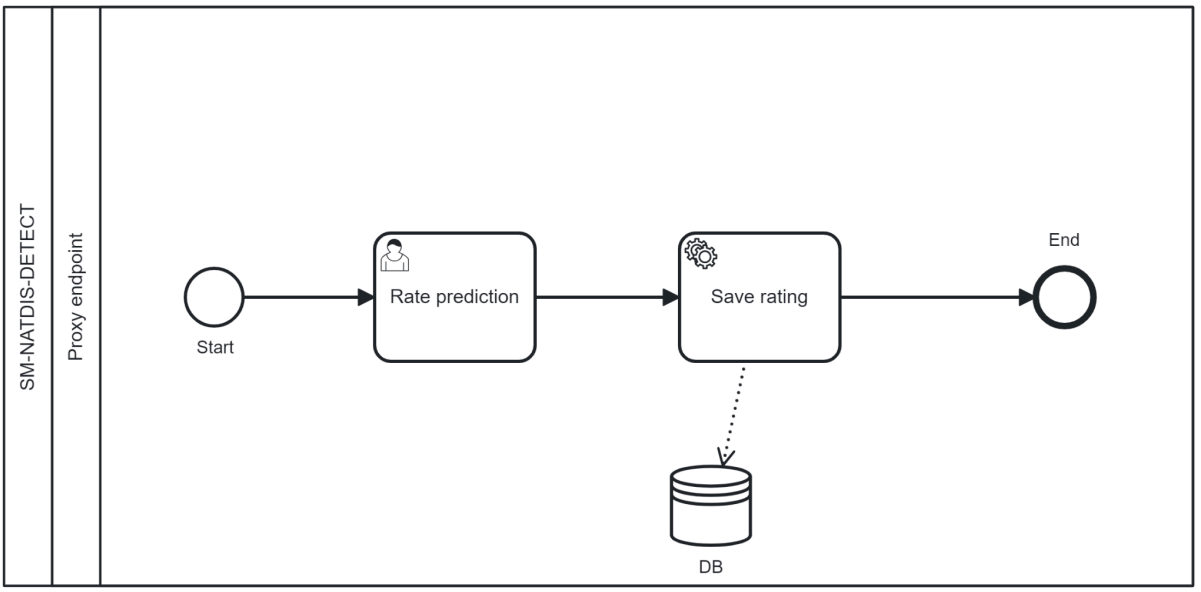
4. **Stiahnutie článkov a extrakcia entít z tela článkov**

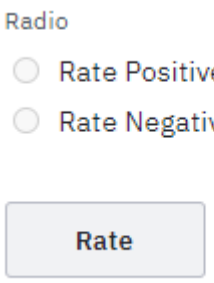
Následne, po ukončení procesu 02 a prípade že text je informatívny pokračujeme ďalej do pocesu 03. V tomto procese vidíme dve vetvy. V spodnej vetve je nutné získať počet URL adries, ak je nenulový aj ich stiahnuť. Tento proces sa opakuje s iným parametrom, keďže chceme zo stiahnutých článkov extrahovať aj entity. Na rozdiel od procesu 02 je na vstupe text článku. Ide o dve nezávislé volania.



