Projekt Zespołowy Inżynierii Oprogramowania

System Rejestracji Studentów na Kursy

Artur Sojka[1], Michał Krzempek[2], Piotr Waluszek[3], Kacper Machnik[4], Tomasz Madeja[5], Patryk Przybysz[6]

[1]Numer indeksu: 415075, E-mail: asojka@student.agh.edu.pl
[2]Numer indeksu: 417591, E-mail: krzempekm@student.agh.edu.pl
[3]Numer indeksu: 415044, E-mail: waluszekp@student.agh.edu.pl
[4]Numer indeksu: 417710, E-mail: kmachnik@student.agh.edu.pl
[5]Numer indeksu: 416288, E-mail: tomaszmadeja@student.agh.edu.pl
[6]Numer indeksu: 410705, E-mail: pprzybysz@student.agh.edu.pl

Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie
Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej

28 kwietnia 2025

Spis treści

1 Opis Projektu

Celem projektu jest stworzenie systemu obsługującego zapisy na zajęcia na wyższych uczelniach. Ma on ułatwić zarządzanie kursami, grupami zajęciowymi, kontami studentów oraz ich zapisami na zajęcia. Pracownik dziekanatu będzie miał możliwość zarządzania treścią systemu, w tym dodawania i usuwania kursów oraz grup zajęciowych, a także administracyjnego zapisywania i wypisywania studentów. Każdy kurs będzie identyfikowany przez nazwę, kod, podczas gdy grupy zajęciowe będą dodatkowo charakteryzować się numerem grupy, nazwiskiem prowadzącego, liczbą miejsc oraz salą. Student, jako klient systemu, będzie mógł między innymi przeglądać dostępne grupy, dokonywać zapisów do nich, o ile są dostępne miejsca.

