

POLITECHNIKA WROCŁAWSKA  
WYDZIAŁ ELEKTRONIKI

---

KIERUNEK: INFORMATYKA (INF)

SPECJALNOŚĆ: INŻYNIERIA SYSTEMÓW INFORMATYCZNYCH (INS)

PRACA DYPLOMOWA  
INŻYNIERSKA

Aplikacja webowa wspomagająca układanie planów  
treningowych fitness z użyciem technologii rodziny  
Java

Web application supporting creation of fitness plans  
using Java technologies

AUTOR:

Michał Honc

PROWADZĄCY PRACĘ:

dr inż. Roman Ptak, Katedra Informatyki Technicznej  
(K30W04D03)

---

WROCŁAW, 2020

# Spis treści

Spis rysunków .....	3
Spis listingów .....	5
1. Cel i zakres pracy .....	6
2. Wprowadzenie.....	7
2.1. Tematyka treningów fitness.....	7
2.2. Porównanie dostępnych rozwiązań .....	8
3. Specyfikacja aplikacji .....	10
3.1. Wymagania funkcjonalne .....	10
3.2. Wymagania niefunkcjonalne .....	11
4. Architektura aplikacji .....	13
4.1. Architektura aplikacji internetowej .....	13
4.2. Diagram przypadków użycia .....	14
4.3. Diagram związków encji .....	16
5. Opis środowiska implementacyjnego.....	17
6. Implementacja .....	20
6.1. Autoryzacja i autentykacja .....	20
6.2. Struktura danych.....	23
6.3. Interfejs użytkownika. ....	26
7. Prezentacja aplikacji.....	29
8. Testy aplikacji .....	40
9. Podsumowanie i wnioski.....	43
Literatura .....	44
Dodatek – opis załączonej płyty CD .....	45

# Spis rysunków

Rys. 1. Mapa z oznaczonymi siłowniami we Wrocławiu.....	8
Rys. 2. Aplikacja kulturystyka.pl/atlas.....	9
Rys. 3. Aplikacja Atlas ćwiczeń KFD.PL.....	9
Rys. 4. Struktura aplikacji internetowej.....	14
Rys. 5. Diagram przypadków użycia.....	15
Rys. 6. Diagram ERD.....	16
Rys. 7. Generowanie projektu Spring zawierającego wybrane biblioteki.....	19
Rys. 8. Formularz logowania.....	21
Rys. 9. Zakodowane hasła w bazie danych.....	23
Rys. 10. Rozwijane przyciskiem menu aplikacji.....	28
Rys. 11. Widok logowania.....	29
Rys. 12. Widok rejestracji.....	30
Rys. 13. Widok strony domyślnej z poziomu przeglądarki z rozwiniętym menu.....	30
Rys. 14. Mobilny widok strony domyślnej.....	31
Rys. 15. Widok rozwiniętego menu.....	32
Rys. 16. Widok listy ćwiczeń w danej kategorii.....	32
Rys. 17. Opis danego ćwiczenia.....	33
Rys. 18. Film instruktorzowy danego ćwiczenia.....	34
Rys. 19. Możliwość dodania ćwiczenia do jednego z własnych planów.....	35
Rys. 20. Widok listy planów użytkownika.....	36
Rys. 21. Widok harmonogramu tygodniowego jednego z naszych planów.....	36
Rys. 22. Dodawanie nowego indywidualnego planu.....	37
Rys. 23. Lista przykładowych planów.....	38

Rys. 24. Menu administratora.....	38
Rys. 25. Widok dodawania ćwiczenia wraz z przykładowym linkiem.....	39
Rys. 26. Zaliczone testy jednostkowe użytkownika.....	40
Rys. 27. Zaliczone testy jednostkowe ćwiczeń.....	41
Rys. 28. Zaliczony test automatyczny logowania.....	41
Rys. 29. Wyniki testowania Selenium.....	42

# Spis listingów

Listing 1. Implementacja filtra.....	20
Listing 2. Serwis logowania.....	21
Listing 3. Logowanie po stronie frontendu.....	22
Listing 4. Kontroler logowania.....	22
Listing 5. Kodowanie hasła.....	23
Listing 6. Zawartość pliku konfiguracyjnego application.properties.....	23
Listing 7. Encja użytkownika.....	24
Listing 8. Repozytorium odpowiedzialne za operacje CRUD.....	25
Listing 9. Kontroler rejestracji.....	26
Listing 10. Część widoku odpowiedzialnego za pobranie danych od użytkownika.....	27
Listing 11. Przesłanie widoku wraz z listą kategorii.....	27
Listing 12. Iteracja po liście kategorii.....	27
Listing 13. Zaimportowanie pliku .css.....	28
Listing 14. Użycie bootstrapowych klas do stworzenia responsywnego menu.....	28
Listing 15. Test jednostkowy sprawdzający poprawność pobrania użytkownika z serwisu....	40
Listing 16. Test jednostkowy sprawdzający poprawność zwrócenia danych ćwiczenia.....	41
Listing 17. Test automatyczny logowania.....	41

# 1. Cel i zakres pracy

Celem pracy inżynierskiej jest stworzenie aplikacji webowej posiadającej bazę ćwiczeń pogrupowaną według partii mięśniowych wraz z krótkim opisem danego ćwiczenia, jego wpływu na organizm oraz opisu poprawnego wykonania (wraz z dołączonym filmem poglądowym). Aplikacja zawiera również przykładowe plany treningowe udostępniane przez administratora (np. trening siłowy, wytrzymałościowy) wraz z opisem dla kogo jest taki trening, jaki jest jego cel, jaka dieta powinna z nim współgrać itp. Aplikacja umożliwia także opcję stworzenia własnego, indywidualnego planu treningowego, a także udostępnienie go innym użytkownikom.

Program przeznaczony jest dla każdego, choć jego głównym odbiorcą będą osoby, które na co dzień zajmują się tworzeniem planów ćwiczeniowych. Potencjalnym odbiorcą mogą być siłownie, oraz każdy z trenerów personalnych czy dietetyków.

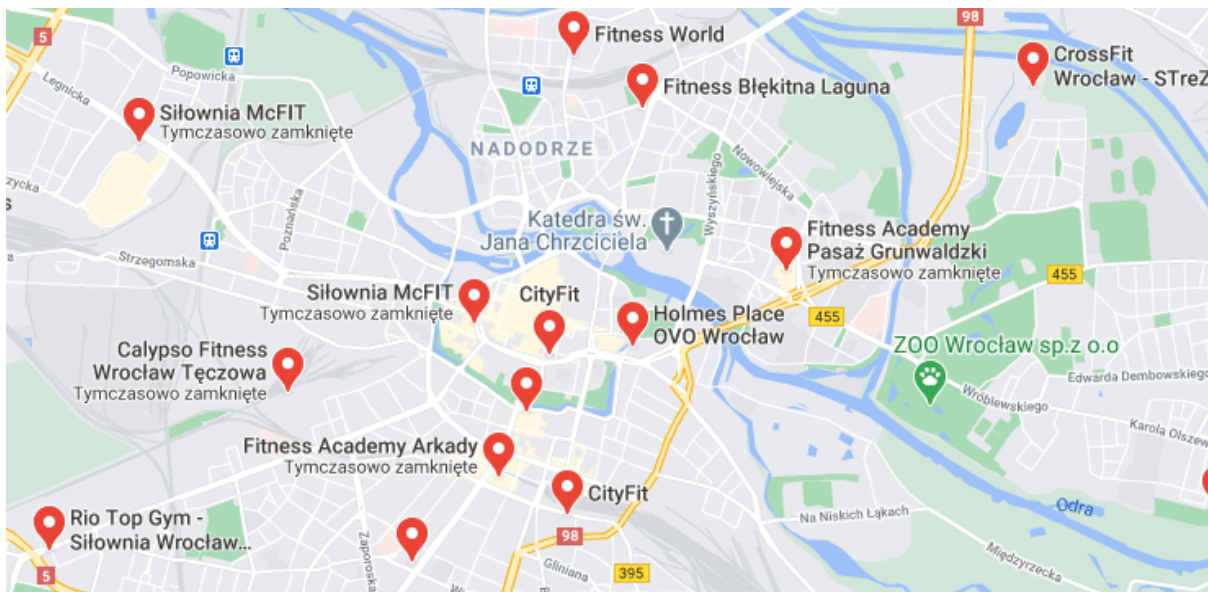
## **2. Wprowadzenie**

### **2.1. Tematyka treningów fitness**

W ostatnich latach bardzo modne stały się zdrowy styl życia, wchodząca w jego skład dieta, oraz usprawnianie kondycji fizycznej. Ludzie coraz częściej zaczynają swoją przygodę z wszelakimi treningami, aby poprawić samopoczucie, muskulaturę, wydolność oraz jakość życia. Trening sportowy to proces poddawania organizmu stopniowo wzrastającym obciążeniom, w wyniku których następuje adaptacja oraz zwiększenie poziomu poszczególnych cech motorycznych. Definicja treningu zawiera w sobie również naukę nawyków ruchowych powiązanych z konkretną dyscypliną sportu. Stosując odpowiedni trening w parze z właściwym odżywianiem można także kształtować niektóre cechy morfologiczne, na przykład zwiększyć poziom masy mięśniowej lub zredukować poziom tkanki tłuszczowej [16].

W związku z rosnącą modą na tego typu treningi rośnie również zapotrzebowanie na aplikacje wspomagające układanie planów treningowych, przeglądanie atlasów ćwiczeń, a także wyszukiwanie wiedzy na ten temat.

Na poniższym rysunku można zaobserwować, jak prężnie rozwija się branża siłowni we Wrocławiu.



