Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH

Instytut Automatyki i Informatyki Stosowanej

Praca dyplomowa inżynierska

na kierunku Informatyka
w specjalności Systemy Informacyjno-Decyzyjne

Aplikacja webowa do wynajmu obiektów sportowych

Dominik Maciej Trusiński
Numer albumu 277349

promotor
dr inż. Mariusz Kamola

Warszawa 2019

Aplikacja webowa do wynajmu obiektów sportowych

Streszczenie

Celem pracy było stworzenie aplikacji webowej umożliwiającej tworzenie, dołączanie i płatność za wydarzenia sportowe. W pracy przedstawiono znaczenie sportu w życiu człowieka, pokazano jaką wartość dodaną prezentuje aplikacja oraz przybliżono istniejące rozwiązania. Przedstawiono aktorów występujących w opracowanej aplikacji, model dziedziny oraz przypadki użycia. Opisano wykorzystane technologie i narzędzia. Zwrócono uwagę na problemy jakie wynikły w czasie tworzenia aplikacji. Zamieszczono możliwości rozwoju oraz wnioski.

Słowa kluczowe: sport, JavaScript, Elixir, Angular, Facebook, Paypal, Google Maps

Web application for renting sport objects

Abstract

The main goal of this thesis was to create web application that allows for creation, participation and paying for sport events. Thesis includes chapters about significance of sport in human's life, added value of this thesis and other currently available solutions. Following aspects were described: actors in the application, domain model and use cases. Technologies and tools used in the application were presented. Problems which occurred during development of the application were pointed out. Last chapter includes possibilities for further development and conclusions.

Keywords: sport, JavaScript, Elixir, Angular, Facebook, Paypal, Google Maps

załącznik do zarządzenia nr 28/2016 r.
Rektora PW

Politechnika Warszawska

„załącznik nr 3 do zarządzenia nr 24/2016 Rektora PW

.....
miejscowość i data

.....
imię i nazwisko studenta

.....
numer albumu

.....
kierunek studiów

OŚWIADCZENIE

Świadomy/-a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie, pod opieką kierującego pracą dyplomową.

Jednocześnie oświadczam, że:

- niniejsza praca dyplomowa nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym,
- niniejsza praca dyplomowa nie zawiera danych i informacji, które uzyskałem/-am w sposób niedozwolony,
- niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów lub tytułów zawodowych,
- wszystkie informacje umieszczone w niniejszej pracy, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami,
- znam regulacje prawne Politechniki Warszawskiej w sprawie zarządzania prawami autorskimi i prawami pokrewnymi, prawami własności przemysłowej oraz zasadami komercjalizacji.

Oświadczam, że treść pracy dyplomowej w wersji drukowanej, treść pracy dyplomowej zawartej na nośniku elektronicznym (płycie kompaktowej) oraz treść pracy dyplomowej w module APD systemu USOS są identyczne.

.....
czytelny podpis studenta"

Spis treści

1 Wstęp

1.1 Uprawianie sportu

Uprawianie sportu powinno być ważną czynnością w życiu każdego człowieka. Odpowiednia ilość aktywności fizycznej przyczynia się do poprawy samopoczucia oraz utrzymania dobrej kondycji całego organizmu. W zależności od sportu, jego uprawianie może wiązać się z koniecznością znalezienia odpowiedniej lokalizacji (np. boiska piłkarskiego) czy też zorganizowania odpowiedniej liczby osób (np. 22 osób do meczu piłki nożnej). Może to być trudne ze względu na dużą popularność danego obiektu lub zbyt małą ilość chętnych osób w otoczeniu danego człowieka.

1.2 Cel

Osoby aktywne fizycznie mogą być zainteresowane znalezieniem takiego obiektu sportowego, który umożliwi im trenowanie danej dziedziny sportu z odpowiednią ilością współgraczy. Niektóre obiekty sportowe umożliwiają telefoniczne bądź mailowe rezerwacje (tak jest w przypadku większości warszawskich orlików)[1]. Niestety, wiążę się to z koniecznością samodzielnego kontaktowania się z właścicielami tych obiektów w celu sprawdzenie czy możliwa jest rezerwacja.

Głównym celem tej pracy było napisanie aplikacji, która umożliwi dodawanie nowych obiektów sportowych (przez właścicieli tych obiektów) oraz wyszukiwanie obiektów, tworzenie rezerwacji i płatność za nie (przez sportowców).

Z punku widzenia sportowców aplikacji ta pozwala osiągnąć następujące cele:

- Oszczędność czasu
  Sportowcy nie są zmuszeni do samodzielnego zbierania informacji o dostępności obiektów.
- Możliwość uprawiania egzotycznych sportów
  Egzotyczne sporty (np. futbol amerykański w Polsce) wymagają odpowiednich obiektów. Ta aplikacji umożliwi ich znalezienie.
- Socjalizacja
  Możliwość poznania osób o podobnych zainteresowaniach.
- Równy podział cen
  Każdy uczestnik wydarzenia płaci taką samą składkę.

Z punktu widzenia właścicieli obiektów sportowych aplikacja umożliwi zapełnienie luk pomiędzy już odbywającymi się na obiekcie wydarzeniami.

1.3 Istniejące rozwiązania

Istniejące rozwiązania można podzielić na dwie kategorie:

- Rozwiązania nieformalne
- Rozwiązania formalne

1.3.1 Rozwiązania nieformalne

Rozwiązania nieformalne to aplikacje i serwisy, które umożliwiają tworzenie grup użytkowników wymieniających się wiadomościami. Taka grupa skupia osoby zainteresowanie danym sportem. Osoby te planują czas i miejsce spotkania sportowego. W zależności od stopnia zorganizowania grupy następnym krokiem może być:

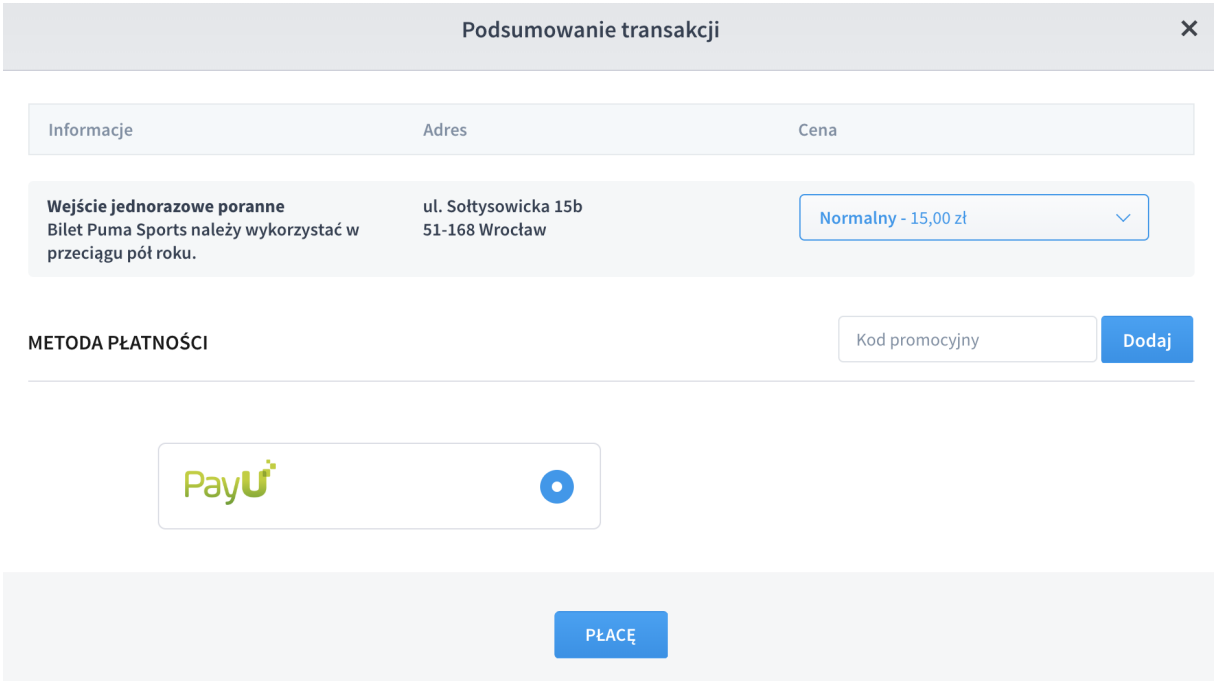
- Skontaktowanie się z właścicielem obiektu w celu wykonania rezerwacji i jej opłacenia
- Stawienie się członków grupy w danym obiekcie o danym czasie z nadzieją, że będzie on dostępny

Jeśli dany obiekt sportowy jest obiektem darmowym (jak w przypadku warszawskich orlików) wtedy wariant drugi nie wiąże się z żadnym nakładem finansowym. Jednakże - jeśli obiekt sportowy będzie w danym czasie zajęty wtedy grupa sportowców może albo dołączyć do już obecnych albo zrezygnować z aktywności fizycznej na tym obiekcie.

Przykładowym narzędziem, które należy do tej kategorii są grupy na Facebooku[2]. Rozwiązania nieformalne nie pozwalają na wyszukiwanie obiektów sportowych – sportowcy muszą to zrobić samodzielnie.

1.3.2 Rozwiązania formalne

Rozwiązania formalne angażują właściciela obiektu sportowego. Grupa sportowców dokonuje rezerwacji poprzez aplikację. Aby potwierdzić rezerwację należy dokonać jej opłaty na konto właściciela obiektu sportowego. Rozwiązania formalne umożliwiają wyszukiwanie obiektów sportowych lub chociaż przegląd dostępnych obiektów bez konieczności opuszczania aplikacji. Przykładem aplikacji formalnej jest `https://rezerwujsport.pl` Umożliwia ona płatność z wykorzystaniem serwisu `https://www.payu.pl/` - zostało to przedstawione na rysunku 1.



Rysunek 1: Podsumowanie transakcji w `https://rezerwujsport.pl` - widoczna możliwość płatności przy użyciu PayU. Źródło: `https://rezerwujsport.pl/`

2 Koncepcja

Praca została zrealizowana zgodnie z modelem kaskadowym. Dlatego też w tym rozdziale poruszone zostaną aspekty dotyczące analizy oraz projektowania. Stanowią one fundament dobrego projektu programistycznego.

2.1 Aktorzy

Aktor to człowiek bądź system, który wchodzi w interakcje ze stworzonym systemem. Zidentyfikowanie wszystkich aktorów stanowiło ważny krok w rozpoznaniu wszystkich przypadków użycia występujących w aplikacji.

Właściciel kompleksów sportowych Jest to pojedyncza osoba (tzn. zwykły „Kowalski") lub pewna instytucja (firma prywatna lub miasto). Właściciel pragnie dodawać nowe obiekty sportowe wraz z informacjami jakie dyscypliny sportu one oferują.

Klient Klient to pasjonat sportu. Pragnie on znaleźć obiekt, który pozwoli mu na uprawianie danej dyscypliny sportu w dogodnym miejscu, czasie i z odpowiednią liczbą osób.

Czas Niektóre interakcje między aktorami a stworzoną aplikacją mogą odbywać się tylko w określonych ramach czasowych. Przykładem na to jest wydarzenie sportowe - zainteresowani sportowcy mają tylko określoną ilość czasu na dołączenie do takiego wydarzenia.

System dostawcy tożsamości Dostawca tożsamości pozwala na zweryfikowanie czy dana osoba jest tą, za którą się podaje. Zwalnia również autora aplikacji z obowiązku przechowywania poufnych danych.

System płatności System płatności umożliwia płatność z konta klienta na konto właściciela kompleksów sportowych. System ten rozwiązuje problem gromadzenia funduszy przez jedną osobę w celu opłacenia rezerwacji (jak ma to miejsce w rozwiązaniach nieformalnych, opisanych w 1.3.1)

System geograficzno-mapowy System ten umożliwia wyświetlanie mapy z obiektami sportowymi. Dodatkowo pozwala on na tłumaczenie adresów (ulica, numer budynku, kod pocztowy i miasto) na współrzędne geograficzne.

2.2 Model dziedziny

Model dziedziny to opis najważniejszych jednostek znajdujących się w obszarze tworzonego systemu. Wraz ze spisem aktorów tworzy on spójny język przy pomocy, którego można opisać wszystkie interakcje zachodzące w systemie. W przypadku większych projektów model ten pozwala na sprawną komunikację z ekspertami dziedzinowymi - osobami nietechnicznymi, które wiedzą jakie relacje biznesowe występują między elementami modelu.

Kompleks sportowy Jest to jednostka, która skupia jeden lub więcej obiektów sportowych. Przykładem kompleksu sportowego jest Ośrodek Sportu i Rekreacji Targówek.

Obiekt sportowy Jednostka, która składa się z jednej lub więcej aren sportowych. Posiada adres składający się z ulicy, numeru budynku, kodu pocztowego i nazwy miasta. Posiada również współrzędne geograficzne. Lokalizacja obiektów sportowych wpływa na ich atrakcyjność dla klientów. Przykładem jest basen Polonez położony na warszawskim Targówku.

Arena sportowa Wydzielona część obiektu sportowego. Umożliwia uprawianie określonej liczby dyscyplin sportowych. Na arenie sportowej odbywają się wydarzenia sportowe.

Wydarzenie sportowe Wydarzenie, które skupia pewną ilość uczestników i jest związane z pewną dyscypliną sportu. Przykładem jest mecz piłki nożnej.

