

**UNIwersytet Gdański**  
**Wydział Matematyki, Fizyki i Informatyki**

Bartłomiej Wnuk, Michał Witkowski, Wiktor Sieracki  
285786, 278862, 285769

Kierunek: Informatyka Praktyczna

**System informatyczny wspomagający  
wyszukiwanie i ocenę lokalizacji  
miejskich - Mapzilla**

Praca licencjacka

napisana pod kierunkiem

dr. Adama Kostulaka, prof. UG

Gdańsk 2025

## Spis treści

1	Opis problemu	2
2	Porównanie dostępnych rozwiązań	4
3	Możliwości zastosowania praktycznego	4
4	Wymagania	8
5	Użytkowanie Aplikacji	8
6	Wzorce projektowe	13
7	Użyte technologie	14
8	Testowanie Aplikacji	15

# 1 Opis problemu

Patrząc na ciągle rozwijający się rynek nieruchomości, konsumenci mogą mieć problem z wyborem odpowiadającego im miejsca do zamieszkania. Brak dostępu do kluczowych zasobów w mieście powoduje u ludzi wybieranie mniej płatnych prac oraz pogłębia różnice społeczne. Dzieci, które mają gorszy dostęp do szkoły, często gorzej radzą sobie w nauce oraz mają mniej czasu na pogłębianie swoich zainteresowań.

## Czym jest miasto 15 minut?

Miasto 15 minut to koncepcja urbanistyczna, w której mieszkańcy mają dostęp do najważniejszych udogodnień w zasięgu 15 minut pieszo. Do udogodnień zalicza się szkoły, sklepy spożywcze, apteki, przychodnie oraz miejsca rekreacji.



Rysunek 1: Miasto 15-minut [4]

## Jak osiągnąć status miasta 15 minut?

Żeby dobrze rozplanować gdzie zbudować nowe miejsca codziennego użytku należy śledzić położenie zbudowanych już struktur. W aplikacji można łatwo zauważyć miejsca, w których jest gorszy dostęp do sklepów. Czy w pobliżu znajduje się szkoła.

Dzięki aplikacji władze miasta mogą monitorować takie krytyczne miejsca i na jej podstawie rozplanowywać inwestycje na przyszłe lata. Należy zauważyć, że sprawna komunikacja miejska nie jest sposobem na zostanie miastem 15 minut. Trzeba jednak pójść o krok dalej i skupić się na inteligentnym rozmieszczeniu każdej budowli. Warto też trzymać się zasad takich jak budowanie większej ilości przestrzeni dla ludzi niż dla samochodów.

## Miasta, które są miastami 15 minut

Przykładowe miasta spełniające wymagania miasta 15 minut to:

- Barcelona - jest prekursorem jeżeli chodzi o rozwiązania miasta 15 minut. Jako miasto liczące ponad półtora miliona mieszkańców, osiągnęło miano miasta 15 minut dzięki swojej

wielkiej gęstości zaludnienia oraz specyficznej zabudowie bloków widocznej na Rysunku 2. Na terenie miasta i okolic (Sant Adrià de Besòs, L'Hospitalet de Llobregat, Esplugues de Llobregat y Cornellà de Llobregat) obowiązuje tzw. Strefa Niskiej Emisji (Zona de Bajas Emisiones – ZBE). Oznacza to ograniczenie wjazdu na teren ZBE dla pojazdów osobowych niespełniających wymogów dot. emisji spalin od poniedziałku do piątku w godzinach od 7 do 20. [1]



Rysunek 2: Barcelona [5]

- Melbourne - wspiera lokalne firmy,
- Portland - buduje mieszkania komunalne w bogatych dzielnicach,
- Paryż - dba o rozbudowę zielonych terenów przyszkolnych.

#### **Zalety miasta 15 minut**

W opisie problemu przedstawiono niektóre wady braku dostępu do kluczowych struktur w mieście. Z kolei do zalet należą:

- mniejsze zanieczyszczenie miasta spalinami,
- poprawa jakości życia mieszkańców,
- oszczędność czasu przeznaczanego na dojazdy,
- zwiększenie integracji społecznej,
- większa aktywność fizyczna mieszkańców.

## 2 Porównanie dostępnych rozwiązań

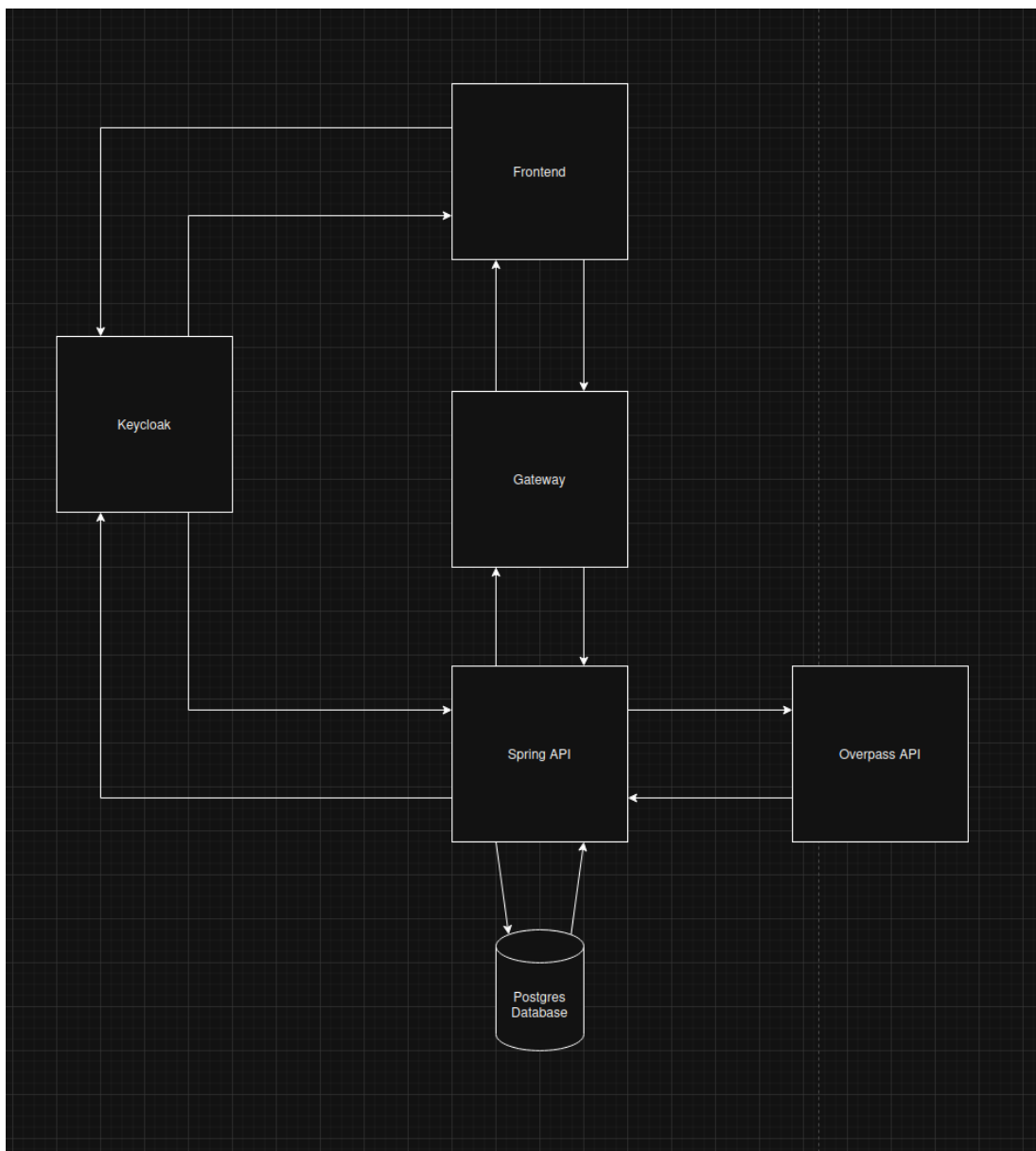
**Mapy Google** — pozwalają zobaczyć znajdujące się dookoła udogodnienia, lecz nie skupiają się na ich kompleksowej analizie i nie przedstawiają ich w dogodny sposób dla użytkownika planującego wybór miejsca zamieszkania.