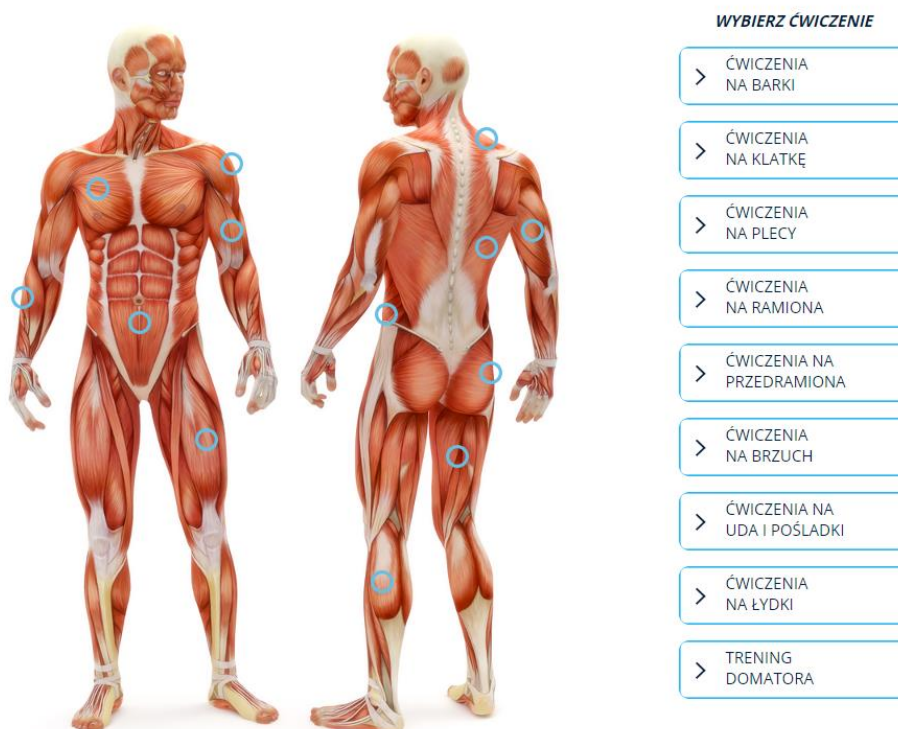
Rys. 1. Mapa z oznaczonymi siłowniami we Wrocławiu

## 2.2. Porównanie dostępnych rozwiązań

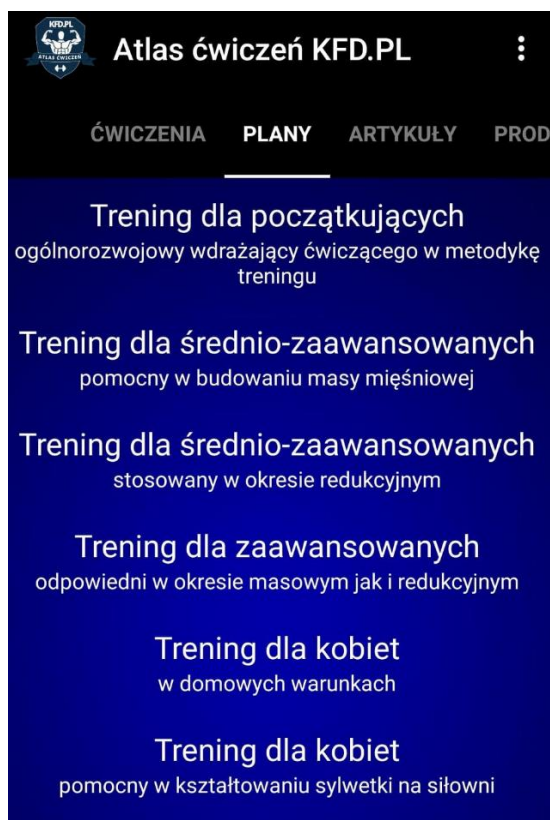
Na rynku pojawiło się sporo aplikacji związanych z tematyką fitness. Niektóre z nich zawierają bogaty atlas ćwiczeń, niektóre oferują możliwość przeglądania planów treningowych, jeszcze inne zostały zaprojektowane do układania planów. Jednak przeglądając gamę dostępnych rozwiązań, rzadko która aplikacja zawiera wszystkie wymienione funkcjonalności. Omawiana aplikacja będąca tematem tej pracy, została stworzona w taki sposób aby łączyła w sobie wyżej wymienione funkcjonalności.

Poniżej możemy zobaczyć dwie najbardziej popularne aplikacje w polskiej wersji językowej, kulturystyka.pl/atlas [5] oraz Atlas ćwiczeń KFD.PL [4]. Obie aplikacje umożliwiają przeglądanie atlasu ćwiczeń, lecz nie posiadają funkcjonalności ułożenia własnego planu treningowego, a także nie posiadają harmonogramu tygodniowego przykładowych planów treningowych, co jest istotną kwestią szczególnie dla początkujących.





Rys. 2. Aplikacja kulturystyka.pl/atlas



Rys. 3. Aplikacja Atlas ćwiczeń KFD.PL

# 3. Specyfikacja aplikacji

## 3.1. Wymagania funkcjonalne

### 1. Rejestrowanie nowego użytkownika

Osoby nie posiadające konta mają możliwość rejestracji.

### 2. Logowanie i wylogowanie

Każdy zarejestrowany użytkownik ma możliwość logowania i wylogowania z aplikacji. Żadne dane aplikacji nie są publicznie dostępne.

### 3. Przeglądanie listy dostępnych kategorii

Domyślnie strona startowa przedstawia zalogowanym użytkownikom podział na kategorie, przykładowo mogą to być ćwiczenia na plecy, brzuch.

### 4. Przeglądanie listy dostępnych ćwiczeń

Każdy użytkownik ma możliwość przeglądania listy ćwiczeń dostępnych w systemie.

### 5. Podgląd ćwiczeń

Każdy użytkownik ma możliwość przeglądania szczegółów dotyczących wybranego ćwiczenia, gdzie zawarte są dokładne informacje o sposobie wykonania oraz materiał multimedialny w postaci video.

### 6. Edycja ćwiczeń

Każdy administrator ma możliwość pełnej edycji ćwiczeń.

### 7. Usuwanie ćwiczeń

Każdy administrator ma możliwość usunięcia wskazanego ćwiczenia.

### 8. Dodanie ćwiczenia do planu tygodniowego

Każdy użytkownik ma możliwość dodania ćwiczenia do swojego prywatnego planu tygodniowego.

### 9. Dodanie nowego planu

Każdy użytkownik ma możliwość dodania nowego planu tygodniowego.

### 10. Przeglądanie harmonogramu planu treningowego

Każdy użytkownik ma możliwość podglądu następnych 7 dni oraz zaplanowanych ćwiczeń. Dodatkowo każdy użytkownik może przejść do szczegółów na temat wykonania ćwiczenia.

#### **11. Edycja planu tygodniowego**

Każdy użytkownik ma możliwość edycji własnego planu tygodniowego.

#### **12. Usunięcie planu tygodniowego**

Każdy użytkownik ma możliwość usunięcia własnego planu tygodniowego.

#### **13. Udostępnienie planu tygodniowego**

Każdy użytkownik ma możliwość udostępnienia własnego planu tygodniowego innym użytkownikom.

### **3.2. Wymagania niefunkcjonalne**

#### **1. Platforma**

Aplikacja powinna być wieloplatformowa, czyli ściśle niezwiązana z wyłącznie jednym systemem operacyjnym. W przypadku stworzonej aplikacji jest to aplikacja webowa która automatycznie spełnia ten warunek.

#### **2. Bezpieczeństwo**

Dane przechowywane w systemie mogą być dostępne jedynie dla zalogowanych użytkowników. Dane zarejestrowanych użytkowników nie są dla nikogo widoczne.

#### **3. Użyteczność**

Aplikacja powinna być intuicyjna i łatwa w nawigowaniu.

#### **4. Przenoszalność**

Aplikacja powinna być w stanie w prosty sposób przenieść swoje dane i pełną konfigurację na dowolnie wybrany serwer.

## **5. Niezawodność**

Aplikacja powinna być wolna od błędów oraz przejść na każdym etapie przez wszystkie fazy testowania oprogramowania.

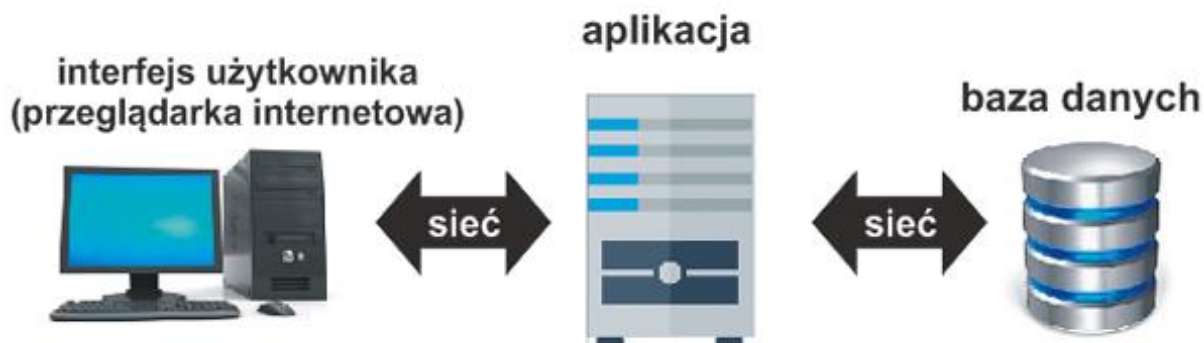
## 4. Architektura aplikacji

### 4.1. Architektura aplikacji internetowej

Wybrana architektura obrazuje aplikację internetową. Jest to program, który działa na serwerze oraz komunikuje się z odbiorcą poprzez sieć komputerową wykorzystując przeglądarkę internetową. Przeglądarka w takim przypadku sprawuje funkcję interfejsu użytkownika. W tego typu programie zakłada się interakcję z użytkownikiem, wykorzystanie baz danych oraz innych usług. Ważnymi cechami tego typu aplikacji jest łatwość i szybkość dostępu do informacji (dowolny komputer, który jest podłączony do Internetu), oraz bezpieczeństwo danych (autoryzacja i autentykacja, szyfrowanie połączenia oraz system uprawnień) [3].

Zalety wybranego rozwiązania:

- dostępność dla wszystkich zainteresowanych osób posiadających dostęp do Internetu,
- brak konieczności posiadania dodatkowego oprogramowania, korzystanie z aplikacji internetowej wymaga tylko przeglądarki internetowej (oraz sprzętu, np. komputera PC lub telefonu z systemem Android),
- łatwość utrzymania oraz usprawniania, aktualizacje oraz zmiany aplikacji webowej są wykonywane na serwerze, bez udziału użytkowników,
- łatwość integracji aplikacji internetowej z innymi usługami, na przykład stroną internetową,
- niskie koszty wytworzenia, uruchomienia oraz utrzymania aplikacji internetowej w zestawieniu z innymi rozwiązaniami, spora część technologii stosowanych w wytwarzaniu aplikacji webowych jest darmowa.