Zewnętrzne wydarzenie sportowe Wydarzenie, które zostało stworzone poza aplikacją. Przykładem są zawody pływackie na basenie uzgodnione w rozmowie telefonicznej między właścicielem kompleksów sportowych a organizatorem zawodów.

Wybrany został taki model dziedziny, ponieważ zdaniem autora najlepiej odzwierciedla on powiązania występujące w rzeczywistości.

2.3 Opis przypadków użycia

Przypadki użycia to prosta, lecz skuteczna metoda opisu procesów zachodzących w tworzonym systemie. Powrót do przypadków użycia po napisaniu pewnej części systemu pozwala łatwo określić jak zaawansowane są prace.

Dla uproszczenia niektórych zapisów przyjęto następujące założenia:

- Użytkownik to klient lub właściciel kompleksów sportowych
- Akcja to dodanie, edycje lub usunięcie
- Jednostka to kompleks sportowy, obiekt sportowy, arena sportowa lub zewnętrzne wydarzenie sportowe

Poniżej znajduje się lista przypadków użycia:

- Rejestracja nowego właściciela kompleksów sportowych
    1. Właściciel kompleksów sportowych podaje niezbędne informacje w formularzu rejestracyjnym
    2. Aplikacja rejestruje właściciela kompleksów sportowych
- Logowanie właściciela kompleksów sportowych

1. Właściciel kompleksów sportowych podaje login i hasło
2. Właściciel kompleksów sportowych zostaje zalogowany do aplikacji

- Logowanie klienta
  1. Klient klika przycisk odpowiedzialny za logowanie przez serwis Facebook
  2. Klient loguje się do serwisu Facebook
  3. Klient zostaje zalogowany do aplikacji
     - Jeśli klient loguje się do aplikacji po raz pierwszy to zostaje w niej zarejestrowany
- Wykonanie akcji na jednostce przez właściciela kompleksów sportowych
  1. Właściciel kompleksów sportowych wchodzi na stronę zawierającą formularz umożliwiający wykonanie danej akcji
  2. Właściciel kompleksów sportowych podaje odpowiednie informacje
  3. Aplikacja weryfikuje poprawność informacji
     - W przypadku błędu właściciel kompleksów sportowych może poprawić informacje
  4. Aplikacja weryfikuje czy żądana akcja nie naruszy integralności danych
     - Wyświetlenie komunikatu o braku możliwości wykonania akcji w przypadku naruszenia integralności danych
  5. Aplikacja wykonuje żądaną akcję
- Wyszukanie wydarzenia sportowego
  1. Klient wchodzi na stronę z wyszukiwarką
  2. Klient podaje wartości następujących parametrów wyszukiwania: dyscyplina sportu, termin, lokalizacja
  3. Aplikacja wyszukuje wydarzenia sportowe spełniające podane kryteria
  4. Aplikacja wyświetla wydarzenia sportowe na mapie z uwzględnieniem ceny wydarzenia
- Płatność za wydarzenie sportowe
  1. Klient wchodzi na stronę wydarzenia sportowego
  2. Klient klika przycisk odpowiedzialny za dokonanie płatności
  3. Klient zostaje przekierowany do serwisu płatniczego, gdzie zatwierdza płatność
  4. Aplikacja dokonuje płatności z konta klienta na konto właściciela kompleksów sportowych
- Utworzenie nowego wydarzenia sportowego
  1. Klient wchodzi na stronę danej areny sportowej
  2. Klient klika przycisk odpowiedzialny za utworzenie nowego wydarzenia sportowego
  3. Klient zostaje przeniesiony na stronę prezentującą formularz tworzenia nowego wydarzenia sportowego
  4. Klient podaje wymagane informacje

5. Aplikacja tworzy nowe wydarzenie sportowe

- Dołączenie do wydarzenia sportowego

1. Klient wchodzi na stronę areny sportowej i wyświetla listę wydarzeń
2. Klient klika na przycisk odpowiedzialny za dołączenie do wydarzenia sportowego
   - W przypadku niemożności dołączenia do wydarzenia klient jest o tym informowany
3. Aplikacja dołączą klienta do wydarzenia sportowego

- Zakończenie etapu dołączania do wydarzenia sportowego (Wydarzenie wyzwalane czasowo)

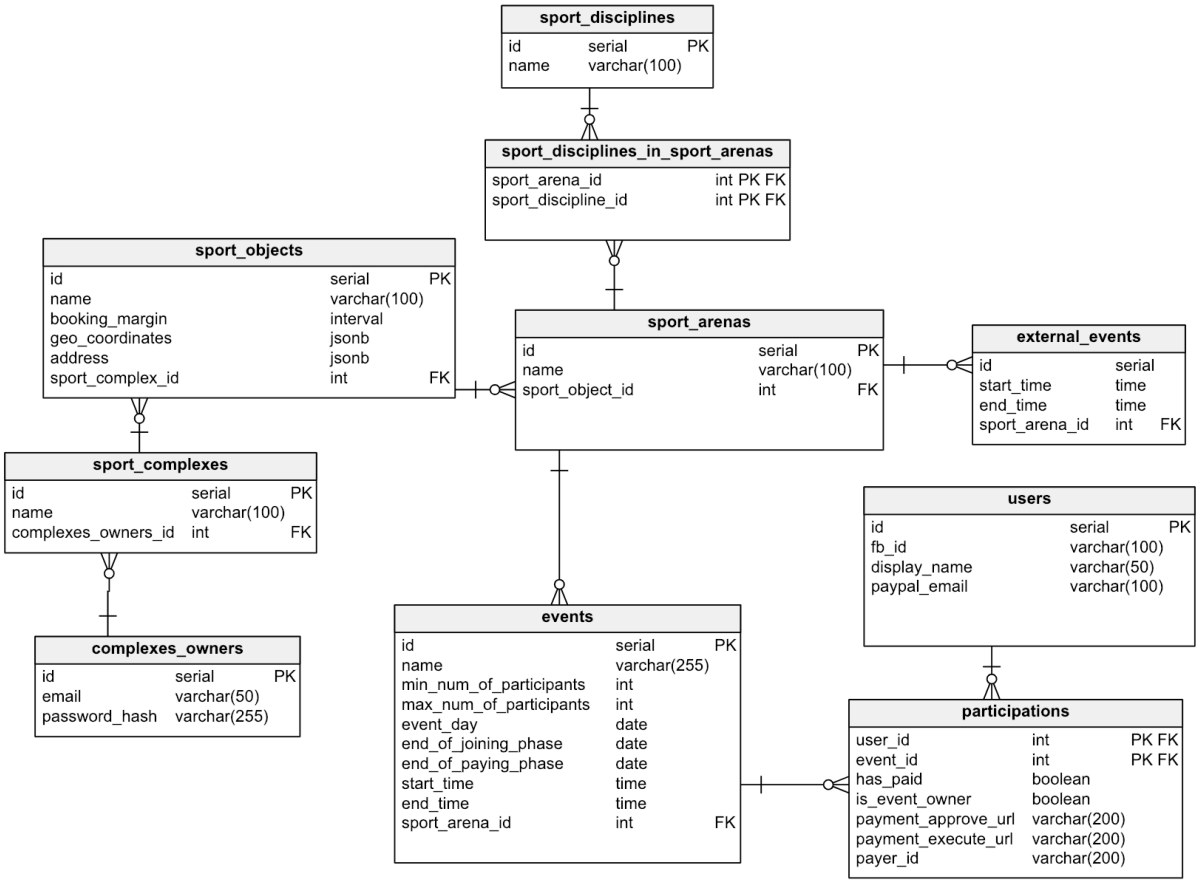
1. Następuje zdefiniowany przez klienta moment zakończenia fazy dołączania do wydarzenia sportowego
2. Aplikacja uniemożliwia dołączanie do wydarzenia sportowego

- Zakończenie etapu płatności za wydarzenie sportowe (Wydarzenie wyzwalane czasowo)

1. Następuje zdefiniowany przez klienta moment zakończenia fazy płatności za wydarzenie sportowe
2. Aplikacja sprawdza czy wszyscy użytkownicy wnieśli opłatę
   - Jeśli któryś z użytkowników nie wniósł opłaty wydarzenie nie odbywa się

2.4 Diagram związków encji

Wcześniejsze podrozdziały pozwoliły na zrozumienie struktury danych i procesów zachodzących w stworzonej aplikacji. Na tej podstawie skonstruowano diagram związków encji. Diagram widoczny jest na rysunku 2. Korzystając z niego stworzono bazę danych.



Rysunek 2: Diagram związków encji.

3 Wykonanie

3.1 Architektura

Stworzona aplikacji składa się z trzech części:

- Aplikacji typu SPA określanej w dalszej części tej pracy jako "aplikacja frontendowa"
- API, które udostępnia interfejs dla aplikacji frontendowej
- Aplikacja backendowej, która wykonuje operacje na danych i komunikuje się z innymi systemami

3.1.1 Czym jest SPA?

SPA oznacza *Single Page Application*. Jest to architektura, która umożliwia tworzenie aplikacji o bogatym interfejsie użytkownika. Określnie *Single Page* bierze się stąd, iż w czasie interakcji z taką aplikacją użytkownik widzi ciągle ten sam dokument HTML. W aplikacjach tego typu przejście do nowej strony oznacza usunięcie niektórych obecnie wyświetlanych elementów HTML oraz dodanie nowych.

W aplikacjach SPA elementy HTML grupuje się w komponenty. W ramach jednej strony komponent może być wielokrotnie wyświetlony. W celu wypełnienia nowych komponentów danymi aplikacja wysyła w tle żądania do serwera. W czasie pierwszego żądania do serwera pobierana jest zawartość wszystkich wyświetlanych w aplikacji komponentów.

Serwer, z którego aplikacja SPA pobiera dane nie musi być tym samym, na którym jest ona zainstalowana. Częstym scenariuszem w przypadku takich aplikacji jest pobieranie danych z różnych API znajdujących się na innych maszynach.

Alternatywą i historycznie pierwszym rozwiązaniem są aplikacje typu *Multi Page Application*, w skrócie MPA. W przypadku takich aplikacji w momencie kliknięcia na odnośnik do innej strony wysyłane jest żądanie HTTP GET do serwera. Serwer przetwarza żądanie i zwraca odpowiedź w postaci gotowego do wyświetlenia dokumentu HTML.

W przypadku MPA nie ma konieczności wykorzystywania API w celu dostępu do danych. Ponieważ do serwera trafia żądanie wyświetlenia nowej strony może on ją od razu wypełnić danymi pobranymi z bazy danych.

Użycie SPA może być uzasadnione jeśli projektowana aplikacja wyświetla strony o zbliżonym układzie. Jeśli powtarzające się elementy zostaną pogrupowane w komponenty to wtedy pobranie ich będzie odbywało się tylko raz - w czasie pierwszego żądania do serwera. Gdyby w takiej aplikacji zamiast architektury SPA użyć MPA to przejście do nowej strony wiązałoby się z pobraniem powtarzających się komponentów.

Wadą aplikacji SPA jest długi (w porównaniu do MPA) czas potrzebny do wyświetlenia aplikacji po raz pierwszy. Jak już wcześniej wspomniano pierwsze żądanie do serwera związane jest z pobraniem wszystkich wyświetlanych w aplikacji komponentów. Rozmiar tych komponentów jest najczęściej dużo większy niż rozmiar pojedynczej strony internetowej. Z tego powodu użytkownik aplikacji może być zmuszony do oglądania pustej strony w oczekiwaniu na pobranie wszystkich komponentów.

