Dokumentacja projekt zaliczeniowy Programowanie III

Jan Bartnicki (311491), Jakub Piotrowski (311000)

Politechnika Śląska

Wydział Matematyki Stosowanej, Informatyka, semestr III 2024/25

1. Wstęp

a. Opis projektu

Projekt Free-Events to aplikacja webowa skierowana do ogółu społeczeństwa, umożliwiająca tworzenie i zapisywanie się na różnego rodzaju wydarzenia. Celem aplikacji jest ułatwienie organizacji darmowych eventów oraz zapewnienie użytkownikom wygodnej platformy do ich przeglądania i rejestracji.

Projekt czerpie inspirację z istniejących rozwiązań, takich jak Going, jednak wyróżnia się tym, że wszystkie wydarzenia są całkowicie darmowe.

b. Cel aplikacji

Głównym celem aplikacji jest stworzenie intuicyjnego narzędzia, które pozwala użytkownikom na:

Tworzenie wydarzeń – organizatorzy mogą dodawać nowe eventy, określając ich nazwę, opis, datę i lokalizację.
Zapisywanie się na wydarzenia – użytkownicy mogą przeglądać dostępne wydarzenia i rejestrować się na te, które ich interesują.
Zarządzanie wydarzeniami – organizatorzy mogą edytować i usuwać swoje wydarzenia.

Aplikacja obsługuje dwa główne typy użytkowników:

- Organizator – użytkownik, który może tworzyć wydarzenia.
- Uczestnik – użytkownik, który może przeglądać wydarzenia i zapisywać się na nie.

c. Zakres funkcjonalności

- rejestracja i logowanie użytkowników
- tworzenie, edycja i usuwanie wydarzeń przez organizatorów
- przeglądanie listy wydarzeń
- zapisywanie się na wydarzenia
- wybór dokładnej lokalizacji wydarzenia na mapie (Google Maps)
- generowanie plików PDF w formie biletów dla zapisanych użytkowników

d. Przyszłe rozszerzenia

- mapa z zaznaczonymi wszystkimi dostępnymi wydarzeniami

- powiadomienia e-mail dla użytkowników (np. potwierdzenie zapisu, przypomnienie o wydarzeniu)

2. Opis technologii

a. Frontend

React – biblioteka do budowy interfejsu użytkownika
TypeScript - nadzbiór języka JavaScript, który dodaje do niego statyczne typowanie
Tailwind CSS + DaisyUI – stylowanie i gotowe komponenty UI

Dodatkowe biblioteki:

- @react-google-maps/api – integracja Google Maps
- axios – komunikacja z backendem poprzez REST API
- date-fns – obsługa i formatowanie dat
- react-datepicker – wygodny wybór daty dla użytkowników
- lucide-react, react-icons – zestaw ikon dla interfejsu użytkownika

Aplikacja frontendowa komunikuje się z backendem poprzez REST API.

b. Backend

Backend został zbudowany w języku Java z wykorzystaniem Spring Boot 3.3.5.

Główne technologie backendowe:

- Spring Boot – framework do budowy aplikacji webowych
- Spring Web – obsługa REST API
- Spring Security – autoryzacja i uwierzytelnianie użytkowników
- MongoDB – baza danych dokumentowa
- Firebase Admin SDK – obsługa autoryzacji i przechowywania plików
- Google Cloud Storage – przechowywanie plików w chmurze

Dodatkowe biblioteki:

- Apache PDFBox – generowanie plików PDF (bilety na wydarzenia)
- ZXing – generowanie kodów QR dla wydarzeń
- Thymeleaf – silnik szablonów dla e-maili lub raportów

Backend pełni rolę serwera API, obsługującego żądania HTTP z frontendu oraz generującego bilety w formacie PDF.

c. Baza danych

Aplikacja korzysta z MongoDB jako systemu bazodanowego.

- MongoDB - nierelacyjna baza danych typu NoSQL, która przechowuje dane w elastycznym formacie JSON-owym

- Brak migracji schematu – nie korzystano z narzędzi do migracji, takich jak Flyway czy Liquibase

d. Inne narzędzia

- Postman – testowanie zapytań do REST API
- GitHub – system kontroli wersji, repozytorium kodu
- Brak Docker i CI/CD – aplikacja nie korzysta z konteneryzacji ani zautomatyzowanych wdrożeń

3. Opis funkcjonalności

a. Tworzenie wydarzeń

Każdy zarejestrowany użytkownik ma możliwość stworzenia nowego wydarzenia, co automatycznie nadaje mu rolę organizatora tego wydarzenia. Podczas tworzenia wydarzenia użytkownik musi podać następujące informacje:

- Nazwa wydarzenia
- Opis wydarzenia
- Data i godzina
- Zdjęcie wydarzenia

Po stworzeniu wydarzenia, użytkownik staje się organizatorem tego wydarzenia i może zarządzać nim (np. edytować szczegóły).

b. Zapisywanie się na wydarzenia

Każdy zalogowany użytkownik może przeglądać dostępne wydarzenia i zapisywać się na te, które go interesują. Po zapisaniu się na wydarzenie, użytkownik może pobrać swój bilet. Generowanie biletu PDF: Po zapisaniu się użytkownik może pobrać plik PDF z wygenerowanym kodem QR. Bilet jest dostępny w zakładce "Moje bilety".

c. Logowanie i rejestracja użytkowników

Użytkownicy mogą rejestrować się i logować przy użyciu adresu e-mail i hasła. Rejestracja nie wymaga dodatkowego potwierdzenia (np. e-mailem).

4. Opis architektury

a. Diagram architektury systemu

Aplikacja jest podzielona na trzy główne warstwy: frontend, backend i baza danych.

- Frontend: Zbudowany przy użyciu React. Frontend komunikuje się z backendem za pomocą REST API. Użytkownicy mogą przeglądać wydarzenia, zapisywać się na nie oraz zarządzać swoimi biletami.
- Backend: Zbudowany przy użyciu Spring Boot. Backend odpowiada za logikę biznesową aplikacji, zarządzanie danymi (MongoDB) oraz obsługę zapytań API.

b. Struktura folderów

Struktura folderów frontendowych jest podzielona na kilka głównych sekcji, co umożliwia łatwiejsze zarządzanie aplikacją:

- src: Główna lokalizacja kodu źródłowego.
  - assets: Zawiera zasoby statyczne, takie jak obrazy i pliki CSS.
  - components: Zawiera komponenty UI, które są wykorzystywane w różnych częściach aplikacji.
  - contexts: Zawiera konteksty React, takie jak AlertContext i UserContext, które zarządzają globalnym stanem aplikacji.
  - pages: Zawiera komponenty reprezentujące strony aplikacji, np. stronę logowania, stronę szczegółów wydarzenia itp.
  - types: Zawiera definicje typów i interfejsów.

