

Zaawansowane technologie internetowe

Dokumentacja projektu zaliczeniowego

Autor: Artur Róg

Spis treści

| | | |
|----------|---------------------------------------|-----------|
| 1 | Opis i temat projektu | 2 |
| 2 | Baza danych | 3 |
| 2.1 | Architektura | 3 |
| 2.2 | ElephantSQL | 5 |
| 2.3 | Serwisy korzystające z bazy | 6 |
| 3 | Rozwiązanie chmurowe | 8 |
| 4 | Front aplikacji | 10 |
| 5 | Aplikacja | 12 |
| 5.1 | Bezpieczeństwo | 12 |
| 5.2 | Logowanie i rejestracja | 12 |
| 6 | Podręcznik użytkownika | 15 |
| 6.1 | Panel główny | 15 |
| 6.2 | Wykresy | 17 |
| 6.2.1 | Treningi | 17 |
| 6.2.2 | Ćwiczenia | 18 |
| 6.3 | Treningi | 19 |
| 6.4 | Ćwiczenia | 19 |
| 6.5 | Dodaj ćwiczenie | 20 |

1 Opis i temat projektu

Aplikacja stworzona w ramach projektu to personalny trener online. Aplikacja pozwala na:

- dodawanie własnych ćwiczeń,
- dodawanie zrealizowanych treningów,
- śledzenie swojej wagi,
- śledzenie spożytych kalorii,
- śledzenie progresu ciężaru dla poszczególnych ćwiczeń,
- śledzenie ilości treningów w poszczególnych tygodniach roku,

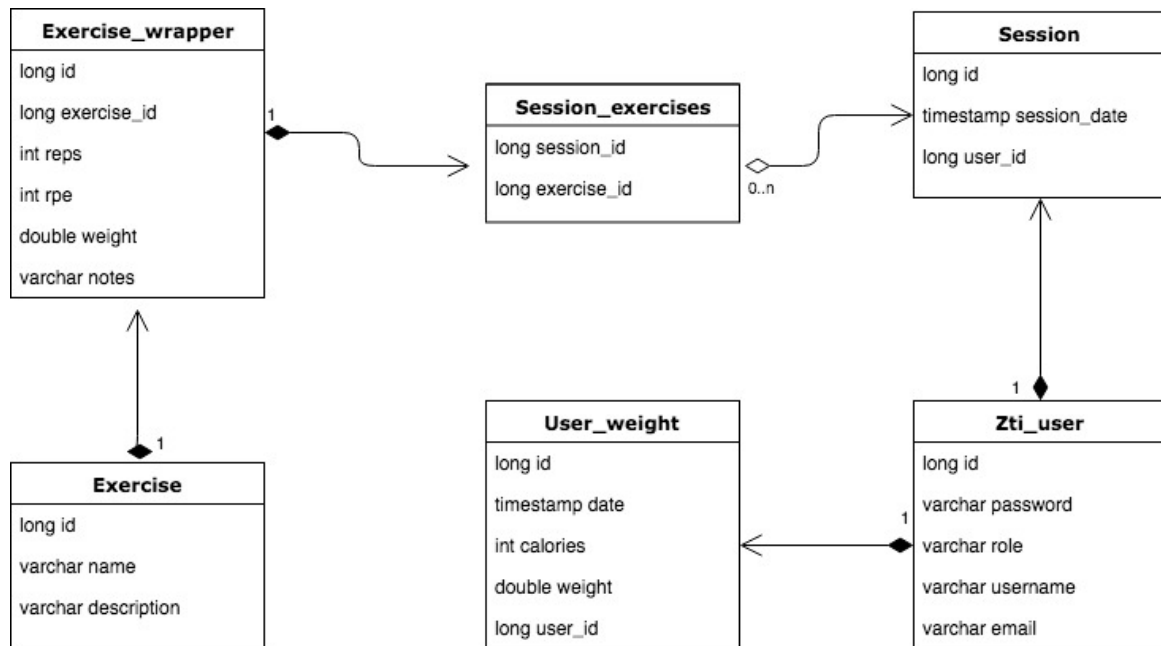
Aplikacja została zrealizowana zgodnie ze wzorcem RESTful z wykorzystaniem technologii Spring oraz Spring Boot.

Dostęp do bazy danych zrealizowany został w oparciu o Spring Data w rozwiązaniu chmurowym z wykorzystaniem bazy ElephantSQL.

Strona klienta zrealizowana została z wykorzystaniem serwisu WWW i języka JavaScript.

2 Baza danych

2.1 Architektura



Rysunek 1: Schemat bazy danych

Tabela Zti_user

W tabeli przechowywani są zarejestrowani użytkownicy.
Składa się z kolumn:

- id** - unikalne id,
- password** - hasło użytkownika,
- role** - rola użytkownika (ADMIN/USER),
- username** - nazwa użytkownika,
- email** - email użytkownika

Tabela User_weight

W tabeli przechowywane są dane z dzienniczka użytkownika, tj. waga i spożyte danego dnia kalorie.

