Sprawozdanie z Projektu

Temat: Aplikacja MiniStrava

Autorzy:
Mateusz Tęcza (.NET)
Jakub Trznadel (Flutter)

2025

s4PAM Grupa1(1)

Prowadzący: Kamil Piech

Spis treści

1. Opis przedmiotu zamówienia i zakres prac

Celem projektu było zaprojektowanie i implementacja kompletnego systemu informatycznego służącego do rejestracji i analizy treningów sportowych. Projekt miał na celu stworzenie architektury rozproszonej, składającej się z serwera aplikacji (REST API), bazy danych oraz klienta mobilnego.

Głównym założeniem było stworzenie środowiska odwzorowującego realia komercyjnych rozwiązań typu *fitness tracker* (np. Strava), z naciskiem na poprawną obsługę danych geolokalizacyjnych oraz synchronizację danych w modelu online/offline.

1.1. Zaimplementowane funkcjonalności

System został podzielony na dwa główne moduły funkcjonalne: aplikację kliencką (mobilną) oraz warstwę serwerową (backend).

1.1.1. Aplikacja Mobilna

Interfejs użytkownika realizujący logikę biznesową po stronie klienta:

- **Rejestracja parametrów treningowych:** Wykorzystanie sensorów urządzenia do zapisu trasy, dystansu, prędkości i tempa w czasie rzeczywistym.
- **Praca w tle:** Implementacja usługi systemowej (Background Service) umożliwiającej ciągłe śledzenie lokalizacji nawet po zminimalizowaniu aplikacji.
- **Tryb Offline-First:** Mechanizm lokalnego buforowania aktywności w przypadku utraty połączenia z siecią i automatyczna synchronizacja po jego odzyskaniu.
- **Eksport do GPX:** Funkcja dostępna w widoku szczegółów aktywności, inicjująca żądanie do API w celu wygenerowania i pobrania pliku z trasą treningu w standardzie XML.
- **Wizualizacja danych:** Prezentacja przebytej trasy na interaktywnej mapie (OpenStreetMap) oraz wykresów statystycznych.

1.1.2. Backend i Panel Administracyjny

Centralny punkt przetwarzania danych i zarządzania systemem:

- **API RESTful:** Punkty końcowe obsługujące autoryzację (JWT), operacje CRUD na aktywnościach oraz przesyłanie plików multimedialnych (zdjęcia).
- **Panel Administratora:** Webowy interfejs renderowany po stronie serwera (MVC) do nadzoru nad użytkownikami i treściami w systemie.
- **Logika biznesowa:** Przeliczanie statystyk globalnych użytkownika (agregacja danych) oraz generowanie plików eksportu.

2. Opis technologiczny rozwiązania

W projekcie zastosowano nowoczesny stos technologiczny, zapewniający skalowalność, wydajność oraz przenośność kodu.

2.1. Warstwa serwerowa (Backend)

Backend został zrealizowany w oparciu o platformę **.NET 8.0** i język **C#**. Architektura rozwiązania łączy w sobie cechy Web API oraz tradycyjnej aplikacji webowej:

1. **Framework:** ASP.NET Core 8.0.
2. **Uwierzytelnianie:** Hybrydowe podejście wykorzystujące:
   - `JwtBearer` – do bezstanowej autoryzacji zapytań z aplikacji mobilnej.
   - `CookieAuthentication` – do utrzymania sesji administratora w panelu webowym.
3. **Komunikacja z bazą danych:** Zastosowano wzorzec *Code-First* przy użyciu biblioteki ORM **Entity Framework Core**. Pozwoliło to na definicję modeli w kodzie C# i automatyczną generację schematu bazy danych poprzez migracje.
4. **Dokumentacja:** Wykorzystano bibliotekę Swagger (OpenAPI) do automatycznego generowania dokumentacji technicznej endpointów.

2.2. Warstwa kliencka (Mobile)

Aplikacja mobilna została napisana w frameworku **Flutter** (język Dart), co umożliwiło kompilację kodu na platformy Android oraz iOS.

1. **Zarządzanie stanem:** Wykorzystano natywne mechanizmy Fluttera do zarządzania stanem interfejsu.
2. **Geolokalizacja:** Użyto bibliotek `geolocator` oraz `flutter_background_service` do obsługi sensorów GPS w oddzielnym wątku (Isolate), co zapobiega ubijaniu procesu przez system operacyjny.
3. **Mapy:** Zaimplementowano `flutter_map` – rozwiązanie oparte na otwartych danych OpenStreetMap, redukujące koszty licencyjne w porównaniu do Google Maps.
4. **Pamięć lokalna:** Wykorzystano `SharedPreferences` do przechowywania tokenów sesji oraz zbuforowanych aktywności (JSON) w trybie offline.

2.3. Struktura bazy danych

System wykorzystuje relacyjną bazę danych Microsoft SQL Server. Struktura została zaprojektowana zgodnie z podejściem *Code-First* przy użyciu Entity Framework Core. Baza składa się z czterech głównych encji:

2.3.1. Tabela `Users` (Użytkownicy)

Przechowuje dane uwierzytelniające oraz profilowe użytkowników aplikacji mobilnej.

