Inžinierske dielo

Inžinierske dielo

Tím č. 03 FireAnt

Vedúci tímu: Branislav Pecher, Ivan Srba

Členovia tímu: Dominik Hančuľák, Peter Mačinec, Simona Miková, Denis Mitana, Vajk Pomichal, Miroslav Sumega, František Šefčík

Dátum vytvorenia: 10.11.2019

Big picture

Úvod

Tento dokument obsahuje dokumentáciu projektu tímu číslo 3, FireAnt v roku 2019/2020 vrámci projektu Tímový projekt. Dokumentácia opisuje stav projektu ku dňu 22. 11. 2019 - teda po ukončení troch šprintov. Táto dokumentácia je určená pre členov tímu FireAnt, product-ownerov a ďalšie tímy, ktoré budú pokračovať v práci na projekte.

V tomto dokumente opisujeme globálne ciele projektu ako aj celkový pohľad na systém. Ako ďalšie prikladáme dokumentácie, ktoré v našom tíme počas troch šprintov vznikli.

Globálne ciele projektu na zimný semester

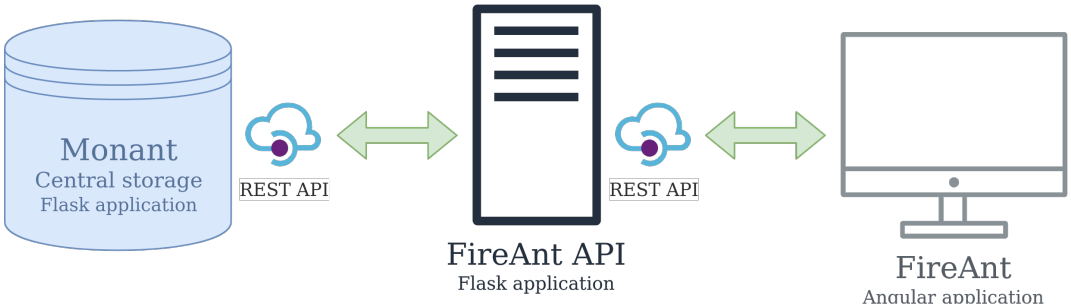
Cieľom projektu na zimný semester je vytvorenie webovej aplikácie, ktorá umožňuje využívanie dát zozbieraných v platforme Monant, vytvorenej počas minuloročného TP a ich následné spracovanie pre bežných používateľov. Aplikácia bude poskytovať používateľom informácie o webových portáloch, článkoch a aj ohľadom krátkych tvrdení (ako napríklad "očkovanie spôsobuje rakovinu", ďalej nazývané aj ako *claim*). Chceme poskytnúť podrobný prehľad na základe dostupných informácií, ktoré sú overené expertmi a aj ďalšie štatistiky. Sem patria dôveryhodnosť zdroja článku, reputácia autora, popularita článku na sociálnych sietiach (predovšetkým Facebook), šírenie článku/claim-u po rôznych portáloch, pravdivosť claim-ov nachádzajúcich sa v článku a ďalšie. Po zimnom semestri plánujeme mať implementovaných 90% MVP, čo by nám umožnilo v prvom šprinte v letnom semestri pripraviť produkt na nasadenie do produkcie.

Globálne ciele projektu na letný semester

V zimnom semestri sme úspešne implementovali veľkú časť funkcionality našej aplikácie. Svedčí o tom aj to, že sme splnili 90% nášho MVP. Po pridaní statických informačných stránok splníme MVP a naša aplikácia bude pripravená na nasadenie do produkcie. Toto je naším cieľom hneď v prvom šprinte letného semestra.

Ako ďalšie sa začneme zaoberať vytvorením funkcionality na monitorovanie, pridávaním prehľadov o populárnych falošných zdrojoch, článkoch a tvrdeniach do Dashoardu a implementáciou Chatbota. Taktiež pripravíme spomínané rozhranie do prehliadača, ktoré umožní skontrolovať pravdivosť aktuálne prezeraného článku a ponúkne presmerovanie na náš portál. V neposlednom rade sa pokúsime nadviazať spoluprácu s expertmi v danej oblasti a pridáme možnosť diskusie s nimi.

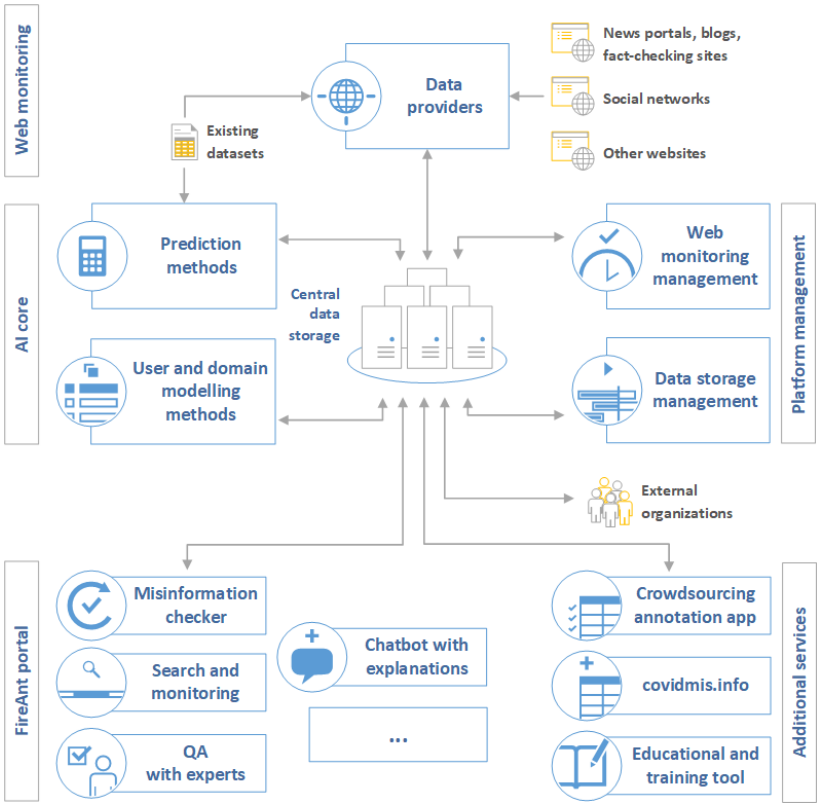
Celkový pohľad na systém



Náplňou nášho projektu je boj s antisociálnym správaním v oblasti medicíny, prevažne falošnými správami a tvrdeniami. Na tento účel vytvárame webový portál, ktorý bude využívať umelú inteligenciu na analýzu iných webových stránok za účelom poskytnutia prehľadných informácií a štatistík o dôveryhodnosti danej stránky. Náš portál umožňuje vyhľadať článok, zdroj alebo tvrdenie a zobraziť k nemu príslušné štatistiky a grafy vo forme widgetov. Medzi ďalšie funkcionality patrí možnosť filtrovania podporovaných entít s možnosťou uloženia filtra a následnými emailovými notifikáciami pri pribudnutí nových záznamov alebo taktiež možnosť položiť otázku týkajúcu sa akéhokoľvek článku, tvrdenia alebo zdroja dostupného na našej stránke jednému z expertov. Najviac zaujímavou funkcionalitou je pridanie chatbota s viacerými dostupnými scenármi, , ktoré prevedú používateľa postupne našimi dostupnými widgetmi alebo sa snažia poukázať na veci, ktoré sú pri niektorých nepravdivých článkoch podozrivé.

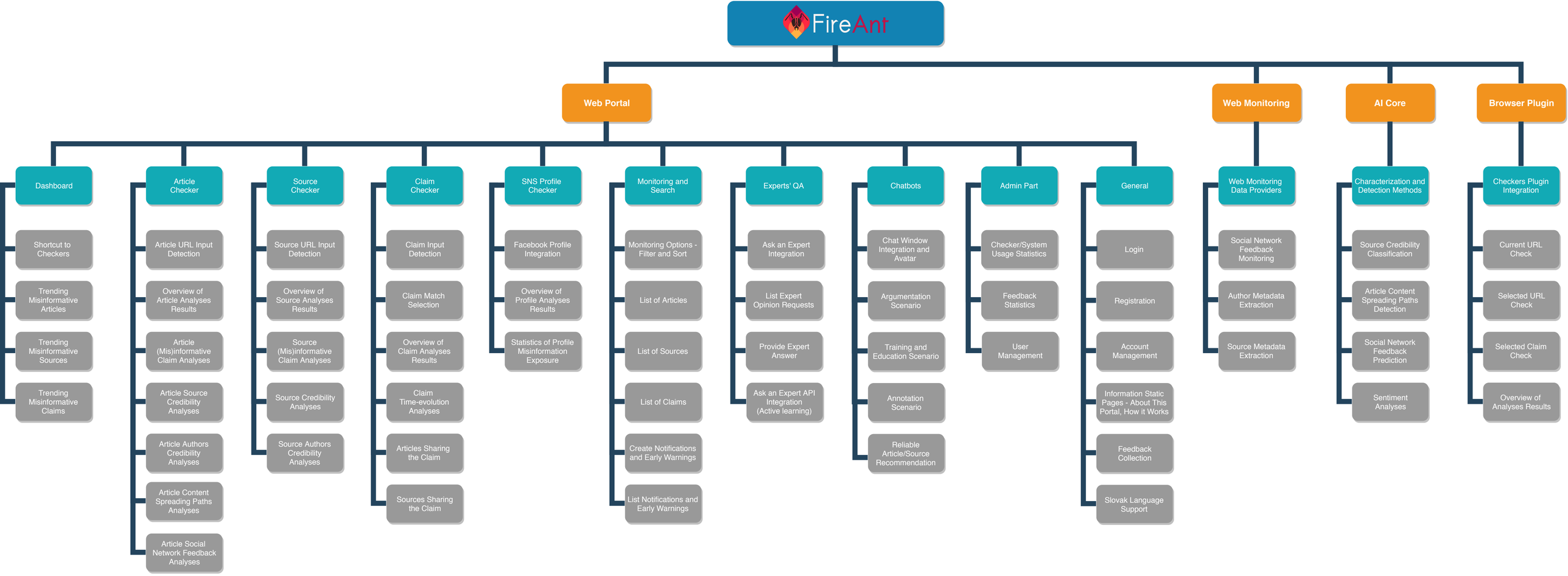
Všetkými vyššie spomenutými funkcionalitami nášho portálu by sme chceli prispieť k zníženiu šírenia dezinformácií a pomôcť ľuďom rozhodnúť sa o pravdivosti informácií, s ktorými sa stretávajú na internete. Za účelom zjednodušenia používania nášho systému na boj s falošnými správami bolo cieľom aj rozšírenie do webových prehliadačov. Toto rozšírenie malo poskytovať v prehľadnej podobe kľúčové funkcionality z našej webovej stránky, aby používatelia mali rýchlo dostupnú pomoc v prípade, ak sa nevedia rozhodnúť, či majú stránku považovať za vierohodnú alebo nie. Avšak toto rozšírenie sa nám už kvôli časovému rozsahu tímového projektu nepodarilo implementovať.

Naša aplikácia je rozdelená na backend a frontend. Backend predstavuje Flask aplikácia komunikujúca s centrálnym úložiskom Monantu, z ktorého získavame všetky dáta týkajúce sa článkov, zdrojov a tvrdení. Na ukladenie informácií o používateľoch, notifikáciách a otázky aj odpovede expertov ukladáme v našej databáze. Architektúru Monantu spolu s naším systémom môžeme vidieť na obrázku nižšie. Získané dáta spracuvávame u nás na backende a následne odosielame frontendu reprezentovanému Angular aplikáciou. Na frontende prostredníctvom widgetov zobrazujeme prehľadné štatistiky o vyhľadanom zdroji, článku alebo tvrdení. Bližšie informácie k samotným modulom aplikácie ako aj databázový model sa nachádzajú nižšie v časti Dokumentácia backendu systému a Dokumentácia frontendu systému.



WBS diagram

V tejto časti je znázornený WBS diagram nášho projektu:


FireAnt
Web Portal
Web Monitoring
AI Core
Browser Plugin
Dashboard
Shortcut to Checkers
Trending Misinformative Articles
Trending Misinformative Sources
Trending Misinformative Claims
Article Checker
Article URL Input Detection
Overview of Article Analyses Results
Article (Mis)informative Claim Analyses
Article Source Credibility Analyses
Article Authors Credibility Analyses
Article Content Spreading Paths Analyses
Article Social Network Feedback Analyses
Source Checker
Source URL Input Detection
Overview of Source Analyses Results
Source (Mis)informative Claim Analyses
Source Credibility Analyses
Source Authors Credibility Analyses
Claim Checker
Claim Input Detection
Claim Match Selection
Overview of Claim Analyses Results
Claim Time-evolution Analyses
Articles Sharing the Claim
Sources Sharing the Claim
SNS Profile Checker
Facebook Profile Integration
Overview of Profile Analyses Results
Statistics of Profile Misinformation Exposure
Monitoring and Search
Monitoring Options - Filter and Sort
List of Articles
List of Sources
List of Claims
Create Notifications and Early Warnings
List Notifications and Early Warnings
Experts' QA
Ask an Expert Integration
List Expert Opinion Requests
Provide Expert Answer
Ask an Expert API Integration (Active learning)
Chatbots
Chat Window Integration and Avatar
Argumentation Scenario
Training and Education Scenario
Annotation Scenario
Reliable Article/Source Recommendation
Admin Part
Checker/System Usage Statistics
Feedback Statistics
User Management
General
Login
Registration
Account Management
Information Static Pages - About This Portal, How it Works
Feedback Collection
Slovak Language Support
Web Monitoring Data Providers
Social Network Feedback Monitoring
Author Metadata Extraction
Source Metadata Extraction
Characterization and Detection Methods
Source Credibility Classification
Article Content Spreading Paths Detection
Social Network Feedback Prediction
Sentiment Analyses
Checkers Plugin Integration
Current URL Check
Selected URL Check
Selected Claim Check
Overview of Analyses Results

Dokumentácia backendu systému

Inštalácia a konfigurácia

Tím č. 03 FireAnt

Vedúci tímu: Branislav Pecher, Ivan Srba

Autor dokumentácie: Denis Mitana

Dátum vytvorenia: 12.11.2019

Úvod

FireAnt backend je naprogramovaný v programovacom jazyku Python v. 3.7. Na zabezpečenie API je použitý webový aplikačný rámec Flask.

Inštalácia a spustenie

Poznámka: Je potrebné nahradiť `{environment}` v niektorých príkazoch za názov aktuálne používaného prostredia, definovaného premennou prostredia `FIREANT_ENV` (predvolené je `staging`).

1. Naklonujeme existujúci repozitár.

   ```
   $ git clone git@github.com:BranislavPecher/fireant-backend.git
   $ cd fireant-backend
   ```

2. Ďalej pokračujeme nastavením databázy a spustením serveru, postupujeme podľa krokov v sekcii **Spustenie FIreAnt-u v dockeri.**

Spustenie FireAnt-u v dockeri

Poznámka: Nasledujúce kroky sú pre spustenie FireAnt-u v development prostredí (môžu byť použité aj pri nastavení production a staging prostredia, avšak len lokálne - kvôli databáze). Súbor `docker-compose.yml` je určený pre nasadzovanie na Monant server, kde databáza nie je potrebná (je vytvorená z compose súboru centrálneho úložiska).

1. Vo **vývoji** aj v **produkcii** je možné spustiť server príkazom `docker-compose up --build -d`.V tomto prípade je nutné nastaviť všetky premenné prostredia v .env súbore.

   ```
   FIREANT_ENV=development
   FIREANT_FLASK_PORT_HOST=5000
   FIREANT_FE_URL=https://team03-19.studenti.fiit.stuba.sk/fireant-staging
   MONANT_CENTRAL_STORAGE_HOST=https://api-staging.monant.fiit.stuba.sk
   MONANT_CENTRAL_STORAGE_USERNAME=fireant
   MONANT_CENTRAL_STORAGE_PASSWORD='API_KEY'
   SENDGRID_API_KEY='API_KEY'
   JWT_ENCODING_PASSWORD='SECRET_KEY'
   JWT_EXPIRATION_HOURS=24
   ```

   Vo vývoji je mozné použiť hocijakú hodnotu pre `FIREANT_ENV`, napr. `development.` Ak sa zvolí hodnota `production` alebo `staging`, spustí sa v kontajneri gunicorn server namiesto flask development serveru.

2. Vytvoríme docker volume pre databázu:

   ```
   $ docker volume create --name=fireant-database-{environment}
   ```

3. Vytvoríme docker network pre komunikáciu s databázou:

   ```
   $ docker network create {environment}
   ```

4. Následne spustíme server príkazom :

   ```
   $  docker-compose -f docker-compose-dev.yml up --build -d
   ```

   Nesmie sa zabudnúť prepínač `-f` pre spustenie správneho docker compose súboru.

5. Ak spustenie serveru prebehlo správne, vytvoríme databázového používateľa nasledujúcimi príkazmi:
   a. Dostaneme sa do Docker kontajneru:

      ```
      $ docker exec -it fireant_database_{environment}_con bash
      ```

   b. Vytvoríme používateľa `fireant_api` s heslom `password` a pridelíme mu všetky oprávnenia databázy (podľa premenných prostredia `FIREANT_DB_USERNAME`, `FIREANT_DB_PASSWORD` a `FIREANT_DB_DATABASE` môžeme zmeniť meno používateľa, heslo, alebo názov databázy):

      ```
      $ su - postgres
      $ createuser fireant_api
      $ createdb fireant_api_test
      $ psql
      $ alter user fireant_api with encrypted password 'password';
      $ grant all privileges on database fireant_api to fireant_api;
      $ grant all privileges on database fireant_api_test to fireant_api;
      ```

6. Spustíme migrácie pre aktualizáciu databázy:

   ```
   $ docker exec -it fireant_backend_{environment}_con bash
   $ python -m api.manage db upgrade
   ```

7. Pri opätovnom spúšťaní serveru ho spustíme jednoducho pomocou príkazu:

   ```
   $ docker-compose -f docker-compose-dev.yml --build -d
   ```

Rasa

Poznámka: Je potrebné nahradiť <version> v niektorých príkazoch za aktuálnu verziu.

