UNIWERSYTET RZESZOWSKI

WYDZIAŁ NAUK ŚCISŁYCH I TECHNICZNYCH

INSTYTUT INFORMATYKI

Oliwier Hędrzak, Michał Janik

134913, 134915

Informatyka

Dziennik treningowy AwareFit – dokumentacja projektowa

Praca projektowa

Praca wykonana pod kierunkiem
dr inż. Piotr Grochowalski

Rzeszów 2026

Spis treści

1. Wprowadzenie

Projekt **AwareFit** powstał z przekonania, że większość dostępnych na rynku aplikacji treningowych to jedynie cyfrowe wersje papierowego notesu. Choć pozwalają one zapisać wykonane serie, rzadko dają użytkownikowi realną wartość dodaną w postaci analizy. W dzisiejszych czasach osoby trenujące siłowo wiedzą, że progres nie bierze się z przypadku, ale z precyzyjnego zarządzania objętością (tonażem), intensywnością oraz regeneracją.

Głównym celem AwareFit było stworzenie inteligentnego systemu, który nie tylko archiwizuje jednostki treningowe, ale przede wszystkim je „rozumie". Na rynku wciąż brakuje narzędzi, które potrafiłyby wyciągać wnioski z historii treningów i zamieniać surowe liczby w konkretne wskazówki. System AwareFit automatycznie przelicza progresję, analizuje tonaż i identyfikuje aktualny etap planu użytkownika. To aplikacja stworzona dla tych, którzy chcą trenować w pełni świadomie, co zresztą podkreśla sama jej nazwa. System automatyzuje kluczowe procesy np. oblicza całkowitą objętość sesji, porównuje ją z wynikami z poprzednich tygodni, a dzięki autorskim algorytmom zaimplementowanym po stronie bazy danych, samodzielnie identyfikuje rodzaj realizowanego systemu treningowego.

W warstwie technologicznej projekt skupia się na maksymalnym wykorzystaniu możliwości relacyjnej bazy danych PostgreSQL. Logika aplikacji nie spoczywa wyłącznie na barkach interfejsu, lecz jest głęboko zakorzeniona w procedurach składowanych i funkcjach bazodanowych (pgSQL). Takie podejście gwarantuje spójność danych, szybkość obliczeń oraz skalowalność systemu. AwareFit to kompletne środowisko, które łączy świat bazy danych, logiki serwerowej PHP oraz nowoczesnego interfejsu użytkownika, tworząc spójne narzędzie do świadomego kształtowania własnej formy fizycznej.

2. Specyfikacja

Projektowanie systemu **AwareFit** rozpoczęto od analizy tego, jak naprawdę wygląda trening siłowy i co tak naprawdę pomoże użytkownikowi w jego przygodzie z siłownią. Kluczowe było to, aby nie tworzyć kolejnego notesu do zapisywania treningu, lecz takiego dziennika, który faktycznie będzie wspierał użytkownika i prowadził go przez swoją drogę do upragnionej formy. Poniżej znajduje się opis, w jaki sposób rzeczywistość została przeniesiona z siłowni do logiki systemu bazodanowego. Ważnym aspektem była elastyczność danych, co pozwala na swobodny rozwój aplikacji. Na bazie własnych obserwacji oraz doświadczeniu utworzony został koncept, który zawiera odpowiednią strukturę oraz algorytmy w języku pgSQL stanowiące serce całego systemu.

2.1. Scenariusz interakcji z systemem

Aby w pełni zrozumieć strukturę projektowanej bazy danych, należy prześledzić typową ścieżkę użytkownika, która determinuje sposób przepływu informacji w systemie:

1. **Rejestracja/Logowanie:** Pierwszym krokiem interakcji z systemem jest utworzenie unikalnego konta użytkownika. Logowanie to mechanizm, który pozwala systemowi wyizolować dane konkretnej osoby. Dzięki temu, po podaniu danych uwierzytelniających, baza filtruje setki rekordów, aby wyświetlić tylko te trendy i statystyki, które należą do danego użytkownika, zapewniając mu prywatność i spersonalizowany widok dashboardu.

2. **Przygotowanie i cel:** Użytkownik rozpoczyna korzystanie z aplikacji od wprowadzenia swojego celu oraz kluczowych pomiarów ciała, co pozwala na zdefiniowanie momentu, w którym się znajduje. Dzięki zaawansowanym algorytmom po stronie bazy danych możliwe jest obliczenie zapotrzebowania kalorycznego, makroskładników, czy chociażby procenta tkanki tłuszczowej w ciele. Zarówno cel jak i pomiary mogą być aktualizowane przez użytkownika w przeznaczonym do tego panelu.

3. **Rozpoczęcie sesji i inteligentne wsparcie:** Po przyjściu na siłownię, użytkownik inicjuje nową jednostkę treningową. W tym momencie system tworzy w bazie danych unikalny rekord treningu przypisany do zalogowanej osoby i rozpoczyna pomiar czasu trwania sesji. Użytkownik wybiera partię mięśniową, a następnie konkretne ćwiczenie z nią powiązane.

   Kluczowym elementem systemu jest wsparcie w czasie rzeczywistym. Podczas wpisywania danych, użytkownik widzi podpowiedzi w polach formularza, które przypominają mu o użytym ciężarze i liczbie powtórzeń z ostatniego treningu tego samego ćwiczenia. Dzięki temu nie musi on co chwilę spoglądać w historię zapisanych treningów. Warto dodać, że system analizuje treningi wstecz i wyświetla rekomendację, czy użytkownik powinien zwiększyć obciążenie, czy pozostać przy obecnym.

   Po każdej serii użytkownik zapisuje swoje osiągi, budując strukturę treningu krok po kroku, aż do jego zakończenia. Cała ta logika jest wynikiem zaawansowanych algorytmów działających bezpośrednio po stronie silnika bazy danych.

4. **Analiza potreningowa:** Po zakończeniu i zapisaniu sesji, system natychmiastowo przetwarza zebrane informacje, aby zaktualizować centralny panel sterowania – Dashboard. W tym momencie dane z tabel dotyczących treningów są agregowane przez funkcje bazodanowe, które przeliczają całkowitą objętość (tonaż) z ostatnich 7 dni i porównują ją z analogicznym okresem w przeszłości.

   Dzięki temu, przy kolejnym uruchomieniu aplikacji, użytkownik od razu widzi dynamiczne wskaźniki postępu, takie jak procentowy wzrost objętości czy aktualną passę treningową (streak). System generuje również interaktywne wykresy progresji siłowej dla wybranych ćwiczeń, pozwalając na wizualną ocenę progresu. Dodatkowo do systemu zaimplementowany został algorytm, który samodzielnie rozpoznaje rodzaj treningu, na bazie wykonanych jednostek w skali tygodnia.

   W tym samym miejscu dane treningowe spotykają się z danymi dietetycznymi. Dashboard wyświetla podsumowanie spożytych kalorii i makroskładników w formie czytelnych kart, co pozwala użytkownikowi błyskawicznie ocenić, czy jego regeneracja i odżywianie są spójne z intensywnością wykonanej pracy na siłowni. Cały ten proces sprawia, że AwareFit przestaje być tylko dziennikiem, a staje się osobistym analitykiem sportowym dostępnym na żądanie.

5. **Panel progresu:** Kluczowym elementem AwareFit jest zaawansowany panel progresu, który służy do monitorowania rozwoju. Najważniejszym wskaźnikiem jest tutaj objętość, która pokazuje użytkownikowi jego tendencję progresu. Dzięki temu możliwe jest, aby użytkownik doszedł do wniosku, czy przyjęta przez niego metoda treningowa przynosi zamierzone efekty.

   Baza danych analizuje wszystkie treningi z ostatnich 7 dni i wylicza procentowy udział każdej partii mięśniowej w całym planie. Jeśli użytkownik zaniedba jakąś grupę (np. pominie trening nóg), system natychmiast to wyłapie i wyświetli komunikat o pominiętej partii. Ma to na celu zapobieganie dysproporcjom sylwetkowym i kontuzjom. Dodatkowo, na podstawie historii serii, aplikacja estymuje aktualny rekord maksymalny (1RM) w najważniejszych bojach, takich jak wyciskanie, przysiad czy martwy ciąg, co pozwala na bieżąco śledzić wzrost czystej siły bez konieczności ryzykownego sprawdzania jej w każdej sesji.

Podsumowując, powyższy scenariusz pokazuje, że system **AwareFit** został zaprojektowany z myślą o rzeczywistym problemie. Struktura została zaprojektowana w taki sposób, aby możliwe było prowadzenie użytkownika przez jego przygodę z siłownią. Od momentu pomiarów ciała, przez wyliczenie diety oraz zapisywanie treningu, po analizę danych i końcowe wskazówki i kluczowe wartości. Dzięki temu użytkownik otrzymuje intuicyjny w obsłudze dziennik treningowy, który dba o jego progres, zdrowie i buduje jego świadomość ciała.

2.2. Użyte technologie

Realizacja systemu **AwareFit** opiera się na sprawdzonych rozwiązaniach, które zapewniają stabilność przetwarzania danych oraz responsywność interfejsu użytkownika:

- **PostgreSQL (Silnik bazy danych):** Centralny element systemu. Wybrany ze względu na zaawansowane wsparcie dla języka **pgSQL**, który pozwolił na przeniesienie logiki analitycznej i algorytmów progresji bezpośrednio na stronę serwera bazy danych.
- **PHP (Backend):** Odpowiada za bezpieczną komunikację między interfejsem użytkownika a bazą danych, obsługę sesji oraz logikę przesyłania formularzy.
- **HTML5 / CSS3 / JavaScript (Frontend):** Wykorzystane do budowy intuicyjnego interfejsu oraz dynamicznych dashboardów. Za warstwę wizualną wykresów progresji odpowiada biblioteka **Chart.js**, która w czasie rzeczywistym renderuje dane dostarczane z bazy. Projektowanie GUI opierało się o metodę mobile-first.
- **XAMPP:** Zintegrowany pakiet oprogramowania, który posłużył jako lokalne środowisko deweloperskie. Zapewnił on niezbędne komponenty, takie jak serwer Apache do obsługi skryptów PHP.

3. Opis struktury bazy danych

3.1. Zdefiniowane struktury tabel

Tabela: users

Tabela ta stanowi centralny punkt bazy danych, przechowując dane identyfikacyjne użytkowników oraz informacje niezbędne do procesów autoryzacji i personalizacji.

Nazwa kolumny	Typ danych	Opis i ograniczenia
id (PK)	serial	Unikalny identyfikator użytkownika.
username	varchar(50)	Nazwa użytkownika wykorzystywana do logowania.
password	varchar(255)	Zahaszowane hasło użytkownika.
email	varchar(100)	Adres e-mail przypisany do konta.
first_name	varchar(50)	Imię użytkownika.
last_name	varchar(50)	Nazwisko użytkownika.
gender	text	Płeć użytkownika.

Tabela 3.1. Struktura tabeli users

Tabela: body_measurements

Przechowuje historię pomiarów antropometrycznych użytkownika oraz zdefiniowane przez niego cele i poziomy aktywności.

Nazwa kolumny	Typ danych	Opis i ograniczenia
id (PK)	serial	Unikalny identyfikator pomiaru.
user_id (FK)	integer	Powiązanie z tabelą users.
date	timestamp	Data i godzina wykonania pomiaru.
height	double precision	Wzrost użytkownika.
weight	double precision	Masa ciała.
chest	double precision	Obwód klatki piersiowej.
waist	double precision	Obwód pasa.
biceps	double precision	Obwód ramienia.
thighs	double precision	Obwód uda.
hips	double precision	Obwód bioder.
neck	numeric	Obwód szyi.
goal	varchar(50)	Cel treningowy (np. redukcja, masa).
activity_level	numeric(4,3)	Współczynnik aktywności fizycznej.

Tabela 3.2. Struktura tabeli body_measurements

Tabela: muscle_groups

Słownikowa tabela zawierająca nazwy partii mięśniowych, co pozwala na kategoryzację ćwiczeń.

Nazwa kolumny	Typ danych	Opis i ograniczenia
id (PK)	serial	Identyfikator grupy mięśniowej.
name	varchar(50)	Nazwa partii (np. klatka piersiowa, plecy).

Tabela 3.3. Struktura tabeli muscle_groups

Tabela: exercises

Katalog dostępnych w systemie ćwiczeń wraz z ich opisami i przypisaniem do konkretnych partii.

Nazwa kolumny	Typ danych	Opis i ograniczenia
id (PK)	serial	Identyfikator ćwiczenia.
name	varchar(100)	Nazwa ćwiczenia.
description	text	Instrukcja lub opis techniki ćwiczenia.
muscle_group_id (FK)	integer	Powiązanie z tabelą muscle_groups.

Tabela 3.4. Struktura tabeli exercises

Tabela: workouts

Rejestruje każdą rozpoczętą sesję treningową użytkownika wraz z czasem jej trwania.

Nazwa kolumny	Typ danych	Opis i ograniczenia
id (PK)	serial	Identyfikator sesji treningowej.
user_id (FK)	integer	Identyfikator użytkownika wykonującego trening.
date	timestamp	Data i godzina rozpoczęcia treningu.
duration	interval	Czas trwania sesji treningowej.

Tabela 3.5. Struktura tabeli workouts

Tabela: workout_exercises

Tabela pośrednicząca, która przypisuje konkretne ćwiczenia do danej jednostki treningowej.

Nazwa kolumny	Typ danych	Opis i ograniczenia
id (PK)	serial	Identyfikator wpisu ćwiczenia w treningu.
workout_id (FK)	integer	Powiązanie z konkretną sesją (workouts).
exercise_id (FK)	integer	Powiązanie z katalogiem ćwiczeń (exercises).

Tabela 3.6. Struktura tabeli workout_exercises

Tabela: workout_sets

Najbardziej szczegółowa tabela systemu, przechowująca parametry każdej wykonanej serii.

Nazwa kolumny	Typ danych	Opis i ograniczenia
id (PK)	serial	Identyfikator serii.
workout_exercise_id (FK)	integer	Powiązanie z ćwiczeniem w ramach treningu.
weight	double precision	Obciążenie użyte w serii.
reps	integer	Liczba wykonanych powtórzeń.
set_number	integer	Numer porządkowy serii w danym ćwiczeniu.

Tabela 3.7. Struktura tabeli workout_sets

3.2. Powiązania pomiędzy tabelami i ich interpretacja

Analiza diagramu ERD oraz struktur tabel pozwala na zdefiniowanie relacji, które gwarantują spójność danych oraz umożliwiają zaawansowaną analitykę treningową. Poniżej przedstawiono szczegółową interpretację kluczowych powiązań:

- **Relacja users – body_measurements (1:N):** To powiązanie pozwala na przypisanie wielu pomiarów ciała do jednego profilu użytkownika. Z punktu widzenia systemu jest to kluczowe dla monitorowania zmian w czasie (np. spadku wagi czy wzrostu obwodów). Dzięki tej relacji dashboard może generować wykresy trendów, porównując najnowszy wpis z danymi historycznymi.
- **Relacja users – workouts (1:N):** Fundament rejestracji aktywności. Każdy trening jest trwale przypisany do konkretnego użytkownika poprzez klucz obcy *user_id*. Umożliwia to izolację danych i obliczanie statystyk takich jak aktualna passa treningowa (streak) czy całkowita objętość podniesiona przez daną osobę w skali tygodnia.
- **Struktura hierarchiczna treningu (workouts – workout_exercises – workout_sets):** Zastosowano tutaj potrójne powiązanie typu jeden-do-wielu, co odzwierciedla naturalną strukturę sesji na siłowni. Jeden rekord w *workouts* (sesja) zawiera wiele wpisów w *workout_exercises* (wykonane ćwiczenia), a każde z tych ćwiczeń posiada wiele rekordów w *workout_sets* (serie). Taka dekompozycja danych pozwala na precyzyjne wyliczanie tonażu (objętości) dla każdego ćwiczenia z osobna.
- **Relacja exercises – workout_exercises (1:N):** Łączy konkretne wykonanie ćwiczenia z jego definicją w katalogu. Dzięki temu system "wie", jakie ćwiczenie wykonuje użytkownik i może pobrać z tabeli *exercises* jego opis lub przypisaną grupę mięśniową.
- **Relacja muscle_groups – exercises (1:N):** Każde ćwiczenie jest sklasyfikowane pod jedną grupą mięśniową. To powiązanie jest niezbędne do działania algorytmu balansu strukturalnego. System sumuje serie z tabeli *workout_sets*, przechodzi przez powiązania do *muscle_groups* i na tej podstawie wylicza procentowy udział treningu klatki piersiowej, pleców czy nóg w skali tygodnia.