Rys. 4. Struktura aplikacji internetowej [3]

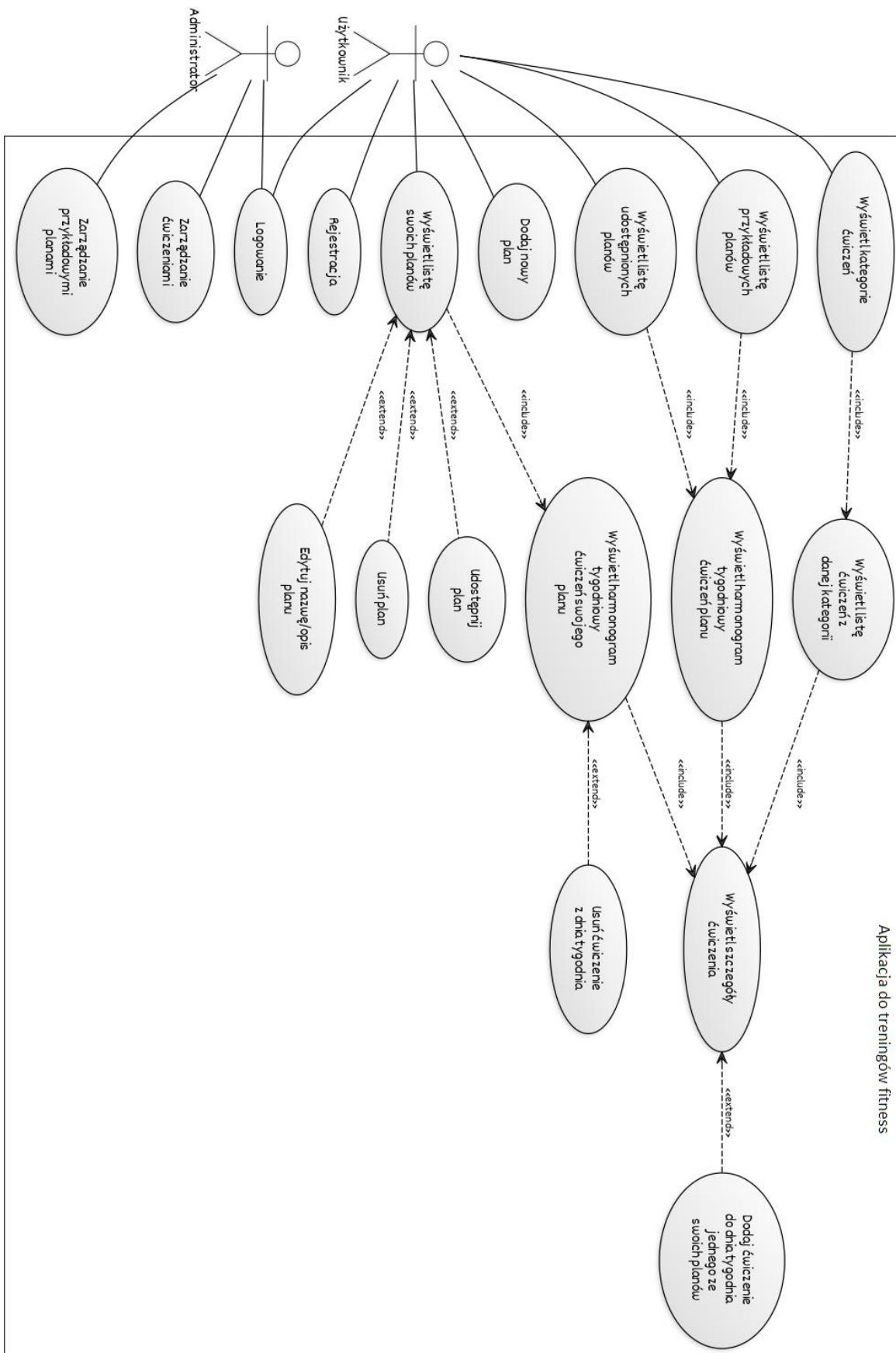
Aplikacja wykorzystuje wzorzec projektowy MVC (ang. *Model-View-Controller*), którego założeniem jest podzielenie aplikacji na trzy części:

- Model – reprezentuje problem lub logikę aplikacji,
- Widok – opisuje wyświetlanie modelu w ramach interfejsu użytkownika,
- Kontroler – przyjmuje dane wejściowe od użytkownika i reaguje na jego działania. Zarządza aktualizowaniem modelu i odświeżaniem widoków.

Wszystkie trzy części są ze sobą wzajemnie połączone [13].

## 4.2. Diagram przypadków użycia

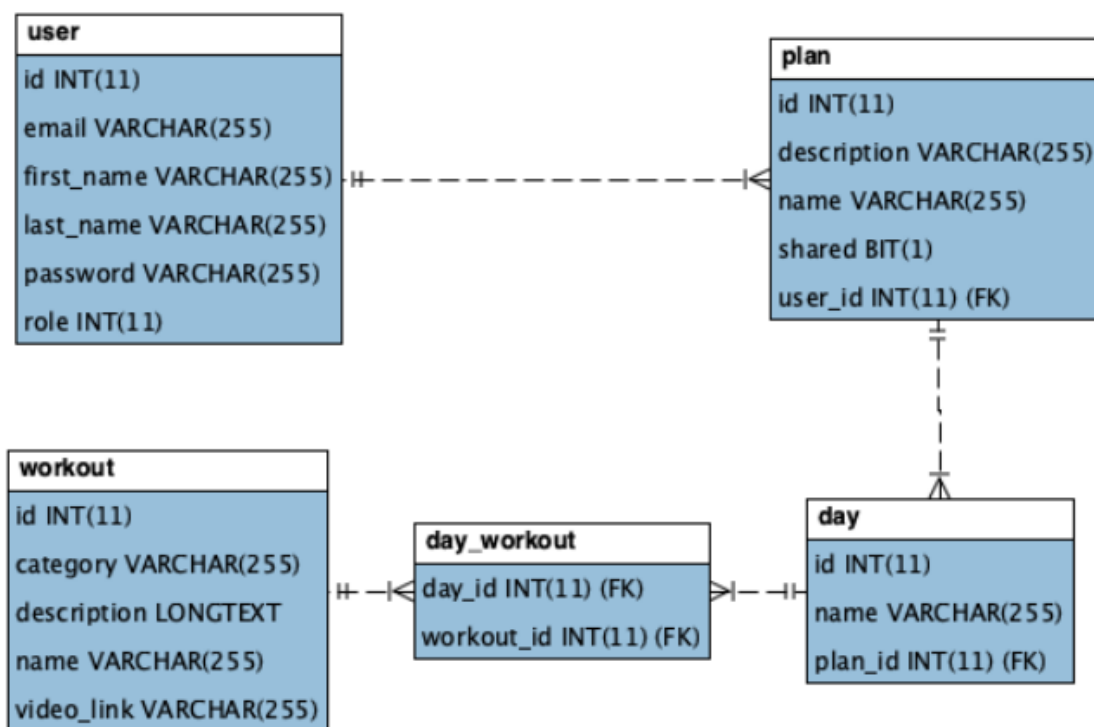
Poniższy diagram UML (ang. *Unified Modeling Language*) przedstawia odwzorowanie wymagań funkcjonalnych systemu. Na diagramie przedstawiono szczegółowo przypadki użytkownika, natomiast przypadki administratora są bardziej ogólne, żeby rysunek był jak najbardziej czytelny.



Rys. 5. Diagram przypadków użycia

### 4.3. Diagram związków encji

Diagram ERD (*Entity-Relationship Diagram*) jest graficznym przedstawieniem związków pomiędzy encjami. Demonstruje on konceptualne modele danych użyte w systemie. Etap modelowania diagramu jest niezwykle ważny, ponieważ na jego podstawie zwykle tworzy się encje w kodzie.



Rys. 6. Diagram ERD

Kod programu opiera się o powyższy diagram związków encji. Przedstawiono na nim następujące relacje:

1. Jeden do wiele – każdy użytkownik może mieć wiele planów oraz każdy plan może mieć wiele dni.
2. Wiele do wiele – dzień może mieć wiele ćwiczeń, poszczególne ćwiczenie może odnosić się do wielu dni. Relacja ta została ukazana na diagramie za pomocą łącznika.



## 5. Opis środowiska implementacyjnego

Wiodącym językiem w projekcie jest Java. Jest to język obiektowy, wymagający wirtualnej maszyny. W projekcie została użyta wersja dostarczona przez firmę Oracle. Bardzo istotnym elementem są pojęcia JRE i JDK.

JRE (ang. *Java Runtime Environment*) – jest to maszyna wirtualna nie zawierająca dodatkowych narzędzi koniecznych dla programisty. Zawiera tylko implementację wirtualnej maszyny.

JDK (ang. *Java Development Kit*) – jest to zestaw narzędzi dla programisty. Narzędzia te są konieczne do wytwarzania programów w języku Java. Dodatkowym elementem zawartym w pakiecie jest javac, czyli kompilator języka [9].

Proces instalacji zależy od posiadania systemu operacyjnego, jednak ogólna instrukcja mówi, że ściągnięcie oraz instalacja odpowiedniej wersji JDK jest wystarczające (np. ze strony Oracle).

Następnym krokiem jest środowisko w którym będzie tworzony kod. Do implementacji aplikacji użyto środowiska IntelliJ IDEA [15]. Jest to jedno z najpopularniejszych narzędzi do budowania nowoczesnych rozwiązań programistycznych, które bazuje na licencji Apache. W projekcie została wykorzystana wersja darmowa.

Projekt u swoich podstaw bazuje na Mavencie. Jest to narzędzie wykorzystywane do budowania projektów. Jego innym znanym odpowiednikiem może być np. Gradle lub Ant. Maven pozwala na zautomatyzowanie całego procesu budowania aplikacji poprzez użycie języka domenowego DSL. Dodatkowo poza samym budowaniem projektu, zajmuje się on również kompilowaniem, tworzeniem plików jar, testowaniem czy zarządzaniem zależnościami [10].

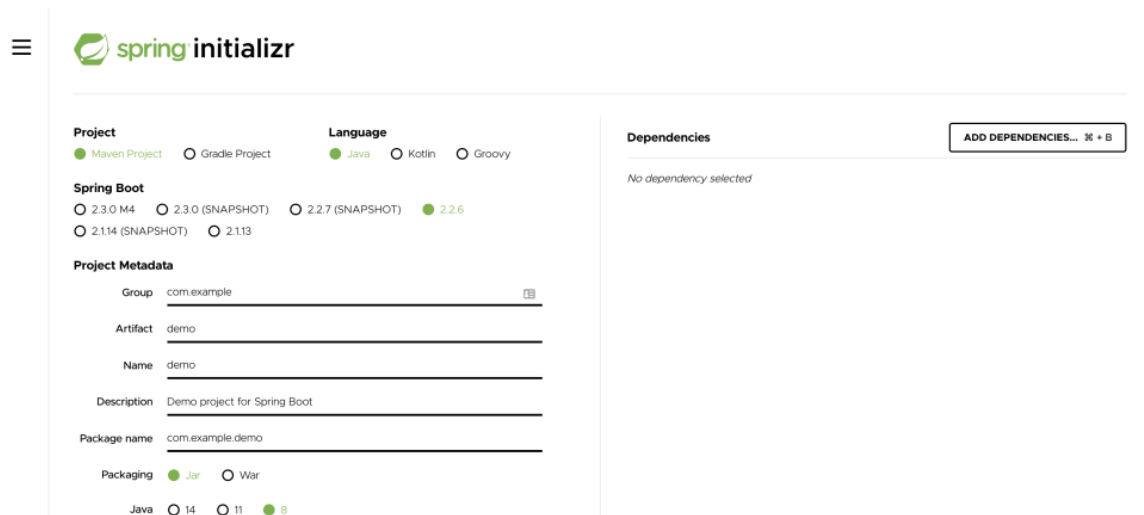
Następnym elementem jest baza danych. W tym przypadku jest to rozwijany przez firmę Oracle MySQL. Jest to darmowy system zarządzania relacyjną

bazą danych. Jej największymi zaletami jest skalowalność, uniwersalność oraz ochrona danych. Można ją w bardzo łatwy oraz szybki sposób włączyć do aplikacji webowej [6].