Aby sme mohli používat Rasa chatbota musíme najskôr vytvorit iniciálny Rasa projekt a potom spustiť Rasa server.

1. Iniciálny Rasa projekt môžeme vytvoriť nasledovne:

   ```
   $ mkdir rasa
   $ docker run -u $(id -u):$(id -g) -v $(pwd)/rasa:/app rasa/rasa:1.7.0-full init --no-prompt
   ```

2. Následne môžeme Rasa server spustiť použitím

   ```
   $ docker run -u $(id -u):$(id -g) -v $(pwd)/rasa:/app -p 5005:5005 rasa/rasa:<version> run
   ```

   alebo príkazom

   ```
   $ docker-compose -f docker-compose-dev.yml up --build -d
   ```

Trénovanie

Poznámka: Natrénovaný model nie je uložený na gite a preto je ho nutné pridať ručne do zložky rasa/models/ Pre každý release je na Gite uložený model, ktorý je možné stiahnuť a použiť na základe aktuálnej verzie aplikácie.

1. Pred príkazom na trénovanie je nutné vypnúť rasa kontajner príkazom:

   ```
   $ docker stop rasa_{environment}_con
   ```

2. Spustíme trénovanie príkazom:

   ```
   $ docker run -u $(id -u):$(id -g) -v $(pwd)/rasa:/app -p 5005:5005 rasa/rasa:<version> train
   ```

3. Po dokončení trénovania spustíme kontajner:

   ```
   $ docker start rasa_{environment}_con
   ```

Vytváranie endpointov

Tím č. 03 FireAnt

Vedúci tímu: Branislav Pecher, Ivan Srba

Autor dokumentácie: Denis Mitana

Dátum vytvorenia: 6.11.2019

Obsah

- Vytváranie endpointov
- Pravidlá umiestňovania endpointov
- Vytvorenie modulu pre endpointy
- Vytvorenie handleru pre endpoint
    - Endpointy s autentifikáciou používateľa

Pravidlá umiestňovania endpointov

- Endpointy umiestňujeme do modulov */api/views/<version>/<filename>.py*.
- V jednom module môže byť definovaných aj viac endpointov.
- Názov modulu v ktorom sú endpointy umiestnené agreguje ich použitie v časti systémovej funkcionality (napr. *checker.py*).
- Ak sa endpoint hodí do existujúceho modulu umiestnite ho tam. Ak sa vytvára endpoint pre novú časť systémovej funkcionality, tak preň vytvorte nový modul s vhodným názvom podľa špecifikej systémovej funkcionality, ktorú bude podporovať.

Vytvorenie modulu pre endpointy

Tento postup uplatňujeme vtedy, ak vytvárame endpoint pre novú časť systémovej funkcionality. To znamená, že endpoint sa nehodí do žiadneho už existujúceho modulu v balíku pre endponty */api/views/<version>/*.

1. Vytvoríme nový modul s výstižným názvom v baliku */api/views/<version>/*.
2. Vytvoríme *Blueprint* pre nový modul.
   V príklade uvažujeme, že vytvorený modul sa nazýva *checker.py* a vytvárame jeho prvú verziu. Ako vidieť v priklade, tak *Blueprinty* pomenuvávame podľa názvov modulov.

Vytvorenie Blueprintu pre nový modul checker.py

```
from flask import Blueprint
from api.views.v1 import v1_api_prefix

checker = Blueprint(
    'checker',
    __name__,
    url_prefix=f'{v1_api_prefix}/checker'
)
```

3. Zaregistrujeme novovytvorený *Blueprint* v aplikácii. *Blueprint* sa registruje v module */api/__init__.py* vo funkcii *create_app(...)*.
   V príklade uvažujeme zaregistrovanei *Blueprintu* pre vyššie vytvorený modul *checker.py*.

Zaregistrovanie Blueprintu checker

```
...
from api.views.v1.checker import checker as checker_api_v1
...


def create_app(name):
    ...

    # Register blueprints
    ...
        # This line register new blueprint checker
    flask_app.register_blueprint(checker_api_v1)
    ...

        return flask_app


app = create_app(__name__)
```

Vytvorenie handleru pre endpoint

Ak už existuje vhodný modul pre endpoint, ktorý chceme vytvoriť, tak budeme postupovať podľa nasledovného návodu pre vytvorenie handleru.

Pri vytváraní handlerov pre endpoint budeme využívať dekorátory knižnice *Flask* a *Apistrap*, ktoré nám pomáhajú validovať parametre požiadaviek a odpovede a taktiež nám automaticky vygenerujú API dokumentáciu podľa správne formátovaného docs tringu podľa Metodika štýlu písania kódu pre Python na ceste */apidocs*.

1. Vytvoríme handler pre endpoint a špecifikujeme cestu k endpointu.
   Pri vytváraní handleru využívame dekorátor **@<blueprint_name>.route(path, methods=...)** pre nastavenie cesty k endpointu.
   Názov handleru (funkcie) **je rovnaký** ako cesta k endpointu.

Určenie cesty k endpointu

```
@foo.route('/fn', methods=['GET'])
def fn():
    pass
```

2. Pre definovanie parametrov požiadavky (query pre GET požiadavku, body pre POST požiadavku) používame nasledovné dekorátory :

   a. **@oapi.accepts_qs('param1', 'param2', 'param3', ...)**: dekorátor definuje vstupné query string parametre. V nasledujúcom príklade vidíme, ako definovať povinné a nepovinné query string parametre.

Určenie query string parametrov

```
...
@foo.route('/fn', methods=['GET', 'POST'])
@oapi.accepts_qs('param_required1', 'param_required2', 'param_optional')
def fn(param_required1: str, param_required2: int, param_optional: str = 'test'):
        ...
    pass
```

> Ak nebudú zadané všetky povinné query string parametre, tak endpoint vráti **400 Bad Request**.

b. **@oapi.accepts(Model):** dekorátor definuje vstupné body parametre podľa definovaného *Schematics modelu*. V nasledujúcom príklade vidíme, ako definovať body parametre pomocou *Schematics modelu*. *Schematics modely definujem podľa Vytváranie Schematics modelov.*

Určenie body parametrov

```
...
@foo.route('/fn', methods=['POST'])
@oapi.accepts(Article)
def fn(article: Article):
        ...
    pass
```

> Ak nebudú zadané všetky povinné body parametre, tak endpoint vráti **400 Bad Request**.

3. Pre definovanie parametrov v ceste používame vyššie spomenutý dekorátor **@<blueprint_name>.route(path, methods=...).** V nasledujúcom príklade vidíme, ako definovať parametre v ceste k endpointu.

Parametre v ceste k endpointu

```
@foo.route('/fn/<string:param1>/<integer:param2>', methods=['GET'])
def fn(param1, param2):
        ...
    pass
```

4. Pre definovanie odpovede používame dekorátor **@oapi.responds_with(Model, code=...)** , ktorý nám umožňuje odpovedať pomocou *Schematics modelov* (výsledná odpoveď bude skonvertovaná na JSON, tieto modely nám pomáhajú validovať odpoveď). *Schematics modely definujem podľa Vytváranie Schematics modelov.*
V nasledujúcom príklade vidíme, ako definovať odpoveď endpointu vzhľadom na rôzne status kódy odpovede:

Definovanie odpovedí endpointu

```
@foo.route('/fn', methods=['GET'])
@oapi.accepts_qs('param')
@oapi.responds_with(Model, code=200)
@oapi.responds_with(BadRequestResponse, code=400)
@oapi.responds_with(InternalServerErrorResponse, code=500)
def fn(param: str):
        ...
    pass
```

> Ak definujeme nejaké vstupné parametre, tak **vždy** pridáme odpoveď **400 Bad Request**, aj keď túto odpoveď explicitne nevraciame. Táto odpoveď sa vráti automaticky, ak nie sú zadané všetky povinné vstupné parametre. **Definovanie odpovede je potrebné pre účely dokumentácie API.**

Endpointy s autentifikáciou používateľa

Ak endpoint vyžaduje autentifikáciu používateľa cez JWT token, je potrebné aby bol tento token pridaný do hlavičky ***Authentication*** požiadavky (*request*). Pri vytváraní takéhoto endpointu je potrebné k už spomínaným krokom pridať nasledovné:

1. Pridať k funkcii endpointu dekorátor **@authenticate**. Tento dekorátor za nás vyrieši všetky náležitosti ohľadom autentifikácie používateľa - overí prijatý token v hlavičke požiadavky.
2. Pridať k funkcii endpointu argument ***user*** typu ***User*** (*databázový model User)*:
   V nasledujúcom príklade vidíme, ako definovať endpointu dekorátor ***@authenticate*** a zároveň pridať argument ***user*** (poznámka: argument ***article*** má len ilustratívny účel):

Pridanie dekorátoru pre autentifikáciu

```
@foo.route('/fn', methods=['POST'])
@authenticate
...
@oapi.accepts(Article)
def fn(article: Article, user: User):
        """
        Comment...

        :param article, ...
        :param user: User, authenticated user whom notification will be
                assigned.
        :return: ...
        """
    ...
    pass
```

> Ak autentifikácia neprebehne v poriadku (ak nebol v hlavičke *Authentication* vložený token alebo je token nevalídny), endpoint automaticky vráti HTTP status code **401** a správu o zlyhaní autorizácie (správa v závislosti od chyby).

Endpointy s autorizáciou používateľa

Pri vytváraní endpointu, ktorý vyžaduje autorizáciu používateľa, je potrebné okrem ostatných krokov spraviť aj nasledovné:

1. Autorizácia sa vykonáva po autentifikácii používateľa, preto je potrebné pridať k funkcii endpointu dekorátor **@authorize**, ktorý musí byť umiestnený pod dekorátorom **@authenticate**. Tento dekorátor za nás vyrieši všetky náležitosti ohľadom autorizácie používateľa - overí, či existuje entita, ku ktorej sa snaží používateľ pristúpiť a či má na to právo.
   a. Dekorátor **@authorize** požaduje 2 parametre - databázový model entity, ku ktorej sa snažíme pristúpiť a názov parametra pre id danej entity (default ***entity_id***).
2. *V nasledujúcom príklade vidíme, ako definovať endpointu dekorátor* ***@authorize*** *spolu s dekorátorom* ***@authenticate*** a zároveň pridanie argumentu ***user****, potrebného pri autentifikácii* (poznámka: argument ***article*** má len ilustratívny účel):

Pridanie dekorátotu autorizácie

```
@foo.route('/<int:entity_id>', methods=['POST'])
@authenticate
@authorize(Article)
...
@oapi.accepts(Article)
def fn(article: Article, user: User):
        """
        Comment...

        :param article, ...
        :param user: User, authenticated user whom notification will be
                assigned.
        :return: ...
        """
    ...
    pass
```

Ak autorizácia neprebehne v poriadku, endpoint automaticky vráti HTTP status code **404** (ak entita, ku ktorej sa používateľ snaží pristúpiť neexistuje), alebo **403** (ak daný používateľ nemá právo na vykonávanie zmien) a správu o zlyhaní autorizácie (správa v závislosti od chyby).

Vytváranie Schematics modelov

Tím č. 03 FireAnt

Vedúci tímu: Branislav Pecher, Ivan Srba

Autor dokumentácie: Denis Mitana

Dátum vytvorenia: 13.11.2019

Pravidlá Schematics modelov

- Ak sú Schematics modely importované v rámci súboru spolu s inými, rovnako pomenovanými modelmi (napr. databázové modely), pridávame im pri importe prefix **Schematics.**

  ```
  from api.schematics.users import User as SchematicsUser
  ```

- Modely umiestňujeme do balíka */api/schematics/*.
- Ak vytvárame nový modul pre modely, tak ho pomenuje výstižne tak, aby jeho názov agregoval funkcionalitu všetkých modelov, ktoré sú v ňom vytvorené.

Momentálne existujú nasledujúce moduly modelov:

- *monant/:*
  - obsahuje modely pre integráciu s produktom *MonAnt*
  - všetky modely by mali odrážať modely na aplikácii MonAnt
- *responses.py*:
  - obsahuje modely pre klasické a často používané odpovede endpointov
  - napr. odpovede pre HTTP status kódy, prípadne generické: EmailResponse, IdResponse.
  - detailné odpovede, ktoré sa týkajú konkrétnej entity patria do príslušného súboru (napr. UserAuthenticationSuccessResponse do *users.py*)
- iné, špecifické modely

Komentáre Schematics modelov

Každý model (až na výnimku dedenia z dôvodu rôzneho HTTP kódu) by mal byť okomentovaný. V prípade, že ide o model, ktorý môže slúžiť ako *payload* alebo *response* (prípadne oboje), mala by byť táto informácia uvedená v komentári.

Príklad komentáru, ktorý obsahuje tiež informáciu o tom, že je používaný ako *payload* aj *response*:

```
class Notification(Model):
    """
    Defines model scheme of notification.

        This schematics model is used as a payload and an api response.
    """
    ...
```

Analýza rozbehania testov CÚ

Tím č. 03 FireAnt

Vedúci tímu: Branislav Pecher, Ivan Srba

Autor dokumentácie: Denis Mitana

Dátum vytvorenia: 29.11.2019

Nutné kroky k rozbehaniu testov pre lokálne testovanie funkcií a endpointov, ktoré sú závislé na CÚ:

- Nastavenie lokálnej inštancie CÚ s funkčnými endpointami na komunikáciu s DB
    - Je potrebný jednotný návod, aby si to každý vedel rozbehať.
    - Najlepšie by bolo vytvoriť skript, ktorý by inicializoval celú DB, aby to nebolo nutné robiť vždy ručne.
- Je nutné mocknut triedu *CentralStorageConnector* aby komunikovala s našou lokálnou verziou CÚ
    - Ako to spraviť?
- Je nutné zabezpečiť pripájanie na lokálnu verziu DB.
    - Môžeme sa inšpirovať pripájaním z monantu.
- Je nutné zabezpečiť pomocné funkcie, ktoré budú vytvárať záznamy v DB pre účely testovania:
    - Myslím, že bude stačiť aj jednoduché ORM (SQLAlchemy), ktoré bude vedieť ako vyzerá jeho model a bude ho vedieť nad ním vykonať základny CRUD. V takomto prípade nebudú potrebné žiadne pomocné funkcie.
    - ORM sa môžeme inšpirovať z monantu, ale iba inšpirovať. Netreba nám všetky veci, ktoré tam sú.
    - Druhá možnosť je vytvoriť pomocné funkcie, ktoré budú volať endpointy na lokálnej verzii monantu a pomocou nich vytvárať záznamy v DB za účelom testovania. **Tento spôsob môže byť jednoduchší, ale menej efektívny.**
- Je nutné premazať DB po každom teste, ktorý ju využíval.
    - Môžeme sa inšpirovať premazávaním z monantu.
- Rozbehať testy v travise:
    - Travis musí pullovať (git pull) najnovšiu verziu CÚ pri vykonávaní testov.
    - Na testy vo vetve develop sa bude pullovať CÚ z vetvy develop, analogicky pre testy v master vetve.
    - DB sa musí inicializovať - tu by sa mohol hodiť vyššie spomínaný skript na inicializovanie DB.

Autentifikácia a autorizácia

Tím č. 03 FireAnt

Vedúci tímu: Branislav Pecher, Ivan Srba

Autor dokumentácie: Denis Mitana

Dátum vytvorenia: 19.03.2020

Úvod

Autentifikácia a autorizácia na portáli FireAnt je potrebná na zabezpečenie poskytovaného notifikačného mechanizmu. Každý používateľ si môže vytvoriť vlastné notifikácie na rôzne filtre pre články, zdroje alebo tvrdenia.

Proces autentifikácie

Na autentifikáciu je použitý JWT token s platnosťou 24 hodín, ktorý prvýkrát príde používateľovi na zadaný email. Pri takomto riešení nie je potrebné žiadne heslo.

1. Pri prihlásení sa odošle nasledujúca požiadavka, ktorá odošle na zadaný *email* JWT token. Odoslaná *url* je nutná, aby sa používateľ po potvrdení prihlásenia v emali vrátil na správnu stránku.

POST	/users/login	`{ "email": "...", "url": "..." }`

2. Po kliknutí na odkaz v emaili sa odošle nasledujúca požiadavka, ktorá overí *token* z emailu. Z tokenu sa dekóduje email a ak sa používateľ prihlásil prvýkrát, tak sa preňho vytvorí záznam v databáze. Ak sa nepodarí token dekódovať alebo už expiroval, tak sa vrati odpoveď so stavovým kódom 401.

POST	/users/authenticate	`{ "token": "..." }`

3. Týmto sa stal používateľ autentifikovaný na 24 hodín a môže využívať endpointy vyžadujúce autentifikáciu.

Proces autorizácie

Autorizácia prebieha po úspešnom vykonaní autentifikácie. Jej účelom je preveriť, či autentifikovaný používateľ má právo na zmeny, ktoré sa pokúša vykonať. **Pred autorizáciou sa vždy musí použiť dekorátor check_for_existence, pretože vracia objekty, ktoré sa v autorizácii kontrolujú.**

Autorizácia prebieha tak, že na základe endpointu, pre ktorý sa autorizácia vykonáva sa overí, či používateľ má právo na zmeny danej entity (právo na zmeny má, ak je majiteľom danej entity). Ak používateľ práva na úpravu nemá, vráti sa odpoveď so stavovým kódom 403.

