

# BDBT

## Projekt cz. II

---

Klub lekkoatletyczny

Jakub Grzechnik 325278

Jakub Strzelczyk 325325

---

14 marca 2025



**Wydział Elektroniki  
i Technik Informatycznych**

**POLITECHNIKA WARSZAWSKA**

---

## Spis treści

<b>1. Zakres i cel projektu</b>	3
<b>2. Wprowadzenie</b>	3
2.1. Baza danych	3
2.2. Perspektywy użytkowników	4
<b>3. Zastosowana technologia</b>	4
3.1. Baza danych	5
3.2. Spring Boot	5
3.3. JPA, Hibernate	5
3.4. Bootstrap	5
3.5. Thymeleaf	5
3.6. Maven	5
<b>4. Działanie aplikacji</b>	6
4.1. Strona główna	6
4.2. Kontakt	6
4.3. Logowanie	7
4.4. Strona użytkownika	7
4.5. Tabele	8
4.6. Obsługa błędów	8
4.7. Dodatkowe funkcje	10
<b>5. Wnioski</b>	10

## 1. Zakres i cel projektu

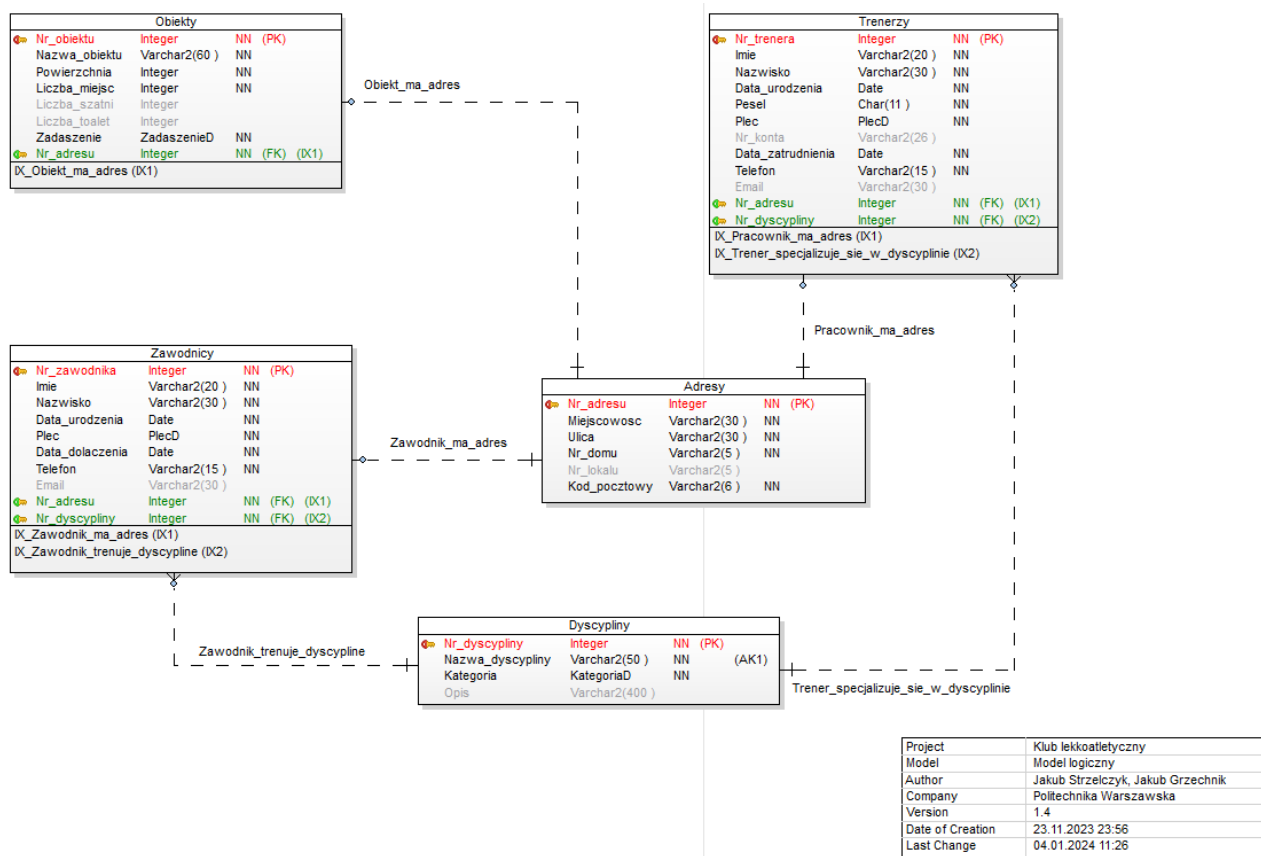
Celem II części projektu z przedmiotu BDBT jest przygotowanie przez zespół projektowy aplikacji współpracującej z wcześniej zaprojektowaną bazą danych. Aplikacja nie musi pokrywać w pełni funkcjonalności wcześniej zaprojektowanej bazy danych (jedynie część, kilka wybranych transakcji bazodanowych – kilka tabel).