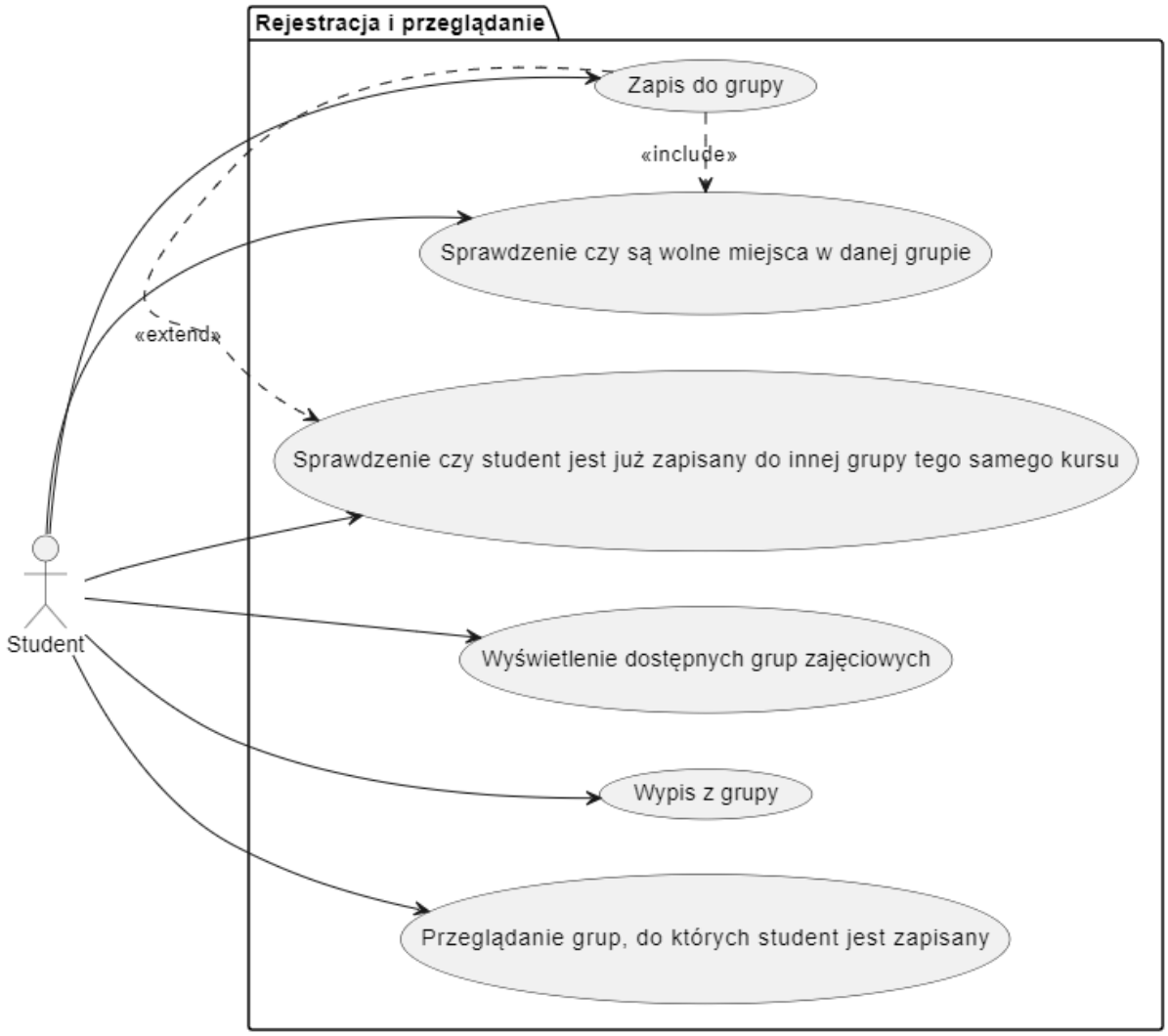
2 Analiza wymagań funkcjonalnych

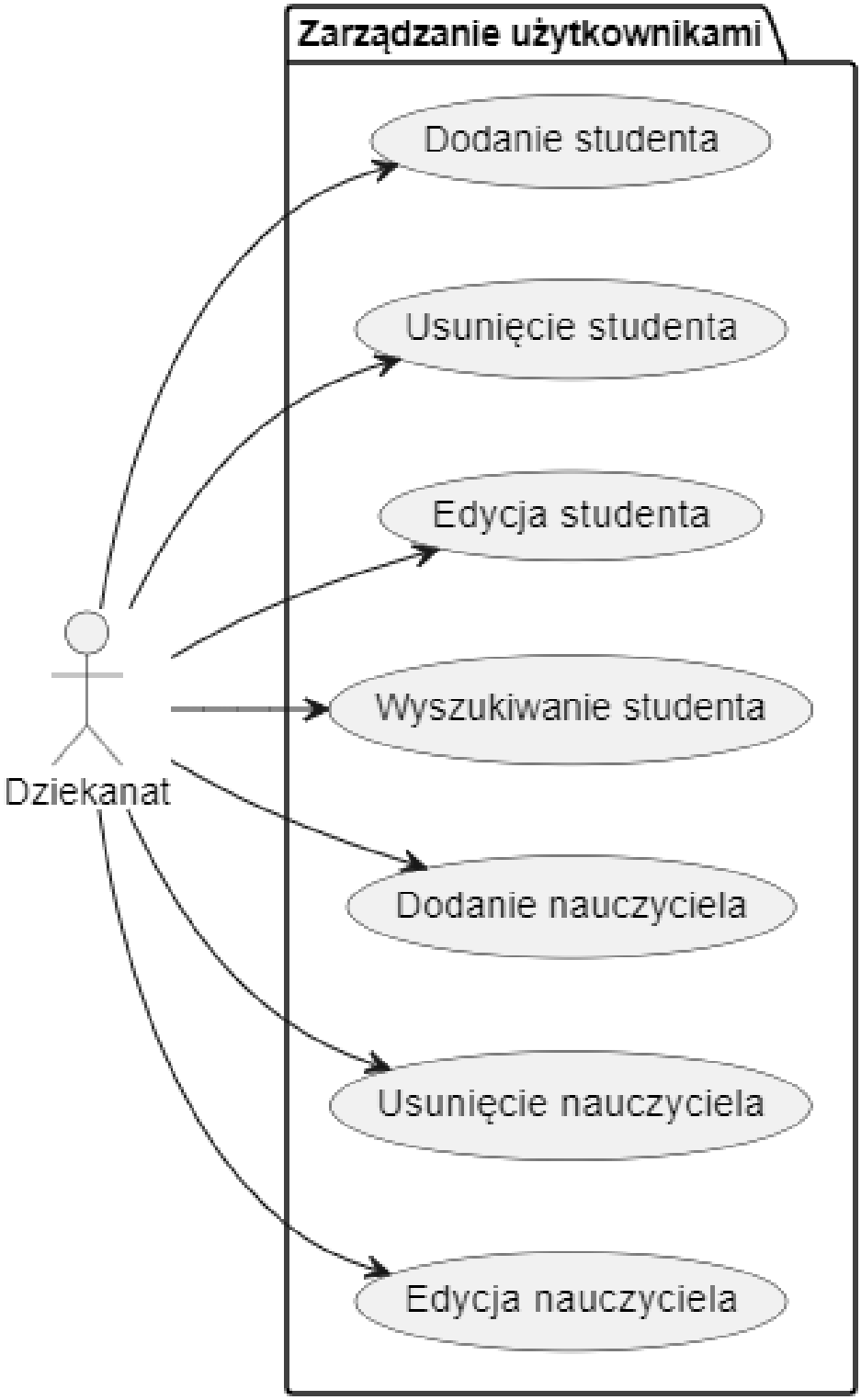
2.1 Definicja aktorów

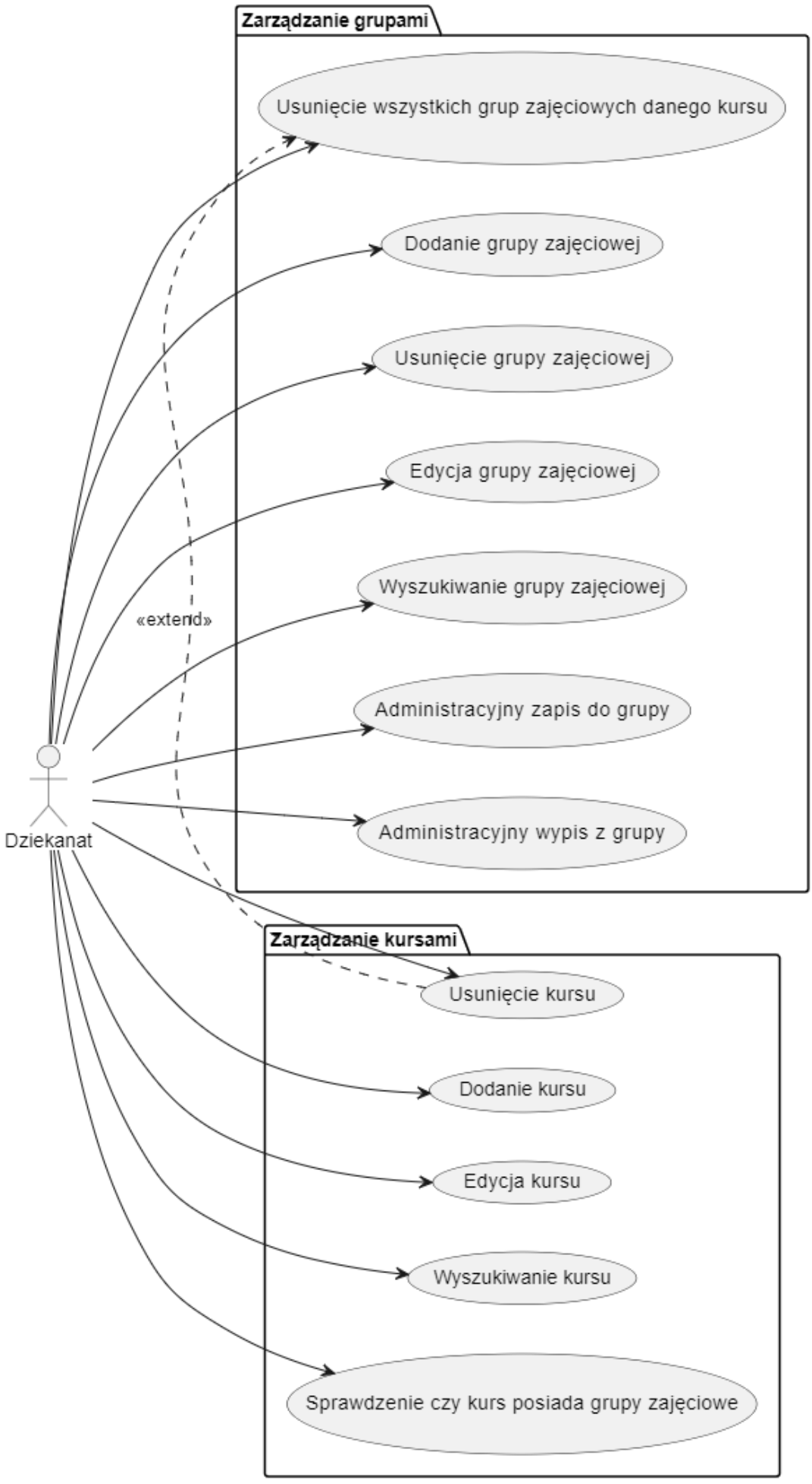
Rola	Opis	Przypadki użycia
Student	Student to podstawowy użytkownik systemu, który korzysta z funkcjonalności związanych z zarządzaniem swoją ścieżką edukacyjną. Do głównych aktywności studenta należą zapisywanie się na kursy i grupy zajęciowe.	• Zapis do grupy powiązany poprzez «include» z: • Sprawdzenie czy są wolne miejsca w danej grupie powiązany poprzez «extend» z: • Sprawdzenie czy student jest już zapisany do innej grupy tego samego kursu • Wyświetlenie dostępnych grup zajęciowych • Wypis z grupy • Przeglądanie grup, do których student jest zapisany
Nauczyciel	Nauczyciel to użytkownik systemu odpowiedzialny za dydaktyczną część procesu kształcenia. Nauczyciele tworzą materiały kursowe, wystawiają oceny oraz monitorują obecność studentów na zajęciach. Są również odpowiedzialni za komunikację z uczniami w ramach prowadzonych przez siebie kursów.	• Dodanie oceny dla studenta • Edycja oceny dla studenta • Usunięcie oceny dla studenta • Przeglądanie obecności na zajęciach • Przeglądanie kursów prowadzonych przez nauczyciela

