Temat projektu: GymTracker

Aplikacja do monitorowania treningów siłowych i progresu sylwetki użytkownika z możliwością zapisywania się na wspólne treningi oraz testy siłowe ze znajomymi.

Autorzy: Jan Dyląg, Szymon Barczyk

Cele funkcjonalne:

1. **CRUD**:
    - Dodawanie/edycja/usuwanie:
        - użytkowników,
        - ćwiczeń
        - treningów (daty, ćwiczenia, serie, powtórzenia, ciężar)
        - pomiarów ciała (waga, biceps, klatka itd.)
        - wydarzeń specjalnych
2. **Operacje transakcyjne i kontrola równoczesnego dostępu**:
    - Rejestracja na **limitowane wydarzenia**, np. testy siły, wspólne treningi – z ograniczoną liczbą miejsc.
3. **Raportowanie / zapytania agregujące**:
    - Najczęściej wykonywane ćwiczenia użytkownika
    - Ilość treningów użytkownika w danym miesiącu
    - Ranking punktów użytkowników

Model danych:

- users: dane użytkownika
- exercises: dostępne ćwiczenia
- workouts: zapisy treningów z datą i seriami
- measurements: pomiary sylwetki
- events: limitowane wydarzenia (z możliwością rejestracji)

Technologie:

Warstwa	Technologia
Baza danych	**MongoDB**
Backend	Node.js + Express + Mongoose
Frontend (opcjonalnie.) -	Testy w Postmanie
Narzędzia	Docker, MongoDB Compass, DataGrip

Link do projektu na githubie: https://github.com/sbarczyk/PROJEKT-bazy_danych

Ogólny schemat bazy:

Nasza baza danych jest zorganizowana wokół kluczowych encji, które odzwierciedlają główne aspekty aplikacji fitness. Dzięki temu jest elastyczna i skalowalna, co ułatwia dodawanie nowych funkcji. Oto główne modele:

- **User (Użytkownik)**: Zarządza danymi użytkowników, w tym ich rolami i punktami.
- **Exercise (Ćwiczenie)**: Przechowuje szczegóły ćwiczeń, umożliwiając ich tworzenie i modyfikację.
- **Workout (Trening)**: Rejestruje sesje treningowe, łącząc ćwiczenia z użytkownikami.
- **Measurement (Pomiar)**: Zapisuje dane o pomiarach ciała użytkowników.
- **Event (Wydarzenie)**: Organizuje różne wydarzenia, od treningów grupowych po testy siłowe.
- **EventLog (Dziennik Wydarzeń)**: Śledzi akcje związane z wydarzeniami, jak rejestracje.

Dlaczego taka struktura?

Zrezygnowaliśmy ze zagnieżdżeń w MongoDB, aby zachować przejrzystość i elastyczność struktury danych. Oddzielenie encji ułatwia zarządzanie relacjami, aktualizacjami i skalowaniem aplikacji, a także ogranicza ryzyko problemów z wydajnością i limitem rozmiaru dokumentów.

Dodatkowe komentarze:

W modelach zastosowano **ścisłą walidację** (enum, min, required, default, trim), co zmniejsza ryzyko błędów i poprawia jakość danych.

Wszystkie operacje (CRUD) zostały pokryte odpowiednimi endpointami REST API oraz zabezpieczone middleware JWT. Operacje zostały dokładnie przetestowane w Postmanie, z uwzględnieniem:

- obsługi błędów (404, 401, 400),
- walidacji ID (ObjectId.isValid()),
- sprawdzania istnienia obiektów przed operacjami (np. Workout sprawdza, czy Exercise istnieje),
- kontroli dostępu (admin / użytkownik).

Plik konfiguraycjny POSTMANA znajduje się w repozytorium projektu.

Szczegółowe omówienie modeli:

USER:

Field	Type	Required	Default / Constraints	Opis
name	String	tak	trim	Imię i nazwisko użytkownika
email	String	tak	unique, lowercase	Login / identyfikator e-mail
password	String	tak	minlength 6	Haszowane bcryptem hasło
isAdmin	Boolean	nie	false	Flaga uprawnień administratora
points	Number	nie	0	Punkty gamifikacyjne
createdAt	Date	nie	Date.now	Znacznik utworzenia dokumentu

EXERCISE:

Field	Type	Required	Default / Constraints	Opis
name	String	tak	trim	Nazwa ćwiczenia
category	String	tak	enum (chest, back, legs…)	Kategoria ruchu
muscleGroups	[String]	tak	≥ 1 element	Zaangażowane grupy mięśniowe
description	String	nie	–	Opis słowny
instructions	[String]	nie	–	Krok-po-kroku / wskazówki
createdBy	ObjectId → User	nie	-	
isPublic	Boolean	nie	true	Widoczne dla innych użytkowników
createdAt	Date	nie	Date.now	Data dodania ćwiczenia