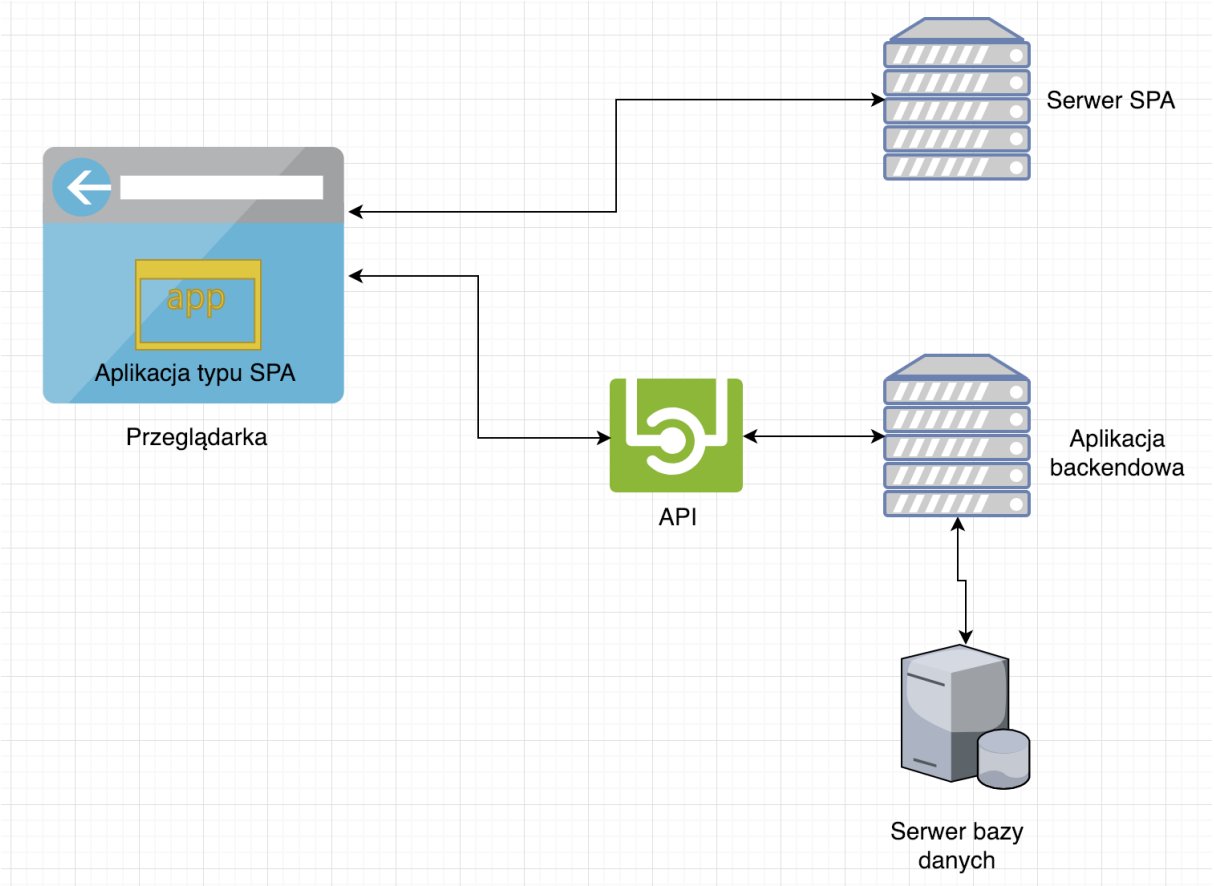
3.1.2 Uzasadnienie wyboru architektrury

Wcześniej opisana architektura była znana autorowi wyłącznie z teoretycznych opracawań. Została ona tutaj zastosowana, aby porównać teorię z praktyką. Autor chciał wyrobić sobie pogląd na to, która z architektur jest lepsza: *Multi Page Application* czy *Single Page Application*

Drugim z powodów zastosowania takiej architektury jest możliwość prostego rozszerzenia systemu o apliakcje na urządzenia mobilne. Takie aplikacje stanowiłyby jedynie warstwę widoku dla już istniejącej aplikacji backendowej.

3.1.3 Schemat komunikacji

Schemat komunikacji widoczny jest na rysunku 3. Przy pierwszym wejściu na stronę następuje pobranie aplikacji frontendowej. Następnie komunikuje się ona z API, które przekazuje żądania do aplikacji backendowej. Aplikacja backendowa przeprowadza operacje na danych, które są później zapisywane w bazie danych.



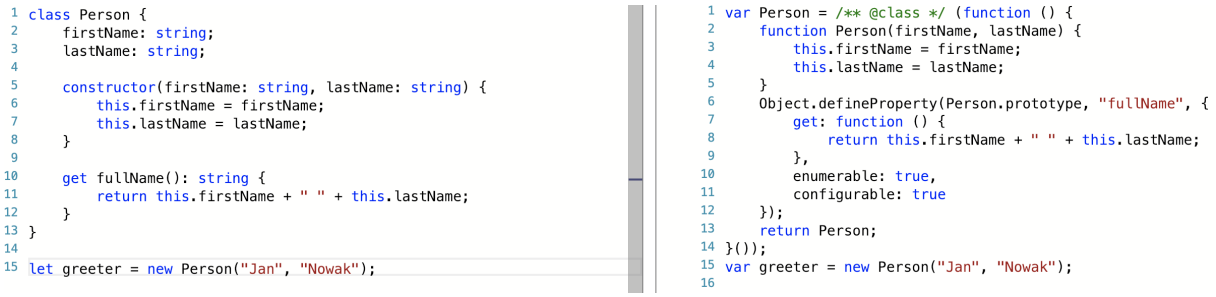
Rysunek 3: Schemat komunikacji.

3.2 Przegląd wybranych technologii i narzędzi

3.2.1 JavaScript (TypeScript)

JavaScript to interpretowany oraz dynamicznie typowany język programowania. Jest to jedyny język programowania, który jest rozumiany przez wszystkie współczesne przeglądarki internetowe.

Dynamiczne typowanie ułatwia tworzenie prototypów aplikacji. Może jednak stanowić utrudnienie w tworzenie dużych systemów. Z tego powodu firma Microsoft stworzyła TypeScript[3]. Jest to statycznie typowany nadzbiór JavaScriptu, który umożliwia użycie klas, typów wyliczeniowych, interfejsów oraz wiele więcej[4]. Kod TypeScriptu jest kompilowany do JavaScriptu - wszędzie tam, gdzie użyty jest JavaScript można użyć TypeScriptu. Efekt tej kompilacji jest widoczny na rysunku 4.



```typescript
class Person {
    firstName: string;
    lastName: string;

    constructor(firstName: string, lastName: string) {
        this.firstName = firstName;
        this.lastName = lastName;
    }

    get fullName(): string {
        return this.firstName + " " + this.lastName;
    }
}

let greeter = new Person("Jan", "Nowak");
```

```javascript
var Person = /** @class */ (function () {
    function Person(firstName, lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }
    Object.defineProperty(Person.prototype, "fullName", {
        get: function () {
            return this.firstName + " " + this.lastName;
        },
        enumerable: true,
        configurable: true
    });
    return Person;
}());
var greeter = new Person("Jan", "Nowak");
```

Rysunek 4: Po lewej kod TypeScriptu, po prawej odpowiadający kod JavaScriptu. Źródło: [5]

Język ten został wybrany ze względu na dużą liczbę funkcjonalności, o którą rozszerza JavaScript oraz statycznie typowanie. W opinii autora statyczne typowanie lepiej nadaje się do tworzenia oprogramownia, które ma być rozwijane przez długi okres czasu (jak w przypadku aplikacji stworzonej w ramach tej pracy dyplomowej).

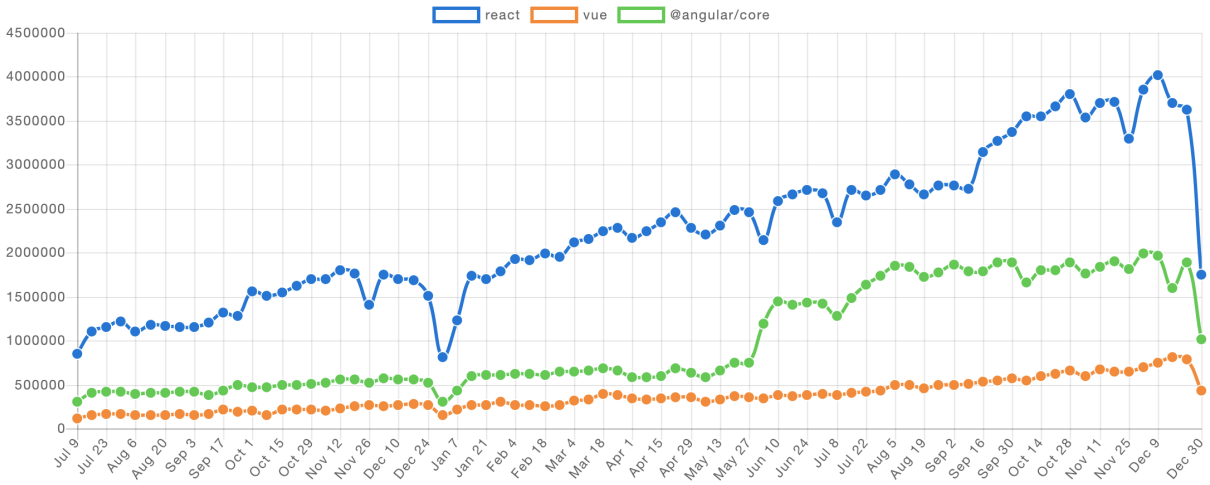
3.2.2 API do geolokalizacji

Geolokalizacja to funkcjonalność udostępniana przez wszystkie współczesne przeglądarki. Przeglądarki udostępniają API napisane w JavaScripcie, które umożliwia pobranie i śledzenie pozycji użytkownika.

Aplikacja frontendowa wykorzystuje geolokalizacje, aby wyświetlić mapę wyśrodkowaną na obecnej lokalizacji użytkownika.

3.2.3 Angular

Angular to architektura umożliwiająca tworzenie aplikacji webowych w TypeScripcie autorstwa korporacji Google[6]. Jest to jedno z trzech najpopularniejszych rozwiązań używanych do tworzenia aplikacji typy SPA. Ilustruje to rysunek 5.



Rysunek 5: Liczba pobrań 3 najpopularniejszych architektur umożliwiających tworzenie aplikacji SPA na przestrzeni ostatnich 2 lat. Angular na zielono. Źródło: [8]

Angular dostarcza oprogramowanie typu CLI (*Command line interface*)[7], które generuje pliki konfiguracyjne z rozsądnymi wartościami domyślnymi co znacznie skraca czas potrzebny na skonfigurowanie projektu.

Opinionated software (posłużono się tutaj angielską nazwą, gdyż w polskiej literaturze brak jest jednoznacznego tłumaczenia) oznacza oprogramowanie, którego twórcy wyznają zasadę, iż dany problem ma jedno, poprawne rozwiązanie. Użycie alternatywnych rozwiązań w takim oprogramowaniu wydłuża czas implementacji i może doprowadzić do obniżenia wydajności lub pojawienia się niespodziewanych błędów.

Angular należy do kategorii *opinionated software*. Jedną z sytuacji, kiedy się to objawia jest obsługa danych asynchronicznych. Angular wykorzystuje do tego celu bibliotekę RxJS [19] i jej typ danych *Observable*. Zamiast typu danych *Observable* można użyć natywnego dla JavaScriptu typu danych *Promise*, lecz wiąże się to z koniecznością konwersji z typu *Promise* do typu *Observable*, gdyż większość funkcji udostępnianych przez Angulara operuje na typie *Observable*.

Angular został wybrany, ze względu na jego dużą integrację z TypeScriptem, która pozwoliła uniknąć problemów związanych z dynamicznym typowaniem.

3.2.4 Elixir

Elixir to dynamicznie typowany, funkcyjny język programowania[9]. Dzięki wsparciu dla paradygmatu funkcyjnego język ten umożliwia wykorzystanie funkcji wyższego rzędu, dopasowania do wzorca oraz pozwala na użycie funkcji takich jak *reduce, filter, map*. Własności te pozwalają na łatwą manipulację danymi.

Język ten został wybrany ze względu na chęć poznania przez autora paradygmatu funkcyjnego. Dodatkową zaletą jest fakt, iż Elixir posiada dobrą i wyczerpującą dokumentacje. Autorzy języka uznają, iż dokumentacja kodu jest równie ważna jak sam kod.[15]

3.2.5 MVC - Model-Widok-Kontroler

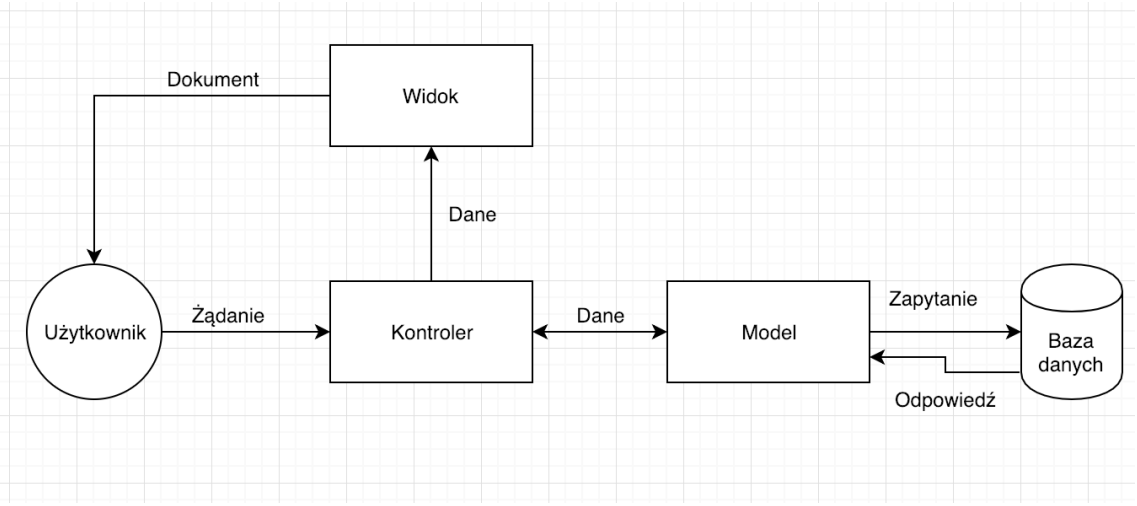
Model architektury, pozwalający odseparować funkcjonalności związane z wyświetlaniem od tych związanych z manipulacją danymi. Model ten jest często wybierany w przypadku tworzenia aplikacji webowych.

Widok to szablon, który jest wypełniany danymi. Widok definiuje wyłącznie układ dokumentu, który jest zwracany klientowi. Nie zawiera on żadnej logiki biznesowej.

Model zapewnia dostęp do bazy danych oraz mapuje dane z domeny relacyjnej do domeny obiektowej bądź strukturalnej.

Kontroler łączy model z widokiem oraz zarządza obsługą żądań, które trafiają do aplikacji. W tym celu pobiera on dane z modelu, przetwarza je a następnie przekazuje do widoku.