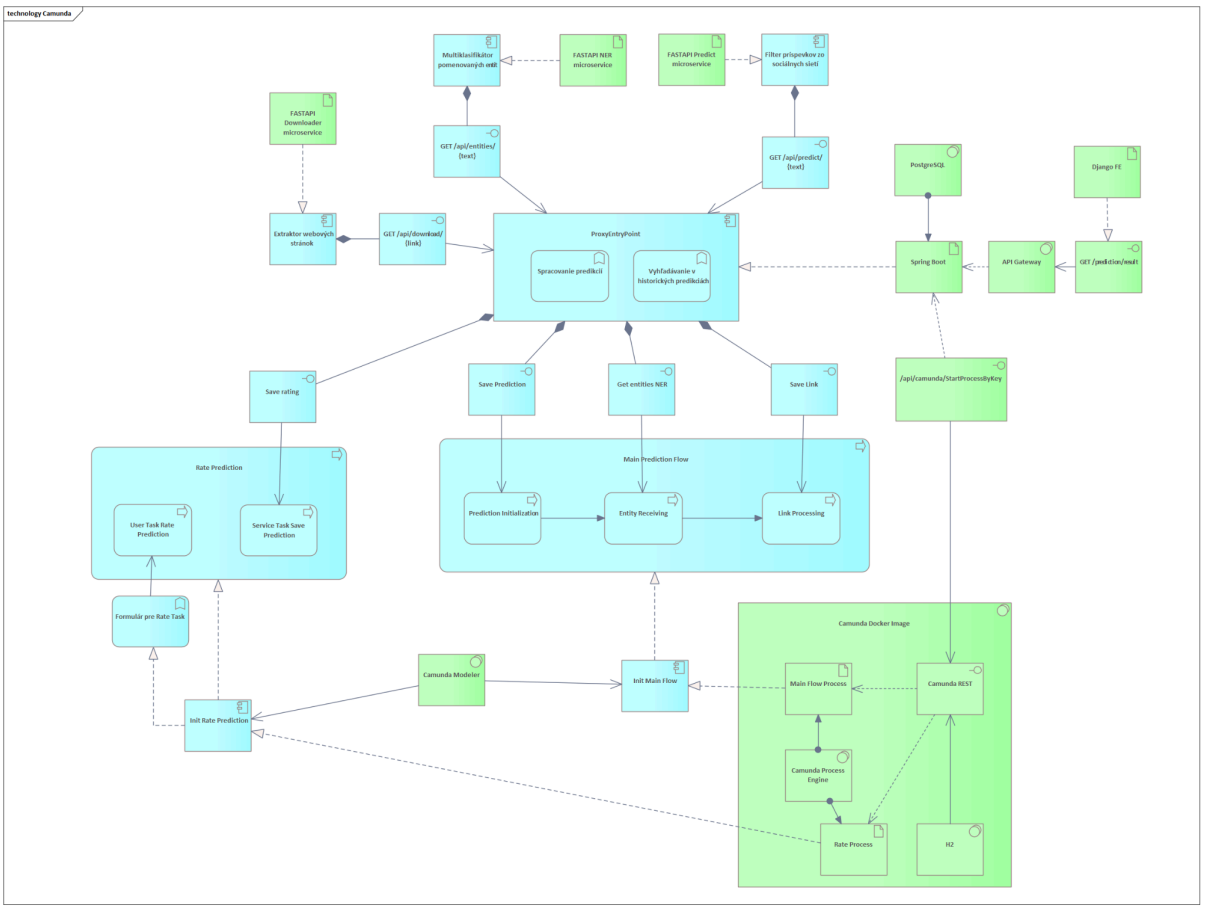
5. Ohodnotenie predikcií - spätná väzba od používateľa

Po uložení entít a článkov do databázy, spustí sa automaticky posledný proces: Ohodnotenie obsahuje úlohu na používateľa, ktorú je nutné vykonať pomocou formulára, ktorý je zobrazený pod telom procesu. Ide o jednoduchý výber pozitívnej alebo negatívnej spätnej väzby na základe výstupu AI modelu.





4.5.2. Architektúra pomocou Camunda



4.6. Event Driven Architecture - Kafka

Poslednou kategóriou implementácie je architektúra riadená udalosťami. V tomto prípade sme ako aplikáciu, ktorá bude spravovať jednotlivé udalosti v systéme zvolili Kafka. Kafka je platforma na distribuované spracovanie udalostí. Je navrhnutá na spracovanie veľkého objemu udalostí v reálnom čase a poskytuje robustné nástroje na publikovanie, odber a spracovanie udalostí. V Kafke sú udalosti publikované do tzv. topic-ov (tém). Budeme používať návrhový vzor Publisher/Subscriber, kedy vyprodukované správy ostanú v kafke aj po ich “skonzumovaní”. V rámci implementácie správy, ktoré posielame do jednotlivých topicov sú zaenkódované do base64.

4.6.1. Kafka topic

Pri snahe zjednodušiť implementáciu pomocou udalostí sme vytvorili nasledovné topicy.