**15-min city** — (<https://app.developer.here.com/15-min-city-map/>) mała aplikacja działająca tylko na terenie stanów zjednoczonych. Brak możliwości wyboru pożądanych miejsc.

## 3 Możliwości zastosowania praktycznego

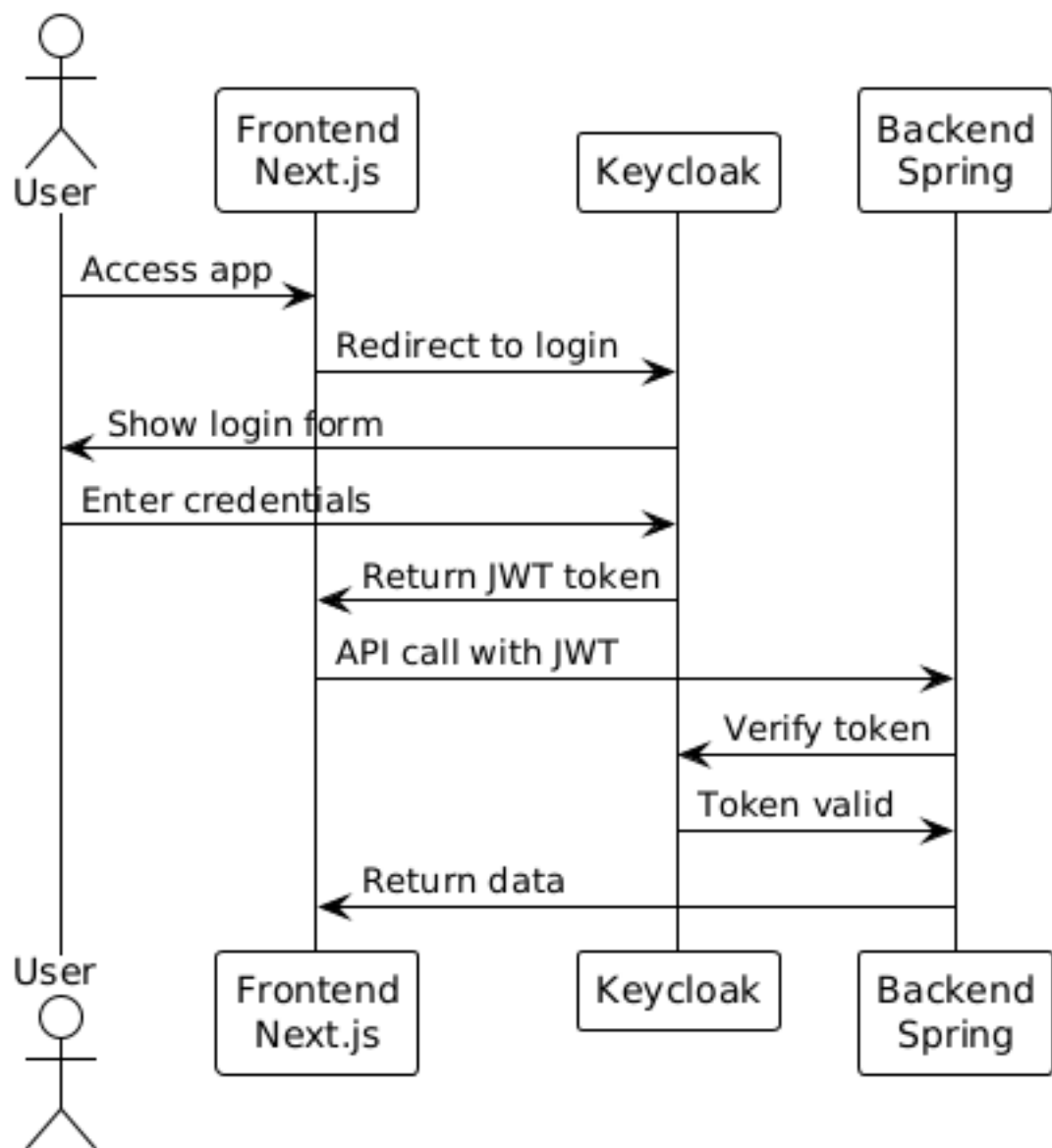
- **Dla mieszkańców** — pomoc w wyborze odpowiedniego miejsca do zamieszkania.
- **Dla deweloperów** — planowanie inwestycji mieszkaniowych w najlepszych lokalizacjach, poprawiających komfort życia lokatorów.
- **Dla władz miasta** — rozwój infrastruktury społecznej oraz planowanie przestrzenne zgodne z potrzebami mieszkańców.

Poniżej widać struktury komunikacji poszczególnych serwisów w naszej aplikacji.