Kolejnym bardzo ważnym elementem jest *framework* o nazwie Spring. W skrócie Spring jest zestawem bibliotek umożliwiającym kompleksowe tworzenie aplikacji webowych w Javie. Ponieważ jest to dosyć obszerny *framework*, nie było wymagane użycie wszystkich jego modułów i funkcjonalności. Do wykonania aplikacji zostały użyte następujące moduły [8]:

1. Spring core – zawierający podstawową funkcjonalność Springa – *beans*, komponenty, *dependency injection* i wiele innych.
2. Spring boot – biblioteka znacznie ułatwiająca uruchomienie nowego projektu. Rozwiązała dwa problemy obecne w pierwszych wersjach Springa:
  - a. Serwer – Spring Boot zawiera *embedded server*, czyli automatycznie wbudowany serwer dostępny bez instalacji, ściągany jako zależność mavenowa. Domyślnie jest to Tomcat.
  - b. Konfiguracja – konfiguracja aplikacji za pomocą adnotacji, prekonfigurowane „startery” do każdej części Springa. Dzięki temu aplikacja podczas pierwszego uruchomienia nie wymaga czasochłonnej konfiguracji każdego jej elementu.
3. Spring data – wspomaga proces i dostarcza narzędzia potrzebne do wymiany danych między aplikacją oraz bazą danych.
4. Spring security – pozwala zabezpieczyć swoją aplikację i wprowadzić autoryzację i autentykację.
5. Spring Web – bazujący na technologii serwletów, kluczowej specyfikacji Javy EE. Do pracy wymaga kontenera serwletów, może to być Tomcat lub Jetty.
6. Spring Thymeleaf – zestaw narzędzi ułatwiających komunikację między backendem a frontendem.

Najprostszym sposobem na utworzenie aplikacji opartej na Springu jest użycie generatora projektów Spring Boot, jednym z nich jest initializr.



The screenshot shows the Spring Initializr web interface. It features a sidebar with a hamburger menu icon and the 'spring initializr' logo. The main content area is divided into three sections: 'Project', 'Language', and 'Dependencies'. The 'Project' section has radio buttons for 'Maven Project' (selected) and 'Gradle Project'. The 'Language' section has radio buttons for 'Java' (selected), 'Kotlin', and 'Groovy'. The 'Spring Boot' section has radio buttons for versions: '2.3.0 M4', '2.3.0 (SNAPSHOT)', '2.2.7 (SNAPSHOT)', '2.2.6' (selected), '2.1.14 (SNAPSHOT)', and '2.1.13'. The 'Project Metadata' section contains input fields for 'Group' (com.example), 'Artifact' (demo), 'Name' (demo), 'Description' (Demo project for Spring Boot), and 'Package name' (com.example.demo). The 'Packaging' section has radio buttons for 'Jar' (selected) and 'War'. The 'Java' section has radio buttons for versions '14', '11', and '8' (selected). The 'Dependencies' section has a button 'ADD DEPENDENCIES... 0 + 8' and the text 'No dependency selected'.

Rys. 7. Generowanie projektu Spring zawierającego wybrane biblioteki

Jeśli chodzi o *frontend*, aplikacja wykorzystuje *framework* Bootstrap, który zawiera zestaw narzędzi ułatwiający tworzenie interfejsu graficznego. Główną bazą bootstrapa są gotowe rozwiązania HTML oraz CSS. Wykorzystuje także JavaScript. Interfejs użytkownika opiera się o jeden z gotowych szablonów tej technologii [7]. Bootstrap jest oparty na licencji MIT [12], co oznacza nieograniczone prawo do użytku, kopiowania, modyfikacji i rozpowszechniania (również sprzedaży) programu.

# 6. Implementacja

## 6.1. Autoryzacja i autentykacja

Do autoryzacji i autentykacji aplikacja wykorzystuje Spring Security. Baza danych przechowuje dane użytkowników. W aplikacji wyróżnionych jest kilka ról. Dostęp do poszczególnych widoków posiadają użytkownicy z określonymi rolami. Przed nadaniem dostępu do każdej strony został zdefiniowany filtr, przez który przechodzi cały ruch zanim serwer zwróci odpowiedź. Poniższa konfiguracja definiuje dostęp do wszystkich plików graficznych oraz skryptów jednak w przypadku braku wygenerowanej sesji przekieruje każdego użytkownika na stronę logowania.

Listing. 1. Implementacja filtra

```
@Override
public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse,
                    FilterChain filterChain) throws IOException, ServletException {
    HttpServletRequest req = (HttpServletRequest) servletRequest;
    HttpServletResponse resp = (HttpServletResponse) servletResponse;
    User userSession = (User) req.getSession().getAttribute("user");
    String requestUri = req.getRequestURI();

    if (
        (userSession != null) ||
        requestUri.startsWith("/resources/") ||
        requestUri.endsWith(".css") ||
        requestUri.endsWith(".png") ||
        requestUri.endsWith(".jss") ||
        requestUri.endsWith(".jpg") ||
        requestUri.endsWith("register") ||
        requestUri.endsWith("login")
    ) {
        filterChain.doFilter(req, resp);
    } else {
        resp.sendRedirect(resp.encodeRedirectURL("login"));
    }
}
```

Cały proces logowania można opisać w krokach:

1. Podanie danych do logowania i wysłanie zapytanie o typie POST z dwoma parametrami „email” oraz „password”.
2. Kontroler przyjmuje parametry a następnie sprawdza czy użytkownik o podanych danych istnieje.
3. Jeśli istnieje, to następuje nadanie sesji i przekierowanie do strony głównej aplikacji, jeśli nie, użytkownik jest przekierowywany na stronę logowania.

Listing. 2. Serwis logowania

```
public boolean login(String email, String password, HttpServletRequest req,
Model model){
    Optional<User> user = userRepository.findByEmail(email);
    if (user.isPresent() && user.get().getPassword().equals(DigestU-
tils.md5Hex(password).toUpperCase())) {
        req.getSession().setAttribute("user", user.get());
        return true;
    }
    model.addAttribute("messageType", "danger");
    model.addAttribute("message", "Błąd, dane logowania niepoprawne");
    return false;
}
```

The image shows a login form with a light gray header containing the title "Zaloguj". Below the header, there are two input fields. The first is labeled "Email" and contains the placeholder text "Email". The second is labeled "Password" and contains the placeholder text "Hasło". Both fields have a small icon on the right side. Below the input fields is a dark blue button with the text "Zaloguj". At the bottom of the form, there is a link that says "Potrzebujesz konta? Zarejestruj się!" in a teal color.

Rys. 8. Formularz logowania

Listing. 3. Logowanie po stronie frontendu

```
<form id="loginForm" method="post" action="login">
  <div class="form-group">
    <label class="small mb-1" for="inputEmailAddress">Email</label>
    <input class="form-control py-4" name="email" id="inputEmailAd-
dress"
        type="email" placeholder="Email"/>
  </div>
  <div class="form-group">
    <label class="small mb-1" for="inputPassword">Password</label>
    <input class="form-control py-4" name="password" id="inputPassword"
        type="password" placeholder="Hasło"/>
  </div>
  <div class="text-center">
    <input class="btn btn-primary" type="submit" value="Zaloguj">
  </div>

  <div th:if="{message} != null" th:class="'my-4 alert alert-' + {mes-
sageType}"
      th:text="{message}" role="alert">
  </div>
</form>
```

Listing. 4. Kontroler logowania

```
@Controller
public class LoginViewController {

    @Autowired
    private LoginService loginService;

    @GetMapping("login")
    public ModelAndView loginView() {
        return new ModelAndView("login");
    }

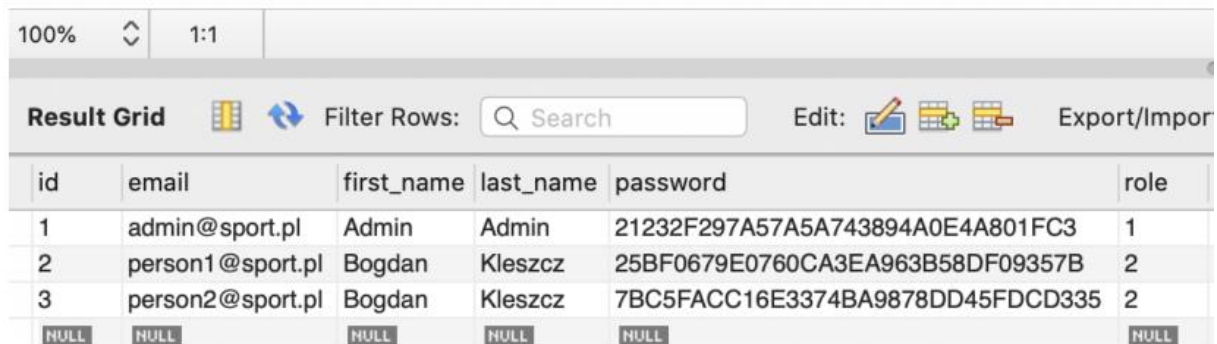
    @PostMapping("login")
    public String login(String email, String password, HttpServletRequest
req, Model model) {
        if (loginService.login(email, password, req, model)) {
            return "redirect:home";
        }
        return "login";
    }
}
```

Proces rejestracji ma zbliżoną strukturę jednak dodatkowo została wzboga-  
cona o kodowanie hasła przy użyciu algorytmu kryptograficznego. Do tego celu  
została użyta biblioteka commons-codec, zawierająca implementacje koderów  
oraz dekoderów. Konkretnie użyte zostało kodowanie do formatu md5. Przykład  
implementacji i użycia poniżej.

