Dokumentacja – Aplikacja do wyszukiwania odpowiedniego miejsca do aktywnego spędzania wakacji lub urlopu

# Wersja

|  |  |  |  |
| --- | --- | --- | --- |
| Wersja | Data | Autor | Zakres zmian |
| 1.0 | 2023-05-12 | Śpiewakowski Kacper | Dokumentacja |

# Spis treści

[Wersja 1](#_Toc135478665)

[Spis treści 1](#_Toc135478666)

[1. Wprowadzenie 3](#_Toc135478667)

[1.1 Cel 3](#_Toc135478668)

[Celem niniejszego dokumentu jest zidentyfikowanie koniecznych wymagań funkcjonalnych i niefunkcjonalnych dla aplikacji. Celem aplikacji jest znalezienie odpowiedniego miasta przez użytkownika w aplikacji, a następnie aby ten użytkownik mógł znaleźć dla siebie odpowiednie miejsca. Aplikacja może działać jako mini przewodnik po danym mieście. Ten dokument jest przeznaczony dla różnych interesariuszy, w tym dla programistów, kierowników projektów, użytkowników, testerów. Dzięki niemu wszyscy ci użytkownicy będą mieli wspólne zrozumienie wymagań funkcjonalnych i niefunkcjonalnych dotyczących . 3](#_Toc135478669)

[1.2 Przyjęte konwencje 3](#_Toc135478670)

[1.3 Zakres projektu 4](#_Toc135478671)

[Zakres projektu obejmuje opracowanie i wdrożenie internetowego systemu wyszukiwania idealnych miejsc na wakacje lub urlop oraz planów dostosowanych do kryteriów każdego użytkownika z osobna, wykorzystując Django jako framework webowy oraz bazę danych Django + SQLite. 4](#_Toc135478672)

[1.4 Odwołania 4](#_Toc135478673)

[2. Opis ogólny 4](#_Toc135478674)

[2.1 Perspektywa produktu 4](#_Toc135478675)

[2.2 Klasy i charakterystyki użytkowników 4](#_Toc135478676)

[2.3 Środowisko działania 4](#_Toc135478677)

[2.4 Ograniczenia projektowe i wykonawcze 5](#_Toc135478678)

[2.5 Założenia i zależności 5](#_Toc135478679)

[3. Funkcjonalności systemu 5](#_Toc135478680)

[3.1 Wyszukiwanie miejsc 5](#_Toc135478681)

[3.1.1 Opis 5](#_Toc135478682)

[3.1.2 Wymagania funkcjonalne 5](#_Toc135478683)

[3.2 Mapa 5](#_Toc135478684)

[3.2.1 Opis 5](#_Toc135478685)

[3.2.2 Wymagania funkcjonalne 6](#_Toc135478686)

[3.3 Logowanie 6](#_Toc135478687)

[3.3.1 Opis 6](#_Toc135478688)

[3.3.2 Wymagania funkcjonalne 6](#_Toc135478689)

[3.4 Rejestracja 6](#_Toc135478690)

[3.4.1 Opis 6](#_Toc135478691)

[3.4.2 Wymagania funkcjonalne 6](#_Toc135478692)

[4. Wymagania dotyczące danych 6](#_Toc135478693)

[4.1 Logiczny model danych 6](#_Toc135478694)

[4.2 Raporty 6](#_Toc135478695)

[4.3 Pozyskiwanie, integralność przechowywanie I usuwanie danych 6](#_Toc135478696)

[5. Wymagania dotyczące interfejsu zewnętrznego 7](#_Toc135478697)

[5.1 Interfejsy użytkownika 7](#_Toc135478698)

[5.2 Interfejsy programowe 7](#_Toc135478699)

[5.3 Interfejsy sprzętowe 7](#_Toc135478700)

[5.4 Interfejsy komunikacyjne 7](#_Toc135478701)

[6. Cechy jakości 7](#_Toc135478702)

[6.1 Użyteczność 7](#_Toc135478703)

[Wymagania dotyczące użyteczności mają na celu zapewnienie optymalnego doświadczenia użytkownika poprzez łatwość użytkowania, wydajność interakcji, bezpieczeństwa i dostępności systemu. Aplikacja : 7](#_Toc135478704)

[6.2 Wydajność 7](#_Toc135478705)

[6.3 Zabezpieczenia 8](#_Toc135478706)

[6.4 Bezpieczeństwo użytkowania 8](#_Toc135478707)

[7. Wymagania internacjonalizacji i lokalizacji 8](#_Toc135478708)

[8. Wdrożenie systemu 8](#_Toc135478709)

[8.1 Diagram wdrożenia 8](#_Toc135478710)

[8.2 Wdrożenie 8](#_Toc135478711)

[9. Proces wytworzenia produktu 8](#_Toc135478712)

[9.1 Wykonane zadania 8](#_Toc135478713)

[9.2 Zmiany wykonane w czasie 8](#_Toc135478714)

[9.3 Indywidualny wkład członków zespołu 8](#_Toc135478715)

[9.4 Dostarczone produkty 9](#_Toc135478716)

[9.5 Środowisko pracy zespołu 9](#_Toc135478717)

[9.6 Napotkane problemy 9](#_Toc135478718)

[9.7 Potencjalne możliwości produktu 9](#_Toc135478719)

[9.8 Wykonane zadania 9](#_Toc135478720)

[10. Przewodnik po oprogramowaniu 9](#_Toc135478721)

# 1. Wprowadzenie

Aplikacja internetowa służy do wyszukiwania odpowiedniego miejsca do aktywnego spędzania wakacji lub urlopu. System ma na celu ułatwienie zarejestrowanym użytkownikom znalezienia odpowiednich miejsc takich jak ( hotele, restauracje itp. ) , które pozwolą im na spędzenie wolnego czasu. Projekt zakłada, że aplikacja będzie dostępna na komputerze. Dokumentacja składa się z wielu sekcji, w którym szczegółowo zostały opisane wymagania funkcjonalne jak i niefunkcjonalne. Przedstawiona zostanie struktura systemu, projekt interfejsu, testy użytkowe. Wszystkie działania zostaną szczegółowo udokumentowane.

