

Dokumentacja projektu

Projekt TAB

Aplikacja do przeglądania i zapisywania ciekawych
miejsc w architekturze mikroservisów

Autorzy

Tomasz Nowok

Filip Pyziak

Karol Kadłubowski

Mateusz Myga

Michał Gorawski

Jakub Tomczak

Spis treści

1. Opis projektu
 - a. Algorytm rekomendacyjny
2. Stack technologiczny
3. Architektura projektu
4. Opis poszczególnych serwisów
 - a. Venue.API
 - b. Identity.API
 - c. UserProfile.API
 - d. VenueList.API
 - e. VenueReview.API
 - f. Category.API
 - g. VenueAlgorithm.API
 - h. FileStorage.API
5. Opis aplikacji klienckiej
6. Biblioteki współdzielone
7. Instrukcja obsługi

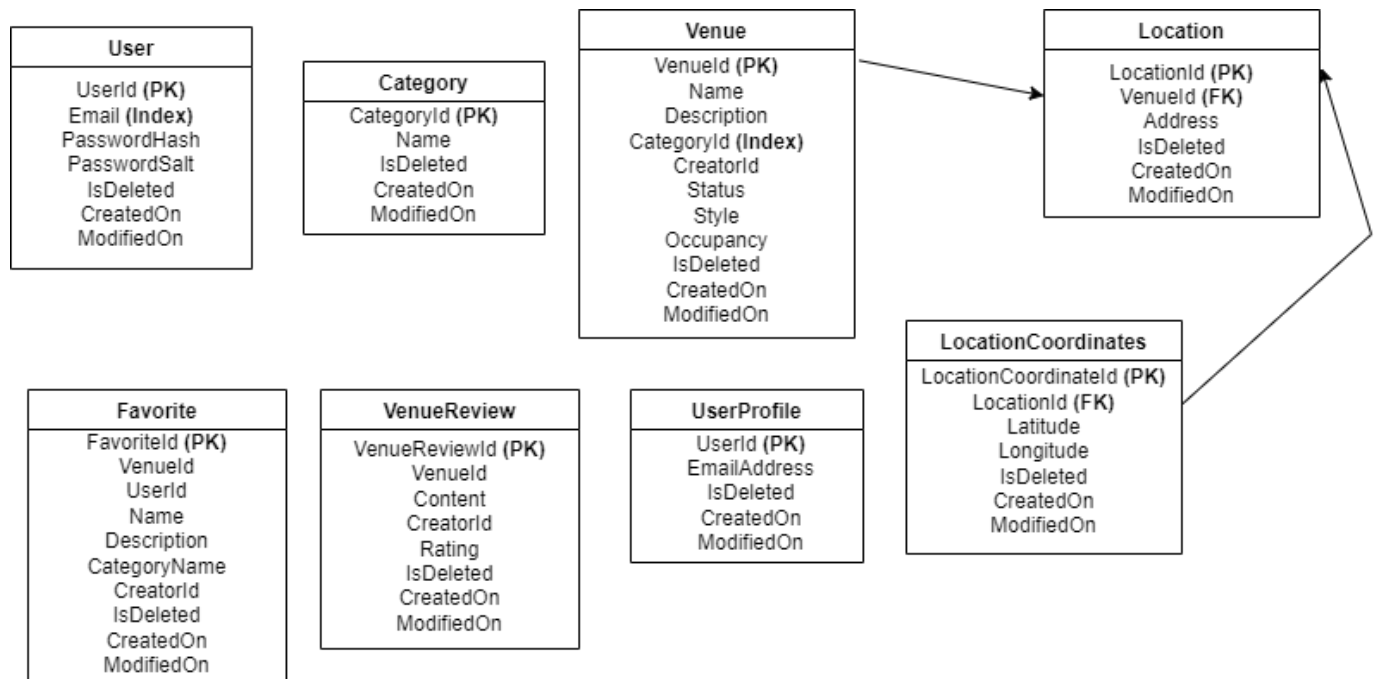
1. Opis projektu

Celem aplikacji jest możliwość przeglądania ciekawych dla użytkownika miejsc w podobny sposób jak umożliwia to aplikacja Tinder. Dodatkowo uwierzytelniony użytkownik może dodawać te miejsca do swojej listy ulubionych, ale także tworzyć nowe. Aplikacja wspiera autentykację użytkownika z prostą możliwością zmiany adresu email i hasła (bez dodatkowego zabezpieczenia w formie wiadomości email).

Diagram use-case:



Diagram encji:



2. Stack technologiczny

Technologie wykorzystane w projekcie:

- .NET
- React
- Flask
- Kafka
- PostgreSQL
- MongoDB
- Redis Cache
- Docker
- Neo4J
- Tailwind

4. Opis poszczególnych serwisów

Venue.API

localhost:8000/swagger

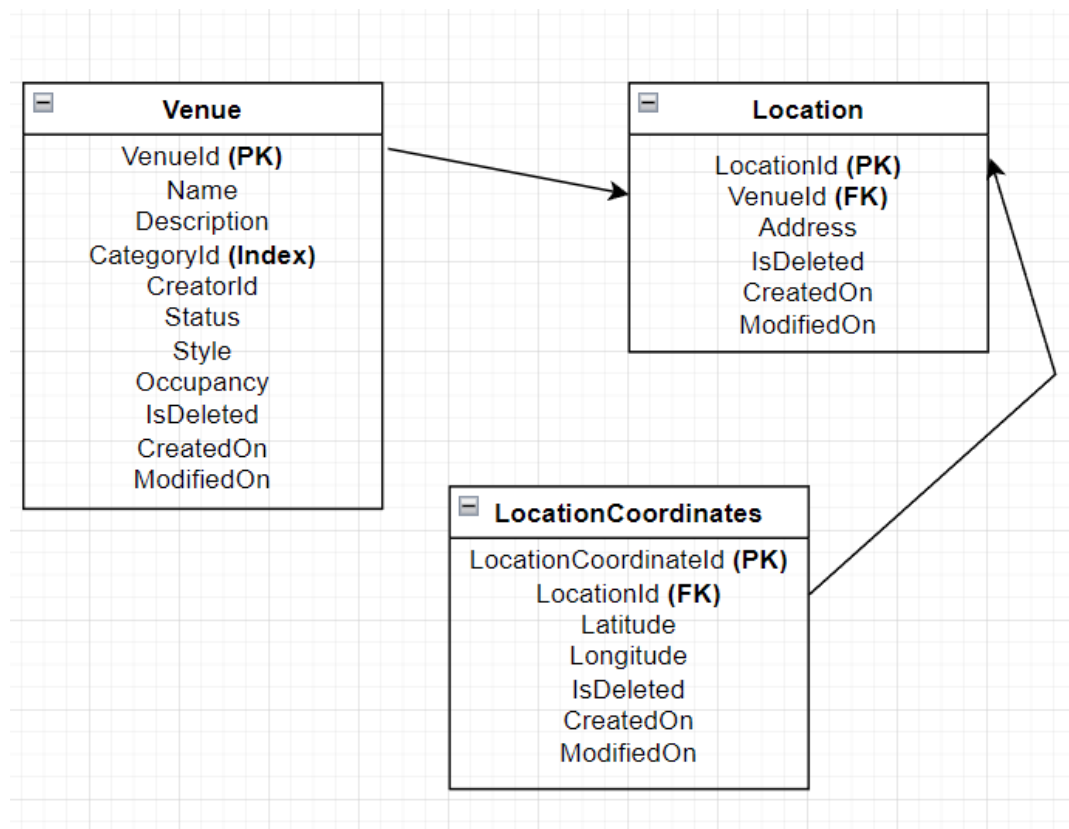
Venue.API ^{v1} ^{OAuth3}
/swagger/v1/swagger.json