3.2.1. Opcjonalność relacji

Dla zachowania spójności danych, w systemie zdefiniowano następujące zasady opcjonalności:

- **Relacja users – workouts (Obowiązkowa):** Każdy rekord w tabeli *workouts* musi posiadać przypisane *user_id*. Nie jest możliwe istnienie treningu w systemie bez przypisanego autora (użytkownika).
- **Relacja workouts – workout_exercises (Obowiązkowa):** Każdy wpis o wykonanym ćwiczeniu musi odwoływać się do konkretnej sesji treningowej. Usunięcie treningu powoduje kaskadowe usunięcie przypisanych do niego ćwiczeń.
- **Relacja exercises – muscle_groups (Obowiązkowa):** W systemie każde ćwiczenie musi być skategoryzowane pod konkretną partią mięśniową, aby algorytmy analityczne mogły poprawnie przeliczać objętość tygodniową.
- **Relacja users – body_measurements (Opcjonalna):** Użytkownik może, ale nie musi posiadać wpisów w tabeli pomiarów. System dopuszcza istnienie konta bez historii pomiarów, choć ogranicza to wtedy dostęp do niektórych funkcji dashboardu (np. wykresów zmiany wagi).
- **Relacja workout_exercises – workout_sets (Obowiązkowa):** Seria nie może istnieć bez powiązania z konkretnym ćwiczeniem w ramach danego treningu.

3.3. Diagram związków encji (ERD)

Poniższy diagram przedstawia graficzną reprezentację struktury bazy danych systemu. Ilustruje on wszystkie opisane wcześniej tabele, ich atrybuty oraz relacje zachodzące pomiędzy poszczególnymi encjami, z uwzględnieniem kluczy głównych (PK) oraz obcych (FK).

Rys. 3.1. Diagram związków encji (ERD) projektowanej bazy danych.

3.4. Kod implementujący strukturę bazy danych

Poniższy przedstawiono kod, który posłużył do fizycznego utworzenia struktury tabel bazy danych zgodnie z zaprojektowanym diagramem ERD.

Listing 3.1. Kod implementujący strukturę tabel bazy danych

```
-- 1. Tabela uzytkownikow
CREATE TABLE public.users (
    id SERIAL PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    gender TEXT
);

-- 2. Tabela pomiarow ciala
CREATE TABLE public.body_measurements (
    id SERIAL PRIMARY KEY,
    user_id INTEGER NOT NULL,
    date TIMESTAMP WITHOUT TIME ZONE NOT NULL,
    height DOUBLE PRECISION,
    weight DOUBLE PRECISION,
    chest DOUBLE PRECISION,
    waist DOUBLE PRECISION,
    biceps DOUBLE PRECISION,
    thighs DOUBLE PRECISION,
    hips DOUBLE PRECISION,
    neck NUMERIC,
    goal VARCHAR(50),
    activity_level NUMERIC(4,3),
    FOREIGN KEY (user_id) REFERENCES public.users(id)
);

-- 3. Tabela grup miesniowych
CREATE TABLE public.muscle_groups (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL
);

-- 4. Tabela cwiczen
CREATE TABLE public.exercises (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    muscle_group_id INTEGER NOT NULL,
    FOREIGN KEY (muscle_group_id) REFERENCES public.muscle_groups(id)
);
```

```
-- 5. Tabela treningow (naglowki)
CREATE TABLE public.workouts (
    id SERIAL PRIMARY KEY,
    user_id INTEGER NOT NULL,
    date TIMESTAMP WITHOUT TIME ZONE NOT NULL,
    duration INTERVAL,
    FOREIGN KEY (user_id) REFERENCES public.users(id)
);

-- 6. Tabela cwiczen w ramach treningu
CREATE TABLE public.workout_exercises (
    id SERIAL PRIMARY KEY,
    workout_id INTEGER NOT NULL,
    exercise_id INTEGER NOT NULL,
    FOREIGN KEY (workout_id) REFERENCES public.workouts(id),
    FOREIGN KEY (exercise_id) REFERENCES public.exercises(id)
);

-- 7. Tabela serii treningowych
CREATE TABLE public.workout_sets (
    id SERIAL PRIMARY KEY,
    workout_exercise_id INTEGER NOT NULL,
    weight DOUBLE PRECISION,
    reps INTEGER,
    set_number INTEGER,
    FOREIGN KEY (workout_exercise_id) REFERENCES public.workout_exercises(id)
);
```

4. Logika operacyjna i funkcjonalność bazy danych

Struktura bazy danych została zaprojektowana tak, aby wspierała kluczowe funkcje aplikacji AwareFit. Funkcjonalność opiera się na dwóch aspektach: standardowej obsłudze danych poprzez mechanizmy CRUD oraz zaawansowanej logice algorytmicznej, która pozwala na przekształcenie surowych wpisów treningowych w użyteczne informacje analityczne dla użytkownika.

4.1. Założenia i wymagania dla mechanizmów CRUD

Mechanizm CRUD (ang. Create, Read, Update, Delete) stanowi fundament interakcji użytkownika z systemem. W ramach aplikacji AwareFit zdefiniowano, w schemacie *crud* zaimplementowany zestaw funkcji i procedur. Poniżej przedstawiono podział na komponenty CRUD.

Zarządzanie tożsamością i dostępem (Komponent: users)

Komponent ten dotyczny zarządzeniem kontami użytkowników, zaimplementowany w procedurach schematu *crud*.

- **Operacja Create (Rejestracja):** Wykorzystywana jest procedura `insert_user`, która przyjmuje komplet danych: login, hasło, email oraz dane personalne. Zapewnia ona poprawność wpisu w tabeli głównej *public.users*.
- **Operacja Read (Logowanie i Profil):** Funkcja `get_all_users` pozwala na weryfikację uprawnień i pobieranie danych użytkowników.
- **Operacja Update i Delete:** Zaimplementowano procedury `update_user` oraz `delete_user`. Warto zaznaczyć, że choć są one gotowe w warstwie bazy danych, aktualna wersja interfejsu graficznego (GUI) nie wykorzystuje ich bezpośrednio, co stanowi przygotowanie pod przyszłą rozbudowę modułu administracyjnego.

Rejestracja i obsługa sesji treningowych (Komponent: workouts)

Jest to najbardziej rozbudowany fragment schematu *crud*, obsługujący wielopoziomową strukturę jednostki treningowej.

- **Struktura hierarchiczna:** Proces zapisu treningu jest atomowy i sekwencyjny. Najpierw wywoływana jest procedura `insert_workout`, a następnie dla każdego wybranego ćwiczenia `insert_workout_exercise`. Detale wykonania (ciężar, powtórzenia) są zapisywane przez `insert_workout_set`.
- **Zarządzanie zmianami i perspektywy rozwoju:** Procedury `update_workout_set` oraz `update_workout_exercise` zostały zaprojektowane i wdrożone w warstwie bazy danych jako fundament pod przyszłą rozbudowę funkcjonalności systemu. W obecnej iteracji aplikacji, interfejs użytkownika (GUI) nie udostępnia możliwości edycji treningów historycznych, co jest podyktowane dbałością o spójność danych i rzetelność generowanych statystyk. Niemniej jednak, pełna implementacja tych procedur w schemacie *crud* umożliwia ich natychmiastowe wykorzystanie w przyszłości bez konieczności modyfikacji logiki serwera bazy danych.

Monitoring parametrów biometrycznych (Komponent: body_measurements)

Komponent ten dedykowany jest gromadzeniu i analizie danych o stanie fizycznym użytkownika. Logika bazodanowa została zaprojektowana tak, aby wspierać rzetelne śledzenie progresji w czasie.

- **Rejestracja pomiarów:** Główną operacją w interfejsie użytkownika jest *Create*, realizowana przez zestaw procedur `insert_body_measurement`. Użytkownik ma możliwość regularnego dodawania nowych rekordów zawierających wagę oraz obwody poszczególnych partii ciała.
- **Niezmienność danych historycznych:** Przyjęto założenie projektowe, według którego raz wprowadzone dane biometryczne nie podlegają edycji z poziomu interfejsu graficznego (GUI). Ma to na celu zachowanie autentyczności historii pomiarów i uniemożliwienie manipulacji danymi archiwalnymi.
- **Zaimplementowany mechanizm Update:** Pomimo ograniczeń w interfejsie, w schemacie *crud* w pełni zaimplementowano procedurę `update_body_measurement`. Pozwala to na ewentualne przyszłe wdrożenie funkcji korekty błędnych wpisów przez użytkownika.
- **Wyświetlanie i prezentacja danych:** Za warstwę *Read* odpowiada funkcja `get_all_body_measurements` oraz dedykowane zapytania filtrujące, które zasilają interfejs graficzny. Mechanizm ten pozwala na dynamiczne prezentowanie parametrów w formie zestawień tabelarycznych oraz stanowi źródło danych dla poszczególnych funkcji prezentujących informacje na temat ciała.

Zarządzanie strukturą słownikową (Komponent: exercises & muscle_groups)

Tabele słownikowe stanowią bazę wiedzy systemu, definiując dostępne ćwiczenia oraz ich przynależność do grup mięśniowych.

- **Dostęp użytkownika:** Z poziomu interfejsu standardowego użytkownika, dostęp do danych słownikowych jest ograniczony wyłącznie do operacji *Read*. Wykorzystywane są do tego funkcje `get_all_exercises` oraz `get_all_muscle_groups`, które zasilają listy wyboru podczas kreowania nowej sesji treningowej.
- **Perspektywa administracyjna:** Mimo braku odpowiednich modułów w bieżącym GUI, w schemacie *crud* zaimplementowano pełen zestaw procedur zarządczych: `insert_exercise`, `update_exercise`, `delete_exercise` (oraz ich odpowiedniki dla grup mięśniowych).
- **Założenia projektowe:** Istnienie tych mechanizmów w warstwie bazy danych zostało przewidziane jako fundament pod dedykowany "Panel Administratora". Pozwoli on w przyszłości na dynamiczne rozbudowywanie bazy ćwiczeń, czy edycję opisów bez konieczności bezpośredniej ingerencji programisty w strukturę bazy danych.

4.2. Implementacja mechanizmów CRUD na przykładzie komponentu użytkowników

W celu zapewnienia integralności danych oraz odciążenia warstwy logicznej aplikacji (PHP), kluczowe reguły zostały zaimplementowane bezpośrednio w procedurach składowanych schematu *crud*.

Tworzenie rekordu (Create)

Za dodawanie nowych użytkowników odpowiada procedura `crud.insert_user`.

- **Założenia:** Procedura przyjmuje komplet danych profilowych i dokonuje ich ostatecznej weryfikacji przed trwałym zapisem w bazie.
- **Implementacja i walidacja:** Poza operacją `INSERT`, procedura aktywnie sprawdza poprawność formatu adresu e-mail przy użyciu wyrażeń regularnych (*regex*).
- **Obsługa błędów unikalności:** Zastosowano blok `EXCEPTION`, który przechwytuje naruszenia kluczy unikalnych (`unique_violation`). Pozwala to na precyzyjne rozróżnienie, czy błąd rejestracji wynika z duplikatu nazwy użytkownika, czy zajętego adresu e-mail, co jest kluczowe dla komunikatów zwrotnych w interfejsie użytkownika.

Odczyt danych (Read)

Funkcja `crud.get_all_users` stanowi standardowy interfejs dostępu do danych dla warstwy aplikacji.

- **Założenia:** Zapewnienie szybkiego dostępu do listy użytkowników przy zachowaniu struktury rekordowej.
- **Implementacja:** Funkcja wykorzystuje mechanizm `RETURNS SETOF users`, co pozwala traktować jej wynik jak wirtualną tabelę. Zastosowanie instrukcji `RETURN QUERY` optymalizuje proces pobierania danych przez sterownik PDO w PHP.

Aktualizacja danych (Update)

Procedura `crud.update_user` wprowadza warstwę weryfikacji przed modyfikacją istniejących zasobów.

- **Założenia:** Edycja danych jest dopuszczalna wyłącznie dla istniejącego w systemie identyfikatora użytkownika.
- **Mechanizm walidacji:** Przed wykonaniem polecenia `UPDATE`, procedura sprawdza istnienie rekordu:

```
IF NOT EXISTS (SELECT 1 FROM public.users WHERE id = p_id) THEN
   RAISE EXCEPTION 'Nie znaleziono użytkownika o ID %', p_id;
END IF;
```

Gwarantuje to spójność operacji i ułatwia debugowanie komunikacji na linii API-Baza danych.

Usuwanie danych (Delete)

Procedura `crud.delete_user` odpowiada za bezpieczne usuwanie kont przy zachowaniu integralności referencyjnej.

- **Założenia:** Blokada usunięcia użytkowników posiadających aktywne powiązania w innych modułach systemu (np. historia treningowa).
- **Obsługa wyjątków:** Wykorzystano przechwytywanie błędu `foreign_key_violation`:

```
EXCEPTION WHEN foreign_key_violation THEN
   RAISE EXCEPTION 'Nie można usunąć użytkownika - posiada powiązane dane...';
```

  Dzięki temu system chroni historię treningową przed powstaniem rekordów osieroconych (*orphaned records*).

4.3. Opis pytań algorytmicznych

Zaimplementowane procedury składowane zostały podzielone na dwie grupy w zależności od stopnia złożoności realizowanej logiki. Kryterium podziału stanowi charakter wykonywanych operacji oraz zakres przetwarzania danych.

4.3.1. Algorytmy operacyjne

Do algorytmów operacyjnych zaliczono procedury składowane realizujące podstawowe operacje na danych, takie jak ich pobieranie, filtrowanie oraz zapis w bazie danych. Algorytmy te opierają się głównie na zapytaniach typu *SELECT* oraz operacjach *INSERT* i nie wykonują złożonych obliczeń ani analizy danych.

Ich zadaniem jest zapewnienie spójnego i wydajnego dostępu do danych dla warstwy aplikacyjnej oraz uporządkowanie logiki dostępu do informacji przechowywanych w bazie danych.

get_user_profile_data

Funkcja służy do pobierania podstawowych danych profilowych użytkownika, takich jak nazwa użytkownika, adres e-mail oraz dane personalne. Wykorzystywana jest do prezentacji informacji profilowych w warstwie aplikacyjnej.

get_user_measurements

Algorytm umożliwia odczyt zapisanych pomiarów biometrycznych użytkownika w kolejności chronologicznej. Dane te stanowią podstawę do prezentacji historii zmian parametrów ciała.

get_user_workout_history

Funkcja odpowiada za pobranie listy treningów przypisanych do danego użytkownika. Wynik wykorzystywany jest do prezentacji historii aktywności treningowej.

get_detailed_workout

Algorytm umożliwia pobranie szczegółowych informacji dotyczących pojedynczej sesji treningowej, obejmujących ćwiczenia oraz wykonane serie. Dane te prezentowane są użytkownikowi w formie podsumowania treningu.

get_exercises_by_muscle_group

Funkcja umożliwia pobranie listy ćwiczeń przypisanych do określonej partii mięśniowej. Algorytm wykorzystywany jest podczas tworzenia lub edycji planu treningowego.

get_last_exercise_stats

Algorytm służy do pobrania ostatnio zapisanych parametrów wykonania danego ćwiczenia, takich jak ciężar i liczba powtórzeń. Informacje te wykorzystywane są jako punkt odniesienia w kolejnych treningach.

get_weekly_workout_count

Funkcja zwraca liczbę treningów wykonanych przez użytkownika w ostatnich siedmiu dniach. Wynik wykorzystywany jest do prezentacji statystyk aktywności treningowej.

login_by_username

Algorytm realizuje proces uwierzytelniania użytkownika na podstawie podanej nazwy użytkownika oraz hasła. Jego zadaniem jest weryfikacja danych dostępowych i kontrola dostępu do systemu.

save_complete_workout

Funkcja odpowiada za zapis kompletnej sesji treningowej wraz z jej parametrami i strukturą ćwiczeń. Zapewnia spójność danych zapisywanych w bazie danych podczas rejestracji treningu.

Ze względu na stosunkowo niski poziom złożoności algorytmów operacyjnych nie zamieszczono w dokumentacji szczegółowych listingów kodu ani rozbudowanych opisów ich implementacji. Algorytmy te realizują podstawowe operacje dostępu do danych, których logika jest jednoznaczna i nie wymaga dodatkowych wyjaśnień.

Pełna implementacja procedur składowanych dostępna jest w bazie danych systemu, której eksport został dołączony do repozytorium projektu udostępnionego na platformie GitHub. Takie podejście umożliwia zachowanie przejrzystości dokumentacji przy jednoczesnym zapewnieniu pełnej dostępności kodu źródłowego.

