Wydział Elektroniki
i Technik Informacyjnych
POLITECHNIKA WARSZAWSKA

Instytut Radioelektroniki i Technik Multimedialnych

Sprawozdanie do pracy dyplomowej inżynierskiej
Kierunek Telekomunikacja
Specjalizacja Techniki Teleinformatyczne

Filip Jaworski

Projekt i wdrożenie aplikacji webowej interaktywnej mapy wydarzeń w czasie rzeczywistym

imię i nazwisko promotora
dr inż. Piotr Bobiński

Warszawa 2024

Spis treści

1. Wstęp

W dzisiejszym cyfrowym świecie rozwój aplikacji webowych odgrywa kluczową rolę we wszystkich dziedzinach życia, redefiniując sposób, w jaki ludzie pracują, uczą się i bawią. Wraz z narastającymi wymaganiami dotyczącymi wydajności, elastyczności i skalowalności aplikacji internetowych, coraz większa liczba organizacji i projektów przechodzi na architekturę serverless, aby efektywniej zarządzać swoimi zasobami i skoncentrować się na tworzeniu innowacyjnych funkcji aplikacyjnych.
W odpowiedzi na te wyzwania, niniejsza praca skupia się na zastosowaniu architektury serverless do projektowania i wdrażania aplikacji webowej opartej na usługach chmurowych oferowanych przez Google Cloud Platform (GCP). Celem pracy jest przeanalizowanie możliwości i korzyści związanych z wykorzystaniem tych technologii oraz praktyczna demonstracja ich zastosowania w projekcie.

1.1. Cel pracy

Cel niniejszego projektu obejmuje stworzenie i wdrożenie interaktywnej mapy wydarzeń w czasie rzeczywistym. Wykorzystując technologie webowe takie jak React i Node.js oraz integrując się z usługami zewnętrznymi dostarczającymi informacje o wydarzeniach, projekt ten ma na celu umożliwienie użytkownikom przeglądania bieżących i planowanych wydarzeń w ich okolicy za pomocą intuicyjnej mapy wykorzystując Google Maps API. Wykorzystanie formatów Graph API Facebooka pozwoli na pobieranie i przetwarzanie informacji o wydarzeniach bezpośrednio z bazy danych Facebooka.
Wykorzystane technologie umożliwią wprowadzanie i odczytywanie zmian na stronie webowej bez konieczności jej odświeżania, co pozwoli jej użytkownikom na przyjemne i responsywne użytkowanie. Projekt ten może być nie tylko praktycznym narzędziem dla lokalnej społeczności, ale również otwierać możliwość na rozbudowę o dodatkowe funkcje, takie jak recenzje wydarzeń, integrację z systemami biletowymi czy analizę trendów w lokalnych aktywnościach społecznościowych.

1.2. Technologia

W ramach sprawozdania skupimy się na wykorzystaniu konkretnych usług chmurowych oferowanych przez GCP, takich jak Firebase Hosting, Cloud Storage, Cloud Run, Cloud Functions, Cloud SQL, Firebase Authentication, Google Cloud Pub/Sub, Identity and Access Management (IAM) oraz Cloud Build. Wykorzystane zostaną również API Google Maps i Graph API Facebooka.
Firebase Hosting zapewni nam możliwość hostowania front-endowej części aplikacji oraz plików statycznych, zapewniając szybkie ładowanie i skalowalność. Cloud Storage posłuży do przechowywania plików, a Cloud CDN przyspieszy dostarczanie treści globalnie.
Cloud Run umożliwi nam uruchamianie kontenerów Docker dla backendowej części aplikacji, zapewniając automatyczne skalowanie i wydajność. Cloud Functions posłuży do przetwarzania zdarzeń, a Cloud SQL do przechowywania danych w bazie PostgreSQL.
Firebase Authentication umożliwi zarządzanie użytkownikami, a Google Cloud Pub/Sub posłuży do zarządzania komunikatami między różnymi komponentami systemu. Natomiast Identity and Access Management (IAM) pozwoli na kontrolę dostępu do usług i zasobów na GCP.

W dalszej części pracy szczegółowo omówiona zostanie każda z technologii, opiszemy ich rolę w architekturze aplikacji oraz przedstawimy proces implementacji i integracji.

