Politechnika Rzeszowska

Wydział Elektrotechniki i Informatyki

***Katedra Informatyki i Automatyki***

**Projekt**

*Najlepsza uczelnia*

*Funkcjonalność – usuwanie użytkownika   
z poziomu panelu administratora*

**Bazy Danych**

**Inżynieria oprogramowania**

**Wykonał:**

Mateusz Miś

nr indeksu 152175

II EF-DI

Grupa L09

Rzeszów 2018

Spis treści

[1. Założenia projektowe 3](#_Toc517695014)

[2. Implementacja kodu 3](#_Toc517695015)

[2.1. Widok 3](#_Toc517695016)

[2.2. Kontroler 4](#_Toc517695017)

[2.3. Serwis 4](#_Toc517695018)

[2.4. Repozytorium 5](#_Toc517695019)

[2.5. Model 5](#_Toc517695020)

[3. Elementy diagramów projektu wykorzystane w implementacji funkcjonalności 5](#_Toc517695021)

[3.1. Diagram związków encji 5](#_Toc517695022)

[3.2. Diagram hierarchii funkcji 5](#_Toc517695023)

[3.3. Diagram klas 6](#_Toc517695024)

[3.4. Diagram przypadków użycia 6](#_Toc517695025)

[4. Diagram przepływu danych 7](#_Toc517695026)

[5. Diagram sekwencji 8](#_Toc517695027)

[6. Podsumowanie i wnioski 8](#_Toc517695028)

# Założenia projektowe

Celem niniejszej części projektu było zaimplementowanie funkcjonalności systemu informatycznego umożliwiającej administratorowi tegoż systemu usunięcie zarejestrowanego już użytkownika z bazy danych wraz z wszelkimi danymi z nim powiązanych.

Operacja ta przeprowadzana jest z poziomu panelu administratora, w którym to przy każdym wyświetlonym na liście użytkowniku dodany zostaje przycisk umożliwiający usunięcie go   
z bazy danych. Jednakże przed wykonaniem tejże czynności zostaję również wyświetlone okienko z zapytaniem o potwierdzenie chęci wykonania operacji usunięcia.

# Implementacja kodu

Implementując funkcjonalności w systemie informatycznym należy pamiętać o dostosowaniu działania kodu do modelu warstwowego używanego w danym systemie. Dlatego też opisane zostaną tu metody odpowiednich warstw umożliwiające przejście przez każdą z nich w celu realizacji zadania usunięcia użytkownika z bazy danych.

## Widok

Omawianie poszczególnych warstw w niniejszym raporcie rozpoczęte zostanie od warstwy najbliższej użytkownikowi tj. widoku.

Listing 2.1. Fragment kodu zawartego w pliku userList.html odpowiedzialny za interakcję   
z użytkownikiem systemu.

|  |
| --- |
| 1 <**td**>  2 <**button** *type*="button" *class*="btn btn-danger" *data-toggle*="modal"  3 *th:attr*="data-target='#exampleModal' + ${currentUser.id}"  4 >Usuń</**button**>  5 <**div** *class*="modal fade" *th:attr*="id = 'exampleModal' + ${currentUser.id}"  6 *tabindex*="-1" *role*="dialog" *aria-labelledby*="exampleModalLabel"  7 *aria-hidden*="true">  8 <**div** *class*="modal-dialog" *role*="document">  9 <**div** *class*="modal-content">  10 <**div** *class*="modal-header">  11 <**h5** *class*="modal-title" *id*="exampleModalLabel">Potwierdzenie  12 operacji</**h5**>  13 <**button** *type*="button" *class*="close" *data-dismiss*="modal"  14 *aria-label*="Close">  15 <**span** *aria-hidden*="true">**&times;**</**span**>  16 </**button**>  17 </**div**>  18 <**div** *class*="modal-body">Czy na pewno chcesz usunąć wybranego  19 użytkownika?</**div**>  20 <**div** *class*="modal-footer">  21 <**button** *type*="button" *class*="btn btn-secondary"  22 *data-dismiss*="modal">Nie</**button**>  23 <**a** *th:href*="@{/admin/delete(userId=${currentUser.id})}"  24 *class*="btn btn-danger">Tak</**a**>  25 </**div**>  26 </**div**>  27 </**div**>  28 </**div**>  29 </**td**> |