Listing. 5. Kodowanie hasła

```
userRepository.save(new User(null, firstName, lastName, email,
    DigestUtils.md5Hex(password).toUpperCase(), 2, null));
```

```
1 • SELECT * FROM sportplan.user;
```



The screenshot shows a database client interface with a 'Result Grid' tab. The query 'SELECT \* FROM sportplan.user;' is entered in the top bar. The result is displayed in a table with 6 columns: id, email, first\_name, last\_name, password, and role. There are 3 data rows and one row with NULL values. The interface includes a search bar, 'Filter Rows' button, and 'Edit' and 'Export/Import' options.

id	email	first_name	last_name	password	role
1	admin@sport.pl	Admin	Admin	21232F297A57A5A743894A0E4A801FC3	1
2	person1@sport.pl	Bogdan	Kleszcz	25BF0679E0760CA3EA963B58DF09357B	2
3	person2@sport.pl	Bogdan	Kleszcz	7BC5FACC16E3374BA9878DD45FD335	2
NULL	NULL	NULL	NULL	NULL	NULL

Rys. 9. Zakodowane hasła w bazie danych

## 6.2. Struktura danych

Spring umożliwia w łatwy sposób konfigurację całej aplikacji. Jednym z takich miejsc jest plik konfiguracyjny `application.properties` opisujący między innymi mechanizm tworzenia danych w bazie, start aplikacji, zdefiniowany dostęp do bazy danych. Przykładowa konfiguracja aplikacji wygląda jak poniżej.

Listing. 6. Zawartość pliku konfiguracyjnego `application.properties`

```
spring.jpa.hibernate.ddl-auto=create-drop
spring.datasource.url=jdbc:mysql://localhost:3306/fitnessplan?autoRecon-
nect=true&useSSL=false&serverTimezone=UTC&allowPublicKeyRetrieval=true
spring.datasource.username=root
spring.datasource.password=root1234
```

Powyższa część obrazuje konfigurację połączenia z bazą danych. Aby cały proces zautomatyzować, do tego celu zostały użyte następujące technologie:

1. JPA (*Java Persistence API*) – część Javy EE, zestaw klas i metod pozwalający zapisywać obiekty Javowe w bazach danych.
2. ORM (*Object Relational Mapping*) – technologia umożliwiająca mapowanie (zamianę) obiektów Javowych na encje bazodanowe.
3. Hibernate – *framework* ORM zgodny ze specyfikacją JPA.

Klasa może stać się encją bazodanową, dodając nad definicją klasy adnotację `@Entity`. Spowoduje to automatyczne przekonwertowanie obiektu klasy na tabelę w bazie danych. Każda encja musi zawierać klucz główny, który określa się przy pomocy dodania adnotacji `@Id` nad odpowiednim polem. Jeżeli ustawiona wartość własności `ddl-auto` (*application.properties*) jest inna niż *none*, to Hibernate podczas uruchomienia aplikacji automatycznie utworzy w bazie danych tabelę na tę encję [17].

Listing. 7. Encja użytkownika

```
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String firstName;
    private String lastName;
    private String email;
    private String password;
    private int role;
    @OneToMany
    @JoinColumn(name = "user_id")
    private List<Plan> planList;
}
```



Spring pozwala na automatyczne generowanie DAO/repozytoriów zawierających operacje umożliwiające operowanie na danych. W ten sposób można w prosty sposób stworzyć repozytorium odpowiedzialne za operacje CRUD (*create, read, update, delete*), poprzez interfejs rozszerzający `CrudRepository` `<ENTITY_TYPE, ID_TYPE>`. Spring automatycznie wygeneruje nam komponent tego repozytorium z podstawowymi operacjami.

Listing. 8. Repozytorium odpowiedzialne za operacje CRUD

```
public interface UserRepository extends CrudRepository<User, Integer> {  
    List<User> findAll();  
    Optional<User> findByEmail(String email);  
}
```

Hibernate wspiera mapowanie relacji między obiektami. Jest to możliwe za pomocą odpowiednich adnotacji:

1. `@OneToMany` – jeden do wiele.
2. `@ManyToOne` – wiele do jeden.
3. `@ManyToMany` – wiele do wiele.
4. `@OneToOne` – jeden do jednego.

`FetchType` określa w jaki sposób zostaną wyciągnięte powiązane ze sobą dane z bazy danych.

1. `FetchType.EAGER` – jeśli obiekt zawiera w sobie inne obiekty, ich dane zostaną automatycznie pobrane.
2. `FetchType.LAZY` – jeśli obiekt zawiera w sobie inne obiekty, ich dane zostaną pobrane dopiero podczas próby dostępu do nich.

Ze względu na fakt, że `EAGER` może spowodować pobranie bardzo dużej ilości danych, domyślnym i zalecanym ustawieniem jest `LAZY` [17].

### 6.3. Interfejs użytkownika.

Interfejs użytkownika początkowo miał zostać stworzony za pomocą technologii Angular, lecz na dalszym etapie rozwoju aplikacji zdecydowano się skorzystać z silnika szablonów Theamleaf, na którego korzyść przemawia pełna integracja ze Springiem, w połączeniu z frameworkiem Bootstrap, który okazał się być wystarczający do stworzenia widoków. Theamleaf oraz Bootstrap to technologie mniej zaawansowane od Angulara, lecz za to bardziej intuicyjne (wykorzystuje podstawowe technologie takie jak HTML, CSS oraz JavaScript, zamiast TypeScriptu), a także do uruchomienia aplikacji wystarcza jeden serwer (w przypadku aplikacji stworzonej za pomocą Angulara są potrzebne dwa).

Połączenie interfejsu użytkownika z resztą aplikacji polega na stworzeniu kontrolerów zarządzających widokami oraz samych widoków HTML-owych. Poniżej znajduje się kontroler widoku rejestracji, w którym pierwsza metoda typu GET służy do przekazania widoku do przeglądarki, druga natomiast typu POST służy do odebrania danych od użytkownika.

Listing. 9. Kontroler rejestracji

```
@Controller
public class RegisterViewController {

    @Autowired
    private RegisterService registerService;

    @GetMapping("register")
    public ModelAndView registerView() {
        return new ModelAndView("register");
    }

    @PostMapping("register")
    public String register(String firstName, String lastName, String email,
String password, String confirmPassword,
        Model model){
        if (registerService.register(firstName, lastName, email, password,
confirmPassword, model)) {
            return "login";
        }
        return "register";
    }
}
```

Część widoku stworzonego w HTML oraz siatce bootstrapowej odpowiadającej za pobranie danych od użytkownika prezentuje się następująco:

Listing. 10. Część widoku odpowiedzialnego za pobranie danych od użytkownika

```
<form method="post" action="register">
  <div class="form-row">
    <div class="col-md-6">
      <div class="form-group">
        <label class="small mb-1" for="inputFirstName">Imie</label>
        <input class="form-control py-4" name="firstName" id="inputFirstName" type="text" placeholder="Podaj imie" />
      </div>
    </div>
    <div class="col-md-6">
      <div class="form-group">
        <label class="small mb-1" for="inputLastName">Nazwisko</label>
        <input class="form-control py-4" name="lastName" id="inputLastName" type="text" placeholder="Podaj nazwisko" />
      </div>
    </div>
  </div>
</form>
```

Theamleaf umożliwia także przesyłanie obiektów javowych do widoków. W kontrolerze wygląda to następująco:

Listing. 11. Przesłanie widoku wraz z listą kategorii

```
@GetMapping("home")
public ModelAndView homeView(Model model) {
    model.addAttribute("categoryList", workoutService.getWorkoutCategories());
    return new ModelAndView("home");
}
```

Odwołanie się do obiektów javowych w widoku zostało pokazane na poniższym przykładzie, gdzie pobrana zostaje lista kategorii w pętli.

Listing. 12. Iteracja po liście kategorii

```
<div class="col-xxl-3 col-lg-6" th:each="category: ${categoryList}">
  <div class="card bg-primary text-white mb-4">
    <div class="card-body">
      <div class="d-flex justify-content-between align-items-center">
        <div class="mr-3">
          <a class="text-white text-lg font-weight-bold text-lg font-weight-bold" th:href="'categoryWorkout?category=' + ${category}" th:text="${category}"></a>
        </div>
        <i class="feather-xl text-white-50" data-feather="grid"></i>
      </div>
    </div>
  </div>
</div>
```

```
</div>  
</div>
```

Aby wykorzystać w szablonach .html Bootstrap, należy zaimportować do szablonu .html odpowiednie pliki, między innymi gotowy plik .css, dostępny pod adresem <https://getbootstrap.com/>.

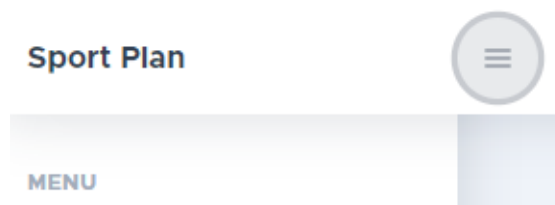
Listing. 13. Zaimportowanie pliku .css

```
<link href="css/styles.css" rel="stylesheet"/>
```

Kiedy odpowiednie pliki zostaną zaimportowane, można zacząć używać klas bootstrapowych, dodając do elementów HTML-owych DOM (*Document Object Model*) odpowiednie klasy CSS. Poniżej znajduje się przykład implementacji menu za pomocą bootstrapowych klas.

Listing. 14. Użycie bootstrapowych klas do stworzenia responsywnego menu

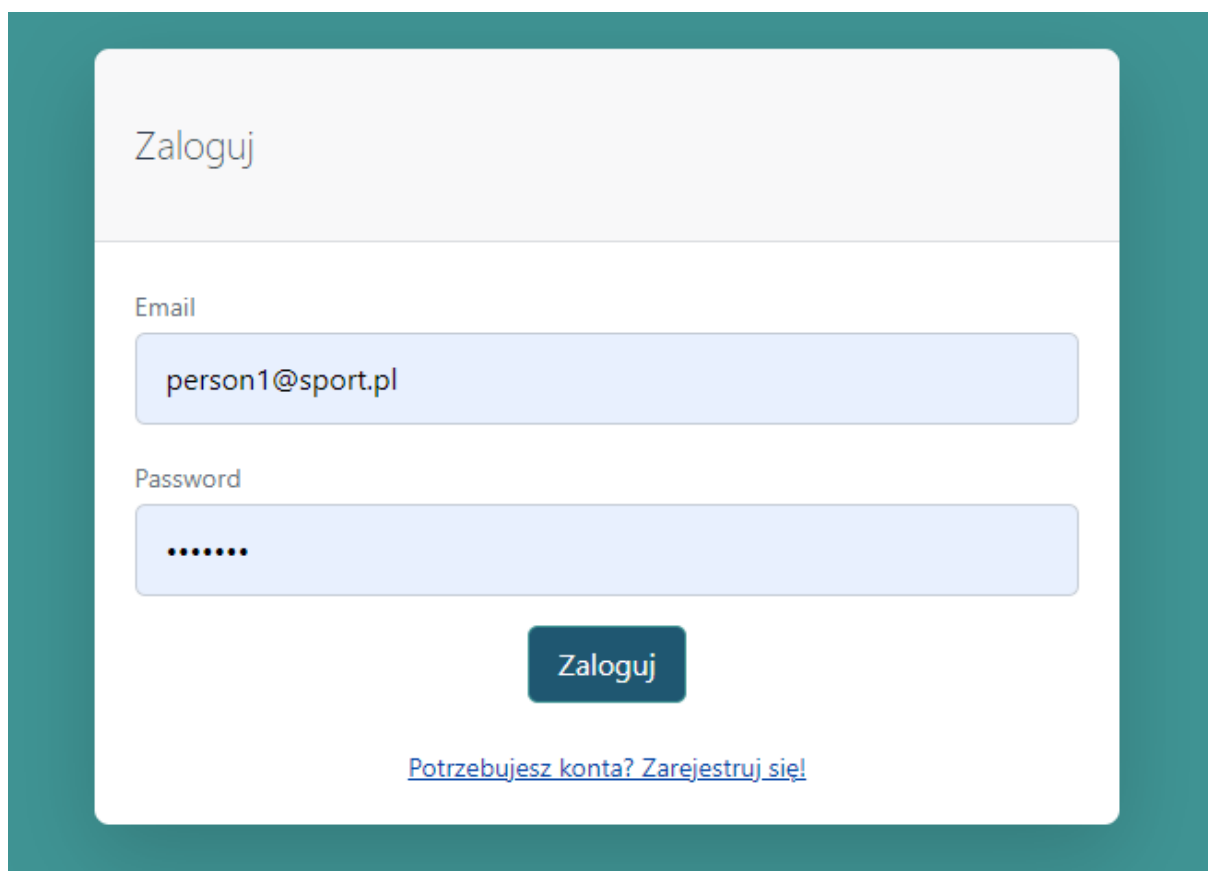
```
<nav class="topnav navbar navbar-expand shadow navbar-light bg-white"  
id="sidenavAccordion">  
  <a class="navbar-brand" href="/home">Sport Plan</a>  
  <button class="btn btn-icon btn-transparent-dark order-1 order-lg-0 mr-  
lg-2" id="sidebarToggle" href="#"><i  
    data-feather="menu"></i></button>  
</nav>
```