## 1.1 Cel

## Celem niniejszego dokumentu jest zidentyfikowanie koniecznych wymagań funkcjonalnych i niefunkcjonalnych dla aplikacji. Celem aplikacji jest znalezienie odpowiedniego miasta przez użytkownika w aplikacji, a następnie aby ten użytkownik mógł znaleźć dla siebie odpowiednie miejsca. Aplikacja może działać jako mini przewodnik po danym mieście. Ten dokument jest przeznaczony dla różnych interesariuszy, w tym dla programistów, kierowników projektów, użytkowników, testerów. Dzięki niemu wszyscy ci użytkownicy będą mieli wspólne zrozumienie wymagań funkcjonalnych i niefunkcjonalnych dotyczących .

## 1.2 Przyjęte konwencje

Nazwy zmiennych zostały zapisywane z wykorzystaniem notacji camelCase. Wersje frameworków odpowiednio:

* Django dla python w wersji 4.0.0
* Next.js w wersji 13
* NextAuth.js w wersji 1.0.6
* CSS Tailwind w wersji 3.3.2
* tRCP
* Prisma

Projekt Interfejsu – projekt strony głównej: wykorzystanie ikon ( fun, history, hotels, night\_life, recreations, restourant), strony wejściowa, logowanie/ rejestracja, wyszukiwarka, strona docelowa z listą w.w wraz z mapą przedstawiającą, gdzie dany obiekt się znajduję. Projekt przewiduję skalę ocen gwiazdkową od 1 do 5., strona 404.

Sekcje : header, Map, QueryButtons, Section, skeleton, Toast .

Architektura systemu – baza danych Python / Django + SQLite, visual studio code, API opentripmap.

Testowanie, plany testowe – Użytkownik otrzyma przewodnik po oprogramowaniu.

Biblioteki używane w projekcie będą dynamicznie dodawane i niekoniecznie muszą być one dostępne w standardowej dystrybucji wykorzystywanych języków programowania. Wszystkie nazwy plików powinny być zapisane wykorzystując tylko i wyłącznie małe litery z ewentualnym dodatkiem liczb.

# 1.3 Zakres projektu

## Zakres projektu obejmuje opracowanie i wdrożenie internetowego systemu wyszukiwania idealnych miejsc na wakacje lub urlop oraz planów dostosowanych do kryteriów każdego użytkownika z osobna, wykorzystując Django jako framework webowy oraz bazę danych Django + SQLite.

Celem biznesowym projektu jest dostarczanie klientowi usługi / aplikacji, dzięki której w sposób płynny znajdzie dla siebie w danym mieście hotel, restauracje, klub, obiekt historyczny czy też obiekt rekreacyjny. Kolejnym celem jest stworzenie nowej marki / aplikacji internetowej. Jest to zupełnie nowy produkt.

## 1.4 Odwołania

(Odwołania do plików, wykresów na później.)

Inspiracja różnymi portalami dot. szeroko rozumianej turystyki takimi jak:

* TripAdvisor
* Booking
* Travelplanet
* Trivago
* wakacje.pl

# 2. Opis ogólny

## 2.1 Perspektywa produktu

Oferowany produkt jest zupełnie nowy i zostanie stworzony od podstaw na wzorze z podobnych zaobserwowanych rozwiązań. Aplikacja będzie działać jako wyszukiwarka, która pozwoli użytkownikom na wyszukanie interesujących ich miejsc w danym mieście w oparciu o wybrane kryteria, takie jak hotele, restauracje, obiekty historyczne itp.

Wszystkie interakcje z aplikacją internetową będą odbywać się przez poprzez połączenie http://localhost:3000, a dane użytkowników będą przechowywane w sposób zgodny z obowiązującymi przepisami dotyczącymi prywatności i ochrony danych osobowych. Aplikacja działa na wszystkich systemach operacyjnych, przeznaczona do użytku na komputerach.

## 2.2 Klasy i charakterystyki użytkowników

Użytkowników w aplikacji możemy podzielić na 2 klasy. Standardowi użytkownicy / turyści, którzy szukają informacji o dowolnym mieście czy też szukają w nim dowolnego obiektu, który ich zainteresuję. Kolejną klasą użytkowników są turyści grupowi, którzy na podstawie ocen danych miejsc, wybiorą dla nich interesujące obiekty, hotele czy też restauracje i na podstawie tych danych będą mogli planować trasę wycieczki.

Ostatnią klasą są administratorzy – są to osoby zarządzające całym systemem. Posiadają niezbędną wiedzę techniczną do obsługi aplikacji.

## 2.3 Środowisko działania

Środowisko – Przeglądarka internetowa PL, EN

Frontend – HTML, CSS/SCSS, JS

Backend – Python

Baza danych – SQLite

SVN – Github

Korzystanie z Visual Studio Code.

## 2.4 Ograniczenia projektowe i wykonawcze

W ramach projektu aplikacji istnieją ograniczenia projektowe i wykonawcze.

Ograniczeniem projektowym w wyżej wymienionej aplikacji jest wykorzystanie frameworku webowego Django oraz bazy danych Django+SQLite. Kolejnym ograniczeniem jest framework do backendu jakim jest Next.js . Inne ograniczenia to użycie NextAuth.js , CSS Tailwind. Kolejnym ograniczeniem jest wykorzystanie API Opentripmap do lokalizowania miejsc i obiektów na mapie.

Ograniczeniem wykonawczym w dużej mierze są zasoby finansowe, kadrowe oraz ograniczenia czasowe. Projekt ograniczony czasowo z pewnością wpłynie na ilość zrealizowanych funkcji opcjonalnych.