1. **Prediction Topic**

Webová aplikácia publikuje správu s textom, ktorý je odoberaný predikčnou mikroslužbou. Predikčná mikroslužba asynchrónne odoberá daný topic a vykoná predikciou pomocou AI modelu. Predikciu publikuje do spoločného topicu pre databázovo-dátové objekty dbTopicsTopic

2. **Topic databázových objektov (DB objects topic)**

Centrálne dátové úložisko ako jediné tento topic odoberá, a objekty, ktoré doň prídu budú eventuálne zapísané do perzistentného úložiska (databázy). Všetky služby svoje vyprodukované dáta publikujú práve sem a v prípade potreby pridávajú k správam identifikátory aby bolo možné asociovať vytvorené objekty s existujúcimi vzťahmi v databáze. Po skonzumovaní správy o novej predikcií pošle do topicu NER správu o žiadosti na extrakciu entít z predikcie. K správe priloží vytvorené unikátne id predikcie

3. **NER topic**

Služba na extrakciu entít po prijatí správy extrahovať entity, tak aj vykoná a opäť rovnakým spôsobom publikuje do topicu databázových objektov dané informácie. Centrálne úložisko očakáva 3 typy správ - predikcie, entity a linky (ukážka nižšie). Entity musia obsahovať asociované id predikcie inak budú zahodené. Môžu alebo nemusia obsahovať aj link id - informáciu o tom, že patria k článku (URL adrese). Po zapísaní relácie medzi predikciou a entitami, zapíše správu s id predikciou do topicu na získanie URL adries a článkov

4. **Link Topic**

Ako to bolo aj pri predchádzajúcich dvoch službách tento topic odoberá služba na sťahovanie článkov. Po prijatí požiadavky na stiahnutie článkov z textu, ich okamžite aj stiahne. V prípade, že text neobsahuje žiadny ďalší link skončí. Stiahnuté články sú publikované naspäť do databázových objektov a uložené centrálny dátovým úložiskom do databázy.

Text článkov je poslaný opätovne na NER topic, ale v tomto prípade aj s id asociovaného linku a samozrejme predikcie. NER je aplikované na telá článkov a entity publikované v topicu databázových objektov. Keďže tentokrát entity obsahujú aj id predikcie aj id linku, proces sa končí. Ak by neskončil prišlo by rekurzívnemu volaniu dokým by text neobsahoval už žiadny odkaz.

Ako aj pri camunde aj tu sa akcie a volania vykonávajú kompletne asynchrónne a používateľ je po vytvorení predikcie presmerovaný na historické predikcie kde po uložení predikcie sa postupne zobrazujú všetky dodatočné metadáta.

V rámci docker služieb je skombinovaná implementácia zookeepera a samotnej kafky do jedného docker obrazu a na monitorovanie a kontrolu topicov sme pridali jednoduché UI.

