SPDB

Dokumentacja końcowa

Skład zespołu:

- Antoni Charchuła 283713
- Jakub Tokarzewski 283778

Spis treści

1. Treść zadania **2**

2. Źródła danych **2**

3. Opis metody wyszukiwania **3**

3.1. Sprecyzowanie ograniczeń dot. wyszukiwania miejsc 3

3.2. Tworzenie sieci przestrzennej 3

3.3. Algorytm rekomendujący 4

3.4. Wyświetlanie rezultatów użytkownikowi 4

4. Informacje o implementacji **5**

4.1. Schemat architektury aplikacji 5

4.2. Zastosowany model danych 5

4.3. Znajdowanie punktów do wyszukiwania POI 5

4.4. Algorytm rekomendujący 6

4.5. Stos technologiczny 7

4.5.1. Backend 7

4.5.2. Frontend 7

5. Testowanie skuteczności zaimplementowanego algorytmu **7**

5.1. Przykład działania aplikacji 7

5.2. Testowanie parametru opóźnienia wyszukiwania 9

6. Podsumowanie **9**

1. Treść zadania

Zadanie nr. 16
Zadane jest miejsce początkowe i miejsce docelowe podróży. Napisać aplikację, która wskaże miejsca do odwiedzenie (dowolnie wybrane tzw. punkty zainteresowanie -POI) oraz wyznaczy trasę przejazdu przy spełnieniu ograniczeń dotyczącej trasy: wydłużenia czasu przejazdu, długości trasy oraz okresu w czasie podróży, w którym dany punkt ma być odwiedzony (np. po godzinie od rozpoczęcia podróży).

Aplikacja powinna zawierać autonomiczny moduł do pokazania wyznaczonej trasy na mapie. Do napisania aplikacji należy wykorzystać dostępne serwisy oferujące dane przestrzenne (np. serwis OpenStreetMap http://www.openstreetmap.org/).

2. Źródła danych

Google Directions API - serwis pozwalający na wyznaczenie trasy pomiędzy dwoma punktami, z możliwością ustawienia punktów pośrednich. Podczas wyznaczania tras wykorzystujemy parametr “avoidHighways”, który powoduje, że otrzymywane trasy nie prowadzą przez autostrady. Spowodowane jest to tym, że czas przejazdu autostradą byłby znacznie szybszy niż zwykłymi drogami prowadzącymi do ciekawych miejsc oraz autostrady często posiadają niewiele zjazdów. Wyznaczenie jednej trasy kosztuje 0,005 USD.

Foursquare Places API - serwis zapewniający dane na temat punktów zainteresowania znajdujących się w określonej odległości od danego miejsca. Można w nim zastosować konkretne kategorie miejsc. Wykorzystana została wersja konta “Personal Tier”, która pozwala dziennie na 99,500 darmowych zapytań.

Google Distance Matrix API - serwis pozwalający na otrzymanie informacji na temat długości trasy pomiędzy dwoma punktami oraz czasu podróży. Każda informacja o odległość od jednego do drugiego punktu kosztuje 0,005 USD. Jako, że zapytań o te dane jest dużo, jest to najbardziej kosztowny element systemu.

Baza danych MongoDB - baza danych, w której zapisywane są generowane przez aplikacje oceny miejsc. Pozyskiwanie ocen wszystkich rozważanych miejsc byłaby bardzo kosztowna, ponieważ wymagałoby to wykonywania oddzielnego zapytania dla każdego miejsca. Z tego powodu zdecydowaliśmy się na losowanie ocen dla danych miejsc i zapisywanie ich we własnej bazie danych. Dzięki temu, podczas sprawdzania oceny miejsca szukamy czy jest już w bazie danych, jeżeli tak to ocena pobierana jest z bazy danych, w przeciwnym wypadku losujemy ocenę i zapisujemy miejsce wraz z oceną.

