Dokumentacja Wroflats

Wydanie pierwsze

Marek Kochanowski

29 cze 2018

Spis treści

ROZDZIAŁ 1

O projekcie

1.1 Wprowadzenie

Poszukiwanie mieszkania na wynajem jest obecnie źródłem wielu frustacji.
Serwisy ogłoszeniowe są nieintuicyjne i wypełnione powtarzającymi się, niskiej jakości ofertami.
Jak znaleźć i wybrać właściwą ofertę, która będzie odpowiadać wszystkim przyszłym lokatorom?

Wroflats jest systemem, który rozwiązuje wiele najpopularniejszych problemów dotykających każdą osobę aktywnie poszukującą lokum do wynajęcia. Pozwala on w miły i przyjemny sposób przeszukiwać oferowane mieszkania oraz pokoje biorąc pod uwagę lokalizację, budżet oraz czas dojazdu komunikacją miejską do najważniejszych części miasta.

1.2 Główne zalety Wroflats

1.2.1 Wykrywanie oraz usuwanie zduplikowanych ogłoszeń

Popularnym zachowaniem jest regularne umieszczanie identycznych ofert na portalach ogłoszeniowych – wynajmujący stosują taką taktykę w celu poprawienia pozycjonowania własnych nieruchomości oraz dla możliwości zaprezentowania ich w kategorii „ostatnio dodane ogłoszenia".

System automatycznie wykrywa oraz usuwa duplikaty, weryfikując poziom podobieństwa pomiędzy dostępnymi ofertami.

W praktyce oznacza to, że usunięcie ogłoszenia przez użytkownika ukrywa je na stałe – oferty ze wszystkich portali ogłoszeniowych są scalane do wspólnego formatu, uniemożliwiając powtórne wyświetlanie tych samych treści.

1.2.2 Wydajna praca w grupie

Wroflats wyposażony jest w system grup, które podnoszą komfort wspólnego poszukiwania idealnego lokum. Grupy przechowują wszystkie ustawienia wyszukiwania (maksymalny budżet, typ nieruchomości, lokalizacja), udostępniając współdzielony interfejs pomiędzy wszystkich członków grupy.

Decyzja podjęta przez jedną osobę jest natychmiastowo widoczna przez pozostałych członków, co znacznie usprawnia proces decyzyjny.

Każdy użytkownik może być członkiem wielu grup, nawet jeżeli są one wyłącznie jednoosobowe, dzięki temu może równolegle wyszukiwać mieszkania w różnych cenach, w różnych lokalizacjach oraz z różnymi parametrami.

1.2.3 Dynamiczne ocenianie

Ogłoszenia są sortowane według końcowego wskaźnika (oceny końcowej), wyliczanego na podstawie predefiniowanych parametrów wyszukiwania każdej grupy. Każdy parametr może mieć przydzieloną dowolną wagę.

Użytkownik własnoręcznie dobiera parametry wyszukiwania. Dzięki temu ma możliwość znalezienia oferty, która:
- posiada najlepszy współczynnik ceny za metr kwadratowy
- udostępnia najszybsze połączenie komunikacją miejską do wybranej uczelni
- mieści się w podanym budżecie
- jest w preferowanej części (dzielnicy) miasta
- prezentuje się najlepiej (jakość zdjęć załączonych do ogłoszenia)

1.2.4 Interaktywna mapa nieruchomości

Wszystkie oferty są automatycznie umieszczane na interaktywnej mapie, która zawiera najistotniejsze informacje.

Decyzje podejmowane w grupie są natychmiastowo widoczne – mieszkania dodane do ulubionych są odpowiednio wyróżnione żółtym kolorem, mieszkania usunięte nie wyświetlają się wcale.

Co najważniejsze – do każdej oferty przyporządkowana jest cena, widoczna bezpośrednio na mapie.
Aby dowiedzieć się więcej informacji o konkretnej nieruchomości, wystarczy jedno kliknięcie.