## 2. Wprowadzenie

### 2.1. Baza danych

W pierwszej części projektu stworzyliśmy relacyjną bazę danych dla klubu lekkoatletycznego, opartą na technologii Oracle. W kolejnej fazie projektu przeprowadziliśmy uproszczenia w strukturze bazy danych, skupiając się jedynie na kluczowych funkcjonalnościach. Obecnie korzysta ona łącznie z pięciu tabel.

Poniżej przedstawiamy model logiczny uproszczonej wersji projektu.



Rysunek 1. Model logiczny

W bazie danych klubu lekkoatletycznego gromadzone są dane dotyczące trenerów współpracujących z klubem, specjalizujących się w konkretnych dyscyplinach sportowych. Dyscypliny mogą być także trenowane przez zawodników, którzy tworzą relacje z adresami. Adresy trenerów również są uwzględnione w bazie. Dodatkowo, dostępne są informacje dotyczące obiektów sportowych, z których klub korzysta.

## 2.2. Perspektywy użytkowników

Aplikacja obsługuje 3 perspektywy użytkowników wraz z określonymi uprawnieniami przedstawionymi w tabeli poniżej:

- Administrator (A)
- Trener (T)
- Zawodnik (Z)

	A	T	Z
Dodawanie, usuwanie i modyfikacja danych dotyczących zawodników	Tak	Tak	Tak
Podgląd danych dotyczących zawodników	Tak	Tak	Tak
Dodawanie, usuwanie i modyfikacja danych dotyczących trenerów	Tak	Tak	Nie
Podgląd danych dotyczących trenerów	Tak	Tak	Nie
Dodawanie, usuwanie i modyfikacja danych dotyczących adresów	Tak	Tak	Tak
Podgląd danych dotyczących adresów	Tak	Tak	Tak
Dodawanie, usuwanie i modyfikacja danych dotyczących dyscyplin	Tak	Tak	Nie
Podgląd danych dotyczących dyscyplin	Tak	Tak	Tak
Dodawanie, usuwanie i modyfikacja danych dotyczących obiektów	Tak	Nie	Nie
Podgląd danych dotyczących obiektów	Tak	Nie	Nie

Rysunek 2. Tabela - perspektywy użytkowników

## 3. Zastosowana technologia

Podczas pracy najpierw korzystaliśmy z materiałów udostępnionych na ćwiczeniach, a następnie doskonaliliśmy aplikację, tak aby spełniała wszystkie funkcjonalności opisane w wymaganiach.

W celu napisania aplikacji współpracującej z bazą danych wykorzystaliśmy technologię Java w architekturze wielowarstwowej - cienki klient w postaci przeglądarki internetowej. Przy przygotowywaniu rozwiązania korzystaliśmy z różnych frameworków programistycznych, głównie Java Spring, Java Hibernate. Do utworzenia aplikacji wykorzystaliśmy Spring Initializer - narzędzie, które pomaga w tworzeniu aplikacji Spring Boot od podstaw lub na podstawie istniejących projektów Gradle lub Maven. My skorzystaliśmy z narzędzia Maven automatyzującego budowę oprogramowania na platformę Java. Tworząc projekt w Spring Initializer początkowo dodaliśmy także najważniejsze zależności - Spring Web oraz Thymeleaf. Spring Web to część Spring Framework, która dostarcza narzędzi do tworzenia aplikacji internetowych webowych, natomiast Thymeleaf to narzędzie do tworzenia szablonów stron internetowych. W celu utworzenia poszczególnych stron skorzystaliśmy z języka znaczników HTML, a także biblioteki Bootstrap CSS w celu utworzenia odpowiedniego interfejsu graficznego aplikacji.

