Dokumentacja Techniczna: Aplikacja do Rejestrowania Treningów Pływackich

---

Tytuł projektu

Aplikacja do rejestrowania treningów pływackich

Krótki opis działania projektu

Aplikacja umożliwia rejestrowanie i zarządzanie treningami pływackimi. System obsługuje różne role użytkowników (zawodnik, trener, admin) z różnym zakresem uprawnień. Użytkownicy mogą przeglądać i edytować treningi, śledzić obecności oraz analizować kilometraż.

Autorzy projektu

- Filip Kluczniok
- Waldemar Krakowiak

Specyfikacja wykorzystanych technologii

1. **Backend:** C# .NET 8
2. **Baza danych:** MS SQL Server
3. **Środowisko uruchomieniowe:** Visual Studio 2022
4. **Biblioteki i frameworki:**
    - Entity Framework Core 8.0
    - ASP.NET Core 8.0
    - AutoMapper (opcjonalnie do mapowania obiektów)
    - FluentValidation (opcjonalnie do walidacji)
5. **Inne narzędzia:**
    - Docker (opcjonalnie dla ułatwienia konfiguracji środowiska)
    - Swagger (dokumentacja API)

Instrukcja pierwszego uruchomienia projektu

1. **Klonowanie repozytorium:**

    bash

    Skopiuj kod

    git clone <adres-repozytorium>

    cd <nazwa-repozytorium>

2. **Konfiguracja bazy danych:**

   Edytuj plik appsettings.json, podając połączenie do MS SQL Server:

   json

   Skopiuj kod

   "ConnectionStrings": {

   "DefaultConnection": "Server=localhost;Database=SwimTrainingDB;User Id=<username>;Password=<password>;"

   }

3. **Migracja bazy danych:**

   W terminalu wykonaj:

   Skopiuj kod

   dotnet ef migrations add InitialCreate

   dotnet ef database update

4. **Uruchomienie aplikacji:**

   Skopiuj kod

   dotnet run

5. **Dostęp do dokumentacji API:**
   - Aplikacja wystawia dokumentację pod adresem http://localhost:<port>/swagger.

Opis struktury projektu

1. **Warstwa modelu:**

   Reprezentuje dane w systemie, zawiera klasy encyjne i DTO.

2. **Warstwa kontrolera:**

   Obsługuje żądania HTTP, mapuje je na odpowiednie operacje.

3. **Warstwa usługi:**

   Implementuje logikę biznesową.

4. **Warstwa dostępu do danych:**

   Komunikuje się z bazą danych za pomocą Entity Framework Core.

Model Trening

- **Opis:** Reprezentuje pojedynczy trening pływacki.
- **Pola:**
    - Id (int): Klucz główny.
    - Data (DateTime): Data treningu.
    - Relacje:
        - Powiązanie jeden-do-wielu z tabelą ZadanieTreningowe.
- **Walidacje:** Data musi być przyszła lub bieżąca.

Model ZadanieTreningowe

- **Opis:** Reprezentuje pojedyncze zadanie w ramach treningu.
- **Pola:**
    - **Id (int):** Klucz główny.
    - **TreningId (int):** Klucz obcy do tabeli Trening.
    - **CzęśćTreningu (string):** Sekcja treningu (np. Rozgrzewka, Główna część, Schłodzenie).
    - **OpisZadania (string):** Szczegółowy opis zadania.
    - **Metraż (int):** Dystans w metrach dla zadania.
    - **TypZadania (enum):** Typ zadania (np. NN, RR, ANC, ANP, AEC1, AEC2).
- **Walidacje:**
    - **CzęśćTreningu:** Pole wymagane.
    - **OpisZadania:** Pole wymagane, maksymalna długość 500 znaków.
    - **Metraż:** Wartość musi być większa od 0.
    - **TypZadania:** Pole wymagane.

Model Obecność

- **Opis:** Informacje o obecności zawodników.
- **Pola:**
    - Id (int): Klucz główny.
    - TreningId (int): Klucz obcy do tabeli Trening.
    - ZawodnikId (int): Klucz obcy do tabeli Zawodnik.
    - Obecny (bool): Czy zawodnik był obecny.
- **Walidacje:** TreningId i ZawodnikId muszą istnieć w bazie.