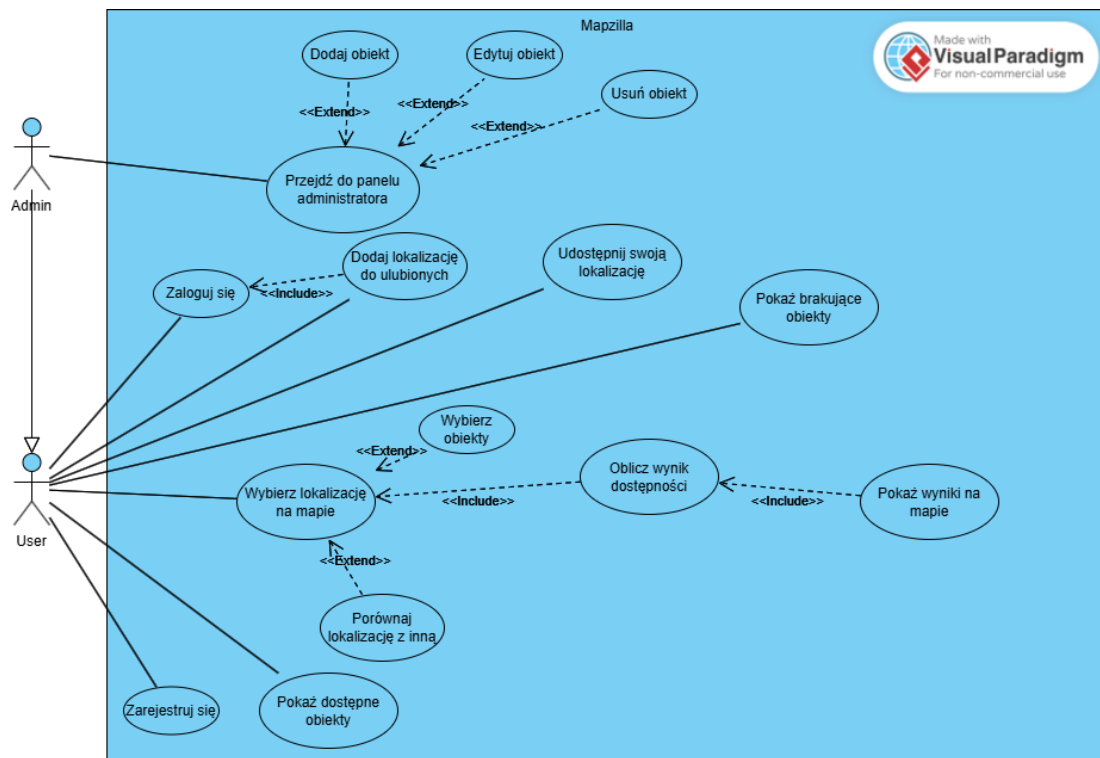
Rysunek 3: Struktura projektu

Diagram autentykacji użytkownika.



Rysunek 4: Autentykacja

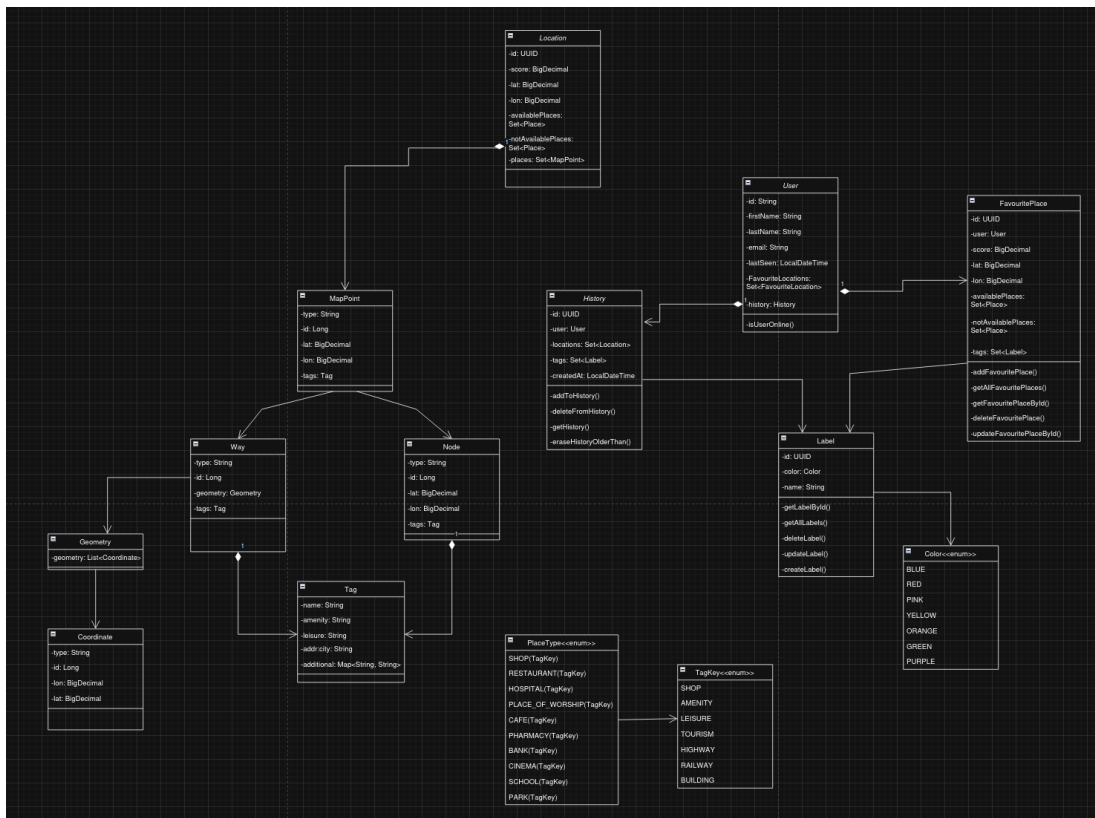
Przypadki użycia aplikacji.



Rysunek 5: Przypadki użycia

Diagram Klas.





Rysunek 6: Diagram Klas

## 4 Wymagania

### Funkcjonalne

Mapa w Aplikacji znajduje obiekty w całej Polsce.

Dostępne jest wiele atrakcji do wyboru z listy.

Możliwa jest rejestracja użytkownika.

Możliwe jest logowanie użytkownika.

### Niefunkcjonalne

Mapa działa płynnie.

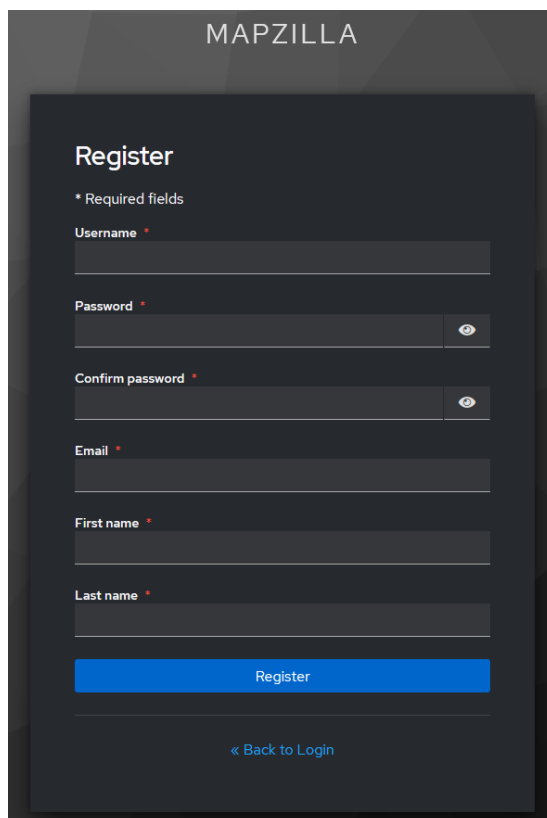
Użytkownik zostaje wylogowany po 5 minutach nieaktywności.

Dane użytkowników są bezpiecznie przechowywane.

## 5 Użytkowanie Aplikacji

### Rejestracja Użytkownika

Aby zarejestrować się w aplikacji, należy podać swoje dane osobowe oraz adres e-mail. Odbywa się to poprzez Keycloak, który jest zintegrowany z aplikacją. Dzięki temu, że użyliśmy sprawdzonego pośrednika do zarządzania użytkownikami, zapewniamy większe bezpieczeństwo ich danych.