4.3.2. Zaawansowane algorytmy analityczne

Do algorytmów analitycznych zaliczono procedury składowane realizujące złożone przetwarzanie danych treningowych i biometrycznych użytkownika. Algorytmy te odpowiedzialne są za wyznaczanie wartości pochodnych, analizę trendów, porównania historyczne oraz klasyfikację danych na podstawie określonych reguł.

W przeciwieństwie do algorytmów operacyjnych, procedury tej grupy wykorzystują agregacje, obliczenia matematyczne oraz logikę warunkową, a ich implementacja stanowi istotny element logiki systemu.

1. calculate_exercise_1rm

Funkcja realizuje estymację ciężaru maksymalnego (One Repetition Maximum), jakiego użytkownik mógłby użyć w danym ćwiczeniu na podstawie najlepszej serii treningowej.

Wykorzystanie w interfejsie użytkownika Implementację tego algorytmu widzimy w panelu progresu, gdzie wyświetlane są wyliczone wartości estymowanego 1RM dla trzech głównych bojów (wyciskanie leżąc, przysiad, martwy ciąg). Dane te pozwalają użytkownikowi monitorować wzrost siły bez konieczności każdorazowego sprawdzania realnego rekordu, co zwiększa bezpieczeństwo treningu.

```
CREATE OR REPLACE FUNCTION public.calculate_exercise_1rm(
    p_user_id integer,
    p_exercise_id integer)
    RETURNS numeric
    LANGUAGE 'plpgsql'
AS $BODY$
DECLARE
    v_weight float;
    v_reps integer;
    v_1rm numeric;
BEGIN
    SELECT h.weight, h.reps
    INTO v_weight, v_reps
    FROM (
        SELECT ws.weight, ws.reps, w.date as workout_date, ws.set_number
        FROM public.workout_sets ws
        JOIN public.workout_exercises we ON ws.workout_exercise_id = we.id
        JOIN public.workouts w ON we.workout_id = w.id
        WHERE w.user_id = p_user_id::integer
          AND we.exercise_id = p_exercise_id::integer
    ) h
    WHERE h.set_number = 1
    ORDER BY h.workout_date DESC
    LIMIT 1;

    IF v_weight IS NULL OR v_reps IS NULL OR v_reps = 0 THEN
        RETURN 0;
    END IF;

    IF v_reps = 1 THEN
        v_1rm := v_weight;
    ELSE
        -- Implementacja wzoru Wathena
        v_1rm := v_weight / (1.0278 - (0.0278 * v_reps));
    END IF;
```

```
    RETURN ROUND((v_1rm * 2)::numeric, 0) / 2;
END;
$BODY$;
```

Opis działania algorytmu Algorytm działa w następujących etapach:

- **Pozyskanie danych wejściowych:** System wyszukuje w bazie danych ostatnią wykonaną jednostkę treningową dla wskazanego użytkownika i konkretnego ćwiczenia.
- **Selekcja serii bazowej:** Algorytm pobiera parametry (ciężar i liczbę powtórzeń) z pierwszej serii (*set_number = 1*), która w metodologii treningowej zazwyczaj reprezentuje największy wysiłek siłowy.
- **Zastosowanie modelu matematycznego:** Na podstawie pobranych danych stosowany jest wzór Wathena. Jeśli liczba powtórzeń wynosi 1, system przyjmuje aktualny ciężar jako 1RM. W przeciwnym razie wylicza estymację według współczynnika siły.
- **Normalizacja wyniku:** Otrzymany wynik jest zaokrąglany do najbliższego 0,5 kg, co odpowiada standardowemu skokowi obciążenia na siłowni, czyniąc wynik praktycznym dla użytkownika.

2. calculate_user_bf

Funkcja odpowiada za estymację procentowej zawartości tkanki tłuszczowej (Body Fat Percentage) w organizmie użytkownika na podstawie pomiarów antropometrycznych.

Wykorzystanie w interfejsie użytkownika Implementację tego algorytmu widzimy w panelu diety. Obliczona wartość pozwala użytkownikowi śledzić zmiany w kompozycji sylwetki, co jest bardziej miarodajnym wskaźnikiem progresu niż sama masa ciała, szczególnie w procesie rekompozycji lub redukcji.

```
CREATE OR REPLACE FUNCTION public.calculate_user_bf(
    p_user_id integer)
    RETURNS numeric
    LANGUAGE 'plpgsql'
AS $BODY$
DECLARE
    v_gender TEXT;
    v_height NUMERIC;
    v_waist NUMERIC;
    v_neck NUMERIC;
    v_hips NUMERIC;
    v_bf NUMERIC;
BEGIN
    -- Pobranie płci użytkownika
    SELECT LOWER(TRIM(gender)) INTO v_gender FROM public.users WHERE id = p_user_id;

    -- Pobranie najnowszych pomiarów ciała
    SELECT height, waist, neck, hips INTO v_height, v_waist, v_neck, v_hips
    FROM public.body_measurements
    WHERE user_id = p_user_id
    ORDER BY date DESC LIMIT 1;
```

```
    -- Zabezpieczenie przed brakiem danych lub błędnymi wartościami
    IF v_height IS NULL OR v_waist IS NULL OR v_neck IS NULL OR (v_waist - v_neck) <= 0 THEN
        RETURN NULL;
    END IF;

    IF v_gender = 'male' THEN
        -- Wzór US Navy dla mężczyzn
        v_bf := 495 / (1.0324 - 0.19077 * log(v_waist - v_neck) + 0.15456 * log(v_height)) - 450;
    ELSE
        -- Wzór US Navy dla kobiet
        IF v_hips IS NULL OR (v_waist + v_hips - v_neck) <= 0 THEN RETURN NULL; END IF;
        v_bf := 495 / (1.29579 - 0.35004 * log(v_waist + v_hips - v_neck) + 0.22100 * log(v_height)) - 4
    END IF;

    -- Korekta wyników skrajnych
    IF v_bf < 2 THEN RETURN 2.0; END IF;

    RETURN ROUND(v_bf::numeric, 1);
END;
$BODY$;
```

Opis działania algorytmu Algorytm działa w następujących etapach:

- **Identyfikacja profilu użytkownika:** System pobiera płeć użytkownika, ponieważ wzór matematyczny różni się istotnie dla mężczyzn i kobiet ze względu na różnice w rozmieszczeniu tkanki tłuszczowej.
- **Agregacja danych antropometrycznych:** Pobierane są najnowsze wprowadzone wymiary: wzrost, obwód pasa, szyi oraz (w przypadku kobiet) bioder.
- **Walidacja matematyczna:** Funkcja sprawdza, czy dane są kompletne oraz czy różnice obwodów są dodatnie, co zapobiega błędom krytycznym podczas obliczeń logarytmicznych.
- **Obliczenia według modelu US Navy:** Zastosowany zostaje oficjalny model Marynarki Wojennej USA, oparty na logarytmach dziesiętnych z proporcji obwodów do wzrostu.
- **Normalizacja i zaokrąglenie:** Wynik jest ograniczany do fizjologicznego minimum (2%) i zaokrąglany do jednego miejsca po przecinku.

3. calculate_user_diet_calories

Funkcja służy do kompleksowego wyliczania zapotrzebowania energetycznego użytkownika, uwzględniając jego parametry fizyczne, poziom aktywności oraz obrany cel sylwetkowy. Dodatkowo funkcja ta jest wykorzystywana jest również w funkcji get_user_macros(p_user_id).

Wykorzystanie w interfejsie użytkownika Algorytm ten jest kluczowym elementem analitycznym aplikacji, a jego wyniki prezentowane są w dwóch głównych sekcjach:

- **Panel Dashboard:** Wynik wyświetlany jest w formie podsumowującej karty, informującej użytkownika o jego dziennym limicie kalorii zaraz po zalogowaniu.
- **Panel Dieta:** Wyliczona wartość stanowi punkt odniesienia dla dziennego spożycia makroskładników oraz jest wykorzystywana do prezentacji bilansu energetycznego.

```
CREATE OR REPLACE FUNCTION public.calculate_user_diet_calories(
    p_user_id integer)
    RETURNS TABLE(recommended_calories integer, goal_label character varying, difference_from_tdee integer)
    LANGUAGE 'plpgsql'
AS $BODY$
DECLARE
    v_gender varchar;
    v_height float;
    v_weight float;
    v_activity float;
    v_goal varchar;
    v_bmr float;
    v_tdee integer;
    v_age_const integer := 28; -- Przyjeta stala wieku
BEGIN
    -- Pobranie danych podstawowych i pomiarow
    SELECT gender INTO v_gender FROM public.users WHERE id = p_user_id;
    SELECT weight, height, activity_level, goal
    INTO v_weight, v_height, v_activity, v_goal
    FROM public.body_measurements
    WHERE user_id = p_user_id
    ORDER BY date DESC LIMIT 1;

    -- Obsluga wartosci domyslnych (COALESCE)
    v_weight := COALESCE(v_weight, 70.0);
    v_height := COALESCE(v_height, 175.0);
    v_activity := COALESCE(v_activity, 1.2);
    v_goal := COALESCE(v_goal, 'Rekompozycja ciała');

    -- Obliczenie BMR wg wzoru Mifflina-St Jeora
    IF lower(v_gender) LIKE 'm%' THEN
        v_bmr := (10 * v_weight) + (6.25 * v_height) - (5 * v_age_const) + 5;
    ELSE
        v_bmr := (10 * v_weight) + (6.25 * v_height) - (5 * v_age_const) - 161;
    END IF;

    -- Wyliczenie TDEE (Total Daily Energy Expenditure)
    v_tdee := round(v_bmr * v_activity);

    -- Logika modyfikacji kalorycznosci pod wybrany cel
```

```
    IF v_goal = 'Zbudowanie masy mięśniowej' THEN
        recommended_calories := round(v_tdee * 1.10); -- Nadwyzka 10%
        goal_label := 'Masa';
    ELSIF v_goal = 'Redukcja tkanki tłuszczowej' THEN
        recommended_calories := round(v_tdee * 0.80); -- Deficyt 20%
        goal_label := 'Redukcja';
    ELSE
        recommended_calories := v_tdee;
        goal_label := 'Rekompozycja';
    END IF;

    difference_from_tdee := recommended_calories - v_tdee;
    RETURN NEXT;
END;
$BODY$;
```

Opis działania algorytmu Algorytm działa w następujących etapach:

- **Ekstrakcja danych profilowych:** Funkcja agreguje dane z tabeli użytkowników oraz najnowsze rekordy z tabeli pomiarów (*body_measurements*).
- **Zastosowanie mechanizmu COALESCE:** W przypadku braku wpisów antropometrycznych, system przypisuje wartości uśrednione, co zapobiega błędom w interfejsie graficznym.
- **Kalkulacja BMR:** Wyliczany jest wskaźnik podstawowej przemiany materii przy użyciu równania Mifflina-St Jeora, różnicowanego ze względu na płeć.
- **Wyznaczenie TDEE:** Wynik BMR jest mnożony przez współczynnik aktywności fizycznej (*Physical Activity Level*), co daje realne zapotrzebowanie na utrzymanie wagi.
- **Dostosowanie do celu sylwetkowego:** W zależności od wybranego celu (Masa/Redukcja/Rekompozycja), algorytm nakłada matematyczną korektę (nadwyżkę lub deficyt procentowy) na finalny wynik kaloryczny.

4. get_user_macros

Funkcja ta odpowiada za precyzyjny podział wyliczonej puli energetycznej na poszczególne makroskładniki (białka, tłuszcze i węglowodany). Co istotne, algorytm ten nie działa w izolacji – wykorzystuje on bezpośrednio wyniki zwracane przez opisaną wcześniej funkcję `calculate_user_diet_calories`, co zapewnia spójność danych w całym systemie.

Wykorzystanie w interfejsie użytkownika Wyniki tej funkcji są kluczowe dla panelu diety, gdzie użytkownik otrzymuje konkretne wytyczne , co do makroskładników w gramach.

```
CREATE OR REPLACE FUNCTION public.get_user_macros(
    p_user_id integer)
    RETURNS TABLE(protein_g integer, fat_g integer, carbs_g integer, calories_total integer)
    LANGUAGE 'plpgsql'
AS $BODY$
DECLARE
    v_weight float;
    v_kcal integer;
BEGIN
    -- Wywolanie zaleznosci: pobranie rekomendowanych kalorii
    SELECT recommended_calories INTO v_kcal
    FROM public.calculate_user_diet_calories(p_user_id);

    -- Pobranie aktualnej wagi do wyliczenia podazy bialka
    SELECT weight INTO v_weight
    FROM public.body_measurements
    WHERE user_id = p_user_id
    ORDER BY date DESC LIMIT 1;

    v_weight := COALESCE(v_weight, 70.0);
    calories_total := v_kcal;

    -- OBLICZENIA LOGICZNE
    -- Bialko: 2g na kg masy ciala (wg standardów ISSN)
    protein_g := round(v_weight * 2.0);

    -- Tluszcze: Staly udzial 25% energii (1g = 9 kcal)
    fat_g := round((v_kcal * 0.25) / 9);

    -- Weglowodany: Uzupelnienie pozostalej puli kcal (1g = 4 kcal)
    carbs_g := round((v_kcal - (protein_g * 4) - (fat_g * 9)) / 4);

    RETURN NEXT;
END;
$BODY$;
```

Opis działania algorytmu Algorytm realizuje proces dystrybucji makroskładników w następujących krokach:

- **Integracja z modułem kalorii:** Funkcja wywołuje `calculate_user_diet_calories`, aby uzyskać dynamicznie obliczony cel energetyczny dostosowany do aktualnego profilu użytkownika.
- **Priorytetyzacja białka:** W pierwszej kolejności obliczana jest podaż białka na podstawie aktualnej masy ciała ($2g/kg$), co jest kluczowe dla regeneracji mięśniowej zgodnie z literaturą przedmiotu.
- **Alokacja tłuszczy:** System rezerwuje 25% całkowitego zapotrzebowania na tłuszcze, co zapewnia odpowiednią podaż energii z lipidów przy jednoczesnym zachowaniu równowagi hormonalnej.
- **Dopełnienie węglowodanowe:** Algorytm oblicza pozostałą liczbę kalorii (po odjęciu białek i tłuszczy) i zamienia ją na gramaturę węglowodanów. Taka kolejność gwarantuje, że suma makroskładników zawsze odpowiada założonemu limitowi kalorii.

5. calculate_workout_total_volume

Funkcja ta odpowiada za obliczanie całkowitej objętości treningowej (*Total Volume*), czyli sumarycznego tonażu przerzuconego przez użytkownika podczas pojedynczej jednostki treningowej.

Wykorzystanie w interfejsie użytkownika Wynik tej funkcji jest wykorzystywany w panelu historia. Przy każdym archiwalnym treningu użytkownik widzi sumaryczną wartość objętości (np. w kilogramach), co pozwala na szybką ocenę intensywności danej sesji oraz porównywanie obciążeń na przestrzeni czasu bez konieczności ręcznego sumowania poszczególnych serii.

```
CREATE OR REPLACE FUNCTION public.calculate_workout_total_volume(
    p_workout_id integer)
    RETURNS double precision
    LANGUAGE 'plpgsql'
AS $BODY$
DECLARE
    v_total_volume float;
BEGIN
    -- Sumujemy iloczyn ciezaru i powtorzen dla wszystkich serii w danym treningu
    SELECT SUM(ws.weight * ws.reps)
    INTO v_total_volume
    FROM public.workout_sets ws
    JOIN public.workout_exercises we ON ws.workout_exercise_id = we.id
    WHERE we.workout_id = p_workout_id;

    -- Zabezpieczenie przed zwróceniem wartosci pustej (NULL)
    RETURN COALESCE(v_total_volume, 0);
END;
$BODY$;
```

Opis działania algorytmu Algorytm realizuje obliczenia w następujących krokach:

- **Agregacja danych serii:** Funkcja łączy tabelę serii (*workout_sets*) z tabelą ćwiczeń (*workout_exercises*), aby odfiltrować wszystkie rekordy należące do konkretnego, wskazanego identyfikatorem treningu.
- **Kalkulacja tonażu:** Dla każdego wiersza wykonywany jest iloczyn ciężaru (*weight*) oraz liczby powtórzeń (*reps*). Jest to standardowa miara objętości w treningu oporowym.
- **Sumowanie końcowe:** System sumuje iloczyny ze wszystkich serii i ćwiczeń wchodzących w skład danej jednostki treningowej.
- **Obsługa wyjątków:** Dzięki zastosowaniu funkcji `COALESCE`, w przypadku gdy trening został zarejestrowany, ale nie dodano do niego jeszcze żadnych serii, funkcja zwraca wartość 0 zamiast błędu lub wartości nieokreślonej.