1.2.1. Opis szczegółowy zastosowanych technologii

1. **React:**
   Rola: React jest biblioteką JavaScript do budowy dynamicznych i responsywnych interfejsów użytkownika w aplikacji webowej.
   Funkcje: Umożliwia tworzenie komponentów UI, zarządzanie stanem aplikacji, obsługę interakcji użytkownika.
   Zalety: Deklaratywność, składność, efektywność renderowania, szeroka społeczność i wsparcie.

2. **Node.js:**
   Rola: Node.js to środowisko uruchomieniowe JavaScript po stronie serwera.
   Funkcje: Umożliwia tworzenie serwerów i aplikacji internetowych, obsługę zapytań HTTP, dostęp do systemu plików.
   Zalety: Wydajność, skalowalność, jednolity język programowania dla front-endu i back-endu, bogaty ekosystem modułów.

3. **JavaScript (Node.js):**
   Rola: JavaScript jest językiem programowania używanym zarówno po stronie klienta (front-end), jak i po stronie serwera (back-end).
   Funkcje: Obsługa interakcji użytkownika, manipulacja DOM, komunikacja z serwerem, logika biznesowa.
   Zalety: Uniwersalność, prostota, popularność, szybki rozwój, bogate zasoby bibliotek i frameworków.

4. **Google Cloud Platform (GCP):**
   Rola: GCP to platforma chmurowa oferująca różnorodne usługi do hostowania, zarządzania i skalowania aplikacji w chmurze.
   Funkcje: Elastyczne rozwiązania chmurowe, automatyczne skalowanie, zarządzanie zasobami, usługi analityczne i sztucznej inteligencji.
   Zalety: Wysoka dostępność, bezpieczeństwo danych, wsparcie dla różnych języków programowania, integracja z narzędziami deweloperskimi.

5. **Google App Engine (GAE):**
   Rola: Google App Engine to platforma jako usługa (PaaS), która umożliwia łatwe wdrażanie i zarządzanie aplikacjami bez konieczności zarządzania infrastrukturą.
   Funkcje: Automatyczne skalowanie, obsługa wielu języków programowania, integracja z innymi usługami GCP.
   Zalety: Szybkie wdrożenie aplikacji, elastyczność w zakresie konfiguracji, wsparcie dla różnych typów aplikacji.

6. **Cloud Run:**
   Rola: Cloud Run umożliwia uruchamianie kontenerów Docker dla backendowej części aplikacji w chmurze.
   Funkcje: Automatyczne skalowanie, wysoka wydajność, obsługa wielu języków programowania.
   Zalety: Elastyczność w zakresie konfiguracji zasobów, łatwe zarządzanie kontenerami, szybkie wdrażanie.

7. **Cloud Build:**
   Rola: Cloud Build to usługa automatyzacji procesu budowania i wdrażania aplikacji w chmurze.
   Funkcje: Kontrola wersji, budowanie i testowanie aplikacji, wdrażanie na platformie GCP.
   Zalety: Skuteczne zarządzanie cyklem życia aplikacji, automatyczne skalowanie zasobów, integracja z popularnymi narzędziami deweloperskimi.

8. **Cloud Storage:**
   Rola: Cloud Storage służy do przechowywania plików w chmurze.
   Funkcje: Zapewnia nieograniczoną skalowalność, trwałość danych oraz integrację z innymi usługami chmurowymi.
   Zalety: Wydajne zarządzanie dużymi zasobami danych, obsługa wielu typów plików, automatyczne tworzenie kopii zapasowych.

9. **Cloud Functions:**
   Rola: Cloud Functions służy do wykonywania kodu w chmurze w odpowiedzi na zdarzenia.

Funkcje: Serverless computing, obsługa zdarzeń w czasie rzeczywistym, skalowalność na żądanie.
Zalety: Brak konieczności zarządzania infrastrukturą, niski koszt, szybkie wdrażanie i skalowanie.

10. **Cloud SQL PostgresSQL:**
    Rola: Cloud SQL to zarządzana usługa baz danych relacyjnych w chmurze.
    Funkcje: Przechowywanie danych, skalowalność, automatyczne tworzenie kopii zapasowych.
    Zalety: Bezpieczeństwo danych, wysoka dostępność, łatwe zarządzanie.

11. **Firebase Hosting:**
    Rola: Firebase Hosting umożliwia hostowanie front-endowej części aplikacji oraz plików statycznych w chmurze.
    Funkcje: Zapewnia szybkie ładowanie, skalowalność oraz łatwe zarządzanie zasobami.
    Zalety: Prosta integracja z innymi usługami Firebase, automatyczne wdrażanie, obsługa niestandardowych domen.