### 3.1. Baza danych

Baza danych to dostępny dla wielu użytkowników zbiór logicznie powiązanych danych. Jest tworzona w celu zaspokojenia potrzeb informacyjnych klientów. Dzięki systemowi zarządzania bazą danych (SZBD) istnieje możliwość zarządzania, definiowania, tworzenia i utrzymywania bazy danych.

Relacyjna baza danych to rodzaj bazy danych, która opiera się na modelu relacyjnym. Model ten zakłada, że dane są przechowywane w postaci dwuwymiarowych tabel, a relacje między tabelami są reprezentowane za pomocą kluczy obcych. Nasza aplikacja korzysta z relacyjnej bazy danych zaprojektowanej przez nas, a połączenie z nią jest skonfigurowane w pliku `application.properties`.

### 3.2. Spring Boot

Spring Boot to framework oparty na platformie Spring, który umożliwia szybkie i efektywne tworzenie zaawansowanych aplikacji Java. Dostarcza on wbudowany serwer aplikacyjny, co eliminuje konieczność manualnej konfiguracji serwera. Aplikacje napisane w Spring Boot zawierają wszystkie niezbędne zależności oraz komponenty, co umożliwia szybkie i proste uruchomienie aplikacji.

Spring Boot oferuje automatyczną konfigurację, co oznacza, że wiele ustawień i konfiguracji jest domyślnie obsługiwanych przez framework. Domyślne ustawienia pozwalają na uruchomienie zasadniczej aplikacji bez potrzeby ręcznej konfiguracji. Dodatkowo, istnieje możliwość dostosowania konfiguracji do indywidualnych potrzeb, co jest opcjonalne.

Dzięki prostocie i wydajności Spring Boot, proces developmentu staje się szybszy i bardziej efektywny. Framework ten dostarcza gotowych szablonów, ułatwiając skupienie się na innych aspektach, zamiast na konfiguracji i ustawieniach. Szybkość tworzenia aplikacji przekłada się na krótszy czas dostarczania nowych funkcji i zmniejsza koszty rozwoju.

### 3.3. JPA, Hibernate

JPA (Java Persistence API) to specyfikacja w języku Java umożliwiająca mapowanie obiektów Java na struktury baz danych relacyjnych. Hibernate jest popularnym frameworkiem implementującym JPA, który ułatwia zarządzanie danymi w bazach relacyjnych. JPA definiuje encje jako obiekty reprezentujące dane, a Hibernate dostarcza funkcji ORM (Object-Relational Mapping), obsługuje transakcje, oraz umożliwia korzystanie z JPQL (Java Persistence Query Language) do operacji na danych. Wspólnie, JPA i Hibernate stanowią efektywne narzędzie dla pracy z bazami danych w aplikacjach Java, eliminując konieczność bezpośredniego korzystania z zapytań SQL.

### 3.4. Bootstrap

Bootstrap to otwarta biblioteka projektowania stron internetowych (CSS, JavaScript), stworzona przez zespół Twittera. Zapewnia gotowe komponenty interfejsu użytkownika, takie jak przyciski, nawigacja, formularze, oraz wiele innych, co ułatwia projektowanie responsywnych i estetycznych stron internetowych. Bootstrap jest oparty na technologii CSS, a także wykorzystuje komponenty JavaScript, co pozwala na łatwe tworzenie interaktywnych elementów interfejsu.

### 3.5. Thymeleaf

Thymeleaf to silnik szablonów przeznaczony dla warstwy widoku w aplikacjach opartych na Java. Pozwala na łatwe integrowanie dynamicznych treści z kodem Java w plikach HTML. Thymeleaf umożliwia wstawianie danych, warunkowe renderowanie elementów.

### 3.6. Maven

Maven odegrał kluczową rolę w procesie zarządzania projektem. Dzięki niemu ustandaryzowaliśmy budowę projektu, zarządzanie zależnościami oraz automatyzację wielu procesów związanych z developmentem. Plik konfiguracyjny `pom.xml` stał się centralnym punktem zarządzania projektem, zawierając informacje o strukturze projektu, zależnościach, pluginach i innych kluczowych ustawieniach. W efekcie uzyskaliśmy jednolity i uporządkowany sposób zarządzania projektem w kontekście technologii Java.