6. detect_training_split

Funkcja realizuje zaawansowaną analizę wzorców treningowych użytkownika w celu automatycznej identyfikacji stosowanego systemu (splitu) treningowego na podstawie historii aktywności z ostatnich 30 dni.

Wykorzystanie w interfejsie użytkownika Algorytm ten znajduje zastosowanie w dwóch kluczowych obszarach aplikacji:

- **Panel Dashboard:** Wynik wyświetlany jest w jednej z kart podsumowujących, dając użytkownikowi natychmiastowy feedback na temat charakteru jego aktualnego planu.
- **Panel Progresu:** Informacja o typie splitu służy jako kontekst do analizy wzrostu siły i objętości, pomagając zrozumieć, czy dany system treningu przynosi oczekiwane rezultaty.

```
CREATE OR REPLACE FUNCTION public.detect_training_split(
    p_user_id integer)
    RETURNS text
    LANGUAGE 'plpgsql'
AS $BODY$
DECLARE
    v_avg_groups_per_workout FLOAT;
    v_days_active_30d INT;
    v_has_legs BOOLEAN := FALSE;
    v_has_push BOOLEAN := FALSE;
    v_has_pull BOOLEAN := FALSE;
    v_result TEXT;
BEGIN
    -- Agregacja danych o trenowanych grupach miesniowych
    SELECT
        COUNT(DISTINCT s.workout_id),
        EXISTS (SELECT 1 FROM crud.get_user_training_stats(p_user_id) x
                WHERE x.muscle_group_name ILIKE ANY (ARRAY['%Nogi%', '%Uda%', '%Lydki%', '%Posladki%'])),
        EXISTS (SELECT 1 FROM crud.get_user_training_stats(p_user_id) x
                WHERE x.muscle_group_name IN ('Klatka␣piersiowa', 'Barki', 'Triceps')),
        EXISTS (SELECT 1 FROM crud.get_user_training_stats(p_user_id) x
                WHERE x.muscle_group_name IN ('Plecy', 'Biceps'))
```

```
    INTO v_days_active_30d, v_has_legs, v_has_push, v_has_pull
    FROM crud.get_user_training_stats(p_user_id) s;

    -- Obliczenie sredniej liczby grup na jednostke treningowa
    SELECT AVG(daily_count) INTO v_avg_groups_per_workout
    FROM (
        SELECT COUNT(DISTINCT muscle_group_name) as daily_count
        FROM crud.get_user_training_stats(p_user_id)
        GROUP BY workout_id
    ) AS subquery;

    IF v_days_active_30d = 0 OR v_avg_groups_per_workout IS NULL THEN
        RETURN 'Brak aktywności (30 dni)';
    END IF;

    -- Logika klasyfikacji systemu treningowego
    IF v_avg_groups_per_workout >= 4.0 THEN
        v_result := 'Full Body Workout';
    ELSIF v_has_push AND v_has_pull AND v_has_legs THEN
        v_result := 'Push Pull Legs';
    ELSIF v_has_push AND v_has_pull AND NOT v_has_legs THEN
        v_result := 'Push Pull (No Legs)';
    ELSIF v_has_legs AND v_days_active_30d >= 4 THEN
        v_result := 'Upper Lower';
    ELSE
        v_result := 'Własny system';
    END IF;

    RETURN v_result;
END;
$BODY$;
```

Opis działania algorytmu Algorytm klasyfikuje system treningowy w oparciu o następujące kroki:

- **Ekstrakcja statystyk mięśniowych:** Funkcja analizuje dane z pomocniczej procedury `get_user_training_stats`, sprawdzając obecność kluczowych ruchów: wypychających (*Push*), przyciągających (*Pull*) oraz angażujących dolne partie ciała.
- **Analiza wszechstronności sesji:** Obliczana jest średnia liczba unikalnych grup mięśniowych trenowanych podczas jednej wizyty na siłowni.
- **Kategoryzacja logiczna:**
  - Jeżeli średnia liczba grup na trening wynosi ≥ 4, system rozpoznaje **FBW (Full Body Workout)**.
  - Jeżeli użytkownik wykonuje ruchy Push, Pull oraz nogi w różnych sesjach, rozpoznawany jest system **PPL (Push Pull Legs)**.
  - Przy braku treningu nóg, ale obecności ruchów Push i Pull, system zwraca informację o niepełnym splicie.
  - Wysoka częstotliwość (4+ dni) przy obecności treningu nóg sugeruje system **Upper/Lower**.
- **Obsługa braku danych:** Algorytm poprawnie identyfikuje okresy roztrenowania lub braku aktywności w zdefiniowanym oknie czasowym.

7. get_exercise_progression_status

Funkcja ta służy do dynamicznej analizy porównawczej objętości konkretnego ćwiczenia pomiędzy dwiema ostatnimi jednostkami treningowymi, w których dane ćwiczenie wystąpiło.

Wykorzystanie w interfejsie użytkownika Algorytm ten jest wykorzystywany w momencie, gdy użytkownik dodaje nową sesję treningową. W momencie, gdy użytkownik wybiera konkretne ćwiczenie z listy, system natychmiast wyświetla podpowiedź.. Informuje ona, czy na ostatniej sesji nastąpił progres, stagnacja, czy regresja w stosunku do poprzedniego razu. Pozwala to użytkownikowi na bieżąco korygować obciążenie, aby zachować zasadę progresywnego przeładowania (*progressive overload*).

```
CREATE OR REPLACE FUNCTION public.get_exercise_progression_status(
    p_user_id integer,
    p_exercise_id integer)
    RETURNS text
    LANGUAGE 'plpgsql'
AS $BODY$
DECLARE
    v_last_total_vol float;
    v_prev_total_vol float;
    v_entry_count integer;
BEGIN
    -- Pobranie objetosci z dwóch ostatnich sesji treningowych
    WITH exercise_history AS (
        SELECT
            w.id as workout_id,
            w.date,
            SUM(ws.weight * ws.reps) as total_volume
        FROM public.workout_sets ws
        JOIN public.workout_exercises we ON ws.workout_exercise_id = we.id
        JOIN public.workouts w ON we.workout_id = w.id
        WHERE w.user_id = p_user_id
          AND we.exercise_id = p_exercise_id
        GROUP BY w.id, w.date
        ORDER BY w.date DESC
        LIMIT 2
    ),
    ranked_history AS (
        SELECT
            total_volume,
            ROW_NUMBER() OVER (ORDER BY date DESC) as rn
        FROM exercise_history
    )
    SELECT
        MAX(CASE WHEN rn = 1 THEN total_volume END),
        MAX(CASE WHEN rn = 2 THEN total_volume END),
        (SELECT COUNT(*) FROM exercise_history)
    INTO v_last_total_vol, v_prev_total_vol, v_entry_count
    FROM ranked_history;

    -- Klasyfikacja statusu na podstawie historii
    IF v_entry_count < 2 THEN
        RETURN 'NEW';
    END IF;
```

```
    IF v_last_total_vol > v_prev_total_vol THEN
        RETURN 'PROGRESS';
    ELSIF v_last_total_vol = v_prev_total_vol THEN
        RETURN 'STAGNATION';
    ELSE
        RETURN 'REGRESSION';
    END IF;
END;
$BODY$;
```

Opis działania algorytmu Algorytm realizuje analizę porównawczą w następujących etapach:

- **Agregacja historii ćwiczenia:** Wykorzystując wspólne wyrażenia tablicowe (*Common Table Expressions - CTE*), system selekcjonuje dwa najświeższe treningi użytkownika, w których brało udział dane ćwiczenie, obliczając dla nich sumaryczną objętość (*weight * reps*).
- **Rankowanie rekordów:** Funkcja okna `ROW_NUMBER()` przypisuje indeksy (1 i 2) do sesji, co pozwala na precyzyjne odróżnienie ostatniego treningu od przedostatniego.
- **Weryfikacja stażu:** Jeżeli w historii znajduje się tylko jedna sesja (lub zero), algorytm zwraca status `NEW`, co informuje interfejs o braku bazy do porównania.
- **Komparacja wyników:** System porównuje tonaż sesji n z sesją $n-1$:
  - `PROGRESS` – gdy objętość wzrosła,
  - `STAGNATION` – gdy tonaż pozostał identyczny,
  - `REGRESSION` – gdy odnotowano spadek obciążenia lub liczby powtórzeń.

8. get_exercise_volume_progression

Funkcja ta agreguje dane historyczne dotyczące tonażu konkretnego ćwiczenia w czasie, umożliwiając śledzenie długofalowej progresji siłowej użytkownika.

Wykorzystanie w interfejsie użytkownika Wyniki tej funkcji są wykorzystywane do generowania interaktywnego wykresu liniowego w panelu Dashboard. Dzięki danym zwracanym w formie tabelarycznej (data i suma objętości), aplikacja kliencka może w sposób czytelny zaprezentować użytkownikowi trend wzrostowy, co jest kluczowe dla utrzymania motywacji oraz weryfikacji skuteczności planu treningowego.

```
CREATE OR REPLACE FUNCTION public.get_exercise_volume_progression(
    p_user_id integer,
    p_exercise_id integer)
    RETURNS TABLE(workout_date date, total_volume numeric)
    LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
    RETURN QUERY
    SELECT
        w.date::DATE,
        -- Obliczenie objetosci suma (ciezar * powtorzenia)
        SUM(COALESCE(ws.weight, 0) * COALESCE(ws.reps, 0))::NUMERIC
    FROM public.workouts w
    JOIN public.workout_exercises we ON w.id = we.workout_id
    JOIN public.workout_sets ws ON we.id = ws.workout_exercise_id
    WHERE w.user_id = p_user_id
      AND we.exercise_id = p_exercise_id
      AND w.date >= CURRENT_DATE - INTERVAL '3 months'
    GROUP BY w.date::DATE
    ORDER BY w.date::DATE ASC;
END;
$BODY$;
```

Opis działania algorytmu Algorytm realizuje proces przygotowania danych pod wykresy w następujących etapach:

- **Wielopoziomowe łączenie danych:** Funkcja dokonuje złączenia (*JOIN*) trzech kluczowych tabel: nagłówków treningów (*workouts*), przypisanych do nich ćwiczeń (*workout_exercises*) oraz szczegółowych serii (*workout_sets*).
- **Filtrowanie czasowe:** System ogranicza analizę do ostatnich 3 miesięcy (*INTERVAL '3 months'*). Jest to zabieg optymalizacyjny, który zapewnia czytelność wykresu oraz szybkość odpowiedzi bazy danych, skupiając się na aktualnej formie użytkownika.
- **Agregacja matematyczna:** Dla każdej unikalnej daty treningowej obliczana jest suma iloczynów obciążenia i liczby powtórzeń. Zastosowanie `COALESCE` gwarantuje, że ewentualne puste rekordy nie przerwą procesu sumowania i zostaną potraktowane jako wartość 0.
- **Sortowanie chronologiczne:** Zwracany zestaw danych jest uporządkowany narastająco według daty, co jest niezbędne do poprawnego wyrenderowania osi X na wykresie liniowym przez bibliotekę Chart.js

9. get_user_muscle_balance

Funkcja realizuje analizę dystrybucji obciążenia treningowego na poszczególne partie mięśniowe w ujęciu procentowym, pozwalając na identyfikację priorytetów lub zaniedbań w planie treningowym.

Wykorzystanie w interfejsie użytkownika Algorytm ten jest wykorzystywany w panelu progresji, gdzie w dedykowanej karcie użytkownik widzi wykres przedstawiający sposób angażowania partii mięśniowych w ciągu ostatnich 7 dni. Dzięki temu system może pełnić rolę doradczą, sugerując użytkownikowi, czy zachowuje on odpowiedni balans między grupami takimi jak klatka piersiowa a plecy czy nogi. Pozwala też wyświetlać komunikat w przypadku, gdy któraś partia nie dostała żadnej stymulacji.

```
CREATE OR REPLACE FUNCTION public.get_user_muscle_balance(
    p_user_id integer)
    RETURNS TABLE(muscle_group_name text, volume_percentage numeric)
    LANGUAGE 'plpgsql'
AS $BODY$
DECLARE
    v_total_volume numeric;
BEGIN
    -- Obliczenie calkowitej objetosci z ostatnich 7 dni
    SELECT SUM(ws.weight * ws.reps)::numeric INTO v_total_volume
    FROM public.workout_sets ws
    JOIN public.workout_exercises we ON ws.workout_exercise_id = we.id
    JOIN public.workouts w ON we.workout_id = w.id
    WHERE w.user_id = p_user_id::integer
      AND w.date > NOW() - INTERVAL '7 days';

    -- Zabezpieczenie przed dzieleniem przez zero
    IF v_total_volume IS NULL OR v_total_volume = 0 THEN
        RETURN;
    END IF;

    -- Obliczenie procentowego udzialu kazdej grupy miesniowej
    RETURN QUERY
    SELECT
        mg.name::text,
        ROUND(((SUM(ws.weight * ws.reps) / v_total_volume) * 100)::numeric, 1) as percentage
    FROM public.workout_sets ws
    JOIN public.workout_exercises we ON ws.workout_exercise_id = we.id
    JOIN public.workouts w ON we.workout_id = w.id
    JOIN public.exercises e ON we.exercise_id = e.id
    JOIN crud.get_all_muscle_groups() mg ON e.muscle_group_id = mg.id
    WHERE w.user_id = p_user_id::integer
      AND w.date > NOW() - INTERVAL '7 days'
    GROUP BY mg.name
    ORDER BY percentage DESC;
END;
$BODY$;
```

Opis działania algorytmu Algorytm realizuje proces analizy balansu mięśniowego w następujących etapach:

- **Wyznaczenie bazy odniesienia:** W pierwszym kroku funkcja oblicza sumaryczny tonaż (*Total Volume*) ze wszystkich treningów użytkownika z ostatniego tygodnia. Stanowi on 100% wkładu treningowego.
- **Weryfikacja aktywności:** System sprawdza, czy w danym okresie wystąpiła jakakolwiek aktywność. W przypadku braku danych wejściowych, funkcja kończy działanie bez zwracania wyników, co zapobiega błędom dzielenia przez zero.
- **Mapowanie ćwiczeń na grupy mięśniowe:** Algorytm dokonuje złożonego złączenia pięciu tabel, przechodząc od pojedynczych serii, przez ćwiczenia, aż do słownika grup mięśniowych (*muscle_groups*).
- **Kalkulacja relatywna:** Dla każdej unikalnej grupy mięśniowej obliczana jest suma objętości, która następnie jest dzielona przez całkowity tonaż tygodniowy.
- **Formatowanie danych:** Wyniki są zaokrąglane do jednego miejsca po przecinku i sortowane malejąco, co pozwala frontendowi na natychmiastowe wyświetlenie najbardziej dominujących partii mięśniowych w planie użytkownika.

10. get_volume_comparison

Funkcja ta odpowiada za zestawienie całkowitej objętości treningowej z bieżącego tygodnia z analogicznym okresem poprzedzającym, co pozwala na błyskawiczną ocenę tendencji progresu.

Wykorzystanie w interfejsie użytkownika Wyniki tej funkcji są prezentowane w panelach *Dashboard* oraz *Progress* w formie wskaźników trendu (np. procentowy wzrost lub spadek tonażu). Pozwala to użytkownikowi na szybką weryfikację, czy zachowuje on założoną intensywność treningową w skali mikrocyklu, co jest kluczowe dla uniknięcia przetrenowania lub stagnacji.

```
CREATE OR REPLACE FUNCTION public.get_volume_comparison(
    p_user_id integer)
    RETURNS TABLE(current_volume numeric, previous_volume numeric)
    LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
    RETURN QUERY
    SELECT
        -- Tonaz z biezacego tygodnia (0-7 dni temu)
        COALESCE((
            SELECT SUM(ws.weight * ws.reps)::numeric
            FROM crud.get_all_workout_sets() ws
            JOIN crud.get_all_workout_exercises() we ON ws.workout_exercise_id = we.id
            JOIN crud.get_all_workouts() w ON we.workout_id = w.id
            WHERE w.user_id = p_user_id::integer
              AND w.date > NOW() - INTERVAL '7 days'
        ), 0),

        -- Tonaz z ubieglego tygodnia (8-14 dni temu)
        COALESCE((
            SELECT SUM(ws.weight * ws.reps)::numeric
            FROM crud.get_all_workout_sets() ws
            JOIN crud.get_all_workout_exercises() we ON ws.workout_exercise_id = we.id
            JOIN crud.get_all_workouts() w ON we.workout_id = w.id
            WHERE w.user_id = p_user_id::integer
              AND w.date <= NOW() - INTERVAL '7 days'
              AND w.date > NOW() - INTERVAL '14 days'
        ), 0);
END;
$BODY$;
```

Opis działania algorytmu Algorytm porównuje aktywność fizyczną w dwóch oknach czasowych w następujący sposób:

- **Selekcja bieżącego okresu:** System oblicza sumaryczny tonaż wszystkich serii wykonanych przez użytkownika w ciągu ostatnich 7 dni od momentu zapytania.
- **Selekcja okresu referencyjnego:** W drugim kroku funkcja izoluje dane z przedziału między 8. a 14. dniem wstecz, co stanowi bazę porównawczą dla bieżącego tygodnia.
- **Wykorzystanie widoków/procedur CRUD:** Funkcja korzysta z abstrakcji `get_all_...`, co zapewnia, że analiza zawsze operuje na spójnym zestawie danych, niezależnie od ewentualnych zmian w strukturze tabel podstawowych.