12. **Firebase Authentication:**
    Rola: Firebase Authentication zapewnia uwierzytelnianie użytkowników w aplikacji.
    Funkcje: Zarządzanie użytkownikami, obsługa wielu metod uwierzytelniania, integracja z innymi usługami Firebase.
    Zalety: Proste w użyciu, obsługa autoryzacji wielu platform, bezpieczeństwo danych użytkowników.

13. **Google Cloud Pub/Sub:**
    Rola: Google Cloud Pub/Sub służy do przesyłania wiadomości między różnymi komponentami systemu.
    Funkcje: Asynchroniczna komunikacja, obsługa zdarzeń w czasie rzeczywistym, skalowalność.
    Zalety: Wydajne przetwarzanie komunikatów, elastyczność konfiguracji, integracja z innymi usługami chmurowymi.

14. **Identity and Access Management (IAM):**
    Rola: IAM to usługa zarządzania tożsamością i dostępem w chmurze.
    Funkcje: Kontrola dostępu do zasobów, zarządzanie uprawnieniami użytkowników, audyt działań.
    Zalety: Wysoki poziom bezpieczeństwa, elastyczność w zakresie konfiguracji, integracja z innymi usługami GCP.

15. **Graph API (Facebook):**
    Rola: Graph API to interfejs programistyczny udostępniany przez Facebooka do pozyskiwania danych z platformy Facebook.
    Funkcje: Umożliwia dostęp do informacji o użytkownikach, grupach, wydarzeniach, komentarzach, zdjęciach itp.
    Zalety: Bogata funkcjonalność, dostęp do danych społecznościowych, integracja z aplikacjami webowymi.

16. **Google Maps API:**
    Rola: Google Maps API dostarcza narzędzi do integracji interaktywnej mapy w aplikacjach internetowych.
    Funkcje: Wyświetlanie map, dodawanie znaczników, wyznaczanie tras, geolokalizacja, obsługa map w czasie rzeczywistym.
    Zalety: Łatwość integracji, bogata dokumentacja, różnorodność funkcji, wsparcie dla wielu platform.

17. **Docker:**
    Rola: Docker to platforma konteneryzacyjna, która umożliwia izolację aplikacji w kontenerach, co ułatwia zarządzanie zależnościami i wdrażanie aplikacji.

Funkcje: Tworzenie, uruchamianie i udostępnianie kontenerów, zarządzanie obrazami, izolacja zasobów.
Zalety: Konsystencja środowiska, przenośność aplikacji, szybkie wdrażanie, łatwe skalowanie.

18. **Kubernetes:**
    Rola: Kubernetes to system do automatyzacji zarządzania aplikacjami w kontenerach, który umożliwia orchestrację i skalowanie aplikacji w środowisku chmurowym.
    Funkcje: Orchestracja kontenerów, automatyczne skalowanie, monitorowanie zasobów, wdrażanie i zarządzanie aplikacjami.
    Zalety: Elastyczność w zakresie konfiguracji, wysoka dostępność, samonaprawiające się mechanizmy, wsparcie dla różnych dostawców chmurowych.

19. **Git:**
    Rola: Git to system kontroli wersji, który służy do śledzenia zmian w kodzie aplikacji.
    Funkcje: Wersjonowanie kodu, zarządzanie historią zmian, współpraca zespołowa.
    Zalety: Skalowalność, rozproszone środowisko, wsparcie dla wielu gałęzi kodu.

20. **GitHub:**
    Rola: GitHub to platforma hostingowa dla projektów opartych na Git, która umożliwia współpracę i zarządzanie kodem.
    Funkcje: Hosting repozytoriów, zarządzanie problemami i zgłoszeniami, integracja z narzędziami CI/CD.
    Zalety: Społeczność i ekosystem, narzędzia do rozwoju i monitorowania pracy nad oprogramowaniem.