Backend składa się z następujących folderów i ich zawartości:

- config: Zawiera konfigurację aplikacji, np. konfiguracje bezpieczeństwa, połączeń z bazą danych itp.
- controller: Zawiera kontrolery, które obsługują zapytania HTTP i przekazują je do odpowiednich serwisów.
- model: Zawiera modele danych, które odpowiadają strukturom danych przechowywanych w bazie (np. użytkownicy, wydarzenia).
- repository: Zawiera repozytoria, które odpowiadają za interakcję z bazą danych i wykonywanie operacji CRUD.
- service: Zawiera logikę biznesową aplikacji, która przetwarza dane i komunikuję się z repozytoriami.

Wszystkie dane aplikacji przechowywane są w MongoDB, która jest bazą danych dokumentową, doskonale nadającą się do przechowywania danych o zmiennej strukturze (np. użytkownicy, wydarzenia, zapisy).

c. Wzorce projektowe

Chociaż aplikacja backendowa nie korzysta z rozbudowanych wzorców projektowych jak MVC (Model-View-Controller) lub Repository Pattern w pełnej formie, zauważalny jest podział na warstwy:

- Controller – odpowiedzialny za obsługę zapytań HTTP i przekazywanie ich do odpowiednich serwisów.
- Service – logika biznesowa, która zarządza procesami w aplikacji.
- Repository – warstwa odpowiedzialna za interakcję z bazą danych, obsługując operacje CRUD.

W frontendzie aplikacja korzysta z dwóch contextów:

- AlertContext – zarządza stanem powiadomień w aplikacji.
- UserContext – zarządza stanem zalogowanego użytkownika, jego danymi i sesją.

Komponenty są zorganizowane w sposób, który umożliwia ich wielokrotne użycie w różnych częściach aplikacji (np. w components), a stan globalny jest zarządzany przy użyciu useContext, co pozwala na efektywne przekazywanie danych między komponentami bez konieczności używania zewnętrznych bibliotek takich jak Redux.

5. API
    a. Lista dostępnych endpointów

        a) POST /api/events/create

        Opis: Tworzenie nowego wydarzenia.

        Parametry:

        - event (w formacie JSON): Nazwa, opis, lokalizacja, data wydarzenia, ID organizatora.
        - image (plik): Obrazek do wydarzenia.

        Przykład zapytania:

```
POST /api/events/create
Content-Type: multipart/form-data

{
  "event": {
    "name": "Event Name",
    "description": "Event Description",
    "location": "Event Location",
    "eventDate": "2025-02-10T10:00:00",
    "organizerId": "12345"
  },
  "image": [FILE]
}
```

        Przykład odpowiedzi:

```
{
  "id": "eventId",
  "name": "Event Name",
  "description": "Event Description",
  "location": "Event Location",
  "eventDate": "2025-02-10T10:00:00",
  "organizerId": "12345",
  "imageUrl": "https://path_to_image.com",
  "participants": []
}
```

        b) GET /api/events/all

        Opis: Pobieranie wszystkich wydarzeń.

        Parametry: Brak.

        Przykład zapytania:

```
GET /api/events/all
```

Przykład odpowiedzi:

```
[
  {
    "id": "eventId",
    "name": "Event Name",
    "description": "Event Description",
    "location": "Event Location",
    "eventDate": "2025-02-10T10:00:00",
    "organizerId": "12345",
    "imageUrl": "https://path_to_image.com",
    "participants": []
  }
]
```

c) GET /api/events/nearest

Opis: Pobieranie najbliższego nadchodzącego wydarzenia.

Parametry: Brak.

Przykład zapytania:

```
GET /api/events/nearest
```

Przykład odpowiedzi:

```
{
  "id": "eventId",
  "name": "Event Name",
  "description": "Event Description",
  "location": "Event Location",
  "eventDate": "2025-02-10T10:00:00",
  "organizerId": "12345",
  "imageUrl": "https://path_to_image.com",
  "participants": []
}
```

d) GET /api/events/by-organizer

Opis: Pobieranie wydarzeń według ID organizatora.

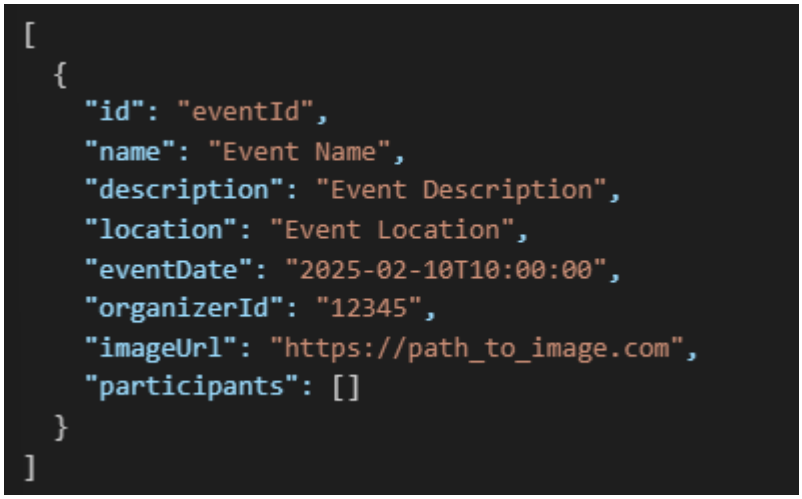
Parametry:

- organizerId (query): ID organizatora.