3. Opis metody wyszukiwania

3.1. Sprecyzowanie ograniczeń dot. wyszukiwania miejsc

W celu stworzenia trasy użytkownik podaje ograniczenia dotyczące wyszukiwanych miejsc. Poniżej przedstawione są wszystkie parametry, które użytkownik może ustawić:

- miejsce początkowe podróży (obowiązkowe)
- miejsce docelowe podróży (obowiązkowe)
- średni czas spędzony w jednym miejscu (wartość domyślna 30 min)
- minimalna ocena miejsc (wartość domyślna 0, aby brać pod uwagę wszystkie POI)
- kategorie miejsc branych pod uwagę (domyślnie brane pod uwagę wszystkie kategorie)
- maksymalny dodatkowy czas o jaki zostanie wydłużona podróż (domyślnie 2.5 godz.)
- maksymalny dodatkowy dystans przebyty podczas podróży (domyślnie 50 km)
- czas od startu podróży, po jakim może być proponowane miejsce do odwiedzenia (domyślnie 0 minut)

3.2. Tworzenie sieci przestrzennej

Po otrzymaniu argumentów użytkownika tworzenie sieci przebiega w następujących etapach:

- Uzyskanie pierwotnej trasy od punktu startowego do punktu docelowego z Google Directions API
- Znajdowanie punktów na trasie, w których należy przeszukać obszar o określonym promieniu (w aplikacji ustawiony na 50 km) w celu znalezienia POI.
- W każdym z punktów, wysłanie zapytania do Foursquare Places API, które zwraca interesujące miejsca
- Sprawdzenie w bazie danych czy znalezione POI mają wygenerowaną ocenę i jej pobranie. Jeśli, któreś z nich jej nie posiada, ocena jest generowana i zapisywana do bazy.
- Filtrowanie znalezionych POI względem, narzuconej przez użytkownika, minimalnej oceny
- Dodanie do znalezionych miejsc, punktu startowego i końcowego
- Pobranie długości tras (czasu i odległości) pomiędzy wszystkimi punktami, używając do tego Google Distance Matrix API
- Zapisanie uzyskanych miejsc i ich połączeń w formie grafu

3.3. Algorytm rekomendujący

Na stworzonym grafie uruchamiany jest algorytm znajdujący trasę zawierającą interesujące miejsca pomiędzy punktem startowym i końcowym. W trakcie jego działania brana jest pod uwagę:

- maksymalna długość trasy
- maksymalny czas w podróży
- maksymalna ilość POI - wynosi ona 10 i jest narzucona przez Google Directions API
- brak możliwości cofania się do POI, które znajdują się na wcześniejszym odcinku trasy
- czas, od którego należy szukać POI

Ocena trasy polega na zsumowaniu ocen wszystkich POI oraz podzieleniu przez ich ilość. W ten sposób mamy pewność, że priorytetyzowane są trasy z mniejszą ilością POI, ale lepszą oceną.

W przypadku, gdy argumenty podane przez użytkownika nie pozwalają na znalezienie takiej trasy, zwracany jest komunikat z błędem.

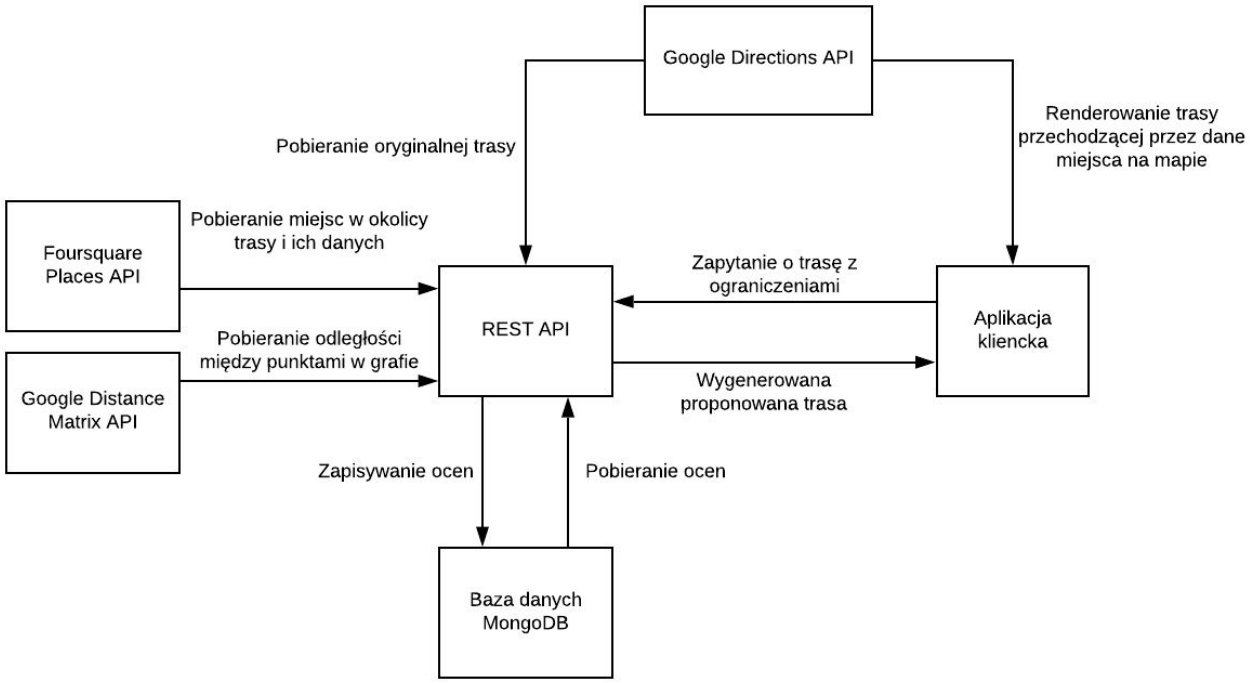
3.4. Wyświetlanie rezultatów użytkownikowi