- **Zapewnienie ciągłości danych:** Dzięki zastosowaniu `COALESCE`, nawet w przypadku braku treningów w jednym z analizowanych okresów, funkcja zwraca wartość 0. Zapobiega to błędom w interfejsie graficznym i pozwala na poprawne obliczenie delty progresu przez aplikację kliencką.

Katalog algorytmów zaawansowanych

- **calculate_exercise_1rm(p_user_id, p_exercise_id)** *Dlaczego zaawansowany:* Dokonuje matematycznej estymacji wartości pochodnej (siły maksymalnej) na podstawie analizy historycznych serii treningowych przy użyciu wzoru Wathena.

- **calculate_user_bf(p_user_id)**
  Dlaczego zaawansowany: Implementuje model logarytmiczny US Navy, przetwarzając dane antropometryczne w celu wyznaczenia składu ciała użytkownika.

- **calculate_user_diet_calories(p_user_id)**
  Dlaczego zaawansowany: Wykorzystuje dietetyczne wzory prognostyczne (Mifflin-St Jeor) do wyznaczania BMR i TDEE, dynamicznie dostosowując wynik do parametrów fizycznych i celów użytkownika.

- **calculate_workout_total_volume(p_workout_id)**
  Dlaczego zaawansowany: Przetwarza wielopoziomowe struktury danych (JOIN-y między treningami a seriami), wykonując agregację matematyczną tonażu treningowego.

- **detect_training_split(p_user_id)**
  Dlaczego zaawansowany: Wykorzystuje heurystykę i logikę warunkową do automatycznego rozpoznawania wzorców behawioralnych użytkownika i kategoryzacji jego systemu treningowego.

- **get_exercise_progression_status(p_user_id, p_exercise_id)**
  Dlaczego zaawansowany: Stosuje analizę porównawczą (trendów) na danych historycznych, interpretując kierunek zmian (progres/stagnacja/regres) w wydolności siłowej.

- **get_exercise_volume_progression(p_user_id, p_exercise_id)**
  Dlaczego zaawansowany: Agreguje dane w oknie czasowym i analizuje dynamikę zmian objętości, przygotowując zestawy danych pod wizualizację trendów.

- **get_user_macros(p_user_id)**
  Dlaczego zaawansowany: Wylicza proporcje makroskładników poprzez integrację z algorytmem kaloryczności, stosując reguły priorytetyzacji składników odżywczych.

- **get_user_muscle_balance(p_user_id)**
  Dlaczego zaawansowany: Dokonuje analizy rozkładu obciążenia na struktury anatomiczne, przeliczając udziały procentowe zaangażowania poszczególnych partii mięśniowych.

- **get_volume_comparison(p_user_id)**
  Dlaczego zaawansowany: Wykonuje zestawienie zagregowanej objętości treningowej w dwóch odrębnych interwałach czasowych, umożliwiając analizę trendów tydzień do tygodnia.

Wnioski projektowe Przeniesienie powyższej logiki bezpośrednio do warstwy bazy danych (PL/pgSQL) pozwoliło na:

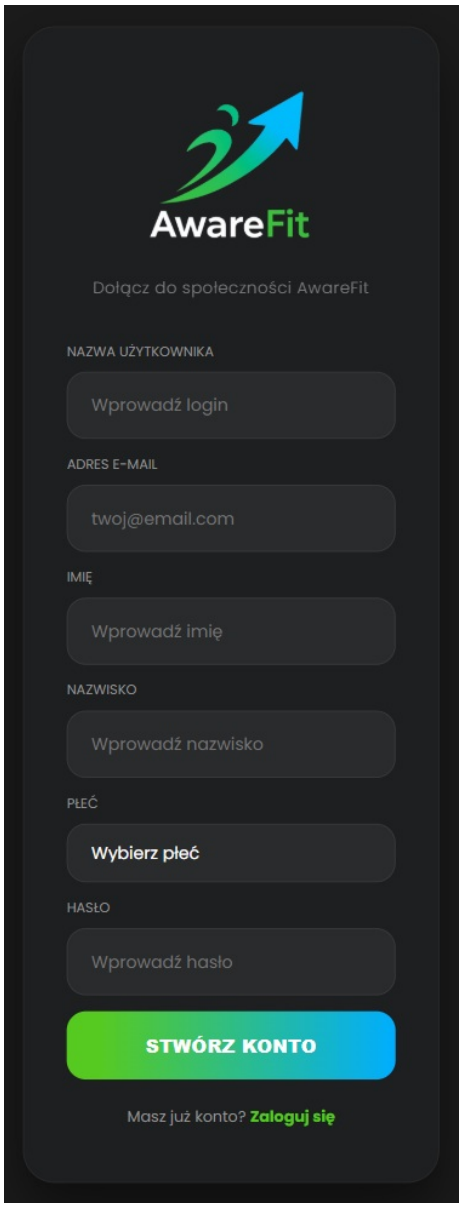
1. **Zwiększenie wydajności:** Redukcję ilości przesyłanych danych między serwerem bazy danych a aplikacją (przesyłane są tylko gotowe wyniki obliczeń).
2. **Centralizację logiki:** Zapewnienie identycznych wyników obliczeń niezależnie od tego, czy zapytanie pochodzi z panelu webowego, czy mobilnego.
3. **Spójność danych:** Gwarancję, że skomplikowane wzory matematyczne są wykonywane zawsze na najbardziej aktualnych danych wewnątrz transakcji.

5. Prezentacja interfejsu graficznego (GUI)

W niniejszym rozdziale przedstawiono warstwę wizualną aplikacji *AwareFit*, która stanowi interfejs komunikacji użytkownika z bazą danych. Każdy z prezentowanych widoków został zaprojektowany z myślą o intuicyjności oraz responsywności. Kluczowym aspektem opisu jest wskazanie bezpośrednich powiązań pomiędzy elementami interfejsu a logiką bazodanową i algorytmami szczegółowo opisanymi w rozdziale 4.

5.1. Widok rejestracji użytkownika

Pierwszym etapem interakcji z systemem jest proces rejestracji, który pozwala na utworzenie unikalnego konta użytkownika. Interfejs gromadzi podstawowe dane uwierzytelniające oraz parametry profilowe.



Rys. 5.1. Interfejs formularza rejestracji użytkownika

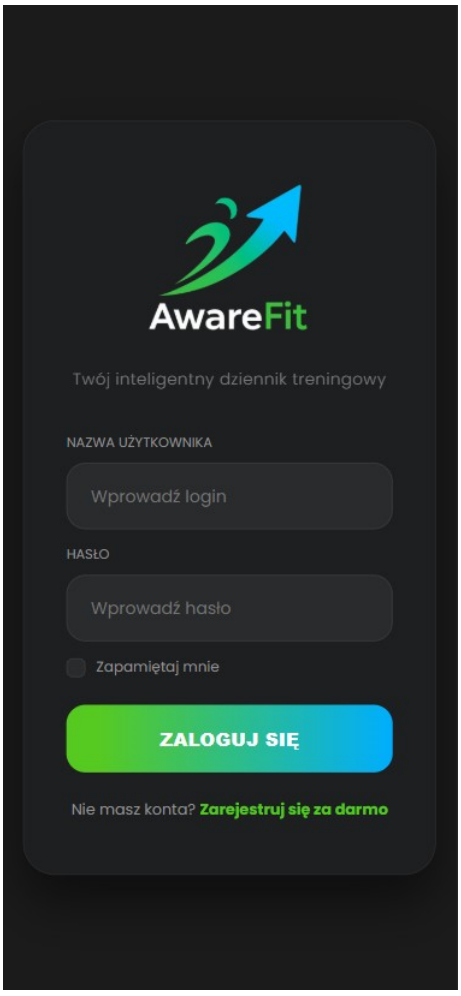
Powiązanie z logiką bazy danych: Powiązanie z logiką bazy danych:
Proces rejestracji obsługiwany jest przez skrypt backendowy `register.php`. Po stronie serwera następuje odebranie danych z formularza metodą `POST`, a następnie wywoływana jest funkcja bazodanowa `crud.insert_user`. Odpowiada ona za:

- **Walidację unikalności:** Sprawdzenie, czy podany adres e-mail lub username nie istnieje już w tabeli `users`.
- **Persystencję danych:** Trwałe zapisanie hasła oraz danych profilowych.
- **Inicjalizację konta:** Automatyczne nadanie identyfikatora użytkownika (`user_id`), który jest niezbędny do działania pozostałych modułów systemu.

Funkcjonalność GUI: Użytkownik wypełnia pola tekstowe, a system po zatwierdzeniu formularza przesyła dane do procedury bazodanowej.

5.2. Widok logowania

Widok logowania stanowi główny punkt dostępu do chronionej części aplikacji. Został zaprojektowany w sposób minimalistyczny, z wyraźnym podziałem na sekcję wizualną (branding) oraz funkcjonalną (formularz).



Rys. 5.2. Interfejs ekranu logowania systemu AwareFit

Powiązanie z logiką bazy danych:
Logika autoryzacji realizowana jest w pliku `index.php`. Po przesłaniu formularza metodą `POST`, skrypt nawiązuje połączenie z bazą danych i wykorzystuje funkcję bazodanową `public.login_by_username`. Proces ten obejmuje:

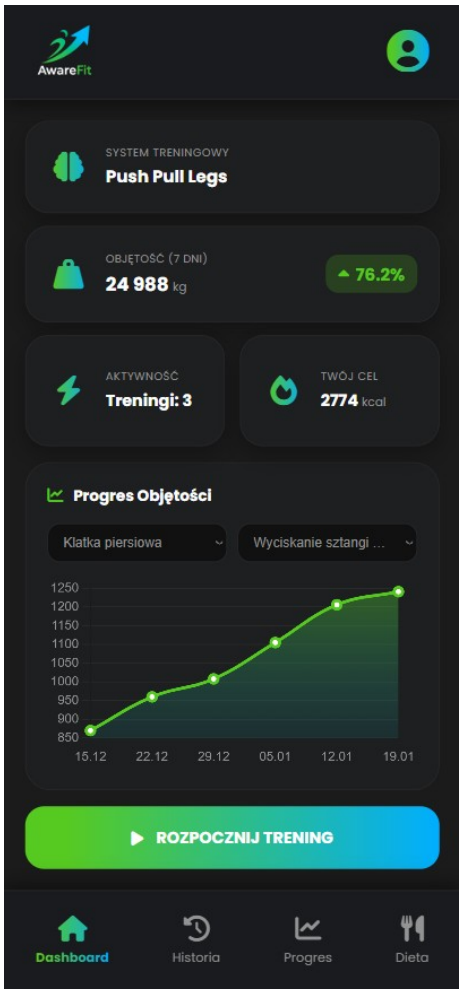
- **Weryfikację poświadczeń:** Funkcja przyjmuje login (`username`) oraz hasło, a następnie sprawdza ich poprawność w tabeli `users`.
- **Ekstrakcję danych sesyjnych:** W przypadku sukcesu, funkcja zwraca rekord zawierający `user_id` oraz `first_name`, co pozwala na zainicjowanie zmiennych sesyjnych w PHP (`$_SESSION`).
- **Zarządzanie dostępem:** Jeśli funkcja nie zwróci żadnego rekordu, skrypt generuje komunikat o błędzie (*„Nieprawidłowy login lub hasło”*), blokując dostęp do dalszych modułów systemu.

Funkcjonalność GUI:
Interfejs zawiera mechanizmy poprawiające doświadczenie użytkownika (*User Experience*), takie jak:

- Przejrzysta sekcja komunikatów o błędach (widoczna w przypadku nieudanej próby logowania).
- Zastosowanie atrybutów `autocomplete`, co zwiększa bezpieczeństwo i wygodę korzystania z menedżerów haseł.
- Bezpośredni odnośnik do formularza rejestracji (`register.php`) dla nowych użytkowników.

5.3. Panel główny (Dashboard)

Panel główny (`dashboard.php`) pełni funkcję centrum analitycznego dla użytkownika. Agreguje on surowe dane treningowe i dietetyczne, prezentując je w formie czytelnych kart informacyjnych (widżetów) oraz interaktywnego wykresu progresji.



Rys. 5.3. Panel główny aplikacji z systemem widżetów i wykresem progresu

Powiązanie z logiką bazy danych:
Widok ten jest jednym z najbardziej zaawansowanych elementów systemu pod kątem integracji z bazą danych. Wykorzystuje on szereg funkcji opisanych w poprzednich rozdziałach:

- **System treningowy i aktywność:** Skrypt wywołuje funkcję `public.detect_training_split` w celu identyfikacji aktualnego schematu ćwiczeń oraz wykonuje zliczanie rekordów z tabeli `workouts` z ostatnich 7 dni.
- **Analiza objętości i trendów:** Poprzez funkcję `public.get_volume_comparison` system pobiera dane o sumarycznym ciężarze podniesionym w obecnym i poprzednim tygodniu, wyliczając procentowy trend (`$diff_percent`).
- **Moduł żywieniowy:** Wykorzystywane są funkcje `calculate_user_diet_calories` oraz `get_user_macros`, które na podstawie profilu użytkownika zwracają dobowe zapotrzebowanie na kalorie i makroskładniki.

- **Progresja ćwiczeń (Wykres):** Dynamiczny wykres generowany jest dzięki funkcji `get_exercise_volume_progression`, która dostarcza historyczne dane o objętości dla konkretnego, wybranego przez użytkownika ćwiczenia.

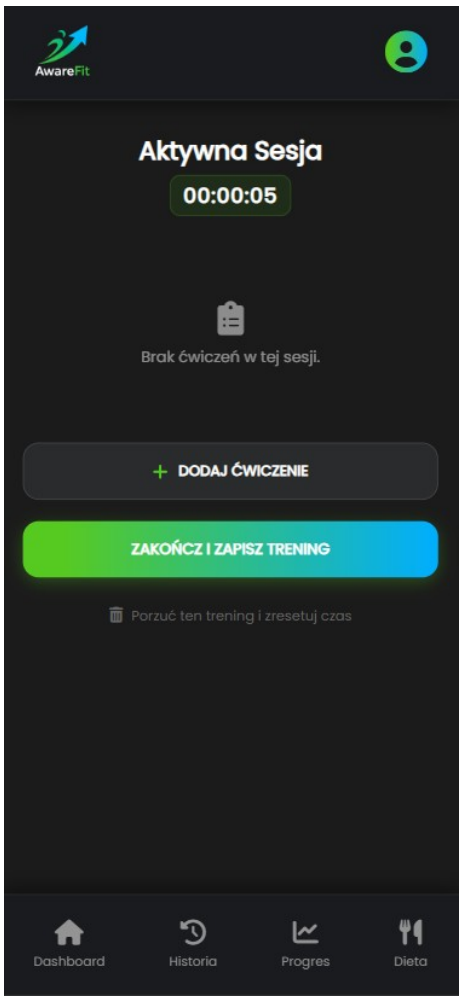
Funkcjonalność GUI:

Interfejs oferuje wysoką interaktywność dzięki zastosowaniu technologii *Chart.js* oraz asynchronicznej komunikacji:

- **Karty Progress Card:** Wykorzystują system ikon (*FontAwesome*) do szybkiej wizualizacji statusu treningowego i trendów objętości.
- **Interaktywny Wykres:** Użytkownik może filtrować dane za pomocą list rozwijanych (*Select*), co powoduje przeładowanie widoku z nowymi parametrami dla wybranego ćwiczenia.
- **Modal Makroskładników:** Kliknięcie w kartę kalorii wywołuje okno modalne (`macroModal`), prezentujące szczegółowy rozkład białek, tłuszczy i węglowodanów w formie graficznej.