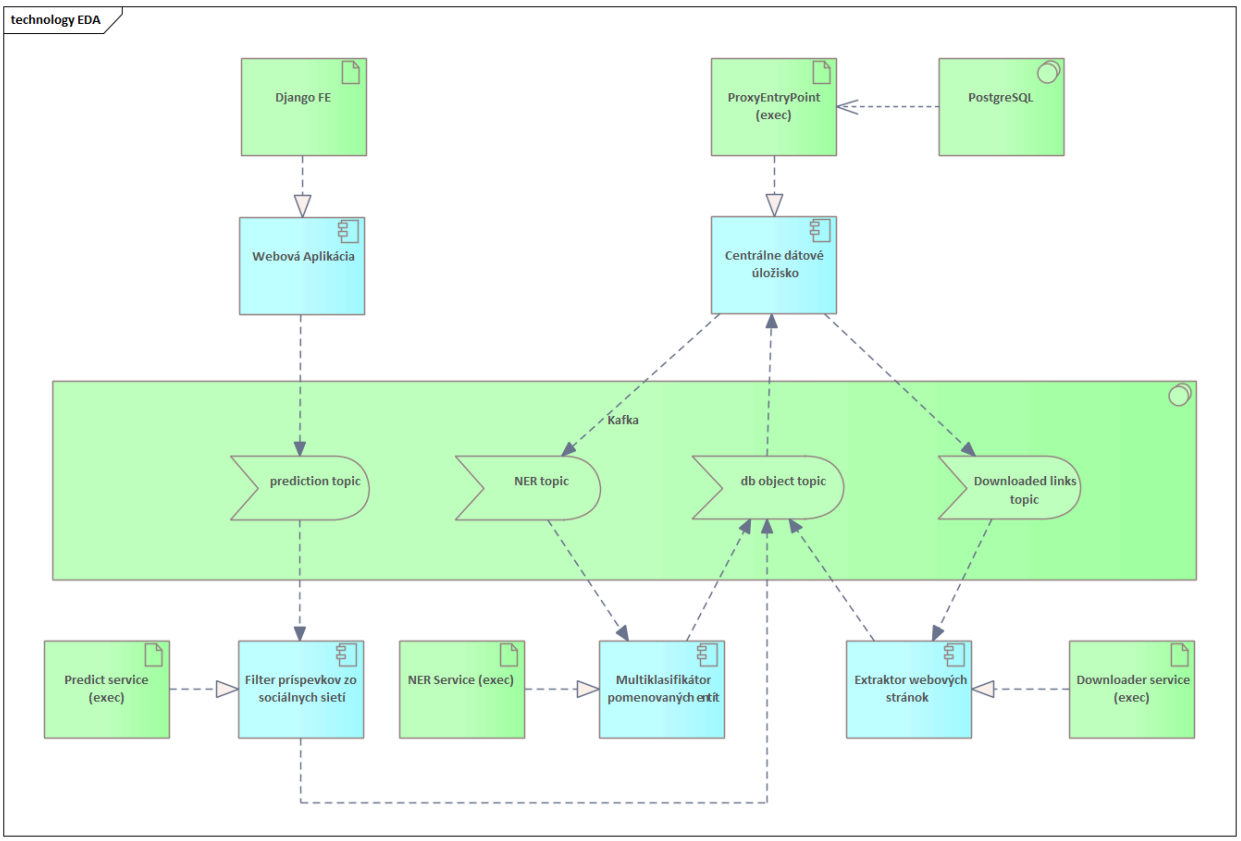
Ukážka objektov, ktoré topic databázových objektov očakáva:

```
{
  "objectType": "entity",
  "predictionId": 7,
  "linkId": 15,
  "obejctData": "eyJwdWJsaXN"
}
```

Objekt popisuje typ entity, ktorá patrí k predikcii s id 7 a linku s id 15. Typ objektu môže byť jeden z: *["prediction", "entity", "link"]*.
V samotnom dátovom objekte vidíme spomínanú zaenkódovanú hodnotu dát v base64, ktorá je dekódovaná do rovnakého objektu ako je to pri všetkých ostatných kategóriách implementácie.