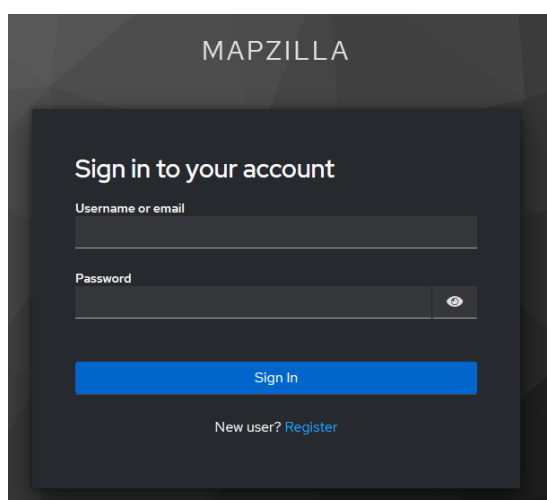


The image shows a dark-themed registration form titled "MAPZILLA Register". It includes a legend for required fields, followed by input fields for Username, Password, Confirm password, Email, First name, and Last name. Each field has a red asterisk indicating it is required. Password and Confirm password fields have toggle icons for visibility. A blue "Register" button is at the bottom, with a link "« Back to Login" below it.

Rysunek 7: Rejestracja

### Logowanie Użytkownika

Aby zalogować się do aplikacji, należy podać swoją nazwę użytkownika oraz hasło. Po zalogowaniu użytkownik zostaje przekierowany do strony głównej aplikacji. Stamtąd może dodawać do ulubionych lokalizacje.

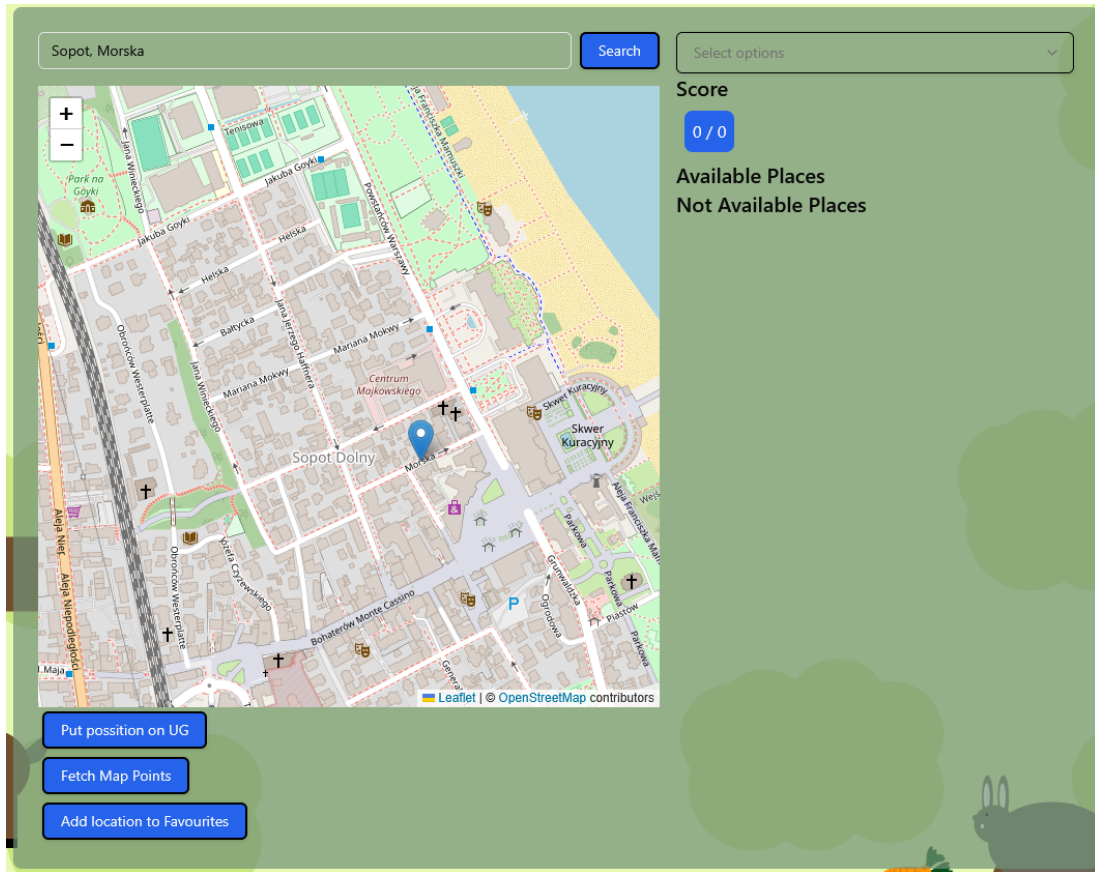


The image shows a dark-themed login form titled "MAPZILLA Sign in to your account". It includes input fields for "Username or email" and "Password". The Password field has a toggle icon for visibility. A blue "Sign In" button is at the bottom, with a link "New user? Register" below it.

Rysunek 8: Logowanie

### Wyszukiwanie miejsc po nazwie

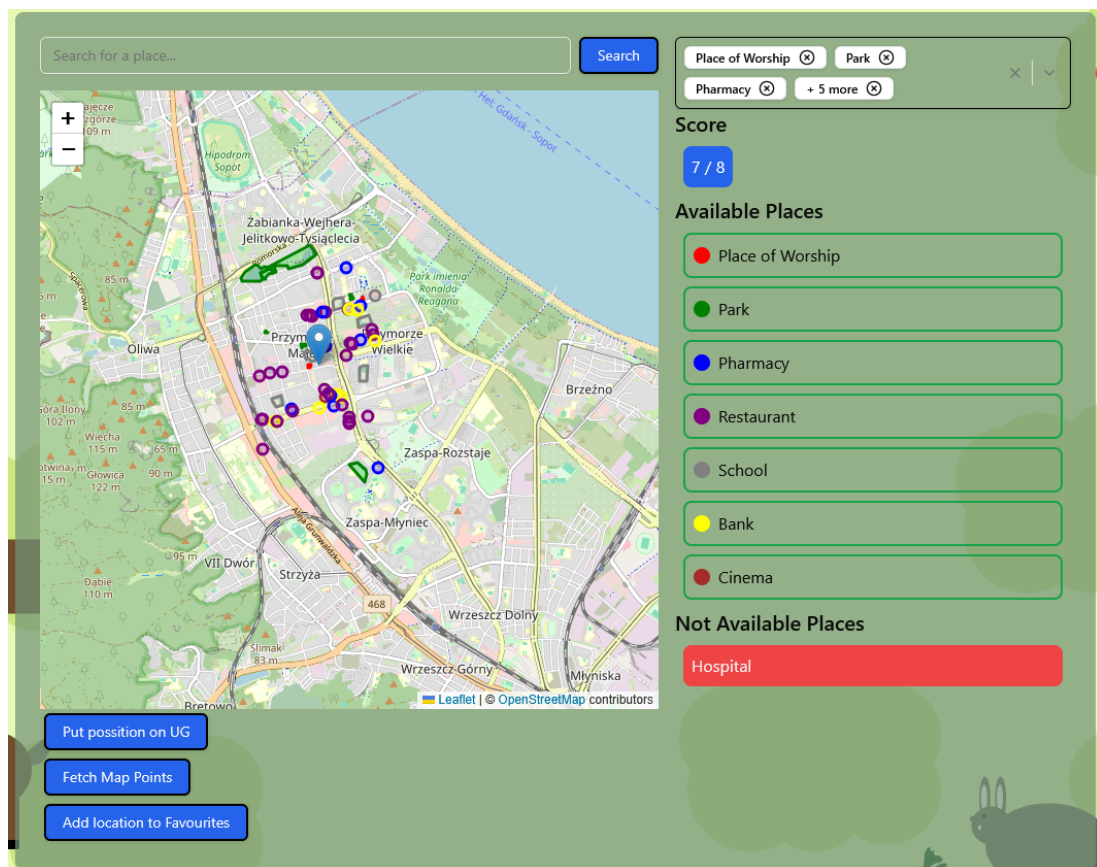
Dzięki wyszukiwarce użytkownik może znaleźć interesujące go miejsca. Wystarczy wpisać nazwę miejsca, miasta lub ulicy, a aplikacja przeniesie nas do wybranego miejsca, z którego możemy dalej szukać interesujących nas obiektów.