Rys. 10. Rozwijane przyciskiem menu aplikacji

## 7. Prezentacja aplikacji

Pierwszym widokiem z którym spotka się użytkownik po wejściu na stronę aplikacji jest strona logowania. Użytkownik nie posiadający konta musi oczywiście przejść przez proces rejestracji.

The image shows a login form within a teal-colored frame. At the top, the word "Zaloguj" is displayed in a light gray header. Below this, there are two input fields: "Email" containing the text "person1@sport.pl" and "Password" containing seven dots. A dark teal "Zaloguj" button is positioned below the password field. At the bottom, there is a blue hyperlink that reads "Potrzebujesz konta? Zarejestruj się!".

Zaloguj

Email

person1@sport.pl

Password

.....

Zaloguj

[Potrzebujesz konta? Zarejestruj się!](#)

Rys. 11. Widok logowania

Rejestracja

Imie Nazwisko

Podaj imie Podaj nazwisko

Email

Podaj email

Hasło Powtórz Hasło

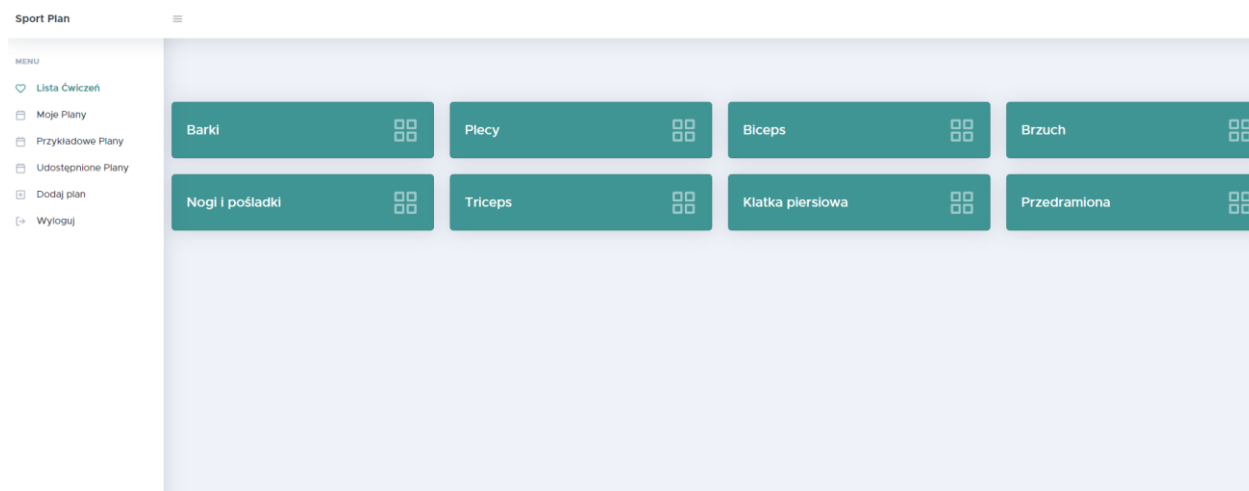
Podaj Hasło Powtórz Hasło

Zarejestruj

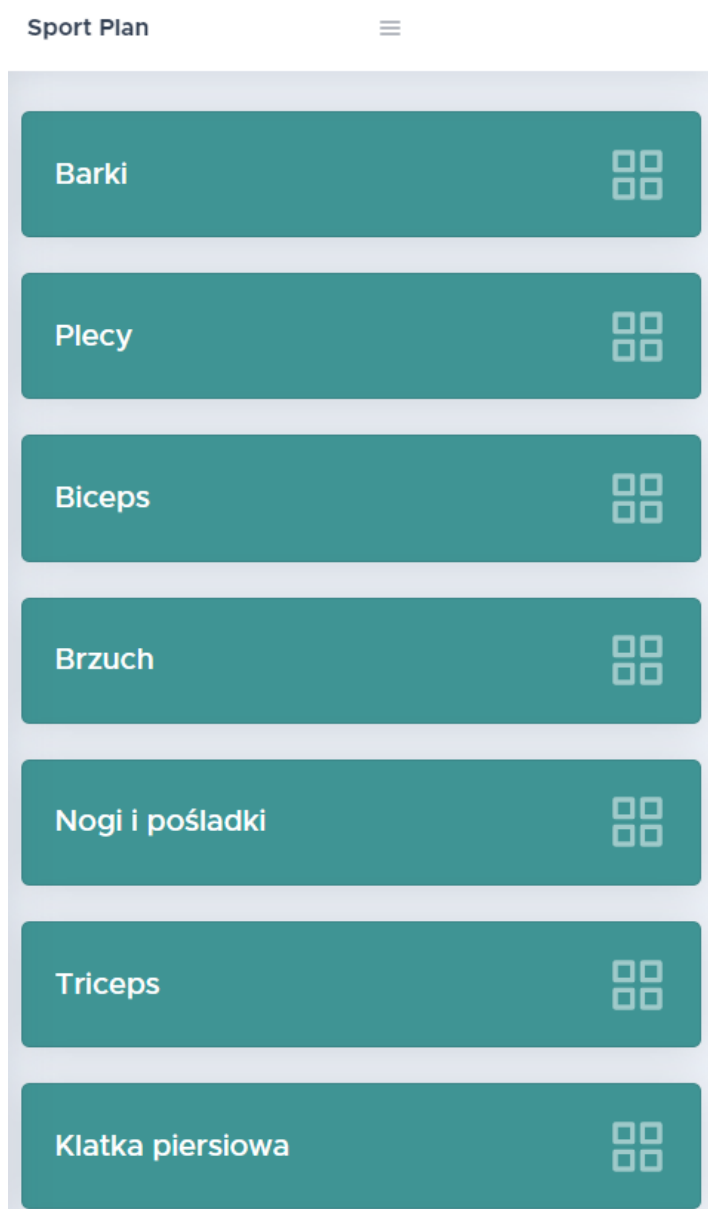
[Powrót do strony logowania](#)

Rys. 12. Widok rejestracji

Po zalogowaniu się na stronę domyślnie pojawia się widok z podziałem na kategorie ćwiczeń. W lewym górnym rogu znajduje się rozwijane menu, umożliwiające przejście w inne funkcjonalności aplikacji.

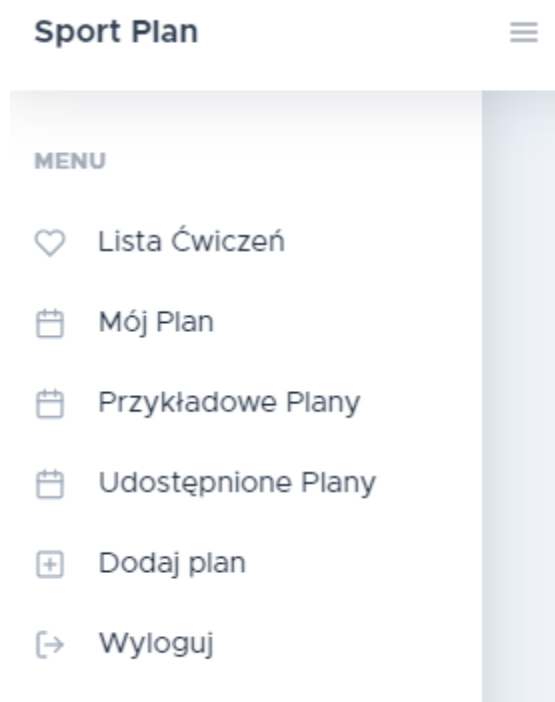


Rys. 13. Widok strony domyślnej z poziomą przeglądarką z rozwiniętym menu



Rys. 14. Mobilny widok strony domyślnej

Poniżej znajduje się widok rozwiniętego menu.