ROZDZIAŁ 2

Zaimplementowane klasy i modele

2.1 Użytkownicy

Użytkownik jest reprezentowany przez klasę `User` będącą modelem bazy danych.
Zawiera ona następujące atrybuty:

username - nazwa użytkownika
full_name - imię, nazwisko
password - hasło hashowane przy pomocy funkcji SHA256
avatar - odnośnik do zdjęcia użytkownika

```
class User(db.Model):
    __tablename__ = 'users'

    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(40), unique=True, nullable=False)
    full_name = db.Column(db.String(128), unique=False, nullable=True)
    password = db.Column(db.String(200), nullable=False)
    groups = db.relationship(
        'Group',
        secondary=users_groups_assoc,
        backref=db.backref('users', lazy=True))
    avatar = db.Column(db.String(512), nullable=True)
    sessions = db.relationship('Session', backref='user')
    created = db.Column(db.DateTime, nullable=False,
                        default=datetime.utcnow)
```

Użytkownik może być uwierzytelniony poprzez zapytanie typu `POST` pod adres `/auth/signin`, które jest obsługiwane przez klasę zasobów `AuthSignIn`.

Analogicznie, w celu utworzenia nowego użytkownika należy wysłać zapytanie typu `POST` pod adres `/auth/signup` z danymi w formacie `JSON`. Proces rejestracji jest obsługiwany przez klasę `AuthSignUp`.

2.2 Sesje

Podczas procesu uwierzytelniania, użytkownikowi przydzielany jest indywidualny identyfikator sesji.

Sesja jest reprezentowana przez klasę `Session` będącą modelem bazy danych.
Zawiera ona następujące atrybuty:

user_id - identyfikator użytkownika
created - data utworzenia sesji
expires - data wygaśnięcia sesji

```
class Session(db.Model):
    __tablename__ = 'sessions'

    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'))
    created = db.Column(db.DateTime, nullable=False,
                        default=datetime.utcnow)
    expires = db.Column(db.DateTime, nullable=True,
                        default=None)
```

2.3 Grupy

Grupy zawierają preferencje filtrowania mieszkań (ustawienia takie jak budżet, maksymalny czas dojazdu) oraz listę osób zapisanych do danej grupy.

Grupa jest reprezentowana przez klasę `Group` będącą modelem bazy danych.
Zawiera ona następujące atrybuty:

hash - unikalny, alfanumeryczny identyfikator grupy
title - tytuł grupy
city - miasto docelowe
parameters - obiekt przechowujący konfigurację parametrów
owner_id - identyfikator właściciela grupy
status - stan danej grupy, na przykład: *active* - grupa aktywna.

```
class Group(db.Model):
    __tablename__ = 'groups'

    id = db.Column(db.Integer, primary_key=True)
    hash = db.Column(db.String(10), unique=True)
    title = db.Column(db.String(120))
```

(ciąg dalszy na następnej stronie)

(kontynuacja poprzedniej strony)

```
    city = db.Column(db.String(255))
    parameters = db.Column(db.PickleType)
    owner_id = db.Column(db.Integer, db.ForeignKey('users.id'))
    status = db.Column(db.String(40), default='activated')
    created = db.Column(db.DateTime, nullable=False,
                        default=datetime.utcnow)
```

2.4 Obszary

Obszary służą do zaznaczania stref na interaktywnej mapie.
W zależności od trybu, ich funkcjonalność może być następująca:

- dla trybu *forbidden* - wszystkie ogłoszenia znajdujące się w zaznaczonym obszarze będą ignorowane
- dla trybu *preferred* - w przypadku pojawienia się nowego ogłoszenia w zaznaczonym obszarze nastąpi wysłanie powiadomienia do wszystkich członków grupy

Obszar jest reprezentowany przez klasę `Area` będącą modelem bazy danych.
Zawiera ona następujące atrybuty:

group_id - identyfikator grupy, do której należy obszar
type - typ obszaru, na przykład: *forbidden* - obszar na „czarnej liście”
center - punkt środkowy zaznaczanego obszaru
radius - promień zasięgu obszaru

```
class Area(db.Model):
    __tablename__ = 'areas'

    id = db.Column(db.Integer, primary_key=True)
    group_id = db.Column(db.Integer, db.ForeignKey('groups.id'))
    type = db.Column(db.String(40), default='forbidden')
    center = db.Column(db.Integer, db.ForeignKey('coordinates.id'))
    radius = db.Column(db.Integer)
    created = db.Column(db.DateTime, nullable=False,
                        default=datetime.utcnow)
```

2.5 Zgłoszenia / oferty

Oferty (zwane również zgłoszeniami) to elementy pobrane z portali ogłoszeniowych sprowadzone do wspólnego formatu.