## 4. Działanie aplikacji

### 4.1. Strona główna

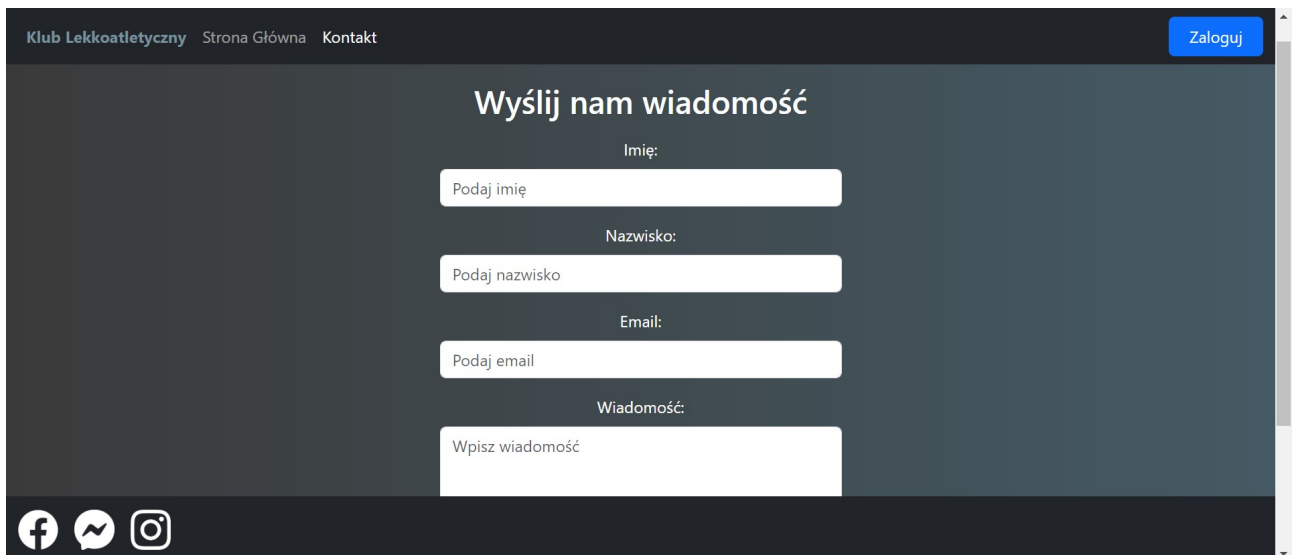
Po uruchomieniu aplikacji znajdujemy się na domyślnej stronie startowej przedstawionej na rysunku 3, dostępnej dla wszystkich użytkowników.



Rysunek 3. Strona startowa

### 4.2. Kontakt

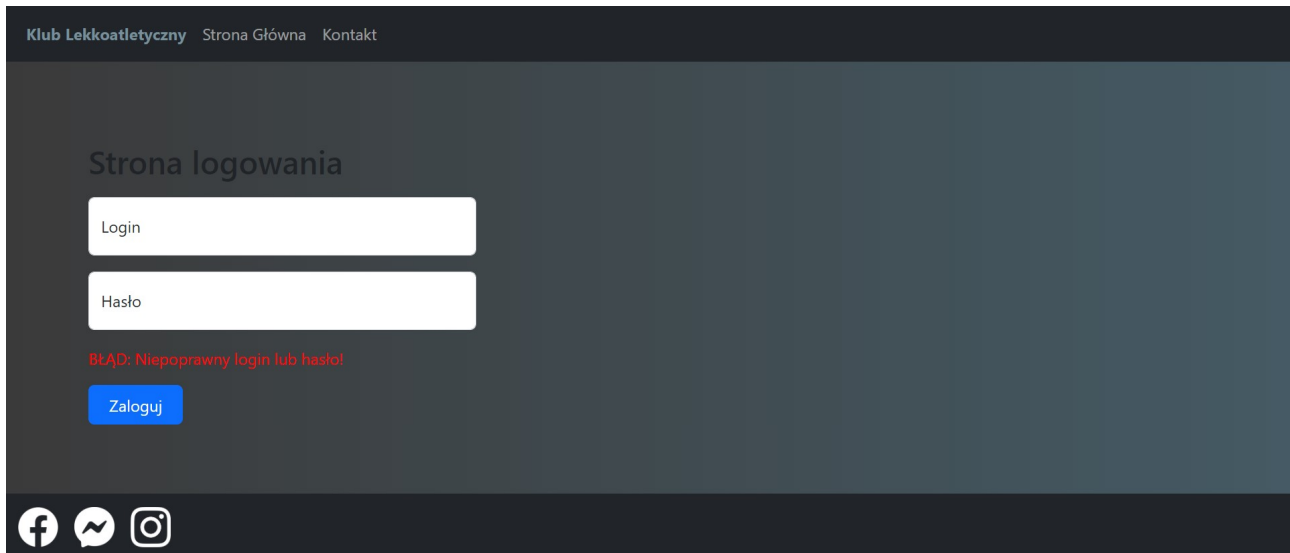
Możemy również przejść do karty Kontakt przedstawionej na rysunku 4, w której docelowo możliwy będzie kontakt z właścicielem klubu.