- **UserId (PK):** Unikalny identyfikator użytkownika (int, auto-inkrementacja).
- **Email:** Adres e-mail służący do logowania (wymagany).
- **PasswordHash:** Bezpieczny skrót hasła (BCrypt).
- **Dane profilowe:** Imię, Nazwisko, Data urodzenia, Płeć, Wzrost, Waga.
- **AvatarUrl:** Ścieżka do pliku graficznego z awatarem użytkownika.
- **CreatedAt:** Data utworzenia konta.

2.3.2. Tabela `Activities` (Aktywności)

Główna tabela przechowująca zarejestrowane treningi. Relacja jeden-do-wielu z tabelą `Users` (jeden użytkownik może mieć wiele aktywności).

- **ActivityId (PK):** Unikalny identyfikator aktywności.
- **UserId (FK):** Klucz obcy powiązany z tabelą `Users` (kaskadowe usuwanie).
- **Name, Type:** Nazwa treningu i typ aktywności (np. "running", "cycling").
- **Parametry treningu:** Dystans (Distance), Czas trwania (Duration), Tempo (Pace), Średnia prędkość (AverageSpeed).
- **GpsTrack:** Pełny ślad trasy GPS zapisany jako zserializowany ciąg znaków (JSON String), co pozwala na przechowywanie złożonych tablic współrzędnych bez tworzenia osobnej tabeli na punkty.
- **PhotoUrl:** Ścieżka do zdjęcia dodanego do aktywności.

2.3.3. Tabela `UserStats` (Statystyki)

Tabela optymalizacyjna (agregująca), która przechowuje podsumowanie osiągnięć użytkownika. Eliminuje konieczność przeliczania sumy wszystkich treningów przy każdym zapytaniu API.

- **UserStatsId (PK):** Identyfikator rekordu.
- **UserId (FK):** Powiązanie z użytkownikiem.
- **Agregaty:** Całkowita liczba treningów (TotalWorkouts), łączny dystans (TotalDistance), łączny czas (TotalDuration), najszybsze tempo (FastestPace), maksymalny dystans (MaxDistance).
- **LastUpdated:** Data ostatniej aktualizacji statystyk.

2.3.4. Tabela `Admins` (Administratorzy)

Osobna tabela dla panelu administracyjnego, niezależna od użytkowników mobilnych.

- **Id (PK):** Identyfikator administratora.
- **Username:** Nazwa użytkownika (domyślnie "admin").
- **PasswordHash, IsPasswordSet:** Hash hasła oraz flaga wymuszająca ustawienie hasła przy pierwszym logowaniu.

3. Instrukcja uruchomienia systemu

System został przygotowany do uruchomienia w środowisku lokalnym w celach deweloperskich i prezentacyjnych.

3.1. Wymagania środowiskowe

- Środowisko uruchomieniowe .NET 8.0 SDK.
- Flutter SDK (kanał stable).
- Serwer bazy danych MS SQL Server (LocalDB lub pełna instancja).
- Środowisko Android SDK (dla emulatora lub urządzenia fizycznego).

3.2. Konfiguracja i uruchomienie backendu

1. Należy otworzyć projekt w katalogu głównym backendu. 2. W pliku `appsettings.json` zweryfikować poprawność łańcucha połączenia (`ConnectionStrings`) do lokalnej instancji SQL Server. 3. Zainicjować bazę danych poprzez wykonanie migracji:

```
dotnet ef database update
```

4. Uruchomić aplikację poleceniem:

```
dotnet run
```

Serwer nasłuchuje domyślnie na porcie HTTPS (zdefiniowanym w `launchSettings.json`).

3.3. Konfiguracja i uruchomienie aplikacji mobilnej

1. Przejść do katalogu projektu mobilnego i pobrać zależności:

```
flutter pub get
```

2. W pliku konfiguracyjnym serwisu autoryzacji (`auth_service.dart`) ustawić adres IP komputera hosta (np. `https://192.168.X.X:port`), aby emulator/urządzenie miało dostęp do API w sieci lokalnej. 3. Skompilować i uruchomić aplikację:

```
flutter run
```

4. Wnioski z pracy projektowej

4.1. Stopień realizacji przedmiotu zamówienia

Projekt zakończył się sukcesem, dostarczając kompletny system zgodny z założeniami przedmiotu zamówienia. System składa się z aplikacji mobilnej (Flutter), panelu administracyjnego (ASP.NET Core MVC) oraz interfejsu REST API. Poniżej przedstawiono szczegółową analizę realizacji poszczególnych grup wymagań.

4.1.1. Realizacja wymagań aplikacji mobilnej

Zaimplementowano wszystkie kluczowe wymagania funkcjonalne stawiane aplikacji klienckiej:

1. **Dostępność i Rejestracja (Pkt 1-4):** Aplikacja została przygotowana do pracy na systemach Android oraz iOS. Zaimplementowano pełny proces rejestracji, logowania oraz resetowania hasła (z wykorzystaniem tokenów e-mail). Użytkownik ma możliwość edycji pełnego profilu, w tym danych biometrycznych (waga, wzrost) oraz zdjęcia profilowego (Avatar).