Rys. 15. Widok rozwiniętego menu

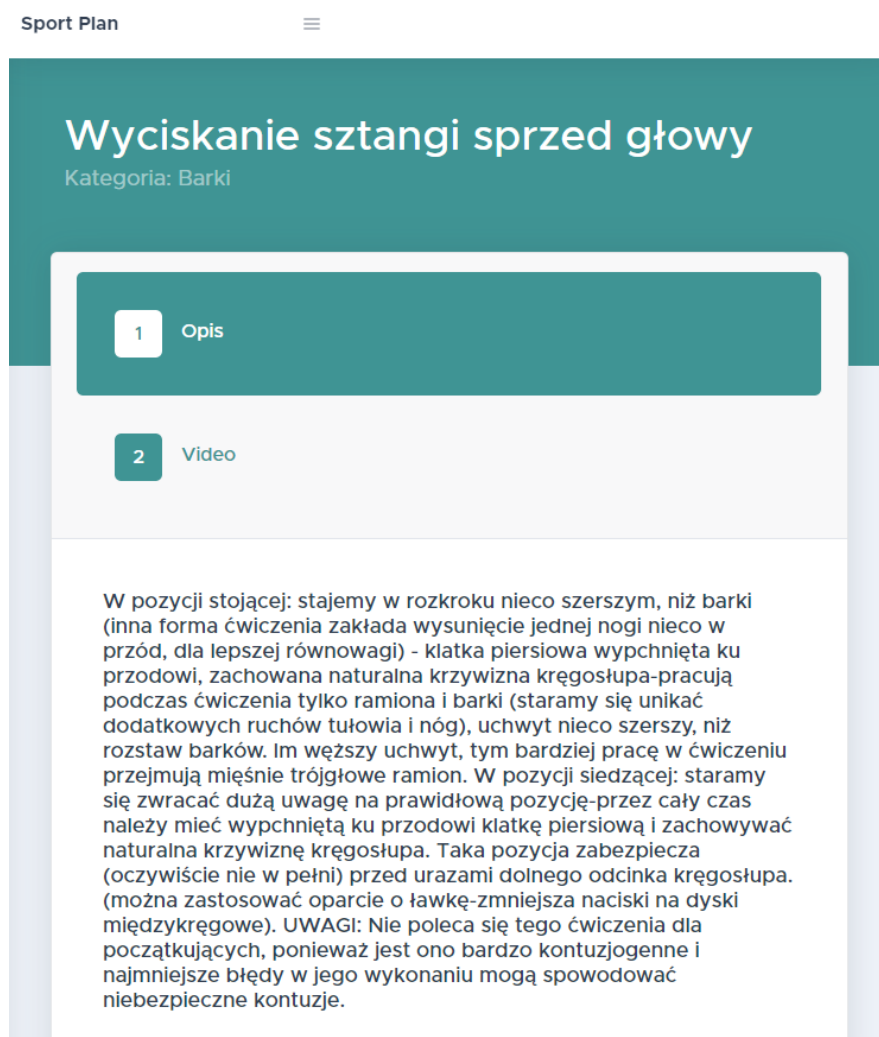
Przechodząc w daną kategorię, otrzymujemy listę ćwiczeń znajdującą się w tej kategorii.

Kategoria	Nazwa	Opis
Barki	Wyciskanie sztangi sprzed głowy	Szczegóły
Barki	Wyciskanie sztangielek	Szczegóły

Rys. 16. Widok listy ćwiczeń w danej kategorii



Przechodząc w szczegóły można zauważyć opis danego ćwiczenia a także film instruktażowy.



Rys. 17. Opis danego ćwiczenia

# Wyciskanie sztangi sprzed głowy

Kategoria: Barki

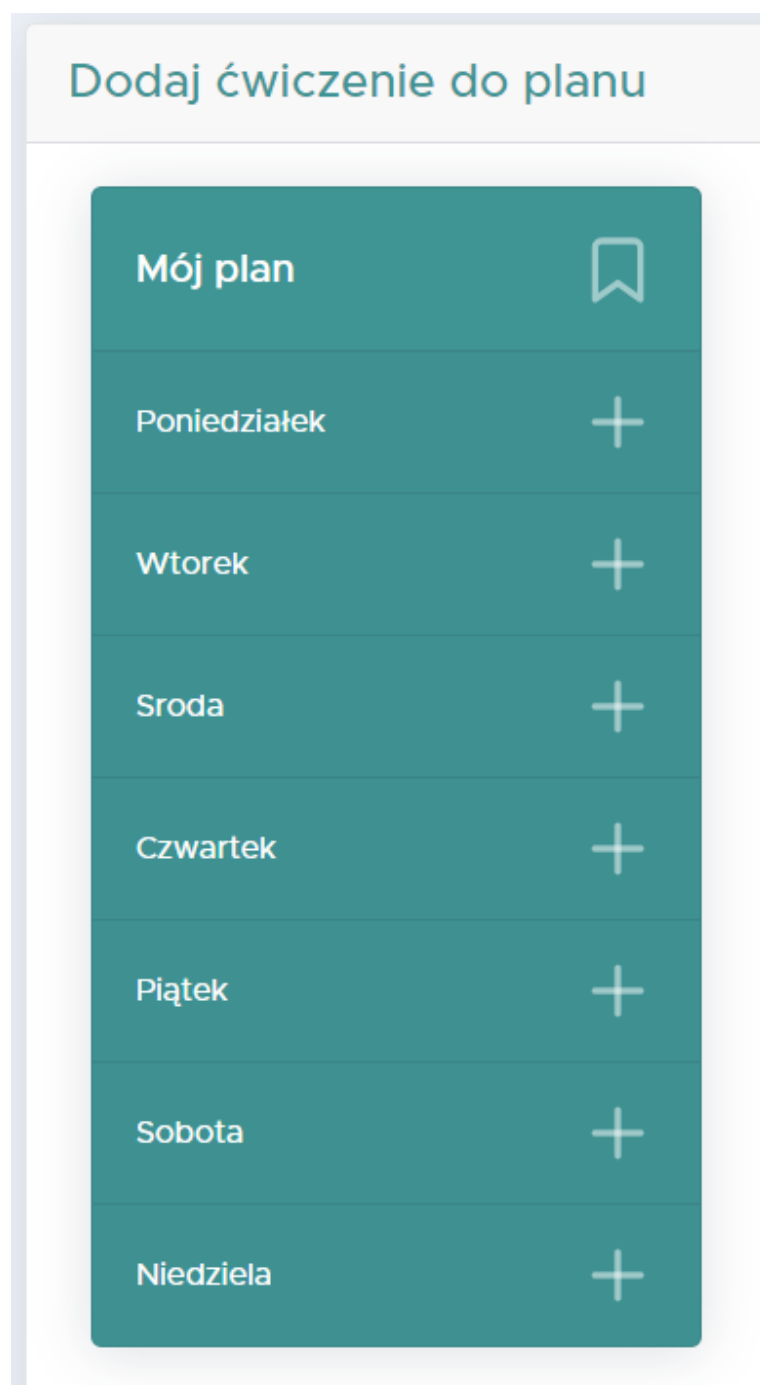
1 Opis

2 Video



Rys. 18. Film instruktażowy danego ćwiczenia

Na samym dole tej strony znajduje się funkcjonalność pozwalająca dodać dane ćwiczenie do jednego z dni naszych indywidualnych planów treningowych.



Rys. 19. Możliwość dodania ćwiczenia do jednego z własnych planów

Poprzez kliknięcie w dany dzień, ćwiczenie zostaje do niego dodane. Przechodząc w zakładkę Mój Plan, oraz wybierając wybrany plan z listy można zobaczyć jego harmonogram tygodniowy, z poziomu którego można usunąć ćwiczenie z planu lub wejść w jego szczegóły. W zakładce Mój Plan możemy też udostępnić dany plan innym użytkownikom.

Nazwa	Opis	Szczegóły	Edytuj	Usuń	Udostępnij
Mój plan	Trening FBW	Szczegóły	Edytuj	Usuń	Udostępnij
Mój drugi plan	Trening siłowy	Szczegóły	Edytuj	Usuń	Udostępnij

Poprzedni 1 Kolejny

Rys. 20. Widok listy planów użytkownika

## Mój plan

Trening FBW

Poniedziałek

Wyciskanie sztangi przed głową

Usuń

Wtorek

Sroda

Czwartek

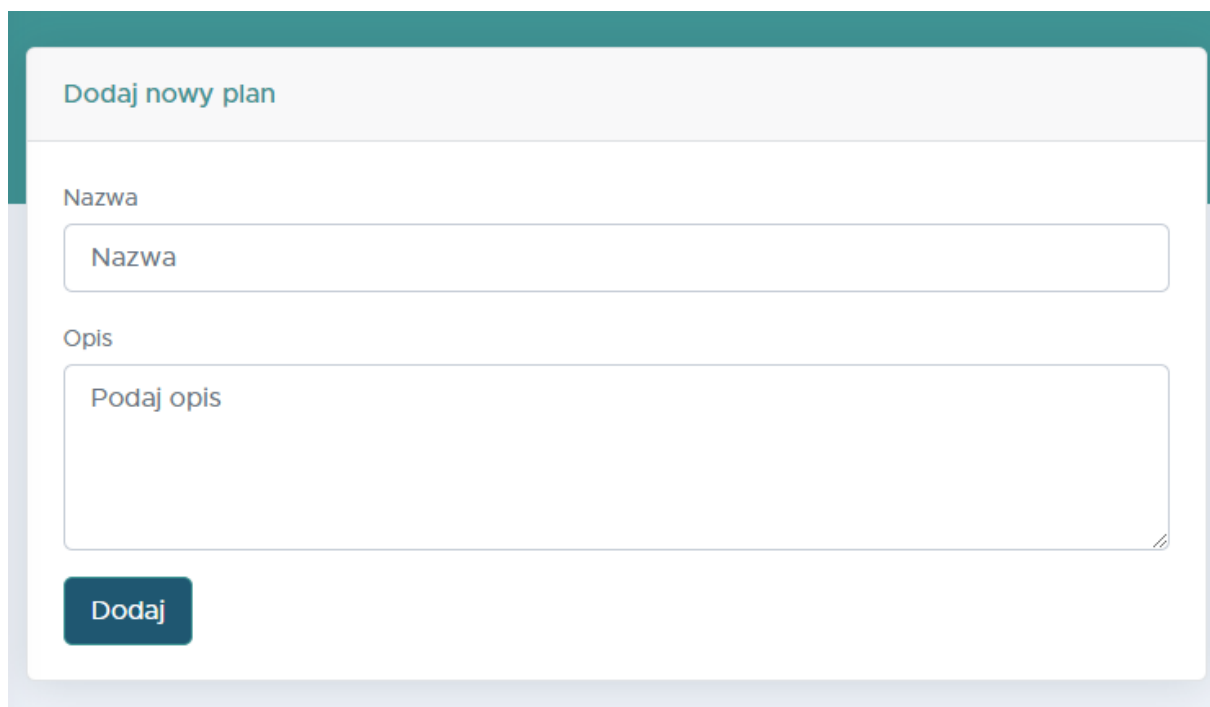
Piątek

Sobota

Niedziela

Rys. 21. Widok harmonogramu tygodniowego jednego z naszych planów

Dodanie własnego planu z kolei następuje po wejściu w zakładkę Dodaj plan.

The image shows a web form titled "Dodaj nowy plan" (Add new plan) in a teal header. Below the header, there are two input fields. The first is labeled "Nazwa" (Name) and contains the placeholder text "Nazwa". The second is labeled "Opis" (Description) and contains the placeholder text "Podaj opis" (Provide description). At the bottom left of the form is a dark teal button labeled "Dodaj" (Add).