Zgłoszenie jest reprezentowane przez klasę `Submission` będącą modelem bazy danych.
Zawiera ona następujące atrybuty:

submission_id - unikalny identyfikator ogłoszenia przypisany do platformy
category - kategoria, na przykład: *flat* - mieszkanie na wynajem, *room* - pokój na wynajem
origin - pochodzenie ogłoszenia, na przykład: *olx*

city - miasto, w którym znajduje się dana nieruchomość
title - tytuł ogłoszenia
url - link do pełnej wersji ogłoszenia
description - opis ogłoszenia
price - opublikowana cena za nieruchomość
source_latitude - pierwsza współrzędna bez normalizacji
source_longitude - druga współrzędna bez normalizacji
thumbnail - obrazek podglądowy
images - obiekt przechowujący listę wszystkich opublikowanych zdjęć
attributes - obiekt przechowujący parametry danej nieruchomości takie jak metraż, ilość pokoi
is_scraped - wartość boolowska mówiąca, czy wszystkie dane zostały pobrane z treści ogłoszenia

```
class Submission(db.Model):
    __tablename__ = 'submissions'
    # __bind_key__ = 'scraping'

    id = db.Column(db.Integer, primary_key=True)
    submission_id = db.Column(db.String(255))
    category = db.Column(db.String(40), nullable=True)
    origin = db.Column(db.String(40))
    city = db.Column(db.String(255))
    title = db.Column(db.String(255))
    url = db.Column(db.String(512))
    description = db.Column(db.Text)
    price = db.Column(db.Integer)
    source_latitude = db.Column(db.Float)
    source_longitude = db.Column(db.Float)
    thumbnail = db.Column(db.String(512))
    images = db.Column(db.PickleType)
    attributes = db.Column(db.PickleType)
    coordinates_id = db.Column(db.Integer, db.ForeignKey('coordinates.id'))
    is_scraped = db.Column(db.Boolean, default=False)
    submitted = db.Column(db.DateTime, nullable=True)
    updated = db.Column(db.DateTime, nullable=False,
                        default=datetime.utcnow)
    created = db.Column(db.DateTime, nullable=False,
                        default=datetime.utcnow)
```

2.6 Ocenione zgłoszenia

Każda grupa może dopasować parametry oceniania do własnych potrzeb, dlatego każde zgłoszenie może być ocenione na różne sposoby.
Z tego powodu powstała osobna struktura ocenionego zgłoszenia przypisanego do grupy - zawierająca wyliczone parametry oraz końcowy wynik.

Ocenione zgłoszenie jest reprezentowane przez klasę `CalculatedSubmission` będącą modelem bazy danych.
Zawiera ona następujące atrybuty:

hash - unikalny, alfanumeryczny identyfikator ocenionego zgłoszenia
submission_id - identyfikator oferty źródłowej

group_id - identyfikator grupy, na podstawie której wykonywane są obliczenia
rating - końcowy wynik wyliczony z zastosowaniem wszystkich parametrów
parameters - obiekt przechowujący wyliczone parametry oraz ich wartości
status - aktualny stan ogłoszenia, na przykład: *removed* - usunięte, *expired* - wygaszone

```
class CalculatedSubmission(db.Model):
    __tablename__ = 'submissions_calculated'

    id = db.Column(db.Integer, primary_key=True)
    hash = db.Column(db.String(20), unique=True)
    submission_id = db.Column(db.Integer, db.ForeignKey('submissions.id'))
    group_id = db.Column(db.Integer, db.ForeignKey('groups.id'))
    cords_pairs = db.relationship(
        'PairOfCoordinates',
        secondary=submissions_pairs_coordinates_assoc,
        backref=db.backref('submissions', lazy=True))
    rating = db.Column(db.Float)
    parameters = db.Column(db.PickleType)
    status = db.Column(db.String(40))
    created = db.Column(db.DateTime, nullable=False,
                        default=datetime.utcnow)
```

2.7 Współrzędne

W celu optymalizacji obliczeń dla nieruchomości leżących w nieznaczej odległości od siebie, zapisywane współrzędne są normalizowane.

Współrzędne są reprezentowane przez klasę `Coordinates` będącą modelem bazy danych.
Zawiera ona następujące atrybuty:

latitude - pierwsza współrzędna po normalizacji
longitude - druga współrzędna po normalizacji

```
class Coordinates(db.Model):
    __tablename__ = 'coordinates'

    id = db.Column(db.Integer, primary_key=True)
    latitude = db.Column(db.Float)
    longitude = db.Column(db.Float)
    submissions = db.relationship('Submission')
    created = db.Column(db.DateTime, nullable=False,
                        default=datetime.utcnow)
```

2.8 Pary współrzędnych

Aby uniknąć duplikowania obliczeń, połączenia między dwoma parami współrzędnych są zapisywane wraz z danymi takimi jak dystans oraz czasy dojazdu dla wybranych środków komunikacji.