## 2.5 Założenia i zależności

Wymagane założenia dotyczące funkcjonowania aplikacji to :

* Umożliwianie użytkownikom wyszukiwania miejsc i obiektów na mapie dowolnego miasta z bazy danych
* Umożliwienie użytkownikom wybrania dowolnego obiektu na mapie ( sortowanie od najlepszych do najgorszych \*)
* Aplikacja musi korzystać z API Opentripmap do pobierania danych o miejscach i obiektach na mapie
* Intuicyjny interfejs aplikacji
* Baza danych z zapisanymi miastami i obiektami znajdujących się w niej

Wymagane zależności:

* Aplikacja będzie działać w oparciu o framework Django i wymagać będzie uruchomienia serwera Django
* Aplikacja będzie korzystać z bazy danych Django + SQLite
* Aplikacja będzie korzystać z API Opentripmap
* Aplikacja będzie wymagać połączenia z Internetem

# 3. Funkcjonalności systemu

## 3.1 Wyszukiwanie miejsc

### 3.1.1 Opis

Wyszukiwanie miejsc i obiektów na mapie dowolnego miasta: Użytkownicy będą mogli wyszukiwać interesujące ich miejsca i obiekty na mapie, takie jak restauracje, hotele, atrakcje turystyczne itp.

### 3.1.2 Wymagania funkcjonalne

System musi umożliwiać wpisanie nazw miast bez znaczenia czy użytkownik będzie pisał wielką literą czy też nie.

System musi być połączony z bazą danych w celu przechowywania i pobierania danych dotyczących miejsc i obiektów na mapie.

System musi wyświetlać informację, aby wpisać miasto.

System musi sprawdzać czy dane miasto znajduję się w bazie danych.

System musi wyświetlić informację w przypadku niepoprawnego wpisania miasta.

System po wpisaniu pierwszych 3 liter musi podpowiedzieć jakie miasta są do wyboru.

Po odnalezieniu danego miasta system musi przejść do sekcji „mapa”.

## 3.2 Mapa

### 3.2.1 Opis

Funkcja będzie wyświetlać wybrane miasto na mapie + obiekty znajdujące się na niej. Wyświetlane obiekty będą miały oceny ( gwiazdki ) od 1 do . Na mapie będą znajdować się różne lokacje warte zobaczenia lub odwiedzenia (zabytki, kawiarnie,

muzea, itd...).

### 3.2.2 Wymagania funkcjonalne

System powinien umożliwiać wyświetlanie wybranego miasta na mapie.

System musi wyświetlać obszar z wykorzystaniem API Opentripmap.

Obiekty na mapie muszą mieć znaczniki.

Każdy obiekt musi mieć zadeklarowaną ocenę ( gwiazdki ).

## 3.3 Logowanie

### 3.3.1 Opis

Będzie zapewniał funkcjonalność logowania użytkowników, w tym weryfikację adresu email.

### 3.3.2 Wymagania funkcjonalne

Użytkownik będzie musiał się zarejestrować.

Do logowania będzie potrzebny adres e-mail oraz hasło.

Hasło musi być zabezpieczone korzystając z biblioteki bcrypt.

Każdy adres email musi być sprawdzany czy istnieje w bazie.

## 3.4 Rejestracja

### 3.4.1 Opis

Użytkownik będzie miał możliwość zarejestrowania się swoim emailem, loginem oraz będzie mógł ustawić hasło.

### 3.4.2 Wymagania funkcjonalne

Użytkownik ma możliwość wpisania dowolnego hasła.

Użytkownik ma możliwość podania dowolnego email.

Hasło musi być zabezpieczone korzystając z biblioteki bcrypt.

Musi być walidacja adresu e-mail.

Użytkownik może mieć możliwość dodania zdjęcia profilowego.

# 4. Wymagania dotyczące danych

## 4.1 Logiczny model danych



## 4.2 Raporty

System będzie generował dla użytkownika dane, które będzie mógł wykorzystać w dalszym planowaniu trasy/ podróży. Użytkownik otrzyma lista danych obiektów, posortowanych od 1 do 5 gwiazdek.

## 4.3 Pozyskiwanie, integralność przechowywanie I usuwanie danych

Pozyskiwanie danych:

* Pozyskiwanie adresu email użytkowników
* Pozyskanie zdjęcia / logo użytkownika
* Pozyskanie danych o obiektach w danym mieście
* Pozyskanie listy miast z API Opentripmap
* Szyfrowanie hasła przy użyciu algorytmu hashowania bcrypt

Integralność danych:

* System będzie sprawdzał poprawność wpisywania danych
* Walidacja adresu email
* Przechowywanie danych w bazie Django+SQLite
* Zabezpieczenie danych