Składa się z kolumn:

- id** - unikalne id,
- date** - data, kiedy dane są wprowadzane,
- calories** - spożyte danego dnia kalorie,
- weight** - waga danego dnia,
- user_id** - id użytkownika

Tabela Exercise

W tabeli zapisywane są wszystkie ćwiczenia dodane przez użytkowników aplikacji.
Składa się z kolumn:

- id** - unikalne id,
- name** - nazwa ćwiczenia,
- description** - opis ćwiczenia

Tabela Session

W tabeli przechowywane są treningi (sesje treningowe) użytkownika.
Składa się z kolumn:

- id** - unikalne id,
- session_date** - data wykonania treningu,
- user_id** - id użytkownika, który wykonał trening

Tabela Exercise_wrapper

W tabeli przechowywane są ćwiczenia wykonane przez użytkownika podczas sesji treningowej.
Dla każdego wykonywanego ćwiczenia przechowywane są ponadto dodatkowe informacje.
Składa się z kolumn:

- id** - unikalne id,
- exercise_id** - id wykonanego ćwiczenia,
- reps** - ilość wykonanych powtórzeń,
- rpe** - trudność (w skali 1 do 10) wykonanych powtórzeń, gdzie 1 - bardzo łatwo, 10 - bardzo ciężko.
- weight** - ciężar z jakim wykonywane było ćwiczenie,
- reps** - dodatkowe uwagi, spostrzeżenia

Tabela Session_exercises

W tabeli zapisywane są wszystkie ćwiczenia dodane przez użytkowników aplikacji.
Składa się z kolumn:

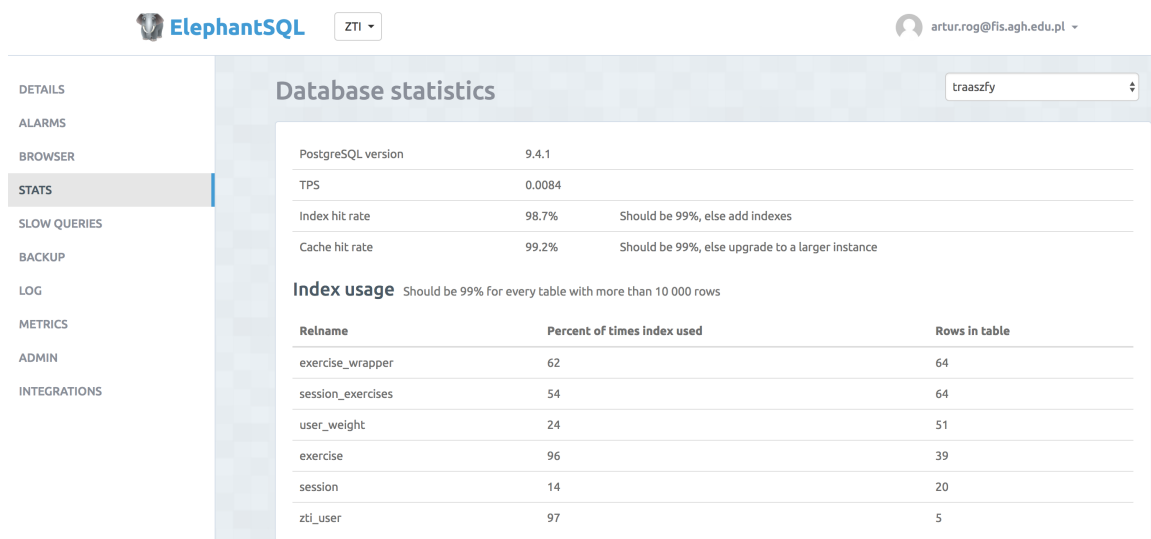
- id** - unikalne id,
- name** - nazwa ćwiczenia,
- description** - opis ćwiczenia

2.2 ElephantSQL

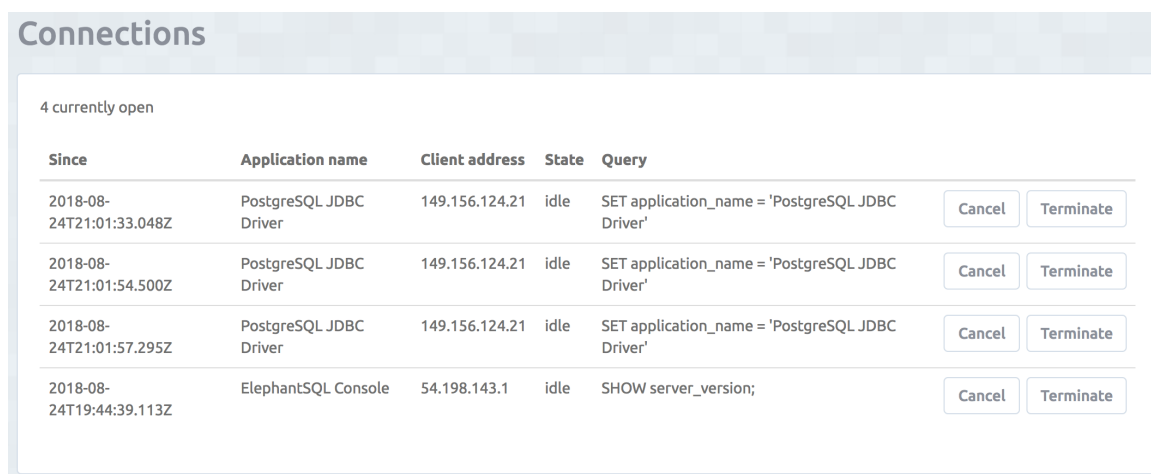
Dostęp do bazy danych zrealizowany został przy użyciu Spring Data. Wykorzystana baza hostowana jest w chmurze (PostgreSQL as a service). ElephantSQL w podstawowym, darmowym

planie udostępnia za darmo możliwość korzystania z baz danych hostowanych na centrach danych Amazona.

