1. API

1.1. Autoryzacja i użytkownicy (Authentication & Users)

POST /register

- Tworzy nowe konto użytkownika lub trenera w systemie.
- Weryfikuje poprawność podanych danych.
- Rejestruje użytkownika w Supabase Auth.
- Dodaje dane użytkownika do tabeli users lub trainers.
- W przypadku sukcesu zwraca ID nowo utworzonego użytkownika oraz status 201.

POST /login

- Loguje użytkownika na podstawie podanego adresu e-mail i hasła.
- Weryfikuje poprawność danych logowania w Supabase Auth.
- Pobiera ID użytkownika z tabeli users lub trainers w zależności od roli.
- Przechowuje user_id w localStorage przeglądarki, aby umożliwić łatwiejszą obsługę autoryzacji w sesji.
- W przypadku sukcesu zwraca token sesji.
- W przypadku błędu zwraca status 401.

POST /logout

- Wylogowuje użytkownika i usuwa token sesji.
- Usuwa user_id z localStorage.
- W przypadku sukcesu zwraca status 200.

GET /user

- Pobiera dane aktualnie zalogowanego użytkownika.
- Wymaga autoryzacji.
- W przypadku sukcesu zwraca dane użytkownika.
- Jeśli użytkownik nie jest zalogowany, zwraca status 401.

1.2. Trenerzy (Trainers)

GET /trainers

- Pobiera listę wszystkich trenerów.
- Obsługuje filtrowanie według lokalizacji, siłowni, specjalizacji, doświadczenia.
- W odpowiedzi zwraca:
    - Tablicę obiektów zawierających dane trenerów.

GET /trainers/:id

- Pobiera dane konkretnego trenera na podstawie jego ID.
- Jeśli trener nie istnieje, zwraca status 404.

POST /trainers

- Tworzy nowy profil trenera na podstawie podanych danych.
- W przypadku sukcesu zwraca ID nowo utworzonego trenera oraz status 201.

PUT /trainers/:id

- Aktualizuje dane trenera o podanym ID.
- Weryfikuje, czy trener istnieje.
- W przypadku sukcesu zwraca status 200.
- Jeśli trener nie istnieje, zwraca status 404.

1.3. Ćwiczenia i dziennik treningowy (Workouts)

GET /workouts?date=YYYY-MM-DD

- Pobiera listę ćwiczeń zaplanowanych na dany dzień.
- Filtruje po dacie (workouts.date) oraz user_id, który jest pobierany z localStorage. Wyniki są sortowane rosnąco po id, aby zapewnić logiczny układ ćwiczeń w interfejsie użytkownika.
- W odpowiedzi zwraca:
    - Tablicę ćwiczeń przypisanych do użytkownika.

POST /workouts

- Dodaje nowe ćwiczenie do dziennika użytkownika.
- Weryfikuje poprawność danych wejściowych.
- W przypadku sukcesu zwraca ID nowego ćwiczenia oraz status 201.

PUT /workouts/:id

- Aktualizuje dane ćwiczenia.
- Weryfikuje, czy ćwiczenie istnieje.
- Jeśli ćwiczenie nie istnieje, zwraca status 404.

DELETE /workouts/:id

- Usuwa ćwiczenie o podanym ID.
- Jeśli ćwiczenie nie istnieje, zwraca status 404.

PATCH /workouts/:id/complete

- Oznacza ćwiczenie jako wykonane (is_completed = true).
- Jeśli ćwiczenie nie istnieje, zwraca status 404.

1.5. Kalendarz (Calendar)

GET /calendar

- Pobiera kalendarz treningowy użytkownika.

- Wykorzystuje dane z workouts, aby zaznaczyć dni, w których użytkownik ma zaplanowane ćwiczenia.
- Kalendarz pobiera ćwiczenia na podstawie workouts.date, dzięki czemu można przeglądać przeszłe oraz przyszłe plany treningowe.
- W przypadku sukcesu zwraca dane potrzebne do wyświetlenia treningów w kalendrzu.

1.6. Rejestracja i logowanie trenerów

POST /register/trainer

- Rejestruje nowego trenera w Supabase Auth.
- Tworzy wpis w tabeli trainers.

POST /login/trainer

- Loguje trenera na podstawie e-maila i hasła.
- Weryfikuje jego istnienie w trainers.