WORKOUT:

Field	Type	Required	Default / Constraints	Opis
user	ObjectId → User	tak	–	Właściciel treningu
name	String	tak	trim	Tytuł / opis sesji
date	Date	tak	Date.now	Data wykonania
exercises	[Sub-doc]	tak	≥ 1, walidacja istnienia	Lista ćwiczeń + setów
duration	Number	nie	min 0	Czas trwania [min]
notes	String	nie	trim	Notatki ogólne
createdAt	Date	nie	Date.now	Znacznik utworzenia

MEASUREMENT:

Field	Type	Required	Default / Constraints	Opis
user	ObjectId → User	tak	–	Właściciel pomiaru
date	Date	tak	Date.now	Data pomiaru
weight	Number	nie	min 0	Masa ciała [kg]
bodyFat	Number	nie	0 – 100	% tkanki tłuszczowej
measurements	Embedded object	nie	–	Obwody: chest, waist, hips, biceps L/R, thigh L/R, neck
notes	String	nie	trim	Uwagi własne
createdAt	Date	nie	Date.now	Znacznik utworzenia

EVENT:

Field	Type	Required	Default / Constraints	Opis
title	String	tak	trim	Nazwa wydarzenia
description	String	nie	trim	Opis szczegółowy
type	String	tak	enum (strength_test, group_training, personal_training)	Rodzaj eventu
trainer	ObjectId → User	nie	–	Prowadzący (opcjonalnie)
date	Date	tak	–	Termin
duration	Number	tak	minuty	Czas trwania
maxParticipants	Number	tak	≥ 1	Limit miejsc
participants	[Sub-doc]	nie	unique index	Uczestnicy + registeredAt
price	Number	nie	0	Koszt uczestnictwa
location	String	nie	trim	Miejsce
isActive	Boolean	nie	true	Aktywne / archiwum
createdAt	Date	nie	Date.now	Data utworzenia

EVENTLOG:

Field	Type	Required	Default / Constraints	Opis
user	ObjectId → User	tak	–	Kto wykonał akcję
event	ObjectId → Event	tak	–	Id wydarzenia
action	String	tak	enum (REGISTER, UNREGISTER)	Typ operacji
pts	Number	nie	0	Punkty przyznane/odjęte
at	Date	nie	Date.now	Znacznik czasu

Krótki opis kontrolerów:

ExercisesController:
Obsługuje katalog ćwiczeń: pobiera pełną listę, a tworzenie, edycja i usuwanie są zastrzeżone dla administratora.

WorkoutsController:
Prowadzi dziennik treningów użytkownika. Przy zapisie sprawdza, czy wszystkie ID ćwiczeń istnieją, a przy odczycie może zwrócić tylko treningi właściciela (lub wszystkie, jeśli żąda tego admin).

MeasurementsController:
Zarządza pojedynczym wpisem pomiarów ciała na użytkownika: waga, obwody, % tłuszczu. Użytkownik może dodać lub edytować tylko swój rekord; admin ma dostęp do wszystkich.

EventsController:
Tworzy wydarzenia (treningi grupowe, testy), prowadzi listę uczestników i rozdziela punkty gamifikacyjne. Rejestracja/wyrejestrowanie odbywa się w transakcji, żeby jednocześnie zaktualizować event, tabelę EventLog i punkty użytkownika.

ReportsController:
dostarcza zagregowane statystyki, np. tablicę liderów wg punktów czy najczęściej wykonywane ćwiczenia danego miesiąca, opierając się na potokach agregacyjnych MongoDB.

UsersController:
udostępnia podstawowy CRUD na kolekcji użytkowników.

Dostępne ENDPOINTY:

Workouts (Treningi)

- **GET** /api/workouts/ - Pobierz wszystkie treningi (wymagana autoryzacja)
- **GET** /api/workouts/:id - Pobierz konkretny trening po ID (wymagana autoryzacja)
- **POST** /api/workouts/ - Utwórz nowy trening (wymagana autoryzacja)
- **PATCH** /api/workouts/:id - Zaktualizuj trening po ID (wymagana autoryzacja)
- **DELETE** /api/workouts/:id - Usuń trening po ID (wymagana autoryzacja)

Users (Użytkownicy)

- **POST** /api/users/login - Logowanie użytkownika
- **POST** /api/users/register - Rejestracja użytkownika
- **POST** /api/users/ - Utwórz nowego użytkownika (wymagana autoryzacja, tylko admin)
- **GET** /api/users/ - Pobierz wszystkich użytkowników (wymagana autoryzacja, tylko admin)
- **PATCH** /api/users/:id - Zaktualizuj dane użytkownika (wymagana autoryzacja, właściciel lub admin)
- **DELETE** /api/users/:id - Usuń użytkownika (wymagana autoryzacja, właściciel lub admin)