2. **Obsługa Aktywności (Pkt 5-7):** Proces rejestracji treningu działa stabilnie, zapisując w czasie rzeczywistym parametry: czas, dystans, tempo, prędkość oraz ślad GPS. Użytkownik ma możliwość edycji metadanych (nazwa, notatka, zdjęcie, typ aktywności) po zakończeniu treningu.

3. **Prezentacja Danych (Pkt 8-10):** Historia aktywności prezentowana jest w formie listy z możliwością filtrowania (po typie) i sortowania (po dacie, dystansie). Szczegóły aktywności zawierają interaktywną mapę oraz wyliczone parametry.

4. **Statystyki i Ranking (Pkt 11-12):** Aplikacja oblicza i wyświetla globalne statystyki użytkownika oraz generuje rankingi oparte na sumarycznych osiągnięciach, pobierane z API.

5. **Integracja i Język (Pkt 13-15):** Zaimplementowano mechanizm synchronizacji danych offline-online. Interfejs aplikacji jest w pełni zlokalizowany w językach polskim i angielskim, z automatycznym wykrywaniem ustawień systemowych.

4.1.2. Realizacja wymagań panelu administracyjnego

Panel webowy został zintegrowany z backendem i spełnia postawione wymagania:

1. **Dostęp (Pkt 16):** Zabezpieczone logowanie dla administratora z oddzielną tabelą w bazie danych.

2. **Zarządzanie treścią (Pkt 17-18):** Administrator posiada wgląd w listę użytkowników i aktywności. Zaimplementowano zaawansowane filtrowanie (po e-mailu użytkownika, datach, dystansie) oraz możliwość usuwania niepożądanych wpisów.

3. **Dashboard (Pkt 19):** Strona główna panelu prezentuje globalne liczniki systemowe (liczba użytkowników, suma dystansów).

4.1.3. Realizacja wymagań API i backendu

Warstwa serwerowa stanowi fundament systemu:

1. **Standardy (Pkt 20-21):** REST API jest zgodne ze specyfikacją OpenAPI. Pełna dokumentacja techniczna została wygenerowana i udostępniona pod endpointem `/api/documentation` (Swagger UI).

2. **Eksport GPX (Pkt 22):** Zaimplementowano funkcjonalność generowania plików .gpx. Proces ten jest inicjowany przyciskiem w aplikacji mobilnej, jednak cała logika generowania pliku XML na podstawie surowych danych GPS odbywa się po stronie backendu, co odciąża urządzenie mobilne i gwarantuje poprawność formatu.

4.1.4. Konfiguracja i wymagania niefunkcjonalne

1. **Dane startowe (Pkt 6):** System posiada mechanizm automatycznego tworzenia konta administratora przy pierwszym uruchomieniu (`Program.cs`), z wymuszeniem ustawienia hasła przy pierwszym logowaniu.

2. **Bezpieczeństwo i UX (Pkt 23-25):** Komunikacja wymuszona jest przez protokół HTTPS. Aplikacja mobilna wykorzystuje dedykowany serwis (`flutter_background_service`) do niezawodnego działania w tle.

4.2. Realizacja rozszerzeń opcjonalnych

W ramach projektu zrealizowano jedno z kluczowych rozszerzeń opcjonalnych – **Tryb offline (Pkt 28)**. Aplikacja buforuje dane lokalnie i synchronizuje je po odzyskaniu połączenia, co jest krytyczne dla użyteczności w terenie.

4.3. Napotkane problemy

Podczas realizacji napotkano wyzwania związane z zarządzaniem uprawnieniami w systemie Android (szczególnie w kontekście lokalizacji w tle w nowszych wersjach API) oraz optymalizacją przesyłu dużych zbiorów punktów GPS między urządzeniem a bazą danych. Rozwiązano je poprzez serializację ścieżki do formatu JSON.

4.4. Możliwości rozwoju

System posiada modułową budowę, co pozwala na jego dalszą rozbudowę. Potencjalne kierunki rozwoju obejmują:

- **Powiadomienia Push:** Implementacja systemu notyfikacji informujących o nowych rekordach, aktywnościach znajomych oraz **zdobytych osiągnięciach** (gamifikacja).

- **Roszerzenie panelu Administratora:** Poprawa UI administratora, stworzenie dodatkowych funkcjonalności.

- **Rozszerzenie typów aktywności:** Dodanie obsługi dyscyplin takich jak pływanie czy trening siłowy.

- **Eksport danych:** Implementacja możliwości eksportu pełnej historii danych użytkownika do pliku w formacie **CSV**.
- **Funkcje AI:** Wykorzystanie algorytmów sztucznej inteligencji do generowania tekstowych podsumowań treningu oraz analizy postępów użytkownika.