5.4. Widok aktywnej sesji treningowej (Workout Session)

Moduł `workout.php` stanowi najbardziej interaktywną część aplikacji, pełniąc rolę cyfrowego asystenta treningowego. Wykorzystuje on architekturę *Single Page Application* (SPA) wewnątrz skryptu PHP – widoki przełączane są dynamicznie przez JavaScript (`workout.js`), co zapewnia płynność pracy bez przeładowywania strony.



Rys. 5.4. Główny ekran aktywnej sesji ze stoperem i listą ćwiczeń

Powiązanie z logiką bazy danych:
Mimo że sesja przechowywana jest tymczasowo w pamięci lokalnej przeglądarki, system stale komunikuje się z bazą danych:

- **Pobieranie struktur:** Wykorzystanie funkcji `crud.get_all_muscle_groups` do wygenerowania kafelków partii mięśniowych.
- **Inteligentne podpowiedzi (Coach Advice):** Skrypt `get_coach_advice.php` analizuje historyczne wyniki (algorytm *PROGRESSION*) i wyświetla sugestie w czasie rzeczywistym.
- **Dynamiczne placeholdery:** Dzięki `get_last_stats.php`, pola formularza podpowiadają ciężar i powtórzenia z poprzedniej sesji.
- **Persystencja danych:** Zakończenie treningu wywołuje skrypt `save_workout.php`, który przesyła obiekt JSON do procedury `save_complete_workout`.

Struktura procesu i widoki interfejsu:

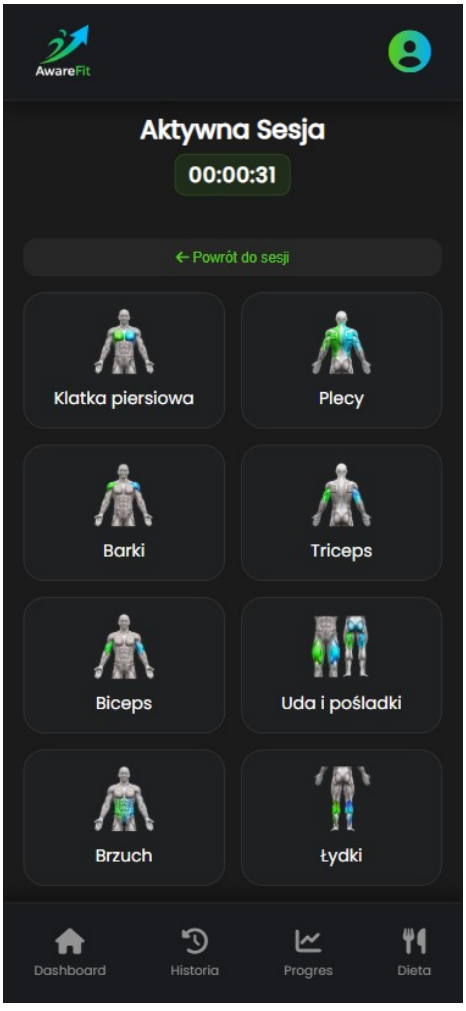
1. **Wybór partii mięśniowej:** Użytkownik widzi siatkę kafelków z ikonami reprezentującymi grupy mięśniowe (Rys. 5.5).

2. **Lista ćwiczeń:** Po wybraniu partii, skrypt `get_exercises.php` zwraca listę ćwiczeń przypisanych do danej grupy (Rys. 5.6).

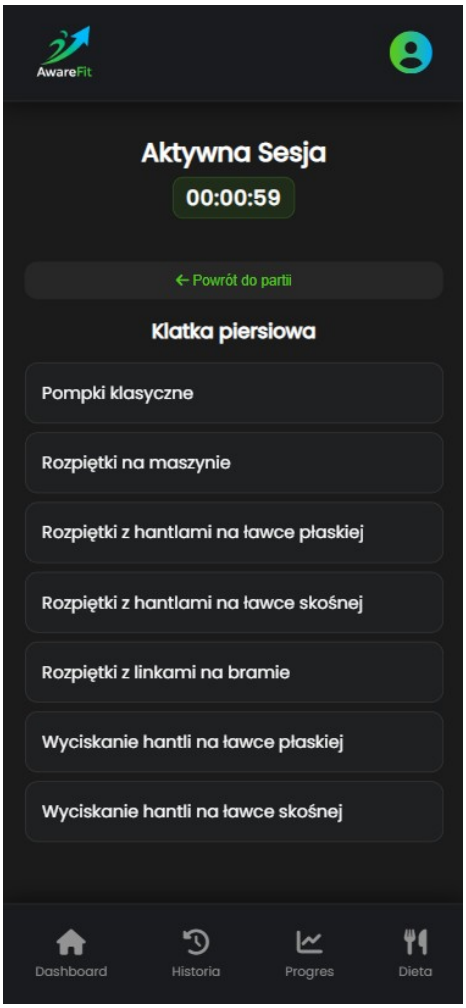
3. **Logowanie serii:** Ekran `log-set` umożliwia wprowadzanie danych. Wykorzystuje on system kolorowych ramek „Coach Advice” zależnych od statusu progresji (Rys. 5.7).

4. **Podsumowanie sesji:** Widok prezentuje wszystkie wykonane ćwiczenia wraz z ich seriami, pozwalając na edycję sesji przed jej finalnym zapisem.

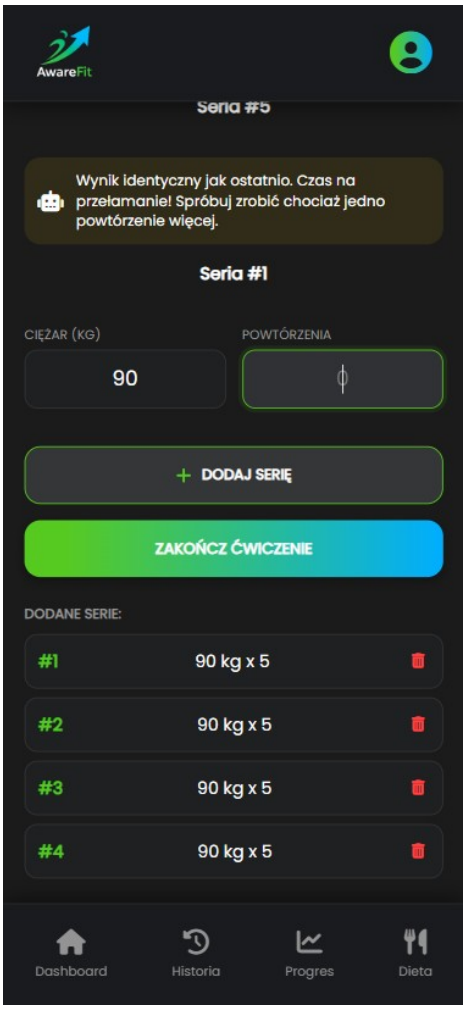
5. **Zakończenie i zapis:** System przelicza czas trwania sesji (`durationSec`) i wysyła kompletny obiekt `currentWorkout` do bazy danych (Rys. 5.8).



Rys. 5.5. Menu wyboru partii mięśniowej



Rys. 5.6. Dynamiczna lista ćwiczeń



Rys. 5.7. Ekran wprowadzania danych serii

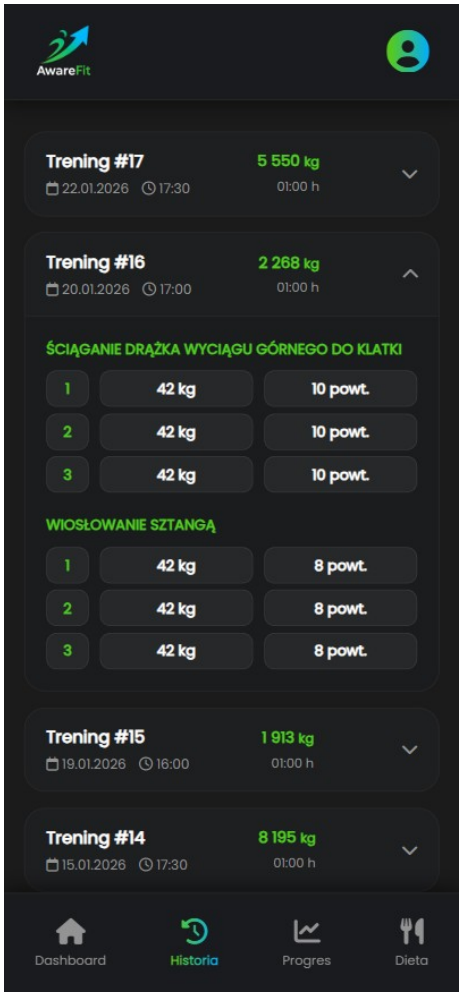
Rys. 5.8. Podsumowanie i finalizacja sesji

Zarządzanie stanem (JavaScript):

Kluczowym rozwiązaniem jest użycie `localStorage` z unikalnym kluczem `activeWorkout_${currentUserId}`. Dzięki temu, w przypadku awarii przeglądarki, dane treningu oraz czas rozpoczęcia sesji (`timerKey`) nie zostają utracone.

5.5. Widok historii treningów

Widok `history.php` służy do analizy aktywności użytkownika. Został on zaprojektowany w formie interaktywnej listy (akordeonu), która pozwala na szybki przegląd ogólnych statystyk sesji oraz ich szczegółowe rozwinięcie.



Rys. 5.9. Interfejs historii treningów z wykorzystaniem systemu akordeonów

Powiązanie z logiką bazy danych:
Skrypt `history.php` wykonuje złożone zapytanie do bazy danych, które łączy dane z funkcji tabelarycznej oraz funkcji skalarnej:

- **Agregacja sesji:** System wykorzystuje funkcję `public.get_user_workout_history`, która zwraca pełną historię ćwiczeń i serii przypisanych do danego `user_id`.
- **Dynamiczne obliczanie objętości:** Wewnątrz zapytania SQL wywoływana jest funkcja `public.calculate_workout_total_volume(workout_id)`. Pozwala to na wyświetlenie sumarycznego tonażu treningu (`db_total_volume`) bez konieczności kosztownych obliczeń po stronie PHP.
- **Mapowanie danych:** Ponieważ wynik zapytania jest „płaską” listą serii, skrypt PHP dokonuje transformacji danych do wielowymiarowej tablicy `$history`, grupując wpisy według identyfikatora treningu (`workout_id`) oraz nazw ćwiczeń (`exercise_name`).

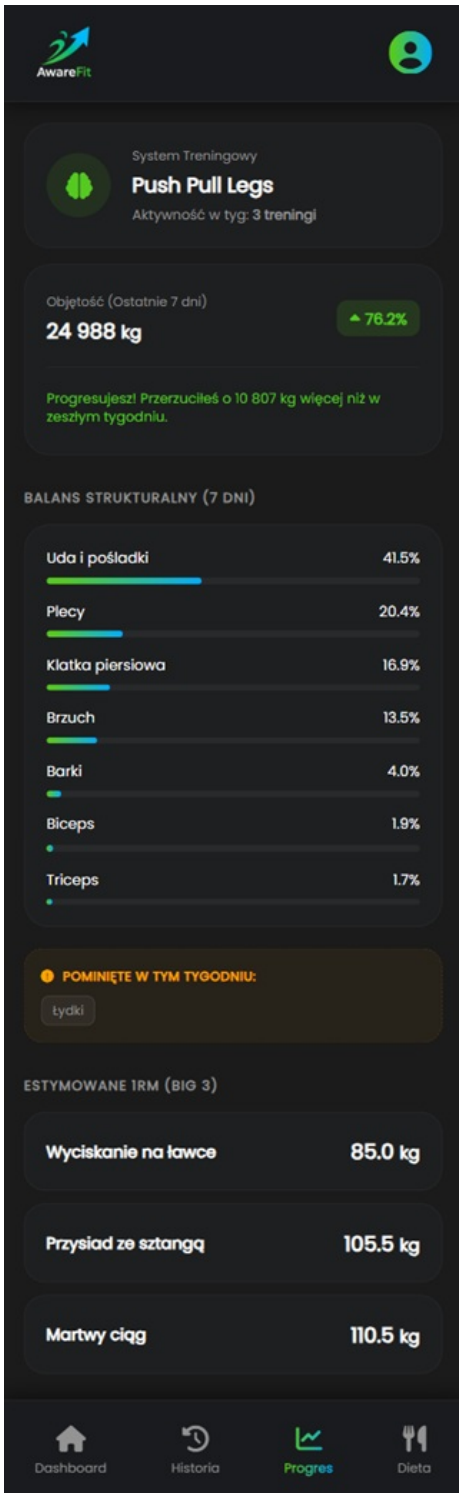
Funkcjonalność interfejsu:
Za warstwę prezentacji odpowiadają dedykowane style `history.css` oraz logika zawarta w `history.js`:

- **System Akordeonów:** Nagłówek każdego treningu wyświetla kluczowe metadane: numer treningu (`user_workout_no`), datę, czas trwania oraz łączną objętość. Dzięki funkcji `toggleAccordion`, użytkownik może dynamicznie rozwijać szczegóły konkretnej sesji.
- **Szczegółowy wykaz serii:** Po rozwinięciu sekcji, wyświetlana jest lista ćwiczeń wraz z tabelarycznym zestawieniem serii (numer serii, ciężar, liczba powtórzeń), co pozwala na precyzyjną analizę wykonanej pracy.
- **Wizualizacja tonażu:** Całkowita objętość sesji jest wyróżniona za pomocą klasy `volume-tag`, co ułatwia śledzenie progresji siłowej na przestrzeni czasu.

Obsługa skryptowa (`history.js`):
Interakcja z dokumentem realizowana jest przez prosty mechanizm manipulacji modelem DOM. Funkcja `toggleAccordion` zarządza stanem widoczności bloku `accordion-content` oraz animacją ikony nawigacyjnej (`chevron`), co znacząco poprawia czytelność interfejsu przy dużej liczbie zarejestrowanych treningów.

5.6. Widok analityczny postępów (Progress)

Moduł `progress.php` stanowi centrum analityczne aplikacji AwareFit. Jego zadaniem jest przetworzenie surowych danych treningowych na informacje o charakterze motywacyjnym i diagnostycznym. Widok ten w stopniu najwyższym wykorzystuje logikę zaszytą w warstwie bazy danych (PostgreSQL).



Rys. 5.10. Panel analityczny z zestawieniem objętości, balansu i rekordów

Integracja z zaawansowanymi funkcjami bazy danych:
W przeciwieństwie do prostych list, widok progresu opiera się na wynikach skomplikowanych obliczeń wykonywanych po stronie serwera bazy danych:

- **Detekcja systemu treningowego:** Wykorzystanie autorskiej funkcji `public.detect_training_split` pozwala systemowi automatycznie zdiagnozować, czy użytkownik trenuje systemem np. „Full Body Workout" czy „Split", bazując na częstotliwości i doborze ćwiczeń.
- **Estymacja 1RM (One Rep Max):** Dla „Wielkiej Trójki" bojów siłowych aplikacja wywołuje funkcję `public.calculate_exercise_1rm`. Wynik obliczany jest na podstawie najlepszych serii użytkownika przy użyciu matematycznych wzorów progresji (np. Brzyckiego lub Epleya) zaimplementowanych w pgSQL.
- **Analiza porównawcza objętości:** Funkcja `public.get_volume_comparison` zwraca tonaż z obecnego i poprzedniego tygodnia, co pozwala skryptowi PHP wyliczyć procentowy trend (`diff_percent`) i dobrać odpowiedni komunikat motywacyjny (trend pozytywny, stagnacja lub regresja).
- **Balans strukturalny:** Dzięki funkcji `public.get_user_muscle_balance`, aplikacja generuje wykresy słupkowe pokazujące procentowe zaangażowanie poszczególnych partii mięśniowych w skali tygodnia.