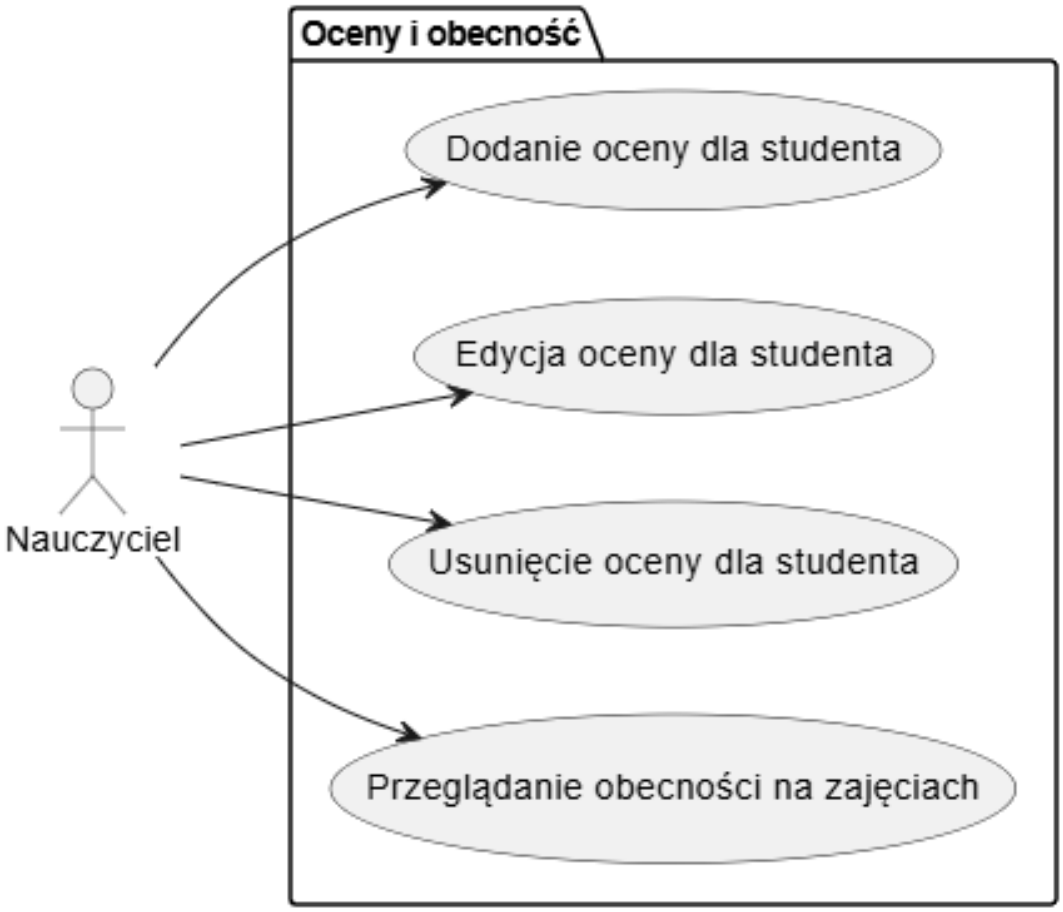
Dziekanat	Dziekanat reprezentuje administracyjną stronę uczelni, zarządzając kluczowymi aspektami systemu edukacyjnego. Do zadań dziekanatu należy administrowanie danymi studentów i nauczycieli, zarządzanie kursami, grupami zajęciowymi.	• Dodanie studenta • Usunięcie studenta • Edycja danych studenta • Dodanie nauczyciela • Usunięcie nauczyciela • Edycja danych nauczyciela • Wyszukiwanie studenta • Wyszukiwanie nauczyciela • Dodanie grupy zajęciowej • Usunięcie grupy zajęciowej • Edycja grupy zajęciowej • Sprawdzenie czy kurs posiada grupy zajęciowe • Dodanie kursu • Usunięcie kursu powiązany poprzez «extend» z: • Usunięcie wszystkich grup zajęciowych danego kursu • Edycja kursu • Wyszukiwanie kursu • Administracyjny zapis do grupy • Administracyjny wypis z grupy

2.2 Diagram przypadków użycia




Zarządzanie użytkownikami
Dodanie studenta
Usunięcie studenta
Edycja studenta
Wyszukiwanie studenta
Dodanie nauczyciela
Usunięcie nauczyciela
Edycja nauczyciela
Dziekanat


Zarządzanie grupami
Usunięcie wszystkich grup zajęciowych danego kursu
Dodanie grupy zajęciowej
Usunięcie grupy zajęciowej
Edycja grupy zajęciowej
Wyszukiwanie grupy zajęciowej
Administracyjny zapis do grupy
Administracyjny wypis z grupy
«extend»
Dziekanat
Zarządzanie kursami
Usunięcie kursu
Dodanie kursu
Edycja kursu
Wyszukiwanie kursu
Sprawdzenie czy kurs posiada grupy zajęciowe


Oceny i obecność
Dodanie oceny dla studenta
Edycja oceny dla studenta
Usunięcie oceny dla studenta
Przeglądanie obecności na zajęciach
Nauczyciel

2.3 Przypadki użycia

2.3.1 Wyświetlenie dostępnych grup zajęciowych

Umożliwia studentom przeglądanie dostępnych grup zajęciowych.

Nazwa przypadku	**Wyświetlenie dostępnych grup zajęciowych**
Krótki opis	Umożliwia studentom przeglądanie dostępnych grup zajęciowych.
Aktor biorący udział	Student
Warunki początkowe	Brak
Warunki końcowe	Wyświetlenie listy dostępnych grup zajęciowych
Główny ciąg zdarzeń	1. Student loguje się do systemu. 2. Student wybiera opcję przeglądania grup zajęciowych. 3. System pobiera listę dostępnych grup zajęciowych z bazy danych. 4. System wyświetla listę grup zajęciowych użytkownikowi.
Alternatywne ciągi zdarzeń	Brak

2.3.2 Zapis do grupy

Pozwala studentowi na zapisanie się do wybranej grupy zajęciowej.

Nazwa przypadku	**Zapis do grupy**
Krótki opis	Pozwala studentowi na zapisanie się do wybranej grupy zajęciowej.
Aktor biorący udział	Student
Warunki początkowe	Wyświetlenie listy dostępnych grup zajęciowych
Warunki końcowe	Potwierdzenie zapisu studenta do wybranej grupy
Główny ciąg zdarzeń	1. Student wybiera grupę zajęciową, do której chce się zapisać. 2. System sprawdza dostępność miejsc w wybranej grupie. 3. Jeśli są dostępne miejsca, sprawdzenie czy student jest już zapisany do innej grupy tego samego kursu. 4. Jeżeli nie jest, to dodaje studenta do grupy. 5. System aktualizuje listę uczestników grupy w bazie danych.
Alternatywne ciągi zdarzeń	1a. Brak dostępnych miejsc w wybranej grupie. 1b. System informuje studenta o braku dostępnych miejsc. 1c. Student może wybrać inną dostępną grupę lub zrezygnować z zapisu. 2a. Student jest już zapisany do innej grupy tego kursy. 2b. System informuje studenta o uczestnictwie do innej grupy. 2c. Student może zmienić grupę lub zrezygnować z zapisu.