Authorize

Venue Controller which provides Venue CRUD functionality

GET	/api/v1/Venue	Return Venue entity from the database. Append photos from the FileStorage API	
POST	/api/v1/Venue	Create new Venue entity in the database	
PUT	/api/v1/Venue	Update existing Venue entity found in the database by the VenueId	
DELETE	/api/v1/Venue	Delete existing Venue entity from the database	
GET	/api/v1/Venue/List	Return Venue entities from the database using pagination	

Baza danych: PostgreSQL - schema 'Venue'



Jedna z ważniejszych mikrouslug w systemie. Odpowiada za tworzenie, edycję i usuwanie własnych miejscówek, a także wystawia punkt końcowy do pobierania miejscówek z wykorzystaniem paginacji oraz punkt końcowy, który zwraca konkretne miejsce razem z jego zdjęciami znajdującymi się w FileStorage.API - jeśli serwis nie będzie dostępny otrzymamy miejsce z pustą kolekcją zdjęć.

Serwis wymaga bycia zalogowanym.

Serwis produkuje następujące zdarzenia:

- VENUE_CREATED - w momencie gdy nowe miejsce zostanie utworzone
- VENUE_UPDATED - w momencie gdy istniejące miejsce zostanie zaktualizowane
- VENUE_DELETED - w momencie gdy istniejące miejsce zostanie usunięte

Serwis konsumuje następujące zdarzenia:

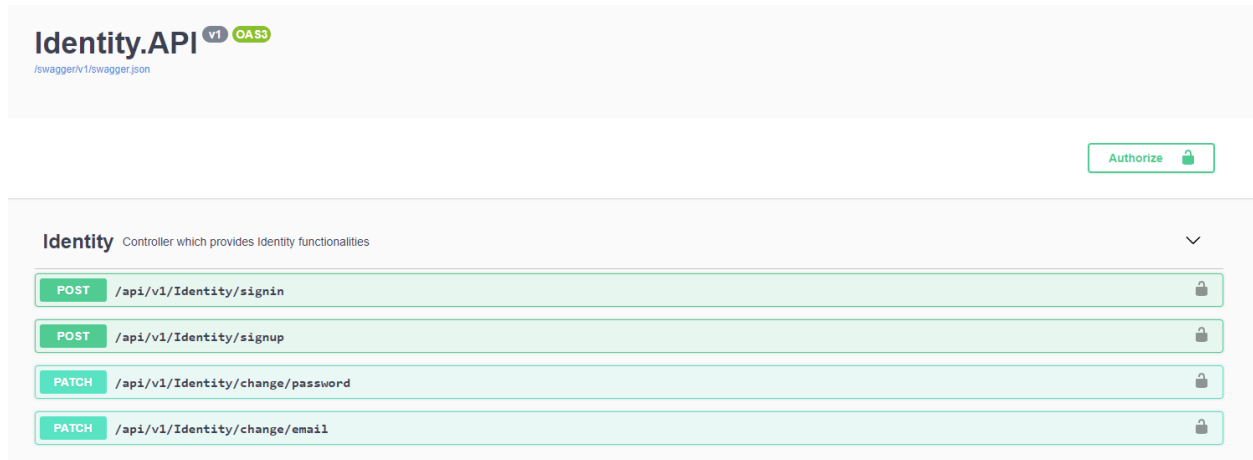
- CATEGORY_ADDED - dodaje kategorię do pamięci podręcznej
- CATEGORY_DELETED - usuwa kategorię z pamięci podręcznej

Na starcie serwisu oraz co kolejne 24 godziny, serwis odpytuje Category.API o wszystkie kategorie, żeby przechować je w pamięci podręcznej (kategorie są wymagane podczas tworzenia i aktualizacji miejsc).

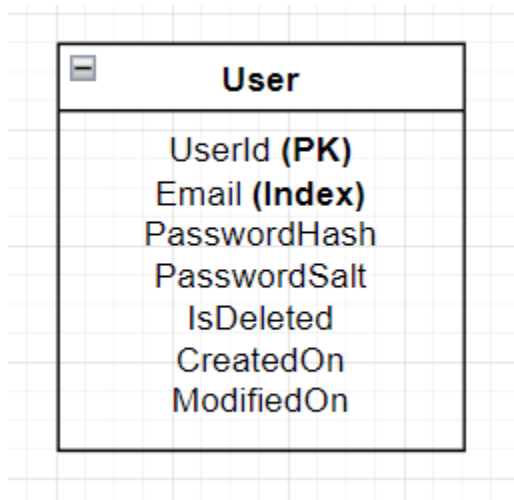
Serwis posiada testy jednostkowe.

Identity.API

localhost:8010/swagger



Baza danych: PostgreSQL - schema 'Identity'



Usługa odpowiada za rejestrację, logowanie użytkowników, a także daje możliwość zmiany hasła oraz adresu email zalogowanemu użytkownikowi.

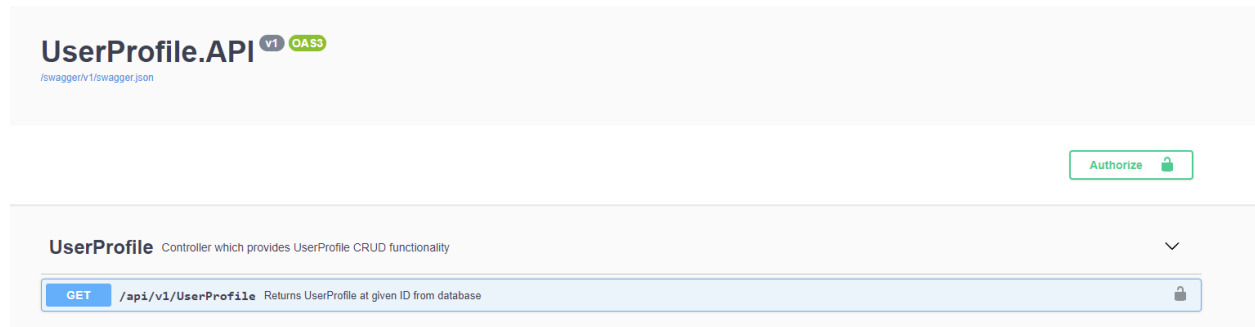
Serwis produkuje następujące zdarzenia:

- USER_CREATED - w momencie gdy użytkownik utworzy nowe konto
- USER_EMAIL_CHANGED - w momencie gdy użytkownik zmieni adres email

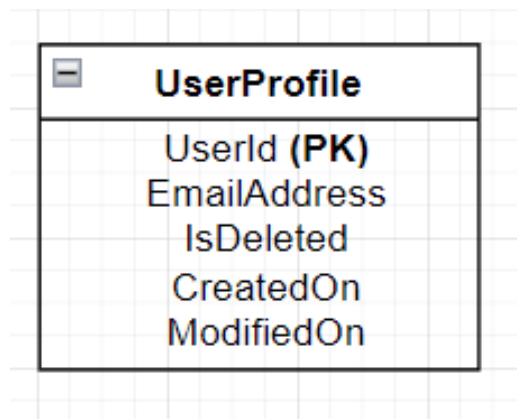
Do przechowywania haseł, usługa wykorzystuje algorytm haszujący Argon2 z użyciem soli.