Endpointy vyžadujúce autentifikáciu

Pri vytváraní požiadaviek (*requests*) na API je potrebné pridať autentifikačný token do hlavičky s názvom *Authentication*.

GET	/notifications
POST	/notifications/<notification_id> /questions/<question_id>/answers
PUT	/notifications/<notification_id> /questions/<question_id>/answers/<answer_id>
DELETE	/notifications/<notification_id>

Endpointy vyžadujúce autorizáciu

PUT	/notifications/<notification_id> /questions/<question_id>/answers/<answer_id>
DELETE	/notifications/<notification_id>

Dátový model

Tím č. 03 FireAnt

Vedúci tímu: Branislav Pecher, Ivan Srba

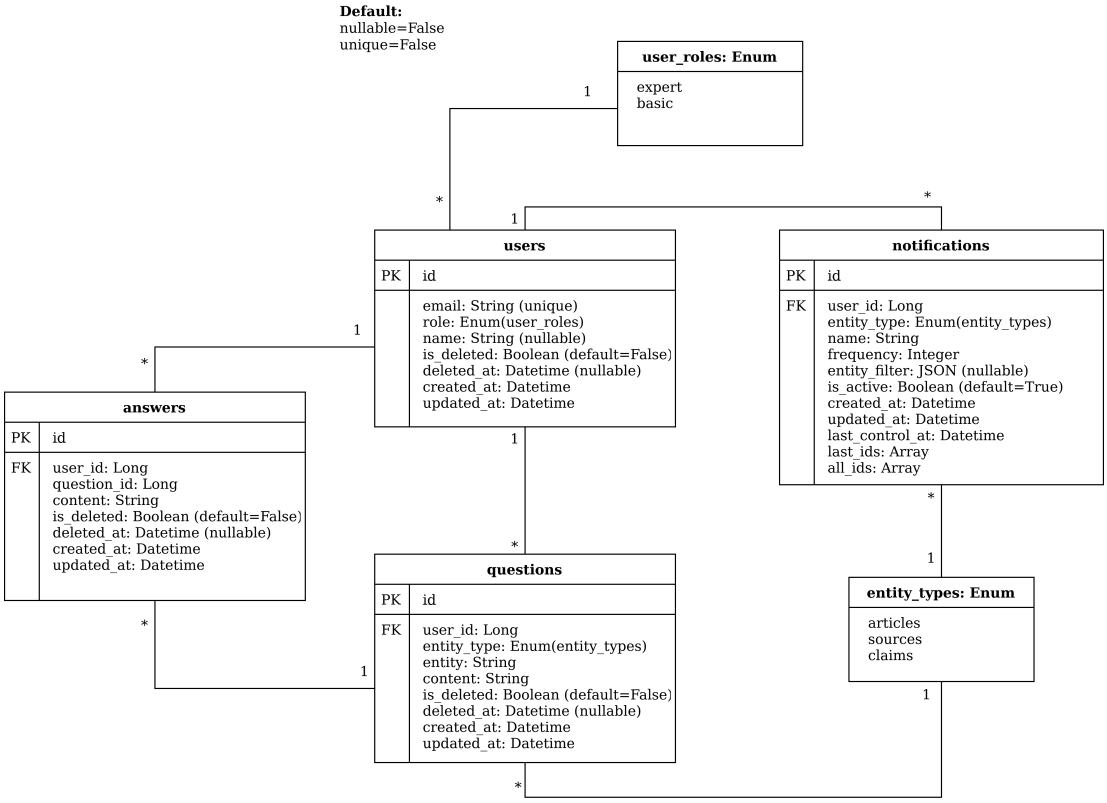
Autor dokumentácie: Denis Mitana

Dátum vytvorenia: 17.03.2020

Úvod

FireAnt backend používa na perzistenciu open source relačnu databázu PostgreSQL.

Relačný dátový model



Odkaz na editovateľný relačný dátový model https://app.diagrams.net/#G1rsyFG38yRbs3_OdzRQ9RYfSFJ62Nhg2O

Zmeny

- 17.03.2020 Denis Mitana : Pridanie iniciálnej dátovej schémy obsahujúcej entity *users, notifications* a enumeráciu *entity_types.*
- 30.03.2020 Miroslav Sumega : Pridanie stĺpcov *last_control_at*, *last_ids* a *all_ids* do entity *notifications*
- 08.04.2020 Miroslav Sumega : Pridanie entity *questions.*
- 12.04.2020 Miroslav Sumega : Pridanie stĺpca *role* do entity *users* a pridanie enumerácie *user_roles*.
- 17.04.2020 Miroslav Sumega : Pridanie entity *answers*.
- 18.04.2020 Peter Mačinec : Pridanie stĺpca *name* do entity *users*.

Nastavenie cron job-ov

Tím č. 03 FireAnt

Vedúci tímu: Branislav Pecher, Ivan Srba

Autor dokumentácie: Peter Mačinec

Dátum vytvorenia: 03.04.2020

Štruktúra súborov

Pre nastavenie nových cron job-ov sú potrebné nasledujúce súbory.

- `api/cron.py`
  - definovanie *flask_script* manager pre cron
  - definovanie potrebných python funkcií, ktoré má cron vykonať pomocou *cron manager*-a
  - registrovanie cron job-ov do crontab-u
- `scripts/cron/`
  - definovanie pomocných shell skriptov pre vykonávanie cronom
- `/var/log/cron.log`
  - ukladanie logov pre job-y, ktoré majú logovanie nastavené

Definovanie nového cron job-u

Pre vytvorenie a nastavenie nového cron job-u, treba ísť cez nasledujúce kroky:

1. Pridať nový cron job do crontabu pomocou funkcie `register_cron_jobs()` v súbore `api/cron.py` podľa uvedených príkladov:

```
@cron_manager.command
def register_cron_jobs():
    """Register all cron jobs."""
    log_output = '>> /var/log/cron.log 2>&1'

    cron = CronTab(user='root')

        # 1. example job
        cron.new(command='echo "Hello"')

        # 2. example job with output logging (/var/log/cron.log)
        cron.new(command=f'echo "Hello logs!" {log_ouput}')

    # 3. example job which needs to use api cron manager function
    job = cron.new(
        command='/app/scripts/cron/example.sh'
    )
    job.minute.every(5)  # Define repeating (default every minute)

    cron.write()
```

Opakovanie cron job-u je možné definovať na rôznej úrovni (minúty, hodiny, dni, ...). Pre viac informácií, pozrieť knižnicu `python-crontab`: https://pypi.org/project/python-crontab/

2. Ak je potrebné v job-e používať flask aplikáciu (napr. používanie jej modelov), je potrebné vykonať ešte nasledujúce kroky:

   a. pridať do súboru `api/cron.py` príkaz pre cron manager-a:

   ```
   @cron_manager.command
   def example_command():
           """Only example command."""
       print("Hello world!")
   ```

   b. pridať súbor s príslušným skriptom do `scripts/cron`:

   ```
   #!/usr/bin/env bash

   # Import environment variables
   set -a
   . /etc/environment_vars
   set +a

   cd /app

   /usr/local/bin/python -m api.cron example_command
   ```

   c. definícia cron job-u musí byť cez príklad č. 3 - využitie scriptu (kontrola kroku č. 1).

Logovanie výstupov

V prípade, že sa vyžaduje logovanie výstupov volaného príkazu cron job-u, je potrebné pridať logovanie do */var/log/cron.log* podľa príklad č. 2 v prvom bode návodu.

Premenné prostredia

Pri spúšťaní skriptu, ktorý používa funkciu api cron manager-a nie sú dostupné premenné prostredia, nastavené pri budovaní a spúšťaní príslušného kontajnera pre fireant backend. Preto je potrebné v shell skripte (bod 2. b.) pridať načítanie premenných prostredia, ktoré sa počas spúšťania aplikácie exportujú.

Viac informácií: https://blog.emumba.com/access-docker-compose-environment-variables-inside-a-cron-job-b24e41cf33da

Dokumentácia frontendu systému

Inštalácia a konfigurácia

Tím č. 03 FireAnt

Vedúci tímu: Branislav Pecher, Ivan Srba

Autor dokumentácie: František Šefčík

Dátum vytvorenia: 10.11.2019

Viď tiež v repozitári fireant-frontend

Inštalácia a spustenie servera

1. Stiahnutie github fireant-frontend repozitára

   ```
   git clone git@github.com:BranislavPecher/fireant-frontend.git
   ```

2. Následne môžeme server pustiť v dockeri alebo lokálne po inštalácií všetkých dependencies

Spustenie serveru v dockeri

Server je možné spustiť príkazom

```
docker-compose up --build -d
```

Premenné prostredia môžu byť nastavené v .env súbore.

Nastavenie pre staging a produkciu:

```
# .env file for staging or production environment
FIREANT_ENV=staging
```

Pre nastavenie development serveru v dockeri pozri článok Dockerizing an Angular App.

Spustenie serveru lokálne

Najprv je potrebné nainštalovať dependencies:

1. Node.js verzie 12.x
2. npm package manager
3. Angular cli verzie 8
4. Inštalácia potrebných knižníc pomocou spustenia nasledovných príkazov z priečinka projektu

```
npm install
npx npm-force-resolutions  # see https://github.com/angular/angular-cli/issues/17262 for more
details
npm install
```

Následne môžeme spustiť development server príkazom

```
ng serve
```

Staging alebo production server môžeme spustiť príkazom (dosadiť namiesto env požadované prostredie)

```
ng serve -c {env}
```

Dokumentácia

Tím č. 03 FireAnt

Vedúci tímu: Branislav Pecher, Ivan Srba

Autor dokumentácie: Vajk Pomichal

Dátum vytvorenia: 9.11.2019

FireAnt ponúka webové rozhranie, ktoré je implementované v platforme pre webové aplikácie Angular.

Vo webovej aplikácií je použitý Material Dashboard Angular template.

Štruktúra projektu

Angular projekt obsahuje jednu aplikáciu, ktorá volá ostatné moduly v aplikácií. Pre každú samostatnú stránku je vytvorený modul, ktorý obsahuje komponenty, modely a služby, ktoré sú využívané v tomto module. Prvky, ktoré sú využívané na viacerých stránkach sú umiestnené v mudule **shared.** Smerovanie dopytov v aplikácií rieši modul **admin-layout**, podľa url adresy načíta príslušný modul stránky.

Webová aplikácia je rozdelená do nasledujúcich modulov:

1. Admin layout module
2. Checker module
3. Dashboard module
4. Navigation module
5. RASA chatbot module
6. Search-filter module
7. Shared module

Admin layout module

Tím č. 03 FireAnt

Vedúci tímu: Branislav Pecher, Ivan Srba

Autor dokumentácie: Vajk Pomichal

Dátum vytvorenia: 20.04.2020

Účel modulu

Tento modul rieši smerovanie dopytov v rámic aplikácii a zároveň je aj najvyšším prvkom v hierarchii modulov.

Checker

Tím č. 03 FireAnt

Vedúci tímu: Branislav Pecher, Ivan Srba

Autor dokumentácie: Vajk Pomichal

Dátum vytvorenia: 9.11.2019

Účel modulu

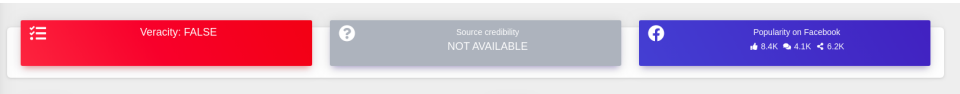
Tento modul rieši funkcionalitu checkera - teda vyhľadávanie na základe zadaného výrazu v databáze Monantu. Po nájdení dát sa zobrazie detailné informácie ohľadom článku, zdroja alebo claimu.

Widgety

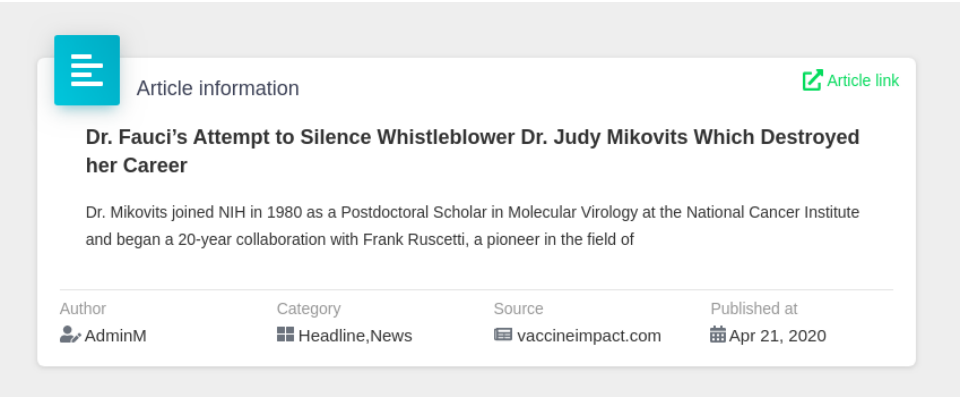
Pre každý typ entity (článok, zdroj, claim) sa zobrazujú rôzne informácie vo forme widgetov.

Widgety pre články

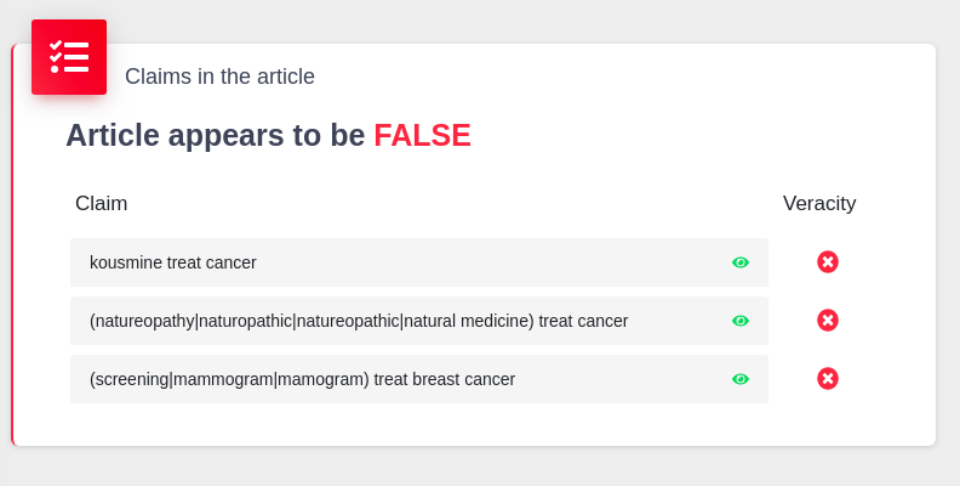
1. **Overview -** obsahuje prehľadové údaje o danom článku s možnosťou prekliknutia sa na daný widget ze účelom získania ďalších detailov



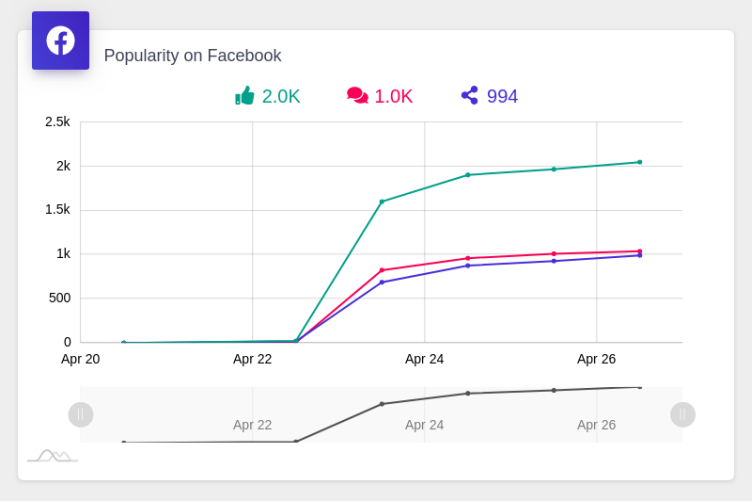
2. **article-metadata** - widget zobrazuje informácie o článku. (titulok, perex, link, autor, kategoriu, zdroj, dátum publikácie)



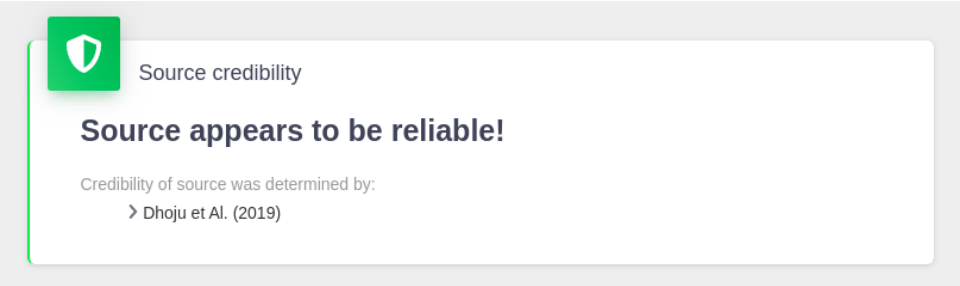
3. **claims-in-article-widget** - widget zobrazuje zoznam claimov, ktoré sa v článku nachádzajú. Ku každému tvrdeniu sú priradené ohodnotenie jeho pravdivosti. Z hodnotení všetkých tvrdení v článku je odvodené aj celkové hodnotenie pre článok.