Diagram 1: Zapis studenta do grupy

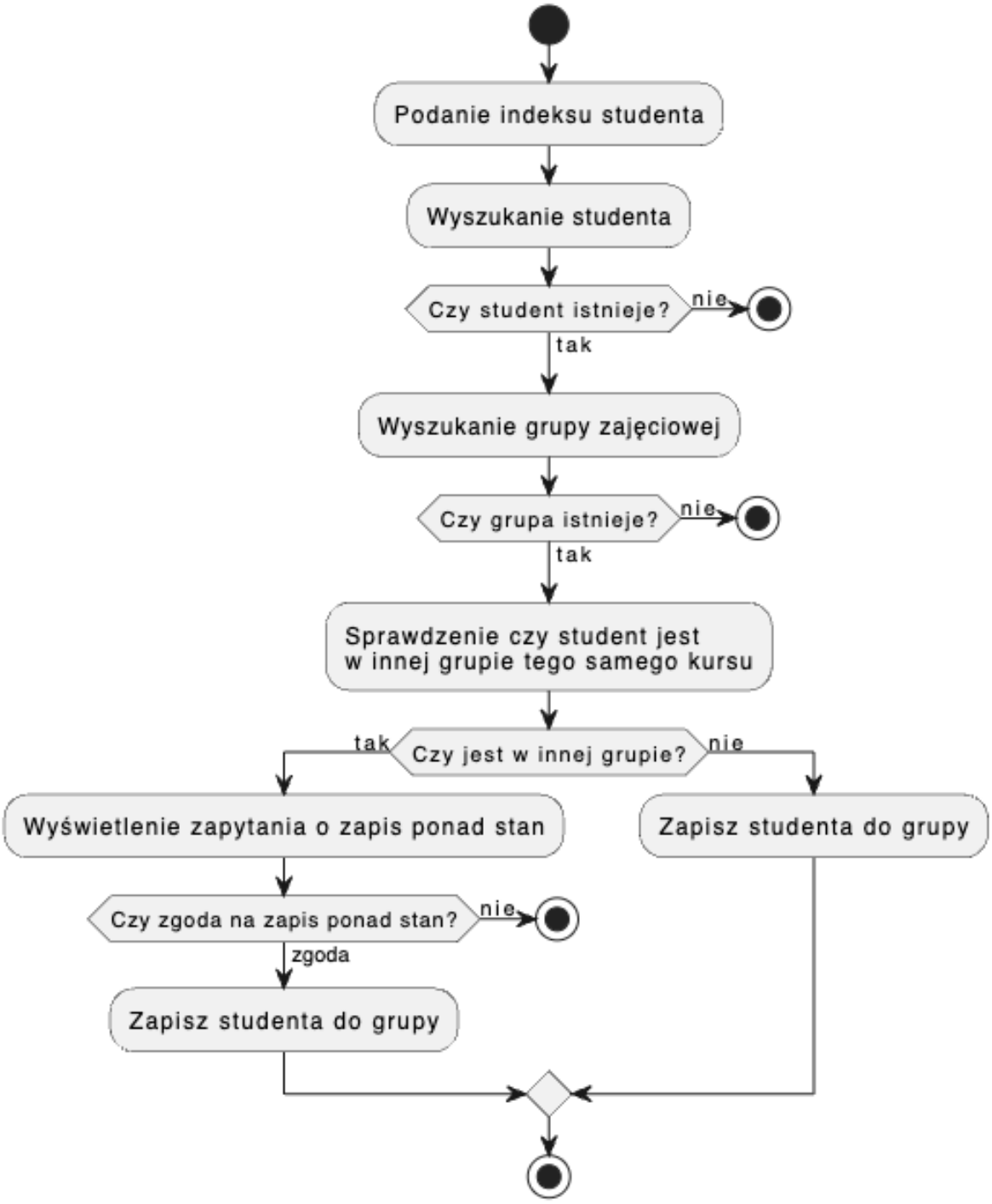
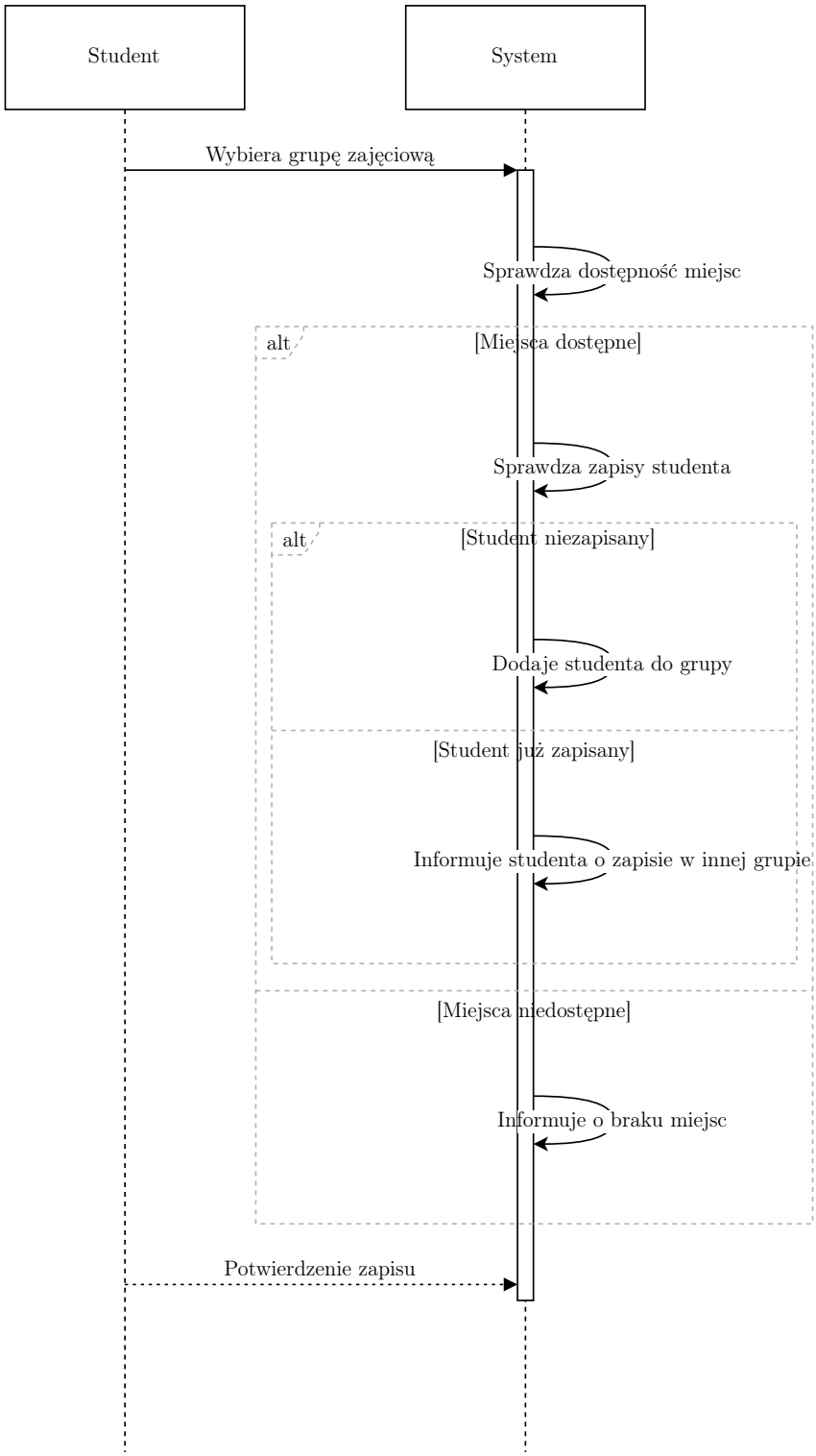