Podstawowy plan pozwala na stworzenie puli zaledwie pięciu połączeń do bazy. Interfejs ElephantSQL pozwala na przeglądanie podstawowych statystyk bazy, wykonywanie zapytań, śledzenie puli połączeń oraz tworzenie backupów bazy.



Rysunek 2: Interfejs ElephantSQL



Rysunek 3: Pula otwartych połączeń

Operacje na danych

Spring Boot dostarczany jest wraz z implementacjami konfiguracji połączeń z bazą danych. To, co pozostaje użytkownikowi, to określenie parametrów połączenia z bazą danych w pliku **application.properties**

```
spring.datasource.url=jdbc:postgresql://horton.elephantsql.com:5432/traaszfy
spring.datasource.username=traaszfy
spring.datasource.password=7W_HbS-pXGMLSfwrMh5LElHGl0rDSV6
spring.datasource.driverClassName=org.postgresql.Driver
spring.datasource.max-idle=2
spring.datasource.hikari.maximum-pool-size=3
spring.datasource.hikari.connection-timeout=60000
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.PostgreSQLDialect
spring.jpa.hibernate.ddl-auto=update
```

Rysunek 4: Parametry połączenia z bazą.

Powyższe parametry pozwalają na określenie:

- url do bazy danych,
- użytkownika,
- hasła użytkownika,
- używanego sterownika,
- maksymalnej ilości połączeń, które są utrzymywane przez cały czas,
- maksymalnej puli połączeń

2.3 Serwisy korzystające z bazy

Po stronie serwera, zapytania do bazy odbywają się z wykorzystaniem **CrudRepository** dostarczanego wraz ze **Spring Data**. Każda z tabel posiada własny interfejs repozytorium, rozszerzający repozytorium Springowe. Takie podejście pozwala na bardzo

intuicyjną pracę z bazą. Dzięki CrudRepository można zrezygnować z tworzenia zapytań ręcznie. Poprzez nazywanie metod zgodnie ze schematem Spring Data sam konstruuje odpowiednie zapytania.

```
public interface ExerciseRepository extends CrudRepository<Exercise, Long> {
    List<Exercise> findByName(String name);
    List<Exercise> findAll();
}
```

Rysunek 5: Repozytorium Exercise

Powyższy interfejs ma zdefiniowane dwie metody:

findAll - zwracający wszystkie ćwiczenia,

findByName - zwracający ćwiczenia o podanej nazwie

Tak zdefiniowana metoda wystarczy, by Spring znalazł ćwiczenia o żądanej nazwie. Nie jest potrzebna żadna dodatkowa implementacja.

Repozytoria wykorzystywane są w **Serwisach**. Każda z tabel ma odpowiednio swój serwis. Jest to kolejna warstwa, gdzie na danych uzyskanych z repozytorium wykonywane są pewne operacje.

```
public Set<Exercise> findAllUserExercises() {  
    List<ExerciseWrapper> wrappers = exercisesFromTrainingSessionsByUser();  
  
    return wrappers.stream()  
        .map(ExerciseWrapper::getExercise)  
        .collect(Collectors.toSet());  
}
```

Rysunek 6: Metoda serwisu

Powyżej przedstawiony jest przykład metody serwisu, która pobiera wszystkie ćwiczenia wykonane przez użytkownika jako wrappery, a następnie wyciąga z nich same ćwiczenia (bez dodatkowych informacji, np. ciężaru czy ilości powtórzeń).

Kolejną warstwą abstrakcji są kontrolery. Tutaj dane uzyskane z serwisów przesyłane są do widoków i prezentowane bezpośrednio użytkownikowi.

```
@RequestMapping("/exercises")  
public String findAllExercises(Model model){  
    List<Exercise> exercises = exerciseService.findAllExercise();  
    model.addAttribute( attributeName: "exercises", exercises);  
    return PREFIX+"exercises";  
}
```

Rysunek 7: Metoda kontrolera

3 Rozwiązanie chmurowe

Aplikacja została umieszczona na platformie chmurowej **Heroku**. Jest to jedna z pierwszych tego typu platform, rozwijana już od ponad dekady. Oferuje dość szeroki asortyment darmowych usług, przy jednoczesnej bardzo prostej integracji.

Platforma Heroku wykorzystuje Git jako podstawowy sposób wdrażania aplikacji (istnieją inne sposoby transportu kodu źródłowego do Heroku, w tym przez interfejs API). Git to potężny, rozproszony system kontroli wersji, używany przez wielu programistów do zarządzania i wersjonowania kodu źródłowego.

By wdrożyć aplikację nie jest potrzebna żadna zbędna konfiguracja. Wystarczy w folderze projektu stworzyć repozytorium **.git**.