Pary współrzędnych są reprezentowane przez klasę `PairOfCoordinates` będącą modelem bazy danych.
Zawiera ona następujące atrybuty:

origin_id - identyfikator współrzędnych źródła
target_id - identyfikator współrzędnych celu
distance - dystans między dwoma punktami podany w kilometrach
time - obiekt przechowujący czas dojazdu oraz środek komunikacji
calculated - data wyliczenia parametrów

```
class PairOfCoordinates(db.Model):
    __tablename__ = 'coordinates_pairs'

    id = db.Column(db.Integer, primary_key=True)
    origin_id = db.Column(db.Integer, db.ForeignKey('coordinates.id'))
    target_id = db.Column(db.Integer, db.ForeignKey('coordinates.id'))
    distance = db.Column(db.Float)
    time = db.Column(db.PickleType)
    created = db.Column(db.DateTime, nullable=False,
                        default=datetime.utcnow)
    calculated = db.Column(db.DateTime, nullable=True,
                        default=None)
```

2.9 Akcje

Akcje to pojedyncze zdarzenia wykonywane przez użytkownika. Przechowywane są aby móc wyświetlać historię wykonywanych operacji na zgłoszeniach przez członków grupy.

Pojedyncza operacja jest reprezentowane przez klasę `Action` będącą modelem bazy danych.
Zawiera ona następujące atrybuty:

target_id - identyfikator ocenionego zgłoszenia
action - wykonywana operacja
user_id - identyfikator użytkownika wykonującego działanie

```
class Action(db.Model):
    __tablename__ = 'actions'

    id = db.Column(db.Integer, primary_key=True)
    target_id = db.Column(db.Integer, db.ForeignKey(
        'submissions_calculated.id'))
    action = db.Column(db.String(40))
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'))
    created = db.Column(db.DateTime, nullable=False,
                        default=datetime.utcnow)
```

ROZDZIAŁ 3

Asynchroniczne zadania

Wroflats wykorzystuje strukturę *kolejki* w celu umieszczania oraz wykonywania zadań w sposób asynchroniczny. Część z dostępnych procesów jest wykonywana automatycznie, pozostałe tworzone są gdy zajdzie potrzeba uzyskania dostępu do dodatkowych obliczeń.

3.1 tasks.scrape_indices_init

Wywoływane automatycznie.

Proces pobiera listę skonfigurowanych portali ogłoszeniowych oraz wywołuje `tasks.scrape_indices` dla każdego elementu.

3.2 tasks.scrape_indices

Proces przyjmuje jako argument nazwę portalu ogłoszeniowego.
Wysyła żądanie typu `GET` do zdefiniowanego adresu, po czym parsuje stronę aby wygenerować listę nowych ogłoszeń.

3.3 tasks.scrape_single_init

Wywoływane automatycznie.

Proces pobiera ogłoszenia z bazy danych, które nie zostały jeszcze w całości opracowane.
Wywołuje `tasks.scrape_single` dla każdego takiego ogłoszenia.

3.4 tasks.scrape_single

Proces przyjmuje jako argument identyfikator zgłoszenia.
Wysyła żądanie typu `GET` do adresu przypisanego do oferty, po czym parsuje stronę aby zaktualizować dane ogłoszenia.

3.5 tasks.calculate_submissions

Proces, wykorzystując strukturę *Ocenionego zgłoszenia* tworzy nowe elementy lub odświeża istniejące wewnątrz wszystkich aktywnych grup. W praktyce polega to na wyliczeniu nowych ocen dla wszystkich ogłoszeń oznaczonych jako *active* – aktywne.

3.6 tasks.calculate_ratings

Proces wylicza końcowy wynik dla danych parametrów ogłoszenia wykorzystując ustawienia wyszukiwania grupy.

ROZDZIAŁ 4

Instalacja

System korzysta z kontenerów tworzonych z użyciem narzędzia `Docker`.
W ich skład wchodzą:

wroflats-api
wroflats-db
wroflats-celery
wroflats-celery-beat
wroflats-client
wroflats-rabbitmq

Uruchomienie całego systemu możliwe jest poprzez wywołanie następującej komendy w katalogu źródłowym projektu:

```
docker-compose up
```