Diagram 2: Zapis do grupy

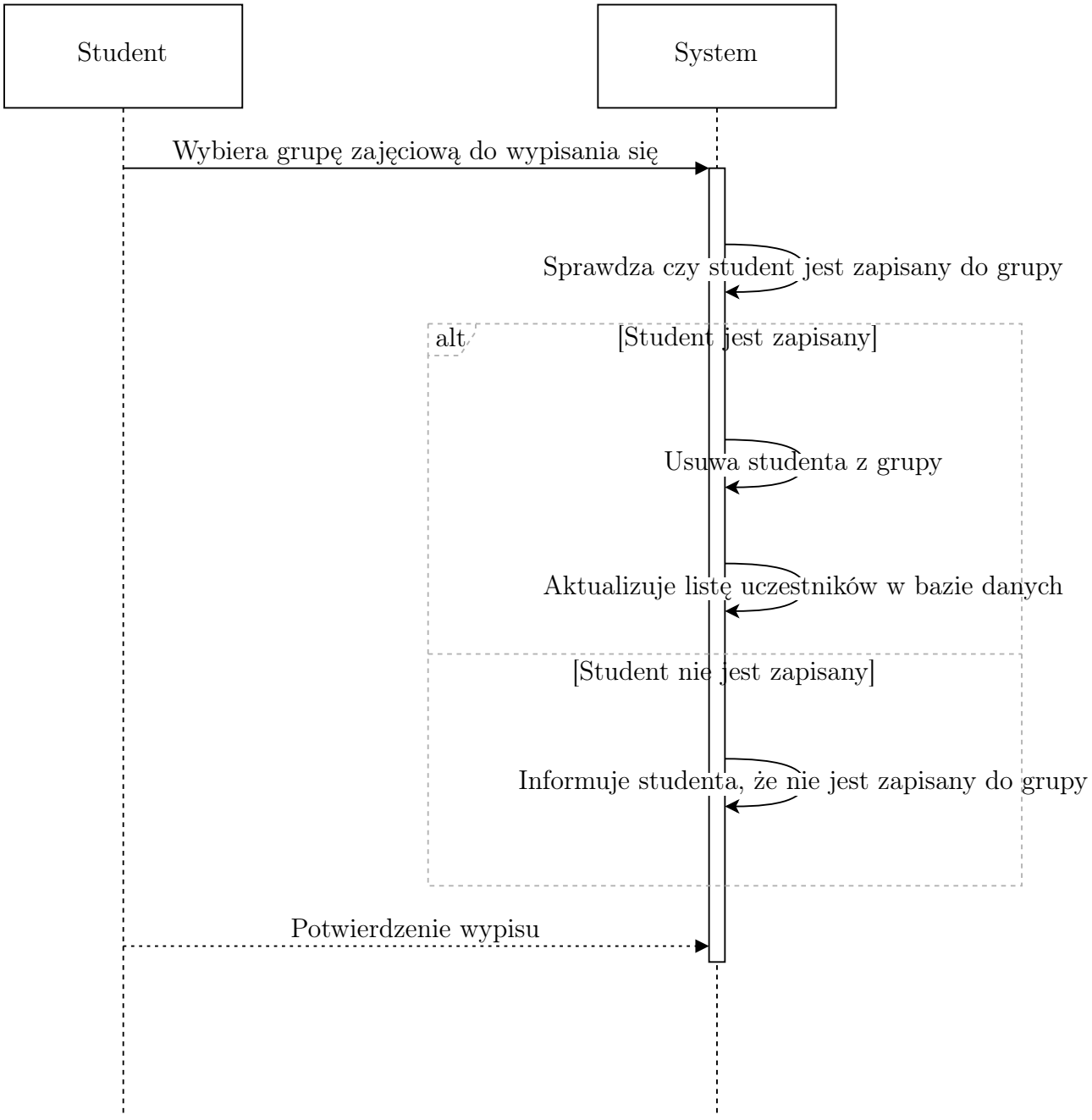


2.3.3 Wypis z grupy

Umożliwia studentowi wypisanie się z grupy zajęciowej.

Nazwa przypadku	**Wypis z grupy**
Krótki opis	Umożliwia studentowi wypisanie się z grupy zajęciowej.
Aktor biorący udział	Student
Warunki początkowe	Zapis studenta do wybranej grupy zajęciowej
Warunki końcowe	Potwierdzenie wypisu studenta z grupy
Główny ciąg zdarzeń	1. Student wybiera grupę zajęciową, z której chce się wypisać. 2. System sprawdza, czy student jest zapisany do wybranej grupy. 3. Jeśli student jest zapisany do grupy, system usuwa go z listy uczestników. 4. System aktualizuje listę uczestników grupy w bazie danych.
Alternatywne ciągi zdarzeń	1a. Student nie jest zapisany do wybranej grupy. 1b. System informuje studenta, że nie jest zapisany do grupy i kończy proces wypisu.

Diagram 3: Wypis z Grupy



2.3.4 Przeglądanie grup, do których student jest zapisany

Umożliwia studentom sprawdzenie, do których grup są już zapisani.

Nazwa przypadku	**Przeglądanie grup, do których student jest zapisany**
Krótki opis	Umożliwia studentom sprawdzenie, do których grup są już zapisani.
Aktor biorący udział	Student
Warunki początkowe	Zapisy studenta do różnych grup zajęciowych
Warunki końcowe	Wyświetlenie listy grup, do których student jest zapisany
Główny ciąg zdarzeń	1. Student loguje się do systemu. 2. Student wybiera opcję przeglądania swoich zapisów. 3. System pobiera listę grup, do których student jest zapisany, z bazy danych. 4. System wyświetla listę grup użytkownikowi.
Alternatywne ciągi zdarzeń	Brak

2.3.5 Sprawdzenie czy student jest już zapisany do innej grupy tego samego kursu

Sprawdza, czy student nie jest już zapisany do innej grupy tego samego kursu.

Nazwa przypadku	**Sprawdzenie czy student jest już zapisany do innej grupy tego samego kursu**
Krótki opis	Sprawdza, czy student nie jest już zapisany do innej grupy tego samego kursu.
Aktor biorący udział	Student
Warunki początkowe	Zapisy studenta do różnych grup zajęciowych
Warunki końcowe	Informacja czy student jest zapisany do innej grupy tego samego kursu
Główny ciąg zdarzeń	1. Student wybiera kurs, do którego chce się zapisać. 2. System sprawdza, czy student jest już zapisany do innej grupy tego samego kursu. 3. System informuje studenta, czy jest już zapisany do innej grupy tego samego kursu.
Alternatywne ciągi zdarzeń	Brak

2.3.6 Sprawdzenie czy są wolne miejsca w danej grupie

Umożliwia sprawdzenie dostępności miejsc w grupie zajęciowej.