Rysunek 6 ilustruje zależności w architekturze MVC.



Rysunek 6: Architektura MVC

3.2.6 Phoenix

Phoenix to oprogramowanie umożliwiające tworzenie aplikacji webowych w architekturze MVC. Cechą charakterystyczną tego rozwiązania jest interfejs o nazwie *Plug*[10]. Interfejs ten umożliwia wygodne i intuicyjne dołączenie dowolnego modułu oprogramowania do kodu, który przetwarza żądanie HTTP.

Oprogramowanie to zostało wybrane, ponieważ jest to jedyne kompleksowe rozwiązane w języku Elixir, które pozwala na tworzenie aplikacji webowych.

3.2.7 PostgreSQL

Dane wytworzone w ramach działania aplikacji są przechowywane w bazie danych. Do tego celu wykorzystano system zarządzania relacyjną bazą danych PostgreSQL. Jest to system z otwartym kodem źródłowym, który obecny jest na rynku od ponad 30 lat.

System ten została wybrany, ponieważ możliwe jest jego wykorzystanie na wielu systemach operacyjnych o różnych parametrach sprzętowych. Dodatkowo wiele języków programowania posiada biblioteki umożliwiające połączenie z tą bazą.

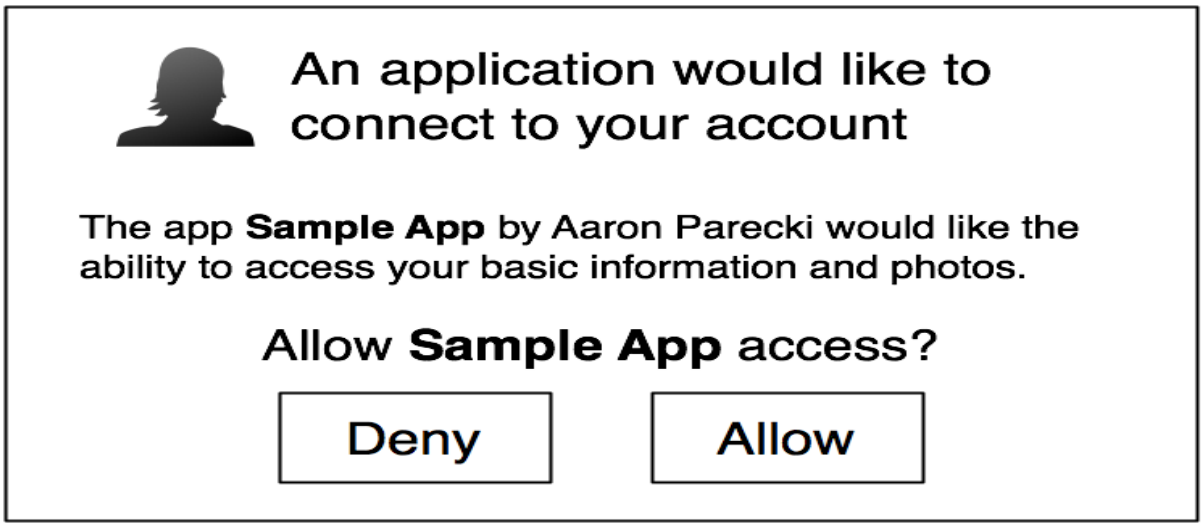
3.2.8 OAuth 2

OAuth 2 to protokół autoryzacyjny, z którego korzystają wymienione w następnych podrozdziałach API (tj.: Facebook, Google Maps oraz Paypal). Zostanie on tutaj omówiony, ponieważ jego znajomość konieczna jest do korzystania ze wspomnianych wcześniej API.

Korzystanie z API zgodnego z OAuth2 należy rozpocząć od rejestracji nowej aplikacji u dostawcy danego API. Aby zarejestrować nową aplikację należy podać jej nazwę, adres, logo itp. oraz adres przekierowania. Adres przekierowania to adres, pod który API przekieruje odebrane żądania po tym jak zostanie one obsłużone. Niektóre API umożliwiają podanie dwóch adresów przekierowania: jednego dla żądania obsłużonego poprawnie i drugiego dla żądania, które nie mogło zostać obsłużone.
Po rejestracji aplikacji programista otrzymuje dwie wartości: identyfikator oraz sekret. Są one wykorzystywane przy wykonywaniu żądań do API. Dzięki temu dostawcy API wiedzą, która aplikacja wykonała żądanie.

Następny krok polega na pozyskaniu tokenu. Token to ciąg znaków, który pozwala na dostęp do chronionych zasobów. Najczęściej jest on przekazywany w nagłówku *Authorization* żądania HTTP. Dla ułatwienie opisu zostanie tutaj pokazany scenariusz pozyskania tokenu, który uprawnia do odczytu adresu mailowego użytkownika serwisu społecznościowego.

Po rejestracji aplikacji należy pozyskać zgodę użytkownika serwisu społecznościowego na dostęp do jego danych. W tym celu wyświetlany jest formularz, który prezentuje informacje o aplikacji oraz o tym do jakich danych chce ona uzyskać dostęp. Wygląd takiego formularza został zaprezentowany na rysunku 7.



Rysunek 7: Formularz zgody na dostęp do danych użytkownika. Źródło: [13]

Po udzieleniu zgody API przekierowuje żądanie pod adres wskazany przez programistę (adres przekierowania) wraz z kodem autoryzacyjnym. Następnie aplikacja wymienia kod autoryzacyjny na token.

Ostatnim krokiem jest pobranie adresu mailowego z użyciem otrzymanego wcześniej tokenu.

Przedstawiony tutaj scenariusz zdobycia tokenu jest tylko jednym z dostępnych. Pełna lista jest dostępna pod adresem `https://oauth.net/2/`. W zależności od użytych technologii i urządzeń należy wybrać odpowiedni wariant.

3.2.9 Architektura REST - Representation State Transfer

Jest to architektura oprogramowania. REST zakłada, iż do obsługi żądania wymagane są jedynie dane pochodzące z żądania. W tym sensie jest to architektura bezstanowa, tak samo jak protokół HTTP. Architektura REST została tutaj wspomniana, ponieważ jej znajomość ułatwia korzystanie z API opisanych w następnych podrozdziałach.

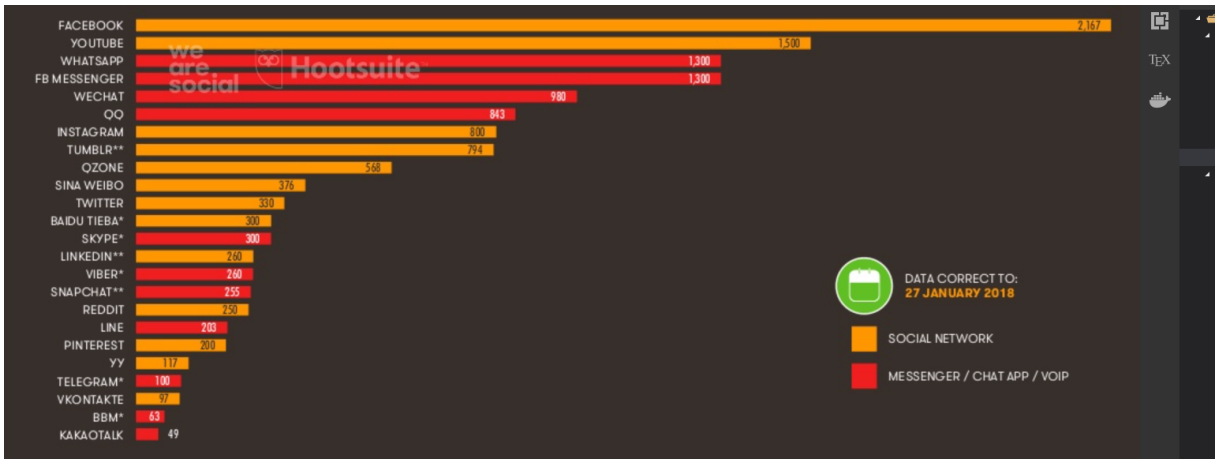
Cechą tej architektury jest modelowanie danych jako zasobów. Każdy zasób identyfikowany jest przez URI. Posiadając URI zasobu można przeprowadzić na nim operacje. Przeprowadzenie operacji oznacza wysłanie żądania określonego typu pod URI zasobu. W przypadku aplikacji webowych wysyłane są żądania HTTP. Rysunek 8 przedstawia typy żądań HTTP i ich wpływ na API w architekturze REST.

Method	Meaning
GET	Read data
POST	Insert data
PUT or PATCH	Update data, or insert if a new id
DELETE	Delete data

Rysunek 8: Operacje na API w architekturze REST. Źródło: `https://lynda.com`

3.2.10 API serwisu Facebook

Facebook to najpopularniejszy serwis społecznościowy na świecie. Ilustruje to rysunek 9. Z tego powodu serwis ten został wykorzystany, aby służyć jako dostawca tożsamości klientów. Klient chcąc zalogować się do aplikacji nie musi tworzyć osobnego konta. Wystarczy, że wyrazi zgodę na dostęp do jego danych na portalu Facebook.



Rysunek 9: Liczba użytkowników serwisów społecznościowych. Źródło: [11]

Portal Facebook udostępnia API, dzięki któremu można uzyskać dostęp do danych jego użytkowników. [12].

3.2.11 API serwisu Google Maps

To API posłużyło jako system geograficzno-mapowy. Udostępnia ono 3 podstawowe funkcjonalności z punktu widzenia aplikacji:

- Geokodowanie
- Odwrotne geokodowanie
- Wyświetlanie mapy z markerami

Geokodowanie to zamiana adresu w formie zrozumiałej dla człowieka na współrzędne geograficznie. Geokodowanie udostępnianie przez API Google Maps nie zakłada, iż adres musi być kompletny - przykładowo adres może składać się wyłącznie z nazwy miejscowości. W takim wypadku usługa geokodowania może zwrócić kilka rezultatów.

Odwrotne geokodowanie to zamiana ze współrzędnych geograficzny na adres zrozumiały dla człowieka. Odwrotne geokodowanie to forma przybliżenia - usługa spróbuje znaleźć najbliższy adres w pewnym obszarze. Jeśli się to nie uda, zwróci błąd.

Narzędzie realizujące operacje geokodowania i odwrotnego geokodowania nazywa się geokoderem.

3.2.12 API serwisu Paypal

Paypal to serwis płatniczy, który umożliwia wykonywanie przelewów pieniężnych między jego użytkownikami. Został wykorzystany, aby wykonywać przelewy między klientami a właścicielami kompleksów sportowych. API tego serwisu umożliwia tworzenie płatności w imieniu klientów.

Istnieje wiele API, które umożliwiają założenie sklepu internetowego i przyjmowanie płatności. Odbiorca płatności jest wtedy odpowiedzialny za integrację z systemem płatności. API serwisu Paypal został wybrane, ponieważ jest to jedyne znane autorowi API, które umożliwia tworzenie płatności bez konieczności integracji ze strony odbiorcy.

3.3 Ciągła integracja (*Continuous integration*)

Ciągła integracja to praktyka programistyczna polegająca na częstym (przynajmniej raz dziennie) i regularnym włączaniu bieżących zmian w kodzie źródłowym do głównego repozytorium projektu oraz budowaniu projektu i uruchamianiu testów jednostkowych.

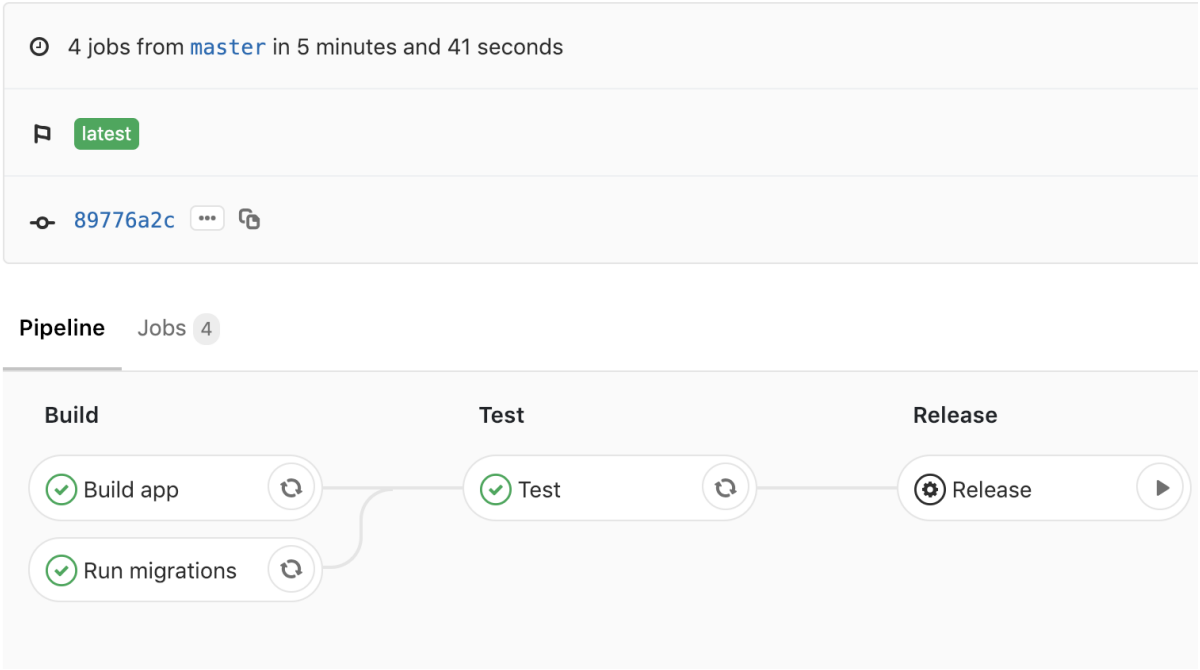
Ciągła integracja redukuje ilość pracy jaką trzeba ponieść, aby połączyć zmiany wykonane przez kilka różnych osób. Dodatkowo umożliwia wcześniejsze wykrywanie błędów.