Przykład zapytania:

```
GET /api/events/by-organizer?organizerId=12345
```

Przykład odpowiedzi:



e) GET /api/events/search

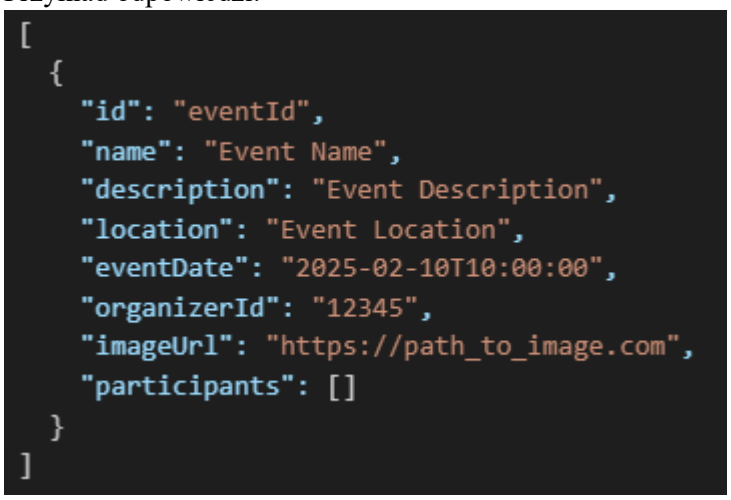
Opis: Wyszukiwanie wydarzeń po nazwie.

Parametry:

- name (query): Nazwa wydarzenia.

Przykład zapytania:

Przykład odpowiedzi:



f) POST /api/events/{eventId}/register

Opis: Rejestracja użytkownika na wydarzenie.

Parametry:

- eventId (path): ID wydarzenia.
- userId (query): ID użytkownika.

Przykład zapytania:

```
POST /api/events/{eventId}/register?userId=12345
```

Przykład odpowiedzi:

```
"User successfully registered to the event."
```

g) GET /api/tickets/{eventId}/download

Opis: Pobieranie biletu PDF na wydarzenie.

Parametry:

eventId (path): ID wydarzenia.
attendeeName (query): Imię uczestnika.
ticketNumber (query): Numer biletu.

Przykład zapytania:

```
GET /api/tickets/{eventId}/download?attendeeName=JohnDoe&ti
```

Przykładowa odpowiedź: Plik PDF zawierający bilet z kodem QR.

h) POST /api/users/register

Opis: Rejestracja nowego użytkownika.

Parametry:
eventId (path): ID wydarzenia.
userId (query): ID użytkownika.

Przykład zapytania:

```
POST /api/events/{eventId}/register?userId=12345
```

Przykład odpowiedzi:

```
"User successfully registered to the event."
```

j) POST /api/users/login

Opis: Logowanie użytkownika.

Parametry:
user (JSON): Obiekt logowania (email, hasło).

Przykład zapytania:

```
POST /api/users/login
Content-Type: application/json

{
  "email": "user@example.com",
  "password": "securepassword"
}
```

Przykład odpowiedzi:

```
{
  "id": "userId",
  "email": "user@example.com",
  "role": "user"
}
```

6. Opis interfejsu użytkownika

   1. Strona główna

   Opis: Strona główna zachęca użytkowników do wejścia w wydarzenia i dołączenia do nich. Jest prosta, przejrzysta i zawiera odnośniki do najważniejszych sekcji aplikacji. Użytkownicy mogą od razu przejść do wyszukiwania wydarzeń.

   Tryby interfejsu:

   Light mode – standardowy tryb, jasny kolor tła, ciemny tekst.

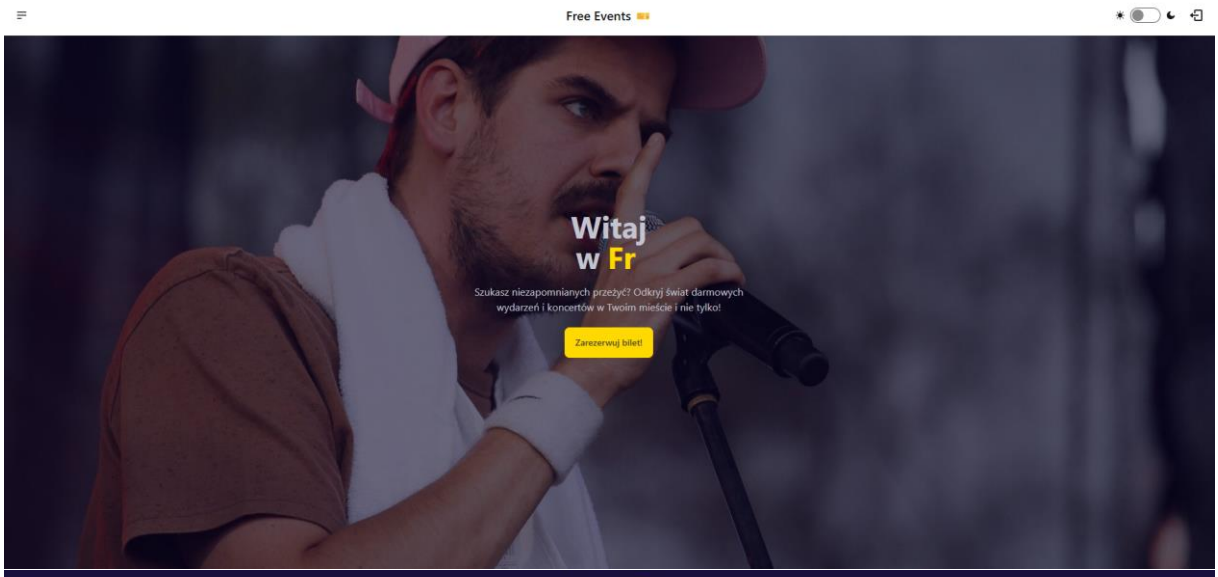
   Dark mode – ciemne tło z jasnym tekstem, przeznaczone dla osób preferujących ciemniejsze ustawienia interfejsu.