Rysunek 4. Strona kontaktowa

### 4.3. Logowanie

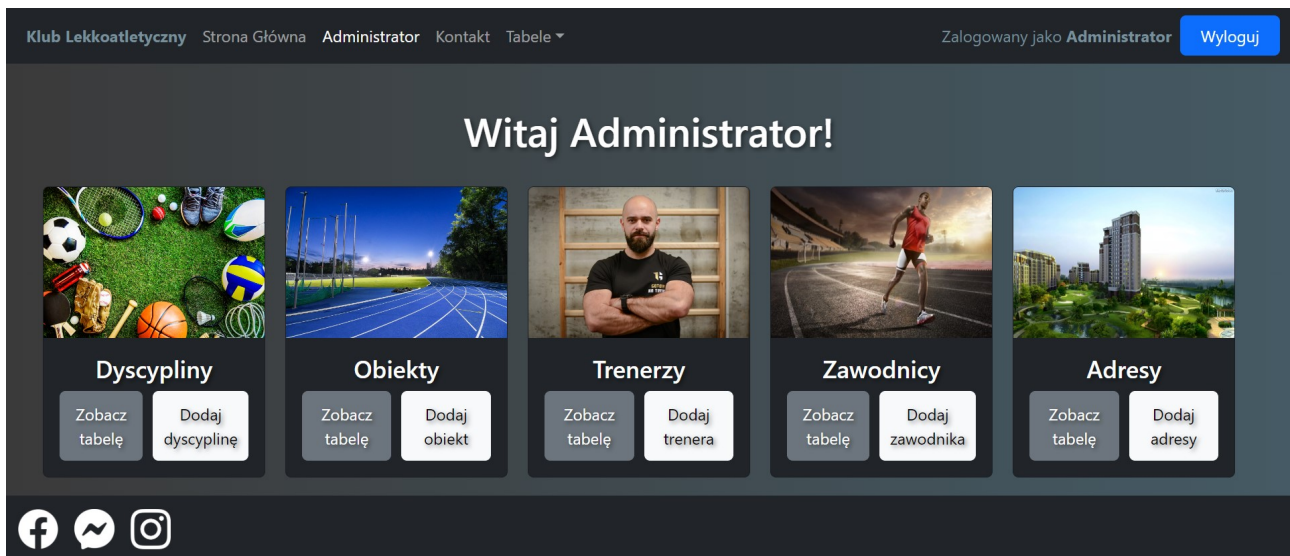
Aby przejść do konkretnych perspektyw użytkowników, musimy przejść na stronę logowania. Możliwe są trzy opcje logowań, jako: administrator, trener lub zawodnik. Ich uprawnienia przedstawione są szczegółowo na rysunku 2. Po wprowadzeniu błędnych danych na naszej stronie logowania wyświetla się komunikat o błędzie, co możemy zobaczyć na rysunku 5.



Rysunek 5. Strona logowania

### 4.4. Strona użytkownika

Działanie aplikacji pokażemy z perspektywy administratora, który ma dostęp do wszystkich możliwych funkcji. Strona główna po zalogowaniu przedstawiona jest na rysunku 6.