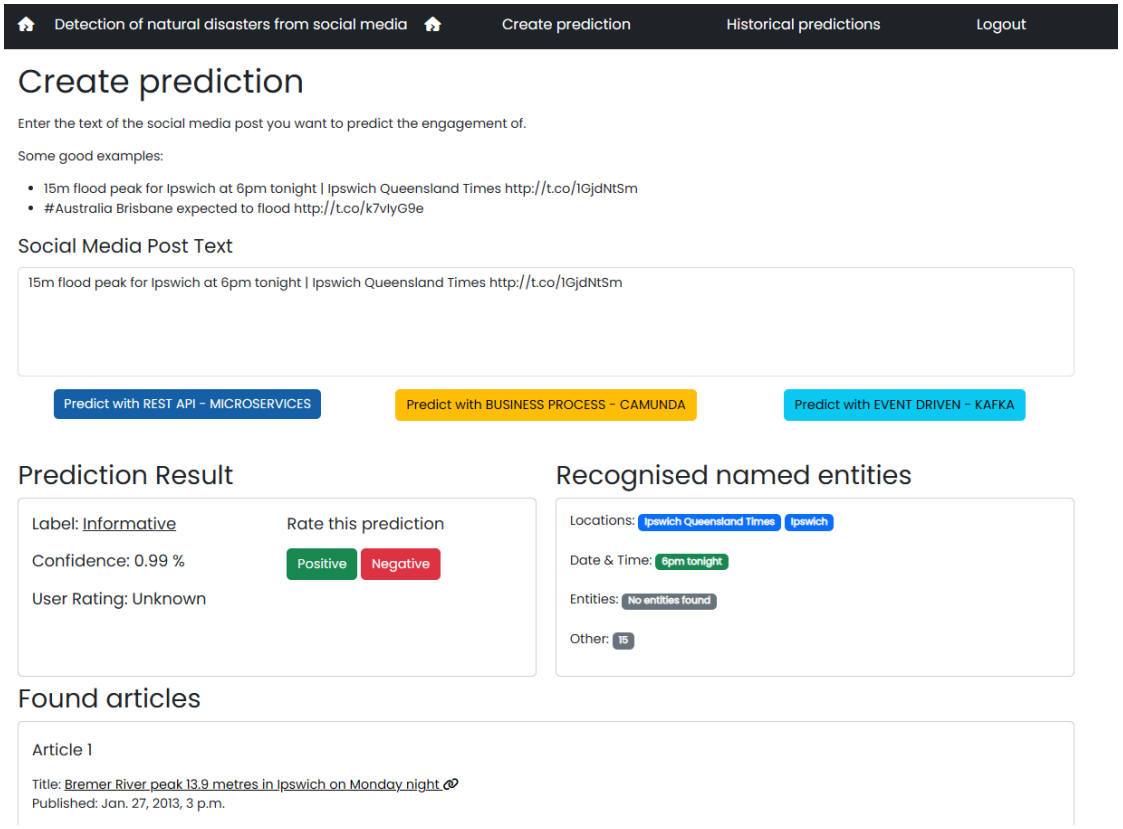
4.6.2. Architektúra pomocou Kafka

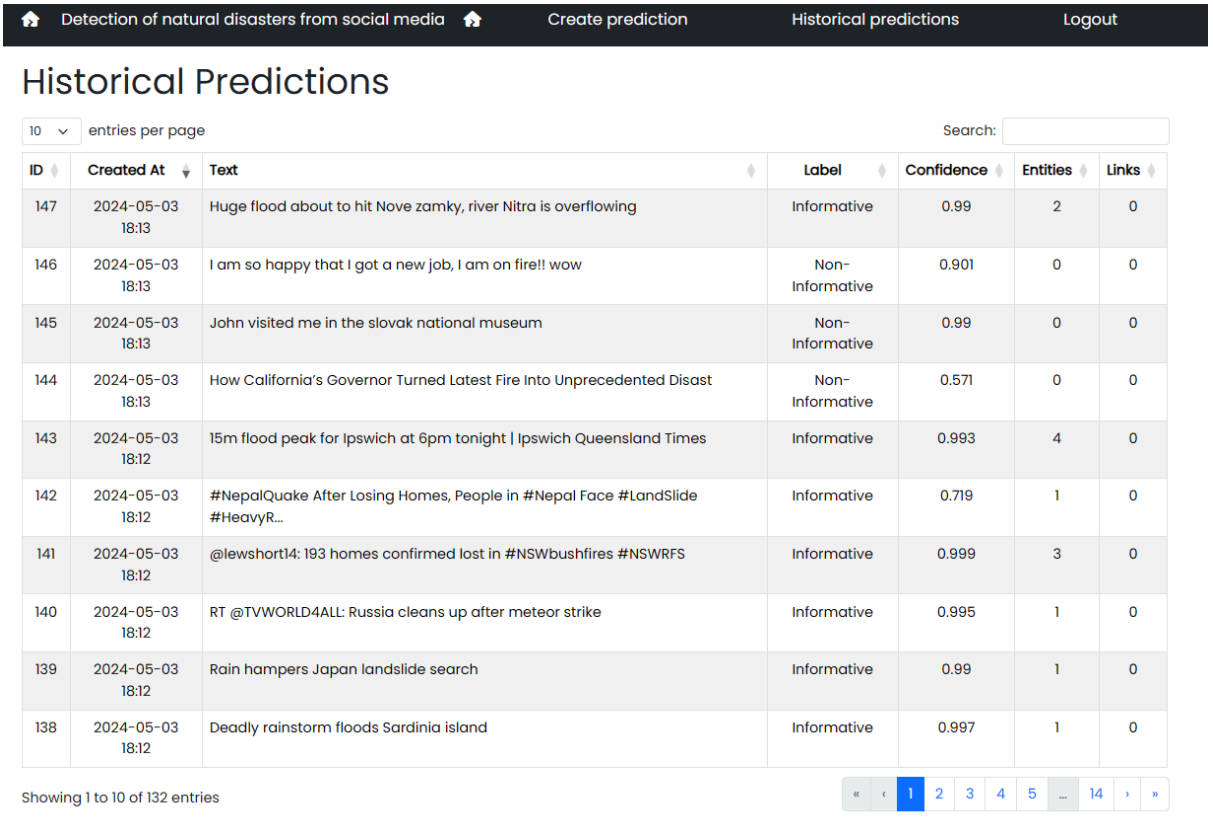
Aplikačná a technologická architektúra pre udalosťami riadený systém na základe Kafky.
Aj keď na nasledujúcom diagrame nie sú znázornené všetky funkcie a služby, ktoré jednotlivé komponenty plnia, hovoríme stále o rovnakých komponentoch ako na diagrame aplikačnej architektúry zo sekcie 4.4.1.