UserProfile.API

localhost:8015/swagger



Baza danych: Redis



UserProfile.API to lekka usługa obsługująca pobieranie danych użytkownika wyświetlanych przez klienta na jego profilu.

Serwis wymaga bycia zalogowanym.

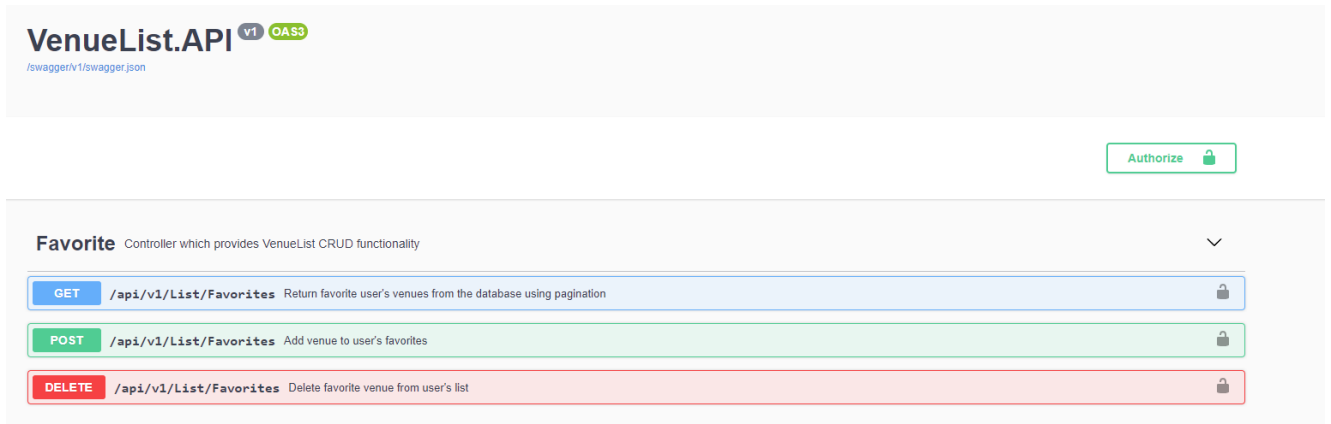
Serwis konsumuje następujące zdarzenia:

- USER_EMAIL_CHANGED - gdy użytkownik zmieni adres email, serwis odbiera to zdarzenie i aktualizuje tą informację w swojej lokalnej bazie

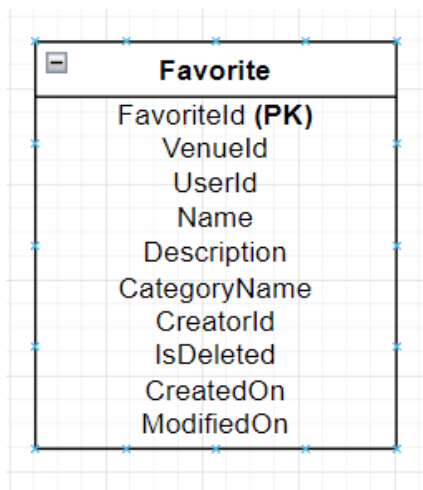
Serwis posiada testy jednostkowe.

VenueList.API

localhost:8003/swagger



Baza danych: MongoDB



Powyższa usługa odpowiada za listę ulubionych miejsc użytkownika. Umożliwia na niej operację pobrania, dodania oraz usunięcia z listy.

Serwis wymaga bycia zalogowanym.

Serwis produkuje następujące zdarzenia:

- VENUE_ADDED_TO_FAVORITES - gdy miejsce zostanie dodane do listy ulubionych
- VENUE_DELETED_FROM_FAVORITES - gdy miejsce zostanie usunięte z listy ulubionych

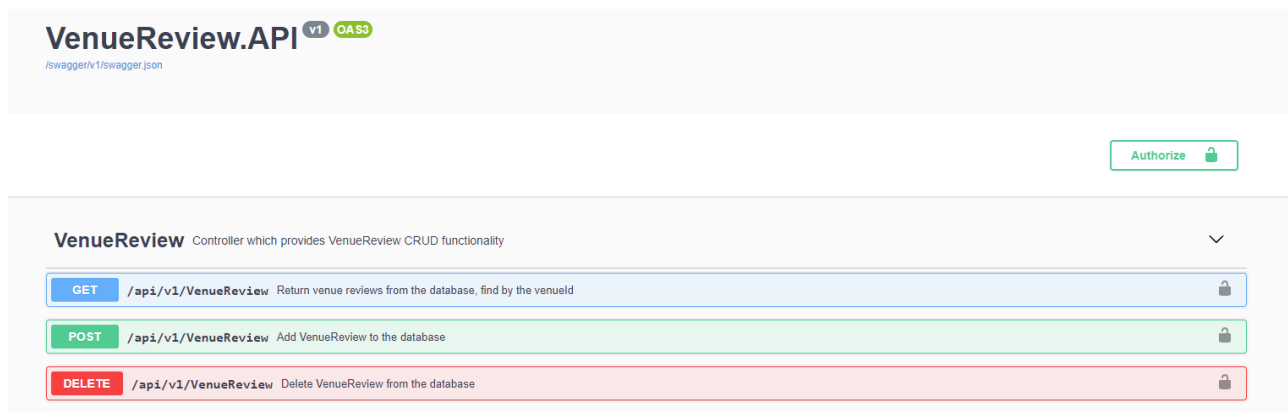
Serwis konsumuje następujące zdarzenia:

- VENUE_DELETED - gdy miejsce zostaje usunięte, należy usunąć wszystkie skojarzone z nim polubione miejsca w bazie lokalnej

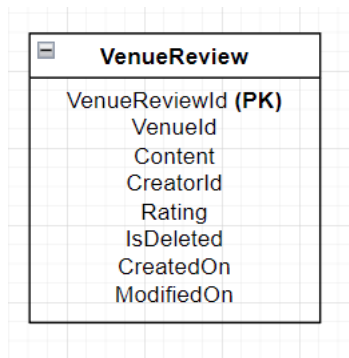
Na starcie serwisu oraz co kolejne 24 godziny, serwis odpytuje Category.API o wszystkie kategorie, żeby przechować je w pamięci podręcznej (kategorie są wymagane podczas tworzenia i aktualizacji miejsc).

VenueReview.API

localhost:8006/swagger



Baza danych: MongoDB



Powyższa usługa odpowiada za ocenianie miejsc. Umożliwia pobieranie opinii na temat konkretnej miejscówki, dodanie własnej opinii oraz usunięcie jej.

Serwis wymaga bycia zalogowanym.