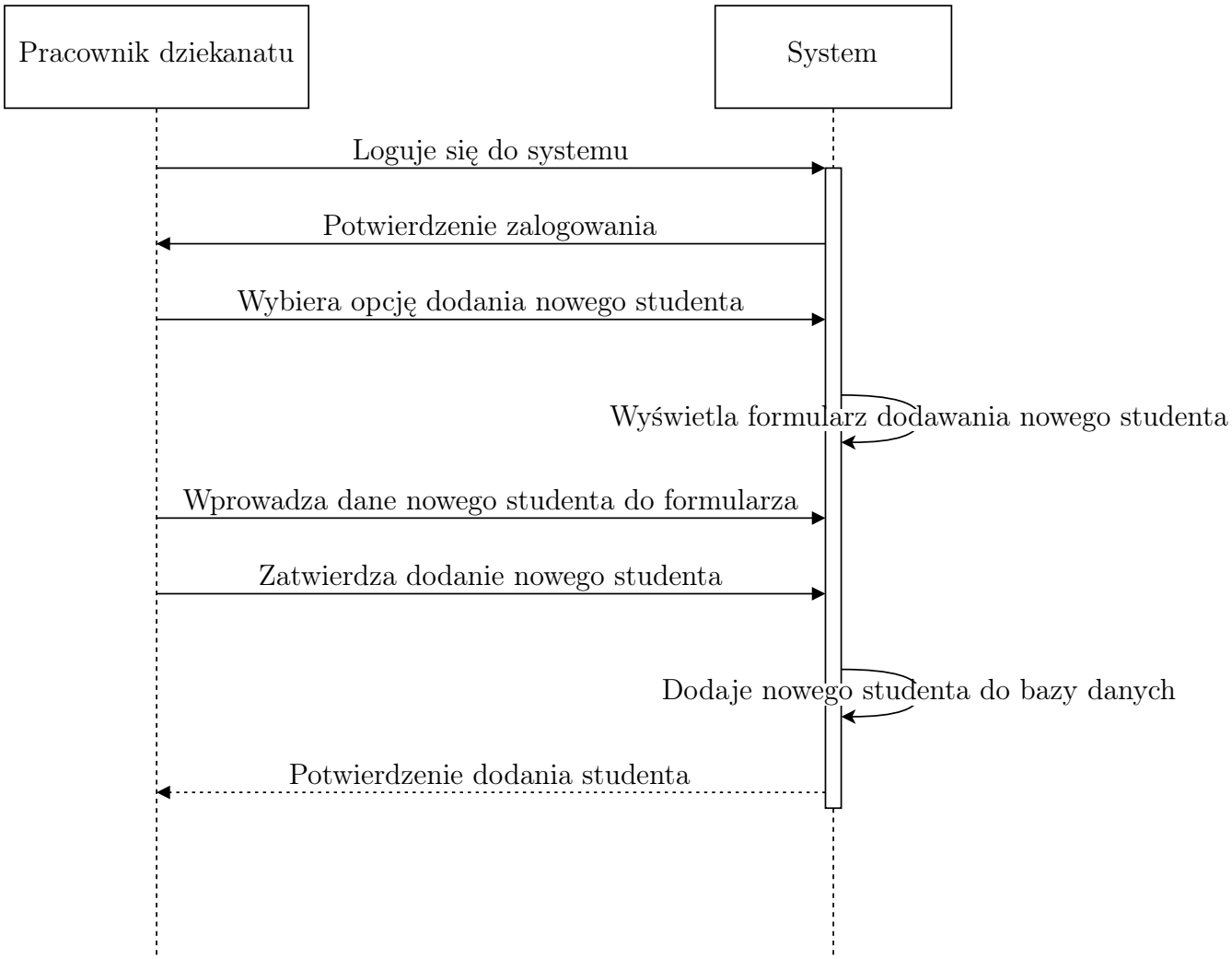
Nazwa przypadku	**Sprawdzenie czy są wolne miejsca w danej grupie zajęciowej**
Krótki opis	Umożliwia sprawdzenie dostępności miejsc w grupie zajęciowej.
Aktor biorący udział	Student
Warunki początkowe	Wybór grupy zajęciowej
Warunki końcowe	Informacja o dostępności miejsc w grupie
Główny ciąg zdarzeń	1. Student wybiera grupę zajęciową, dla której chce sprawdzić dostępność miejsc. 2. System sprawdza, czy w wybranej grupie są wolne miejsca. 3. System informuje studenta o dostępności miejsc w grupie.
Alternatywne ciągi zdarzeń	Brak

2.3.7 Dodanie studenta

Umożliwia dodanie nowego studenta do systemu.

Nazwa przypadku	**Dodanie studenta**
Krótki opis	Umożliwia dodanie nowego studenta do systemu.
Aktor biorący udział	Dziekanat
Warunki początkowe	Brak
Warunki końcowe	Dodanie nowego studenta do systemu
Główny ciąg zdarzeń	1. Pracownik dziekanatu loguje się do systemu. 2. Pracownik dziekanatu wybiera opcję dodania nowego studenta. 3. System wyświetla formularz dodawania nowego studenta. 4. Pracownik dziekanatu wprowadza dane nowego studenta do formularza. 5. Pracownik dziekanatu zatwierdza dodanie nowego studenta. 6. System dodaje nowego studenta do bazy danych.
Alternatywne ciągi zdarzeń	1a. Pracownik dziekanatu anuluje operację dodawania nowego studenta. 1b. System przechodzi do stanu początkowego, bez dodawania nowego studenta.

Diagram 4: Dodanie studenta

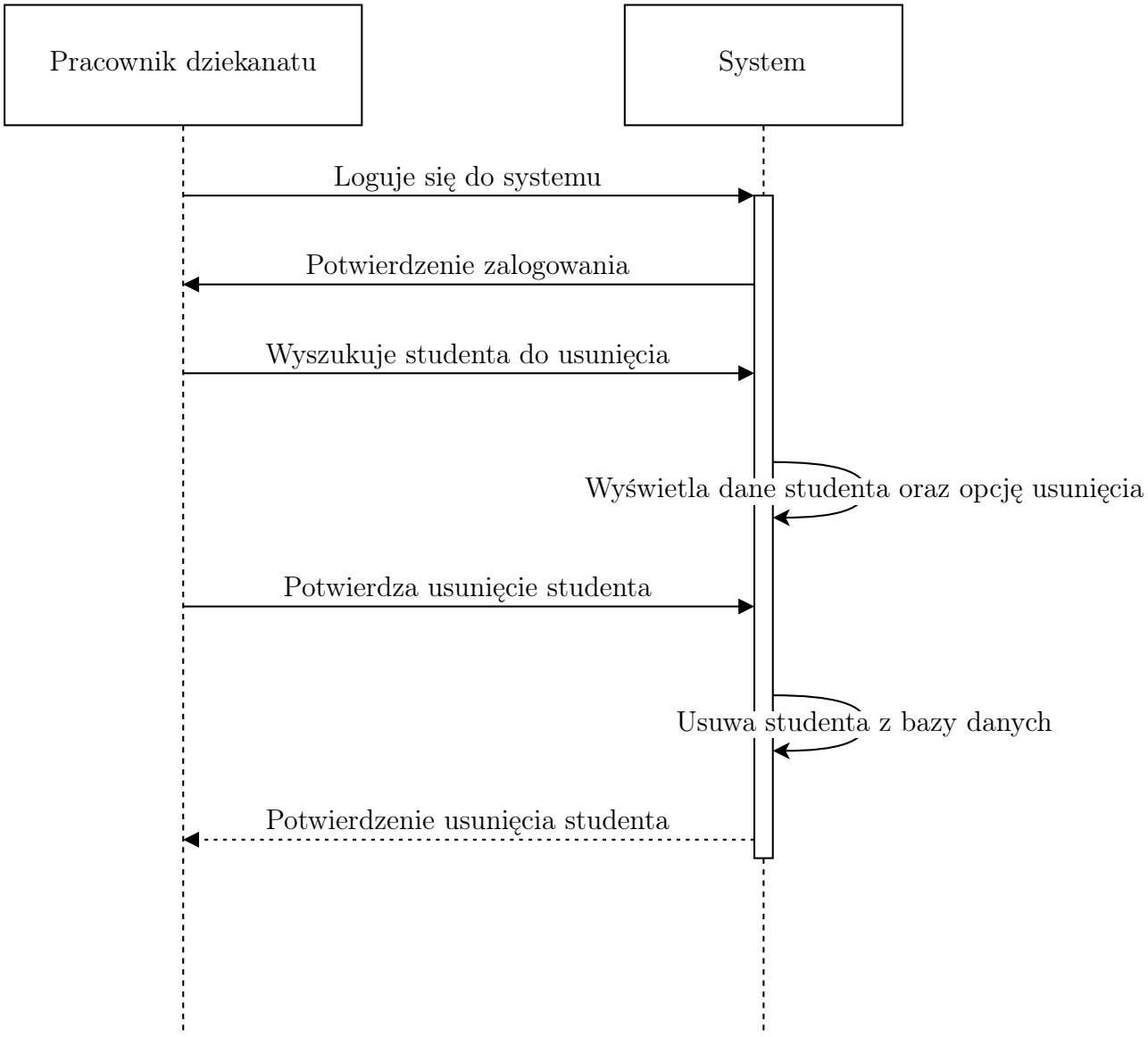


2.3.8 Usunięcie studenta

Umożliwia usunięcie studenta z systemu.