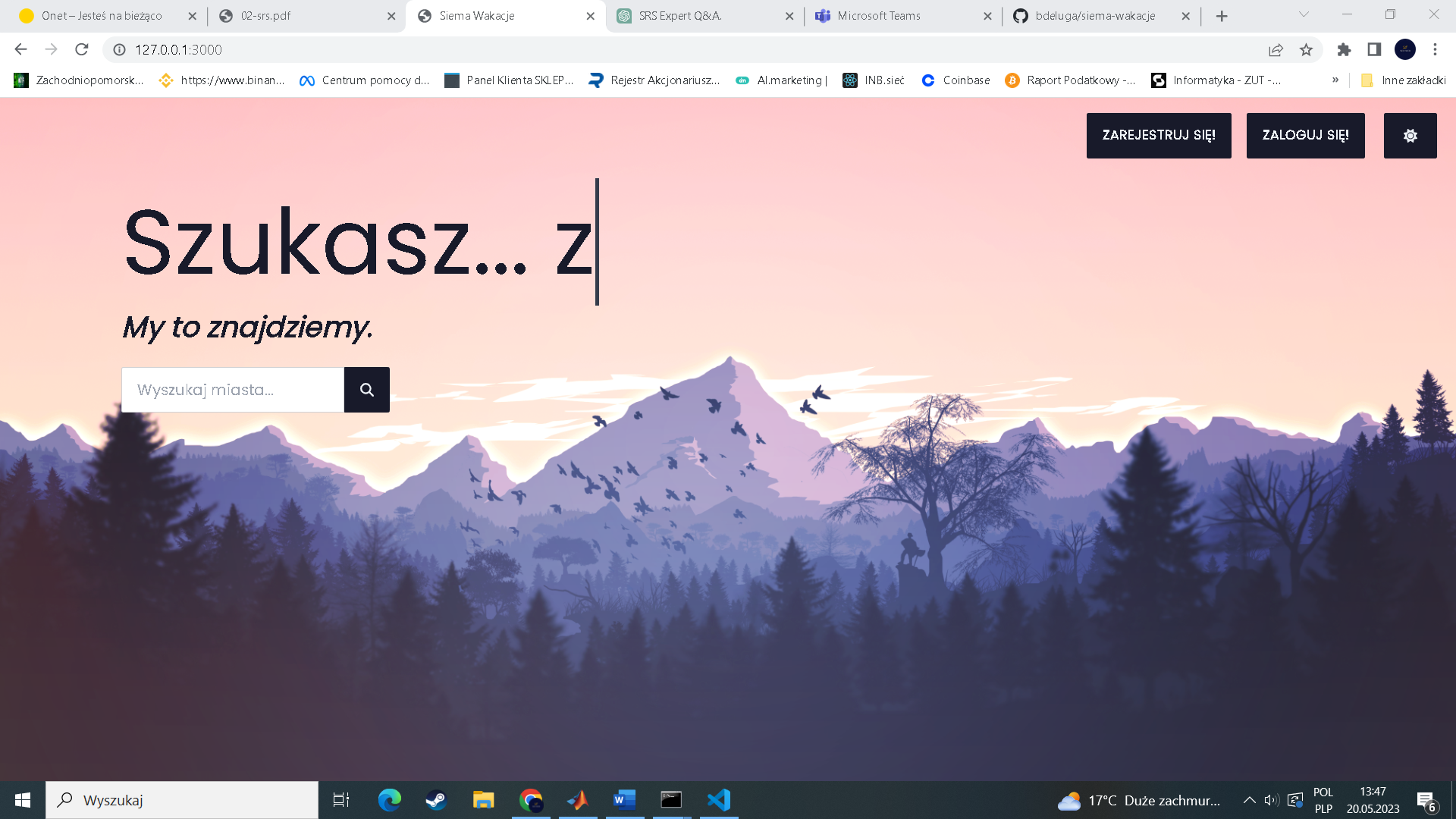
Usuwanie danych:

* Użytkownik ma możliwość usunięcia logo / zdjęcia
* Użytkownik ma możliwość usunięcia konta

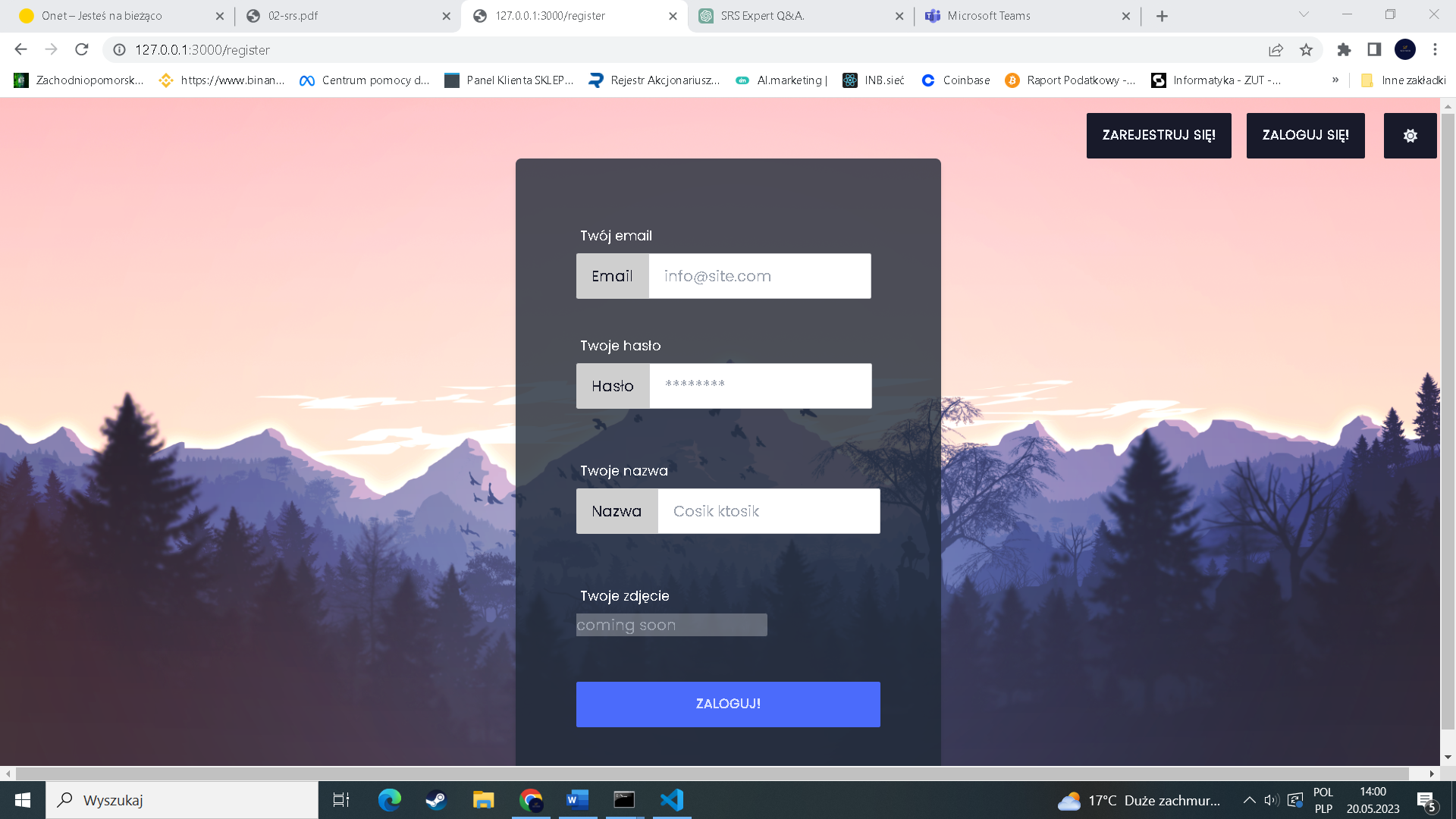
# 5. Wymagania dotyczące interfejsu zewnętrznego

