Inżynieria internetu	
Data oddania: 20.01.2020 r.	
Imię i nazwisko: Krzysztof Jarek, Krzysztof Klimczyk	Kierunek, rok, grupa: IS 3, proj.1

Projekt został wykonany w frameworku Django 3.1.3 z wykorzystaniem SQLite3.

1. Routing i dostępne metody:

a) api/meals/
- get - pobranie wszystkich rekordów w tabeli Meals
- post - wstawienie nowego rekordu posiłku do tabeli Meals

b) api/meals/<int:id>
- get - pobranie jednego rekordu o podanym identyfikatorze z tabeli Meals
- put - edycja rekordu o podanym identyfikatorze z tabeli Meals
- delete - usunięcie rekordu o podanym identyfikatorze z tabeli Meals

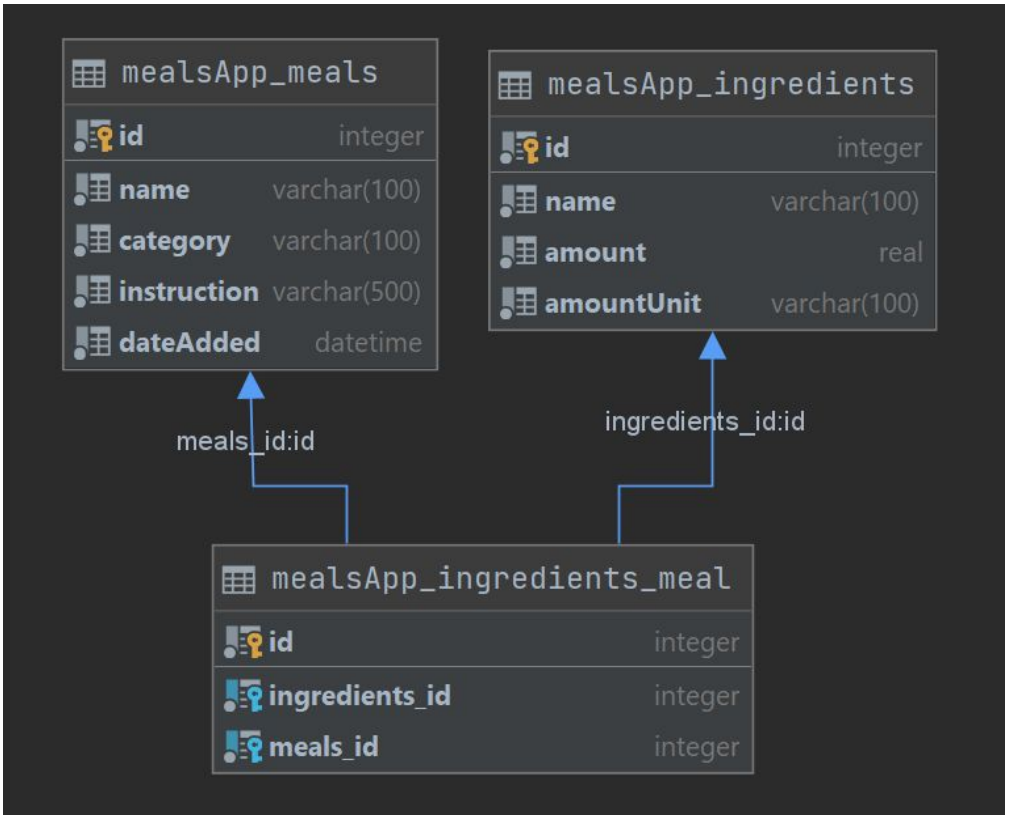
c) api/ingredients/
- get - pobranie wszystkich rekordów z tabeli Ingredients
- post - wstawienie nowego rekordu posiłku do tabeli Ingredients

d) api/ingredients/<int:id>
- get - pobranie jednego rekordu o podanym identyfikatorze z tabeli Ingredients
- put - edycja rekordu o podanym identyfikatorze z tabeli Ingredients
- delete - usunięcie rekordu o podanym identyfikatorze z tabeli Ingredients

Metody zwracają odpowiednie kody HTTP w przypadku powodzenia jak i podania niepoprawnych danych. Każda metoda get posiada indywidualny serializer, który w odpowiedni sposób przedstawi obiekt pobrany z bazy danych, przykładowo poza wyświetleniem wybrany kolumn wizualizuje tablicę z powiązanymi rekordami.

2. Model bazy danych:

Baza danych składa się z dwóch głównych tabel Meals, Ingredients i tabeli pomocniczej służącej do powiązania wcześniej wspomnianych tabel relacją many-to-many. Każda kolumna posiada wartość domyślną i walidator sprawdzający poprawność danych. Tabela Meals posiada pole dateAdded oznaczające datę dodania, która jest automatycznie uzupełniana na podstawie aktualnego czasu.



3. Walidacja danych, użytkownicy:

Walidacja danych jest realizowana z wykorzystaniem specjalnie przygotowanych walidatorów: długość danych przekazywanych do wszystkich pól jest sprawdzana przez *MinLengthValidator* i *MaxLengthValidator*; przy przekazywaniu danych użytkowników zastosowane zostały walidatory sprawdzające: minimalną długość wprowadzonego hasła i to czy nie pokrywa się z hasłami z bazy popularnych, poprawność zapisu adresu email, czy przesłane hasło pokrywa się z hasłem potwierdzającym.

Autoryzacja dokonuje się z wykorzystaniem tokenów JWT, które są generowane przez środowisko Django, a uwierzytelnianie komunikacji z ich pomocą jest kontrolowane za pomocą zainicjowanych obiektów *SessionAuthentication* i *IsAuthenticated* w klasach implementujących interfejs *APIView*.

4. Role użytkowników, routing dla określonych ról, logowanie zdarzeń:

a) api/accounts/register/
   - get - pobranie wszystkich rekordów z tabeli *auth_user*
   - post - wstawienie nowego rekordu użytkownika do tabeli *auth_user*

b) api/accounts/admin/
   - get - pobranie wszystkich rekordów z tabeli *auth_user* ze statusem superuser'a
   - post - wstawienie nowego rekordu użytkownika ze statusem superuser'a do tabeli *auth_user*

Metody zwracają odpowiednie kody HTTP w przypadku powodzenia jak i podania niepoprawnych danych. Każda klasa ma indywidualną klasę serializerz wzbogaconą o liczne, wspomniane walidatory. Wdrożenie logowania zdarzeń zostało zrealizowane poprzez dodanie odpowiedniej konfiguracji dla logowania. Logi są rejestrowane w dedykowanym pliku w katalogu */logs*.

5. Testy integracyjne, dokumentacja:

Testy umieściliśmy w osobnym folderze i zrealizowaliśmy tworząc dedykowane klasy i umieszczając tam 5 przypadków testowych.
Dokumentacja została przygotowana z pomocą oprzyrządowania portalu Swagger.