Rysunek 6. Strona Administratora

#### 4.5. Tabele

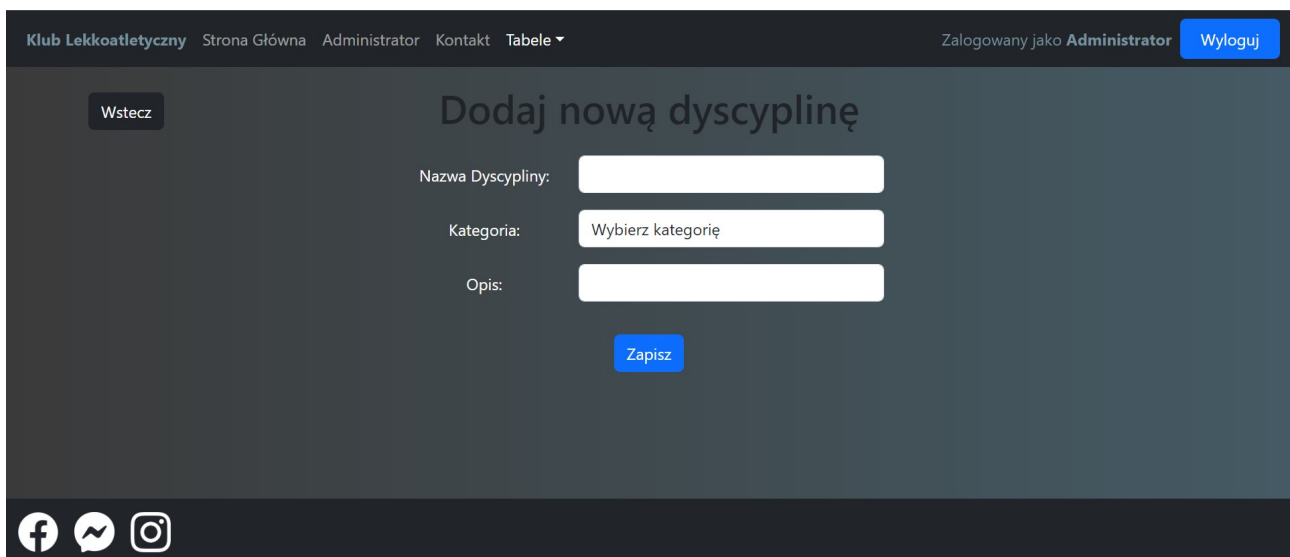
Na poniższym rysunku 7 przedstawiamy przykładową tabelę z naszej bazy danych. Jako administrator, każdy wiersz możemy edytować, usuwać, a także dodawać nowy.



Numer Dyscypliny	Nazwa Dyscypliny	Kategoria	Opis		
2	Bieg na 100m	Biegi	Sprint na dystansie 100 metrów	Edytuj	Usuń
3	Bieg na 3000m	Biegi	Bieg na długim dystansie 3000 metrów	Edytuj	Usuń
4	Skok wzwyż	Skoki	Skok wzwyż przez poprzeczkę	Edytuj	Usuń
5	Skok o tyczce	Skoki	Skok o tyczce przez poprzeczkę	Edytuj	Usuń
6	Rzut dyskiem	Rzuty	Rzut dyskiem	Edytuj	Usuń
7	Rzut oszczepem	Rzuty	Rzut oszczepem	Edytuj	Usuń

Rysunek 7. Tabela dyscypliny

Na rysunku 8 pokazany jest widok dodawania nowego wiersza do tabeli.



Nazwa Dyscypliny:

Kategoria:

Opis:

Rysunek 8. Modyfikacja tabeli dyscypliny

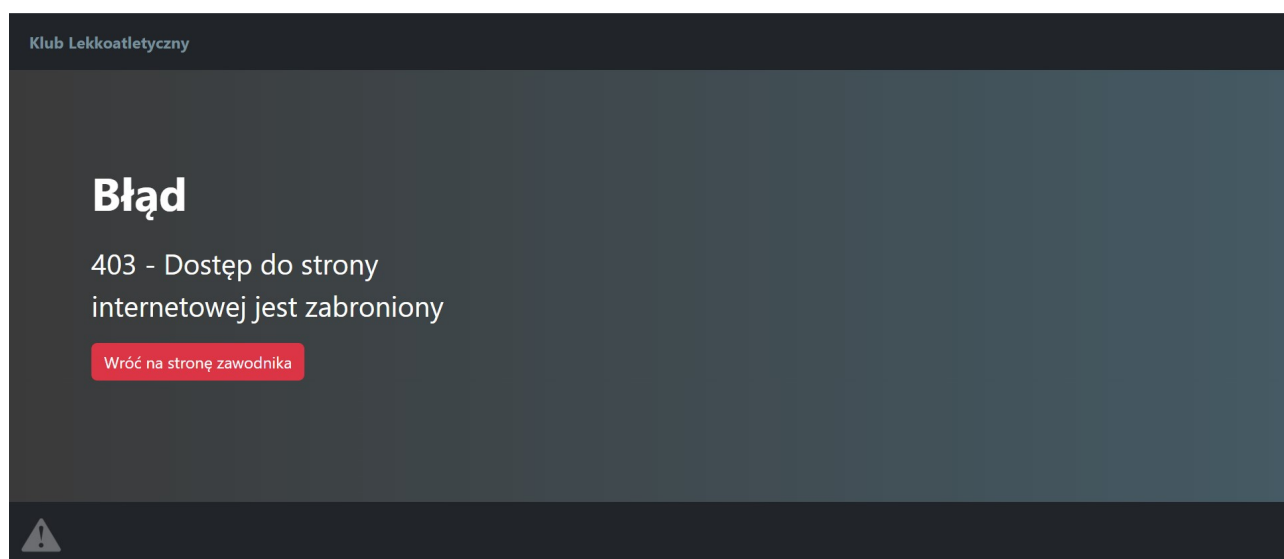
#### 4.6. Obsługa błędów

Nasz program obsługuje następujące błędy:

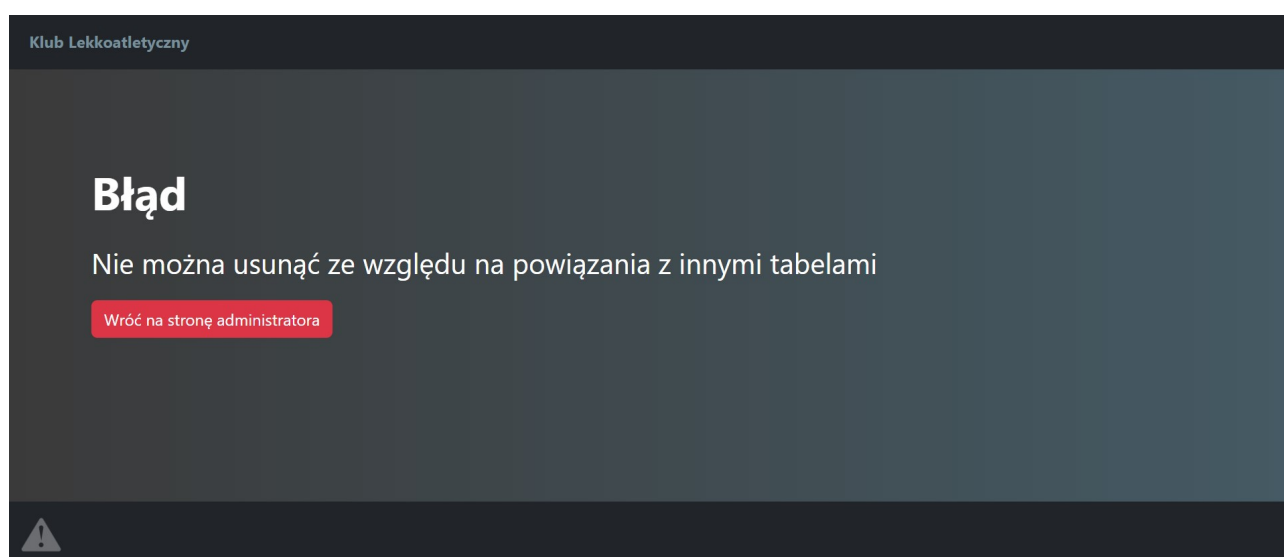
- **403** - występuje gdy nie mamy uprawnień, żeby wejść na stronę
- **404** - występuje gdy strona której szukamy nie istnieje
- **500** - występuje przy błędzie wewnętrznym serwera
- **504** - występuje gdy zostanie przekroczony limit czasu serwera
- **ORA-02292** - występuje gdy nie można usunąć wiersza ze względu na powiązania z innymi tabelami
- **other** - występuje przy innych nieoczekiwanych błędach



Na rysunkach 9 i 10 przedstawione są widoki dwóch przykładowych błędów.