## 5.1 Interfejsy użytkownika

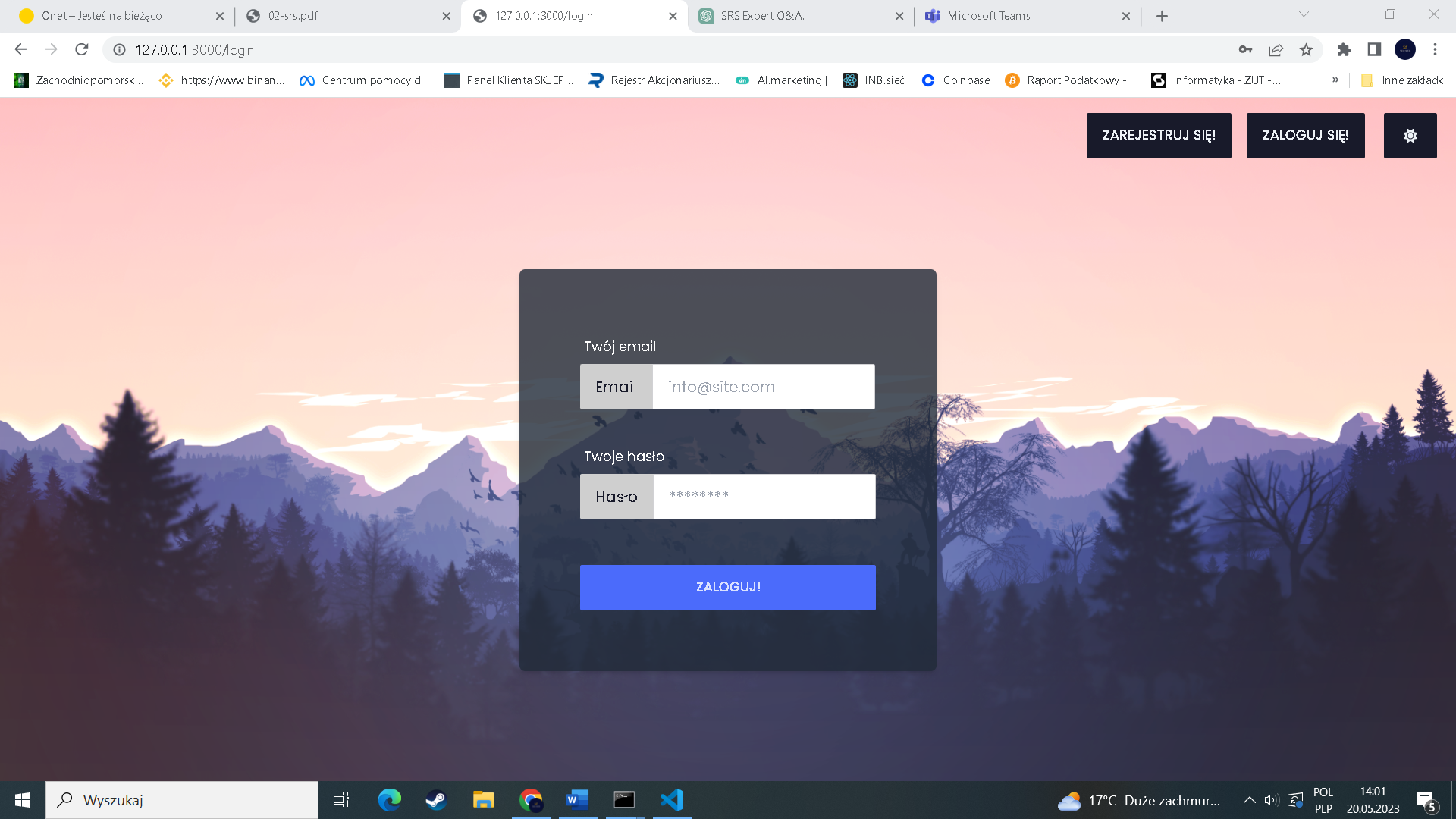
* Czcionka czytelna i dopasowana do ekranu tak, aby była zrozumiana przez użytkwonika
* Ikony jasne i łatwe do zrozumienia
* Zawartość pól będzie sprawdzana przed wysłaniem formularza i poinformowaniem użytkownika o jakichkolwiek błędach. Sprawdzenie będzie pole zarówno „wyszukaj miasta…” jak i dane podawane przy rejestracji czy też logowaniu
* Ikona w prawym górnym rogu służy do zmiany wyświetlania strony głównej ( kolor jasny / ciemny )