Nazwa przypadku	**Usunięcie studenta**
Krótki opis	Umożliwia usunięcie studenta z systemu.
Aktor biorący udział	Dziekanat
Warunki początkowe	Istnienie studenta w systemie
Warunki końcowe	Usunięcie studenta z systemu
Główny ciąg zdarzeń	1. Pracownik dziekanatu loguje się do systemu. 2. Pracownik dziekanatu wyszukuje studenta, którego chce usunąć. 3. System wyświetla dane studenta oraz opcję usunięcia. 4. Pracownik dziekanatu potwierdza usunięcie studenta. 5. System usuwa studenta z bazy danych.
Alternatywne ciągi zdarzeń	1a. Pracownik dziekanatu anuluje operację usunięcia studenta. 1b. System przechodzi do stanu początkowego, bez usuwania studenta.

Diagram 5: Usunięcie studenta

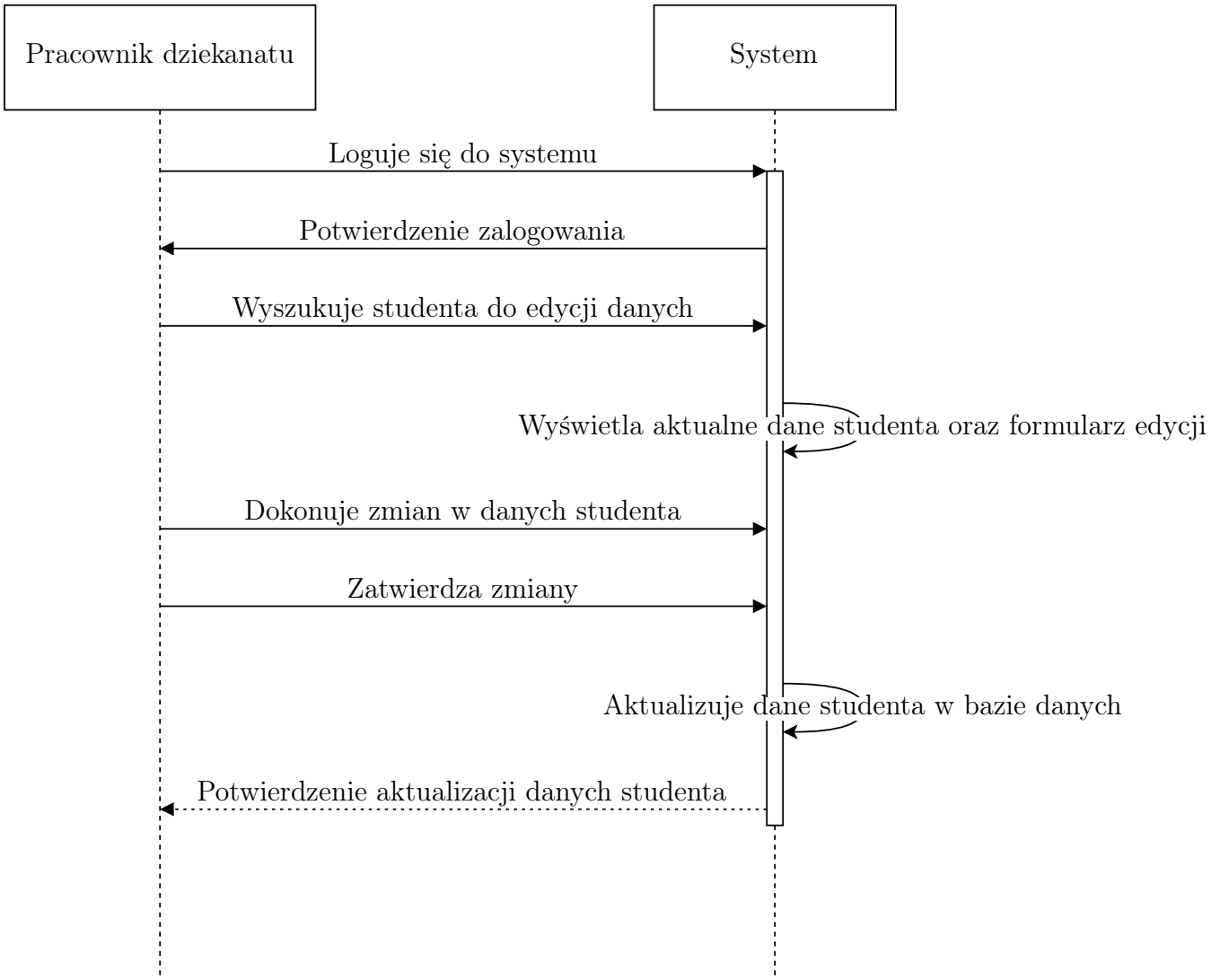


2.3.9 Edycja danych studenta

Pozwala na zmianę danych studenta w systemie.

Nazwa przypadku	**Edycja danych studenta**
Krótki opis	Pozwala na zmianę danych studenta w systemie.
Aktor biorący udział	Dziekanat
Warunki początkowe	Istnienie studenta w systemie
Warunki końcowe	Zmiana danych studenta w systemie
Główny ciąg zdarzeń	1. Pracownik dziekanatu loguje się do systemu. 2. Pracownik dziekanatu wyszukuje studenta, którego dane chce zmienić. 3. System wyświetla aktualne dane studenta oraz formularz edycji. 4. Pracownik dziekanatu dokonuje zmian w danych studenta. 5. Pracownik dziekanatu zatwierdza zmiany. 6. System aktualizuje dane studenta w bazie danych.
Alternatywne ciągi zdarzeń	1a. Pracownik dziekanatu anuluje operację edycji danych studenta. 1b. System przechodzi do stanu początkowego, bez dokonywania zmian w danych studenta.