4. **facebook-feedback-widget** - widget zobrazuje graf vývoja popularity článku na Facebooku. Zobrazuje tri údaje o počte zdieľaní, komentárov a lajkov v čase.



5. **source-credibility-widget** - widget zobrazujúci údaje o dôveryhodnosti zdroja článku (rovnaký widget, ako pre samotné zdroje, viď nižšie)

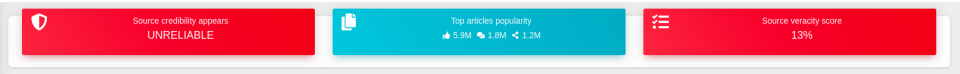


6. **questions-widget -** widget obsahujúci otázky a odpovede expertov ohľadom daného článku

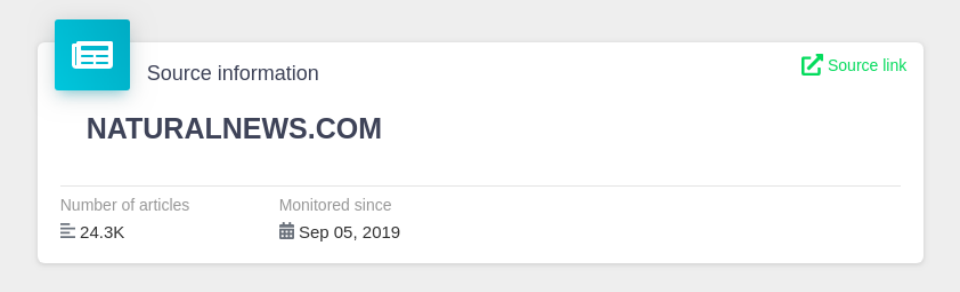
   (obrazok TBA Dominik Hančuľák)

Widgety pre zdroje

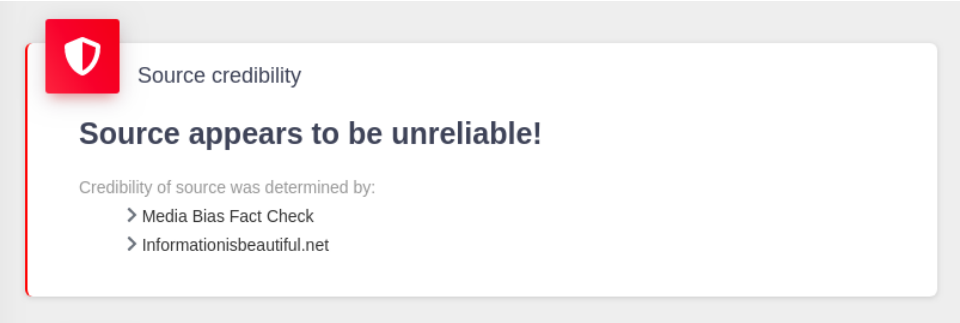
1. **Overview -** obsahuje prehľadové údaje o danom zdroji s možnosťou prekliknutia sa na daný widget ze účelom získania ďalších detailov



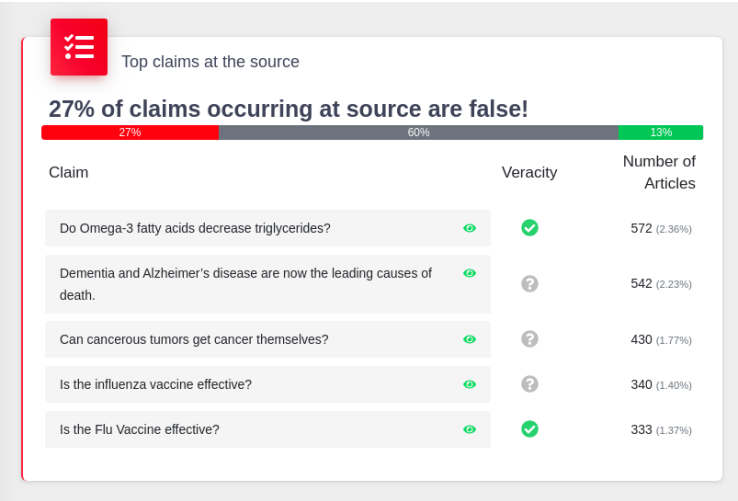
2. **source-metadata -** obsahuje informácie o zdroji (počet článkov, dátom pridania do databázy, ...)



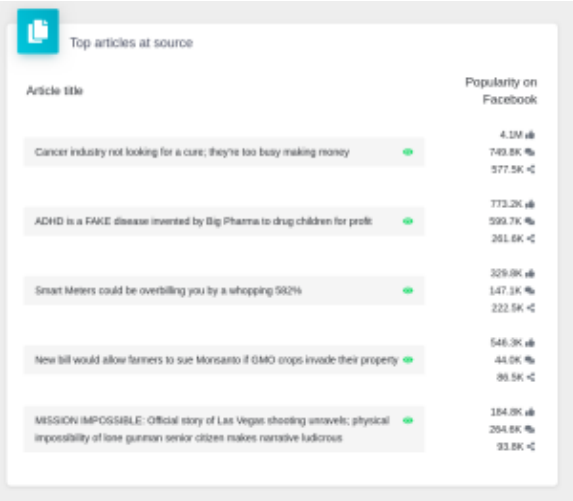
3. **source-credibility-widget** - tento widget obsahuje kartu, ktorá vykresľuje či je zdroj vierohodný alebo nie. (rovnaký widget je aj pri článkoch ohľadom zdroja)



4. **top-claims-at-source -** widget obsahujúci kartu s claims, ktoré sa najviac šíria v článkov z daného zdroja, poskytuje aj informáciu o podielu dôveryhodnosti claimov pre daný zdroj



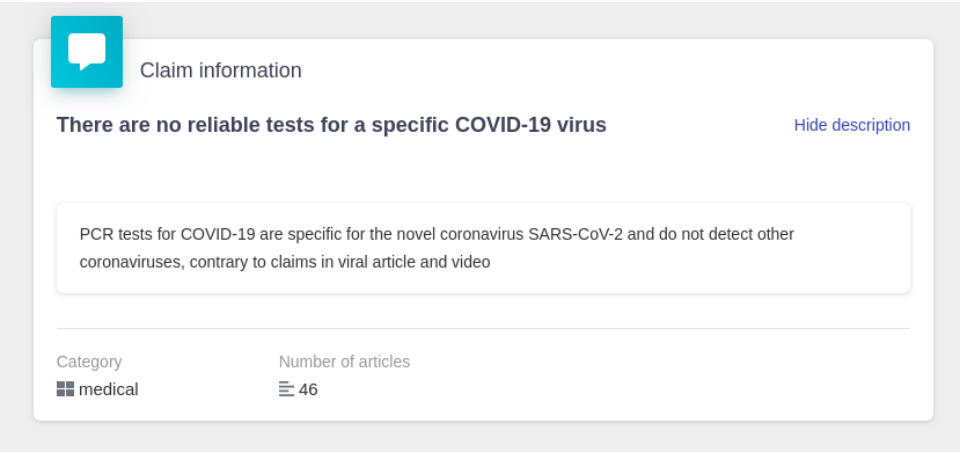
5. **top-articles-widget -** widget obsahujúci kartu, ktorá obsahuje najviac populárne články daného zdroje podľa ich popularity na Facebooku



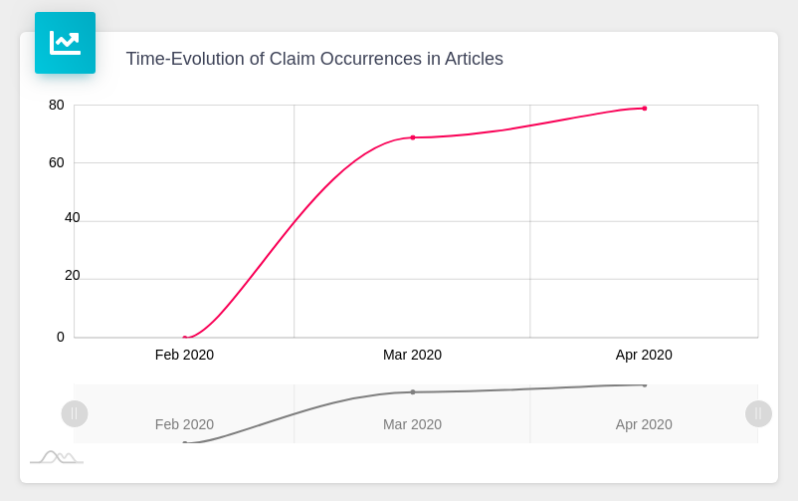
Widgety pre claims

1. **overview -** obsahuje prehľadové údaje o danom claime s možnosťou prekliknutia sa na daný widget ze účelom získania ďalších detailov

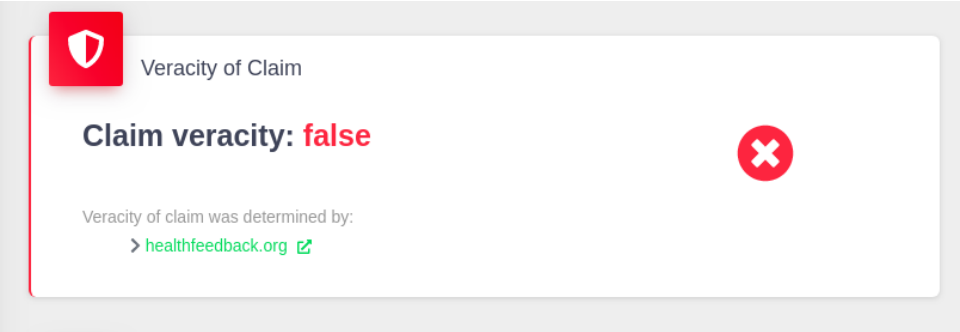
2. **claim-metadata-widget -** komponent obsahujúci kartu s metadátami o danom claim-e



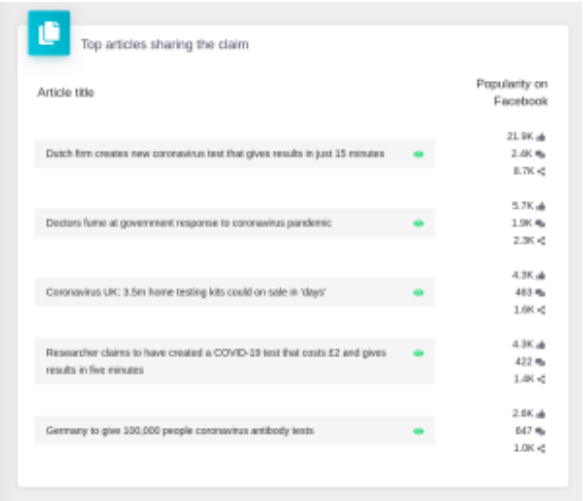
3. **claim-in-articles-count-widget** - komponent obsahuje kartu s grafom, ktorá zobrazuje vývoj počtu článkov v čase, ktoré obsahujú daný claim



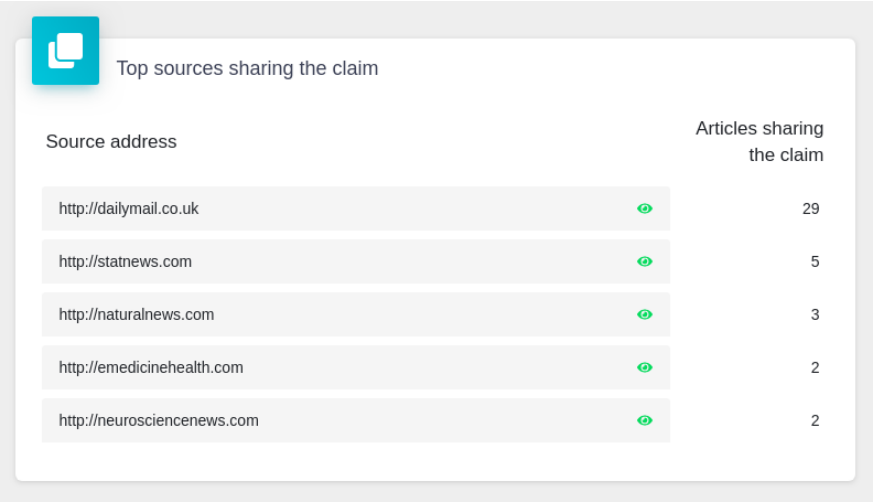
4. **claim-veracity-widget -** komponent obsahuje kartu s informáciami ohľadom dôveryhodnosti claim-u a zoznam expertov, ktorí daný claim overili



5. **top-articles-widget -** komponent obsahujúci kartu s najpopulárnejšími článkami (podľa popularity na Facebook), ktoré zdieľajú daný claim (rovnaký widget, ako pri zdrojoch)

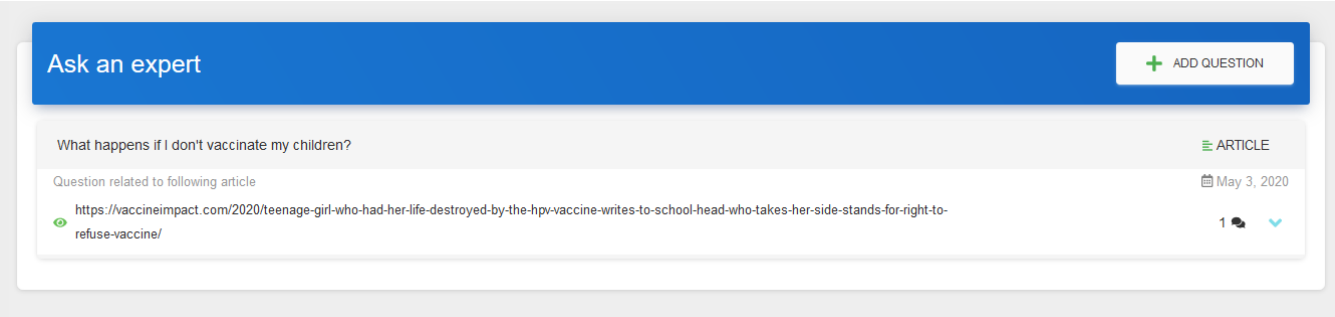


6. **top-sources-for-claim-widget -** komponent obsahujúci kartu s najpopulárnejšími zdrojmi claimy (podľa počtu článkov, ktoré zdieľajú daný claim)



Zdielané widgety

1. **questions-widget -** komponent obsahujúci otázky pre expertov, s možnosťou pre expertov editovať a odpovedať na dané otázky. Pomocou tlačidla "ADD QUESTION" môže používateľ položiť novú otázku expertovi.



Dashboard

Tím č. 03 FireAnt

Vedúci tímu: Branislav Pecher, Ivan Srba

Autor dokumentácie: Vajk Pomichal

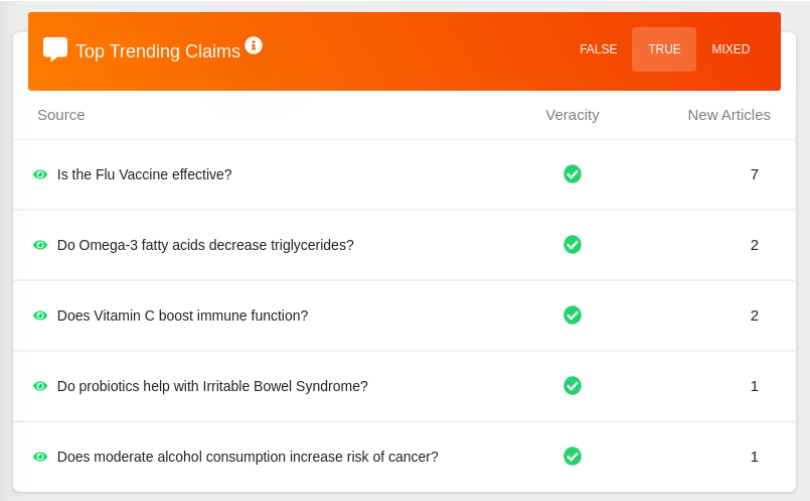
Dátum vytvorenia: 1.11.2019

Účel modulu

Modul slúži ako landing page pre prichádzajúcich používateľov, umožňuje priamo vyhľadávať výrazy (po vyhľadaní výrazu je používateľ presmerovaný na stránnku v rámci Checker module).