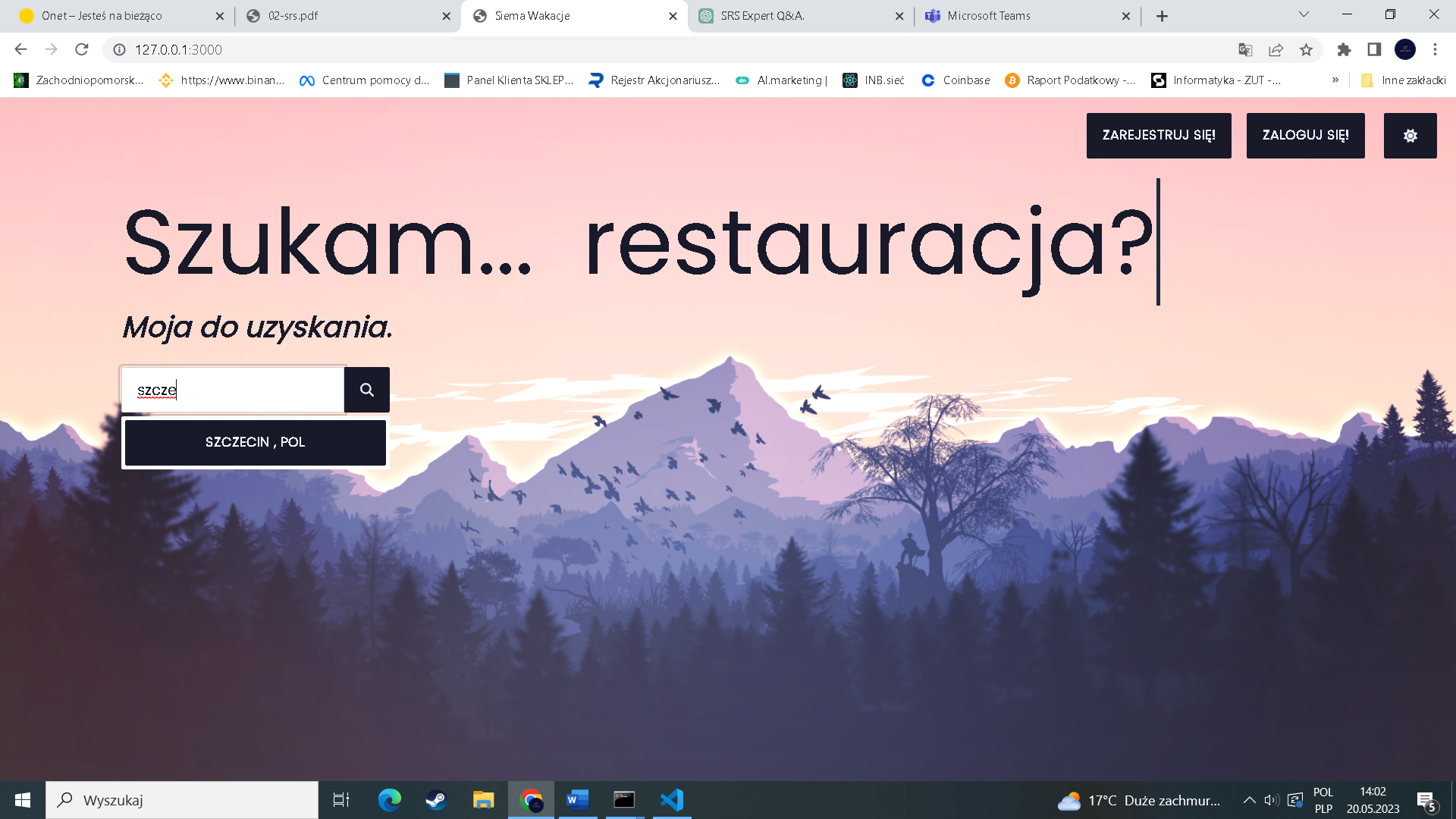
* Przyciski na stronie : „Zarejestruj się!”, „Zaloguj się!”, ikona do zmiany koloru tła na stronie, ikona ( lupa ) do wyszukiwania miasta + obiektów w nim znajdujących się.
* „Zarejestruj się!” –



* „Zaloguj się!” –



* Animacja nad frazą „ My to znajdziemy” składa się z następujących wyrażeń : "hotelu?", "restauracji?", "trasy rowerowej?", "zabytków w okolicy?", "klubu?", "obiektu sportowego?", "parku?". Border color.
* Po wpisaniu w pole wyszukiwarki co najmniej 3 liter pojawiają się podpowiedzi najbardziej możliwych miast.



## 5.2 Interfejsy programowe

Interfejsy programowe są kluczowymi elementami integracji pomiędzy różnymi składnikami oprogramowania aplikacji. Dlatego:

* Wspieranie standardowych protokołów takich jak http
* Korzystanie z API Opentripmap do pobierania nazw miast + obiektów znajdujących się w danym mieście
* Korzystanie z bazy danych
* Wsparcie dla formatów CSV, JSON w celu ewentualnego importu danych

## 5.3 Interfejsy sprzętowe

Aplikacja będzie kompatybilna ze wszystkimi dostępnymi przeglądarkami. Na każdej przeglądarce aplikacja będzie działała tak samo oraz będzie intuicyjna i przejrzysta. Główny cel to komfort obsługi przez danego użytkownika. Docelowo aplikacja będzie postawiona na serwerze, aby użytkownik, który ma dostęp do Internetu mógł swobodnie ze strony korzystać. Dane wejściowe będą pobierane od użytkownika, gdy ten naciśnie klawisz na klawiaturze. Jeśli danego miasta nie będzie w bazie danych, wówczas dostanie jasny komunikat o braku znalezienia i konkretnego miasta. Ewentualne problemy mogą nastąpić w przypadku problemów z internatem, braku synchronizacji z bazą danych.

## 5.4 Interfejsy komunikacyjne

Dane użytkowników takie jak hasła będą szyfrowane, dzięki czemu aplikacja może zapewnić bezpieczeństwo użytkowników.

# 6. Cechy jakości

## 6.1 Użyteczność

## Wymagania dotyczące użyteczności mają na celu zapewnienie optymalnego doświadczenia użytkownika poprzez łatwość użytkowania, wydajność interakcji, bezpieczeństwa i dostępności systemu. Aplikacja :

* Przejrzysty i intuicyjny interfejs
* Działanie na różnych przeglądarkach
* Wyszukanie dowolnego miasta
* Wyszukanie interesującego nas obiektu w danym mieście
* Szybkie wyszukanie informacji potrzebnych do planowania trasy czy też wycieczki przez użytkownika
* Zapisanie swojego profilu – rejestracja i logowanie użytkownika

## 6.2 Wydajność

Wydajność aplikacji zależy od wielu czynników. W przypadku różnych operacji wydajność może być różna. Działania:

* Wyszukiwanie interesujących obiektów w danym mieście w czasie rzeczywistym
* Walidacja adresu email
* Autoryzacja użytkowników w systemie

Przetwarzanie tych danych nie powinno powodować problemów z wydajnością ze względu na korzystania z lokalnej bazy danych SQLite, szybkich ściągnięciach danych przez korzystając z API.