4.7. Snímky používateľského rozhrania

Ukážka tvorby predikcie a vyhľadávania v historických predikciách.





5. Záver

V závere tejto práce môžeme potvrdiť, že sa nám podarilo komplexne navrhnúť a implementovať časť systému na skorú detekciu katastrofických udalostí. Poukázali sme na to, že takýto systém si vyžaduje dôslednú definíciu granurality pre jednotlivé služby a ich vzájomnú orchestráciu.

Ako bolo už viac krát spomenuté cieľom bolo pomôcť záchranárskym jednotkám, ktoré robia kritické rozhodnutia počas prírodnej katastrofe. Práve takouto formou môžu informovať pomôcť občanom ak sa niečo deje, alebo len začína diať.
Na začiatku práce sme sa zamerali na problémy a bolesti, ktoré trápia či už občanov, ale aj zamestnancov krízového štábu. Netreba zabúdať ani na podstatné biznis služby a procesy, ktoré začínajú pri publikovaní nového príspevku na sociálnej sieti, ich priebežným stiahnutím alebo zadaním manuálne do systému, cez extrakciu metadát až po samotnú detekciu prírodnej katastrofy.

Podstatnou súčasťou nášho projektu boli tri typy implementácie zvolenej časti systému. Mikroslužby sa osvedčili ako dnes už zabehnutý spôsob implementácie a prepojenia fungovania jednotlivých kapabilít. Biznis procesy v podobe riešenia cez Camundu nám opäť dala niečo naviac, kedy sme videli že orchestrácia môže byť oddelenou súčasťou asynchrońneho volania služieb na docielenie rovnakého cieľa. Posledným typom bolo zadefinovanie udalosti a prúdenia správ v systéme pomocou sprostredkovateľa Kafka. Koniec koncov toto riešenie už stavalo na existujúcich funkcionalitách a mohli sme už iba zmeniť REST volania na odber a publikovanie do topicov, pričom jednotlivé služby publikovali do zjednoteného databázového topicu. Aj preto implementácia bola veľmi nenáročná a prúdenie správ v systéme sa dalo rovnako jednoducho monitorovať ako aj pri Camunde.

Systém sme zorchestrovali pomocou kontajnerizovaného prístupu na základe docker-compose a je ho možné jediným príkazom jednoducho nasadiť aj s perzistenciou dát do existujúceho komplexnejšieho nástroja na detekciu prírodných katastrof zo sociálnych médií. Teraz sa môžeme ohliadnuť za vynaloženým úsilím a zhodnotiť, že dôsledná príprava, analýza a návrh sú nevyhnutnou súčasťou pri životnom cykle každého projektu. Cesta v budúcich komplexnejších riešeniach na iných predmetoch či v praxi, je otvorená.

Repozitár: https://github.com/melisekm/sm-natdis-detect/

Martin Mališek a Martin Nemec
Máj 2024