Rysunek 9: Wyszukiwanie miejsca

### Mapa z Lokalizacjami

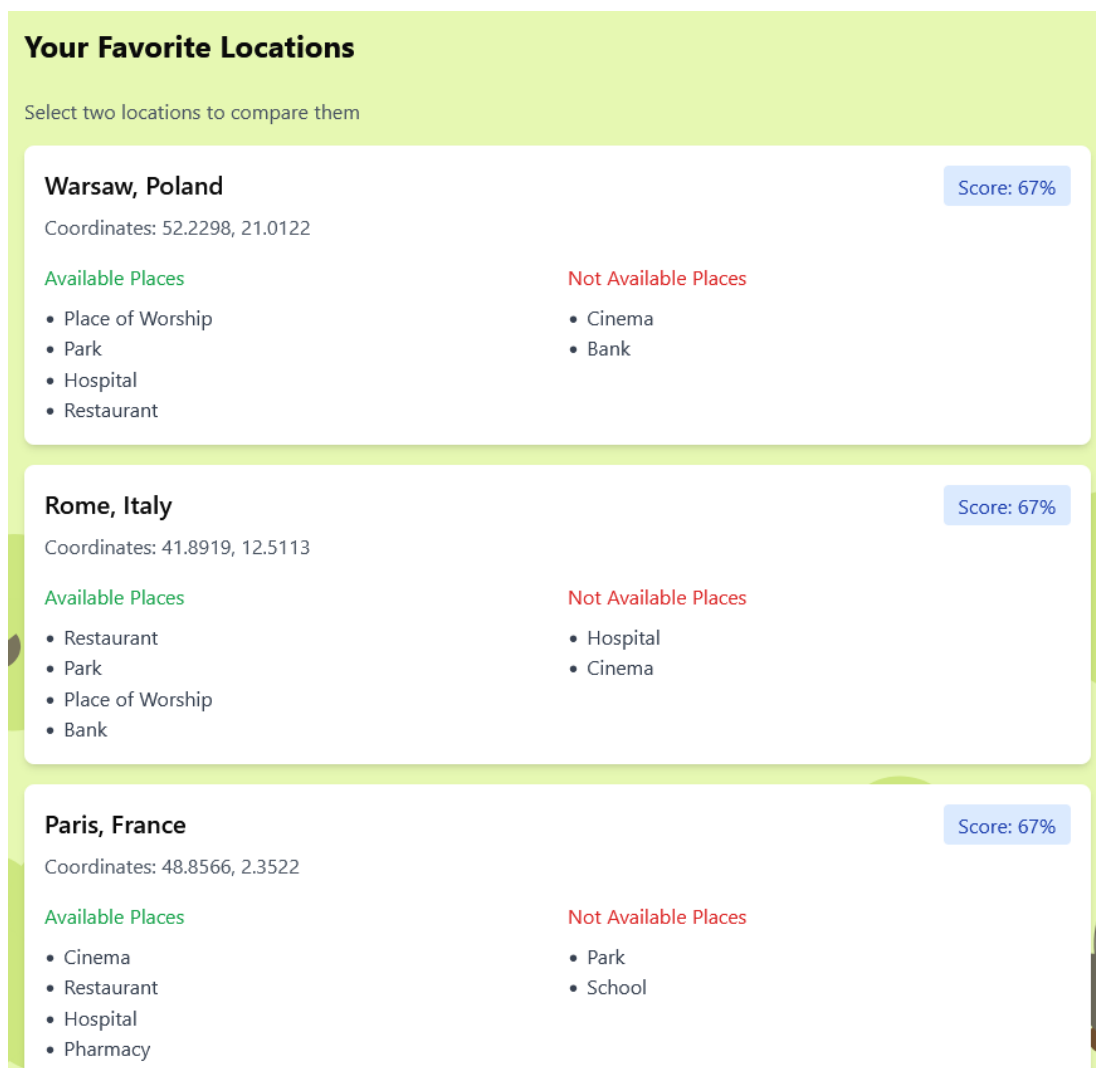
Głównym działaniem naszej aplikacji jest mapa, która pozwala na przeglądanie oraz wyszukiwanie interesujących nas lokalizacji. Użytkownik może wybrać interesujące go miejsca i wyszukać ich w okolicy.