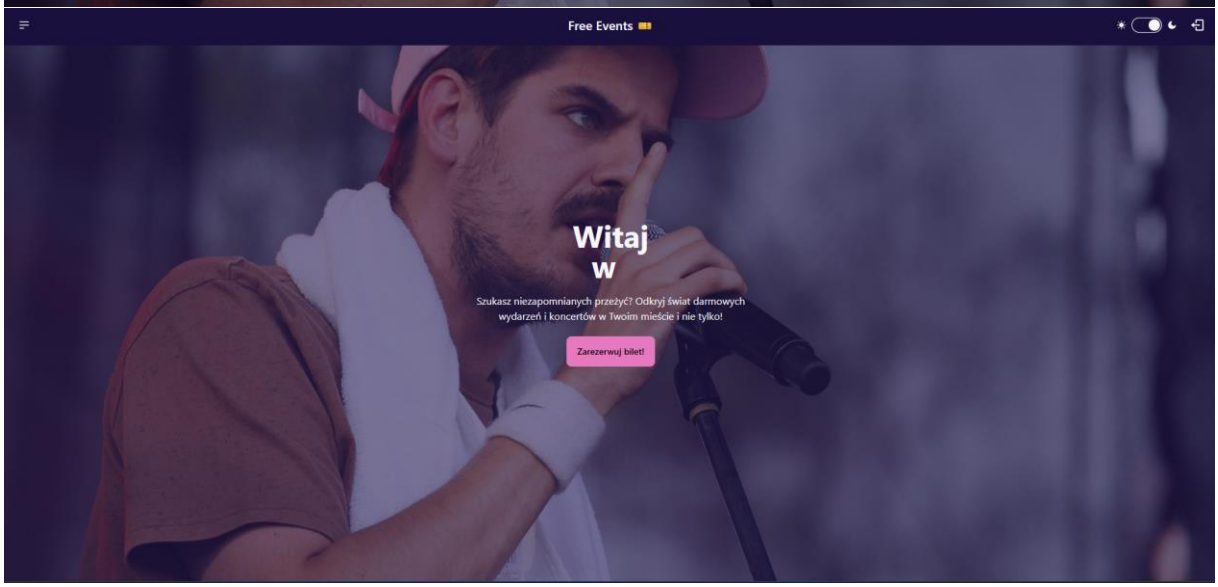
   Akcje użytkownika:

   Możliwość zmiany trybu interfejsu między Light i Dark Mode.

   Przejście do strony wydarzeń, gdzie użytkownicy mogą przeglądać nadchodzące wydarzenia.

   Możliwość rozpoczęcia rejestracji lub logowania.





2. Strona logowania/rejestracji

Opis: Strona logowania i rejestracji umożliwia użytkownikowi szybki dostęp do swojego konta. Można się na niej zalogować lub zarejestrować nowe konto.

Akcje użytkownika:

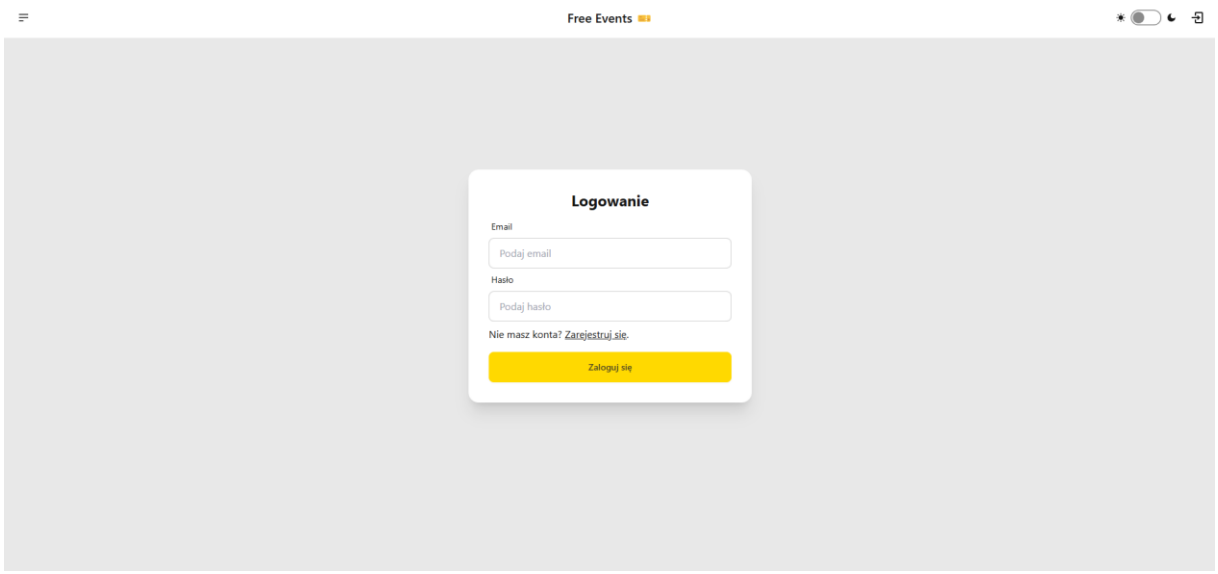
Logowanie: Użytkownik wprowadza e-mail i hasło, aby się zalogować.

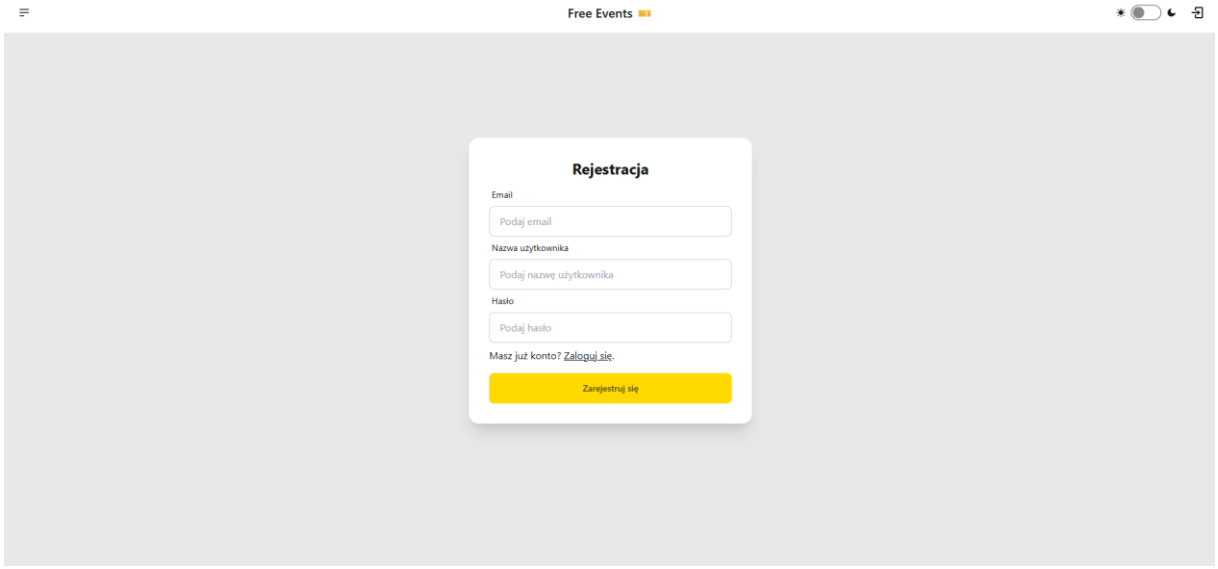
Rejestracja: Użytkownik wprowadza dane takie jak imię, e-mail, hasło, aby utworzyć nowe konto.