1.7. Rejestracja i logowanie użytkowników

POST /register/user

- Rejestruje nowego użytkownika w Supabase Auth.
- Tworzy wpis w tabeli users.

POST /login/user

- Loguje użytkownika na podstawie e-maila i hasła.
- Weryfikuje jego istnienie w users.

Uwierzytelnianie za pomocą Supabase Auth

- Mechanizm authentication wykorzystuje Supabase Auth do weryfikacji użytkownika.
- Użytkownik loguje się przy użyciu adresu e-mail i hasła. Po poprawnej autoryzacji, Supabase zwraca identyfikator użytkownika (user_id), który jest przechowywany w localStorage przeglądarki.
- W przypadku błędnych danych logowania, zwracany jest status 401.

Autoryzacja na podstawie roli użytkownika

- Mechanizm autoryzacji w aplikacji opiera się na rozróżnieniu dwóch typów kont: użytkownika i trenera.
- Podczas rejestracji użytkownik jest przypisywany do tabeli users lub trainers.
- System sprawdza, czy użytkownik istnieje w odpowiedniej tabeli. Dostęp do poszczególnych zasobów jest ograniczony na podstawie roli użytkownika.
- Jeśli użytkownik nie ma dostępu do danego zasobu, zwracany jest status 403.

Walidacja danych wejściowych

- Walidacja danych w aplikacji opiera się na sprawdzeniu poprawności danych przy rejestracji. Weryfikowane są wymagane pola, takie jak adres e-mail, hasło i podstawowe dane użytkownika.
- Błędne lub niepełne dane zwracają status 422 z komunikatem o błędzie.

Zabezpieczenie endpointów

- Po poprawnej weryfikacji dane użytkownika są pobierane na podstawie user_id przechowywanego w sesji.

Bezpieczeństwo w aplikacji

Supabase Auth

- Supabase obsługuje zarządzanie użytkownikami.
- Logowanie, rejestracja i wylogowanie użytkowników odbywa się za pomocą metod Supabase Auth.

HTTPS

Zaleca się korzystanie z HTTPS w środowisku produkcyjnym.

2. Baza danych aplikacji fitness

Relacje w bazie danych

1. **Users i Trainers**: Użytkownicy mogą przeglądać profile trenerów i wybierać odpowiedniego dla siebie.

2. **Users i Auth.Users**: Każdy użytkownik w bazie ma swoje konto powiązane z systemem autoryzacji.

3. **Trainers i Auth.Users**: Każdy trener w bazie ma swoje konto powiązane z systemem autoryzacji.

4. **Workouts i Users**: Każdy wpis w dzienniku treningowym jest przypisany do konkretnego użytkownika.

Bezpieczeństwo

Mechanizm autoryzacji i uwierzytelniania:

- System Supabase Auth jest wykorzystywany do zarządzania sesjami użytkowników.

- Uwierzytelnianie odbywa się za pomocą e-maila i hasła.

- Dane logowania są przechowywane w tabeli auth.users, a dodatkowe dane użytkownika w tabeli users.

- Dane logowania są przechowywane w tabeli auth.users, a dodatkowe dane trenera w tabeli trainers.
- Po zalogowaniu użytkownik otrzymuje token sesyjny, który umożliwia dostęp do zasobów aplikacji.

Zarządzanie rolami i dostępem:

- System umożliwia przypisanie użytkowników do ról user i trainer, co wpływa na dostęp do funkcji aplikacji.
- Tabela trainers przechowuje informacje o trenerach i pozwala na ich filtrowanie według specjalizacji, lokalizacji i doświadczenia.
- Każdy użytkownik może przeglądać listę trenerów, a trenerzy mogą edytować swoje profile.
- Zalogowani użytkownicy mogą używać dziennika aktywności

Technologia bazy danych

Aplikacja korzysta z Supabase (PostgreSQL) jako bazy danych, zapewniając:

- Relacyjne przechowywanie danych, co pozwala na efektywne zarządzanie użytkownikami, trenerami i ćwiczeniami.
- Zintegrowane uwierzytelnianie, co umożliwia łatwe zarządzanie dostępem do aplikacji.
- Realtime API, które umożliwia aktualizację danych w czasie rzeczywistym.