Serwis produkuje następujące zdarzenia:

- VENUE_REVIEW_ADDED - gdy nowa opinia zostanie dodana do bazy lokalnej
- VENUE_REVIEW_DELETED - gdy opinia zostanie usunięta z bazy lokalnej

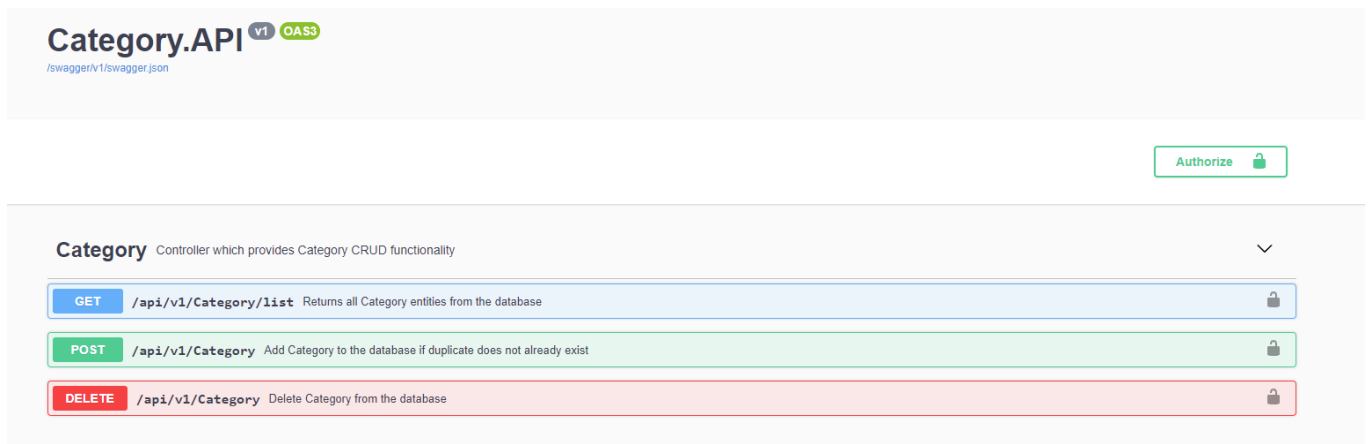
Serwis konsumuje następujące zdarzenia:

- VENUE_DELETED - gdy miejscówka zostanie usunięta, usuwa wszystkie skojarzone z nią opinie

Serwis posiada testy jednostkowe.

Category.API

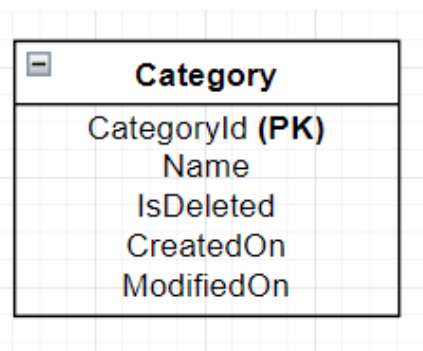
localhost:8020/swagger



The image shows the Swagger UI for the Category.API. At the top, it says "Category.API v1 OAS3" with a link to "/swagger/v1/swagger.json". There is an "Authorize" button with a lock icon. Below this, the "Category" controller is listed, described as "Controller which provides Category CRUD functionality". It contains three endpoints:

- GET** `/api/v1/Category/list` Returns all Category entities from the database
- POST** `/api/v1/Category` Add Category to the database if duplicate does not already exist
- DELETE** `/api/v1/Category` Delete Category from the database

Baza danych: MongoDB



Kategorie, które są shardcodowane w bazie:

- *FOOD*
- *CULTURAL*
- *SPORT*
- *NATURE*

Usługa jest serwisem słownikowym i przechowuje informacje o dostępnych kategoriach miejscówek. Korzysta z nich m.in. usługa Venue oraz VenueList. Udostępnia endpoint do pobierania kategorii oraz do tworzenia i usuwania.

Serwis produkuje następujące zdarzenia:

- CATEGORY_ADDED - gdy nowa kategoria zostanie utworzona
- CATEGORY_DELETED - gdy istniejąca kategoria zostanie usunięta

Serwis posiada testy jednostkowe.

VenueAlgorithm.API

GET /venue/algorithm/venues

Zwraca najbardziej dopasowane do użytkownika miejsca

Returns:

venueIds - lista id dopasowanych miejsc

success - true jeśli zapytanie wykonane poprawnie

PUT /venue/algorithm/like

Tworzy relację pomiędzy miejscem a użytkownikiem

Query params:

venueId - id polubionego miejsca

Returns:

success - true jeśli zapytanie wykonane poprawnie

Serwis konsumuje następujące zdarzenia:

- VENUE_CREATED - gdy zostanie stworzona miejscówka, zapisują ją w bazie grafowej
- VENUE_UPDATED - gdy miejscówka zostanie zaktualizowana, aktualizowana jest również w bazie grafowej
- VENUE_DELETED - gdy miejscówka zostanie usunięta, usuwana jest również z bazy grafowej
- VENUE_ADDED_TO_FAVORITES - gdy miejscówka dodawana jest do ulubionych przez użytkownika, w bazie grafowej tworzona jest relacja
- VENUE_DELETED_FROM_FAVORITES - gdy miejscówka usuwana jest z ulubionych przez użytkownika, w bazie grafowej usuwa się relację
- USER_CREATED - gdy zostanie stworzony nowy użytkownik, zapisuje się go w bazie grafowej

Usługa umożliwia tworzenie relacji pomiędzy miejscami wraz z funkcjonalnością zwracania miejsc najbardziej dopasowanych pod użytkownika.

Zawiera połączenie z bazą grafową *neo4j*, w której znajdują się miejsca, użytkownicy oraz relacje pomiędzy użytkownikami i miejscami. Istnieją dwie możliwe relacje:

- Polubione miejsce,
- Zapisane miejsce.