Informacje po akcji:

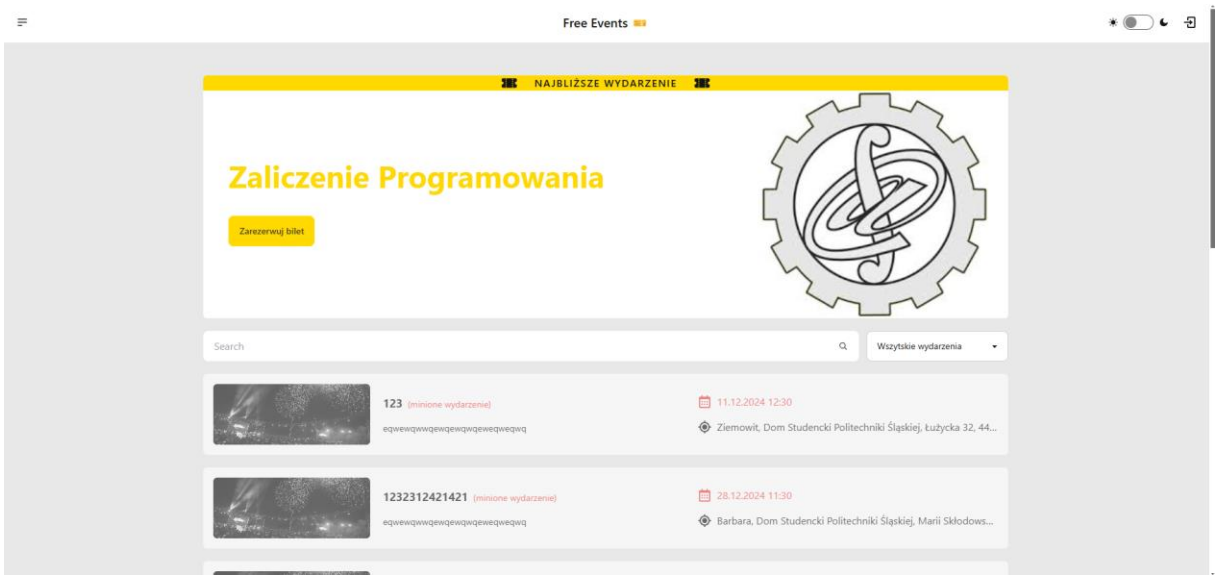
Po udanym zalogowaniu użytkownik zostaje przekierowany do strony głównej lub jego profilu.

Po udanej rejestracji użytkownik otrzymuje powiadomienie o powodzeniu operacji, a następnie jest przekierowywany do strony logowania.





3. Strona wydarzeń



Opis: Strona z listą wydarzeń, gdzie użytkownicy mogą filtrować wydarzenia według kategorii, daty, miejsca i innych kryteriów.

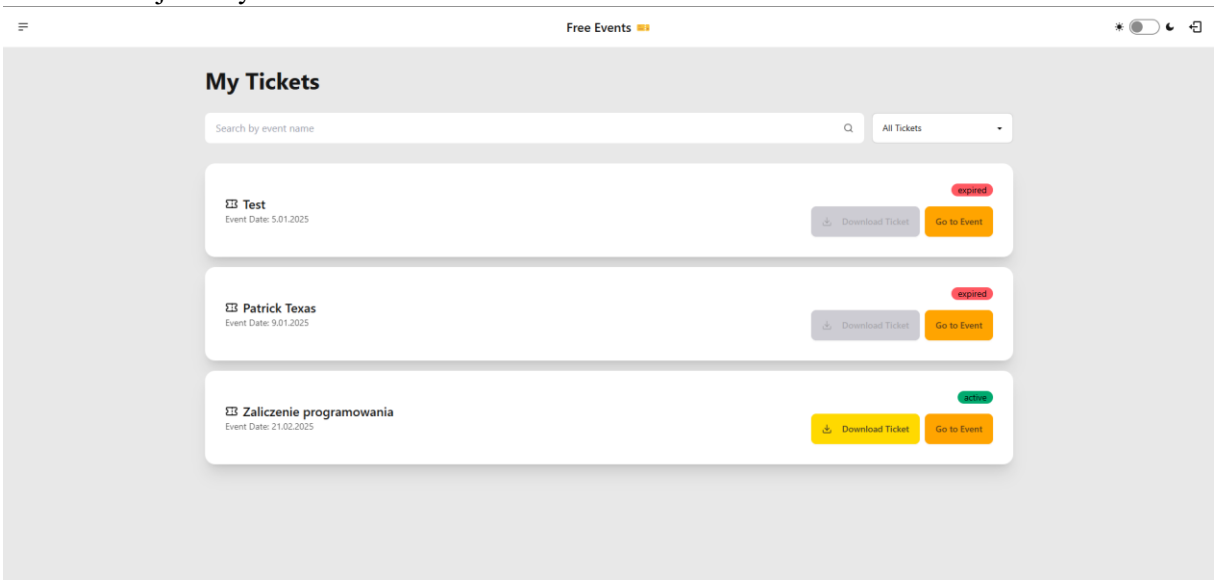
Akcje użytkownika:

Wyszukiwanie wydarzeń: Użytkownik wpisuje nazwę wydarzenia w wyszukiwarce, aby znaleźć konkretne wydarzenie.

Filtrowanie: Użytkownik może filtrować wydarzenia według daty (nadchodzące, przeszłe, dzisiejsze).