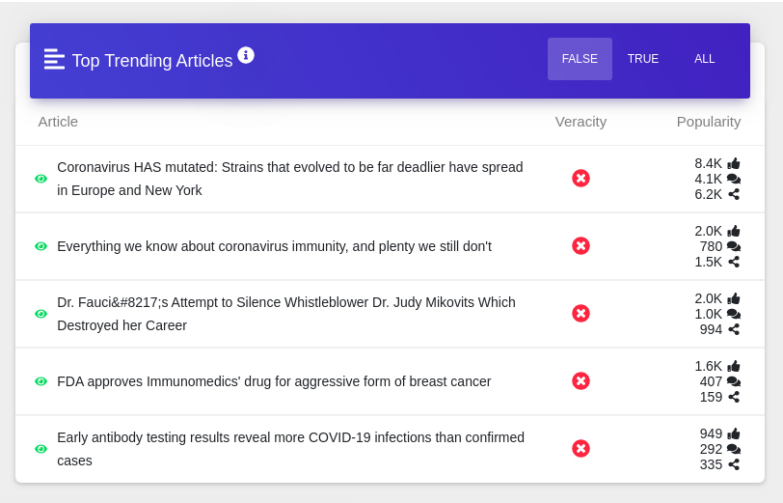
Taktiež obsahuje widgety, ktoré zobrazujú najviac populárne články, zdroje a claimy, pomocou troch "tab"-ov pre každý widget pre falošné, pravdivé resp všetky entity

Top Trending Claims



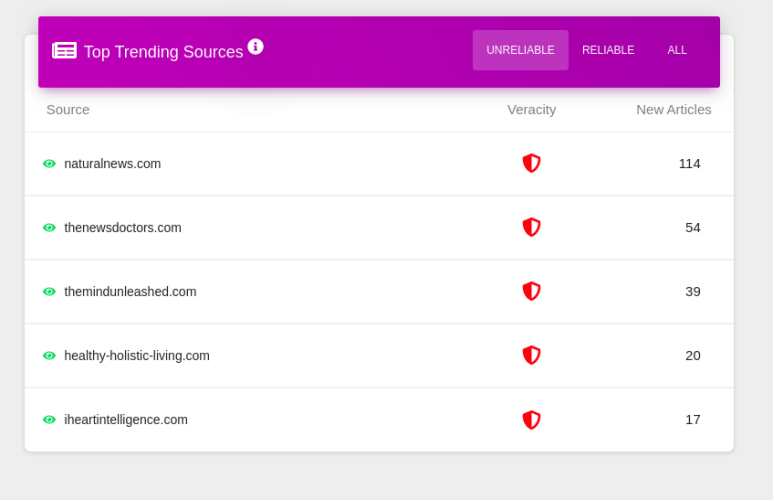
Obsahuje najviac populárne claimy, zoradené podľa počtu článkov, ktoré ich zdielajú za posledný týždeň. Obsahuje zvlášť zoznamy len pre falošné, pre pravdivé claimy ako aj pre claimy so zmiešanou pravdivostnou hodnotou.

Top trending articles



Obsahuje najviac populárne články, zoradené podľa počtu reakcií (súčet share, like, comment) na Facebooku. Obsahuje zvlášť zoznam pravdivých a nepravdivých článkov ako aj zoznam, kde sa nachádzajú všetky články.

Top trending sources



Obsahuje najviac populárne zdroje, zoradené podľa počtu nových článkov za posledný týždeň. Obsahuje zvlášť zoznam pravdivých a nepravdivých zdrojov ako aj zoznam, kde sa nachádzajú všetky zdroje.

Navigation module

Tím č. 03 FireAnt

Vedúci tímu: Branislav Pecher, Ivan Srba

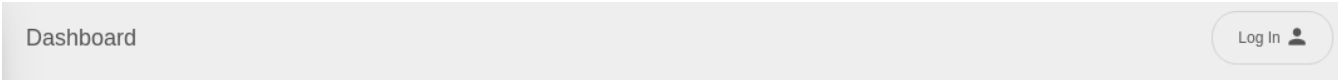
Autor dokumentácie: Vajk Pomichal

Dátum vytvorenia: 20.04.2020

Účel modulu

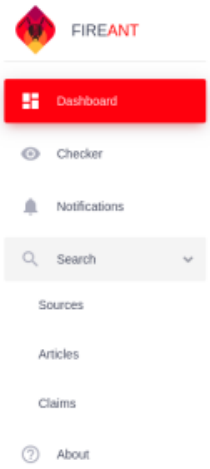
Tento modul obsahuje vrchný a bočný panel a rovnako aj footer.

Vrchný panel



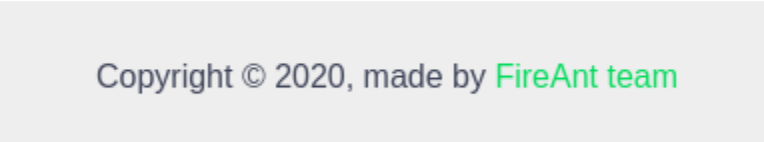
Vrchný panel obsahuje informáciu o súčasnej stránke a umožňuje používateľovi prihlásiť sa.

Bočný panel



Bočný panel umožňuje navigáciu po podstránkach aplikácie.

Footer



Obsahuje odkaz na stránku tímového projektu.

RASA chatbot

Tím č. 03 FireAnt

Vedúci tímu: Branislav Pecher, Ivan Srba

Autor dokumentácie: Vajk Pomichal

Dátum vytvorenia: 17.03.2020

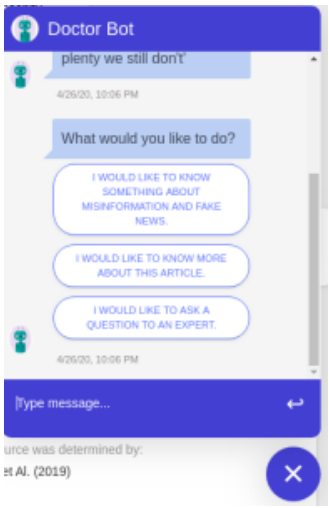
Účel modulu

Tento modul slúži na komunikáciu používateľov s RASA chatbotom.

Vzhľad

Zdroj: https://github.com/contribution-jhipster-uga/angular-chat-widget-rasa

Chat okienko sa otvorí po kliknutí na tlačidlo s otáznikom v pravom dolnom rohu obrazovky.



Chatbot využíva možnosti RASA chatbota. Dokáže zobrazovať tlačidlá na predpripravené odpovede, posielať voľný text a aj zobrazovať obrázky v rámci chat okienka.

Search-Filter

Tím č. 03 FireAnt

Vedúci tímu: Branislav Pecher, Ivan Srba

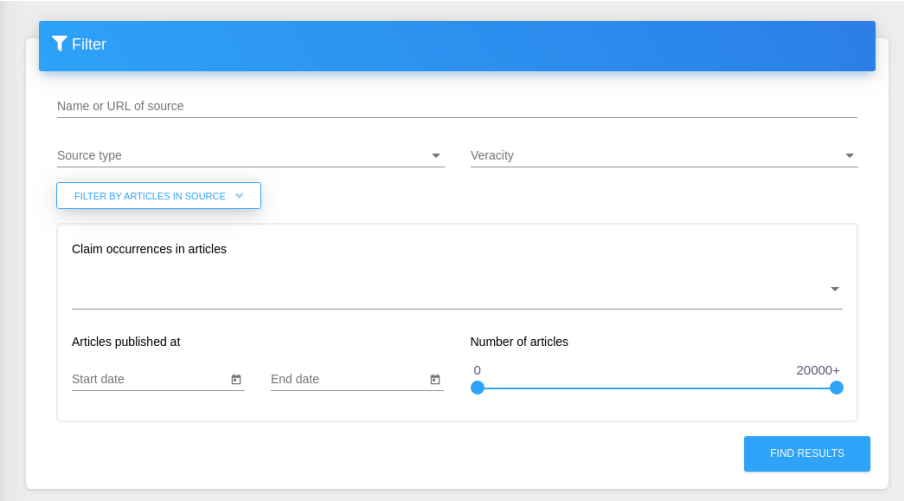
Autor dokumentácie: Vajk Pomichal

Dátum vytvorenia: 17.02.2020

Účel modulu

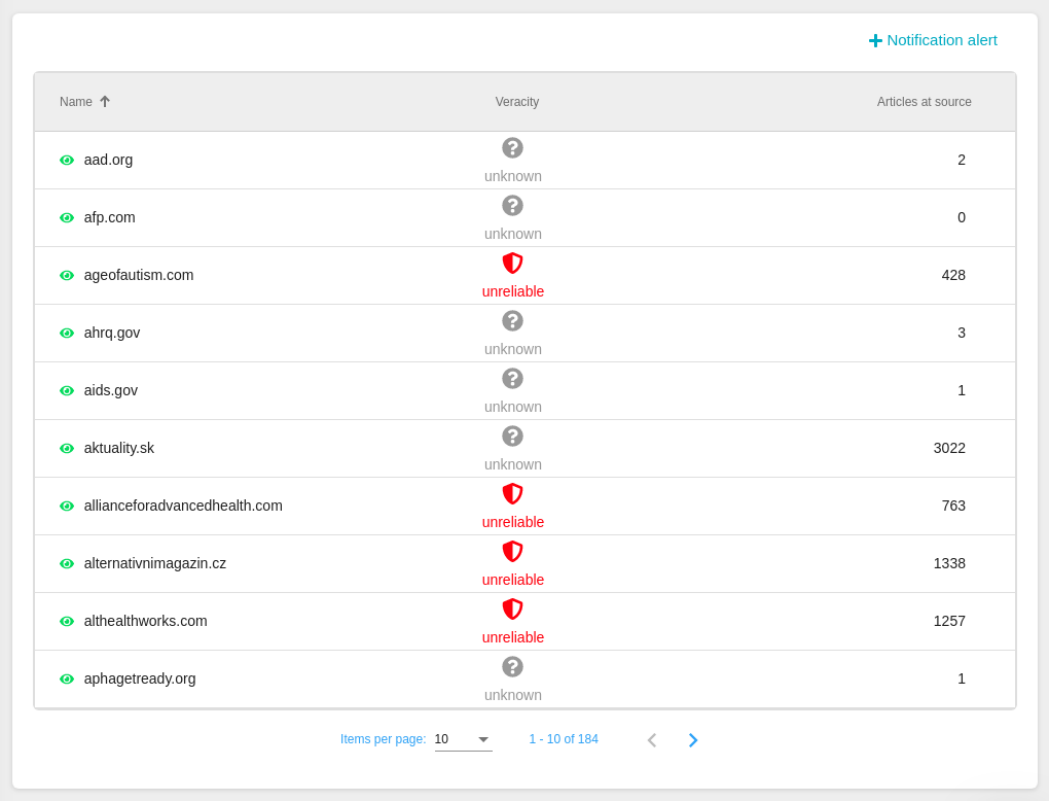
Modul slúži na vyhľadávanie zdrojov, článkov a claim-ov podľa rôznych parametrov a zároveň na vytváranie notifikácií na základe vyfiltrovaných entít (viď opis notifikácií v Shared module).

Sources filter

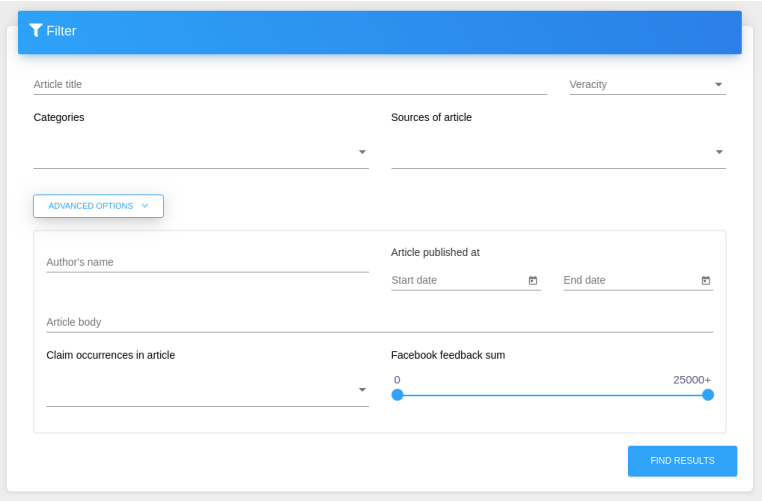


Umožňuje filtrovanie zdrojov podľa názvu/url, podla typu zdroja (fact checking website, blog atď.), podľa dôveryhodnosti zdroja. Zároveň poskytuje aj pokročilejší filter na základe článkov zverejnených na zdroji: podľa claimov objavujúcich sa v článkoch, podľa dátumu zverejnenia článkov (možné nastaviť min aj max hodnotu), alebo podľa počtu článkov na zdroji (pri filtrovaní musia platiť všetky podmienky zároveň, ak sú zadané).

Výsledky sa zobrazujú v tabulke, ktorá má pagination a sorting na backende.

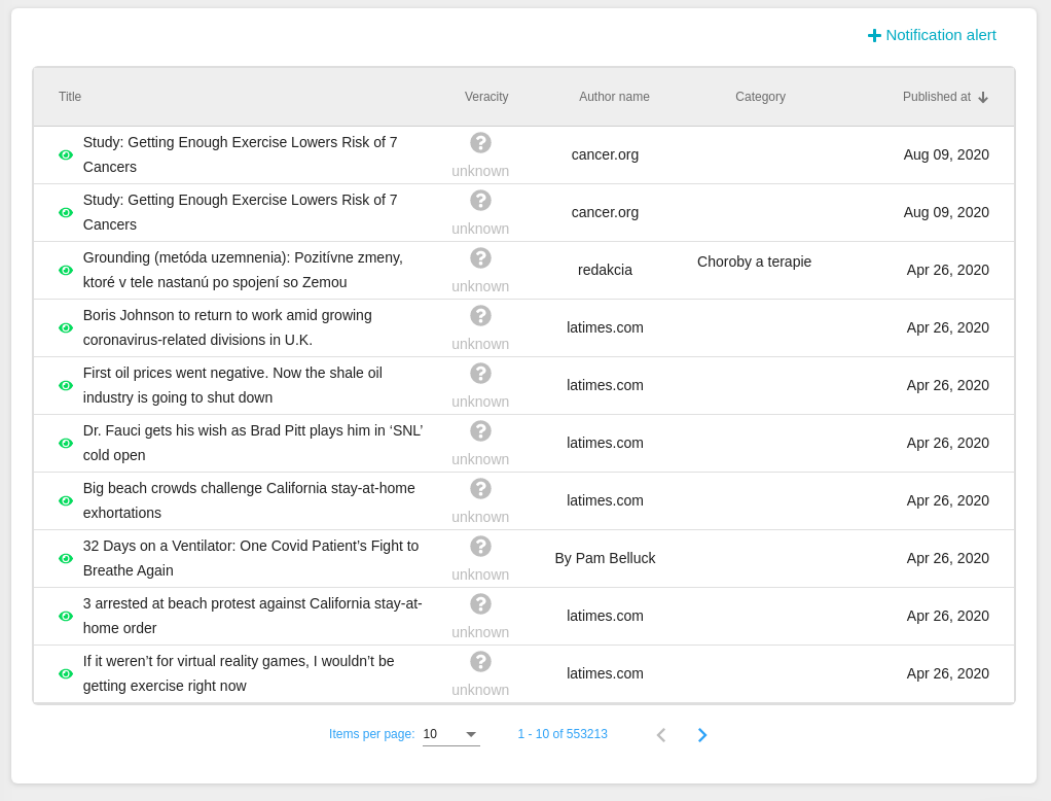


Articles filter

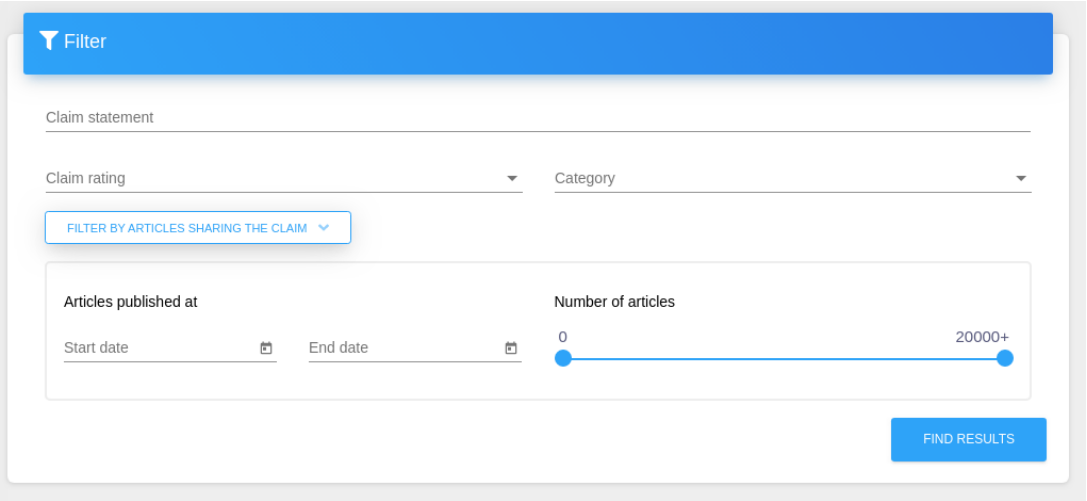


Umožňuje vyhladávať články podľa ich názvov, podľa dôveryhodnosti, podľa rôznych kategórií (kategórie sú dynamicky a automaticky dopĺňané z backendu), podľa zdroju článku (tiež automaticky dopĺňané z backendu), podľa autora, podľa obsahu článku, podľa dátomu zverejnenia (možnost udať minimálny a maximálny dátum), podľa výskytu claimov v článku a podľa súčtu reakcií na Facebooku (like, share comment).

Výsledky sa zobrazujú v tabulke, ktorá má pagination a sorting na backende.



Claims filter



Umožňuje vyhľadávanie claimov podľa ich znenia, podľa hodnotenie (pravdivé, nepravdivé, ...), podľa kategórie a podľa dátumu vzniku článkov, v ktorých sa objavovali a podľa počtu článkov v ktorých sa objavovali (v prípade, že je zadaný aj dátum a aj počet, tak sa do počtu rátajú len články za daný časový úsek.