W jednoosobowych projektach stosowanie tej praktyki może nie przynieść takich korzyści jak w przypadku projektów wieloosobowych. Jednakże w tym projekcie skorzystano z ciągłej integracji. Pozwoliło to autorowi niniejszej pracy lepiej poznać ciągłą integrację oraz narzędzie wspierające tą praktykę.

3.3.1 Gitlab

Gitlab to repozytorium dla systemu kontroli wersji git. Jedną z funkcjonalności jaką Gitlab oferuje jest wsparcie dla ciągłej integracji.

Każde wypchnięcie zmian z lokalnego repozytorium do zdalnego repozytorium powoduje uruchomienie zestawu procesów określanego jako *pipeline*. Procesy te mają na celu sprawdzenie czy projekt zbuduje się oraz czy wszystkie testy przejdą. Dodatkowo, w przypadku wypchnięcia zmian z gałęzi *master* istnieje możliwość załadowania zmian na środowisko produkcyjne. Należy zaznaczyć, iż załadowanie tych zmian odbywa się dopiero po dodatkowym potwierdzeniu tego faktu przez uprawnionego programistę. Rysunek 10 przedstawia przykładowy *pipeline*.



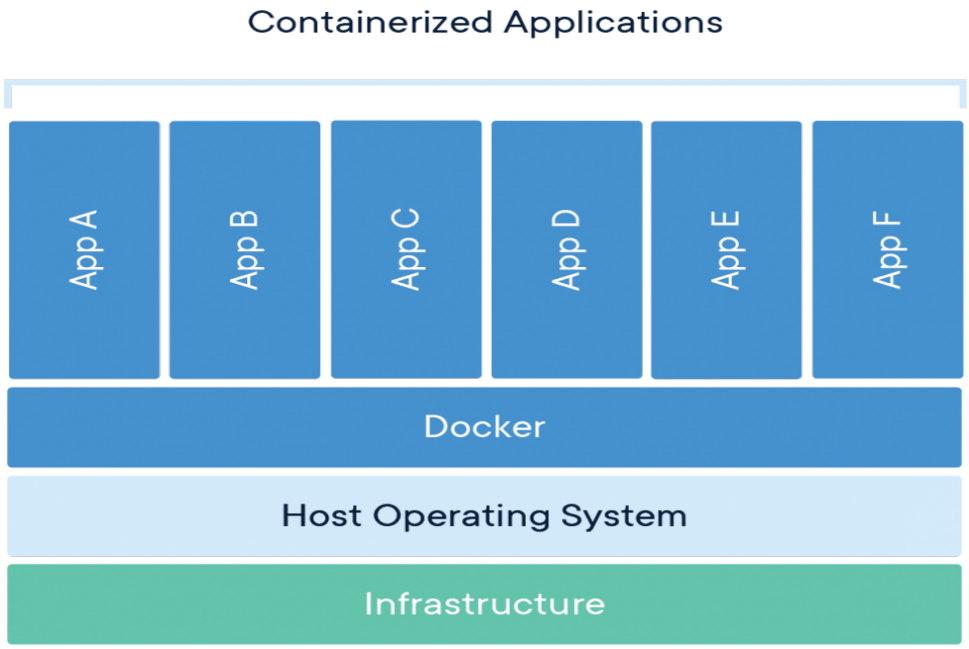
Rysunek 10: *Pipeline* z serwisu `https://gitlab.com`, z gałęzi *master*

Cały *pipeline* składa się z etapów. Każdy etap składa się z zadań. Aby przejść z jednego etapu do drugiego wszystkie zadania muszą zakończyć się powodzeniem. Wyjątkiem są zadania oznaczone flagą, która dopuszcza wystąpienie błędu. Jeśli w takim zadaniu wystąpi błąd to nie wpływa ono na przejście lub nie przejście do następnego etapu.

3.3.2 Docker

Docker to oprogramowanie, które umożliwia uruchamianie programów w osobnym, odizolowanym od systemu operacyjnego środowisku nazywanym kontenerem.[17] Rysunek 11

ilustruje gdzie w architekturze współczesnego systemu operacyjnego znajdują się kontenery.



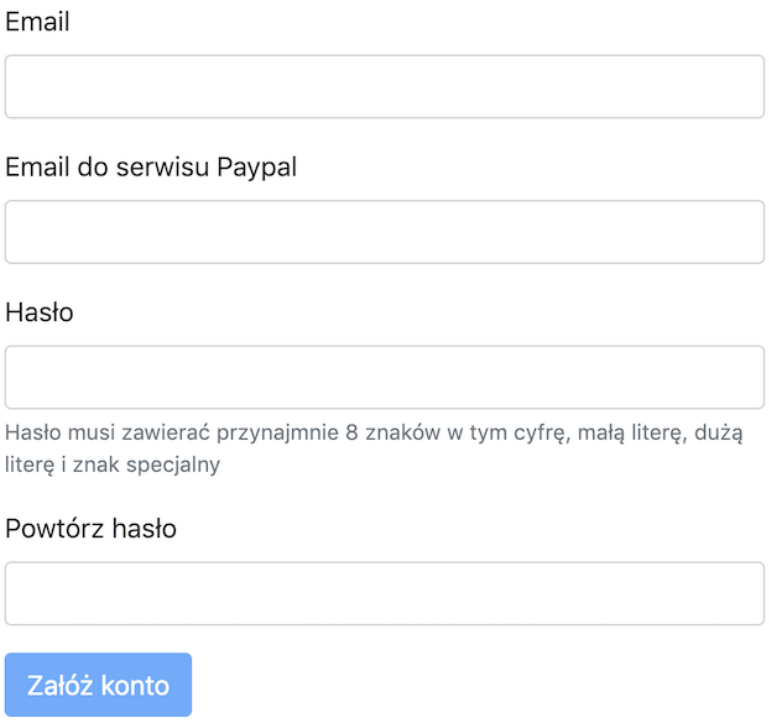
Rysunek 11: Miejsce kontenera w architekturze systemu operacyjnego. Źródło: `https://www.docker.com/resources/what-container`

Gitlab wykorzystuje Dockera w ramach ciągłej integracji. Każde zadanie z etapu jest uruchamiane w osobnym kontenerze. Dzięki temu środowisko uruchomieniowe zadania imituje środowisko produkcyjne - posiada ono jedynie kod źródłowy aplikacji i moduły niezbędne do jej uruchomienia.

3.4 Zrealizowane przypadki użycia

3.4.1 Rejestracja nowego właściciela kompleksów sportowych

Użytkownik wchodzi na stronę zawierającą formularz rejestracyjny i wypełnia wszystkie podane pola. Wygląd formularza został przedstawiony na rysunku 12. Pola "Email" i "Email do serwisu Paypal" muszą zawierać poprawne (w sensie formatu) adresy mailowe. Pole "Hasło" musi składać się z odpowiedniej liczby znaków o czym informuje tekst pod tym polem. Pole "Powtórz hasło" musi zawierać ten sam tekst co pole "Hasło". Przycisk "Załóż konto" jest domyślnie wyłączony. Dopiero odpowiednie wypełnienie pól formularza powoduje jego odblokowanie.



Rysunek 12: Formularz umożliwiający rejestrację nowego właściciela kompleksów sportowych

Po kliknięciu w przycisk "Załóż konto" następuje przesłanie danych nowego właściciela do aplikacji backendowej. Weryfikuje ona poprawność danych i w przypadku braku błędów zwraca informacje o nowym właścicielu do aplikacji frontendowej. Następnie aplikacja frontendowa przekierowuje właściciela do strony zarządzania kompleksami sportowymi.

3.4.2 Rejestracja nowego klienta

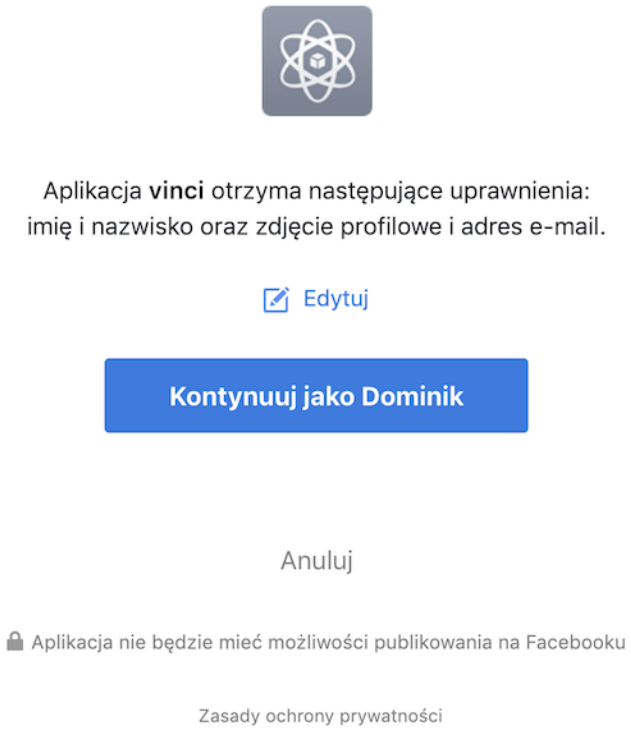
Rejestracja klienta polega na utworzeniu nowego profilu w bazie danych. Ma to miejsce gdy klient po raz pierwszy loguje się do aplikacji (Patrz 3.4.4)

3.4.3 Logowanie właściciela kompleksów sportowych

Logowanie właściciela kompleksów sportowych to logowanie z użyciem loginu i hasła. Właściciel kompleksów sportowych podaje te wartości w formularzu. Następnie są one przesyłane do aplikacji backendowej. Przy użyciu loginu aplikacja backendowa wyszukuje w bazie danych rekord z danymi właściciela. Następnie zostaje określony kryptograficzny skrót hasła dostarczonego do aplikacji backendowej. Skrót ten jest porównywany z wartością obecną w bazie danych. W przypadku zgodności następuje zalogowanie właściciela kompleksów sportowych.

3.4.4 Logowanie klienta

Klient klika w aplikacji na przycisk "Zaloguj się przez Facebooka". Następuje przekierowanie do tego serwisu. Klient widzi komunikat o udzielenie dostępu do danych ze swojego profilu. Treść tego komunikatu została ukazana na rysunku 13. Po wyrażeniu zgody klient jest przekierowywany do aplikacji backendowej, która tworzy w bazie danych rekord zawierający dane nowego użytkownika z unikalnym identyfikatorem pobranym z serwisu Facebook. Na koniec klient jest przekierowywany do aplikacji frontendowej.

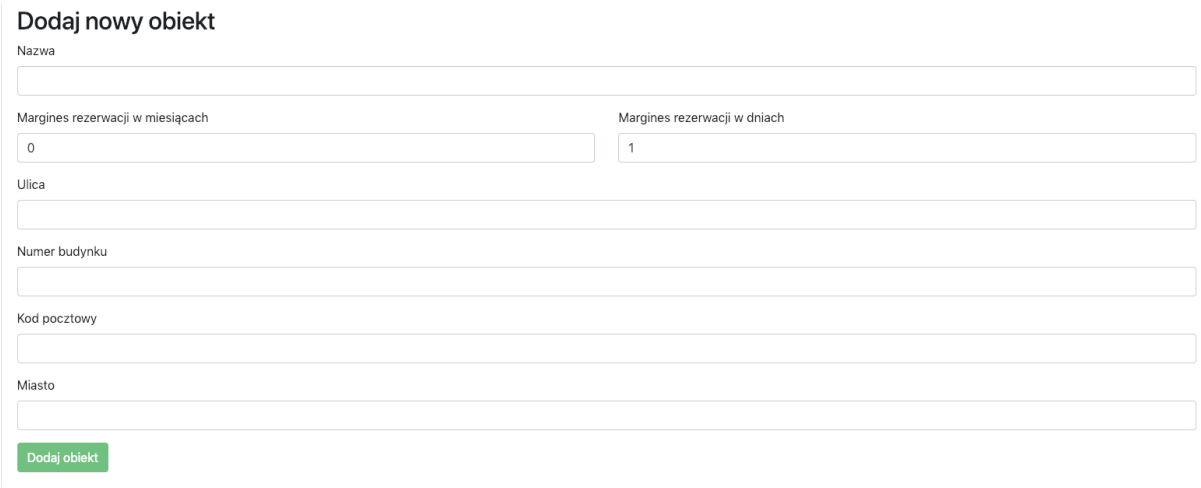


Rysunek 13: Zgoda na dostęp do danych użytkownika Facebooka

Wyrażenie zgody na dostęp do danych ma miejsce tylko raz. Chcąc się ponownie zalogować wystarczy, że klient kliknie przycisk "Zaloguj się przez Facebooka". Aplikacja backendowa połączy się wtedy z serwisem Facebook i pobierze identyfikator użytkownika. Następnie sprawdzi czy użytkownik o takim identyfikatorze figuruje w bazie danych. Jeśli tak będzie to klient zostanie zalogowany.