Tabele

- Tabela: trainers

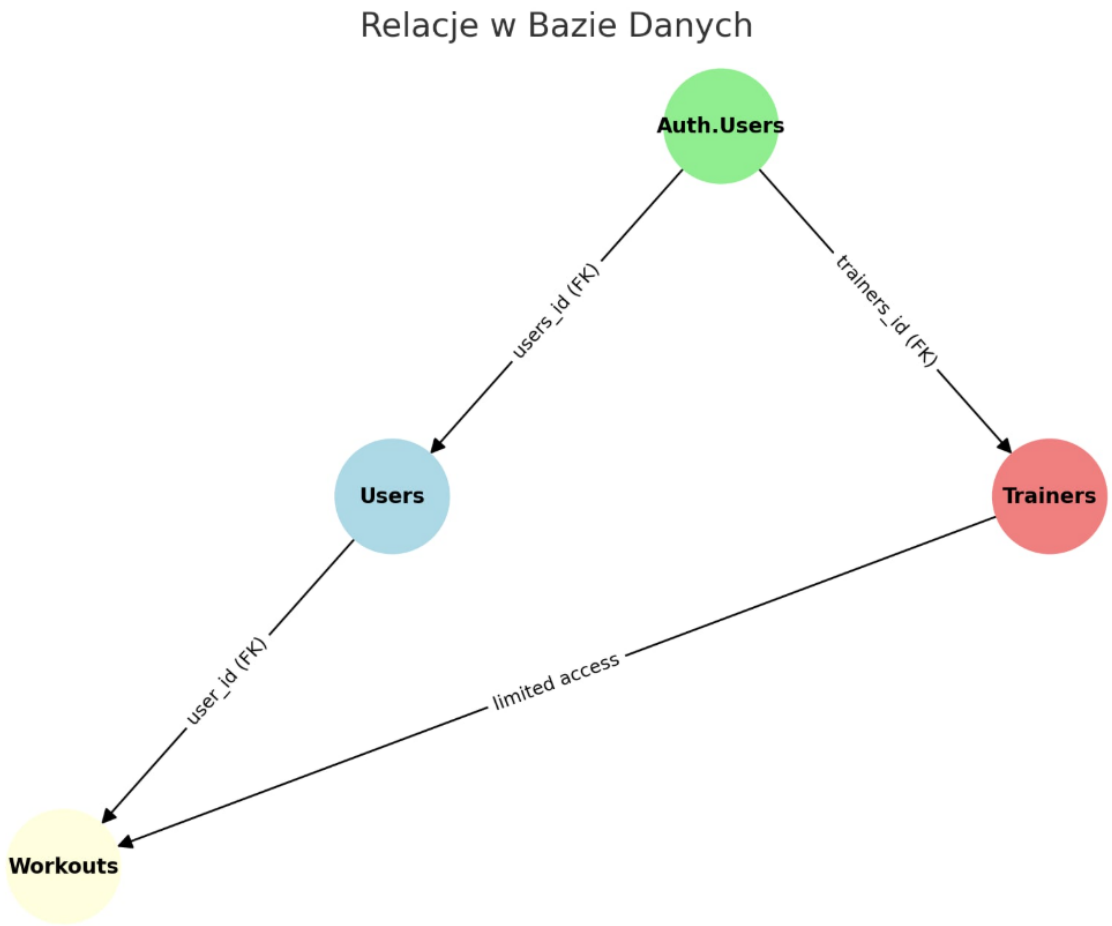
Nazwa pola	Typ	Wymagane	Opis
`id`	UUID	Tak	Unikalny identyfikator trenera
`first_name`	Text	Tak	Imię trenera
`last_name`	Text	Tak	Nazwisko trenera
`email`	Text	Tak	Adres e-mail trenera (unikalny)
`location`	Text	Nie	Miasto trenera
`gym_location`	Text	Nie	Siłownia, w której pracuje trener
`specialization`	Text	Nie	Specjalizacja trenera
`experience`	Int4	Nie	Liczba lat doświadczenia

- Tabela: users

Nazwa pola	Typ	Wymagane	Opis
`id`	UUID	Tak	Unikalny identyfikator użytkownika
`email`	Text	Tak	Adres e-mail użytkownika (unikalny)
`first_name`	Text	Tak	Imię użytkownika
`last_name`	Text	Tak	Nazwisko użytkownika
`location`	Text	Nie	Miasto użytkownika
`gym_location`	Text	Nie	Siłownia, do której uczęszcza użytkownik
`created_at`	Timestamp	Tak	Data utworzenia rekordu, domyślnie `now()`