Zdieľané prvky

Tím č. 03 FireAnt

Vedúci tímu: Branislav Pecher, Ivan Srba

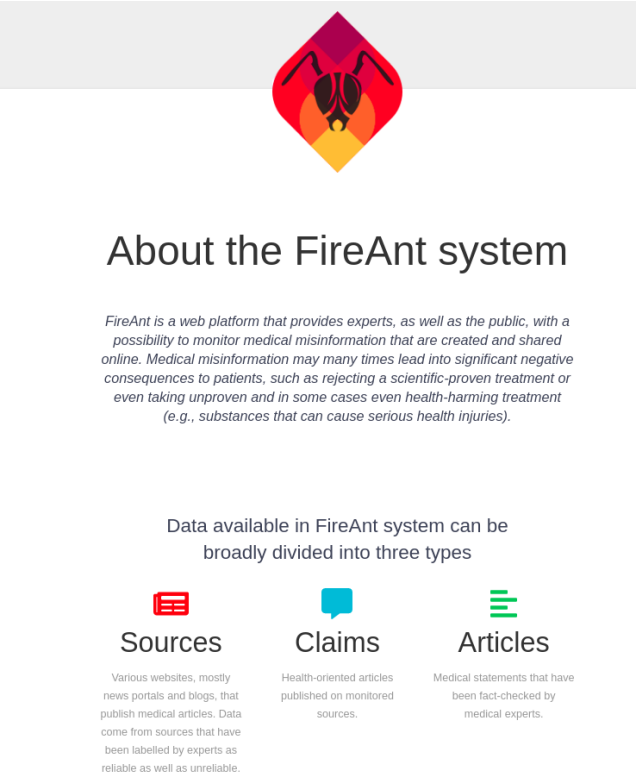
Autor dokumentácie: Vajk Pomichal

Dátum vytvorenia: 1.11.2019

Účel modulu

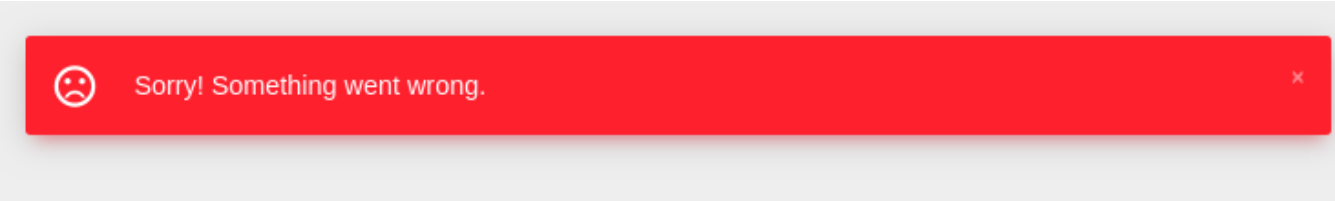
Tento modul obsahuje prvky, ktoré sú vzdielané medzi viacerými modulmi a komponenty, ktoré rozsahom boli príliš malé na to, aby boli vložené do samostatného modulu.

About page



Obsahuje informácie ohľadom aplikácie pre verejnosť, nachadzajú sa tam informácie o funkcionalitách ako aj promo video a obrázky jednotlivých obrazoviek.

Alert



Poskytuje chybové (prípadne aj iné) hlášky používateľom.

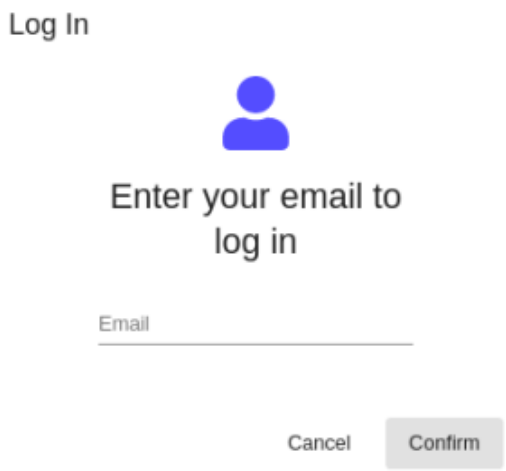
Confirmation dialog

Jednoduchý popup, ktorý slúži na potvrdenie niektorých operácií (ako napríklad vymazanie notifikácie)

Image dialog

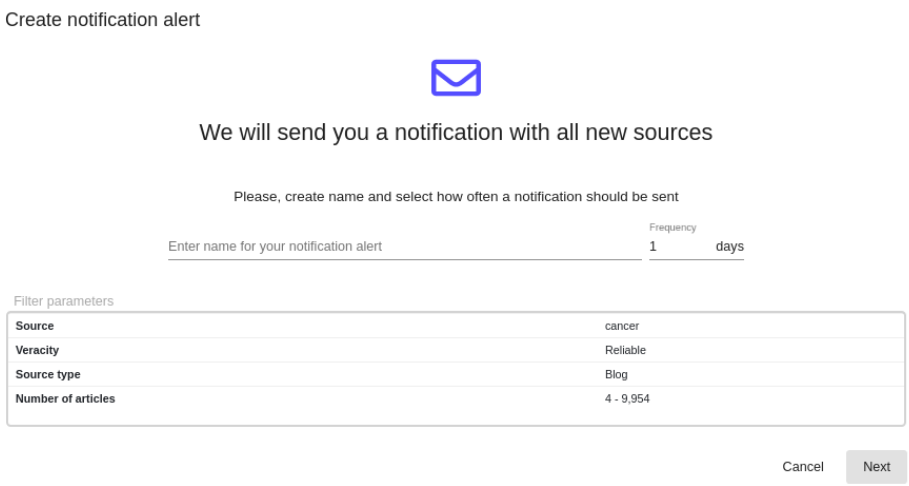
Slúži na zobrazenie obrázkov v popupe po kliknutí na nich, využívané na About page.

Login



Dialógové okno na prihlásenie sa používateľa. Pre prihlásenie používateľ musí zadať email, na ktorý obdrží link, cez ktorý sa môže prihásiť do aplikácie. Doba platnosti daného tokenu je obmedzený, po vypršaní je potrebné prihlásenie zopakovať znovu

Notification form



Dialógové okno, ktoré slúži na vytvorenie notifikácii o pribudnutí nových entít do predom nastavených filtrov v Search-filter module. Notifikáciu je potrebné pomenovať a nastaviť frekvenciu odosielania mailov. Následne používateľ obzdrží potvrdzovací email a následne je v pravidelných intervaloch upozornení o pribudnutí nových entít vo filtri, ktorý vytvoril (a ktoré má možnosť aj filtrovať samostatne, t.j. zobrazovať iba novopribudnuté entity)

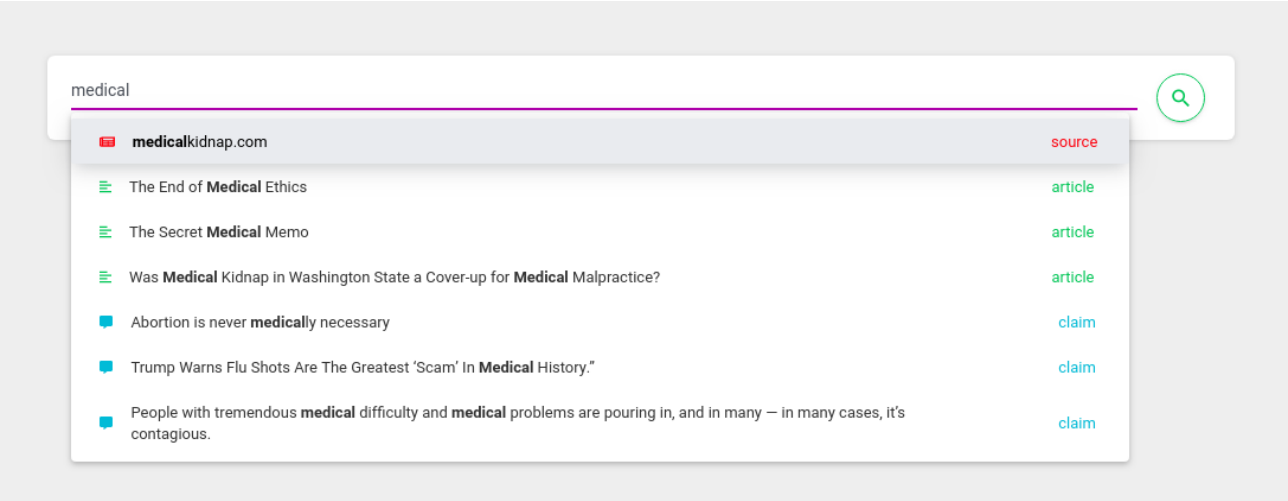
Notifications

Stránka zobrazujúca všetky notifikácie používateľa, ktorá zároveň umožňuje editovanie, vymazávanie a pridávanie nových notifikácií, prípadne dočasné pozastavenie posielanie emailov pre notifikácie. Z tejto stránky je možné prekliknutie sa na stránku filtrov a zobraziť v nich len novopribudnuté entity.

Page not found

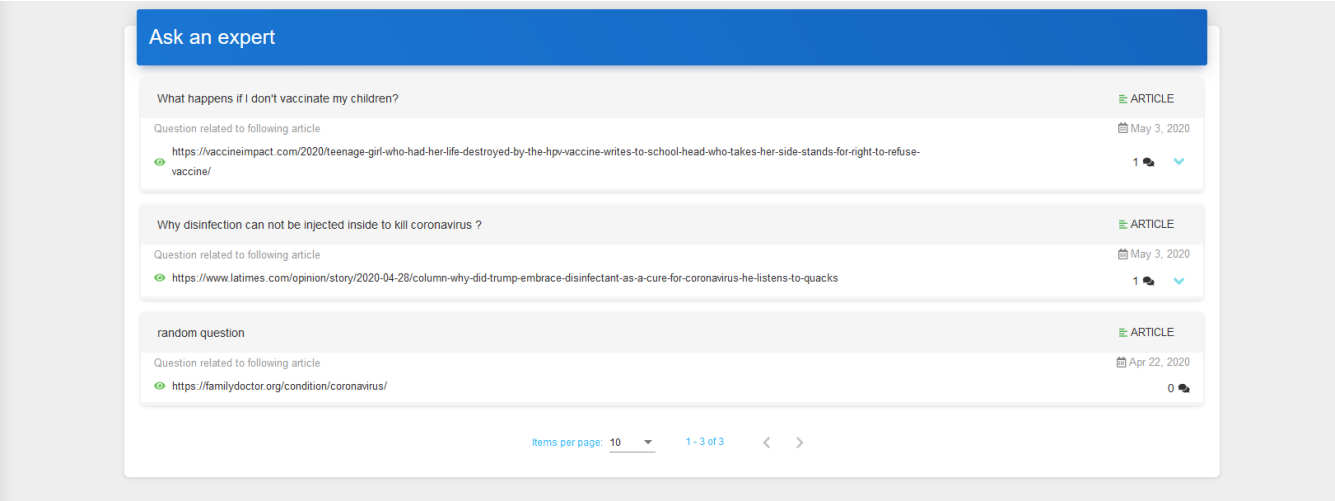
Zobrazí sa v prípade, že používateľ zadá nesprávnu adresu v rámci aplikácie.

Search



komponent vyhľadávacieho okienka zobrazený v rámci Dashboard module a Checker module. Slúži na zadávanie url adries zdrojov, článkov alebo na zadávanie slovných spojení a viet (claims). Pomocou autocomplete-service sa používateľovi zobrazia návrhy na vyhľadávanie podľa už napísaného textu.

Questions



Stránka zobrazujúca všetky položené otázky používateľov expertom so stránkovaním, ktorá zároveň umožňuje expertom na dané otázky odpovedať a editovať. Z tejto stránky je možné prekliknutie sa do checkera s danou entitou.

Dokumentácia inštalácie MonAnt systémov

Inštalácia centrálneho úložiska

Tím č. 03 FireAnt

Vedúci tímu: Branislav Pecher, Ivan Srba

Autor dokumentácie: @Miroslav Sumega

Dátum vytvorenia: 18.11.2019

Inštalácia centrálneho úložiska

Nastavenie používateľa databázy na postgres

V súbore docker-compose.yml zmeňte v **MONANT_POSTGRES_TEST_HOST=${MONANT_POSTGRES_TEST_HOST:-localhost}** na "postgres_staging_con". V prípade, že nefungujú testy je potrebné vypnúť aj Elasticsearch zmenením **MONANT_ES_ENABLED=${MONANT_ES_ENABLED:-true}** na "false". Ďalšie riešenie je exportovať premenné **MONANT_POSGRES_TEST_HOST** a **MONANT_ES_ENABLED** pomocou .env súboru.

Namountovanie git repozitára do docker app volume

Aby bolo možné commitnúť zmeny vykonané vnútri dockeru, je potrebné namountovať zložku, ktorá obsahuje naklonovaný repozitár do zložky **/app** v docker volume. Na to je potrebné do docker-compose.yml pridať do sekcie volumes v sekcii app riadok **"- ./:/app"**. V prípade, že kontajner sa nedá spustiť, namountujte do inej zložky, napr. **"-./:/app/repository"**. Avšak v takom prípade, ak boli v dockeri vykonané nejaké zmeny, treba ich vykonať aj v **/app/repository**, pretože inak po vypnutí kontajneru budú stratené.

Spustenie central storage s dockerom

Dockerfile a docker-compose.yml sú požadované na spustenie central storage.

Ak máte spustený na stroji postgres, je potrebné ho zastaviť, aby sa dokázal spustiť kontajner s postgres databázou. Použite príkaz: `sudo s ervice postgresql stop`

Na zastavenie compose kontajnerov použite `docker-compose stop`

Na odstránenie compose kontajnerov použite `docker-compose down`. V prípade, že kontajnery nie sú zastavené, ich zastaví a potom vymaže.

Na vytvorenie a spustenie kontajnerov použite `docker-compose up --build -d`. Ak nastane chyba, pravdepodobne nemáte vytvorený **stagi ng network**. V takom prípade zadajte príkaz `docker network create staging` a potom znova spustite docker-compose.

Ak chcete vidieť, ktoré kontajnery vám bežia, použite `docker ps`

Ak máte problémy s právami pri spúšťaní docker alebo docker-compose príkazov, skúštajte si pomocou sudo alebo pridajte používateľa, ktorý dané príkazy spúšťa do docker group pomocou príkazu `sudo usermod -aG docker user-name` .

Keď spúsťate docker prvý krát, budete musieť vytvoriť volumes pre postgresql a elasticsearch dáta. Odporúčané príkazy budú navrhnuté dockerom, ale mali by to byť nasledujúce dva:
- `docker volume create postgres-data-staging`
- `docker volume create es-data-staging`

Taktiež budete potrebovať vytvoriť databázu a používateľa pre túto databázu. Spustite nasledovné rpíkazy v uvedenom poradí:
- `docker exec -it {name_of_container} bash` kde `{name_of_container}` je postgresql kontajner
- `su - postgres`
- Vytvorenie používateľa `createuser monant_api`
- Vytvorenie databázy `createdb monant_api`
- Zapnutie psql `psql`
- Nastavenie hesla vytvorenému používateľovi `alter user monant_api with encrypted password 'monant_api';`. **POZOR, pri niektorých príkazoch ako auth_seed môže byť potrebné heslo "password". Ak vám tieto príkazy nepôjdu a bude vám vypisovať chyby súvisiace s autentifikáciou, skúste zmeniť heslo.**
- Nastavenie privilégií pre databázu používateľovi `grant all privileges on database monant_api to monant_api;`

Taktiež bude potrebný dummy používateľ, je možné ho vytvoriť spustením:
- `docker exec -it {name_of_container}` kde `{name_of_container}` je central storage kontajner

- `python -m api.manage db upgrade` spustí migrácie, čo je potrebné na úspešné vykonanie nasledujúceho príkazu
- `python -m api.manage auth_seed`

Vytvorený dummy používateľ má nasledovné údaje:
Login: **user@fiit.sk**
Heslo: **password**

Testovacia databáza

Ak chcete spúšťať testy, musíte maunálne vytvoriť databázu **monant_api_test**. Nie je potrebné spúšťať migrácie, nakoľko sa spúšťajú automaticky pri spustení testov. Avšak môže sa stať, že testy nebudú chcieť zbehnúť, lebo bude chýbať nejaká tabuľka, zmenila sa migrácia a nie je možné ju zvrátiť (angl. revert), prípadne sú problémy s prístupovými právami ku endpointom, je potrebné testovaciu databázu vymazať a vytvoriť nanovo.

DevOps dokumentácia

Inštalácia a konfigurácia webového serveru

Tím č. 03 FireAnt

Vedúci tímu: Branislav Pecher, Ivan Srba

Autor dokumentácie: Denis Mitana

Dátum vytvorenia: 6.11.2019

Úvod

Táto dokumentácia rieši inštalovanie a konfigurovanie webového serveru Apache 2 spolu s nastavením HTTPS prístupu.

Nasadenie HTTPS

Na nasadenie HTTPS použijeme nástroj Certbot.

1. Nainštalujeme nástroj Certbot.

   ```
   $ sudo apt-get update
   $ sudo apt-get install software-properties-common
   $ sudo add-apt-repository universe
   $ sudo add-apt-repository ppa:certbot/certbot
   $ sudo apt-get update
   $ sudo apt-get install certbot python-certbot-apache
   ```

2. Vytvoríme HTTPS certifikát pre doménu *team03-19.studenti.fiit.stuba.sk*. Keďže *Apache* bude bežať v *docker kontajneri*, tak vytvoríme iba certifikáty (možnosť *certonly*) a nepoužijeme žiadny webový server (možnosť *--standalone*).