3.4.5 Dodanie nowego obiektu sportowego

Właściciel kompleksów sportowych wchodzi na stronę kompleksu sportowego i otwiera formularz umożliwiający dodanie nowego obiektu sportowego. Formularz ten przedstawiony jest na rysunku 14.



Rysunek 14: Formularz umożliwiający dodanie nowego obiektu

Kliknięcie przycisku "Dodaj obiekt" powoduje wysłanie zapytanie do usługi geokodowania udostępnionej przez API Google Maps. Przykładową odpowiedź otrzymaną od tej usługi przedstawia rysunek 15. Należy zwrócić uwagę, iż w przypadku podania dokładnego adresu tablica *results* zawiera tylko jeden element.

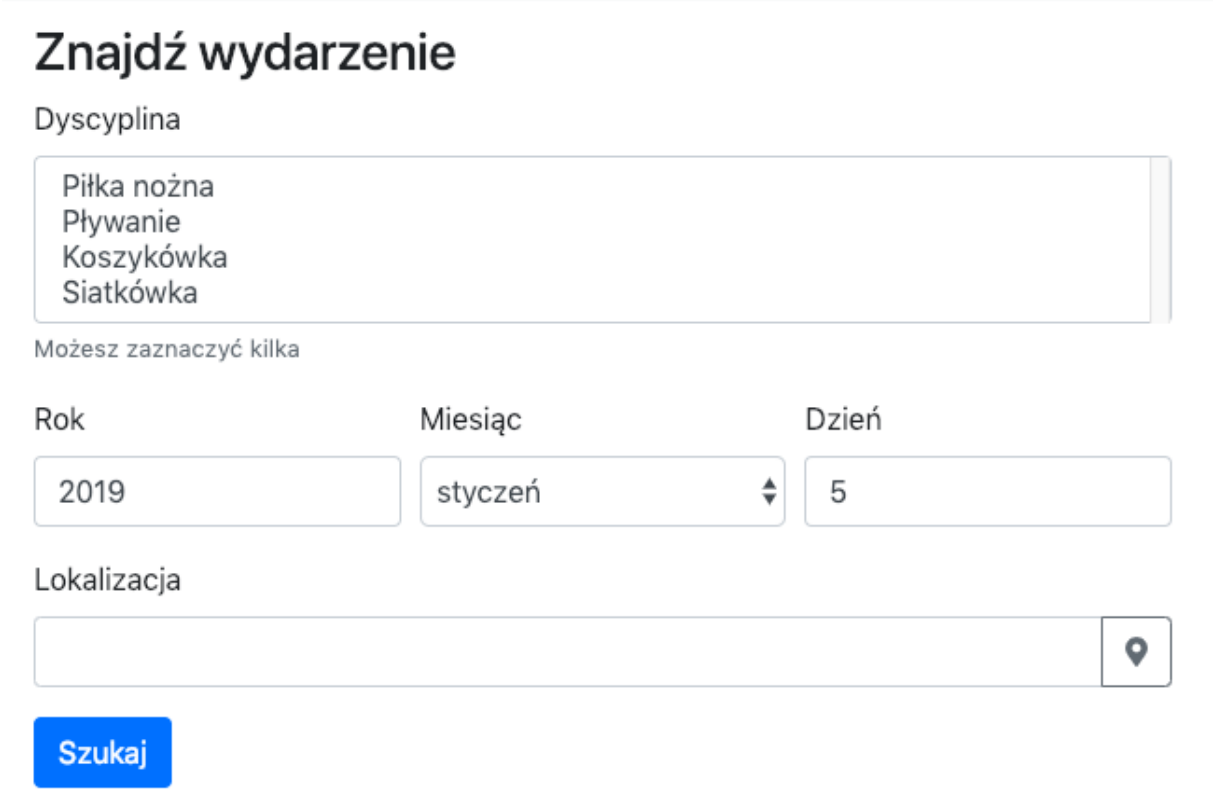

```
▼ results: [{address_components: [{long_name: "15/19", short_name: "15/19", types: ["street_number"]},…],…}]
  ▼ 0: {address_components: [{long_name: "15/19", short_name: "15/19", types: ["street_number"]},…],…}
    ▼ address_components: [{long_name: "15/19", short_name: "15/19", types: ["street_number"]},…]
      ▶ 0: {long_name: "15/19", short_name: "15/19", types: ["street_number"]}
      ▶ 1: {long_name: "Nowowiejska", short_name: "Nowowiejska", types: ["route"]}
      ▶ 2: {long_name: "Śródmieście", short_name: "Śródmieście",…}
      ▶ 3: {long_name: "Warszawa", short_name: "Warszawa", types: ["locality", "political"]}
      ▶ 4: {long_name: "Warszawa", short_name: "Warszawa", types: ["administrative_area_level_2", "political"]}
      ▶ 5: {long_name: "mazowieckie", short_name: "mazowieckie",…}
      ▶ 6: {long_name: "Polska", short_name: "PL", types: ["country", "political"]}
      ▶ 7: {long_name: "00-665", short_name: "00-665", types: ["postal_code"]}
      formatted_address: "Nowowiejska 15/19, 00-665 Warszawa, Polska"
    ▼ geometry: {,…}
      ▼ bounds: {northeast: {lat: 52.2192637, lng: 21.0129685}, southwest: {lat: 52.2187071, lng: 21.0107122}}
        ▶ northeast: {lat: 52.2192637, lng: 21.0129685}
        ▶ southwest: {lat: 52.2187071, lng: 21.0107122}
      ▶ location: {lat: 52.21890399999999, lng: 21.0115164}
        location_type: "ROOFTOP"
      ▶ viewport: {northeast: {lat: 52.22033438029149, lng: 21.0131893302915},…}
      place_id: "ChIJvV1P4ujMHkcRh1oux06dA4k"
    ▶ types: ["premise"]
  status: "OK"
```


Rysunek 15: Odpowiedź z geokodera dla gmachu WEiTI

Jeśli geokoder nie zwróci żadnych wyników to aplikacja wyświetla komunikat, iż podany adres jest nieprawidłowy. Po otrzymaniu pozytywnej odpowiedzi od geokodera następuje wysłanie żądania HTTP POST do aplikacji backendowej w celu utworzenia nowego obiektu sportowego.

3.4.6 Wyszukiwanie areny sportowej

Klient wchodzi na stronę wyszukiwania i wypełnia formularz. Formularz wyszukiwania przedstawia rysunek 16. Rok, miesiąc i dzień zawsze wskazują aktualne wartości. Pole lokalizacja umożliwia wpisanie adresu lub skorzystanie z api do geolokalizacji udostępnianego przez przeglądarki.



Rysunek 16: Formularz wyszukiwania

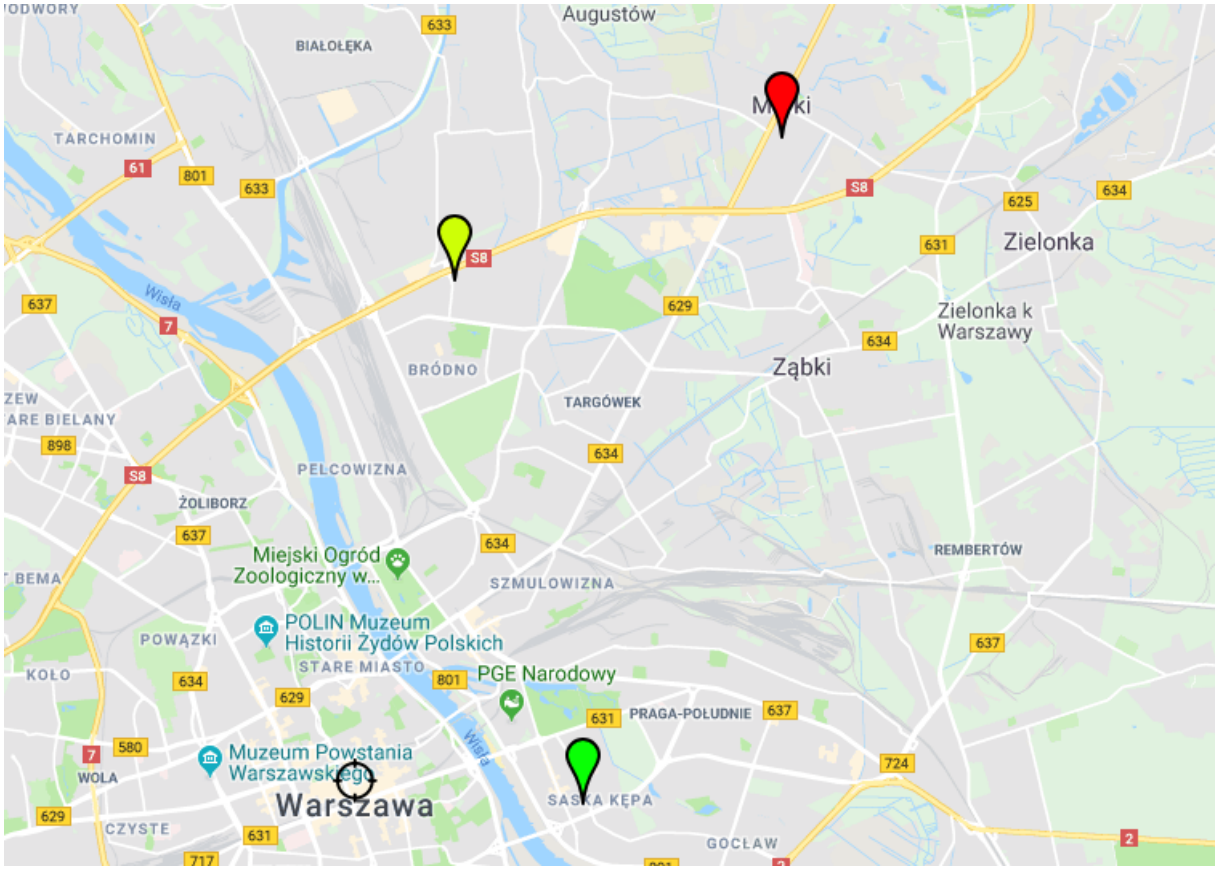
Jeśli użytkownik skorzysta z api do geolokalizacji wtedy zostanie wywołana usługa odwrotnego geokodowania, aby zamienić współrzędne geograficzne użytkownika na adres. Adres ten zostanie wpisany w odpowiednie pole formularza.

Jeśli użytkownik nie skorzysta z api do geolokalizacji wtedy po kliknięciu "Szukaj" wywołana zostanie usługa geokodowania.

Po kliknięciu "Szukaj" do aplikacji backendowej zostaje wysłane żądanie zawierające pożądane dyscypliny sportowe, dzień wydarzenia oraz lokalizację. Aplikacja backendowa wyszukuje te areny sportowe, które:

- wspierają wszystkie otrzymane w żądaniu dyscypliny sportowe,
- posiadają przynajmniej jedno godzinne okienko we wskazanym dniu

Zwracane wyniki są posortowane pod względem ceny. Wyniki prezentowane są na mapie w postaci markerów. Ilustruje to rysunek 17. Symbolem celownika oznaczono lokalizacje wprowadzoną przez użytkownika. Pozostałe markery oznaczają areny sportowe. Kolor tych markerów określa ich atrakcyjność cenową - im bardziej zielony marker tym niższa cena. Do określenia koloru wykorzystano interpolację liniową na przestrzeni barw HSL.



Rysunek 17: Mapa prezentująca wyniki wyszukiwania

Przestrzeń barw HSL Jest to przestrzeń barw, która opisuje kolory za pomocą trzech wartości:

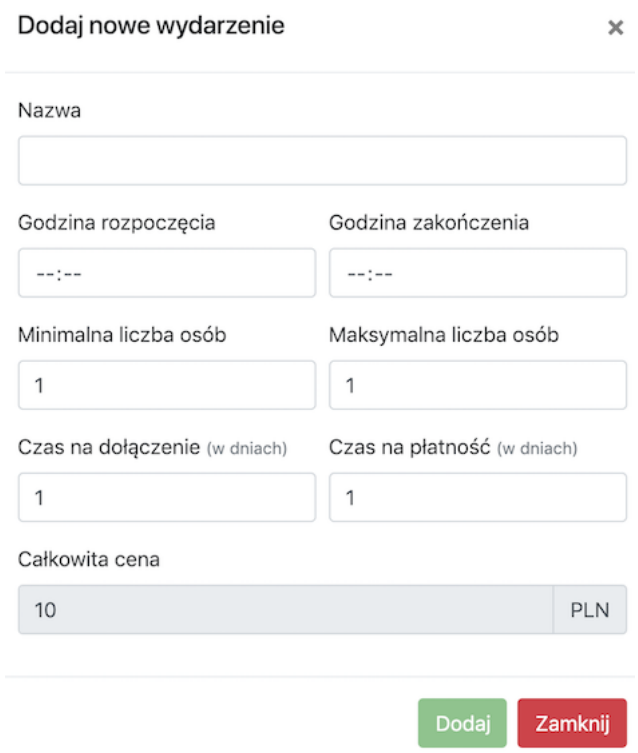
- Odcień (*Hue*), o wartościach z przedziału od 0 do 360 stopni,
- Nasycenie (*Saturation*), o wartościach z przedziału od 0 do 1,
- Średnie światło białe (*Lightness*), o wartościach z przedziału od 0 do 1

Ta przestrzeń barw została wybrana, ponieważ jedną z barw powstałych w wyniku interpolacji jest kolor zółty. Tej barwy nie można uzyskać przy użyciu interpolacji w przestrzeni barw RGB.