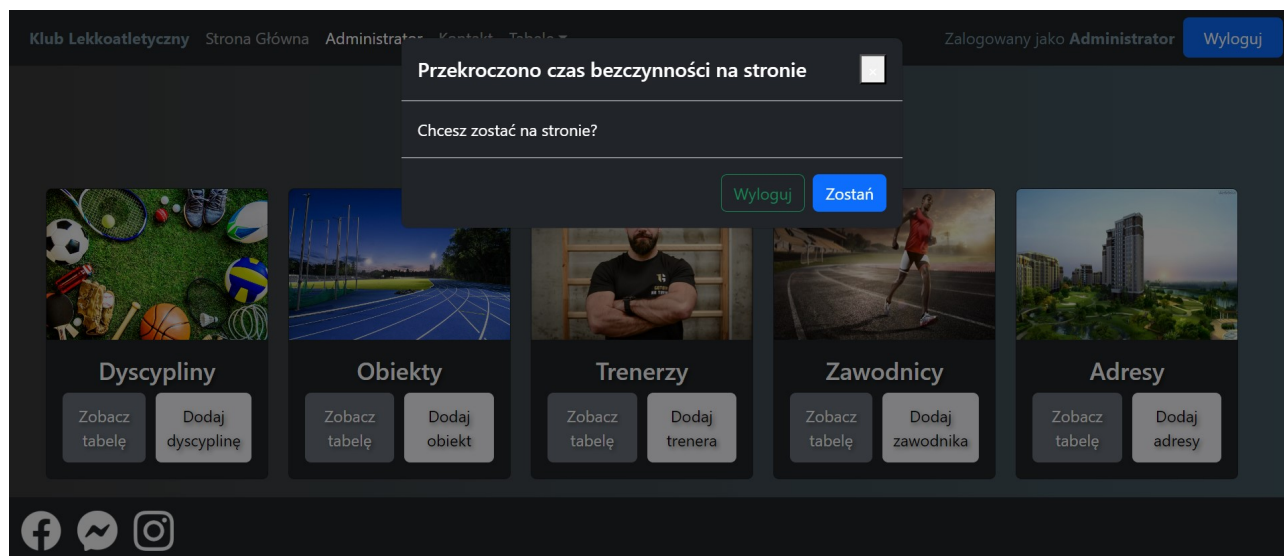
Rysunek 9. Błąd 403



Rysunek 10. Błąd ORA-02292

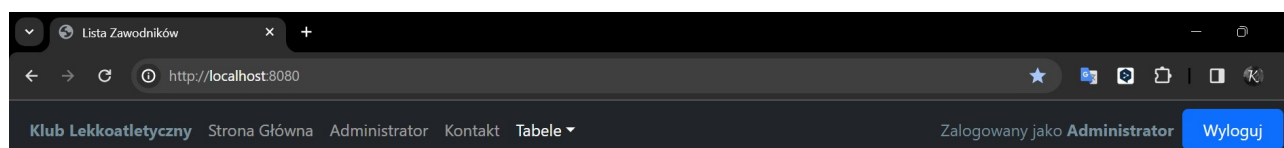
#### 4.7. Dodatkowe funkcje

Dodatkowo ustawiliśmy limit czasu przebywania nieaktywnym na stronie po zalogowaniu. Po 40 sekundach strona wyświetli komunikat o przekroczeniu czasu bezczynności pokazany na rysunku 11.



Rysunek 11.

Dbając o bezpieczeństwo strony ukryliśmy adres url podstrony na której się znajdujemy.



Rysunek 12. Ukryty url

## 5. Wnioski

Podczas pracy nad projektem zdobyliśmy cenne doświadczenie na temat tworzenia aplikacji współpracującej z relacyjną bazą danych. Ćwiczyliśmy projektowanie i implementację interfejsu użytkownika, wykorzystanie technologii takich jak Java, Spring Boot, Hibernate, Thymeleaf, Bootstrap i Maven, a także integrację z bazą danych. Opracowaliśmy trzy perspektywy użytkowników z różnymi poziomami uprawnień. Dodatkowo wprowadziliśmy funkcje takie jak limit czasu nieaktywności, obsługa błędów oraz ukrywanie adresów URL. Ten projekt dostarczył nam sporo praktycznego doświadczenia w zakresie projektowania aplikacji webowej w architekturze wielowarstwowej, umożliwiając nam rozwinięcie naszych umiejętności w obszarze tworzenia kompleksowych rozwiązań.



**Wydział Elektroniki  
i Technik Informatycznych**

**POLITECHNIKA WARSZAWSKA**