Do wdrożenia potrzebne jest utworzyć repozytorium Git dla aplikacji i dodać do niego cały kod, uruchamiając polecenia:

```
$ git init
$ git add .
$ git commit -m "first commit"
```

Rysunek 8: Commit plików projektu

Aby rozpocząć wdrożenie, należy najpierw udostępnić nową aplikację Heroku.

```
$ heroku create
Creating nameless-lake-8055 in organization heroku... done, stack is cedar-14
http://nameless-lake-8055.herokuapp.com/ | git@heroku.com:nameless-lake-8055.git
Git remote heroku added
```

Rysunek 9: Utworzenie aplikacji na heroku

Zostaje też utworzone lokalne repozytorium o nazwie heroku. Dla aplikacji generowana jest losowa nazwa, którą można potem zmienić. Możliwe jest teraz wdrożenie aplikacji:


```
$ git push heroku master
Initializing repository, done.
Counting objects: 110, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (87/87), done.
Writing objects: 100% (110/110), 212.71 KiB | 0 bytes/s, done.
Total 110 (delta 30), reused 0 (delta 0)

-----> Java app detected
-----> Installing OpenJDK 1.8... done
-----> Installing Maven 3.3.3... done
-----> Executing: mvn -B -DskipTests=true clean install
[INFO] Scanning for projects...
...
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 11.417s
[INFO] Finished at: Thu Sep 11 17:16:38 UTC 2014
[INFO] Final Memory: 21M/649M
[INFO] -----
-----> Discovering process types
Procfile declares types -> web
```

Rysunek 10: Wdrożenie aplikacji na heroku

Heroku automatycznie rozpoznaje narzędzie budowy (Gradle, Maven) i przy jego użyciu buduje aplikacje. Po wdrożeniu można uzyskać do niej dostęp poprzez:

```
$ heroku open
```

Rysunek 11: Otworzenie aplikacji heroku

4 Front aplikacji

Wszystko, co prezentowane jest bezpośrednio użytkownikowi zrealizowane zostało z wykorzystaniem frameworku **Thymeleaf**, natomiast sam szablon aplikacji oparty jest na kompozycji **Bootstrapa**.

Czynności takie jak:

- wyszukiwanie ćwiczeń/treningów w tabelach,
- rozwijane menu,
- animacja wykresów,
- paging tabel,
- pobieranie danych wymaganych do rysowania wykresów

realizowane są z wykorzystaniem języka **JavaScript** i biblioteki **jQuery**.

```
function showExercise(e) {
    $.getJSON('/zti/charts-exercises', function (myDataDonut) {
        $("#morris-area-chart-exercises").empty();
        Morris.Area({
            element: 'morris-area-chart-exercises',
            data: myDataDonut.myData,
            xkey: 'sessionDate',
            ykeys: [myDataDonut.names[e]],
            labels: [myDataDonut.namesComplete[e]],
            pointSize: 2,
            hideHover: 'auto',
            resize: true,
            parseTime: false,
            xLabelAngle: 70
        });
    });
}
```

Rysunek 12: Przykładowa funkcja prezentująca dane na wykresie

```
function showMe(show) {
  if(show == "kcal") {
    document.getElementById("morris-area-weight").style.display = "none";
    document.getElementById("morris-area-kcal").style.display = "block";
  } else if (show == "weight") {
    document.getElementById("morris-area-weight").style.display = "block";
    document.getElementById("morris-area-kcal").style.display = "none";
  } else{
    document.getElementById("morris-area-weight").style.display = "block";
    document.getElementById("morris-area-kcal").style.display = "block";
  }
}
```

Rysunek 13: Metoda odpowiadająca za wyświetlanie różnych wykresów w zależności od wyboru użytkownika

```
$(function() {
  $(window).bind("load resize", function() {
    var topOffset = 50;
    var width = (this.window.innerWidth > 0) ? this.window.innerWidth : this.screen.width;
    if (width < 768) {
      $('div.navbar-collapse').addClass('collapse');
      topOffset = 100; // 2-row-menu
    } else {
      $('div.navbar-collapse').removeClass('collapse');
    }

    var height = ((this.window.innerHeight > 0) ? this.window.innerHeight : this.screen.height) - 1;
    height = height - topOffset;
    if (height < 1) height = 1;
    if (height > topOffset) {
      $('#page-wrapper').css("min-height", (height) + "px");
    }
  });

  var url = window.location;
  var element = $('ul.nav a').filter(function() {
    return this.href == url;
  }).addClass('active').parent();

  while (true) {
    if (element.is('li')) {
      element = element.parent().addClass('in').parent();
    } else {
      break;
    }
  }
});
```

Rysunek 14: Metoda odpowiadająca za responsywność aplikacji.

5 Aplikacja

5.1 Bezpieczeństwo

Dostęp do aplikacji możliwy jest tylko dla zarejestrowanych użytkowników. Autentykacja użytkowników odbywa się z wykorzystaniem **Spring security**.