Użytkownik

- **Opis:** Dane o użytkownikach systemu.
- **Pola:**
    - Id (int): Klucz główny.
    - Login (string): Login użytkownika.
    - Hasło (string): Hasło użytkownika.
    - Rola (enum): Rola (Zawodnik, Trener, Admin).

Kontrolery i metody

Kontroler Treningów

1. **GET /api/trenings**
    - **Metoda HTTP:** GET
    - **Opis:** Pobiera listę wszystkich treningów wraz z ich zadaniami.
    - **Zwraca:** Lista obiektów reprezentujących treningi, każdy z powiązanymi zadaniami.
2. **POST /api/trenings**
    - **Metoda HTTP:** POST
    - **Parametry:** Obiekt reprezentujący trening wraz z listą jego zadań.
    - **Opis:** Dodaje nowy trening z jego zadaniami.
    - **Zwraca:** Status operacji (np. sukces lub błąd walidacji).
3. **PUT /api/trenings/{id}**
    - **Metoda HTTP:** PUT
    - **Parametry:** Identyfikator treningu oraz obiekt zawierający zaktualizowane dane treningu i jego zadania.
    - **Opis:** Aktualizuje dane wybranego treningu oraz jego zadań.
    - **Zwraca:** Status operacji (np. sukces lub błąd).

Kontroler Obecności

1. **GET /api/obecnosci/{treningId}**
    - **Metoda HTTP:** GET
    - **Opis:** Pobiera listę obecności dla wybranego treningu.
    - **Zwraca:** Lista zawodników przypisanych do treningu z informacją, czy byli obecni.

2. **POST /api/obecnosci**
    - **Metoda HTTP:** POST
    - **Parametry:** Obiekt zawierający informacje o obecności zawodnika na danym treningu.
    - **Opis:** Dodaje informację o obecności zawodnika na treningu.
    - **Zwraca:** Status operacji.
3. **PUT /api/obecnosci/{id}**
    - **Metoda HTTP:** PUT
    - **Parametry:** Identyfikator obecności oraz zaktualizowane dane o obecności.
    - **Opis:** Aktualizuje informację o obecności zawodnika na treningu.
    - **Zwraca:** Status operacji.

Kontroler Użytkowników

1. **POST /api/uzytkownicy/rola**
    - **Metoda HTTP:** POST
    - **Parametry:** Identyfikator użytkownika oraz nowa rola.
    - **Opis:** Nadaje użytkownikowi określoną rolę w systemie.
    - **Zwraca:** Status operacji (np. sukces lub błąd walidacji).
2. **GET /api/uzytkownicy**
    - **Metoda HTTP:** GET
    - **Opis:** Pobiera listę wszystkich użytkowników.
    - **Zwraca:** Lista użytkowników z ich danymi (np. login, rola).
3. **DELETE /api/uzytkownicy/{id}**
    - **Metoda HTTP:** DELETE
    - **Parametry:** Identyfikator użytkownika.
    - **Opis:** Usuwa użytkownika z systemu.
    - **Zwraca:** Status operacji.

System użytkowników

- **Role:**
    - **Zawodnik:**
        - Widzi swoje treningi.
        - Widzi swoje dane o obecności.
        - Widzi kilometraż treningów.
    - **Trener:**
        - Widzi wszystkich zawodników.
        - Edytuje treningi.
        - Zarządza tabelą obecności.
    - **Admin:**
        - Zarządza bazą danych (dodawanie/usuwanie użytkowników, konfiguracja typów zadań itp.).
- **Funkcje:**
    - Mechanizm logowania i autoryzacji.
    - Powiązanie użytkownika z odpowiednimi danymi.

Najciekawsze funkcjonalności

1. **Analiza kilometrażu:**
    - Automatyczne obliczanie kilometrażu dla każdego zawodnika.
2. **Elastyczna konfiguracja:**
    - Admin może dostosowywać typy zadań i sekcje treningów.

Dodatkowe szczegóły i widoki zostaną uzupełnione w finalnej dokumentacji w postaci screenów z aplikacji.