Przykład interakcji: Po wyszukaniu wyników użytkownik widzi listę wydarzeń pasujących do zapytania i może kliknąć na konkretne wydarzenie, aby zobaczyć szczegóły.

4. Strona Moje bilety



Opis: Strona umożliwia użytkownikom przeglądanie wydarzeń, na które się zapisali, oraz generowanie biletów PDF.

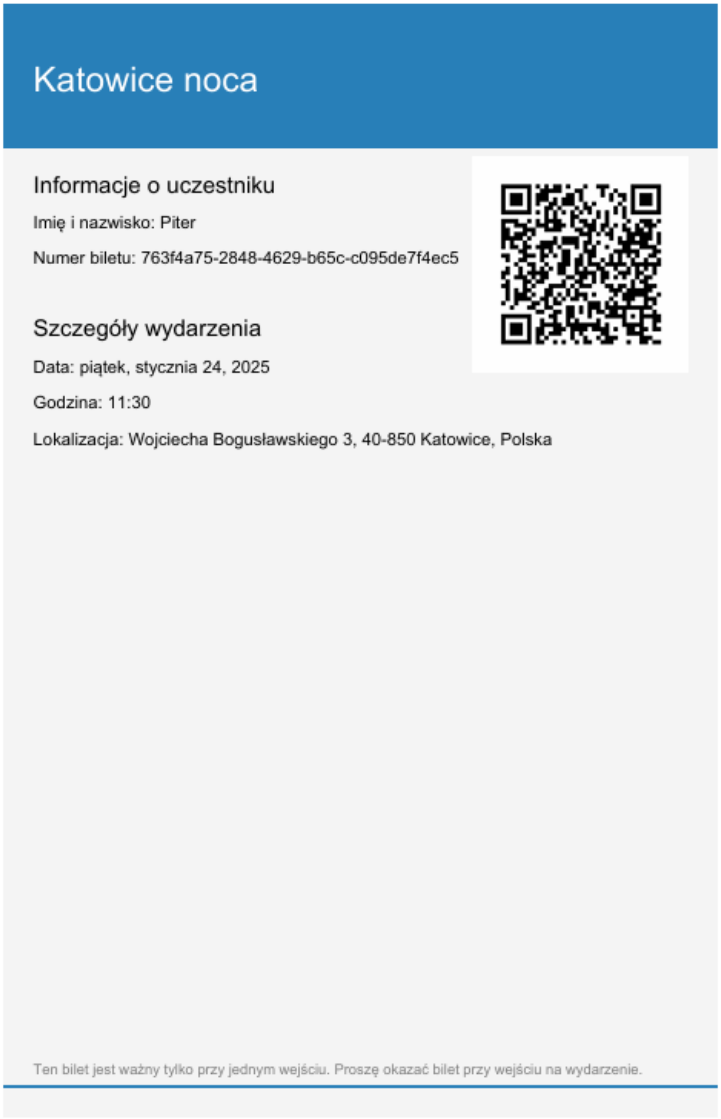
Akcje użytkownika:

Użytkownik może kliknąć na wydarzenie, aby pobrać bilet PDF.

Bilety zawierają unikalny kod QR.

Funkcje:

Generowanie biletu PDF: Po kliknięciu w bilet, plik PDF z kodem QR jest pobierany przez użytkownika.



5. Strona tworzenia wydarzenia

Opis: Strona dla organizatorów, gdzie mogą dodać wydarzenie, wybierając miejsce z mapy, ustalając datę, godzinę i dodając zdjęcie.

Akcje użytkownika:

Wybór lokalizacji: Użytkownik wybiera miejsce za pomocą mapy lub wyszukuje miejsce.

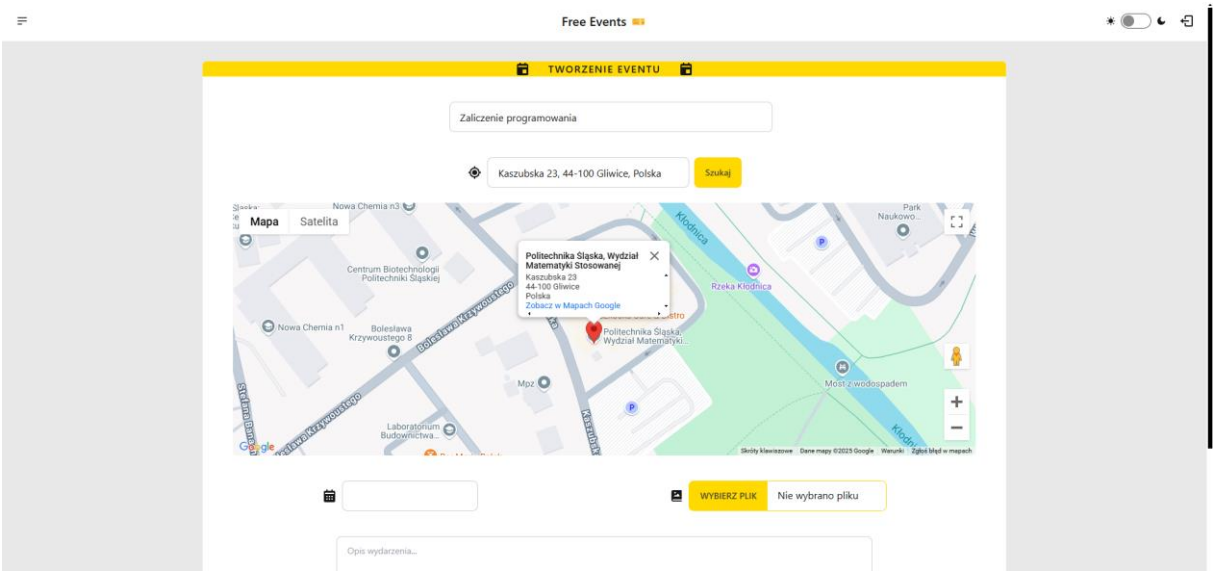
Ustawienie godziny: Wprowadzenie daty i godziny wydarzenia.

Dodanie zdjęcia: Możliwość dodania zdjęcia do wydarzenia.

Funkcje:

Możliwość podglądu wydarzenia przed zapisaniem.

Użytkownik może kliknąć „Stwórz wydarzenie”, aby opublikować je na platformie.