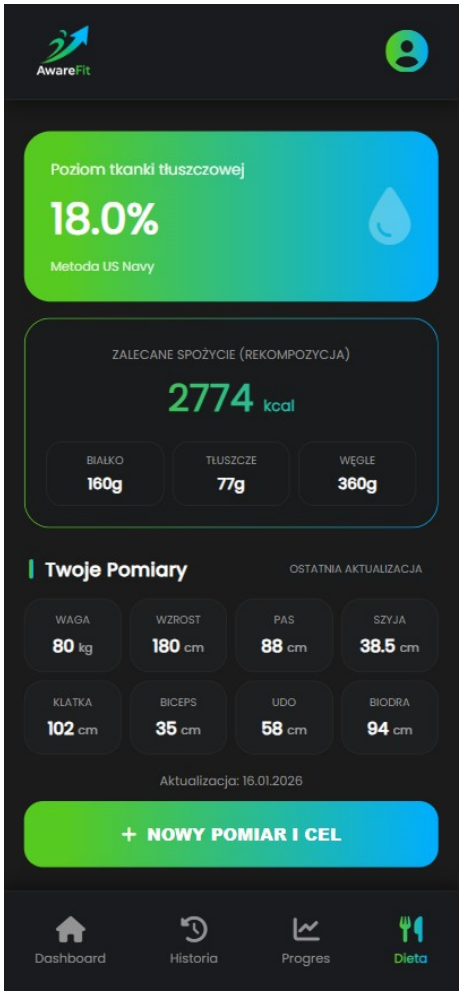
Kluczowe elementy interfejsu:

1. **Karta Systemu i Streaku:** Wyświetla aktualną metodę treningową oraz liczbę jednostek treningowych w ostatnich 7 dniach, co pozwala monitorować systematyczność.
2. **Analizator Objętości:** Sekcja ta dynamicznie zmienia kolorystykę (`trend_class`) w zależności od wyników. Wykorzystuje ikony `fa-caret-up/down` do wizualizacji kierunku zmian tonażu.
3. **Wizualizacja Balansu:** Interaktywne paski postępu (`balance-bar-fill`) pokazują, które partie dominują w planie, a sekcja „Pominięte" (wyliczana przez `array_diff` w PHP) ostrzega o zaniedbanych grupach mięśniowych.
4. **Sekcja Rekordów:** Wyświetla estymowaną siłę maksymalną, co pozwala użytkownikowi śledzić progres bez konieczności wykonywania ryzykownych prób maksymalnych.

Logika prezentacji danych:
Aplikacja dba o czytelność poprzez zaokrąglanie wyników siłowych do jednego miejsca po przecinku oraz formatowanie dużych liczb objętości (np. `number_format` dla tonażu w kg). W przypadku braku danych za dany okres, system wyświetla komunikaty pomocnicze, zachęcając do rejestracji pierwszego treningu.

5.7. Widok diety i zarządzania pomiarami ciała

Moduł `diet.php` integruje dane antropometryczne użytkownika z automatycznym systemem wyliczania zapotrzebowania kalorycznego. Jest to kluczowy element personalizacji planu treningowego, pozwalający na dostosowanie diety do aktualnych celów sylwetkowych.



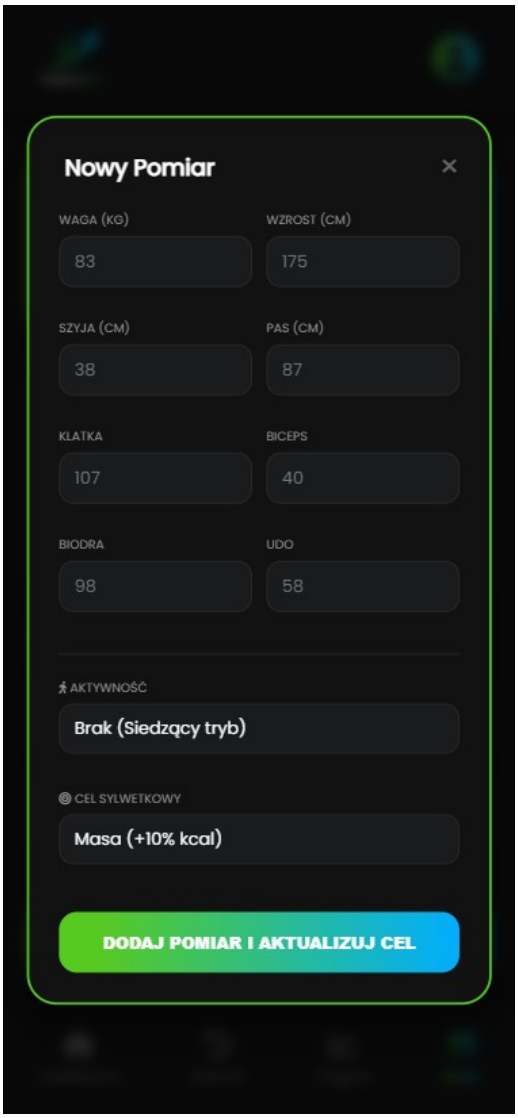
Rys. 5.11. Panel diety z wyliczonymi makroskładnikami i historią pomiarów

Zaawansowana analityka i funkcje SQL:
Strona ta w dużej mierze polega na logice obliczeniowej zaimplementowanej w PostgreSQL, co odciąża warstwę aplikacji:

- **Estymacja Body Fat:** Funkcja `public.calculate_user_bf` implementuje algorytm *US Navy Body Fat*, wykorzystując pomiary wzrostu, szyi, pasa i bioder.
- **Obliczanie zapotrzebowania (TDEE):** Skrypt wywołuje funkcję `public.calculate_user_diet_calories`, która na podstawie wzoru Mifflina-St Jeora oraz zadeklarowanego poziomu aktywności (`activity_level`), wylicza całkowite zapotrzebowanie kaloryczne.
- **Podział Makroskładników:** Funkcja `public.get_user_macros` dokonuje procentowego podziału kalorii na białka, tłuszcze i węglowodany, biorąc pod uwagę wybrany cel (masa, redukcja lub rekompozycja).

Zarządzanie danymi (Modal i Procedury):
Aplikacja umożliwia szybką aktualizację danych bez przeładowywania strony głównej dzięki zastosowaniu interaktywnego modala (`measurementModal`):

- **Zapisywanie danych:** Formularz przesyła dane do `add_measurement_action.php`, który zamiast prostego zapytania `INSERT`, wywołuje procedurę składowaną `crud.insert_body_measurement`. Gwarantuje to spójność danych i poprawność rzutowania typów (np. `double precision`, `numeric`).
- **Interaktywność (JS):** Skrypt `add_measurements.js` zarządza stanem modala oraz zapewnia odpowiednie *User Experience* poprzez wizualne potwierdzenie zapisu (animacja spinnera).



Rys. 5.12. Interfejs wprowadzania nowych pomiarów ciała i wyboru celu treningowego

Prezentacja wyników:
Interfejs użytkownika został podzielony na trzy sekcje priorytetowe:

1. **Highlight Card:** Wyświetla aktualny poziom tkanki tłuszczowej, co jest najważniejszym wskaźnikiem kompozycji ciała.

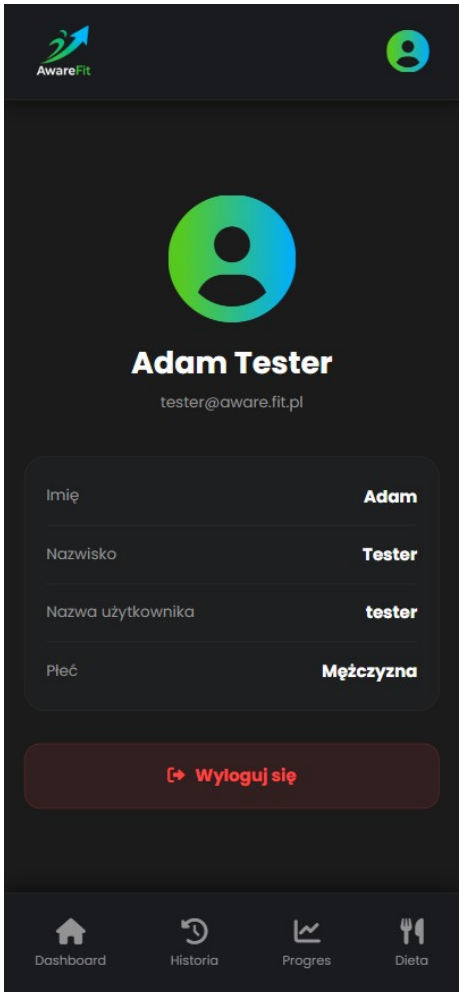
2. **Diet Summary:** Prezentuje docelową kaloryczność oraz gramaturę makroskładników w formie czytelnych kafelków.

3. **Measurements Grid:** Zestawienie wszystkich obwodów ciała w kompaktowej formie, pozwalające na szybką weryfikację ostatniej aktualizacji danych.

W przypadku braku danych w bazie, system dynamicznie wyświetla widok *empty state*, prowadząc użytkownika za rękę do pierwszego wpisu, co jest istotnym elementem *onboardingu* w aplikacji.

5.8. Widok profilu użytkownika

Widok `account.php` jest dedykowany zarządzaniu tożsamością użytkownika w aplikacji. Pozwala on na weryfikację danych konta oraz bezpieczne zakończenie sesji. Dostęp do tego panelu uzyskiwany jest poprzez interaktywną ikonę w nagłówku aplikacji (*header*).



Rys. 5.13. Podgląd profilu użytkownika z danymi osobowymi

Integracja z bazą danych:
Aplikacja wykorzystuje relacyjną strukturę tabeli `public.users` do dynamicznego generowania treści profilu:

- **Personalizacja nagłówka:** Skrypt PHP implementuje logikę priorytetyzacji wyświetlania nazw. Jeśli w bazie uzupełnione są pola `first_name` oraz `last_name`, system wyświetla pełne imię i nazwisko. W przeciwnym razie, jako identyfikator główny, prezentowany jest `username`.
- **Bezpieczeństwo sesji:** Każde żądanie do pliku `account.php` jest poprzedzone inkluzją `auth.php`, co gwarantuje, że dane z tabeli `users` są pobierane wyłącznie dla zweryfikowanego `user_id` zapisanego w sesji.
- **Lokalizacja danych:** System dokonuje mapowania wartości z bazy danych na język polski (np. konwersja wartości pola `gender` z formatu *Male/Female* na *Mężczyzna/Kobieta*).

Komponenty interfejsu:

1. **Sekcja identyfikacji (Avatar):** Centralny punkt widoku z ikoną systemową oraz adresem e-mail, co pozwala użytkownikowi szybko potwierdzić, na jakie konto jest zalogowany.
2. **Lista szczegółów (*Profile Details*):** Zestawienie kluczowych atrybutów konta w formie czytelnych par etykieta-wartość.
3. **Zarządzanie sesją:** Przycisk wylogowania (`logout-btn`) odsyła do `index.php`.

Warstwa wizualna:

Podobnie jak reszta aplikacji, widok profilu korzysta z arkusza `global.css` oraz dedykowanego pliku `account.css`. Zastosowano czcionkę *Poppins* oraz bibliotekę *Font Awesome*, aby zachować spójność wizualną z systemem Android/iOS, co nadaje aplikacji charakter natywnego narzędzia mobilnego.

6. Dostęp zdalny do bazy danych

W niniejszym rozdziale przedstawiono koncepcję oraz techniczną realizację komunikacji pomiędzy aplikacją AwareFit a systemem zarządzania bazą danych PostgreSQL.

6.1. Koncepcja dostępu zdalnego

Model architektury systemowej:
Aplikacja AwareFit została zaprojektowana jako progresywna platforma webowa działająca w modelu *Client-Server*. Koncepcja dostępu zdalnego opiera się na separacji interfejsu użytkownika od fizycznej lokalizacji danych.

- **Dostęp przez protokół HTTP/HTTPS:** Użytkownik uzyskuje dostęp do aplikacji poprzez przeglądarkę internetową. Eliminuje to konieczność instalacji bazy danych na urządzeniu końcowym – wszystkie operacje są procesowane zdalnie na serwerze.
- **Rola serwera pośredniczącego:** Serwer Apache (XAMPP) pełni rolę bramy (Gateway). Przyjmuje on żądania od użytkownika, a następnie nawiązuje zdalne połączenie z silnikiem PostgreSQL, aby pobrać lub zapisać dane.
- **Niezależność lokalizacji:** Dzięki zastosowaniu standardu TCP/IP oraz sterowników PDO, system jest przygotowany do pracy w rozproszonym środowisku sieciowym. Baza danych może znajdować się na dedykowanym hostingu, podczas gdy pliki strony serwowane są z innej lokalizacji.

Zalety koncepcji webowej:
Dzięki takiemu podejściu, dostęp zdalny zapewnia:

- **Mobilność:** Użytkownik może edytować swój trening na telefonie w siłowni, a następnie analizować postępy na komputerze domowym – dane są zawsze zsynchronizowane w centralnej bazie.
- **Bezpieczeństwo:** Kluczowe dane (hasła, wyniki) nigdy nie są przechowywane bezpośrednio w przeglądarce, a jedynie przesyłane bezpiecznym kanałem do bazy danych.

6.2. Opis realizacji dostępu zdalnego

Realizacja połączenia opiera się na rozszerzeniu `PDO` (PHP Data Objects). Środowisko uruchomieniowe zostało oparte na pakiecie **XAMPP** (serwer Apache) oraz systemie **PostgreSQL** zarządzanym przez narzędzie **pgAdmin 4**.

Implementacja techniczna połączenia:
Kluczowym elementem jest plik `includes/db.php`, który inicjuje sesję komunikacyjną. Poniżej przedstawiono kod źródłowy realizujący to zadanie:

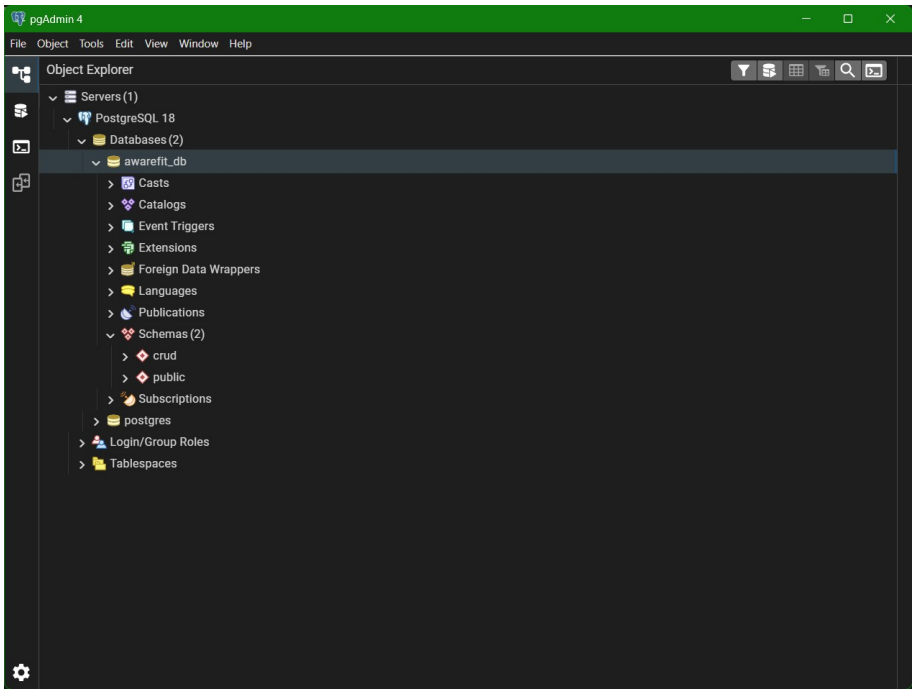
```
<?php
$host = "localhost";
$port = "5432";
$dbname = "awarefit_db";
$user = "postgres";
$password = "psql";

try {
    $dsn = "pgsql:host=$host;port=$port;dbname=$dbname";
    $pdo = new PDO($dsn, $user, $password, [
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION
    ]);
} catch (PDOException $e) {
    die("Błąd połączenia: " . $e->getMessage());
}
?>
```

Autoryzacja dostępu w warstwie aplikacji:
Aby zapewnić, że zdalny dostęp do danych jest bezpieczny, każdy skrypt wymagający połączenia z bazą chroniony jest przez mechanizm sesji zawarty w pliku `auth.php`:

```
<?php
session_start();
if (!isset($_SESSION['user\_id'])) {
    header("Location: index.php");
    exit();
}
?>
```

Prezentacja i monitoring (pgAdmin 4):
Do monitorowania stanu połączeń oraz weryfikacji poprawności zapisu danych wykorzystywane jest narzędzie **pgAdmin 4**. Pozwala ono na podgląd aktywnych procesów serwera oraz zarządzanie strukturą tabel (Rys. 6.1).