Rysunek 10: Mapa

### Ulubione miejsca

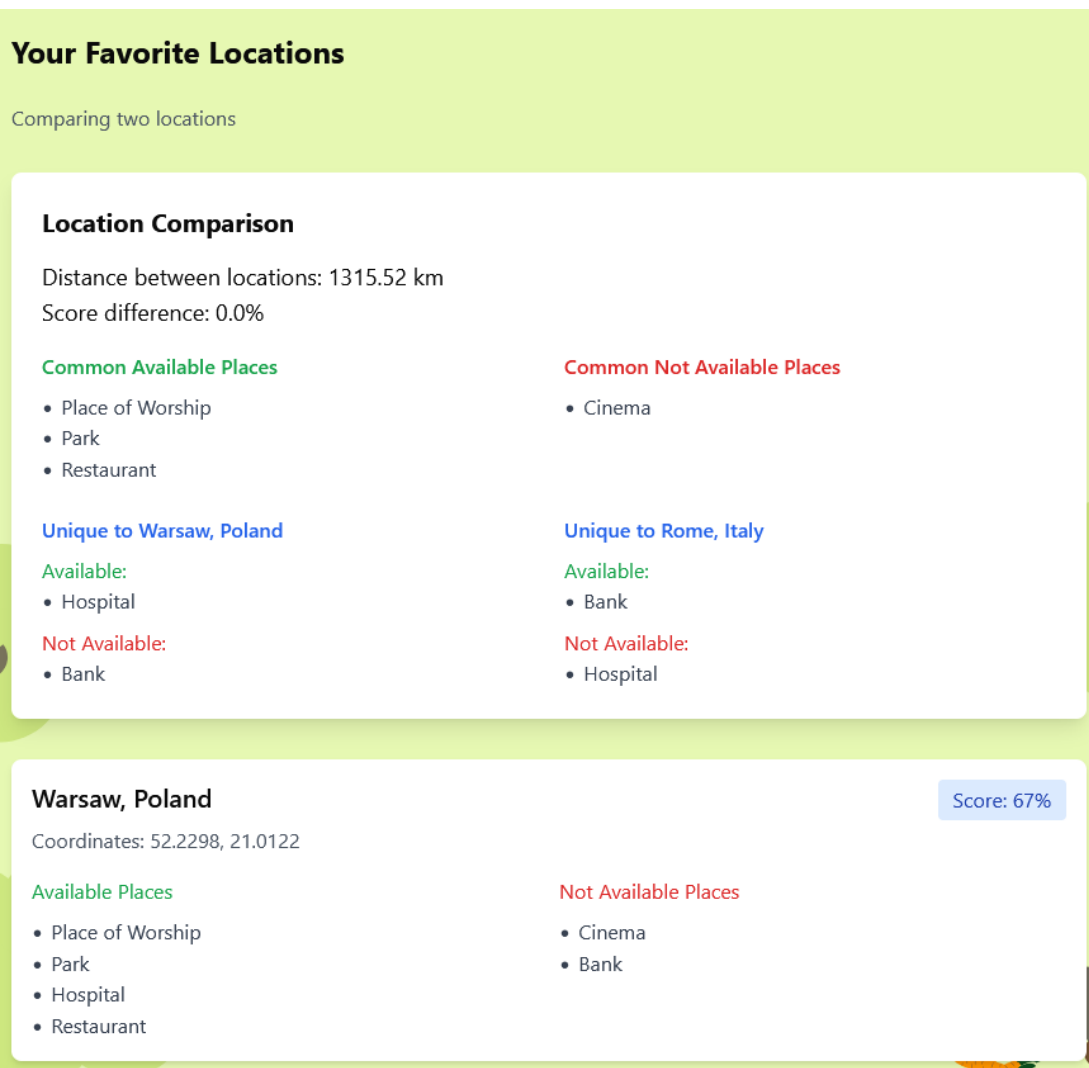
Po kliknięciu w nazwę użytkownika w prawym górnym rogu, możemy przejść do zakładki ulubione miejsca. Można tam zauważyć listę ulubionych miejsc, które użytkownik dodał do swojego konta. Każde miejsce ma opisany wynik, koordynaty oraz dostępne i niedostępne tam udogodnienia.



Rysunek 11: Ulubione

### Ulubione miejsca

Klikając na dwa interesujące nas ulubione miejsca możemy je ze sobą porównać.



Rysunek 12: Porównywanie ulubionych miejsc

## 6 Wzorce projektowe

Tworząc aplikację korzystaliśmy z wzorca RESTful, który służy do wydajnego posługiwania się i korzystania z HTTP oraz jego standardowych metod takich jak GET, POST, PUT, DELETE, PATCH. Wykorzystujemy wersjonowanie zawarte w adresie oraz statusy HTTP. Z innych dobrych praktyk, w adresach posługujemy się liczbą `/favourite-places`.

Standardową praktyką w Spring Boot, z której korzystamy, jest używanie Dependency Injection, czyli wstrzykiwania zależności do obiektów, co pozwala na luźne powiązanie komponentów, które wspiera modularność, ułatwia testowanie oraz sprawia, że projekt jest łatwiejszy w utrzymaniu. Wzorec Strategy pozwala nam na wybór odpowiednich algorytmów służących do serializacji danych. Spring Boot sam w sobie korzysta z wzorców projektowych takich jak Singleton czy Factory. Oba są powiązane z bean'ami. Pierwszy z nich odpowiada za to, że tworzona i zarządzana jest tylko jedna instancja danej klasy. Ułatwia to zarządzanie cyklem ich życia oraz zarządzaniem zasobami. Natomiast drugi odpowiada za samo tworzenie bean'ów na podstawie definicji konfiguracyjnych.

Wzorec Gateway to bardzo użyteczny i ważny wzorec projektowy w kontekście zabezpieczeń aplikacji. Służy on do enkapsulacji dostępu do zewnętrznego systemu lub usługi. Wprowadza on łatwe rozwijanie aplikacji w stronę architektury opartej na mikroserwisach, gdzie

pełni rolę pojedynczego punktu wejścia dla wszystkich żądań, a następnie przekierowywania ich w odpowiednie miejsca. Może on też pełnić rolę agregowania odpowiedzi, uwierzytelniać, autoryzować oraz logować do aplikacji. W kontekście serwera działającego jako monolit, serwis będący jako Gateway pełni rolę izolowania strony serwera przed niechcianym wejściem, natomiast monolit posiadający całą logikę biznesową aplikacji staje się kolejną usługą.

Tworząc stronę serwera, korzystamy również z wzorca 3-Tier Architecture, który dzieli całą aplikację na trzy różne poziomy: poziom prezentacji, poziom aplikacji oraz poziom danych. Takie podejście pomaga odseparować od siebie moduły i pozwala na większe bezpieczeństwo, dzięki temu, że poziom prezentacji oraz poziom danych nie mogą się ze sobą bezpośrednio kontaktować, co zapobiega atakom hakerskim takim jak SQL Injection. Wpływa to też na niezawodność całego projektu, gdyż błąd na jednym poziomie nie zawsze oznacza błąd na drugim poziomie. Pozwala też na większą skalowalność, każdy z poziomów może skalować się niezależnie od reszty. Podejście to też wpływa na szybkość rozwoju aplikacji, która może być rozwijana na każdym poziomie przez oddzielne zespoły programistów. Na samym poziomie aplikacji korzystamy z podejścia 3-Layer Architecture, który dzieli aplikację na trzy niezależne warstwy: warstwę prezentacji, czyli kontrolery, warstwę logiki biznesowej czyli serwisy oraz warstwę danych czyli repozytoria.

## 7 Użyte technologie

Po stronie serwera posiadamy API napisane w frameworku Javy, Spring Framework. Dokładniej, korzystamy z nakładki Spring Boot, która ma za zadanie uprościć proces tworzenia projektu zawierając w sobie serwery takie jak Tomcat, automatycznie konfiguruje aplikację.

Oprócz tego nasz projekt korzysta z bibliotek takich jak Lombok. Jest to biblioteka, która dostarcza adnotacje, które w sposób znaczący zmniejszają ilość potrzebnego kodu i sprawiają, że sama aplikacja jest przejrzysta i bardziej czytelna. Innym narzędziem, z którego korzystamy jest OpenApi, które dostarcza Swagger, czyli instrument do automatycznego generowania dokumentacji na podstawie kontrolerów, jakie nasza aplikacja posiada.

Do mapowania obiektowo-relacyjnego wykorzystujemy framework Hibernate. Pozwala on na mapowanie obiektów w języku Java na tabele w bazie danych i posługiwanie się operacjami bazodanowymi używając wysokopoziomowego API, dzięki czemu nie musimy pisać niskopoziomowego kodu SQL.

Informacje o użytkownikach oraz logikę rejestracji i logowania użytkowników rozwiązaliśmy za pomocą serwisu Keycloak. Dzięki niemu możemy nadawać użytkownikom odpowiednie role oraz weryfikować ich maile.