Rys. 22. Dodawanie nowego indywidualnego planu

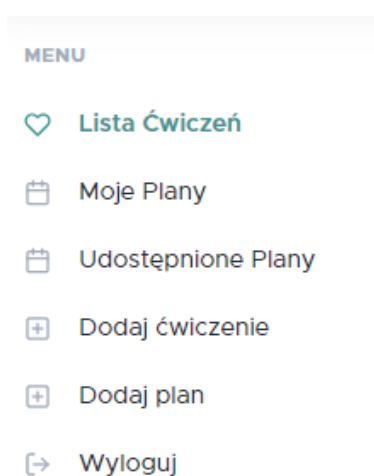
Kolejną funkcjonalnością jest możliwość przeglądania przykładowych oraz udostępnionych planów (wraz z ich opisem i harmonogramem tygodniowym znajdującym się w szczegółach). Plany przykładowe są udostępniane przez administratora, natomiast udostępnione plany są publikowane przez użytkowników.

Nazwa ⇅	Opis ⇅	Szczegóły ⇅
Trening na poprawę siły	Każda grupa mięśniowa trenowana raz w tygodniu. Trening na dużych ciężarach z małą ilością powtórzeń w seriach(8-1).Przerwy pomiędzy seriami wydłużamy do 3-5 minut.	Szczegóły
Trening na masę	Oparty powinien być na ćwiczeniach podstawowych(przysiady, wyciskania, uginania i wyprosty ramion, itp.)z wolnymi ciężarami(sztanga, sztangielki)	Szczegóły

[Poprzedni](#)
[1](#)
[Kolejny](#)

Rys. 23. Lista przykładowych planów

Ostatnim elementem przedstawionym w tym rozdziale jest menu administratora, które delikatnie różni się od menu zwykłego użytkownika. Administrator może zarządzać przykładowymi planami, w taki sposób jak zwykły użytkownik zarządza swoimi (plany administratora są planami przykładowymi), oraz może zarządzać ćwiczeniami (dodawać, edytować, usuwać).



Rys. 24. Menu administratora

Dodaj nowe ćwiczenie

Kategoria

Kategoria

Nazwa

Nazwa

Opis

Podaj opis

Video Link

<https://youtube.pl/embed/mbyTbDJBsR8>

Dodaj

Rys. 25. Widok dodawania ćwiczenia wraz z przykładowym linkiem

## 8. Testy aplikacji

Dla stworzonej aplikacji zostały stworzone testy jednostkowe, wykonane za pomocą JUnit, które weryfikują działanie pojedynczych elementów. Został wykonany test automatyczny (używając oprogramowania i skryptów), sprawdzający poprawne działanie logowania, wykonany za pomocą Selenium WebDriver. Zostały także wykonane testy manualne (wykonane przez człowieka).

Testy jednostkowe sprawdziły poprawność operacji wykonywanych na użytkowniku, między innymi: usuwanie użytkownika, zapisywanie użytkownika, pobieranie danych użytkownika z poziomu serwisu oraz repozytorium.

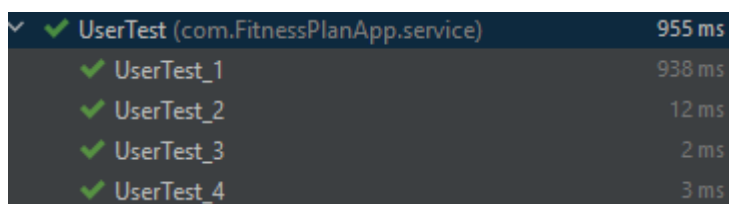
Listing. 15. Test jednostkowy sprawdzający poprawność pobrania użytkownika z serwisu

```
@Test
public void UserTest_4() {
    User exampleUser = new User(null, "admin", "admin", "admin@sport.pl",
    "admin", 1, null);

    when(userRepository.findByEmail("admin@sport.pl")).thenReturn(
    java.util.Optional.of(exampleUser));

    User userFromService = loginService.getByEmail("admin@sport.pl");

    assertEquals("admin", userFromService.getFirstName());
    assertEquals("admin", userFromService.getPassword());
    assertEquals(1, userFromService.getRole());
}
```



✓ ✓ UserTest (com.FitnessPlanApp.service)	955 ms
✓ UserTest_1	938 ms
✓ UserTest_2	12 ms
✓ UserTest_3	2 ms
✓ UserTest_4	3 ms

Rys. 26. Zaliczone testy jednostkowe użytkownika



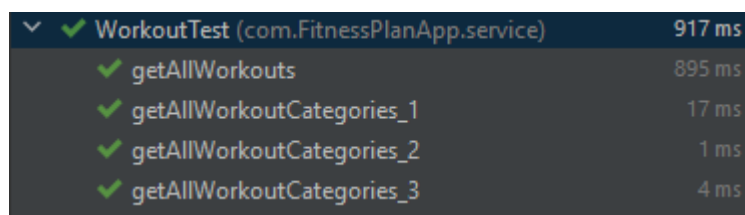
Listing. 16. Test jednostkowy sprawdzający poprawność zwrócenia danych ćwiczenia

```
@Test
public void getAllWorkoutCategories_3() {
    Workout workout = new Workout(null,
        "Cat",
        "Name",
        "Desc",
        "http://test",
        null);

    when(workoutRepository.findById(1)).thenReturn(java.util.Optional.of(workout));

    Workout workoutFromRepo = workoutRepository.findById(1).get();

    assertEquals("Cat", workoutFromRepo.getCategory());
    assertEquals("Name", workoutFromRepo.getName());
    assertEquals("Desc", workoutFromRepo.getDescription());
    assertEquals("http://test", workoutFromRepo.getVideoLink());
}
```

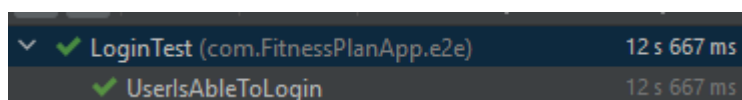
A screenshot of a test runner interface showing the results of a test suite named 'WorkoutTest (com.FitnessPlanApp.service)'. The suite has a total duration of 917 ms and all tests passed, indicated by green checkmarks. The individual tests and their durations are: 'getAllWorkouts' (895 ms), 'getAllWorkoutCategories\_1' (17 ms), 'getAllWorkoutCategories\_2' (1 ms), and 'getAllWorkoutCategories\_3' (4 ms).

✓ WorkoutTest (com.FitnessPlanApp.service)	917 ms
✓ getAllWorkouts	895 ms
✓ getAllWorkoutCategories_1	17 ms
✓ getAllWorkoutCategories_2	1 ms
✓ getAllWorkoutCategories_3	4 ms

Rys. 27. Zaliczone testy jednostkowe ćwiczeń

Listing. 17. Test automatyczny logowania

```
@Test
public void UserIsAbleToLogin() {
    driver.get("http://localhost:8080/");
    driver.findElement(By.xpath("//*[@id='inputEmailAddress']")).sendKeys("admin@sport.pl");
    driver.findElement(By.xpath("//*[@id='inputPassword']")).sendKeys("admin");
    driver.findElement(By.xpath("//input[@value='Zaloguj']")).click();
    driver.getCurrentUrl().contains("home");
}
```

A screenshot of a test runner interface showing the results of a test suite named 'LoginTest (com.FitnessPlanApp.e2e)'. The suite has a total duration of 12 s 667 ms and the test passed, indicated by a green checkmark. The individual test and its duration are: 'UserIsAbleToLogin' (12 s 667 ms).

✓ LoginTest (com.FitnessPlanApp.e2e)	12 s 667 ms
✓ UserIsAbleToLogin	12 s 667 ms

Rys. 28. Zaliczony test automatyczny logowania



## 9. Podsumowanie i wnioski

Głównym celem pracy inżynierskiej było stworzenie aplikacji webowej wspomagającej tworzenie planów treningowych. Cel ten udało mi się zrealizować, implementując wszystkie zakładane początkowo funkcjonalności. Był to pierwszy tak rozległy projekt który stworzyłem. Pisząc tego typu aplikację dowiedziałem się jak wygląda proces tworzenia oprogramowania webowego od początku do końca, wliczając w to testy. Poznałem też szeroki wachlarz nowych technologii, które przydają mi się na dalszych etapach nauki oraz kariery zawodowej. Jeśli chodzi o początkowe założenia technologiczne, to zostały one spełnione za wyjątkiem zakładanego Angulara. Na pewnym etapie projektu zdecydowałem się na użycie mniej skomplikowanej lecz jednocześnie wystarczającej technologii, jaką jest Thymleaf w połączeniu z Bootstrapem. Ostatecznie interfejs aplikacji spełnił zakładane warunki estetyczne, a program stanowi spójną, poprawnie działającą całość.

# Literatura

- [1] Bloch Joshua, Java. Efektywne programowanie, Helion 2018.
- [2] Kathy Sierra i Bert Bates, Head First Java, Helion 2004
- [3] Aplikacje internetowe, [http://www.informatyka.orawskie.pl/?pl\\_aplikacje-internetowe](http://www.informatyka.orawskie.pl/?pl_aplikacje-internetowe), 139, [28.10.2020]
- [4] Atlas ćwiczeń, <http://atlas.kfd.pl/>, [11.12.2020]
- [5] Atlas ćwiczeń, <https://www.kulturystyka.pl/atlas/>, [11.12.2020]
- [6] Dokumentacja bazy danych MySQL, <https://dev.mysql.com/doc/>, [07.11.2020]
- [7] Dokumentacja frameworka Bootstrap, <https://getbootstrap.com/>, [03.12.2020]
- [8] Dokumentacja frameworka Spring, <https://spring.io/>, [27.11.2020]
- [9] Dokumentacja języka programowania Java, <https://docs.oracle.com/en/java/>, [15.11.2020]
- [10] Dokumentacja narzędzia Maven, <https://maven.apache.org/>, [17.10.2020]
- [11] Kanał The ATHLEAN-X, <https://www.youtube.com/user/JDCav24>, [20.10.2020]
- [12] Licencja MIT, [https://pl.wikipedia.org/wiki/Licencja\\_MIT](https://pl.wikipedia.org/wiki/Licencja_MIT), [03.12.2020]
- [13] Model widok kontroler, <https://pl.wikipedia.org/wiki/Model-View-Controller>, [03.12.2020]
- [14] Szablony interfejsu użytkownika Bootstrap, <https://themes.getbootstrap.com/>, [01.12.2020]
- [15] Środowisko IntelliJ, <https://www.jetbrains.com/idea/>, [11.12.2020]
- [16] Trening fitness, <https://pl.wikipedia.org/wiki/Trening>, [11.12.2020]
- [17] Tworzenie aplikacji webowych ze Springiem, <https://kobietydokodu.pl/kurs-javy/>, [02.12.2020]

# **Dodatek – opis załączonej płyty CD**

Załączona płyta CD zawiera plik W04\_229218\_2020\_praca\_inżynierska.pdf z dokumentacją pracy dyplomowej, wykonane oprogramowanie skompresowane do postaci FitnessPlanApp.zip, oraz plik Opis\_uruchomienia.pdf z instrukcją instalacji i obsługi wykonanego oprogramowania.