2. Specyfikacja funkcjonalności modelu

W ramach tego projektu, specyfikacja funkcjonalności modelu obejmuje zestaw cech i zachowań, które aplikacja ma oferować użytkownikom. Poniżej przedstawiam szczegółową specyfikację funkcjonalności modelu:

1. **Przeglądanie Wydarzeń:**
   — Użytkownicy mogą przeglądać listę wydarzeń na interaktywnej mapie.
   — Wydarzenia są reprezentowane za pomocą dynamicznych znaczników na mapie.
2. **Filtrowanie Wydarzeń:**
   — Użytkownicy mają możliwość filtrowania wydarzeń według kategorii, odległości, daty, czy popularności.
   — Filtry umożliwiają spersonalizowane wyświetlanie wydarzeń zgodnie z preferencjami użytkownika.
3. **Integracja Zewnętrznych Źródeł:**
   — Aplikacja integruje się z zewnętrznymi serwisami dostarczającymi informacje o wydarzeniach, takimi jak serwisy społecznościowe, strony wydarzeń, itp.
   — Umożliwia to pozyskiwanie szerokiego spektrum danych o różnorodnych wydarzeniach.
4. **Powiadomienia:**
   — System powiadomień informuje użytkowników o zbliżających się wydarzeniach, które ich mogą zainteresować.
   — Powiadomienia są personalizowane i dostosowane do preferencji użytkownika.
5. **Dodawanie Wydarzeń::**
   — Użytkownicy mają możliwość dodawania własnych wydarzeń do mapy.
   — Funkcja ta umożliwia społeczności lokalnej promowanie lokalnych wydarzeń i inicjatyw.
6. **Integracja Z Google Maps API:**
   — Wykorzystanie API Google Maps umożliwia wygodne przeglądanie interaktywnej mapy.
   — Użytkownicy mogą korzystać z zaawansowanych funkcji mapy, takich jak nawigacja i zasięg widoku.
7. **Recenzje Wydarzeń:**
   — Użytkownicy mogą dodawać recenzje i komentarze do wydarzeń, które uczestniczyli.
   — Recenzje pozwalają na ocenę jakości i atrakcyjności wydarzeń przez społeczność.
8. **Zarządzanie Użytkownikami:**
   — System uwierzytelniania i autoryzacji umożliwia rejestrację, logowanie i zarządzanie kontami użytkowników.
   — Każdy użytkownik ma dostęp do spersonalizowanych ustawień i preferencji.
9. **Integracja z Systemami Biletowymi:**
   — Planowane jest rozszerzenie aplikacji o funkcje integracji z systemami biletowymi, umożliwiającymi zakup biletów online(na razie będą to wyłącznie linki do eBilety.pl)
   — Integracja ta ułatwi użytkownikom uczestnictwo w wydarzeniach poprzez prosty i wygodny proces zakupu biletów.
10. **Analiza Trendów:**
    — System analityczny zbiera i analizuje dane dotyczące trendów w lokalnych aktywnościach społecznościowych.
    — Analiza trendów umożliwia identyfikację popularnych wydarzeń i zainteresowań społeczności lokalnej.
11. **Chmurowe Przechowywanie Danych:**
    — Wykorzystanie usług chmurowych do przechowywania danych dotyczących wydarzeń, takich jak opisy, lokalizacje, daty itp.
    — Chmurowe przechowywanie danych zapewnia bezpieczeństwo, skalowalność i dostępność danych.
12. **Automatyzacja Procesów:**
    — Wykorzystanie chmurowych narzędzi do integracji umożliwia automatyzację procesów, takich jak wysyłanie powiadomień push.

— Automatyzacja procesów zwiększa efektywność i wydajność działania aplikacji.

13. **Zaawansowane Zabezpieczenia:**
    — Usługi chmurowe oferują zaawansowane funkcje zabezpieczeń, zapewniając ochronę danych użytkowników i aplikacji.
    — Zaawansowane zabezpieczenia obejmują kontrolę dostępu, szyfrowanie danych i monitoring bezpieczeństwa.

3. Etapy projektowania aplikacji

3.1. Faza planowania

Faza planowania została przedstawiona w poprzednich punktach. Składa się na nią wybór technologii i narzędzi do tworzenia aplikacji webowych opartych o systemy chmurowe. W tym przypadku wybór padł na platformę GCP ze względu na łatwość implementacji Google Maps API do projektu, jak również wsparcie wielu innych funkcji chmurowych, takich jak Cloud Build, czy Cloud Run. Technologie, które zostały w tej fazie starannie dobrane pod specyfikację modelu projektowego i wymagania przyszłych użytkowników.
W tej fazie również została podjęta decyzja o zastosowaniu konteneryzacji przy użyciu Dockera oraz zarządzaniu kontenerami przy pomocy Kubernetesa i narzędzi 'kubectl'.
Na tym etapie nastąpił podział pracy nad projektem aplikacji webowej i rozbity na mniejsze problemy, takie jak backend, frontend, hosting, integracja usług GCP, czy obsługa Cloud SQL i bazy danych.