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http.authorizeRequests()
        .antMatchers( ...antPatterns: "/register", "/logout")
            .permitAll()
            .anyRequest()
            .authenticated()
            .and()
        .formLogin()
            .loginPage("/login")
            .successForwardUrl("/zti/dashboard")
            .permitAll()
            .and()
        .logout()
            .permitAll()
            .and()
        .csrf().disable();
}
```

Rysunek 15: Konfiguracja autoryzacji

Niezarejestrowany użytkownik ma dostęp tylko do strony logowania oraz rejestracji. Inne zasoby nie są dla niego dostępne.

5.2 Logowanie i rejestracja

Panel logowania prezentuje się następująco:

The login form is titled "Logowanie" in green and "Rejestracja" in black. It contains two input fields: "Nazwa użytkownika" and "Hasło", each with a clear button (three dots). Below the fields is a blue button labeled "ZALOGUJ".

Rysunek 16: Strona logowania

Gdy użytkownik spróbuje zalogować się nieistniejącym kontem, bądź poda nieprawidłowe hasło, wyświetlony zostanie stosowny komunikat:

The login form is identical to the one in Figure 16, but with the following changes: the "Nazwa użytkownika" field contains the text "nieistniejącyuzytkownik", the "Hasło" field contains six dots, and a blue button labeled "ZALOGUJ" is present. Below the button, the error message "Niepoprawny użytkownik lub hasło" is displayed.

Rysunek 17: Błąd logowania

Użytkownicy nie posiadający jeszcze konta mają możliwość rejestracji:

LoginRegister

Nazwa użytkownika

Adres e-mail

Hasło

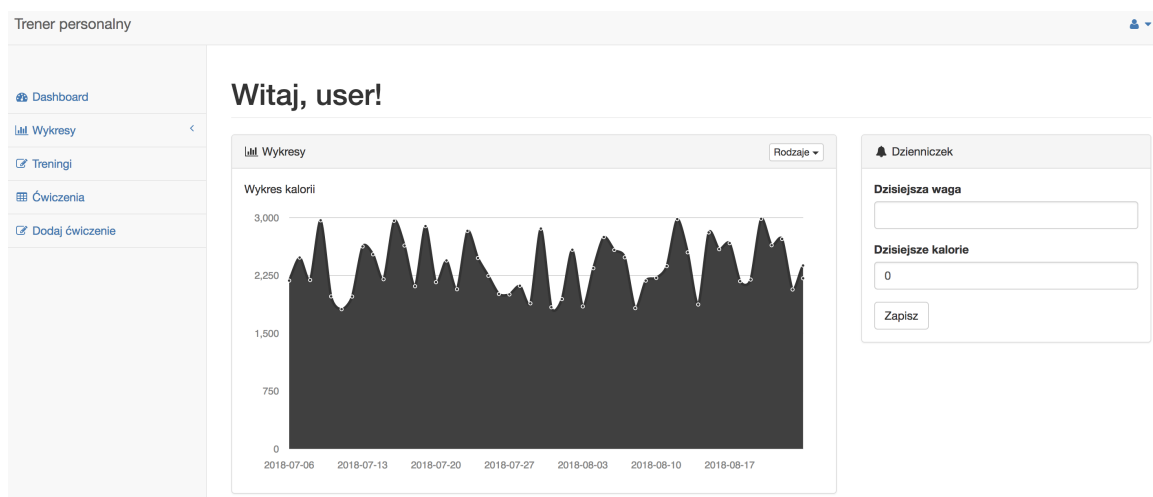
Potwierdź hasło

ZAREJESTRUJ

Rysunek 18: Strona rejestracji

Nazwa użytkownika jest unikalna. Jeśli użytkownik próbuje zarejestrować nazwę już istniejącą, zostanie wyświetlony stosowny komunikat.

