Collegium Witelona Uczelnia Państwowa w Legnicy
Wydział Nauk Technicznych i Ekonomicznych
Kierunek: Informatyka

Dokumentacja Projektu: SportConnect

System łączenia partnerów sportowych

Autorzy:
Sebastian Waga nr. indexu 43894
Michał Żmuda nr. indexu 41926
Adam Rudziewicz nr. indexu 43882
Szymon Rogula nr. indexu 43880

Legnica, 2026

Spis treści

1 Część I: Moduł Backend

1.1 Opis funkcjonalny systemu

Część backendowa aplikacji **SportConnect** została zaprojektowana jako RESTful API, które stanowi fundament funkcjonalności aplikacji. System odpowiada za zarządzanie logiką biznesową, przetwarzanie danych oraz komunikację z bazą danych.

Główne funkcjonalności zaimplementowane po stronie serwera obejmują:

- **Zarządzanie tożsamością i autoryzacja:** Rejestracja, logowanie (JWT), odzyskiwanie i resetowanie haseł, system ról (User, Admin).
- **Zarządzanie użytkownikami:** Edycja profilu, aktualizacja lokalizacji geograficznej, zarządzanie preferowanymi dyscyplinami sportowymi, usuwanie konta.
- **Algorytm wyszukiwania partnerów:** Dopasowywanie użytkowników na podstawie lokalizacji, dostępności i wspólnych zainteresowań.
- **Zarządzanie dyscyplinami sportowymi:** Pełny system CRUD dla sportów (zarządzany przez administratora).
- **System zaproszeń i matchowania:** Tworzenie, akceptowanie, odrzucanie i anulowanie zaproszeń do wspólnego treningu.
- **Panel Administratora:** Agregacja statystyk systemowych, zarządzanie bazą użytkowników oraz blokowanie kont.

1.2 Opis technologiczny

Projekt został zrealizowany w nowoczesnym stosie technologicznym .NET:

- **Framework:** ASP.NET Core 6.0+ (Architektura MVC, Dependency Injection).
- **Język:** C# (Asynchroniczność, silne typowanie).
- **ORM:** Entity Framework Core (Podejście Code-First).
- **Baza danych:** SQL Server / PostgreSQL.
- **Bezpieczeństwo:** JWT (JSON Web Tokens), ASP.NET Core Identity, BCrypt/PBKDF2.
- **Wzorce projektowe:** Repository Pattern, Service Layer, DTO (Data Transfer Objects), Unit of Work.
- **Dokumentacja:** Swagger/OpenAPI.

1.3 Instrukcja uruchomienia systemu

Wymagania: .NET SDK 6.0+, SQL Server, IDE (VS 2022 / VS Code).

1. Pobierz kod źródłowy i wejdź do katalogu `backend`.
2. Przywróć zależności:

```
dotnet restore
```

3. Skonfiguruj Connection String w `appsettings.Development.json`.
4. Uruchom migracje bazy danych:

```
dotnet ef database update
```

5. Uruchom aplikację:

```
dotnet run
```

1.4 Wnioski projektowe

Użycie **Entity Framework Core** znacząco przyspieszyło rozwój dzięki automatycznemu generowaniu schematu bazy danych. Największym wyzwaniem technicznym była implementacja algorytmu matchowania opartego na **wzorze Haversine**, co wymagało optymalizacji zapytań przestrzennych.

2 Część II: Moduł Frontend

2.1 Opis funkcjonalny systemu

Frontend to aplikacja typu SPA (Single Page Application), responsywna i dostosowana do urządzeń mobilnych.

- **Widok Match:** Centralny punkt wyszukiwania partnerów.
- **Interaktywne mapy:** Wizualizacja lokalizacji za pomocą biblioteki **Leaflet**.
- **Skrzynka odbiorcza:** Zarządzanie komunikacją i zaproszeniami.
- **Obsługa motywów:** Tryb jasny i ciemny.
- **Zabezpieczenie tras:** Navigation Guards chroniące dostęp do profilu i panelu admina.

2.2 Opis technologiczny

- **Framework:** Vue 3 (Composition API).
- **Język:** TypeScript.
- **Zarządzanie stanem:** Pinia.
- **Stylizowanie:** Tailwind CSS (Utility-first).
- **Build Tool:** Vite.
- **Komunikacja:** Axios z interceptorami do obsługi tokenów JWT.

2.3 Instrukcja uruchomienia systemu

1. Wejdź do katalogu `frontend`.
2. Zainstaluj pakiety: `npm install`.
3. Uruchom tryb deweloperski: `npm run dev`.
4. Budowanie wersji produkcyjnej: `npm run build`.

2.4 Wnioski projektowe

Zastosowanie **TypeScript** pozwoliło na wyeliminowanie wielu błędów na etapie kompilacji, szczególnie przy mapowaniu modeli danych z API. Wykorzystanie **Vite** znacząco poprawiło komfort pracy (HMR) w porównaniu do starszych narzędzi.

3 Część III: Testowanie i Jakość (QA)

3.1 Zakres testów

Za proces testowy odpowiedzialny był **Szymon Rogula**. Testy objęły:

- **Unit Tests (Backend):** AuthController, TrainingRequestsController, EmailService.
- **Walidacja modeli:** Ponad 25 testów sprawdzających poprawność danych (wiek, współrzędne, długość opisu).
- **Bezpieczeństwo:** Testy hashowania haseł i generowania claims w JWT.

3.2 Narzędzia testowe

- **xUnit:** Główny framework testowy.
- **Moq:** Mockowanie zależności.
- **FluentAssertions:** Bardziej czytelna składnia asercji.
- **InMemory Database:** Testowanie warstwy danych bez fizycznej bazy.

3.3 Statystyki

Zaimplementowano ponad **40 testów automatycznych**, w tym:

- 11 testów kontrolerów API.
- 7 testów serwisów biznesowych.
- 25+ testów walidacji modeli domenowych.

4 Podział obowiązków w zespole – Podsumowanie

Sebastian Waga i Michał Żmuda (Backend Team): Architektura systemu, baza danych, logika biznesowa, system autoryzacji JWT, algorytm matchowania, dokumentacja Swagger.

Adam Rudziewicz (Frontend Team): Konfiguracja Vue 3 + Vite, implementacja komponentów UI, integracja z mapami, zarządzanie stanem (Pinia), responsywność (Tailwind).

Szymon Rogula (Tester Team): Planowanie i wykonanie testów jednostkowych i integracyjnych, raportowanie błędów, weryfikacja wymagań biznesowych.

5 Podsumowanie końcowe

Aplikacja **SportConnect** została zrealizowana zgodnie z najlepszymi praktykami inżynierii oprogramowania. Wykorzystanie nowoczesnych frameworków (.NET Core i Vue 3) pozwoliło na stworzenie stabilnego, wydajnego i skalowalnego systemu. Modułowa architektura oraz wysokie pokrycie testami zapewniają łatwość dalszego rozwoju projektu o funkcje takie jak powiadomienia w czasie rzeczywistym (SignalR) czy caching (Redis).