Rys. 6.1. Struktura bazy danych awarefit_db widoczna w narzędziu pgAdmin 4

Weryfikacja działania:

Poprawność realizacji dostępu zdalnego została potwierdzona poprzez:

1. **Testy połączenia:** Weryfikacja reakcji aplikacji na błędne dane logowania (poprawne przechwycenie wyjątku `PDOException`).
2. **Integracja z Apache:** Poprawne procesowanie żądań przez serwer Apache wewnątrz środowiska XAMPP i stabilne utrzymywanie połączenia z procesem `postgres.exe`.
3. **Zdalne zarządzanie:** Możliwość edycji i przeglądu danych w czasie rzeczywistym bezpośrednio w interfejsie graficznym pgAdmin 4.

7. Implementacja i migracja bazy danych MongoDB

W ramach rozszerzenia projektu oraz zapewnienia skalowalności systemu AwareFit, przeprowadzono proces migracji danych z relacyjnej bazy danych PostgreSQL do nierelacyjnej bazy MongoDB. Poniższy rozdział opisuje techniczne aspekty tego procesu, od mapowania schematu po eksport danych do formatu JSON.

7.1. Proces migracji danych SQL do MongoDB

Do przeprowadzenia transformacji danych wykorzystano narzędzie *MongoDB Relational Migrator*, które umożliwia automatyczne mapowanie struktur relacyjnych na model dokumentowy.

7.1.1. Konfiguracja połączenia i mapowanie schematu

Proces rozpoczęto od nawiązania połączenia z lokalną bazą danych `awarefit_db`. W narzędziu konfiguracyjnym zdefiniowano parametry hosta (*localhost*), nazwę użytkownika oraz hasło systemowe.

Następnie przeprowadzono mapowanie struktur:

- Z bazy źródłowej wybrano wszystkie 7 dostępnych tabel.
- Zastosowano rekomendowany przez narzędzie schemat bazy MongoDB, zachowując oryginalne nazewnictwo kolekcji.
- Wygenerowano finalny schemat dokumentowy, który stał się podstawą do dalszych operacji.

7.1.2. Migracja do chmury MongoDB Atlas

W celu zapewnienia dostępności danych w architekturze rozproszonej, wykorzystano usługę chmurową *MongoDB Atlas*.

1. W chmurze utworzono bazę danych o nazwie zgodnej z oryginałem (`awarefit_db`).
2. Ustanowiono połączenie między lokalnym narzędziem migracyjnym a klastrem w chmurze przy użyciu dedykowanego ciągu znaków (*Connection String*).
3. Uruchomiono proces typu *Snapshot*, który przetworzył rekordy SQL na format dokumentowy BSON i zainicjował ich przesył do chmury.

Po zakończeniu operacji zweryfikowano poprawność transferu. System potwierdził migrację wszystkich 7 kolekcji oraz ponad 1000 rekordów, co wykazano poprzez porównanie zawartości tabel w PostgreSQL z odpowiadającymi im dokumentami w MongoDB.

7.2. Eksport i backup danych NoSQL

W celu zapewnienia przenoszalności projektu oraz możliwości odtworzenia bazy w dowolnym środowisku lokalnym, przeprowadzono procedurę eksportu danych przy użyciu narzędzia **MongoDB Compass**.

7.2.1. Procedura eksportu do formatu JSON

Proces archiwizacji danych przebiegał według następujących kroków:

1. Nawiązano połączenie z bazą produkcyjną przy użyciu zautoryzowanego ciągu znaków.
2. Dla każdej z 7 kolekcji (m.in. `users`, `workouts`, `body_measurements`) wykonano indywidualną operację eksportu.
3. Jako format wyjściowy wybrano **JSON**, co pozwala na łatwy podgląd struktury danych w dowolnym edytorze tekstowym.

7.2.2. Weryfikacja plików wynikowych

W wyniku przeprowadzonych prac uzyskano komplet plików JSON, które zostały umieszczone w repozytorium projektu. Pliki te stanowią kopię zapasową bazy danych i umożliwiają natychmiastowy import danych do nowej instancji MongoDB za pomocą komendy `mongoimport`. Dzięki temu struktura danych jest niezależna od konkretnego hosta i może być weryfikowana przez prowadzącego w dowolnym momencie.

8. Instrukcja uruchomienia aplikacji

Niniejszy rozdział opisuje procedurę konfiguracji środowiska lokalnego oraz kroki niezbędne do poprawnego uruchomienia aplikacji AwareFit.

8.1. Wymagania systemowe i programowe

Do poprawnego działania aplikacji wymagane jest zainstalowanie i skonfigurowanie następujących komponentów:

- **XAMPP**: Serwer lokalny z aktywnym modułem Apache.
- **PostgreSQL**: Silnik bazy danych wraz z narzędziem do zarządzania **pgAdmin 4**.
- **PHP PDO PostgreSQL**: Rozszerzenie `php_pdo_pgsql` musi być odkomentowane w pliku konfiguracyjnym `php.ini`.

8.2. Konfiguracja bazy danych

W celu przygotowania warstwy danych należy wykonać poniższe kroki:

1. Uruchomić narzędzie pgAdmin 4 i utworzyć nową bazę danych o nazwie `awarefit_db`.
2. Zaimportować strukturę bazy danych oraz funkcje systemowe z dostarczonego pliku SQL.
3. Parametry połączenia:
    - **Użytkownik**: `postgres`
    - **Hasło**: `psql` (Uwaga: W przypadku instalacji na innym serwerze należy zaktualizować hasło w pliku konfiguracyjnym aplikacji).

8.3. Instalacja i uruchomienie

Aplikacja jest zaimplementowana jako serwis webowy i powinna być hostowana wewnątrz środowiska XAMPP.

1. Folder z kodem źródłowym (zawierający pliki PHP oraz interfejs graficzny GUI) należy umieścić w katalogu:
   `C:\xampp\htdocs\Projekt_Bazy_Danych\`
2. W panelu kontrolnym XAMPP należy uruchomić usługę **Apache**.
3. Aplikacja jest dostępna pod adresem lokalnym:

   http://localhost/Projekt_Bazy_Danych/index.php

8.4. Konfiguracja połączenia z bazą danych (PHP)

Aplikacja wykorzystuje sterownik **PDO** (*PHP Data Objects*) do komunikacji z bazą PostgreSQL. Kluczowe parametry połączenia zdefiniowane są w pliku konfiguracyjnym:

`include/db.php`

W przypadku zmiany parametrów serwera (np. przeniesienia bazy na inny host lub zmiany hasła systemowego), należy zaktualizować poniższy fragment kodu:

Listing 8.1. Zawartość pliku include/db.php

```
<?php
$host = "localhost";
$port = "5432";
$dbname = "awarefit_db";
$user = "postgres";
$password = "psql";

try {
    $dsn = "pgsql:host=$host;port=$port;dbname=$dbname";
    $pdo = new PDO($dsn, $user, $password, [
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION
    ]);
} catch (PDOException $e) {
    die("Błąd połączenia: " . $e->getMessage());
}
?>
```

8.4.1. Kluczowe zmienne konfiguracyjne

- `$host`: Adres serwera bazy danych (domyślnie `localhost`).
- `$dbname`: Nazwa docelowej bazy danych (`awarefit_db`).
- `$password`: Hasło użytkownika PostgreSQL (w tej konfiguracji ustawione na `psql`).
- `PDO::ATTR_ERRMODE`: Ustawienie `ERRMODE_EXCEPTION` wymusza rzucanie wyjątków w przypadku błędnych zapytań SQL, co ułatwia debugowanie aplikacji.

8.5. Dostęp do konta testowego

W celu poprawnej weryfikacji wszystkich funkcjonalności systemu bazodanowego (takich jak obliczanie objętości treningowej, generowanie wykresów progresji czy automatyczne wykrywanie rodzaju splitu), w bazie danych zostało przygotowane dedykowane konto testera.

Konto to zawiera historyczne dane treningowe niezbędne do poprawnego renderowania statystyk. Aby wejść do aplikacji, należy użyć poniższych danych uwierzytelniających:

- **Nazwa użytkownika**: `tester`
- **Hasło**: `test`

8.6. Instrukcja generowania wykresu

Po zalogowaniu na konto `tester`, użytkownik uzyskuje dostęp do głównego panelu sterowania (*Dashboard*). Aby zweryfikować poprawność działania dynamicznych wykresów progresji objętościowej, należy przeprowadzić procedurę wyboru ćwiczenia zgodnie z historią treningową zapisaną w bazie danych.

Wykresy generowane są na podstawie unikalnych par: partia mięśniowa oraz konkretne ćwiczenie. Aby zobaczyć wykres, należy w panelu wybrać:

- **Partia mięśniowa**: `Klatka piersiowa`
- **Ćwiczenie**: `Wyciskanie sztangi na ławce płaskiej`

8.7. Dostęp do bazy danych MongoDB (NoSQL)

W ramach rozszerzenia projektu, system AwareFit oferuje dostęp do danych zmigrowanych do bazy NoSQL. Baza ta jest utrzymywana w chmurze (*MongoDB Atlas*), co umożliwia jej zdalną weryfikację bez konieczności lokalnej konfiguracji serwera MongoDB.

8.7.1. Procedura logowania przy użyciu narzędzia MongoDB Compass

W celu przeglądania struktury dokumentowej bazy danych wykorzystywane jest darmowe narzędzie **MongoDB Compass**. Proces uzyskania dostępu do danych przebiega według następujących kroków:

1. Po uruchomieniu programu **MongoDB Compass** w menu głównym wybierana jest opcja dodania nowego połączenia (*New Connection*).
2. W polu **URI** umieszczany jest dedykowany ciąg znaków (*Connection String*):

   ```
   mongodb+srv://pgrochowalski:AwareFit2026@cluster0.ecve17l.mongodb.net/?appName=Cluster0
   ```

3. W zakładce **Advanced Connection Options**, w sekcji **Authentication**, weryfikowane są dane uwierzytelniające.
4. W odpowiednie pola wprowadzane są poniższe dane:
   - **Username**: `pgrochowalski`
   - **Password**: `AwareFit2026`
5. Po zatwierdzeniu danych przyciskiem **Connect**, następuje nawiązanie połączenia z bazą danych o nazwie `awarefit_db`, która zawiera zmigrowane kolekcje systemu AwareFit.

8.7.2. Możliwości weryfikacji danych NoSQL

Poprawne nawiązanie połączenia umożliwia dostęp do wszystkich 7 kluczowych kolekcji (odpowiedników relacyjnych tabel SQL). System zapewnia w tym zakresie:

- Podgląd dokumentów w formatach JSON oraz BSON.
- Weryfikację integralności danych poprzez analizę zaimportowanych rekordów.

9. Podsumowanie techniczne projektu AwareFit

Niniejszy rozdział stanowi syntetyczne zestawienie informacji o architekturze, technologiach oraz przeznaczeniu aplikacji AwareFit, bazujące na zaimplementowanych rozwiązaniach programistycznych.

9.1. Przeznaczenie i cele aplikacji

Głównym celem aplikacji AwareFit jest dostarczenie użytkownikowi kompleksowego narzędzia do monitorowania progresji siłowej oraz zarządzania dietą. System został zaprojektowany z myślą o:

- **Efektywnym zapisywaniu jednostek treningowych:** Intuicyjny system rejestrowania sesji treningowych, pozwalający na precyzyjne dokumentowanie każdego ćwiczenia, serii, obciążenia oraz liczby powtórzeń, co stanowi fundament rzetelnej analizy postępów.
- **Automatyzacji obliczeń parametrów treningowych:** Wykorzystanie mechanizmów bazodanowych do bieżącego wyliczania tonażu treningowego oraz ciągłości aktywności użytkownika.
- **Wizualizacji progresu:** Dynamiczne generowanie zestawień graficznych obrazujących wzrost siły i objętości w czasie.
- **Monitorowaniu kompozycji ciała:** Automatyczne dostosowywanie wyliczeń makroskładników i zapotrzebowania kalorycznego na podstawie wprowadzanych pomiarów antropometrycznych.

9.2. Struktura bazy danych i logiki serwerowej

Architektura bazy danych PostgreSQL opiera się na dwóch głównych schematach, co zapewnia separację logiki operacyjnej od fizycznej struktury przechowywania danych:

- **Schemat (`public`):** Przechowuje strukturę tabel bazy danych oraz zaawansowane algorytmy.
- **Schemat (`crud`):** Zawiera wszystkie procedury oraz funkcje CRUD.

9.3. Interfejs użytkownika (GUI) i warstwa frontendowa

Warstwa wizualna została zbudowana w oparciu o podejście *Mobile-First* oraz nowoczesne standardy webowe, zapewniając pełną responsywność na urządzeniach mobilnych:

- **Technologie:** HTML5, CSS3 (z wykorzystaniem zmiennych CSS dla ujednoliconego motywu graficznego) oraz nowoczesny JavaScript (ES6+).
- **Wizualizacja danych:** Wykorzystanie profesjonalnych bibliotek do tworzenia interaktywnych wykresów liniowych, które pozwalają użytkownikowi na intuicyjną analizę historii treningowej.
- **Interaktywność:** Zastosowanie systemowych okien modalnych do prezentacji danych szczegółowych oraz mechanizmów pamięci lokalnej przeglądarki, które umożliwiają kontynuację sesji treningowej nawet po przypadkowym zamknięciu aplikacji.
- **Estetyka:** Wykorzystanie nowoczesnej typografii oraz płynnych animacji interfejsu, które podnoszą komfort użytkowania i czytelność prezentowanych statystyk.

9.4. Integracja technologii (Tech Stack)

System stanowi spójne połączenie warstwy prezentacji, logiki biznesowej oraz trwałego składowania danych:

- **Bezpieczeństwo i komunikacja:** Wykorzystanie języka PHP jako pośrednika pomiędzy interfejsem a bazą danych. Zastosowano bezpieczne połączenia poprzez sterowniki bazodanowe.
- **Asynchroniczność:** Skrypty klienckie dynamicznie przetwarzają dane, co pozwala na aktualizację kluczowych elementów panelu sterowania (*Dashboard*) bez konieczności przeładowywania całej strony.
- **Środowisko uruchomieniowe:** Serwer Apache oraz silnik PostgreSQL, zintegrowane w ramach lokalnego środowiska deweloperskiego, zapewniające wysoką stabilność podczas prac nad projektem.

9.5. Dostępność projektu i kontrola wersji

Projekt AwareFit jest rozwijany w oparciu o nowoczesne standardy zarządzania kodem źródłowym, co zapewnia bezpieczeństwo danych oraz pełną przejrzystość historii zmian:

- **Repozytorium zdalne:** Pełny kod źródłowy aplikacji, wraz z dokumentacją bazy danych oraz plikami konfiguracyjnymi środowiska, jest dostępny pod adresem: *https://github.com/MichalJanikUR/ProjektBazyDanych.git*.
- **System kontroli wersji:** Wykorzystanie systemu Git umożliwiło precyzyjne śledzenie postępów prac, zarządzanie równoległymi zmianami w kodzie oraz bezpieczne wdrażanie poprawek bez ryzyka utraty stabilności głównej wersji systemu.
- **Struktura plików i dokumentacja:** Kod w repozytorium został zorganizowany w sposób modularny, z wyraźnym podziałem na warstwy aplikacji (logika PHP, interfejs użytkownika, skrypty bazy danych). Pozwala to na łatwą weryfikację implementacji oraz ewentualną rozbudowę systemu w przyszłości.

Spis rysunków

Załącznik nr 2 do Zarządzenia nr 228/2021 Rektora Uniwersytetu Rzeszowskiego z dnia 1 grudnia 2021 roku w sprawie ustalenia procedury antyplagiatowej w Uniwersytecie Rzeszowskim

OŚWIADCZENIE STUDENTA O SAMODZIELNOŚCI PRACY

……Oliwier Hędrzak, Michał Janik……
Imię (imiona) i nazwisko studenta

Wydział Nauk Ścisłych i Technicznych

……Informatyka……
Nazwa kierunku

……134913, 134915……
Numer albumu

1. Oświadczam, że moja praca projektowa pt.: Dziennik treningowy AwareFit – dokumentacja projektowa
   1) została przygotowana przeze mnie samodzielnie*,
   2) nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2021 r., poz. 1062) oraz dóbr osobistych chronionych prawem cywilnym,
   3) nie zawiera danych i informacji, które uzyskałem/am w sposób niedozwolony,
   4) nie była podstawą otrzymania oceny z innego przedmiotu na uczelni wyższej ani mnie, ani innej osobie.
2. Jednocześnie wyrażam zgodę/nie wyrażam zgody** na udostępnienie mojej pracy projektowej do celów naukowo–badawczych z poszanowaniem przepisów ustawy o prawie autorskim i prawach pokrewnych.

(miejscowość, data) (czytelny podpis studenta)

* Uwzględniając merytoryczny wkład prowadzącego przedmiot

** – niepotrzebne skreślić