3.2. Faza projektowania

1. **Projekt interfejsu użytkownika (UI):**
    — Wireframing: Stworzenie wireframe interfejsu użytkownika, prezentujący główne komponenty, układ i nawigację.
    — Projektowanie UI: Opracowanie prostego projektu interfejsu użytkownika, w tym kolorystyki, typografii, ikon i grafik (na tym etapie głównie place holdery).
    — Prototypowanie: Utwórzenie interaktywnego prototypu aplikacji, który pozwoli na wstępną ocenę funkcjonalności i użytkowalności(jako ostatni krok)
2. **Projekt architektury aplikacji:**
    — Wybór architektury: Podjąć decyzję czy zastosować architekturę monolityczną czy mikrousługową, uwzględniając wymagania aplikacji i możliwości skalowania.(skłaniam się w kierunku monolitu)
    — Projektowanie komponentów: Sprecyzowanie głównych komponentów aplikacji, ich funkcji i relacji między nimi.
    — Diagramy architektury: Opracowanie diagramów architektury aplikacji, obejmujące frontend, backend, bazę danych oraz integracje z usługami zewnętrznymi.
3. **Projekt bazy danych:**
    — Modelowanie danych: Projekt struktury bazy danych, uwzględniając encje takie jak użytkownicy, wydarzenia, kategorie, itp.
    — Optimizacja wydajności: Wybranie strategii indeksowania, partycjonowania danych i cache'owania, aby zapewnić wysoką wydajność aplikacji.
4. **Projekt bezpieczeństwa:**
    — Identyfikacja zagrożeń: Identyfikacja potencjalnych zagrożenia dla bezpieczeństwa aplikacji, takie jak ataki XSS, CSRF, SQL Injection, itp.(pobieżnie)
    — Strategia autoryzacji: Określenie, jakie mechanizmy autoryzacji będą stosowane w aplikacji, np. Firebase Authentication dla użytkowników, klucze API dla integracji z zewnętrznymi serwisami.
    — Kontrola dostępu: Zdefiniowanie ról i uprawnień użytkowników w aplikacji, aby zapewnić odpowiedni poziom dostępu do zasobów.
5. **Planowanie integracji z usługami zewnętrznymi:**

— Identyfikacja integracji: Identyfikacja zewnętrznych serwisów i API, z którymi aplikacja będzie integrować się, takie jak Google Maps API, serwisy wydarzeń, media społecznościowe, itp.
— Dokumentacja integracji: Przegląd dokumentacji API zewnętrznych usług i identyfikacja wymaganych punktów integracji oraz sposobów autoryzacji.
— Testowanie integracji: Projekt scenariuszy testowych dla integracji zewnętrznych usług i planowanie testów, aby upewnić się, że komunikacja między aplikacją a zewnętrznymi serwisami działa poprawnie.

6. **Dokumentacja projektu:**
— Dokumentacja architektury: Sporządzenie dokumentacji architektury aplikacji, opisującąej główne komponenty, ich funkcje i relacje.
— Dokumentacja interfejsu API: Opisanie interfejsów API, zarówno frontendu jak i backendu.
— Instrukcje użytkownika: Opracowanie instrukcji użytkownika, które pomogą użytkownikom w korzystaniu z aplikacji, w tym rejestracji, logowania, dodawania wydarzeń, itp.

3.3. Faza implementacji

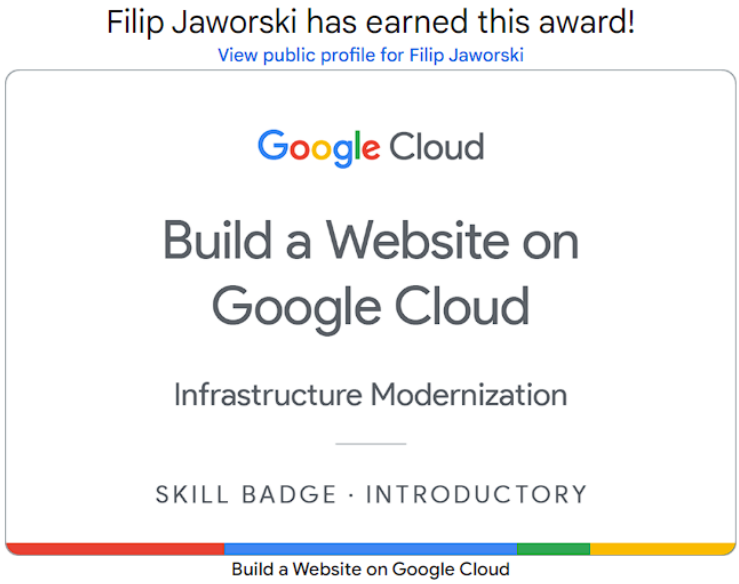
Ten etap skupi się na implementacji gotowych komponentów i łączeniu ich ze sobą w jedną, zgraną całość.
Implementacja frontendu: Rozpocznij tworzenie aplikacji React, uwzględniając interaktywną mapę, formularze, integracje z Firebase Authentication, itp.
Implementacja backendu: Stwórz aplikację Node.js, która będzie obsługiwać logikę biznesową, integracje z zewnętrznymi serwisami, bazą danych, itp.
Integracja z usługami GCP: Skonfiguruj integracje z usługami Google Cloud Platform, takimi jak Firebase Hosting, Cloud Run, Cloud SQL, itp.