Po udanym logowaniu użytkownik zostaje przekierowany na główną stronę:



Rysunek 19: Strona główna

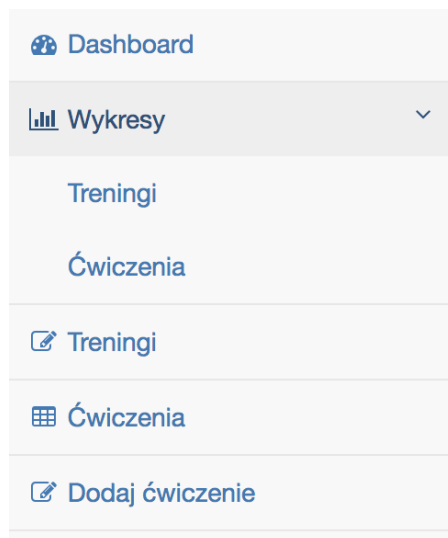
6 Podręcznik użytkownika

W ramach prezentacji udostępnione jest konto z przykładowymi danymi.
Dane konta:

nazwa użytkownika: user

hasło: user

Po zalogowaniu użytkownik zostaje przekierowany na panel główny.
Po lewej stronie dostępne jest menu:




Rysunek 20: Menu aplikacji

Klikając w kolejne pozycje menu, możemy nawigować po aplikacji.

6.1 Panel główny

Inaczej dashboard. Składa się z dwóch części:

- dziennika w którym użytkownik może wprowadzić aktualną wagę oraz spożyte danego dnia kalorie,
- obszaru z wykresami, przedstawiającymi wagę oraz spożyte kalorie na osi czasu
- dziennika w którym użytkownik może wprowadzić aktualną wagę oraz spożyte danego dnia kalorie,
- obszaru z wykresami, przedstawiającymi wagę oraz spożyte kalorie na osi czasu

 Dzienniczek

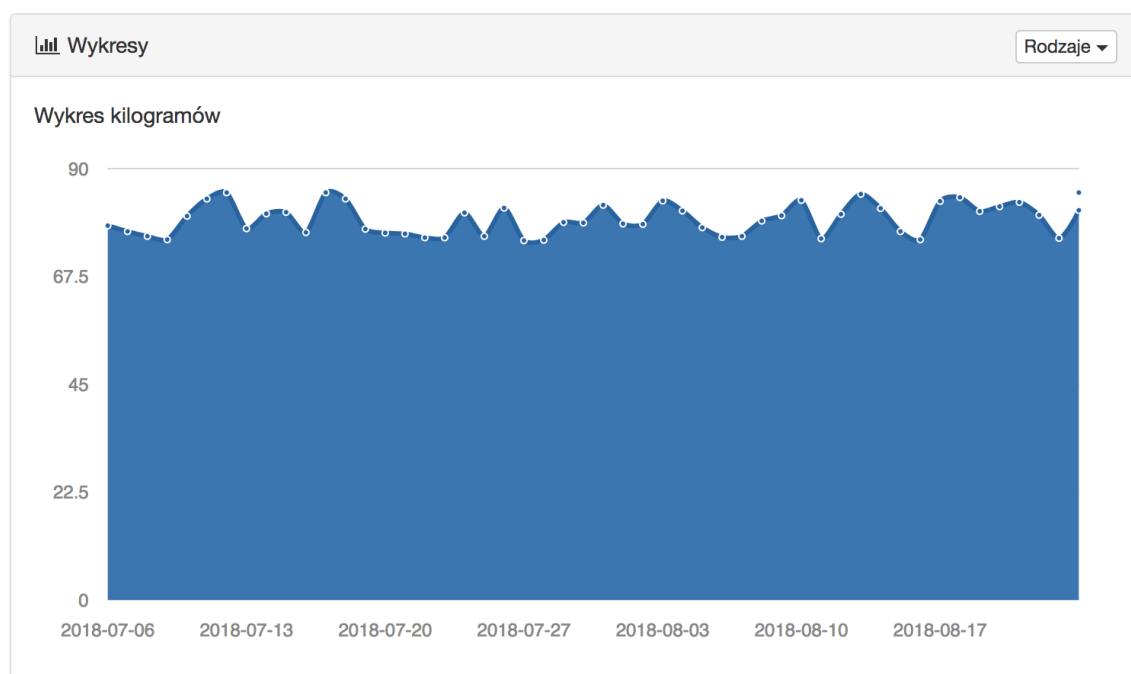
Dzisiejsza waga

Dzisiejsze kalorie

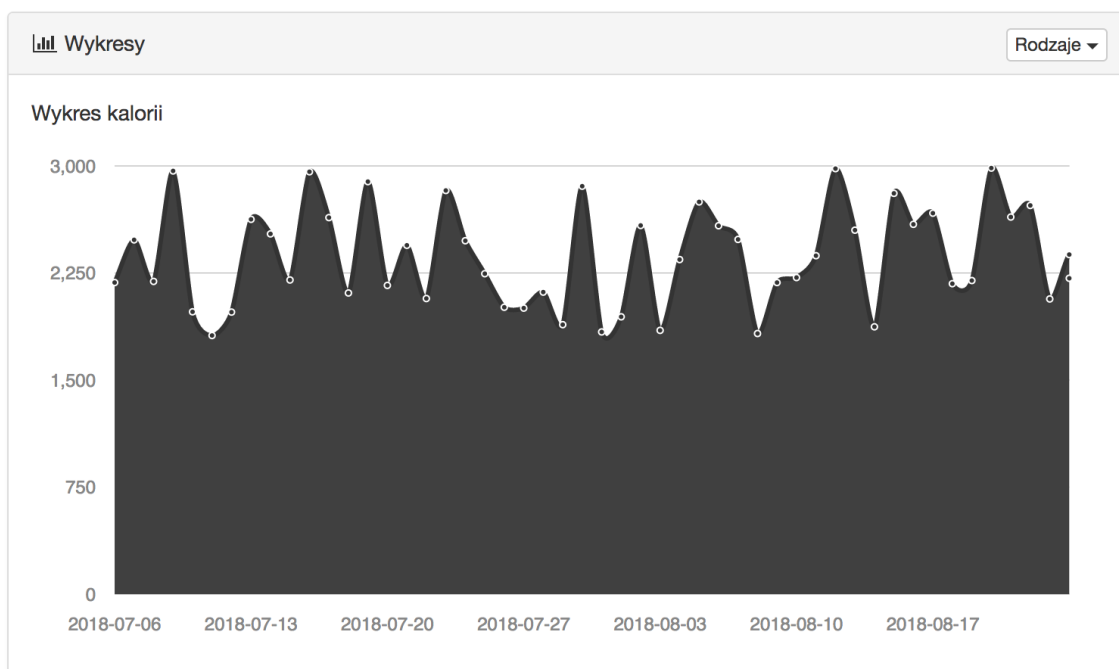
0

Zapisz

Rysunek 21: Dziennik - wprowadzanie wagi i kalorii



Rysunek 22: Wykres wagi

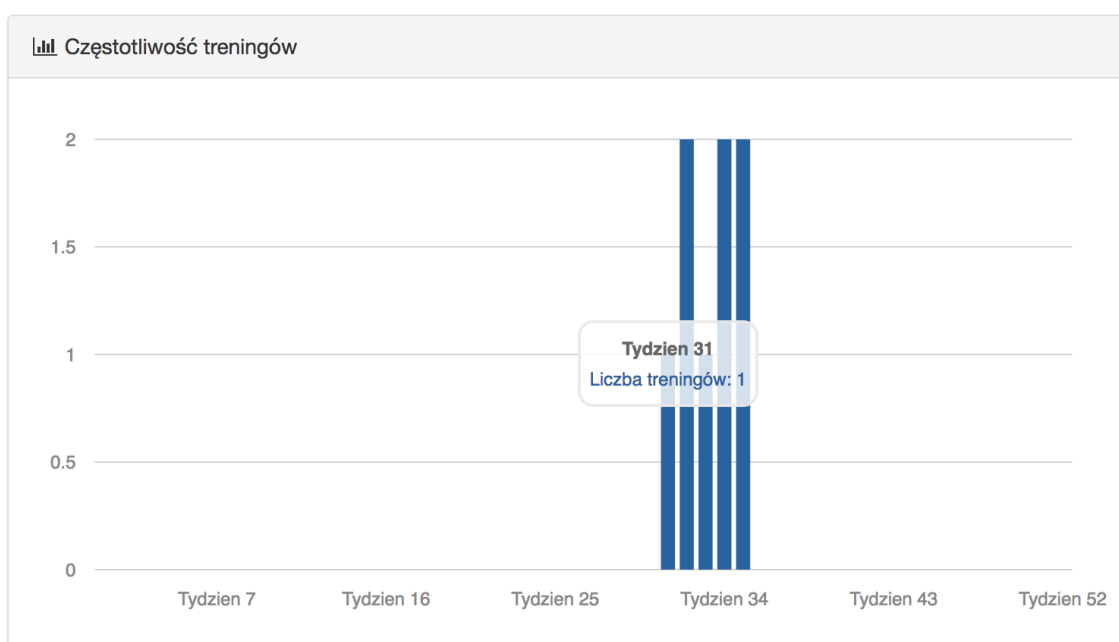


Rysunek 23: Wykres kalorii

6.2 Wykresy

6.2.1 Treningi

Zaprezentowana jest tutaj ilość wykonanych treningów w poszczególnych tygodniach.

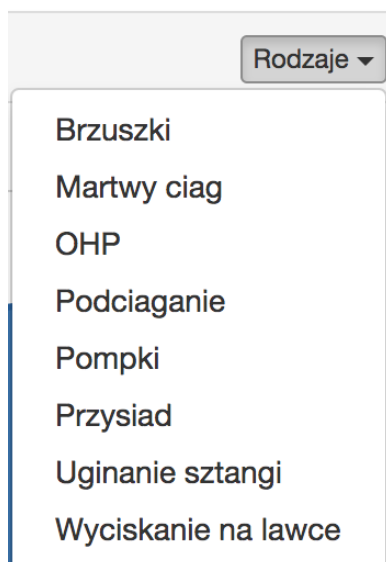


Rysunek 24: Przeprowadzone treningi

Przykładowo, w tygodniu 26 użytkownik wykonał 1 trening.

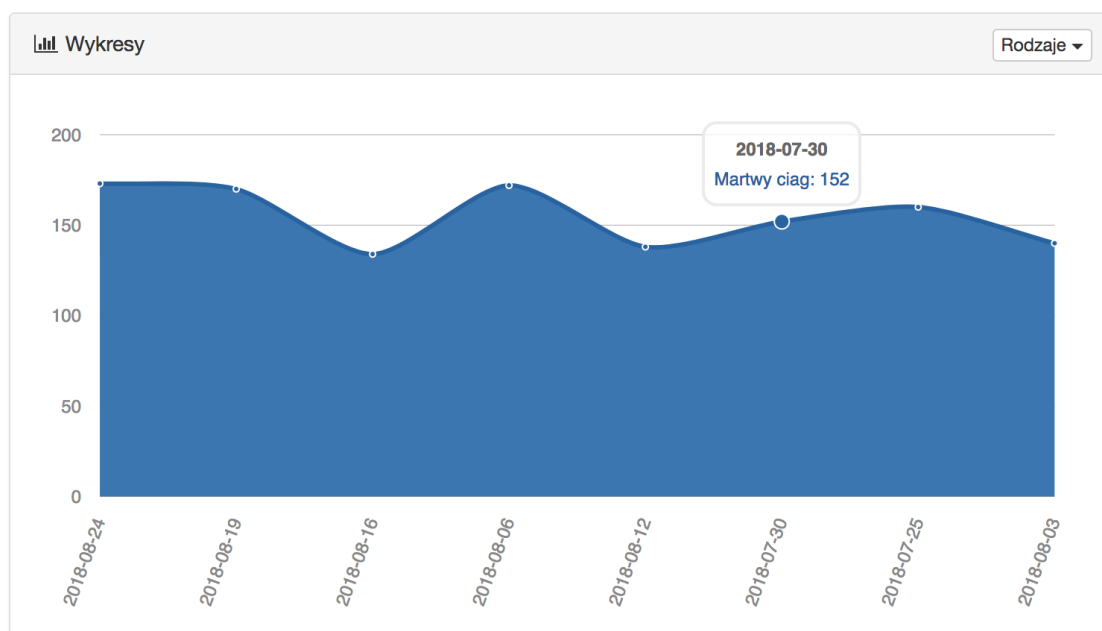
6.2.2 Ćwiczenia

Na tej podstronie można prześledzić progres obciążenia w każdym z ćwiczeń. Ćwiczenia można wybierać z rozwijanego menu w prawym górnym rogu wykresu.



Rysunek 25: Wybór ćwiczenia