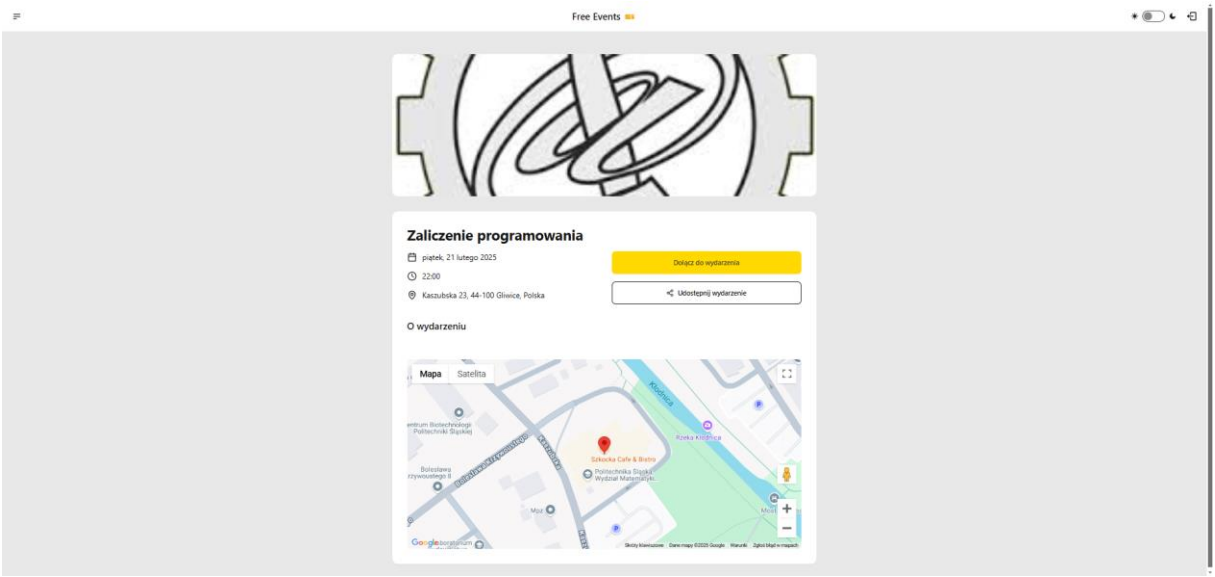
6. Strona wydarzenia

Opis: Strona szczegółów wydarzenia, gdzie użytkownicy mogą zapisać się na wydarzenie, zobaczyć jego lokalizację oraz szczegóły.

Akcje użytkownika:

Zapisanie się na wydarzenie: Użytkownik klikając „Zapisz się na wydarzenie”, dołącza do niego.

Podgląd lokalizacji: Użytkownik może zobaczyć na mapie, gdzie odbywa się wydarzenie.



Informacje wyświetlane na stronie:

Nazwa wydarzenia, data, godzina, lokalizacja, organizator, opis wydarzenia.

Przykładowy Flow użytkownika:

Rejestracja użytkownika:

Użytkownik przechodzi do strony rejestracji, wypełnia formularz i klika „Zarejestruj się”.

Po pomyślnej rejestracji użytkownik jest automatycznie zalogowany i przekierowany na stronę główną.

Zapisanie się na wydarzenie:

Użytkownik przegląda listę wydarzeń.

Znajduje interesujące wydarzenie, klika w nie, aby przejść do szczegółów.

Użytkownik klika „Zapisz się na wydarzenie" i zostaje zapisany.

Po zapisaniu użytkownik może pobrać bilet PDF.

Potwierdzenie:

Po zapisaniu się na wydarzenie użytkownik otrzymuje potwierdzenie (np. „Zarejestrowano na wydarzenie").

Na stronie „Moje bilety" użytkownik widzi zapisane wydarzenia oraz może pobrać bilety.

7. Instrukcja uruchomienia

1. Przygotowanie Backend

Krok 1: Pobranie plików backendu

- Sklonuj repozytorium, które zawiera zarówno frontend, jak i backend:
  - git clone https://github.com/janekbartnicki/free-events.git
  - cd free-events

Krok 2: Instalacja zależności backendu

- Przejdź do katalogu backendu i zainstaluj wymagane zależności przy użyciu Maven:
  - cd backend
  - mvn clean install

Krok 3: Konfiguracja plików backendu

- Skonfiguruj połączenie z bazą danych MongoDB i Firebase:
  - MongoDB: W pliku `src/main/resources/application.properties` lub `application.yml` musisz dodać odpowiednią konfigurację połączenia z bazą MongoDB.
  - Przykład konfiguracji MongoDB w `application.properties`: spring.data.mongodb.uri=mongodb://<username>:<password>@<host>:<port>/<database>
  - **Firebase**: Umieść plik konfiguracyjny Firebase (np. `firebase-config.json`) w folderze `src/main/resources` w projekcie backendu. Skontaktuj się z osobą odpowiedzialną za projekt, aby uzyskać plik konfiguracyjny.

Krok 4: Uruchomienie backendu

- Po zainstalowaniu zależności uruchom aplikację backendową:
    - mvn spring-boot:run

Backend będzie dostępny pod adresem http://localhost:8080.

2. Przygotowanie Frontend

Krok 1: Pobranie plików frontendowych

- Przejdź do katalogu frontend:
    - cd ../frontend

Krok 2: Instalacja zależności frontendowych

- Zainstaluj wymagane zależności frontendowe przy użyciu npm:
    - npm install

Krok 3: Uruchomienie aplikacji frontendowej

- Uruchom frontend:
    - npm run dev

Aplikacja frontendowa powinna być dostępna pod adresem http://localhost:3000.

3. Testowanie połączenia Backend-Frontend

Po uruchomieniu backendu i frontendowej aplikacji, upewnij się, że komunikacja między nimi działa poprawnie.

Sprawdzenie, czy backend działa:

Otwórz http://localhost:8080/api/events/all w przeglądarkę lub użyj Postmana, aby sprawdzić, czy API backendowe zwraca dane wydarzeń.

Sprawdzenie, czy frontend działa:

Otwórz http://localhost:3000 i sprawdź, czy aplikacja frontendowa prawidłowo komunikuje się z backendem (np. wyświetla wydarzenia).