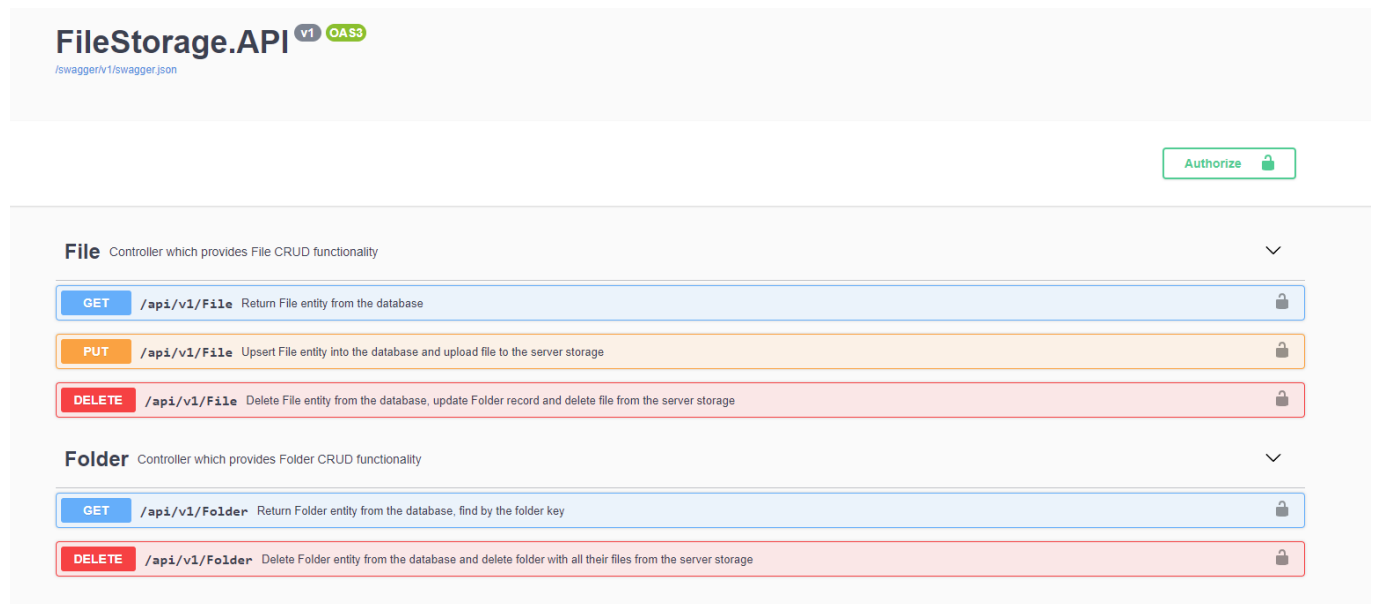
Na podstawie tych relacji działa system rekomendacyjny typu *Collaborative Filtering*. Sprawdzane są miejsca, które polubili lub zapisali użytkownicy mający wspólne polubione lub zapisane miejsca. Dzięki temu możliwe jest wyszukiwanie miejsc dedykowanych dla danego użytkownika a nie losowe ich wyświetlanie.

Rekordy w grafowej bazie danych są przechowywane w następujący sposób:

- Węzły:
 - typu *User* - zawierają pola: *userId*, *name*,
 - typu *Venue* - zawierają pola: *venueId*, *name*, *category*, *occupancy*, *style*
- Relacje:
 - typu *liked*,
 - typu *saved*.

FileStorage.API

localhost:8050/swagger



Usługa umożliwia zarządzanie i przechowywanie plików z pozostałych serwisów. Udostępnia takie punkty końcowe jak pobierz, dodaj/nadpisz lub usuń plik. Dodatkowo możliwe jest pobranie całego folderu oraz usunięcie go razem z zawieranymi plikami.

Usługa nie wymienia żadnych zdarzeń asynchronicznie - jedynie Venue.API synchronicznie odpytuje powyższe API w celu otrzymania lub wrzucenia plików przypisanych do danej miejscówki.

Serwis posiada testy jednostkowe.

5. Opis aplikacji klienckiej

Aplikacja kliencka składa się z trzech głównych funkcjonalności:

-Formularz służący do dodawania miejsca do bazy danych -Widok ze swiperem pozwalającym przesuwając miejsca w prawo w celu zainteresowania się nią , oraz w lewo w celu odrzucenia. Jest tutaj także możliwość polubienia danego miejsca po kliknięciu na ikonę serca, oraz podglądu szczegółów miejsca po kliknięciu na kartę. -Widok z wszystkimi polubionymi miejscami wraz z opisem szczegółowym, zdjęciami, oraz innymi informacjami.

6. Biblioteki współdzielone

W trakcie pisania kodu poszczególnych serwisów napotkaliśmy problem wielokrotnie powielonego kodu stąd padł pomysł, żeby wydzielić go do osobnych bibliotek, które następnie były dołączane do projektów jako pliki .dll.

Library.Shared

Biblioteka, która zawiera logikę współdzieloną dla wszystkich serwisów.

- Kontrakty dla współdzielonych usług (klient HTTP, bazowy kontroler, klient odczytujący zasoby osadzone)
- Statycznie typowane modele konfiguracyjne
- Obsługa wyjątków i walidacji
- Implementacja cachingu
- Współdzielone modele paginacji, zapytań (requestów) i odpowiedzi (responsów)
- Specyficzne wyjątki
- Implementacja usługi do wysyłania i odbierania zdarzeń
- Implementacja wspólnego loggera

Library.Shared.Models

Struktura projektu wygląda tak, że każdy serwis ma tutaj swój własny folder, a w nim znajdują się modele, które są współdzielone z innymi serwisami.

- *Dtos* - modele transferowe (np. podczas transferu HTTP w komunikacji synchronicznej serwis-serwis)
- *Enums* - typy wyliczeniowe
- *Events* - modele zdarzeń, które są przesyłane między serwisami w sposób asynchroniczny

Library.EventBus

Biblioteka, która zawiera wszystkie podstawowe modele, abstrakcje i implementacje dotyczące obsługi message brokera.

W przypadku tego projektu, jako message broker wybraliśmy technologię Kafka - jest ona najlepszym wyborem, jeśli mamy do czynienia z rozbudowaną architekturą mikrousług, jednak w naszym wypadku chcieliśmy po prostu poznać jego możliwości.

Można tutaj wyróżnić następujące obiekty:

- *Event* - bazowa klasa każdego zdarzenia asynchronicznego
- *EventBusTopics* - statyczna klasa zawierająca nazwy wszystkich kanałów w message brokerze
- *EventFactory* - fabryka abstrakcyjna do tworzenia konkretnych zdarzeń
- *EventType* - typ wyliczeniowy zawierający wszystkie typy zdarzeń jakie są obsługiwane w systemie
- *KafkaEventConsumer* - implementacja konsumenta zdarzeń przychodzących z Kafki

- *KafkaEventPublisher* - implementacja producenta zdarzeń wysyłanych do Kafki
- *IEventConsumer* - abstrakcja konsumenta zdarzeń
- *IEventPublisher* - abstrakcja producenta zdarzeń

Library.Database

Biblioteka używana przez serwisy, które korzystają z relacyjnej bazy danych. Zawiera implementacje dotyczące:

- łączenia z bazą danych
- obsługi transakcji
- DI odpowiednich interfejsów

SimpleFileSystem

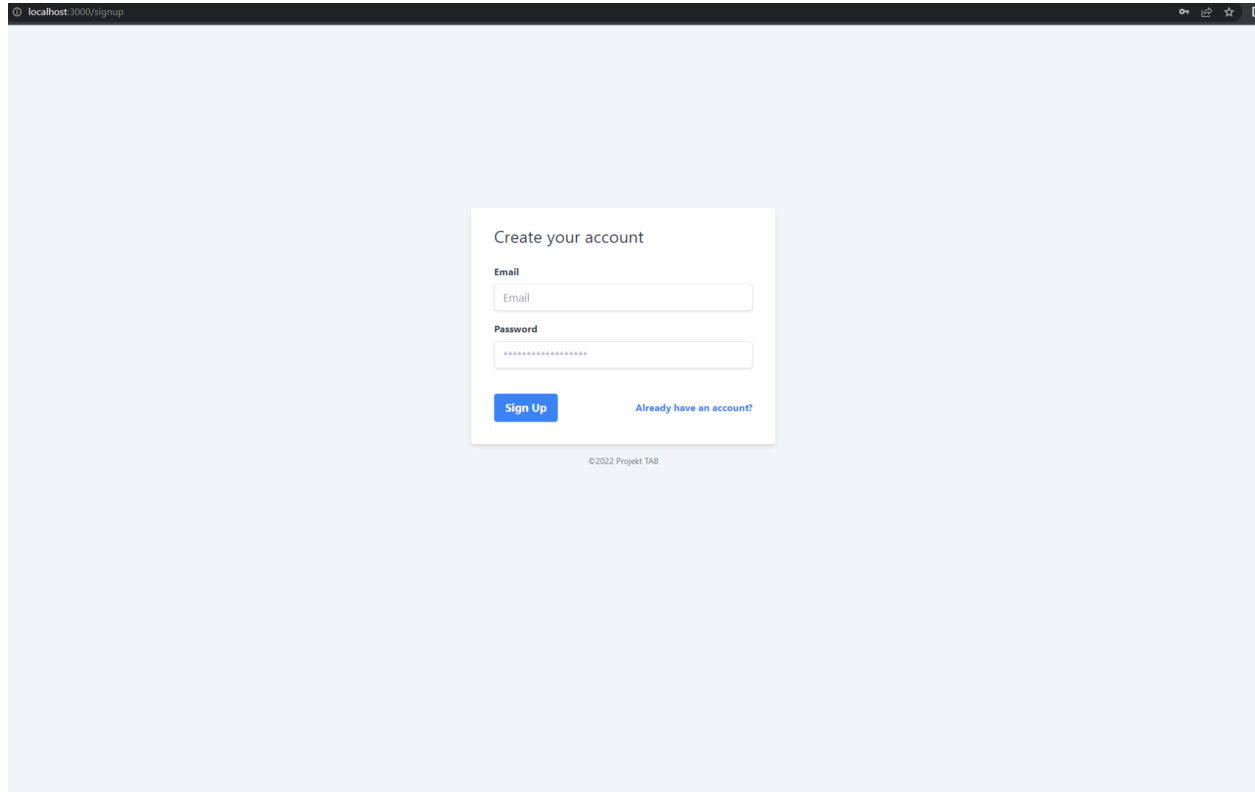
Prosta biblioteka będąca adapterem na operacje na plikach.

SimpleFileSystem.DependencyInjection

Mała biblioteka dająca możliwość wstrzykiwania odpowiednich kontraktów z biblioteki SimpleFileSystem.

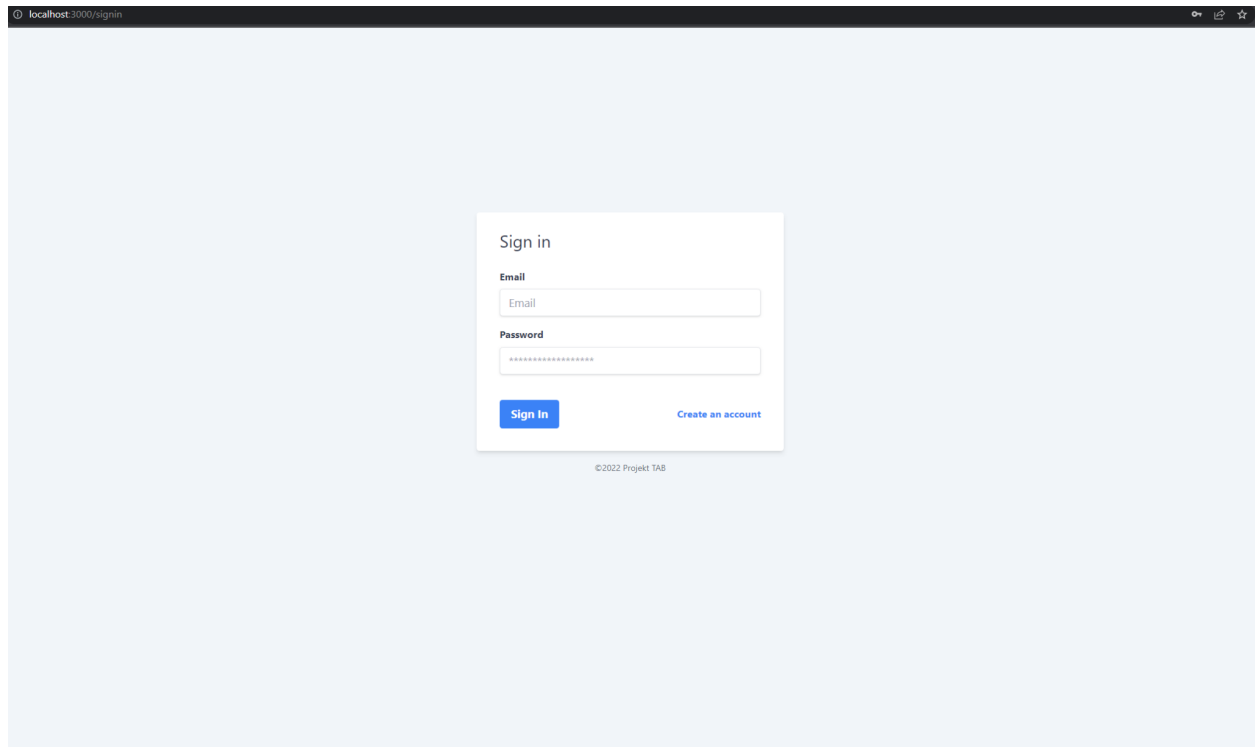
7. Instrukcja obsługi

1. Utworzyć konto podając swój email, oraz hasło:



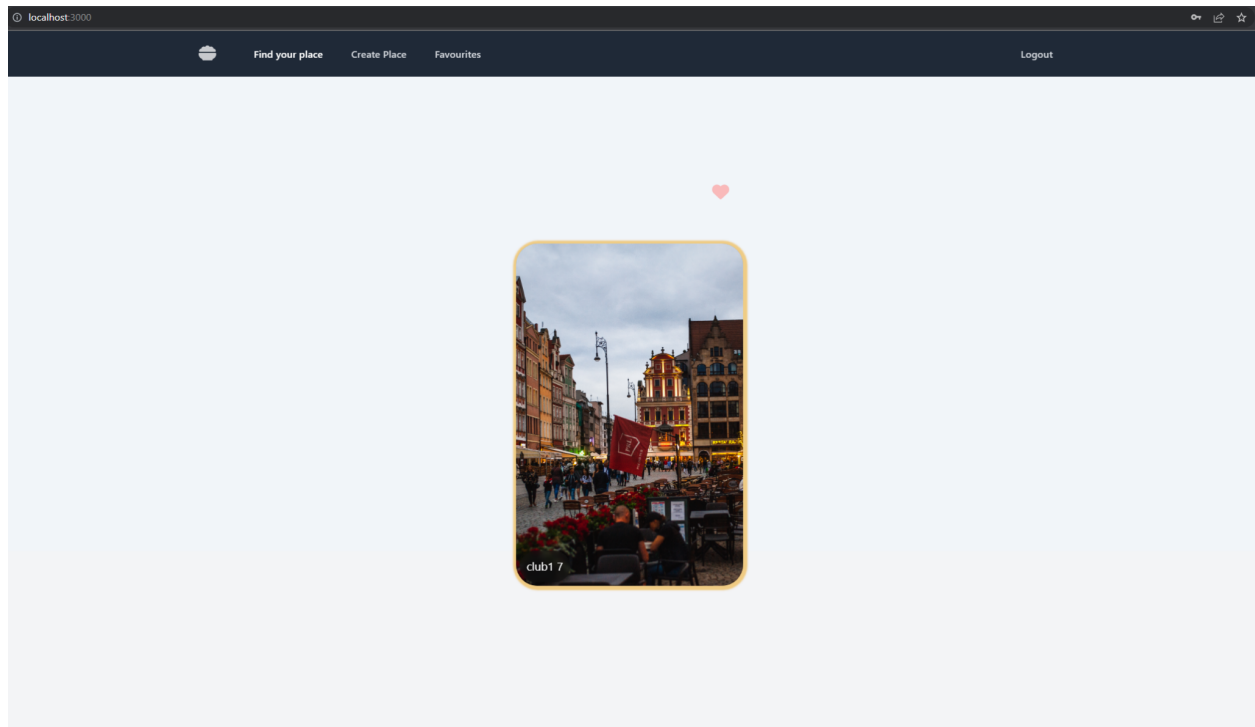
The screenshot shows a web browser window with the address bar displaying 'localhost:3000/signup'. The main content area has a light blue background. In the center, there is a white card titled 'Create your account'. The card contains two input fields: 'Email' and 'Password'. The 'Email' field has a placeholder text 'Email'. The 'Password' field has a placeholder text 'Password' and a series of dots indicating a password field. Below the 'Email' field is a blue button labeled 'Sign Up'. To the right of the 'Sign Up' button is a link that says 'Already have an account?'. At the bottom of the card, there is a small copyright notice: '©2022 Projekt TAB'.