## 6.3 Zabezpieczenia

Bezpieczeństwo danych:

* Autoryzacja i uwierzytelnianie użytkowników
* Rejestracja nowych osób w systemie
* Szyfrowanie hasła przy użyciu algorytmu hashowania

Bezpieczeństwo oprogramowania:

* Filtrowanie danych wejściowych
* Walidacja adresów email
* Wyświetlanie ostrzeżeń i komunikatów

Prywatność użytkowników:

* Gromadzenie danych użytkowników takich jak login, email, hasło, zdjęcie profilowe
* Możliwość edycji / usuwania danych
* Przetwarzanie danych użytkowników

## 6.4 Bezpieczeństwo użytkowania

Aby zapewnić jak największy poziom bezpieczeństwa aplikacji to system powinien:

* Wyświetlać jasne i zrozumiałe komunikaty
* Informować o błędach działania aplikacji
* System powinien być intuicyjny, aby potencjalny użytkownik nie miał problemu z poruszaniem się po aplikacji
* System rejestracji / logowania powinien być zabezpieczony

# 7. Wymagania internacjonalizacji i lokalizacji

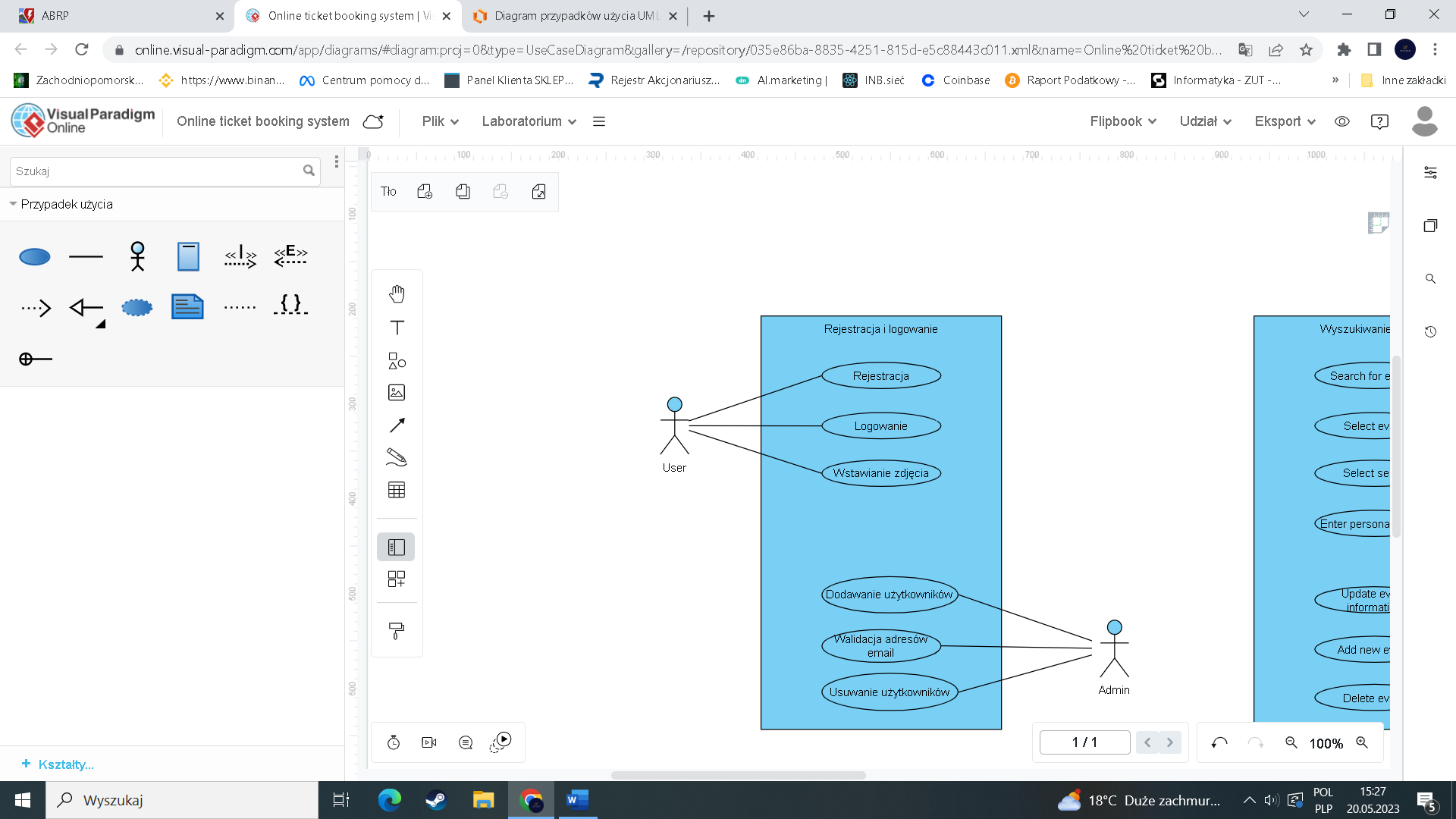
Aplikacja będzie działać w języku polskim. Dane użytkownika nie będą bezpośrednio pobierać lokalizacji, wszystkie dane będą zapisywane do bazy danych.