Użytkownikowi wyświetlana jest trasa na mapie z zaznaczonymi wszystkimi punktami do odwiedzenia. Alfabetyczne oznaczenie punktów definiuje kolejność ich odwiedzenia. Poza wygenerowaną trasą użytkownik widzi również:

- informacje dot. czasu oraz dystansu oryginalnej trasy z punktu początkowego do punktu końcowego
- informacje dot. czasu oraz dystansu rekomendowanej trasy z punktu początkowego do punktu końcowego
- Ograniczenia użyte podczas procesu generowania rekomendowanej trasy

4. Informacje o implementacji

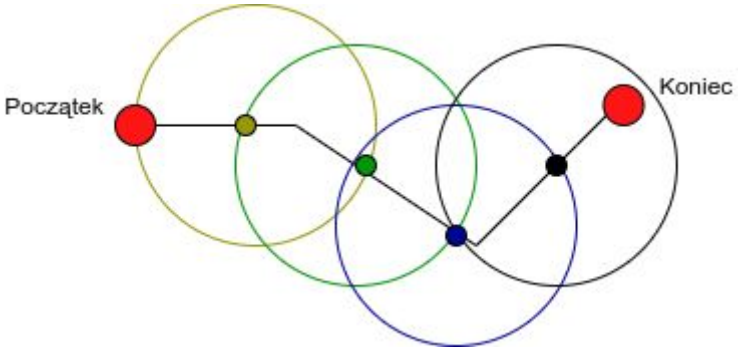
4.1. Schemat architektury aplikacji



4.2. Zastosowany model danych

Zapisywane dane dotyczą tylko ocen POI. Pozostałe dane uzyskiwane są z zewnętrznych API. Używana baza danych to MongoDB, która jest bazą typu NoSQL, gdzie dane przechowywane są w postaci JSON i identyfikuje się je za pomocą klucza. W naszym przypadku kluczem jest identyfikator POI uzyskany z Foursquare Places API. W bazie oprócz identyfikatora miejsca zapisywana jest wygenerowana przez aplikację średnia ocena.

4.3. Znajdowanie punktów do wyszukiwania POI



Do znajdowania POI wykorzystany został endpoint z Foursquare Places API, który znajduje POI z obszaru o określonym promieniu dla danego punktu. Stworzony algorytm wyszukiwania tych punktów znajduje je odliczając długość promienia od punktu startowego, a następnie od poprzedniego punktu przeszukiwania, do momentu aż cała trasa zostanie pokryta. Niestety, trasa otrzymywana z Google Directions API zawiera informacje tylko o krokach w momentach napotkania rozwidlenia dróg (wskazywane są także koordynaty tego miejsca), wobec tego należało obliczać koordynaty wymaganych punktów samodzielnie. Koordynaty te są obliczane na podstawie wzorów znajdujących się na stronie https://www.movable-type.co.uk/scripts/latlong.html. Po znalezieniu wszystkich punktów, wysyłane są zapytania do Foursquare Places API, który zwraca 10 POI dla jednego obszaru. Na samym końcu filtrowane są miejsca zduplikowane. Każde zapytanie oznaczone jest swoim identyfikatorem, który potem wpisujemy do parametrów znalezionego POI, aby później móc łatwo zidentyfikować, które POI są dalej od punktu startowego od innych.