- Tabela: workouts

Nazwa pola	Typ	Wymagane	Opis
`id`	int8	Tak	Unikalny identyfikator wpisu w dzienniku ćwiczeń
`date`	Date	Tak	Data wykonania ćwiczenia
`exercise_name`	Text	Tak	Nazwa ćwiczenia
`sets`	int8	Tak	Liczba serii
`reps`	int8	Tak	Liczba powtórzeń
`weight`	int8	Tak	Obciążenie (waga w kg)
`personal_recc`	Bool	Nie	Czy ćwiczenie jest rekordem osobistym
`is_completed`	Bool	Nie	Czy ćwiczenie zostało ukończone (domyślnie `false`)
`user_id`	UUID	Tak	Identyfikator użytkownika, do którego należy wpis



Procedury

W aplikacji wykorzystywane są głównie operacje CRUD (Create, Read, Update, Delete) na tabelach:

- **users** (użytkownicy)
- **trainers** (trenerzy)
- **workouts** (dziennik ćwiczeń)

Przykładowe operacje na danych:

- **Pobieranie listy treningów dla użytkownika:**

```
const user_id = localStorage.getItem('user_id');
const { data, error } = await supabase
    .from('workouts')
    .select('*')
    .eq('user_id', user_id)
    .order('id', { ascending: true });
```

W większości przypadków operacje na danych są wykonywane bezpośrednio z poziomu kodu aplikacji, dlatego nie ma potrzeby tworzenia klasycznych procedur składowanych w bazie.

Wyzwalacze

- Aplikacja korzysta z PostgreSQL (Supabase), co pozwala na wykorzystanie tradycyjnych wyzwalaczy (triggers) oraz funkcji bazodanowych.
- Supabase wspiera również Realtime Database, co pozwala na nasłuchiwanie zmian w czasie rzeczywistym.

3. Wymagania Systemowe

Wymagania systemowe

System operacyjny:

- **Serwer:** Linux (np. Ubuntu 20.04) lub Windows Server 2019.
- **Klient:** Dowolny system operacyjny obsługujący nowoczesne przeglądarki (Windows 10+, macOS, Linux).

Dodatkowe oprogramowanie:

- **Backend:** Node.js w wersji 16.x lub nowszej.
- **Baza danych:** Supabase (PostgreSQL jako backend bazodanowy).
- **Przeglądarka:** Chrome (zalecana), Firefox, Edge (najnowsze wersje).

Wymagania sprzętowe:

Minimalne wymagania serwera:

- **Procesor:** 2-rdzeniowy, 2.5 GHz lub szybszy.
- **Pamięć RAM:** 4 GB (zalecane 8 GB dla większych obciążeń).
- **Przestrzeń dyskowa:** Minimum 20 GB na potrzeby bazy danych i logów aplikacji.

Minimalne wymagania klienta:

- **Przeglądarka:** Wspierająca nowoczesne standardy (np. Chrome, Firefox, Edge).
- **Rozdzielczość ekranu:** Minimum 1366x768.
- **Połączenie internetowe:** Stabilne, 10 Mbps lub szybsze.

4. Procedura uruchomienia środowiska

Środowisko developerskie:

Klonowanie repozytoriów:

- Pobierz kod źródłowy z repozytorium Git.

Instalacja zależności:

- uruchom polecenie: `npm install`

Uruchomienie środowiska developerskiego:

- uruchom polecenie: `npm run dev`

Środowisko produkcyjne:

Kompilacja aplikacji:

- W Frontend (React) wpisz komendę: `npm run build`

8. Proces budowy nowej wersji

- **Rozwój i testy** – Implementacja nowych funkcji, testy jednostkowe i integracyjne.
- **Budowanie aplikacji**
  - ✓ **Frontend**: Generowanie wersji produkcyjnej : `npm run build`.
  - ✓ **Backend**: Działa na **Supabase**, wymaga jedynie konfiguracji.
- **Monitorowanie** – Analiza logów w **Supabase Dashboard**, wsparcie użytkowników (opinie).
- **Baza danych** – W razie potrzeby aktualizacja przez **Supabase SQL Editor**.