3.4.7 Utworzenie nowego wydarzenia sportowego

W celu dodania nowego wydarzenia klient musi wejść na stronę danego ośrodka sportwego a następnie wybrać dzień, w który chciałby zorganizować nowe wydarzenia. Następnie

musi wypełnić formularz, który widoczny jest na rysunku 18.



Rysunek 18: Formularz dodania nowego wydarzenia

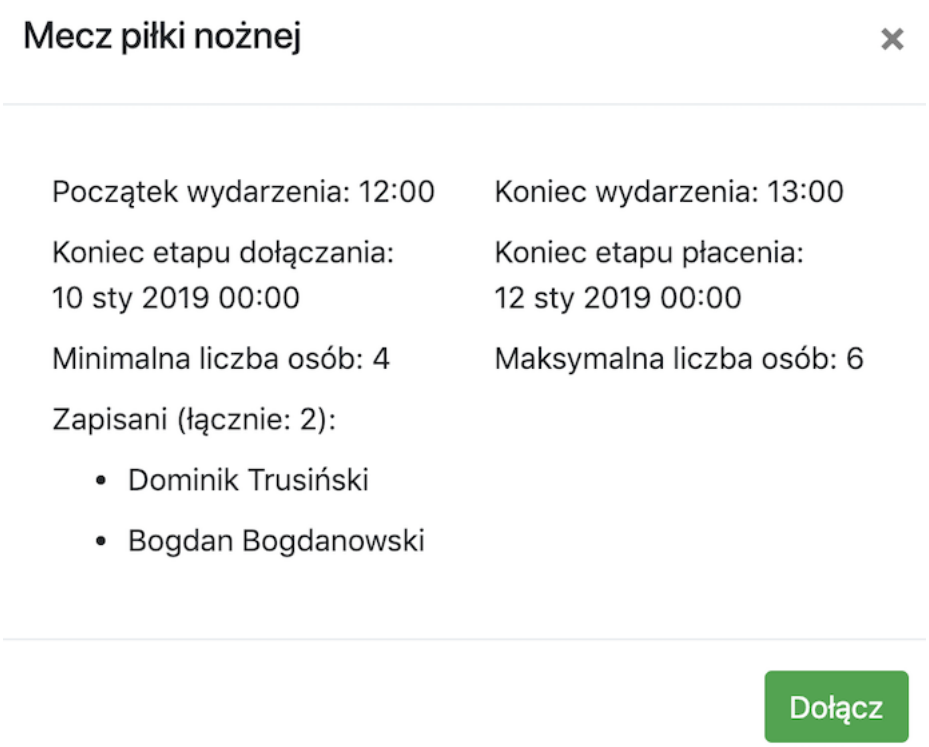
Wszystkie elementy formularza posiadają walidacje na wpisanie poprawnych wartości (tekstu bądź liczby). Dodatkowo, w przypadku pól "Godzina rozpoczęcia" i "Godzina zakończenia" następuje sprawdzenie czy godzina rozpoczęcia wydarzenia nie jest później niż godzina jego zakończenia.

Pole "Całkowita cena" jest zablokowane. Jego wartość zależy od tego jaką cenę ustalił właściciel kompleksów sportowych.

Przycisk "Dodaj" jest początkowo zablokowany. Poprawne wypełnienie formularza sprawia, że staje się on aktywny.

3.4.8 Dołączenie do wydarzenia sportowego

W celu dołączenia do wydarzenia sportowego klient wchodzi na stronę areny sportowej a następnie wykorzystuje kalendarz, aby odnaleźć interesujące go wydarzenie. Następnie otwiera okno dołączenia do wydarzenia. Wygląd tego okno ilustruje rysunek 19



Rysunek 19: Fragment aplikacji prezentujący informacje o wydarzeniu

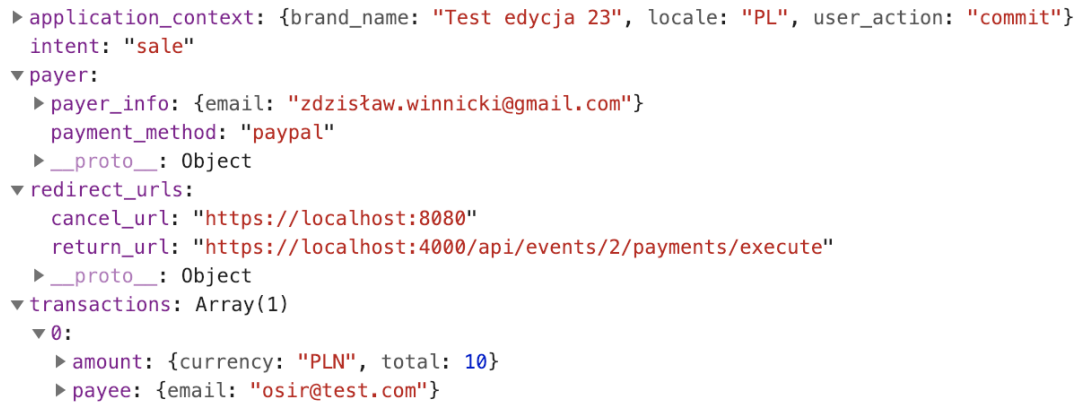
Po kliknięciu w przycisk "Dołącz" klient dołącza do wydarzenia sportowego a jego imię i nazwisko jest wyświetlane na liście uczestników. Po dołączeniu do wydarzenia przycisk "Dołącz" zostaje zamieniony w przycisk "Zrezygnuj".

3.4.9 Płatność za wydarzenie sportowe

Płatność za wydarzenie może nastąpić dopiero w momencie, kiedy zakończy się etap dołączania do wydarzenia sportowego. W tym momencie zapisany do wydarzenia użytkownik widzi w oknie wydarzenia przycisk "Zapłać". Po jego kliknięciu następuje sekwencja 2 procesów:

- Utworzenie płatności przez aplikację backendową
- Przekierowanie klienta do serwisu Paypal w celu zatwierdzenia płatności

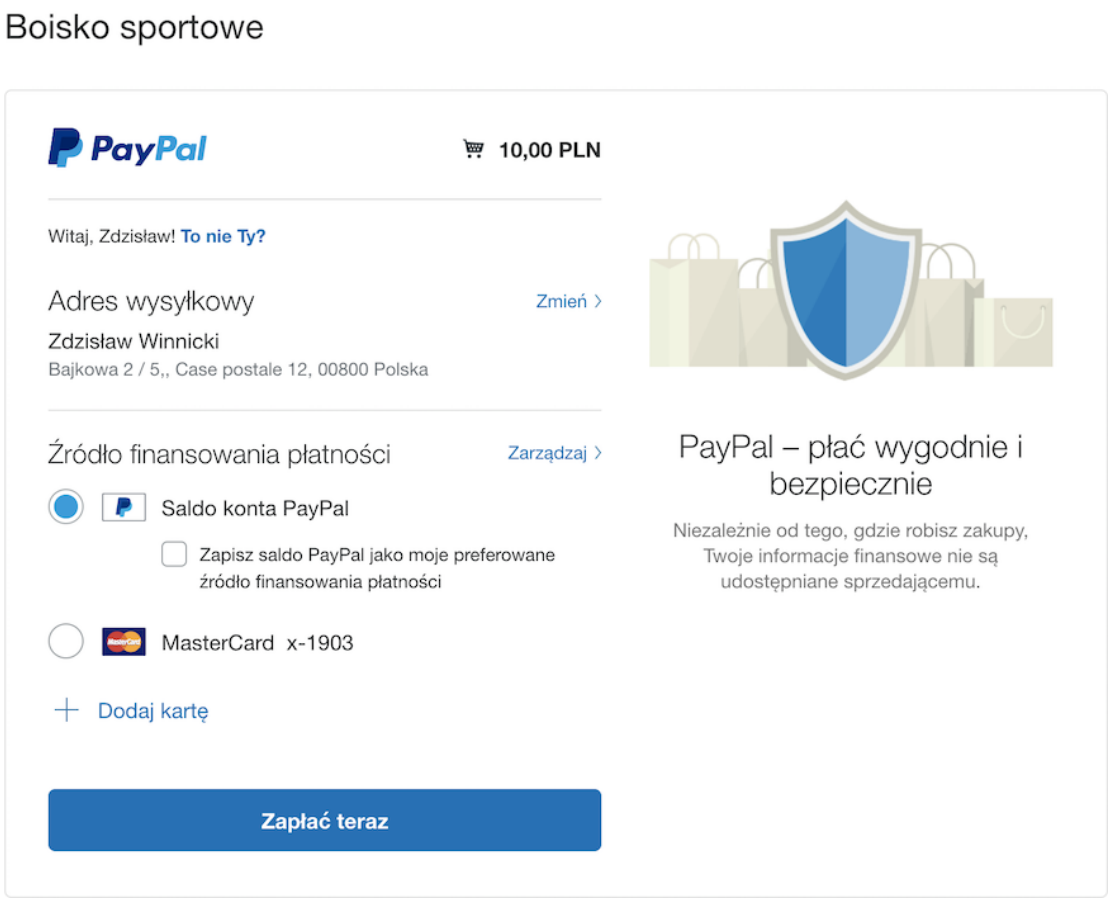
Utworzenie płatności przez aplikację backendową Pierwszym krokiem w celu wykonania płatności jest jej utworzenie przy użyciu API serwisu Paypal. W tym celu aplikacja backendowa pobiera z bazy danych informacje o kliencie oraz właścicielu kompleksów sportowych. Następnie wykonywane jest żądanie HTTP POST do API serwisu Paypal. Rysunek 20 prezentuje wszystkie informacje wysłane w żądaniu do serwisu Paypal.



Rysunek 20: Żądanie wysyłane do API serwisu Paypal

W odpowiedzi na to żądanie API serwisu Paypal zwraca m.in. adres, pod który należy przekierować klienta, aby zatwierdził on dokonanie płatności. Po otrzymaniu odpowiedzi z API serwisu Paypal aplikacja backendowa niezwłocznie przekierowuje klienta pod ten adres.

Przekierowanie klienta do serwisu Paypal Po przekierowaniu klient loguje się do serwisu Paypal. Po zalogowaniu widzi ekran z informacjami o planowanej płatności. Na ekranie tym znajdują się podstawowe informacje dotyczące płatności: kwota, imię i nazwisko klienta oraz nazwa obiektu sportowego, na którym odbywa się wydarzenie. Ilustracja 21 ukazuje ekran płatności.



Rysunek 21: Potwierdzenie płatności w serwisie Paypal. Źródło: `https://paypal.com`

Po dokonaniu płatności klient jest z powrotem przekierowywany do aplikacji frontendowej.

3.5 Testy

Testowanie umożliwia wykrycie błędów jeszcze przed uruchomieniem systemu na środowisku produkcyjnym. Pisanie testów do tworzonego oprogramowania pozwala wprowadzać zmiany w kodzie bez obawy, iż system przestanie działać.

3.5.1 Testy aplikacji backendowej

Przy tworzenie aplikacji backendowej autor korzystał z podejścia opartego na testach (*test driven development*). Najpierw napisano testy, które nie przechodziły. Następnie zaimplementowano funkcjonalności i sprawdzono, czy testy zakończyły się sukcesem. Pozwoliło to wyłapać błędy w zachowaniu implementowanych modułów.

3.5.2 Testy aplikacji frontendowej

Testowanie aplikacji frontendowej różni się od testowania aplikacji backendowej. W przypadku tej drugiej test ma na celu sprawdzenie czy funkcja bądź moduł zwróci prawidłowy wynik. W przypadku testowania aplikacji frontendowej należy przetestować strukturę

(HTML), wygląd (CSS) i zachowanie (JavaScript). Wszystkie te 3 elementy mogą być ze sobą powiązane. Przykładem jest następująca sytuacja: kliknięcie w przycisk wywołuje funkcję z JavaScriptu, która zmienia atrybut *class* pewnego elementu HTML. W takim wypadku konieczne jest zasymulowanie asynchronicznego działania - kliknięcia w przycisk.

3.6 Bezpieczeństwo

Bezpieczeństwo to ważny aspekt każdego systemu informatycznego. Dane użytkowników powinny być należycie chronione. Ich wyciek bądź nieuprawniona modyfikacja wiążą się ze stratą wizerunkową właściciela systemu, utratą zaufania klientów oraz możliwymi konsekwencjami prawnymi lub finansowymi.

Należy podkreślić, iż bezpieczeństwo to proces a nie stan. Nie da się napisać oprogramowania zabezpieczonego przed wszystkimi atakami, ponieważ ciągle powstają nowe. Z tego powodu konieczne jest stałe monitorowanie systemu w celu wykrycia nieprawidłowości.

3.6.1 Szyfrowanie komunikacji przy użyciu HTTPS

HTTPS to szyfrowana wersja protokołu HTTP. Do szyfrowania danych wykorzystuje się protokół SSL lub TLS. Większość obecnych przeglądarek uznaje HTTPS za standard, którego należy używać nawet jeśli dana strona internetowa nie przechowuje poufnych informacji użytkowników.