Po kliknięciu na ćwiczenie zaprezentowany zostaje wykres. Na osi x prezentowane są daty kolejnych treningów, na osi y - ciężar.



Rysunek 26: Wykres progresji na osi czasu

6.3 Treningi

Na stronie wyświetlone są treningi przeprowadzone przez użytkownika, wraz ze wszystkimi szczegółami takimi jak:

- nazwy ćwiczeń,
- użyty ciężar,
- liczby powtórzeń,
- ocena trudności,
- dodatkowe uwagi

Możliwe jest wyszukiwanie treści przy pomocy pola **search**.

| Historia sesji | | | | |
|-----------------|-----|------------------------------|---|--|
| Show 10 entries | | Search: <input type="text"/> | | |
| # | ID | Data | Ćwiczenia | |
| 1 | 210 | 2018-08-24 | Przysiad ciężar: 123.0 liczba powtórzeń: 6 trudność: 123.0 uwagi: Wszystko okej Martwy ciąg ciężar: 173.0 liczba powtórzeń: 3 trudność: 173.0 uwagi: Ciezkie cwiczenie Wyciskanie na ławce ciężar: 113.0 liczba powtórzeń: 3 trudność: 113.0 uwagi: Technika do poprawy Uginanie sztangi ciężar: 21.0 liczba powtórzeń: 9 trudność: 21.0 uwagi: Nie lubię tego cwiczenia Podciąganie ciężar: 5.0 liczba powtórzeń: 5 trudność: 5.0 uwagi: Wolniejsze powtorzenia OHP ciężar: 60.0 liczba powtórzeń: 7 trudność: 60.0 uwagi: Ciezkie cwiczenie Brzuszki ciężar: 2.0 liczba powtórzeń: 29 trudność: 2.0 uwagi: Lepiej sie skupic Pompki ciężar: 22.0 liczba powtórzeń: 17 trudność: 22.0 uwagi: Raczej jako uzupelnienie | |

Rysunek 27: Lista treningów

Dodawanie nowej sesji treningowej

By dodać nowy trening, należy dodać kolejno wykonane ćwiczenia klikając przycisk **Dodaj ćwiczenie**. W celu usunięcia któregoś ćwiczenia należy kliknąć przycisk **Usuń ćwiczenie**.

Gdy dodano już wszystkie ćwiczenia, należy kliknąć przycisk **Dodaj trening**, by dodać nowy trening.

| Dodaj nowy trening | | | | | |
|---|--|----------------------|--------------------------------|--------------------------------|---|