Reports (Raporty)

- **GET** /api/reports/leaderboard - Pobierz ranking
- **GET** /api/reports/top-sets/:userId - Pobierz ćwiczenia z największą ilością wykonanych serii dla użytkownika (wymagana autoryzacja)
- **GET** /api/reports/workout-count/:userId - Pobierz liczbę treningów na miesiąc dla użytkownika (wymagana autoryzacja)

Events (Wydarzenia)

- **GET** /api/events/ - Pobierz wszystkie wydarzenia (publiczne)
- **GET** /api/events/:id - Pobierz konkretne wydarzenie po ID (publiczne)
- **POST** /api/events/ - Utwórz nowe wydarzenie (wymagana autoryzacja, tylko admin)
- **PATCH** /api/events/:id - Zaktualizuj wydarzenie po ID (wymagana autoryzacja, tylko admin)
- **DELETE** /api/events/:id - Usuń wydarzenie po ID (wymagana autoryzacja, tylko admin)
- **POST** /api/events/:id/register - Zarejestruj się na wydarzenie (wymagana autoryzacja)
- **POST** /api/events/:id/unregister - Wyrejestruj się z wydarzenia (wymagana autoryzacja)

Exercises (Ćwiczenia)

- **GET** /api/exercises/ - Pobierz wszystkie ćwiczenia (publiczne)
- **GET** /api/exercises/:id - Pobierz konkretne ćwiczenie po ID (publiczne)
- **POST** /api/exercises/ - Utwórz nowe ćwiczenie (wymagana autoryzacja, tylko admin)
- **PATCH** /api/exercises/:id - Zaktualizuj ćwiczenie po ID (wymagana autoryzacja, tylko admin)

Zastosowanie technologie oraz wybrane możliwości technologiczne:

MongoDB

- **Elastyczna struktura dokumentów**:
  - **Gdzie**: W modelach danych w katalogu `src/models/`.
  - **Jak**: Modele takie jak `User`, `Event`, `Workout` są przechowywane w formacie JSON, co umożliwia łatwe dostosowywanie struktury danych bez potrzeby migracji schematów.
- **Indeksowanie**:
  - **Gdzie**: Wyszukiwania w kontrolerach, np. `src/controllers/events.js`.
  - **Jak**: Indeksowanie jest używane do przyspieszenia operacji wyszukiwania i sortowania, co jest kluczowe dla wydajności aplikacji.
- **Agregacje**:
  - **Gdzie**: W kontrolerach raportów, np. `src/controllers/reports.js`.
  - **Jak**: Umożliwiają wykonywanie złożonych operacji przetwarzania danych, takich jak filtrowanie, grupowanie i sortowanie, bezpośrednio na poziomie bazy danych.
- **Transakcje**:
  - **Gdzie**: W operacjach rejestracji i wyrejestrowania z wydarzeń w `src/controllers/events.js`.
  - **Jak**: Zapewniają atomowość operacji, co jest kluczowe dla spójności danych, szczególnie przy modyfikacjach wielu dokumentów.

Inne Technologie

- **Express.js**:
  - **Gdzie**: W całej aplikacji, szczególnie w plikach `src/routes/`.
  - **Jak**: Używany jako framework do tworzenia serwera HTTP i zarządzania trasami API. Umożliwia szybkie tworzenie endpointów i obsługę żądań HTTP.
- **Mongoose**:
  - **Gdzie**: W modelach danych w katalogu `src/models/`.
  - **Jak**: Używany jako ODM (Object Data Modeling) do MongoDB, zapewnia walidację schematów, relacje między dokumentami i łatwiejszą manipulację danymi.
- **JSON Web Token (JWT)**:
  - **Gdzie**: W pliku `src/middleware/auth.js`.
  - **Jak**: Używany do autoryzacji użytkowników. Tokeny są generowane podczas logowania i weryfikowane przy każdym żądaniu do chronionych zasobów.
- **bcrypt.js**:
  - **Gdzie**: W pliku `src/routes/users.js`.
  - **Jak**: Używany do hashowania haseł użytkowników przed ich zapisaniem w bazie danych, co zwiększa bezpieczeństwo aplikacji.
- **Node.js**:
  - **Gdzie**: Cała aplikacja.
  - **Jak**: Środowisko uruchomieniowe dla JavaScript, które umożliwia uruchamianie serwera i obsługę asynchronicznych operacji.
- **Docker**:
  - Całe środowisko (MongoDB z repliką) można uruchomić jednym poleceniem, co ułatwia testowanie i deployment.
  - Dane można szybko odtworzyć z dumpa (mongorestore), co umożliwia spójne testy.