   ```
   $ sudo certbot certonly --standalone -d team03-19.studenti.fiit.stuba.sk
   ```

3. Otestujeme automatické obnovovanie certifikátov. Nasledujúci príkaz by mal zbehnúť bez chýb.

```
$ sudo certbot renew --standalone --dry-run
```

4. Upravíme certbot systemd service, ktorý bude za nás automaticky obnovovať certifikáty (pomocou systemd timer). Keďže Apache bude bežať v docker kontajneri, tak musíme pred obnovením kontajner vypnúť a potom zapnúť. Ak by sme to nespravili, tak by sme sme skončili s chybou **Problem binding to port 80: Could not bind to IPv4 or IPv6**.
   V súbore /lib/systemd/system/certbot.service pridáme (ExecStartPre, ExecStartPost) a upravíme (ExecStart):

   ```
   ExecStartPre=/usr/bin/docker stop webserver
   ExecStart=/usr/bin/certbot -q --standalone renew
   ExecStartPost=/usr/bin/docker start webserver
   ```

   Následne vykonáme príkazy:

   ```
   $ sudo systemctl daemon-reload
   $ sudo systemctl enable --now certbot.timer
   ```

Inštalácia a konfigurácia webového serveru Apache 2

Webový server *Apache* využívame z *docker kontajneru*.

1. Vytvoríme zložky pre konfigurácie webového serveru.

   ```
   $ sudo mkdir -p /var/lib/apache2/conf/extra
   ```

2. Vytvoríme súbory *httpd.conf* v zložke */var/lib/apache2/conf* a *httpd-ssl.conf* v zložke */var/lib/apache2/conf/extra*.
   V súbore *httpd.conf* sú potrebné konfiguračné nastavenia ako reverse proxy alebo nutné moduly.
   V súbore *httpd-ssl.conf* sú potrebné konfiguračné nastavenia pre fungovanie HTTPS.
3. Spustíme *docker kontajner* kde bude bežať náš webový server so všetkými konfiguráciami a HTTPS, ktoré sme vytvorili skôr. Do kontajnera musíme namapovať vytvorené konfiguračné súbory ako aj HTTPS certifikáty.

   ```
   $ docker run -dit --name webserver -p 80:80 -p 443:443 -v /var/www/fireant-teamweb/:/usr/local
   /apache2/htdocs/ -v /var/lib/apache2/conf/httpd.conf:/usr/local/apache2/conf/httpd.conf -v /var/lib
   /apache2/conf/extra/httpd-ssl.conf:/usr/local/apache2/conf/extra/httpd-ssl.conf -v /etc
   /letsencrypt/:/usr/local/apache2/letsencrypt/ httpd
   ```

Inštalácia Confluence

Tím č. 03 FireAnt

Vedúci tímu: Branislav Pecher, Ivan Srba

Autor dokumentácie: Miroslav Sumega

Dátum vytvorenia: 3.11.2019

Najskôr je potrebné stiahnuť Confluence docker image:

```
$ docker pull atlassian/confluence-server
```

Následne, ak ešte nie je vytvorené, je potrebné vytvoriť miesto, kde bude Confluence ukladať dáta:

```
$ docker volume create --name confluenceVolume
```

Nakoniec je potrebné spustiť docker container, ktorý bude využívať **atlassian/confluence-server** a bude ukadať údaje do **confluenceVolume**.

Je potrebné nastaviť premenné prostredia:

- **ALT_TOMCAT_CONTEXTPATH= /confluence** - správne fungovanie reverse proxy
- **ATL_PROXY_NAME=team03-19.studenti.fiit.stuba.sk** - správne fungovanie reverse proxy
- **ATL_PROXY_PORT=443** - ak sa používa HTTPS
- **ATL_TOMCAT_SCHEME=https** - ak sa používa HTTPS
- **ATL_TOMCAT_SECURE=true** - ak sa používa HTTPS
- **JVM_MINIMUM_MEMORY=1g** - minimum pamäte RAM
- **JVM_MAXIMUM_MEMORY=1g** - maximum pamäte RAM

Spúšťa sa nasledovným príkazom:

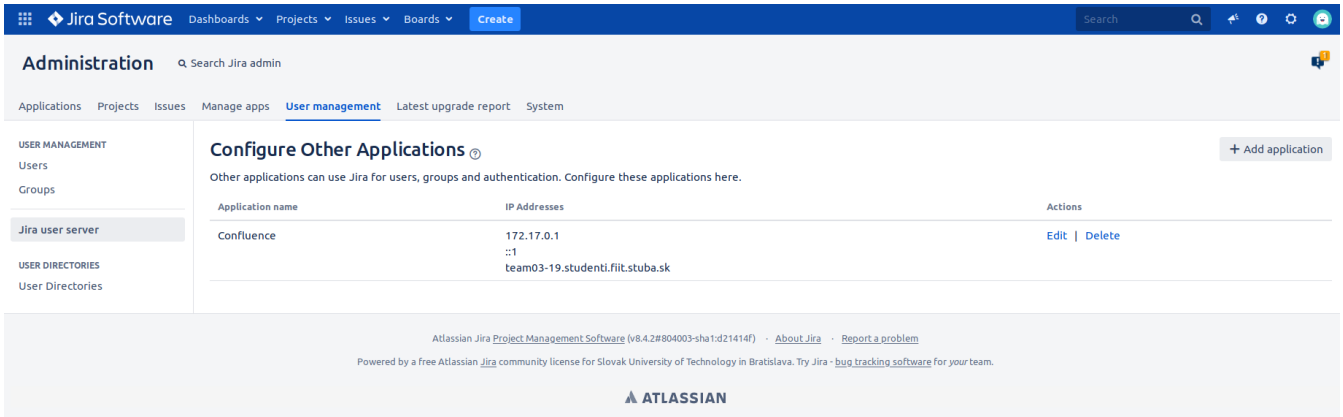
```
$ docker run -v confluenceVolume:/var/atlassian/application-data/confluence --name="confluence" -d -p
8090:8090 -p 8091:8091 -e ATL_TOMCAT_CONTEXTPATH=/confluence -e ATL_PROXY_NAME=team03-19.studenti.fiit.
stuba.sk -e ATL_PROXY_PORT=443 -e ATL_TOMCAT_SCHEME=https -e ATL_TOMCAT_SECURE=true -e
JVM_MINIMUM_MEMORY=1g -e JVM_MAXIMUM_MEMORY=1g atlassian/confluence-server
```

Prepojenie Confluence na Jira User Server

V prípade, že chceme prepojenie používateľov z Jiry aj v Confluence, potrebujeme nastaviť Jira User Server.

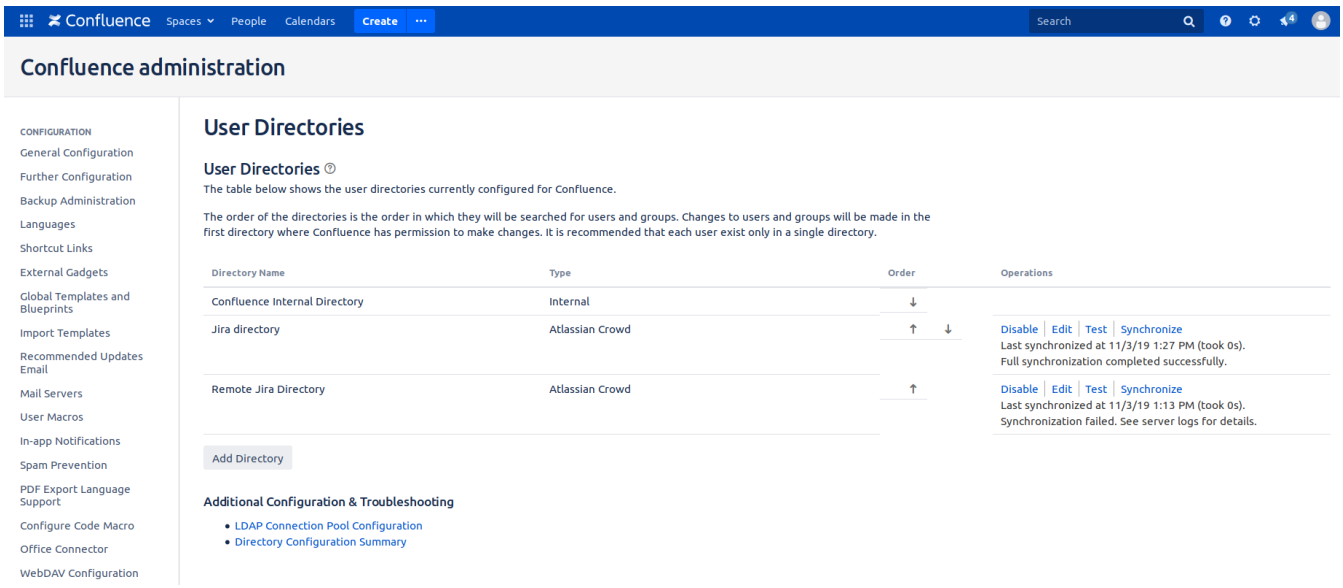
Nastavenie Jira user server

Najskôr potrebujeme nastaviť Jira User Server v Jira nastaveniach. Sú potrebné administrátorské práva. Klikneme *Add application* a nastavíme *meno*, *heslo* a *ip adresu*. Správne nastavenie vidíme na obrázku:



Nastavenie na strane Confluence

Najskôr potrebujeme pridať nastavený Jira User Server. Môžeme tak urobiť v nastaveniach Confluence (opäť treba mať administrátorské práva), v sekcii **User Directories**. Klikneme *Pridať Directory*. Po vyplnení údajov (zobrazuje obrázok Nastavenie Jira User Server v Confluence) by sme mali dosiahnuť stav, ktorý vidíme na obrázku (pridával sa directory s názvom *Jira directory*):



Pre správne nastavenie Jira User Server, potrebujeme zadať nasledovné údaje:

Configure Atlassian Crowd Server

Configure Atlassian Crowd Server

Confluence can use Crowd for user management: users, groups and authentication.

Server Settings

Name*	Jira directory
Server URL*	https://team03-19.studenti.fiit.stuba
	for example, http://www.example.com:8095/crowd/
Application Name*	Confluence
	Used to authenticate to the user management server.
Application Password*	••••••••••••••••••••••••••
	Used to authenticate to the user management server.
Connection Timeout (milliseconds)	5000
	Time to wait when opening a new server connection (default: 5000).
Max Connections	20
	Maximum number of concurrent connections to the remote Crowd server (default: 20).
Proxy Host	
	Hostname of the http proxy server.
Proxy Port	
	Port of the http proxy server.
Proxy Username	
	Username for connecting to the http proxy server.
Proxy Password	
	Password for connecting to the http proxy server.

Po zadaní údajov treba ešte zmeniť **Crowd Permissions** na *Read/write*. Po skončení doplnení treba otestovať pripojenie a ak získame hlášku so zeleným pozadím o úspešnom pripojení, môžme dáta uložiť. Po tomto kroku máme v Confluence dostupných používateľov z Jira aplikácie.

Inštalácia Jiry

Tím č. 03 FireAnt

Vedúci tímu: Branislav Pecher, Ivan Srba

Autor dokumentácie: Miroslav Sumega

Dátum vytvorenia: 2.11.2019

Najskôr je potrebné stiahnuť Jira docker image:

```
$ docker pull atlassian/jira-software
```

Ak nie je vytvorené, tak je potrebné vytvoriť miesto, kde Jira bude ukladať dáta:

```
$ docker volume create --name jiraVolume
```

Nakoniec sa spustí docker container zo stiahnutého imagu **atlassian/jira-software** a nastaví sa, aby používal úložisko **jiraVolume**.

Je potrebné nastaviť premenné prostredia:

- **ALT_TOMCAT_CONTEXTPATH= /jira** - správne fungovanie reverse proxy
- **ATL_PROXY_NAME=team03-19.studenti.fiit.stuba.sk** - správne fungovanie reverse proxy
- **ATL_PROXY_PORT=443** - ak sa používa HTTPS
- **ATL_TOMCAT_SCHEME=https** - ak sa používa HTTPS
- **ATL_TOMCAT_SECURE=true** - ak sa používa HTTPS
- **JVM_MINIMUM_MEMORY=512m** - minimum pamäte RAM
- **JVM_MAXIMUM_MEMORY=1g** - maximum pamäte RAM

```
$ docker run -v jiraVolume:/var/atlassian/application-data/jira --name="jira" -d -p 8080:8080 -e
ATL_TOMCAT_CONTEXTPATH=/jira -e ATL_PROXY_NAME=team03-19.studenti.fiit.stuba.sk -e ATL_PROXY_PORT=443 -e
ATL_TOMCAT_SCHEME=https -e ATL_TOMCAT_SECURE=true -e JVM_MINIMUM_MEMORY=512m -e JVM_MAXIMUM_MEMORY=1g
atlassian/jira-software
```

Nasadzovanie projektu

Tím č. 03 FireAnt

Vedúci tímu: Branislav Pecher, Ivan Srba

Autor dokumentácie: Peter Mačinec

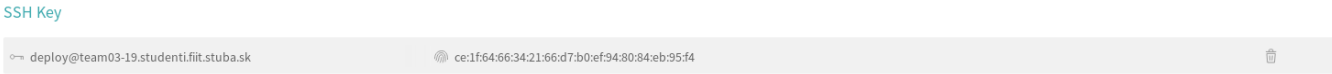
Dátum vytvorenia: 1.11.2019

Aplikácia FireAnt je tvorená dvoma oddelenými aplikáciami - **frontend** a **backend**. Obe z aplikácií sú nasadzované pomocou dockeru ako samostatný container. Obe z aplikácií sú nasadzované na dve prostredia - **production** (naviazané na vetvu *master*) a **staging** (naviazané na vetvu *develop*).

Nastavenie travisu a Githubu

Na nasadenie projektov používame **travis** a machine account na Githube. Repozitáre pre obe aplikácie majú rovnakú štruktúru, rovnaké vetvy a nasadenie je nastavené rovnako (master je produkcia a develop je staging). Pri zapracovaní zmien pomocou *pull requestu* sa spustí automatické nasadzovanie na príslušné prostredie.

Pre nastavenie automatického nasadzovania nie je potrebné nič spraviť z hľadiska kódu (repozitáre obsahujú súbory *Dockerfile*, *docker-compose.yml*, *.travis.yml a scripts/deploy.sh*, ktoré to vykonajú za nás). Je však potrebné nastaviť SSH Key v travise pre oba repozitáre:



Tento SSH kľúč musí byť priradený tiež používateľovi *deploy* na serveri, ktorý ho má vo svojom *home* adresári. Public časť tohto kľúča musí byť vložená v machine account na Githube (fireant-deploy, ktorý musí mať prístup k obom repozitárom ako *Collaborator*).

Nasadzovanie backend aplikácie

Aplikácia sa na serveri nasadzuje do dvoch adresárov, v závislosti od prostredia:

- */var/www/fireant-backend/staging*
- */var/www/fireant-backend/production*

Pre backend je potrebné pridať nastavenie `.env` súboru v nasledujúcom tvare (bez komentárov, uvedených za jednotlivými hodnotami):

.env súbor pre backend

```
FIREANT_ENV=staging # or production

FIREANT_FLASK_PORT_HOST=5001 # 5000 for production
FIREANT_FLASK_PORT_CON=5000

MONANT_CENTRAL_STORAGE_HOST=... # Monant API host (URL)
MONANT_CENTRAL_STORAGE_USERNAME=... # Monant API username
MONANT_CENTRAL_STORAGE_PASSWORD=... # Monant API password

SENTRY_DSN=... # SENTRY_DSN key from fireant-backend project
```

Taktiež je potrebné v travise nastaviť prístup k central storage API (Monant API) pre obe prostredia a nalinkovať ich na príslušnú vetvu. Tieto premenné sú potrebné z dôvodu spúšťania testov v travise (vytvorí sa docker container, ktorý potrebuje mať tieto hodnoty dostupné, pretože integračné testy využívajú aj Monant API). Príklad:

Nasadzovanie frontend aplikácie

Aplikácia sa na serveri nasadzuje do dvoch adresárov, v závislosti od prostredia:

- */var/www/fireant-frontend/staging*
- */var/www/fireant-frontend/production*

Pre frontend je potrebné pridať nastavenie `.env` súboru v nasledujúcom tvare (bez komentárov, uvedených za jednotlivými hodnotami):

.env súbor pre frontend

```
FIREANT_ENV=staging # or production
FIREANT_ENV_FOLDER=fireant-staging # fireant for production

FIREANT_ANGULAR_PORT_HOST=8001 # 8000 for production
FIREANT_ANGULAR_PORT_CON=80
```

Skript na vymazávanie nepoužívaných docker images

Tím č. 03 FireAnt

Vedúci tímu: Branislav Pecher, Ivan Srba

Autor dokumentácie: Miroslav Sumega

Dátum vytvorenia: 6.11.2019

Názov skriptu: manage_unused_docker_images

Lokácia skriptu: /etc/custom-crons

Nastavenie skriptu:

Tento skript sa spustí každý deň o 4:00 GTM 0 pod userom **root**. Čas spustenia skriptu je možné zmeniť v súbore **/etc/cron.d /manage_unused_docker_images**.

Funkcionalita skriptu:

Tento skript zistí, či sa v systéme nachádzajú nepoužívané docker images pomocou nasledovného príkazu:

```
docker images -f "dangling=true" -q
```

Ak také images v systéme sú, vráti pre každý image jeho ID.

Následne sa každý image vymaže pomocou nasledovného príkazu:

```
docker rmi $image
```

kde **$image** je ID aktuálne vymazávaného docker image. **V prípade, že je image používaný neaktívnym docker kontajnerom, nebude vymazaný**.

Zálohovanie dát z Jiry a Confluence

Tím č. 03 FireAnt

Vedúci tímu: Branislav Pecher, Ivan Srba

Autor dokumentácie: Miroslav Sumega

Dátum vytvorenia: 6.11.2019

Názov skriptu: backup_volumes

Lokácia skriptu: /etc/custom-crons

Nastavenie skriptu:

Tento skript sa spustí každý deň o 4:15 GMT 0 pod userom **root**. Čas spustenia skriptu je možné zmeniť v súbore **/etc/cron.d /backup_volumes**.