4.4. Algorytm rekomendujący

Pseudokod:

```
funkcja znajdźNajlepsząŚcieżkę(pozostałyCzas, pozostałyDystans, aktualnaTrasa,
        aktualnyPunkt, sumaOcen, czasRozpoczęciaPoszukiwania) {

        Dla każdego punktu, do którego prowadzą drogi z aktualnegoPunktu {
                if (aktualnaŚcieżka.zawiera(nowyPunkt))
                        pomiń drogę
                if (nowyPunkt znajduje się bliżej punktu startowego)
                        pomiń drogę
                if (droga jest krótsza niż czasRozpoczęciaPoszukiwania)
                        pomiń drogę
                if (po przebyciu drogi pozostały czas i pozostały dystans do pokonania jest
                mniejszy od 0)
                        pomiń drogę
                if (nowyPunkt jest punktem końcowym) {
                        Dodaj punkt do trasy;
                        if (trasa jest lepsza niż poprzednia najlepsza trasa)
                                zapisz jako najlepsza trasa
                } else {
                        if (pozostały czas i pozostały dystans jest równy zero)
                                pomiń drogę
                        if (ilość 10 POI została przekroczona)
                                pomiń drogę
                        Dodaj punkt do trasy;
                        znajdźNajlepsząŚcieżkę(nowyPozostałyCzas, nowyPozostałyDystans,
                                trasa,  nowyPunkt, nowaSumaOcen, 0);
                }
        }
}
```

4.5. Stos technologiczny

4.5.1. Backend

- Java 11
- Spring
- MongoDB
- Maven

4.5.2. Frontend

- React
- React-google-maps - biblioteka opakowująca użycie Google Maps JavaScript API v3 w reużywalne komponenty React'owe.
- Reactstrap - biblioteka z komponentami do stylowania aplikacji z wykorzystaniem biblioteki Bootstrap

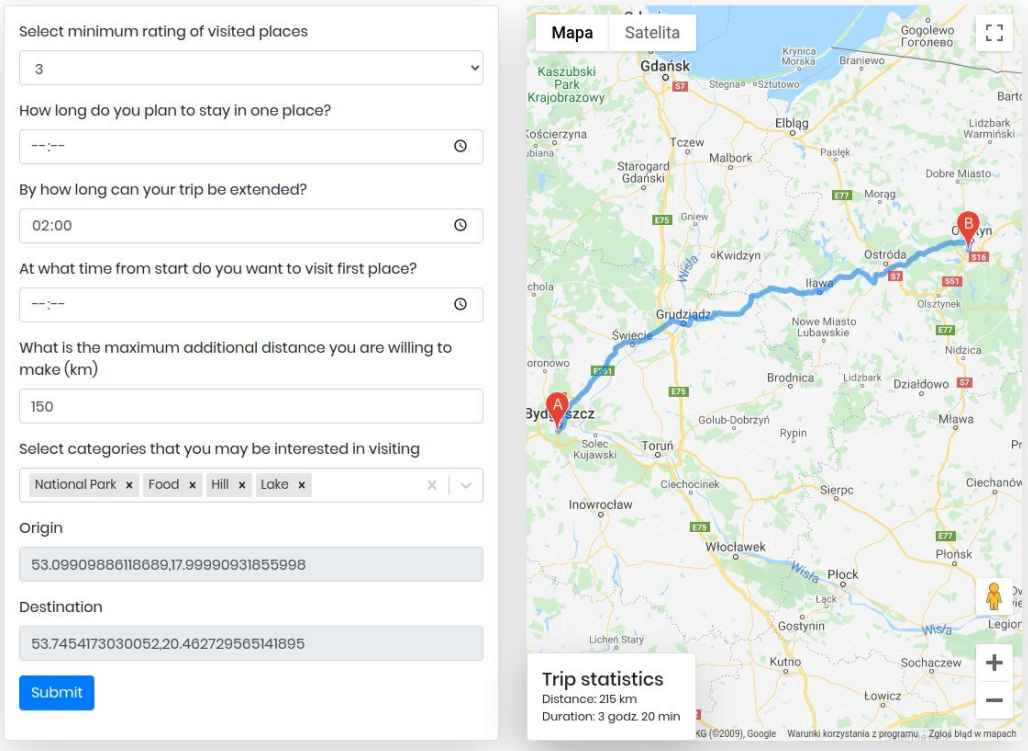
5. Testowanie skuteczności zaimplementowanego algorytmu