Nasza baza danych znajduje się w PostgreSQL, przechowujemy tam informacje o lokalizacjach oraz ich relacje z użytkownikami.

Do napisania warstwy widocznej w przeglądarce użyliśmy biblioteki React. Przy pomocy framework'a Next.js dzielimy pliki na komponenty klienta i serwera. Do budowania responsywnych interfejsów użytkownika użyliśmy frameworka Tailwind CSS. Używamy też gotowych komponentów UI z biblioteki Shadcn.

Do autoryzacji użytkowników w przeglądarce używamy biblioteki Next-Auth. Pozwala na odszyfrowanie tokenów JWT i pozyskanie danych z Keycloak.

Dzięki narzędziu Docker możemy sprawnie konteneryzować nasze serwisy, co pozwala na lepszą kontrolę wersji oraz ułatwia późniejszą skalowalność aplikacji.

## 8 Testowanie Aplikacji

Testowanie aplikacji Mapzilla stanowiło istotny element procesu wytwarzania oprogramowania. Celem testów było sprawdzenie funkcjonalności aplikacji, walidacji danych oraz obsługi błędów. Zastosowano dwa rodzaje testowania:

- **Testy jednostkowe** - automatyczne testy napisane w języku Java z wykorzystaniem biblioteki JUnit oraz narzędzi takich jak Mockito i Spring Boot Test. Przetestowano logikę odpowiedzialną za obsługę żądań związanych z ulubionymi lokalizacjami użytkownika.
- **Testy manualne** - testy wykonywane ręcznie z użyciem narzędzia Postman. Polegały one na wysyłaniu żądań HTTP do endpointów REST API w działającej aplikacji. Sprawdzono poprawność odpowiedzi serwera, walidację danych wejściowych, oraz działanie mechanizmu autoryzacji z wykorzystaniem tokenów JWT.

### Scenariusze testowe

W ramach testowania aplikacji przygotowano zestaw scenariuszy testowych, które miały na celu zweryfikowanie poprawności działania najważniejszych funkcjonalności systemu. Scenariusze te objęły zarówno pozytywne, jak i negatywne przypadki użycia. Poniżej przedstawiono listę metod testowych wraz z ich krótkim opisem:

- **createLocation\_ReturnCreated** - sprawdza, czy po przesłaniu poprawnych danych lokalizacji system poprawnie tworzy nowy wpis.
- **createLocation\_ReturnCreated\_CheckResponse** - sprawdza, czy po przesłaniu poprawnych danych lokalizacji system zwraca poprawne dane, takie jak wartość pola `score`
- **createLocation\_ReturnBadRequest\_Score** - testuje reakcję systemu na przesłanie niepoprawnej wartości parametru `score`.
- **createLocation\_ReturnBadRequest\_Lat** - sprawdza, czy niepoprawna szerokość geograficzna zostaje prawidłowo odrzucona przez walidację danych.
- **createLocation\_ReturnBadRequest\_Lon** - sprawdza, czy niepoprawna długość geograficzna zostaje prawidłowo odrzucona przez walidację danych.
- **getFavouriteLocationsByUserId\_ReturnSuccess** - potwierdza, że użytkownik po zalogowaniu może poprawnie pobrać listę swoich ulubionych lokalizacji.
- **getFavouriteLocationsByUserId\_ReturnSuccess\_Empty** - sprawdza poprawność działania systemu w sytuacji, gdy użytkownik nie ma zapisanych żadnych lokalizacji.
- **deleteFavouriteLocationById\_ReturnSuccess** - testuje poprawność działania operacji usuwania lokalizacji na podstawie ID i oczekuje odpowiedzi z komunikatem sukcesu.
- **deleteFavouriteLocationById\_ReturnNotFound** - weryfikuje, czy system odpowiednio reaguje na próbę usunięcia lokalizacji, która nie istnieje.
- **getFavouriteLocationById\_ReturnSuccess** - sprawdza poprawne pobranie pojedynczej lokalizacji po ID oraz poprawność danych zwróconych w odpowiedzi.
- **getFavouriteLocationById\_ReturnNotFound** - testuje scenariusz, w którym lokalizacja o zadanym id nie istnieje. System powinien zwrócić błąd 404 z odpowiednim komunikatem.
- **updateFavouriteLocationById\_ReturnSuccess** - weryfikuje, czy edycja istniejącej lokalizacji przebiega poprawnie i dane zostają zaktualizowane.
- **updateFavouriteLocationById\_ReturnNotFound** - testuje reakcję systemu na próbę aktualizacji nieistniejącej lokalizacji



- **updateFavouriteLocationById\_BadRequest\_Score** - sprawdza, czy przesłanie niepoprawnej wartości parametru **score** w czasie aktualizacji powoduje błąd walidacji.
- **updateFavouriteLocationById\_BadRequest\_Lat** - sprawdza czy system odpowiednio reaguje na próbę aktualizacji ulubionej lokalizacji z błędną wartością szerokości geograficznej.
- **updateFavouriteLocationById\_BadRequest\_Lon** - sprawdza czy system odpowiednio reaguje na próbę aktualizacji ulubionej lokalizacji z błędną wartością długości geograficznej.

Nazwa Testu	Status
createLocation_ReturnCreated	Done
createLocation_ReturnCreated_CheckResponse	Done
createLocation_ReturnBadRequest_Score	Done
createLocation_ReturnBadRequest_Lat	Done
createLocation_ReturnBadRequest_Lon	Done
getFavouriteLocationsByUserId_ReturnSuccess	Done
getFavouriteLocationsByUserId_ReturnSuccess_Empty	Done
deleteFavouriteLocationById_ReturnSuccess	Done
deleteFavouriteLocationById_ReturnNotFound	Done
getFavouriteLocationById_ReturnSuccess	Done
getFavouriteLocationById_ReturnNotFound	Done
updateFavouriteLocationById_ReturnSuccess	Done
updateFavouriteLocationById_ReturnNotFound	Done
updateFavouriteLocationById_BadRequest_Score	Done
updateFavouriteLocationById_BadRequest_Lat	Done
updateFavouriteLocationById_BadRequest_Lon	Done

Tabela 1: Status testów aplikacji Mapzilla

### Przetestowane endpointy API

W ramach testów manualnych przetestowano następujące endpointy REST API, odpowiedzialne za operacje CRUD systemu informatycznego:

#### POST /realms/Mapzilla/protocol/openid-connect/token

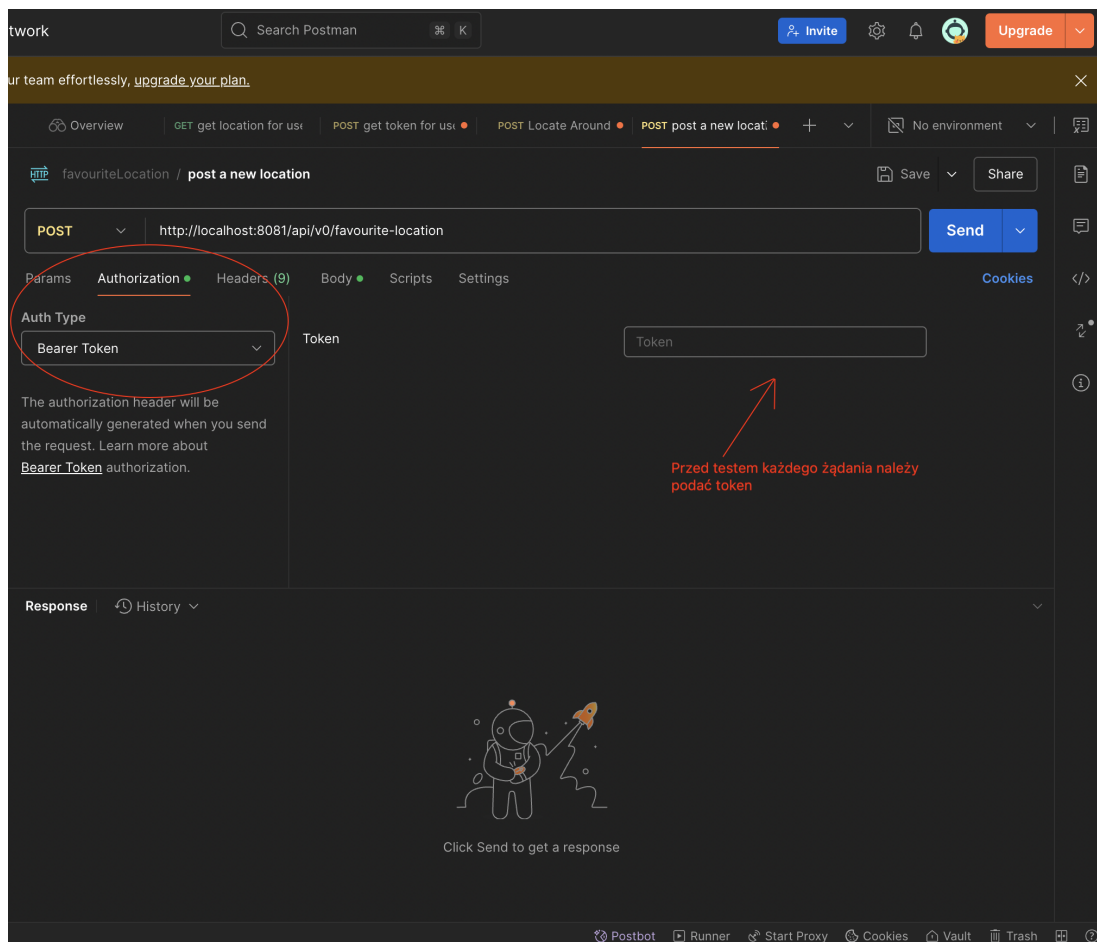
Endpoint pozwala przetestować poprawne działanie funkcjonalności logowania użytkownika. Zwraca token potrzebny do przetestowania pozostałych endpointów.

Parametr	Opis	Typ
grant_type	Typ żądania — zazwyczaj <b>password</b>	String
client_id	Identyfikator klienta (aplikacji)	String
username	Nazwa użytkownika	String
password	Hasło użytkownika	String
client_secret	Tajny klucz klienta	String

Tabela 2: Parametry dla żądania POST /token

Wszystkie opisane endpointy (oprócz uzyskania tokena) wymagają autoryzacji poprzez nagłówek HTTP:

**Authorization:** Bearer <token>



Rysunek 13: testowanie endpointów z użyciem tokena JWT

### POST /api/v0/favourite-location

Endpoint pozwala przetestować poprawne dodanie ulubionej lokalizacji dla użytkownika oraz zapisanie jej w bazie danych. Pozwala również przetestować poprawne pobranie użytkownika z tokena JWT.

Parametr	Opis	Typ
score	Ocena lokalizacji (0–100)	Integer
lat	Szerokość geograficzna	Double
lon	Długość geograficzna	Double
availablePlaces	Lista dostępnych miejsc	Array[String]
notAvailablePlaces	Lista niedostępnych miejsc	Array[String]

Tabela 3: Parametry żądania POST /favourite-location

### GET /api/v0/favourite-location/{id}

Endpoint pozwala na pobranie z bazy danych ulubionej lokalizacji na podstawie jej identyfikatora. Sprawdzono, czy system reaguje odpowiednio, gdy szukana lokalizacja nie istnieje. Endpoint testuje również poprawność działania mechanizmu autoryzacji na podstawie tokena JWT użytkownika.

### PUT /api/v0/favourite-location/{id}

Endpoint służy do testowania funkcjonalności aktualizacji istniejącej lokalizacji na podstawie jej identyfikatora. Pozwala sprawdzić, czy system reaguje odpowiednio na próbę aktualizacji nieistniejącej lokalizacji. Endpoint testuje również poprawność działania mechanizmu autoryzacji na podstawie tokena JWT użytkownika.

Parametr	Opis	Typ
score	Ocena lokalizacji	Float
lat	Szerokość geograficzna	Double
lon	Długość geograficzna	Double
availablePlaces	Lista dostępnych miejsc	Array[String]
notAvailablePlaces	Lista niedostępnych miejsc	Array[String]

Tabela 4: Parametry żądania PUT /favourite-location/{id}

### DELETE /api/v0/favourite-location/{id}

Endpoint pozwala przetestować funkcjonalność usuwania ulubionej lokalizacji z bazy danych na podstawie jej identyfikatora. Pozwala sprawdzić zachowanie systemu w przypadku usunięcia lokalizacji o identyfikatorze, który nie istnieje w bazie danych. Endpoint testuje również poprawność działania mechanizmu autoryzacji na podstawie tokena JWT użytkownika.

### GET /api/v0/favourite-location

Endpoint zwracający ulubione lokalizacje wszystkich użytkowników. Pozwala sprawdzić poprawność działania mechanizmu autoryzacji na podstawie tokena JWT użytkownika.

### GET /api/v0/favourite-location/user

Endpoint pozwalający przetestowanie funkcjonalności pobrania ulubionych lokalizacji użytkownika. Pozwala sprawdzić poprawność działania mechanizmu autoryzacji na podstawie tokena JWT użytkownika

### POST /api/v0/map/locate

Endpoint pozwala na przetestowanie...

Parametr	Opis	Typ
lat	Szerokość geograficzna	Double
lon	Długość geograficzna	Double
radius	opis	typ
types	opis	Array[String]

Tabela 5: Parametry żądania POST /map/locate

## Bibliografia

- [1] *hiszpania*. URL: <https://www.gov.pl/web/hiszpania/zona-bajas-emisiones#:~:text=na%20terenie%20miasta%20Barcelona%20i%20pojazd%C3%B3w%20osobowych%20niespe%C5%82niaj%C4%85cych%20wymog%C3%B3w%20dot..>
- [2] *ibm*. URL: <https://www.ibm.com/think/topics/three-tier-architecture>.
- [3] *medium*. URL: <https://medium.com/@burakkocakeu/jpa-hibernate-and-spring-data-jpa-efa71feb82ac>.

- [4] *spyshop.pl*. URL: <https://www.spyshop.pl/blog/wp-content/uploads/2023/03/pietnastominutowe-miasta-koncepcja-jak-moga-wygladac.png>.
- [5] *timeout.com*. URL: <https://www.timeout.com/barcelona/things-to-do/best-things-to-do-in-barcelona>.