Funkcionalita skriptu:

Tento skript ukladá zálohy do archívu podľa nasldovného vzoru **volumes_backupX.tar.gz**, kde **X** je číslo zálohy, pričom čím väčšie číslo, tým staršia záloha. Počet záloh je **2**, takže najstaršia záloha bude **volumes_backup2.tar.gz**. Pôvodný počet záloh bol 3, ale z dôvodu nedostatku miesta sa počet záloh obmedzil na 2.

Najnovšia záloha je označená ako volumes_backup.tar.gz

Postup zálohovania:

1. Vykoná sa príkaz na presunutie do priečinka obsahujúceho zálohy: **cd /media/fireant/backups**
2. Vymaže sa najstaršia záloha.
3. Ostatným zálohám sa zvýši číslo o 1.
4. Vytvorí sa nová záloha.

Dokumentácia k API

V tejto časti prikladáme vygenerovanú dokumentáciu k API.

Flask API for Firebird platform [BETA]

Base URL: https://firebird.sh/api/v1/

Generate

POST /generate/ Generate a new Firebird platform

Request

Headers

Key	Value
Content-Type	application/x-www-form-urlencoded

Body (x-www-form-urlencoded)

Key	Value
prompt	Create a Flask API for Firebird platform

Response 200 OK — 1.09 s — 1.1 KB

Headers

Key	Value
Content-Type	application/x-www-form-urlencoded
Date	Sun, 22 Jun 2025 17:15:15 GMT
Server	Werkzeug/3.1.3 Python/3.12.3
Transfer-Encoding	chunked

Body (JSON)

```
{
  "Base URL": "https://firebird.sh/api/v1/",
  "Endpoints": {
    "/generate/": {
      "Description": "Generate a new Firebird platform",
      "Method": "POST",
      "Parameters": {
        "prompt": "Create a Flask API for Firebird platform"
      }
    }
    "/status/{id}": {
      "Description": "Check the status of a generation",
      "Method": "GET",
      "Parameters": {
        "id": "Generation ID"
      }
    }
    "/download/{id}": {
      "Description": "Download the generated platform",
      "Method": "GET",
      "Parameters": {
        "id": "Generation ID"
      }
    }
    "/cancel/{id}": {
      "Description": "Cancel a generation",
      "Method": "DELETE",
      "Parameters": {
        "id": "Generation ID"
      }
    }
    "/list": {
      "Description": "List all generations",
      "Method": "GET",
      "Parameters": {
        "limit": "Optional, default 10",
        "offset": "Optional, default 0"
      }
    }
    "/health": {
      "Description": "Health check",
      "Method": "GET",
      "Parameters": {}
    }
    "/stats": {
      "Description": "Statistics",
      "Method": "GET",
      "Parameters": {}
    }
    "/": {
      "Description": "Root",
      "Method": "GET",
      "Parameters": {}
    }
    "/docs": {
      "Description": "Documentation",
      "Method": "GET",
      "Parameters": {}
    }
    "/openapi.yaml": {
      "Description": "OpenAPI specification",
      "Method": "GET",
      "Parameters": {}
    }
    "/openapi.yaml/{id}": {
      "Description": "OpenAPI specification",
      "Method": "GET",
      "Parameters": {
        "id": "Generation ID"
      }
    }
    "/{id}": {
      "Description": "Get",
      "Method": "GET",
      "Parameters": {
        "id": "Generation ID"
      }
    }
    "/{id}/{path}": {
      "Description": "Get",
      "Method": "GET",
      "Parameters": {
        "id": "Generation ID",
        "path": "Path"
      }
    }
    "/{id}/{path}/{subpath}": {
      "Description": "Get",
      "Method": "GET",
      "Parameters": {
        "id": "Generation ID",
        "path": "Path",
        "subpath": "Subpath"
      }
    }
    "/{id}/{path}/{subpath}/{subsubpath}": {
      "Description": "Get",
      "Method": "GET",
      "Parameters": {
        "id": "Generation ID",
        "path": "Path",
        "subpath": "Subpath",
        "subsubpath": "Subsubpath"
      }
    }
    "/{id}/{path}/{subpath}/{subsubpath}/{subsubsubpath}": {
      "Description": "Get",
      "Method": "GET",
      "Parameters": {
        "id": "Generation ID",
        "path": "Path",
        "subpath": "Subpath",
        "subsubpath": "Subsubpath",
        "subsubsubpath": "Subsubsubpath"
      }
    }
    "/{id}/{path}/{subpath}/{subsubpath}/{subsubsubpath}/{subsubsubsubpath}": {
      "Description": "Get",
      "Method": "GET",
      "Parameters": {
        "id": "Generation ID",
        "path": "Path",
        "subpath": "Subpath",
        "subsubpath": "Subsubpath",
        "subsubsubpath": "Subsubsubpath",
        "subsubsubsubpath": "Subsubsubsubpath"
      }
    }
    "/{id}/{path}/{subpath}/{subsubpath}/{subsubsubpath}/{subsubsubsubpath}/{subsubsubsubsubpath}": {
      "Description": "Get",
      "Method": "GET",
      "Parameters": {
        "id": "Generation ID",
        "path": "Path",
        "subpath": "Subpath",
        "subsubpath": "Subsubpath",
        "subsubsubpath": "Subsubsubpath",
        "subsubsubsubpath": "Subsubsubsubpath",
        "subsubsubsubsubpath": "Subsubsubsubsubpath"
      }
    }
    "/{id}/{path}/{subpath}/{subsubpath}/{subsubsubpath}/{subsubsubsubpath}/{subsubsubsubsubpath}/{subsubsubsubsubsubpath}": {
      "Description": "Get",
      "Method": "GET",
      "Parameters": {
        "id": "Generation ID",
        "path": "Path",
        "subpath": "Subpath",
        "subsubpath": "Subsubpath",
        "subsubsubpath": "Subsubsubpath",
        "subsubsubsubpath": "Subsubsubsubpath",
        "subsubsubsubsubpath": "Subsubsubsubsubpath",
        "subsubsubsubsubsubpath": "Subsubsubsubsubsubpath"
      }
    }
    "/{id}/{path}/{subpath}/{subsubpath}/{subsubsubpath}/{subsubsubsubpath}/{subsubsubsubsubpath}/{subsubsubsubsubsubpath}/{subsubsubsubsubsubsubpath}": {
      "Description": "Get",
      "Method": "GET",
      "Parameters": {
        "id": "Generation ID",
        "path": "Path",
        "subpath": "Subpath",
        "subsubpath": "Subsubpath",
        "subsubsubpath": "Subsubsubpath",
        "subsubsubsubpath": "Subsubsubsubpath",
        "subsubsubsubsubpath": "Subsubsubsubsubpath",
        "subsubsubsubsubsubpath": "Subsubsubsubsubsubpath",
        "subsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubpath"
      }
    }
    "/{id}/{path}/{subpath}/{subsubpath}/{subsubsubpath}/{subsubsubsubpath}/{subsubsubsubsubpath}/{subsubsubsubsubsubpath}/{subsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubpath}": {
      "Description": "Get",
      "Method": "GET",
      "Parameters": {
        "id": "Generation ID",
        "path": "Path",
        "subpath": "Subpath",
        "subsubpath": "Subsubpath",
        "subsubsubpath": "Subsubsubpath",
        "subsubsubsubpath": "Subsubsubsubpath",
        "subsubsubsubsubpath": "Subsubsubsubsubpath",
        "subsubsubsubsubsubpath": "Subsubsubsubsubsubpath",
        "subsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubpath"
      }
    }
    "/{id}/{path}/{subpath}/{subsubpath}/{subsubsubpath}/{subsubsubsubpath}/{subsubsubsubsubpath}/{subsubsubsubsubsubpath}/{subsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubpath}": {
      "Description": "Get",
      "Method": "GET",
      "Parameters": {
        "id": "Generation ID",
        "path": "Path",
        "subpath": "Subpath",
        "subsubpath": "Subsubpath",
        "subsubsubpath": "Subsubsubpath",
        "subsubsubsubpath": "Subsubsubsubpath",
        "subsubsubsubsubpath": "Subsubsubsubsubpath",
        "subsubsubsubsubsubpath": "Subsubsubsubsubsubpath",
        "subsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubpath"
      }
    }
    "/{id}/{path}/{subpath}/{subsubpath}/{subsubsubpath}/{subsubsubsubpath}/{subsubsubsubsubpath}/{subsubsubsubsubsubpath}/{subsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubpath}": {
      "Description": "Get",
      "Method": "GET",
      "Parameters": {
        "id": "Generation ID",
        "path": "Path",
        "subpath": "Subpath",
        "subsubpath": "Subsubpath",
        "subsubsubpath": "Subsubsubpath",
        "subsubsubsubpath": "Subsubsubsubpath",
        "subsubsubsubsubpath": "Subsubsubsubsubpath",
        "subsubsubsubsubsubpath": "Subsubsubsubsubsubpath",
        "subsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubpath"
      }
    }
    "/{id}/{path}/{subpath}/{subsubpath}/{subsubsubpath}/{subsubsubsubpath}/{subsubsubsubsubpath}/{subsubsubsubsubsubpath}/{subsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubpath}": {
      "Description": "Get",
      "Method": "GET",
      "Parameters": {
        "id": "Generation ID",
        "path": "Path",
        "subpath": "Subpath",
        "subsubpath": "Subsubpath",
        "subsubsubpath": "Subsubsubpath",
        "subsubsubsubpath": "Subsubsubsubpath",
        "subsubsubsubsubpath": "Subsubsubsubsubpath",
        "subsubsubsubsubsubpath": "Subsubsubsubsubsubpath",
        "subsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubpath"
      }
    }
    "/{id}/{path}/{subpath}/{subsubpath}/{subsubsubpath}/{subsubsubsubpath}/{subsubsubsubsubpath}/{subsubsubsubsubsubpath}/{subsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubpath}": {
      "Description": "Get",
      "Method": "GET",
      "Parameters": {
        "id": "Generation ID",
        "path": "Path",
        "subpath": "Subpath",
        "subsubpath": "Subsubpath",
        "subsubsubpath": "Subsubsubpath",
        "subsubsubsubpath": "Subsubsubsubpath",
        "subsubsubsubsubpath": "Subsubsubsubsubpath",
        "subsubsubsubsubsubpath": "Subsubsubsubsubsubpath",
        "subsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubpath"
      }
    }
    "/{id}/{path}/{subpath}/{subsubpath}/{subsubsubpath}/{subsubsubsubpath}/{subsubsubsubsubpath}/{subsubsubsubsubsubpath}/{subsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubpath}": {
      "Description": "Get",
      "Method": "GET",
      "Parameters": {
        "id": "Generation ID",
        "path": "Path",
        "subpath": "Subpath",
        "subsubpath": "Subsubpath",
        "subsubsubpath": "Subsubsubpath",
        "subsubsubsubpath": "Subsubsubsubpath",
        "subsubsubsubsubpath": "Subsubsubsubsubpath",
        "subsubsubsubsubsubpath": "Subsubsubsubsubsubpath",
        "subsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubpath"
      }
    }
    "/{id}/{path}/{subpath}/{subsubpath}/{subsubsubpath}/{subsubsubsubpath}/{subsubsubsubsubpath}/{subsubsubsubsubsubpath}/{subsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubpath}": {
      "Description": "Get",
      "Method": "GET",
      "Parameters": {
        "id": "Generation ID",
        "path": "Path",
        "subpath": "Subpath",
        "subsubpath": "Subsubpath",
        "subsubsubpath": "Subsubsubpath",
        "subsubsubsubpath": "Subsubsubsubpath",
        "subsubsubsubsubpath": "Subsubsubsubsubpath",
        "subsubsubsubsubsubpath": "Subsubsubsubsubsubpath",
        "subsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubpath"
      }
    }
    "/{id}/{path}/{subpath}/{subsubpath}/{subsubsubpath}/{subsubsubsubpath}/{subsubsubsubsubpath}/{subsubsubsubsubsubpath}/{subsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath}": {
      "Description": "Get",
      "Method": "GET",
      "Parameters": {
        "id": "Generation ID",
        "path": "Path",
        "subpath": "Subpath",
        "subsubpath": "Subsubpath",
        "subsubsubpath": "Subsubsubpath",
        "subsubsubsubpath": "Subsubsubsubpath",
        "subsubsubsubsubpath": "Subsubsubsubsubpath",
        "subsubsubsubsubsubpath": "Subsubsubsubsubsubpath",
        "subsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath"
      }
    }
    "/{id}/{path}/{subpath}/{subsubpath}/{subsubsubpath}/{subsubsubsubpath}/{subsubsubsubsubpath}/{subsubsubsubsubsubpath}/{subsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath}": {
      "Description": "Get",
      "Method": "GET",
      "Parameters": {
        "id": "Generation ID",
        "path": "Path",
        "subpath": "Subpath",
        "subsubpath": "Subsubpath",
        "subsubsubpath": "Subsubsubpath",
        "subsubsubsubpath": "Subsubsubsubpath",
        "subsubsubsubsubpath": "Subsubsubsubsubpath",
        "subsubsubsubsubsubpath": "Subsubsubsubsubsubpath",
        "subsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath"
      }
    }
    "/{id}/{path}/{subpath}/{subsubpath}/{subsubsubpath}/{subsubsubsubpath}/{subsubsubsubsubpath}/{subsubsubsubsubsubpath}/{subsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath}": {
      "Description": "Get",
      "Method": "GET",
      "Parameters": {
        "id": "Generation ID",
        "path": "Path",
        "subpath": "Subpath",
        "subsubpath": "Subsubpath",
        "subsubsubpath": "Subsubsubpath",
        "subsubsubsubpath": "Subsubsubsubpath",
        "subsubsubsubsubpath": "Subsubsubsubsubpath",
        "subsubsubsubsubsubpath": "Subsubsubsubsubsubpath",
        "subsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath"
      }
    }
    "/{id}/{path}/{subpath}/{subsubpath}/{subsubsubpath}/{subsubsubsubpath}/{subsubsubsubsubpath}/{subsubsubsubsubsubpath}/{subsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath}": {
      "Description": "Get",
      "Method": "GET",
      "Parameters": {
        "id": "Generation ID",
        "path": "Path",
        "subpath": "Subpath",
        "subsubpath": "Subsubpath",
        "subsubsubpath": "Subsubsubpath",
        "subsubsubsubpath": "Subsubsubsubpath",
        "subsubsubsubsubpath": "Subsubsubsubsubpath",
        "subsubsubsubsubsubpath": "Subsubsubsubsubsubpath",
        "subsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath"
      }
    }
    "/{id}/{path}/{subpath}/{subsubpath}/{subsubsubpath}/{subsubsubsubpath}/{subsubsubsubsubpath}/{subsubsubsubsubsubpath}/{subsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath}": {
      "Description": "Get",
      "Method": "GET",
      "Parameters": {
        "id": "Generation ID",
        "path": "Path",
        "subpath": "Subpath",
        "subsubpath": "Subsubpath",
        "subsubsubpath": "Subsubsubpath",
        "subsubsubsubpath": "Subsubsubsubpath",
        "subsubsubsubsubpath": "Subsubsubsubsubpath",
        "subsubsubsubsubsubpath": "Subsubsubsubsubsubpath",
        "subsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath"
      }
    }
    "/{id}/{path}/{subpath}/{subsubpath}/{subsubsubpath}/{subsubsubsubpath}/{subsubsubsubsubpath}/{subsubsubsubsubsubpath}/{subsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath}": {
      "Description": "Get",
      "Method": "GET",
      "Parameters": {
        "id": "Generation ID",
        "path": "Path",
        "subpath": "Subpath",
        "subsubpath": "Subsubpath",
        "subsubsubpath": "Subsubsubpath",
        "subsubsubsubpath": "Subsubsubsubpath",
        "subsubsubsubsubpath": "Subsubsubsubsubpath",
        "subsubsubsubsubsubpath": "Subsubsubsubsubsubpath",
        "subsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath"
      }
    }
    "/{id}/{path}/{subpath}/{subsubpath}/{subsubsubpath}/{subsubsubsubpath}/{subsubsubsubsubpath}/{subsubsubsubsubsubpath}/{subsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath}": {
      "Description": "Get",
      "Method": "GET",
      "Parameters": {
        "id": "Generation ID",
        "path": "Path",
        "subpath": "Subpath",
        "subsubpath": "Subsubpath",
        "subsubsubpath": "Subsubsubpath",
        "subsubsubsubpath": "Subsubsubsubpath",
        "subsubsubsubsubpath": "Subsubsubsubsubpath",
        "subsubsubsubsubsubpath": "Subsubsubsubsubsubpath",
        "subsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath"
      }
    }
    "/{id}/{path}/{subpath}/{subsubpath}/{subsubsubpath}/{subsubsubsubpath}/{subsubsubsubsubpath}/{subsubsubsubsubsubpath}/{subsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath}/{subsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubsubpath}": {
      "Description": "Get",
      "Method": "GET",
      "Parameters": {
        "id": "Generation ID",
        "path": "Path",
        "subpath": "Subpath",
        "subsubpath": "Subsubpath",
        "subsubsubpath": "Subsubsubpath",
        "subsubsubsubpath": "Subsubsubsubpath",
        "subsubsubsubsubpath": "Subsubsubsubsubpath",
        "subsubsubsubsubsubpath": "Subsubsubsubsubsubpath",
        "subsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubpath": "Subsubsubsubsubsubsubsubsubpath",
        "subsubsubsubsubsubsubsubsubsubpath": "Subsubsubsubsub
```