Kryterium oceny wygenerowanej trasy jest jej średnia ocena zaproponowanych miejsc do odwiedzenia. Dzięki takiemu podejściu najlepszymi trasami będą takie, które będą proponowały najlepiej oceniane miejsca, a nie np. najwięcej miejsc. Nie wprowadziliśmy kryterium dotyczącego wpływu długości trasy (zarówno dystans jak i czas) na jej ocenę. Według nas te współczynniki nie mają znaczenia, jeżeli mieszczą się w zadanych przez użytkownika wcześniej ograniczeniach.

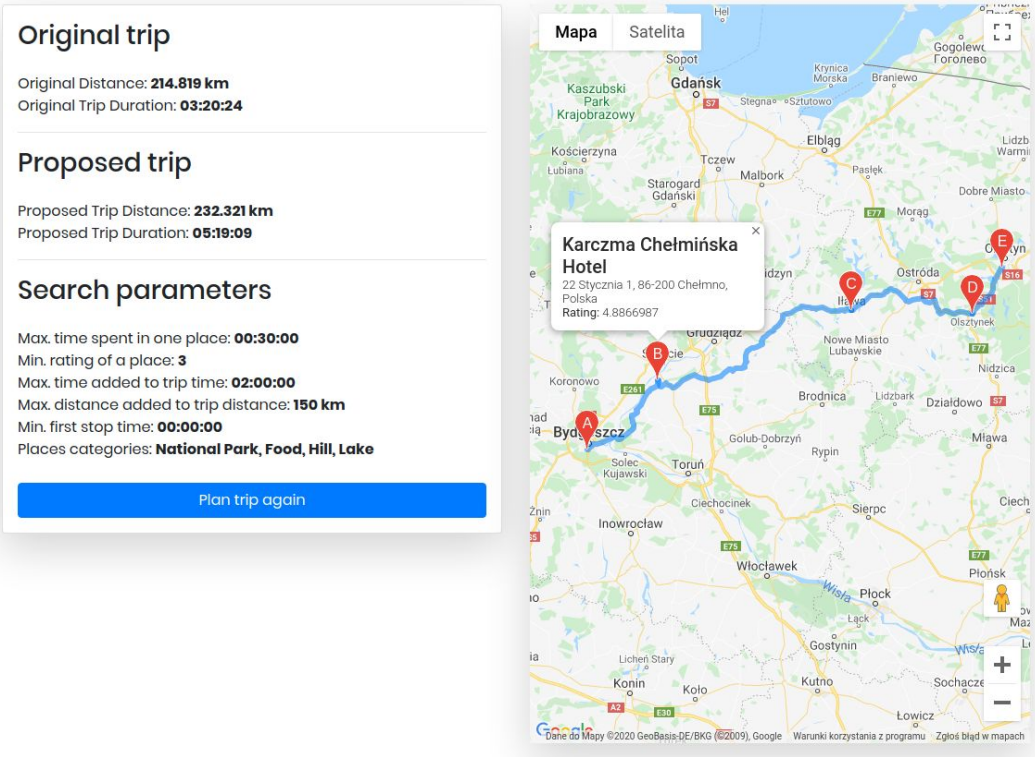
5.1. Przykład działania aplikacji

Poniżej zamieszczone są zdjęcia przedstawiające przykładowe działanie naszej aplikacji.

Z lewej strony widoczny jest formularz do wprowadzania ograniczeń, natomiast z prawej widoczna jest mapa, która dynamicznie pokazuje trasę między wybranym punktem początkowym i końcowym. Aby zmienić punkty początkowy należy kliknąć lewym przyciskiem myszy na mapie, natomiast aby zmienić punkt końcowy należy kliknąć prawym przyciskiem myszy na mapie. W dolnej części mapy wyświetlone są dane dotyczące trasy prowadzącej bezpośrednio od punktu startowego do docelowego.