Fragment kodu przedstawiony na listingu 2.1. stanowi część pliku userList.html. Odpowiada on za interakcję z użytkownikiem, któremu wyświetlany jest przycisk „USUŃ” (linie 2 - 4) przy każdym użytkowniku z listy. a następnie po jego wciśnięciu pojawia się okno   
z potwierdzeniem wykonania akcji (linie 8 - 27). W przypadku zatwierdzenia stworzone zostaje odpowiednie mapowanie strony (linie 23 - 24), przechwytywane przez warstwę kontrolera, któremu przekazywane jest również id użytkownika przeznaczonego do usunięcia.

## Kontroler

Zadaniem kontrolera jest obsługa akcji użytkownika wykonana w warstwie widoku. Implementacja kodu w tejże warstwie widoczna jest na listingu 2.2.

Listing 2.2. Fragment kodu zawartego w pliku AdminController.java odpowiedzialny za obsługę akcji użytkownika.

|  |
| --- |
| 1 @GetMapping(value**=**"/admin/delete")  2 **public** **String** adminUserDelete(@RequestParam(name**=**"userId")**Long** userId) {  3 userService**.**deleteUser(userId);  4 **return** "redirect:userList";  5 } |

W pierwszej linii listingu 2.2. zawarte jest przechwycenie mapowania strony od widoku, do którego przypisana zostanie metoda kontrolera *adminUserDelete(Long userId)*. Jedynym zadaniem kontrolera jest wywołanie odpowiedniej metody serwisu (linia 3) oraz odświeżenie widoku po dokonaniu operacji usuwania użytkownika (linia 4).

## Serwis

Zadaniem warstwy serwisu jest przygotowanie danych pobranych z bazy danych przed przesłaniem ich do wyższych warstw systemu. Jednakże w tym przypadku, jako iż żadne dane nie muszą być przetwarzane serwis wywołuje jedynie odpowiednią metodę warstwy repozytorium. Implementacja kodu w tejże warstwie widoczna jest na listingu 2.3.

Listing 2.3. Fragment kodu zawartego w pliku UserServiceImpl.java odpowiedzialny za komunikację z warstwą repozytorium.

|  |
| --- |
| 1 @**Override**  2 **public** **void** deleteUser(**Long** userId) {  3 userRepository**.**deleteById(userId);  4 } |

Jak wspomniane zostało powyżej jedynym zadaniem warstwy serwisu w trakcie realizacji funkcjonalności usuwania użytkownika jest wywołanie metody warstwy repozytorium *deleteById* (linia 3).

## Repozytorium

Zadaniem warstwy repozytorium jest komunikacja z bazą danych. W wykorzystywanej   
w niniejszym projekcie technologii Spring Data, poszczególne repozytoria   
(np. UserRepository) stanowią rozszerzenie JpaRepository, posiadające przygotowane podstawowe metody CRUD. Stanowi ono również rozszerzenie DAO (Data Access Object) dzięki czemu dokonuje komunikacji z bazą danych, umożliwiając również pisanie własnych zapytań w języku SQL.

JPA Repository potrafi interpretować podstawowe metody z rodziny CRUD na podstawie ich nazwy przykładowo użyta w niniejszej funkcjonalności metoda *deleteById* zostanie przekształcona w odpowiednie zapytanie SQL.

## Model

Na podstawie warstwy modelu budowana jest baza danych ponieważ jednak pomiędzy jej obiektami występują powiązania należy o nich pamiętać w momencie usuwania danych użytkownika. W celu zapewnienia usunięcia wszelkich danych powiązanych   
z użytkownikiem z bazy danych w momencie jego usunięcia należało uzupełnić model odpowiednią adnotacją : *@OnDelete(action = OnDeleteAction.CASCADE).*

# Elementy diagramów projektu wykorzystane w implementacji funkcjonalności

W tej części raportu zaprezentowane zostaną elementy wspólnych dla całego projektu diagramów wykorzystane podczas realizacji niniejszej funkcjonalności.