| Data treningu (YYYY-MM-DD) | | | | | |
| <input type="text" value="2018-08-25"/> | | | | | |
| Ćwiczenia | | | | | |
| # | Nazwa ćwiczenia | Ciężar | L. powtórzeń | RPE | Uwagi |
| 1 | <input type="text" value="Martwy ciąg"/> | <input type="text"/> | <input type="text" value="0"/> | <input type="text" value="0"/> | <input type="text"/> |
| | | | | | <input type="button" value="Usuń ćwiczenie"/> |
| <input type="button" value="Dodaj trening"/> <input type="button" value="Dodaj ćwiczenie"/> | | | | | |

Rysunek 28: Dodanie nowego treningu

6.4 Ćwiczenia

Na stronie prezentowana jest lista ćwiczeń wraz z ich opisami.

Możliwe jest przeszukanie listy. W tym celu należy wpisać poszukiwaną frazę w pole **search**.

Lista ćwiczeń

Show 10 entries Search:

| ID | Nazwa | Opis |
|-----|---------------------|--------------------------------------|
| 138 | Przysiad | Przysiad ze sztanga |
| 139 | Martwy ciąg | Podnoszenie ciężaru z ziemi |
| 140 | Wyciskanie na ławce | Wyciskanie ciężaru na ławce płaskiej |
| 141 | Uginanie sztangi | Uginanie sztangi na biceps |
| 142 | Podciąganie | Podciąganie na drążku |
| 143 | OHP | Wyciskanie ciężaru nad głowę |
| 144 | Brzuszki | Cwiczenie angażujące mięśnie brzucha |
| 145 | Pompki | Podstawowe ćwiczenie ogólnorozwojowe |

Showing 1 to 8 of 8 entries

Previous 1 Next

Dodaj ćwiczenie

Rysunek 29: Lista ćwiczeń

Z tej pozycji możliwe jest również dodanie nowego ćwiczenia. By to zrobić, należy kliknąć przycisk **Dodaj ćwiczenie**, znajdujący się na dole strony, pod tabelą.

6.5 Dodaj ćwiczenie

Strona umożliwia dodanie nowego ćwiczenia.

Nazwa (unikalna)

Krótki opis ćwiczenia

Dodaj

Rysunek 30: Lista ćwiczeń

W tym celu należy wypełnić oba pola i nacisnąć przycisk **Dodaj**.