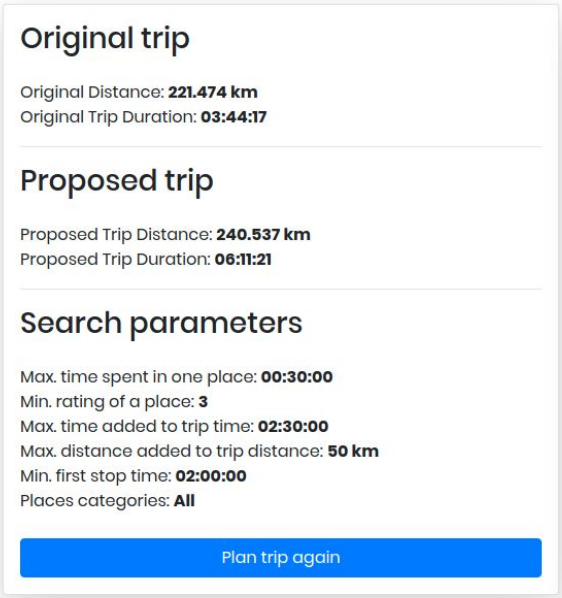


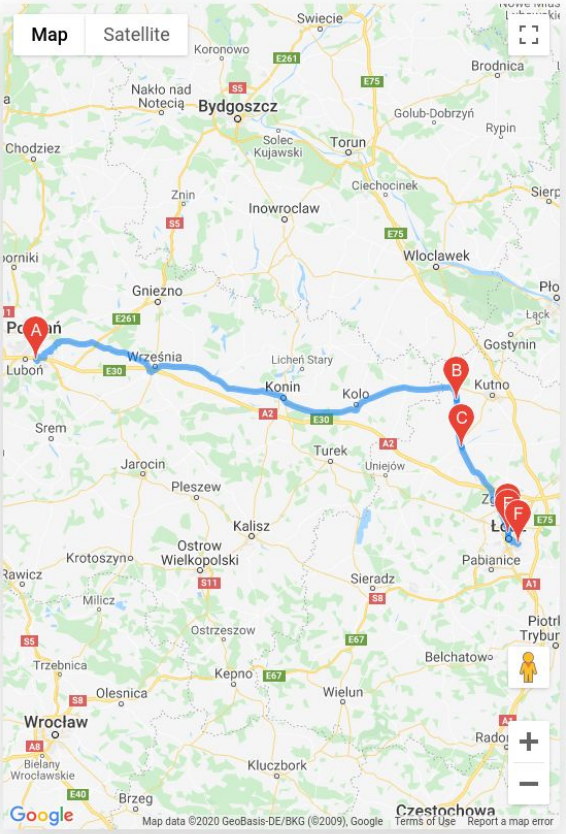
Po zatwierdzeniu formularza aktualizowana jest trasa wyświetlana na mapie, w taki sposób aby zawierała potencjalnie ciekawe miejsca do odwiedzenia proponowane przez nasz system. W miejscu formularza wyświetlany jest panel ze statystykami podróży bezpośredniej oraz tej wygenerowanej przez system. Widoczne są również zadane ograniczenia, dzięki czemu możliwa jest weryfikacja poprawności wygenerowanej trasy.



5.2. Testowanie parametru opóźnienia wyszukiwania

Sposobem na przetestowanie ograniczenia mówiącego o tym od jakiego czasu od rozpoczęcia podróży system powinien proponować miejsca do odwiedzenia jest wyświetlanie wyników zwracanych przez algorytm. Na zdjęciu przedstawionym poniżej ten parametr został ustawiony na 2 godziny i widać, że pierwszy proponowany przystanek (w punkcie B) znajduje się dopiero w okolicach ⅔ całej trasy, co pozwala wnioskować, że system działa poprawnie.





6. Podsumowanie

Wynikiem naszej pracy nad projektem jest działająca aplikacja internetowa, umożliwiająca generowanie tras przechodzących przez potencjalnie ciekawe miejsca między dwoma zadanymi punktami. Generowanie tras jest niestety kosztowne finansowo ze względu na korzystanie z płatnych zewnętrznych API, przez co aplikacja miałaby problem z funkcjonowaniem bez dobrego modelu biznesowego i planu finansowego.

Kody źródłowe aplikacji będących tematem tego projektu znajdują się w repozytoriach github:

- backend: https://github.com/ACharchula/SPDB-Backend
- frontend: https://github.com/tokerson/SPDB-Frontend