## Diagram związków encji

Elementy diagramu związków encji wykorzystane w realizacji funkcjonalności:

* UŻYTKOWNIK
* ODPOWIEDŹ\_UŻYTKOWNIKA
* UŻYTKOWNIK\_ROLA
* UŻYTKOWNIK\_KIERUNEK

Operacje przeprowadzane są głównie na encji UŻYTKOWNIK jednakże poprzez odpowiednio dobrane relacje między encjami oraz wykorzystanej adnotacji *@OnDelete* dodatkowe operacje przeprowadzane są na wszelkich encjach powiązanych relacjami   
z użytkownikiem.

## Diagram hierarchii funkcji

Niniejsz funkcjonalność usytuowana jest na diagramie hierarchii funkcji jako jedna   
z podfunkcji funkcji „ZARZĄDZAJ UŻYTKOWNIKAMI”. Pełna hierarchia tejże funkcji przedstawia się następująco

* PROPONUJ KIERUNKI DLA KANDYDATÓW
  + ZARZĄDAZJ UŻYTKOWNIKAMI
    - USUŃ UZYTKOWNIKA

## Diagram klas

Niestety ze względu na brak organizacji w grupie projektowej diagram klas został przygotowany w sposób niepoprawny i niewystarczający. W przypadku zaś poprawnego jego skonstruowania funkcjonalność usuwania użytkowników korzystała by z następujących jego elementów:

* w warstwie modelu:

Analogicznie do diagramu ERD należy skorzystać z klas zbudowanych na podstawie encji UŻYTKOWNIK tj. klasa User oraz encji będących z nim w relacji: UserAnwser (encja ODPOWIEDŹ\_UŻYTKOWNIKA) oraz encjami normalizującymi pomiędzy użytkownikiem a rolą i kierunkiem

* w warstwie repozytorium:

Z klasy UserRepository będącej rozszerzeniem klasy JpaRepository.

* w warstwie serwisu:

Z klasy UserServiceImpl będącej implementacją interface’u UserService.

* w warstwie kontrolera:

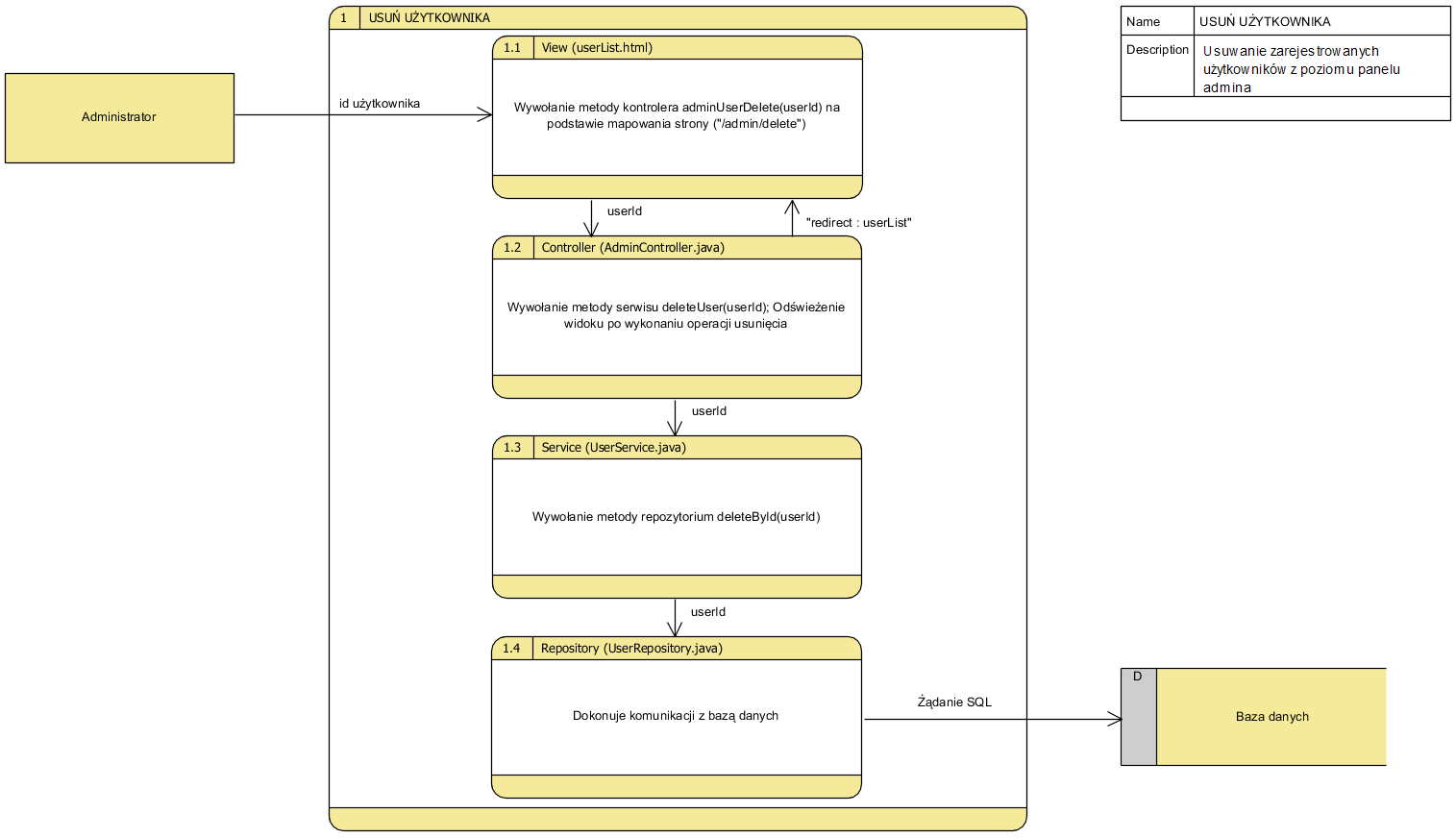
Z klasy AdminController

## Diagram przypadków użycia

W przypadku diagramu przypadków użycia sytuacja jest analogiczna tej zaistniałem   
w diagramie hierarchii funkcji. Funkcjonalność „Usuń użytkownika” stanowi rozszerzenie przypadku „Zarządzaj użytkownikami”. Jest to relacja typu <<Extend>> ponieważ operacja usuwania nie jest wymagana w trakcie realizacji funkcji zarządzania użytkownikami. Ponieważ operacja ta jest bezpośrednio połączona z aktorem administratora nie potrzebny jest żaden dodatkowy przypadek użycia pośredniczący w procesie usuwania. Administrator bowiem podaje jedynie id użytkownika, którego chciałby usunąć.