4. Dodatkowe informacje

- Pamiętaj, aby zawsze mieć odpowiednie pliki konfiguracyjne do Firebase i MongoDB, które są niezbędne do poprawnej pracy aplikacji.
- Jeśli napotkasz jakiekolwiek problemy związane z konfiguracją bazy danych lub Firebase, upewnij się, że odpowiednie dane zostały poprawnie wprowadzone w plikach `application.properties` lub `application.yml` (dla backendu).
- Jeśli frontend działa na porcie `3000`, a backend na porcie `8080`, upewnij się, że komunikacja między nimi jest poprawnie skonfigurowana (np. CORS).
-

8 Podsumowanie realizacji projektu

Projekt systemu do zarządzania wydarzeniami, w którym użytkownicy mogą tworzyć, przeglądać i zapisywać się na wydarzenia, a także generować bilety w formacie PDF, został zrealizowany z sukcesem. Aplikacja składa się z dwóch głównych komponentów: frontend i backend, które komunikują się ze sobą za pomocą API RESTful. Backend obsługuje logikę biznesową oraz połączenie z bazą danych MongoDB i Firebase do przechowywania danych, a frontend zapewnia użytkownikom przyjazny interfejs, umożliwiający korzystanie z funkcji aplikacji.

Projekt obejmował następujące kluczowe funkcjonalności:

- **Tworzenie wydarzeń** przez użytkowników, którzy automatycznie stają się organizatorami.
- **Rejestracja użytkowników** i logowanie za pomocą e-maila i hasła.
- **Generowanie biletów PDF** z unikalnymi kodami QR po zapisaniu się na wydarzenie.
- **Filtracja i wyszukiwanie wydarzeń** według daty, lokalizacji i innych kryteriów.
- **Interfejs użytkownika** z obsługą trybu ciemnego i jasnego oraz mapą do dodawania lokalizacji wydarzenia.

Projekt został zrealizowany przy użyciu nowoczesnych technologii, takich jak:

- **Frontend**: React, JSX, CSS, Axios
- **Backend**: Java, Spring Boot, MongoDB, Firebase
- **PDF Generation**: Generowanie biletów PDF przy użyciu odpowiednich bibliotek
- **Komunikacja**: REST API

Aplikacja została przetestowana zarówno lokalnie, jak i w środowisku developerskim, z pełnym wdrożeniem mechanizmów rejestracji, logowania i rejestracji na wydarzenia. Wszystkie funkcje są w pełni działające, a aplikacja jest gotowa do wdrożenia w produkcji.

Możliwe kierunki rozwoju

1. **Integracja z systemami płatności**
    a. Obecnie wydarzenia są darmowe, ale w przyszłości można wprowadzić możliwość tworzenia wydarzeń płatnych. W takim przypadku należałoby zintegrować system z popularnymi systemami płatności online, np. PayPal, Stripe, aby umożliwić użytkownikom opłacenie biletów przed zapisaniem się na wydarzenie.
2. **Funkcje powiadomień**
    a. Możliwość wdrożenia systemu powiadomień e-mail lub SMS, informujących użytkowników o nadchodzących wydarzeniach, zbliżających się terminach lub przypomnieniach o zapisanych wydarzeniach.
3. **Rozbudowa funkcji profilu użytkownika**
    a. Możliwość rozbudowy profilu użytkownika, umożliwiająca dodawanie zdjęcia profilowego, ustawianie preferencji powiadomień oraz przeglądanie historii zapisanych wydarzeń.
4. **Optymalizacja dla urządzeń mobilnych**

    a. Dalsze optymalizowanie aplikacji pod kątem urządzeń mobilnych, zapewniając lepsze doświadczenie użytkownika na mniejszych ekranach, w tym wprowadzenie widoków responsywnych oraz aplikacji mobilnej.
5. **Rekomendacje wydarzeń**
    a. Implementacja algorytmu rekomendacji, który sugerowałby użytkownikom wydarzenia na podstawie ich poprzednich zapisów lub preferencji. Może to opierać się na analizie zachowań użytkowników lub ich lokalizacji.
6. **Zwiększenie bezpieczeństwa**
    a. Wdrożenie dodatkowych mechanizmów bezpieczeństwa, takich jak uwierzytelnianie dwuskładnikowe (2FA) przy logowaniu i rejestracji użytkowników, aby poprawić ochronę kont użytkowników.
7. **Wydarzenia online**
    a. Rozszerzenie systemu o możliwość tworzenia wydarzeń online, umożliwiając organizatorom prowadzenie transmisji wideo, np. za pomocą platformy Zoom, Google Meet, lub własnych rozwiązań wideo.
8. **Zaawansowane analizy i raportowanie**
    a. Dodanie funkcji umożliwiającej organizatorom wydarzeń tworzenie raportów, analiz i wykresów dotyczących frekwencji na wydarzeniach, liczby zarejestrowanych użytkowników oraz innych danych statystycznych.
9. **Rozbudowa interakcji społecznościowych**
    a. Możliwość dodawania komentarzy do wydarzeń, oceniania wydarzeń przez uczestników oraz dyskusji z innymi uczestnikami. Może to wzbogacić społeczność wokół wydarzeń i zwiększyć zaangażowanie użytkowników.
10. **Integracja z kalendarzami**
    a. Integracja aplikacji z popularnymi kalendarzami online, np. Google Calendar, umożliwiająca użytkownikom dodanie wydarzeń do ich własnych kalendarzy, co zwiększy wygodę i organizację.

Wnioski

Zrealizowanie tego projektu pozwoliło na stworzenie w pełni funkcjonalnej aplikacji umożliwiającej użytkownikom tworzenie wydarzeń, zapisanie się na nie oraz generowanie biletów PDF. Implementacja wszystkich kluczowych funkcji przebiegła zgodnie z planem, a aplikacja jest gotowa do dalszego rozwoju i wdrożenia.

Rozwój aplikacji w kierunku płatnych wydarzeń, powiadomień oraz integracji z systemami zewnętrznymi to obszary, które mogą znacznie zwiększyć jej funkcjonalność i atrakcyjność. Zoptymalizowanie aplikacji pod kątem urządzeń mobilnych oraz dodanie zaawansowanych opcji analitycznych i społecznościowych może przyczynić się do większej interakcji użytkowników oraz rozwoju aplikacji w dłuższej perspektywie.