2. Kiedy już mamy swoje konto należy się zalogować do aplikacji:

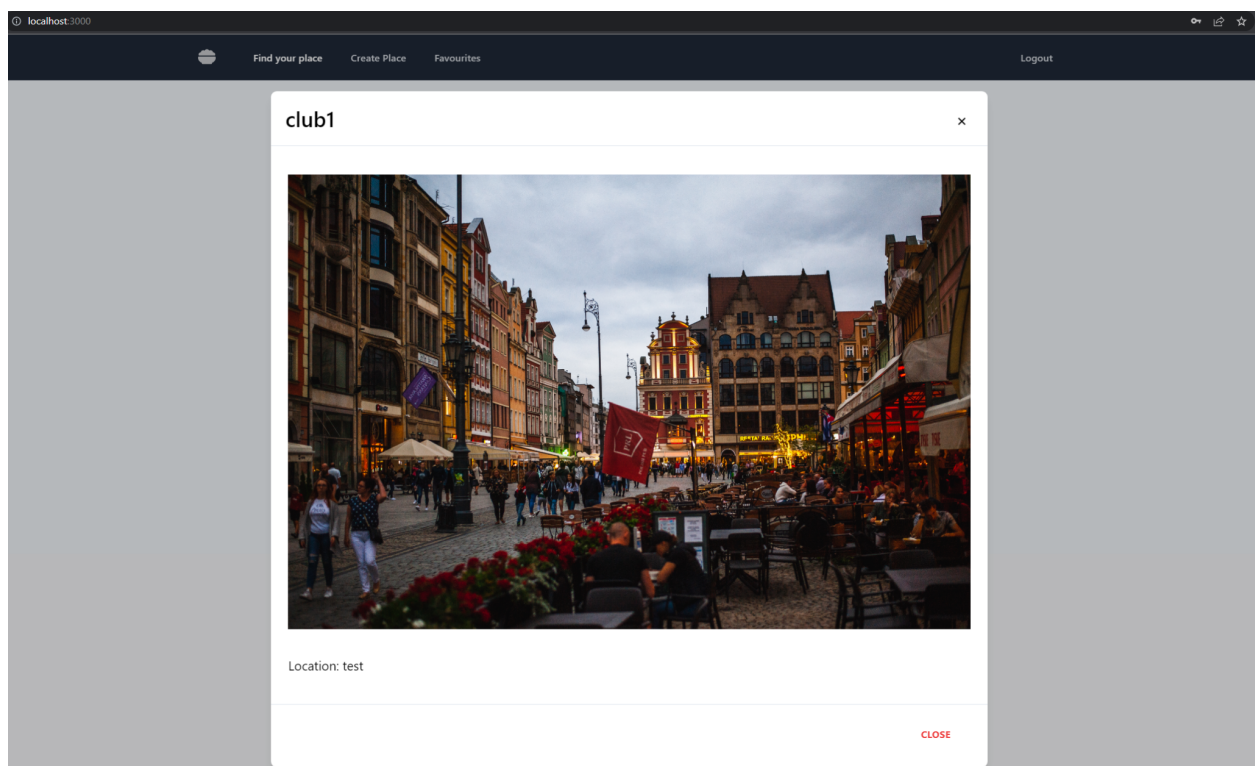


The screenshot shows a web browser window with the address bar displaying 'localhost:3000/signin'. The main content area has a light blue background. In the center, there is a white card titled 'Sign in'. The card contains two input fields: 'Email' and 'Password'. The 'Email' field has a placeholder text 'Email'. The 'Password' field has a placeholder text 'Password' and a series of dots indicating a password field. Below the 'Email' field is a blue button labeled 'Sign In'. To the right of the 'Sign In' button is a link that says 'Create an account'. At the bottom of the card, there is a small copyright notice: '©2022 Projekt TAB'.

3. Po zalogowaniu ukaże nam się główna zakładka aplikacji:

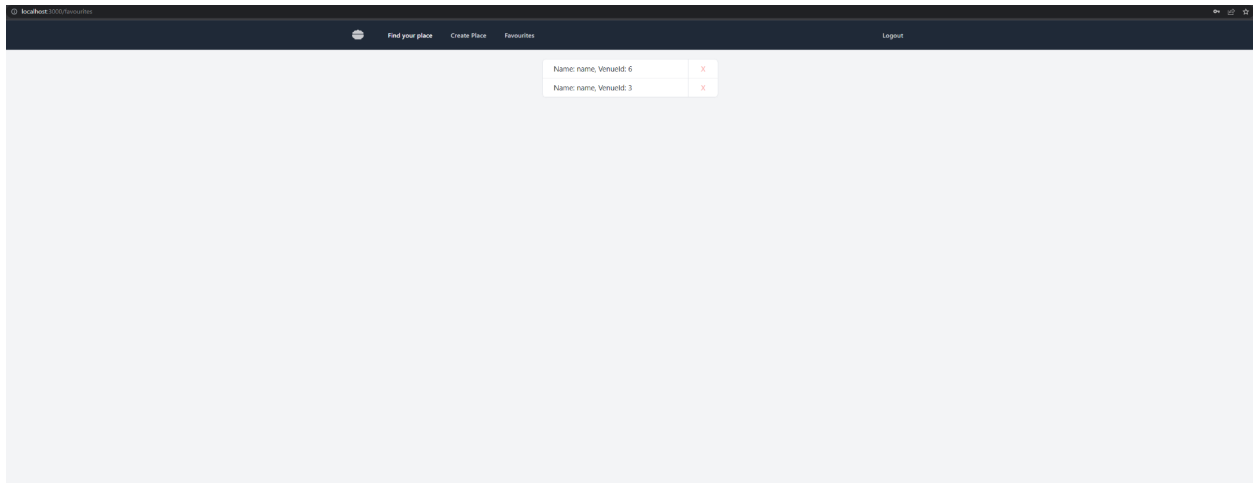


Wyświetlane są tutaj miejsca, które mogą nas zainteresować. Jeśli jakieś miejsce nam się podoba, to przesuwamy kartę w prawo, jeśli zaś nie jesteśmy nim zainteresowani, to przesuwamy w lewo. W celu zapoznania się ze szczegółami miejsca należy kliknąć w dowolnym miejscu na kartę. Otworzy nam się wtedy widok szczegółowy miejsca:

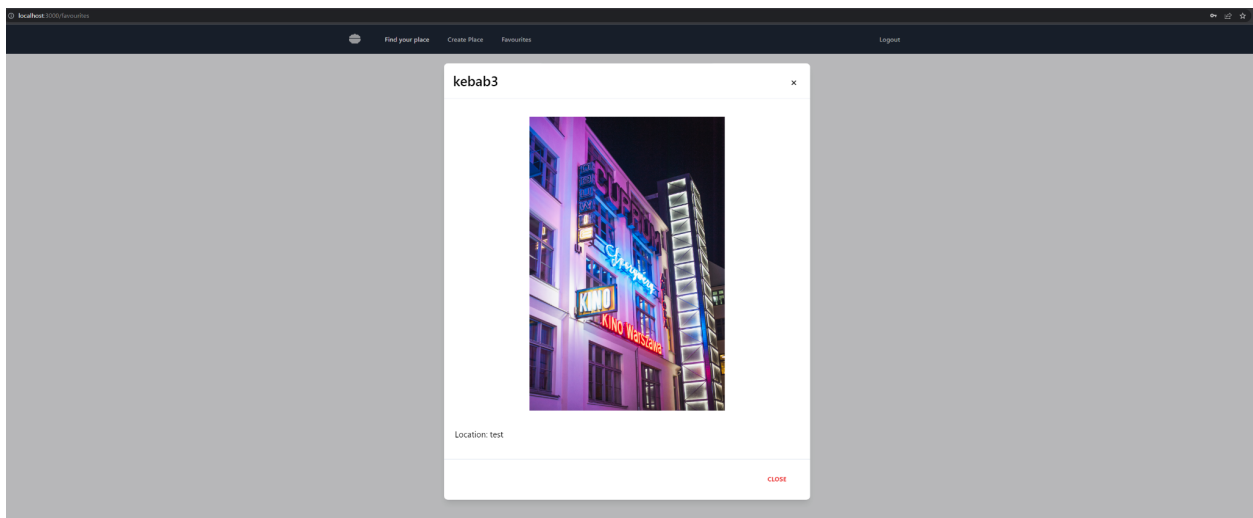


Aby dodać miejsce do listy ulubionych, należy kliknąć różowe serce znajdujące się nad kartą miejsca z prawej strony.

4. Wszystkie polubione przez nas miejsca znajdziemy w zakładce Favourites:

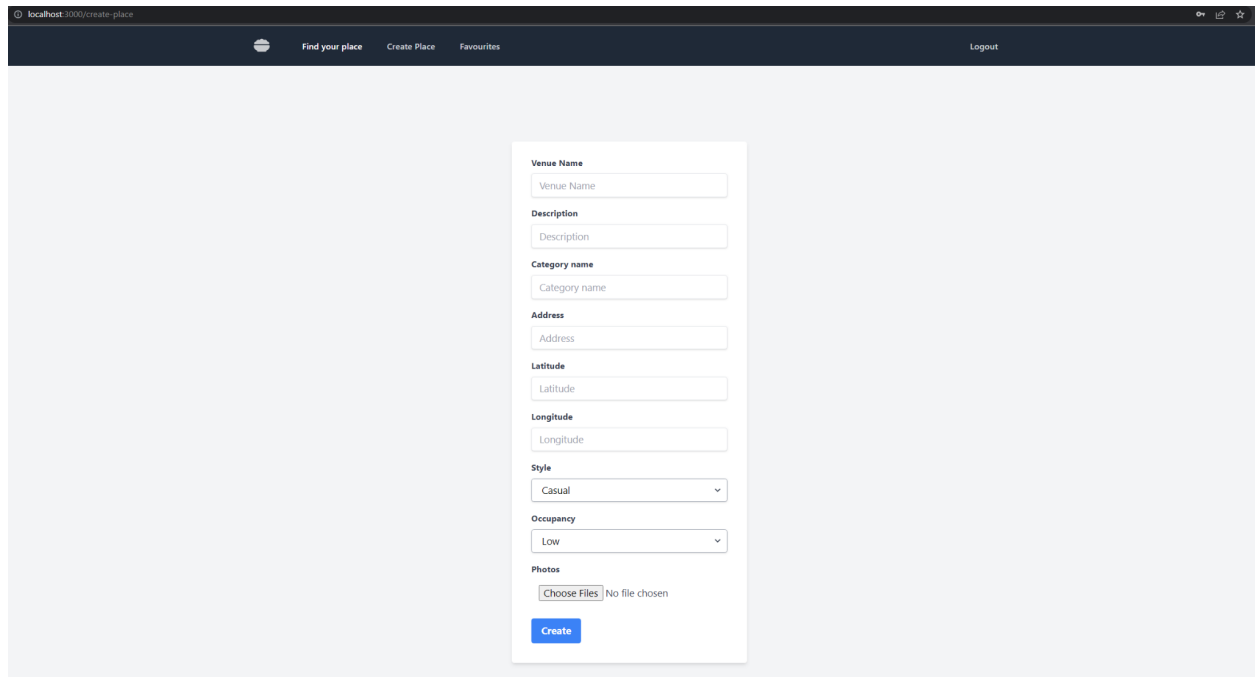


Możemy tutaj kliknąć na interesujące nas miejsce i otworzy się jego widok szczegółowy:



Aby usunąć dane miejsce z listy polubionych należy kliknąć czerwony krzyżyk z prawej strony.

5. Istnieje również możliwość dodania miejsca do bazy danych. Aby to zrobić należy wejść w zakładkę Create Place:



The screenshot shows a web browser window with the address bar displaying 'localhost:3000/create-place'. The page has a dark blue header with a hamburger menu icon, navigation links 'Find your place', 'Create Place', and 'Favourites', and a 'Logout' link on the right. The main content area is light gray and contains a white form for creating a new place. The form fields are as follows:

- Venue Name**: Text input field with placeholder 'Venue Name'.
- Description**: Text input field with placeholder 'Description'.
- Category name**: Text input field with placeholder 'Category name'.
- Address**: Text input field with placeholder 'Address'.
- Latitude**: Text input field with placeholder 'Latitude'.
- Longitude**: Text input field with placeholder 'Longitude'.
- Style**: Dropdown menu with 'Casual' selected.
- Occupancy**: Dropdown menu with 'Low' selected.
- Photos**: File upload section with a 'Choose Files' button and the text 'No file chosen'.
- Create**: A blue button at the bottom of the form.

Uzupełniamy tutaj dane według tytułów pól, oraz na samym dole dodajemy zdjęcia przedstawiające miejsce.