Użycie HTTPS zabezpiecza użytkowników przed atakami typu *man-in-the-middle*. Nawet jeśli atakujący przechwyci dane użytkownika to nie odszyfruje ich bez znajomości klucza prywatnego serwera.

W stworzonej aplikacji cały ruch sieciowy odbywa się z użyciem protokołu HTTPS.

3.6.2 Zabezpieczenie przed atakami typu XSS

XSS to skrót od *cross-site-scripting*. Atak ten polega na wstrzyknięciu do przeglądarki ofiary fragmentu javascriptu, który może być przez przeglądarkę uruchomiony.

Przykładem takiego ataku może być sytuacja, kiedy atakujący zamieszcza komentarz pod artykułem na stronie internetowej. Komentarz ten zawiera kod javascriptu, który wysyła wartość ciasteczka z sesją użytkownika do atakującego. Następnie komentarz odczytuje moderator. W tym momencie ma miejsce kradzież ciasteczka z sesją moderatora.

W celu ochrony przed atakami XSS należy zamieniać znaki w wyświetlanych danych z niebezpiecznych na bezpieczne. Rysunek 22 przedstawia w jaki sposób zamieniać niektóre z popularniejszych znaków.

```
1 & --> &amp;
2 < --> &lt;
3 > --> &gt;
4 " --> &quot;
5 ' --> &#x27;
6 / --> &#x2F;
```

Rysunek 22: Zamiana niebezpiecznych znaków na bezpieczne. Źródło: `https://sekurak.pl/czym-jest-xss/`

W stworzonej aplikacji frontendowej wszystkie wyświetlane znaki przechodzą przez klasę *DomSanitizer*, która zamienia niebezpieczne sekwencje znaków na bezpieczne.[18]

3.6.3 Zabezpieczenie przed atakami typu CSRF

CSRF to skrót od *cross site request forgery*. Atak ten polega na zmuszeniu przeglądarki ofiary do wykonania nieautoryzowanej przez użytkownika akcji. Atak nie jest wynikiem luki bezpieczeństwa w systemie, lecz związany jest z tym w jaki sposób funkcjonują przeglądarki internetowe oraz ciasteczka.

Przykładowy scenariusz takiego ataku wygląda następująco:

- Atakujący umieszcza na stronie evil.com tag img z atrybutem src, który wskazuje na adres powodujący wykonanie przelewu bankowego na stronie bank.com
- Ofiara loguje się na swoje konto w serwisie bankowości internetowej na stronie bank.com
- Atakujący skłania ofiarę do wejścia na stronę evil.com
- Ofiara wchodzi na stronę evil.com Przeglądarka wyświetla tag img co powoduje wykonanie żądania na stronie bank.com przy użyciu ciasteczka sesyjnego ofiary.

Jednym ze sposobów obrony przed tym atakiem jest użycie tokenów. Kiedy użytkownik loguje się do serwisu w odpowiedzi otrzymuje token. Token jest dodawany jako nagłówek HTTP, do każdego żądania, które zmienia zasoby na serwerze (POST, PUT, PATCH, DELETE). Token jest również trzymany w sesji użytkownika. W czasie przetwarzania żądania następuje sprawdzenie czy token otrzymany w żądaniu zgadza się z tokenem pochodzącym z sesji użytkownika. Jeśli nie to żądanie nie jest realizowane.

3.7 Problemy napotkane w czasie tworzenia aplikacji

3.7.1 Dokładność geolokalizacji

W przypadku przeglądarek na urządzeniach mobilnych zwracana lokalizacja z api geolokalizacyjnego jest zgodna z rzeczywistą lokalizacją użytkownika. Jest to zasługa modułu GPS obecnego na urządzeniach mobilnych.

Przeglądarki internetowe nie zwracają satysfakcjonujących rezultatów na komputerach. Otrzymywana lokalizacja jest bardzo często odległa od rzeczywistej o dziesiątki kilometrów.

API geolokalizacyjne umożliwia ustawienie flagi zwiększającej dokładność określania pozycji. Ustawienie tej flagi nie zwiększyło precyzji geolokalizacji na komputerach.

Powodem dużej niedokładności jest fakt, iż komputery nie są wyposażone w moduły GPS. Przeglądarki muszą zatem skorzystać z innej metody. Jej wybór leży wyłącznie w gestii twórców przeglądarek.[16] Często stosowaną metodą jest geolokalizacja oparta o adres IP.

Wbrew nazwie w przypadku takiej geolokalizacji nie jest wykorzystywany jedynie adres IP lecz także:

- Adres sieci
- Nazwa dostawcy internetowego
- Dane ze stron, które ten adres IP odwiedzał
- Dane o urządzeniach w tej sieci

W zależności od ilości i jakości tych danych pozwalają one zawęzić obszar do okręgu o promieniu kilku lub kilkudziesięciu kilometrów. Większą dokładność mogą oferować rozwiązania komercyjne.

3.7.2 Szybkość geolokalizacji

W czasie tworzenia aplikacji frontendowej problemem okazała się szybkość geolokalizacji. W niektórych przypadkach określenie pozycji zajmowało nawet 5 sekund. W aplikacji frontendowej lokalizacja klienta jest wykorzystywana do wyświetlania mapy obiektów sportowych. Do czasu określania lokalizacji mapa pozostaje niewidoczna. Długi czas oczekiwania na wyświetlenie mapy jest niekorzystny ze względu na doświadczenia użytkownika (*user experience*).

Rozwiązaniem tego problemu było określenie maksymalnej ilości czasu (*timeout*), po upływie której wywołana zostanie funkcja obsługi błędu. W tej funkcji następuje zwrócenie zapasowej lokalizacji ustawionej na środek Warszawy.

3.7.3 Testy dla aplikacji frontendowej

Napisanie odpowiedniej liczby testów dla aplikacji frontendowej okazało się problematyczne ze względu na zależności występujące między komponentami. W wielu sytuacjach aby przetestować dany komponent konieczne było wcześniejsze zainicjowanie komponentów, od których dany komponent zależy.

Rozwiązaniem tej sytuacji mogłoby być zastosowanie makiet czyli komponentów o interfejsie identycznym z prawdziwymi komponentami lecz o dużo prostszej implementacji.

Scenariusz zastosowania takiego rozwiązania: należy przetestować komponent, który zależy od modułu pobierającego dane z API. Zamiast inicjować moduł pobierający dane i przygotowywać API należy przygotować makietę, która zamiast wywoływać API po prostu zwróci pewien przykładowy zestaw danych. Format tych danych musi być zgodny z formatem danych zwracanych z API. Natomiast nie ma potrzeby wywoływania API - te dane mogą zostać zapisane "na sztywno" w makiecie.

4 Zakończenie

4.1 Możliwości rozwoju

4.1.1 Open API

Aby w pełni korzystać z aplikacji użytkownik musi posiadać konto w serwisie Paypal. Jeśli użytkownik nie posiadał wcześniej konta w tym serwisie jest zmuszony je założyć. Konieczność tworzenia dodatkowego konta, aby skorzystać z jednej aplikacji jest niedogodnością.

Z punktu widzenia wygody najlepszym rozwiązaniem byłoby użycie konta bankowego - 85% Polaków posiada konto w banku[14]. Obecnie żaden bank w Polsce nie udostępnia publicznie API, które pozwalałoby na wykonywanie przelewów. Jednakże w listopadzie 2015 roku Parlament Europejski uchwalił dyrektywę regulującą rynek płatności - *Payemt Service Directive 2*, w skrócie PSD2. Wprowadza ona instytucje określaną jako TTP (*Trusted Third Party*), która może dokonywać płatności na rzecz właściciela konta bankowego.

Uzyskanie statusu TTP przez firmę korzystającą ze stworzonej w tej pracy aplikacji pozwoliłoby porzucić Paypala na rzecz bezpośredniego dostępu do rachunku bankowego.

4.1.2 Komercyjne wykorzystanie

W obecnym modelu działania aplikacja nie generuje żadnych zysków dla jej właściciela. Jednym z rozwiązań tego problemu byłoby dodanie prowizji od każdej rezerwacji. Należałoby rozważyć, czy prowizja powinna być stała czy też powinna zależeć od kwoty rezerwacji.

Z punktu widzenia klienta stała prowizja oznacza przewidywalne ceny niezależnie od czynników takich jak popularności sportu, ilość osób w wydarzeniu czy też pora rezerwacji. Z punktu widzenia właściciela aplikacji stała prowizja oznacza prawdopodobnie mniejszy zysk, lecz potencjalni użytkownicy mogą postrzegać taki model jako bardziej sprawiedliwy co mogłoby pozytywnie wpłynąć na popularność aplikacji.

4.1.3 Wykorzystanie parametrów wyszukiwania

Na podstawie parametrów wyszukiwania wprowadzonych przez klientów możliwe jest odkrycie takich informacji jak:

- popularność sportów wśród ogółu użytkowników
- popularność sportów wśród poszczególnych użytkowników
- zainteresowanie poszczególnymi lokalizacjami

Na podstawie tych informacji można stworzyć system, który sugeruje klientom te obiekty sportowe i wydarzenia, które pozwolą im na uprawianie pożądanych dyscyplin sporto-

wych. Dodatkowo informacje o zainteresowaniu lokalizacjami można udostępnić właścicielom kompleksów sportowych - uzyskaliby oni informacje jakie lokalizacje są najczęściej podawane w formularzu wyszukiwania i gdzie warto rozważyć utworzenie nowego obiektu.

4.1.4 Aplikacja na urządzenia mobilne

Stworzone w ramach aplikacji API można wykorzystać do stworzenie aplikacji na platformy mobilne (iOS i Android). Pozwoliłoby to dotrzeć do większej ilości użytkowników.

4.1.5 System rekomendacji oraz administrowanie wydarzeniem

Płatność za wydarzenie sportowe nie musi oznaczać, iż klient zjawi się danego dnia na wskazanym obiekcie. Wprowadzenie systemu reputacji pozwoliłoby wykryć, którzy klienci są rzetelni.

Wprowadzenie do aplikacji możliwości usuwania nierzetelnych użytkowników przez twórców tych wydarzeń zwiększyłoby atrakcyjność aplikacji. Jednakże w przypadku takiej funkcjonalności należałoby pomyśleć nad machanizmem obrony przed nadużyciami - twórca nie powinien mieć absolutnej kontroli nad wydarzeniem.

Bibliografia

[1] Spis warszawskich orlików z informacjami kontaktowymi
`http://sportowa.warszawa.pl/obiekty-miejskie/Orliki/`
(data dostępu: 04.01.2019)

[2] Opis grupy w serwisie Facebook
`https://www.facebook.com/notes/facebook/facebook-tips-whats-the-difference-between-a-facebook-page-and-group/324706977130/`
(data dostępu: 04.01.2019)

[3] TypeScript
`https://www.typescriptlang.org/`
(data dostępu: 04.01.2019)

[4] Dokumentacja TypeScriptu.
Z menu bocznego należy wybrać *Handbook*
`https://www.typescriptlang.org/docs/home.html`
(data dostępu: 04.01.2019)

[5] TypeScript Playground
`https://www.typescriptlang.org/play/index.html`
(data dostępu: 04.01.2019)

[6] Angular
`https://angular.io/`
(data dostępu: 04.01.2019)

[7] Angular CLI
`https://cli.angular.io/`
(data dostępu: 04.01.2019)

[8] Najpopularniejsze architektury JavaScriptowe do tworzenia SPA
`https://www.npmtrends.com/react-vs-vue-vs-@angular/core`
(data dostępu: 04.01.2019)

[9] Elixir
`https://elixir-lang.org/`
(data dostępu: 04.01.2019)

[10] Plug
`https://hexdocs.pm/plug/readme.html`
(data dostępu: 04.01.2019)

[11] Liczba użytkowników serwisów społecznościowych
`https://www.slideshare.net/wearesocial/digital-in-2018-global-overview-86860338`

Slajd 59
(data dostępu: 04.01.2019)

[12] API Facebooka
https://developers.facebook.com/docs/graph-api/
(data dostępu: 04.01.2019)

[13] Formularz zgody na dostęp do danych użytkownika
https://aaronparecki.com/oauth-2-simplified/oauth-authorization-prompt.png
(data dostępu: 04.01.2019)

[14] Ilosć Polaków posiadających konto w banku
https://www.bankier.pl/wiadomosc/85-proc-Polakow-ma-konto-w-banku-7410187.html
(data dostępu: 09.01.2019)

[15] Pisanie dokumentacji w języku Elixir
https://github.com/elixir-lang/elixir/blob/master/lib/elixir/pages/Writing%20Documentation.md
(data dostępu: 09.01.2019)

[16] Specyfikacja dotycząca geolokalizacji w przeglądarkach
https://www.w3.org/TR/geolocation-API/#geolocation_interface
(data dostępu: 10.01.2019)

[17] Docker
https://www.docker.com/
(data dostępu: 10.01.2019)

[18] Klasa Angulara do obsługi znaków niebezpiecznych - DomSanitizer
https://angular.io/api/platform-browser/DomSanitizer#description
(data dostępu: 10.01.2019)

[19] RxJS
https://rxjs-dev.firebaseapp.com/
(data dostępu: 23.01.2019)

[20] Phoenix
https://phoenixframework.org/
(data dostępu: 23.01.2019)

Spis rysunków