3.4. Faza utrzymania

Ta faza obejmie:
Testowanie: Przeprowadź testy jednostkowe, integracyjne i funkcjonalne, aby zapewnić wysoką jakość aplikacji.
Wdrożenie: Wdróż aplikację na środowiska produkcyjne, takie jak Firebase Hosting, Cloud Run, dbając o odpowiednie konfiguracje bezpieczeństwa i skalowalności.
Monitorowanie i optymalizacja: Monitoruj wydajność aplikacji, reaguj na ewentualne problemy, optymalizuj wykorzystanie zasobów chmurowych.
Wsparcie i rozwój: Zapewnij wsparcie techniczne dla użytkowników, rozwijaj aplikację o nowe funkcjonalności i usprawnienia na podstawie feedbacku.

3.5. Co zostało zrealizowane

Na tym etapie pracy została przeprowadzona dogłębna analiza istniejących narzędzi i technologii webowych, które mogłyby pomóc w tworzeniu aplikacji webowej interaktywnej mapy wydarzeń w czasie rzeczywistym. Wybrany został framework Google Cloud GCP, co za tym została zgłębiona wiedza i zrozumienie tej platformy wraz z głównymi jej narzędziami i usługami. Na platformie szkoleniowej Google Skills Boost zebrałem łącznie 5000 pkt i wiele odznak tak zwanych badges, w tym za ukończenie kursu 'Cloud Engineer Learning Path' i 'Develop Serverless Applications on Cloud Run'.



4. Podsumowanie

Projekt "Interaktywna mapa wydarzeń w czasie rzeczywistym" ma na celu stworzenie aplikacji webowej, która umożliwi użytkownikom przeglądanie bieżących i planowanych wydarzeń w ich okolicy za pomocą intuicyjnej mapy. Wykorzystując technologie webowe takie jak React i Node.js oraz integrację z usługami zewnętrznymi dostarczającymi informacje o wydarzeniach, aplikacja zapewni interaktywną i atrakcyjną platformę dla lokalnej społeczności.

Architektura aplikacji opiera się na podejściu serverless na platformie Google Cloud Platform (GCP), co pozwala na skalowalność, wydajność i elastyczność. Wdrażając aplikację na usługi takie jak Firebase Hosting, Cloud Run i Cloud SQL, unikamy konieczności zarządzania infrastrukturą, co pozwala skupić się na rozwijaniu funkcjonalności aplikacji.

Wnioski:

Projekt "Interaktywna mapa wydarzeń w czasie rzeczywistym" stanowi innowacyjne narzędzie dla lo-

kalnej społeczności, umożliwiające łatwy dostęp do informacji o wydarzeniach oraz interakcję z nimi. Wykorzystanie technologii webowych i chmurowych, takich jak React, Node.js i usługi Google Cloud Platform, zapewnia solidne podstawy dla dalszego rozwoju aplikacji oraz potencjalnej rozbudowy o dodatkowe funkcje.

5. Bibliografia

— https://www.cloudskillsboost.google/ - Google Skills Boost platform

— https://cloud.google.com/?hl=pl - Google Cloud

— https://developers.google.com/maps?hl=pl - Google Maps Platform

— https://developers.facebook.com/docs/graph-api/ - Meta Graph API

— https://cloud.google.com/sql/docs/postgres - Cloud SQL for PostgreSQL

— https://docs.docker.com/manuals/ - Docker dokumentacja

— https://kubernetes.io/docs/home/ - Kubernetes dokumentacja

— https://pl.react.dev/ - React dev

— https://nodejs.org/docs/latest/api/ - Node.js documentation