# Diagram przepływu danych

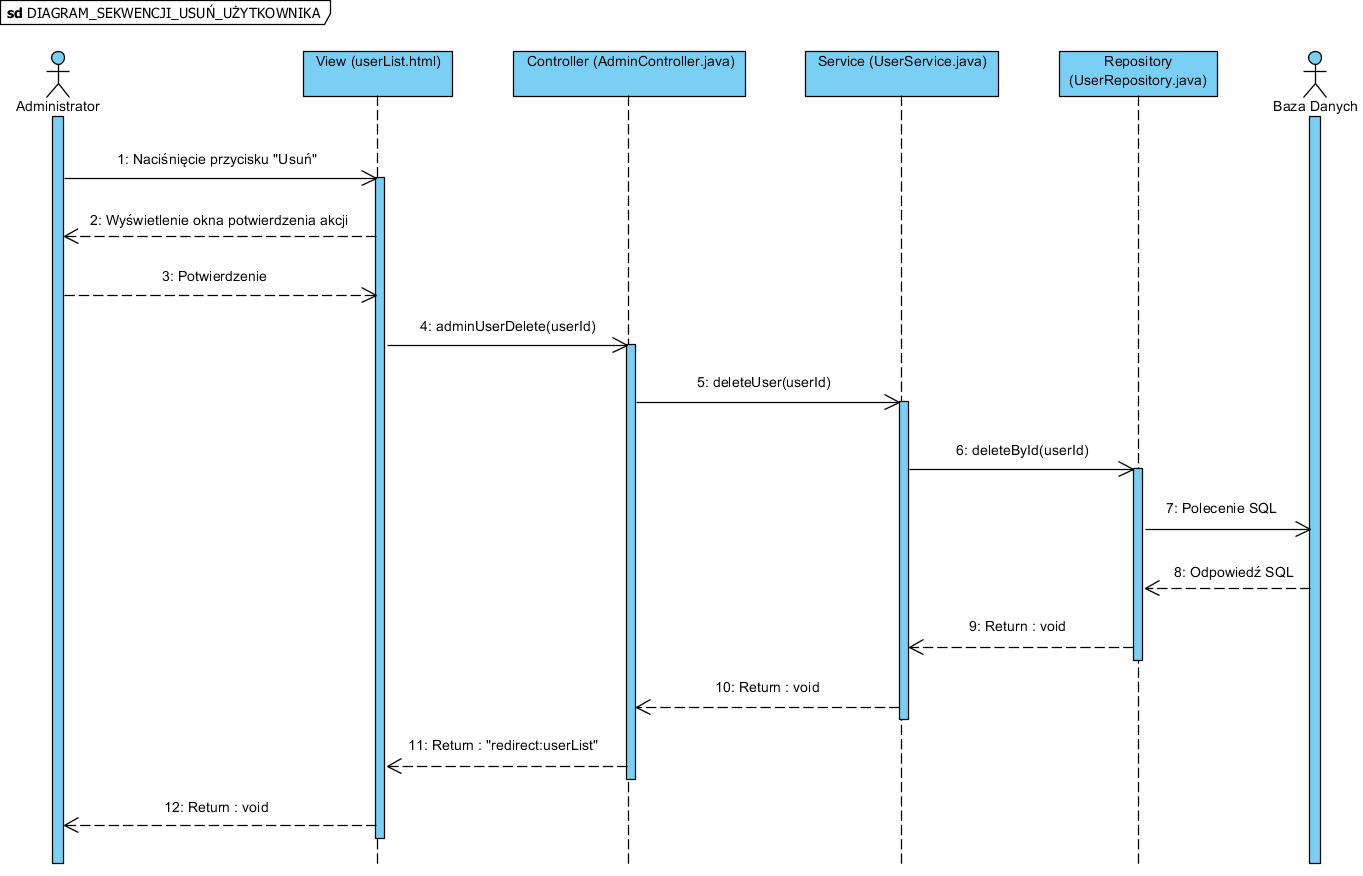
Diagram przepływu danych pokazuje jakie informacje przekazywane są pomiędzy poszczególnymi warstwami systemu informatycznego. Diagram sporządzony dla omawianej w niniejszym raporcie funkcjonalności znajduje się na rysunku 4.1.



Rys. 4.1. Diagram przepływu danych (DFD)

# Diagram sekwencji

Diagram sekwencji w najbardziej przejrzysty sposób pokazuje kolejność operacji wykonywanych w systemie informatycznym po dokonaniu odpowiedniej interakcji użytkownika w warstwie widoku. Diagram sporządzony dla omawianej w niniejszym raporcie funkcjonalności widoczny jest na rysunku 5.1.



Rys. 5.1. Diagram sekwencji

# Podsumowanie i wnioski

Usuwanie danych użytkowników z bazy danych nie jest powszechnie stosowanym rozwiązaniem. Może bowiem w przyszłości zajść potrzeba skorzystania z informacji na temat wszystkich dotychczasowych użytkowników systemu, które byłyby niepełne w przypadku usunięcia niektórych a nich. Znacznie lepszym rozwiązaniem byłoby stworzenie dodatkowej encji przechowującej informacje na temat nieaktywnych użytkowników. To rozwiązanie jest jednak bardziej skomplikowane i wymaga powiększenia aktualnie wykorzystywanej bazy danych. Ponieważ jednak niniejszy projekt nie będzie wykorzystywany w celach komercyjnych, a jedynie zaliczeniowych prostsze rozwiązanie problemu jest wystarczające.

Permanentne usuwanie danych użytkowników niesie ze sobą jeszcze jedną konsekwencje. Mianowicie w przypadku popełnienia przez administratora błędu nie możliwym jest przywrócenie usuniętych danych. Użytkownik będzie zmuszony do stworzenia nowego konta.