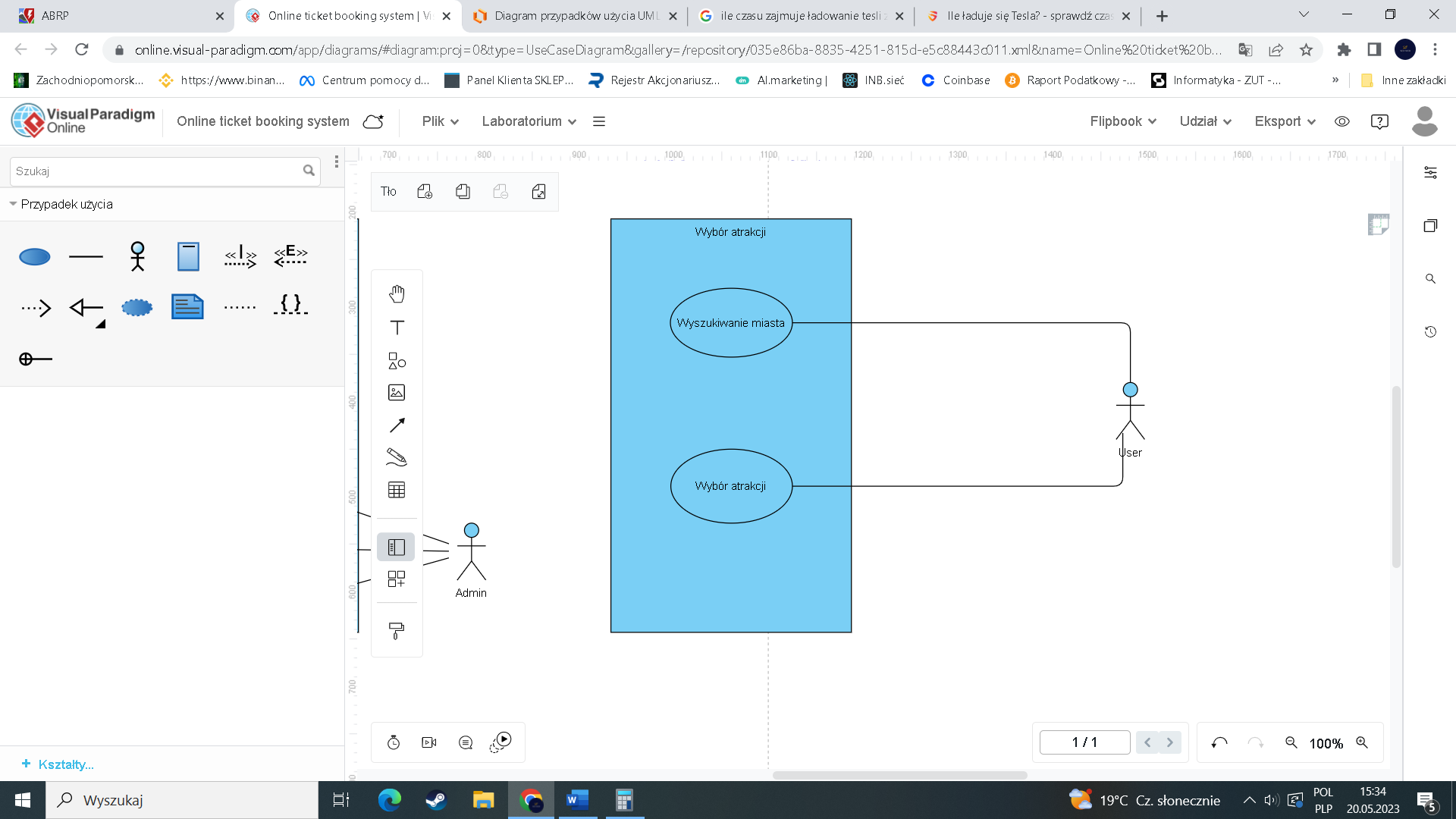
# 8. Wdrożenie systemu

## 8.1 Diagramy

Diagram klas 4.1

Diagram przypadków użycia –





## 8.2 Wdrożenie

[ Diagram wdrożeniowy ]

W pierwszej kolejności należy pobrać katalog i wstawić w dowolne miejsce na komputerze. Następnie przejść do katalogu np. poprzez CMD i przejść do „Frontend”. Należy ustawić odpowiedni .env ustawiając NEXTAUTH\_URL=http://localhost:3000

NEXTAUTH\_SECRET=sdsdsndjsdjsds

DISCORD\_CLIENT\_ID=""

DISCORD\_CLIENT\_SECRET="" . Następnie należy wpisać w terminal kolejno npm i npm run dev.

Proces uruchomienia Frontendu zakończony.

Proces uruchamiania Backendu:

* Przejść do katalogu backend -> siema\_wakacje\_b -> siema\_wakacje\_b
* Utworzyć i skonfigurować plik .env
* DEBUG=True
* SECRET\_KEY=SECRET\_KEY
* TRIP\_KEY=5ae2e3f221c38a28845f05b6b308fc2c3b0012fd089f723250a0d9d9s
* NEXTAUTH\_URL=http://localhost:8000
* NEXTAUTH\_SECRET=sdsdsndjsdjsds
* Zapisać i cofnąć się do siema\_wakacje\_b
* Następnie wpisać szereg komend python
* -m pip install django~=4.0.0 (if is not installed)
* pip install django-cors-headers
* pip install django-environ
* pip install osmnx
* pip install pandas
* pip install bcrypt
* python manage.py runserver

Backend został uruchomiony. Aby przejść do aplikacji należy wpisać http://127.0.0.1:3000/.

# 9. Proces wytworzenia produktu

## 9.1 Wykonane zadania

[ Tabela / Wykres ]

## 9.2 Zmiany wykonane w czasie

Zmiany obejmują różne aspekty takie jak zmiany funkcjonalności, interfejsie użytkownika, bazie danych, użytych narzędziach, środowisko pracy, naprawa błędów. W naszej aplikacji wszystkie te aspekty miały miejsce. Dokonano:

* logowanie – możliwość dodania zdjęcia do profilu
* baza danych, pobieranie z API dostępnych obiektów na konkretne miasto
* wprowadzenie nowych zmian w interfejsie
* naprawa błędów wynikają z mapy, filtrowania danych
* zmiana zadań członków zespołu w trakcie trwania projektu

## 9.3 Indywidualny wkład członków zespołu

Bartosz Deluga – 30 %

Kacper Śpiewakowski – x %

Maciej Sanocki – x %

Kacper Wereszczyński – x %

Pozyskane umiejętności:

* [ ]
* [ ]

Przebieg współpracy:

* [ ]
* [ ]

## 9.4 Napotkane problemy

[ Baza danych ] , [ Mapa / filtrowanie ], [ Mapa - detale ]

## 9.5 Potencjalne możliwości produktu

Potencjalne możliwości produktu odnoszą się do perspektyw rozwoju i rozszerzenia aplikacji w przyszłości. Dodatkowe funkcjonalności takie jak:

* nowe integracje
* możliwość rezerwacji danych obiektów
* dokładna ścieżka planowania wycieczki trasa

Usprawnienie wydajności oraz refaktoryzacja kodu, redukcja zasobów. Inną możliwością jest rozszerzenie liczby użytkowników, osobne funkcje dla osób zalogowanych i nie zarejestrowanych.

## 9.6 Wykonane zadania

Opisane w punkcie 9.3 .

# 10. Przewodnik po oprogramowaniu

Opisanie w punkcie 5.1 .