Diagram 6: Edycja danych studenta

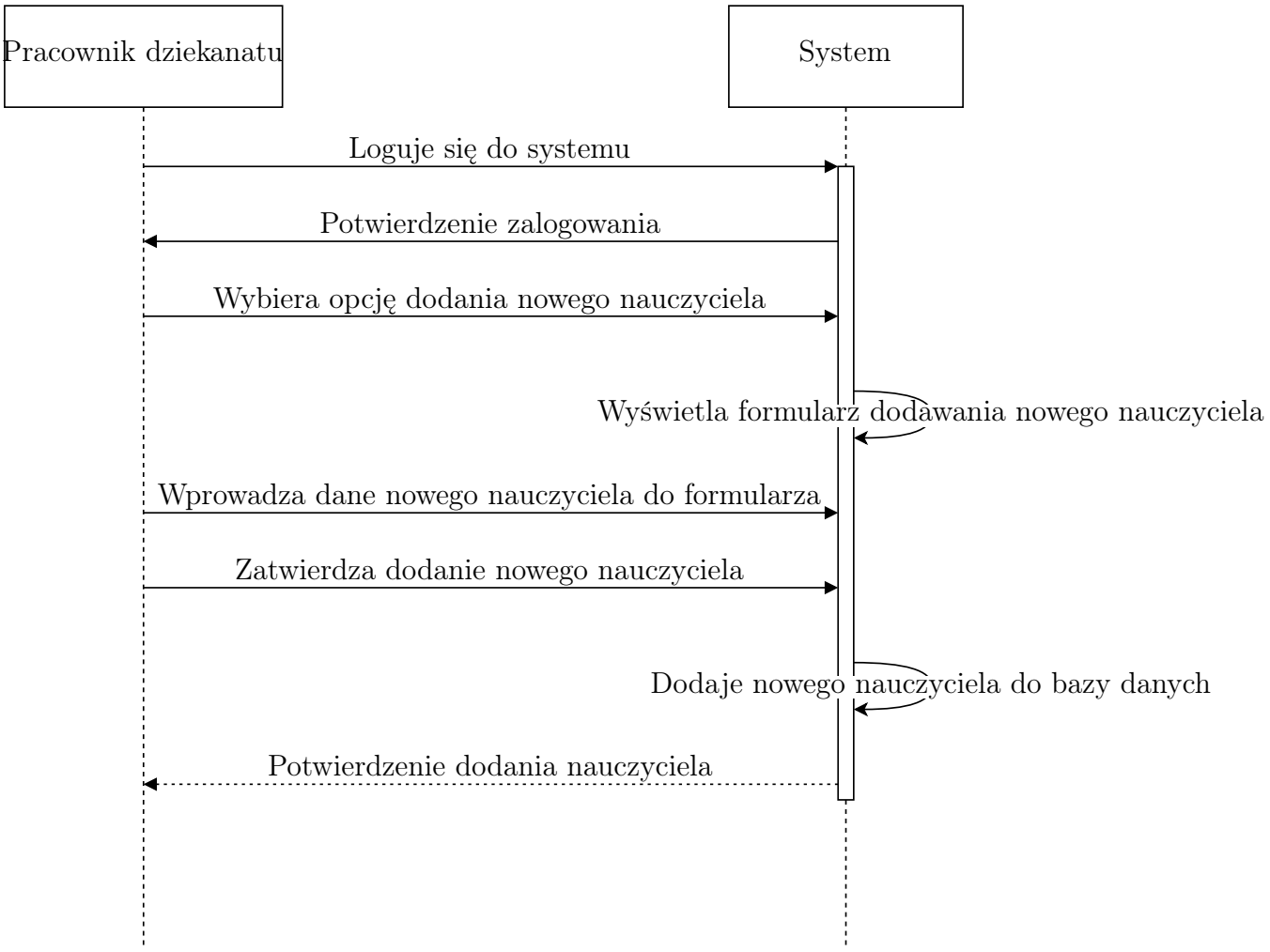


2.3.10 Dodanie nauczyciela

Umożliwia dodanie nowego nauczyciela do systemu.

Nazwa przypadku	**Dodanie nauczyciela**
Krótki opis	Umożliwia dodanie nowego nauczyciela do systemu.
Aktor biorący udział	Dziekanat
Warunki początkowe	Brak
Warunki końcowe	Dodanie nowego nauczyciela do systemu
Główny ciąg zdarzeń	1. Pracownik dziekanatu loguje się do systemu. 2. Pracownik dziekanatu wybiera opcję dodania nowego nauczyciela. 3. System wyświetla formularz dodawania nowego nauczyciela. 4. Pracownik dziekanatu wprowadza dane nowego nauczyciela do formularza. 5. Pracownik dziekanatu zatwierdza dodanie nowego nauczyciela. 6. System dodaje nowego nauczyciela do bazy danych.
Alternatywne ciągi zdarzeń	1a. Pracownik dziekanatu anuluje operację dodania nowego nauczyciela. 1b. System przechodzi do stanu początkowego, bez dodawania nowego nauczyciela.

Diagram 7: Dodanie nauczyciela

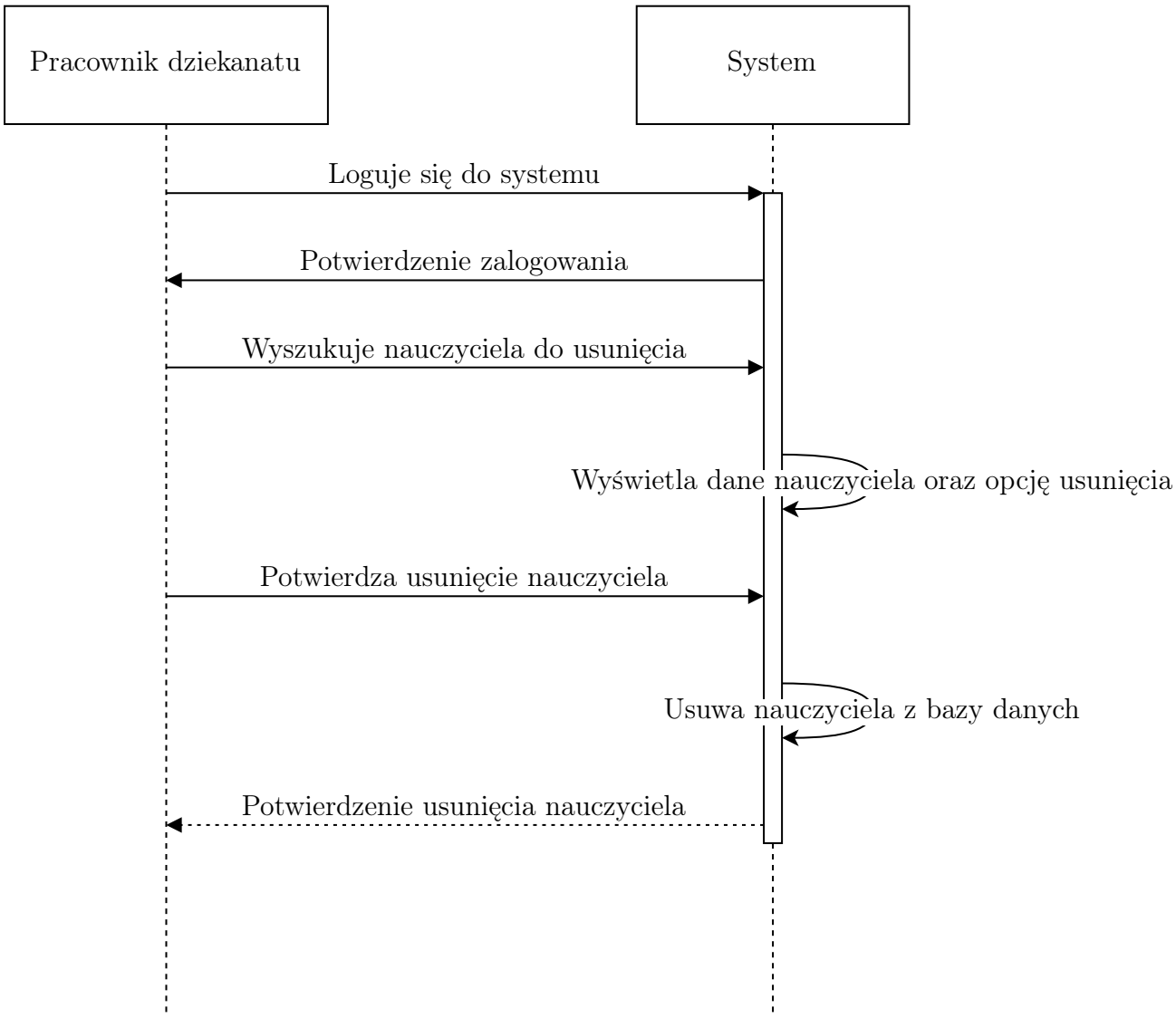


2.3.11 Usunięcie nauczyciela

Umożliwia usunięcie nauczyciela z systemu.

Nazwa przypadku	**Usunięcie nauczyciela**
Krótki opis	Umożliwia usunięcie nauczyciela z systemu.
Aktor biorący udział	Dziekanat
Warunki początkowe	Istnienie nauczyciela w systemie
Warunki końcowe	Usunięcie nauczyciela z systemu
Główny ciąg zdarzeń	1. Pracownik dziekanatu loguje się do systemu. 2. Pracownik dziekanatu wyszukuje nauczyciela, którego chce usunąć. 3. System wyświetla dane nauczyciela oraz opcję usunięcia. 4. Pracownik dziekanatu potwierdza usunięcie nauczyciela. 5. System usuwa nauczyciela z bazy danych.
Alternatywne ciągi zdarzeń	1a. Pracownik dziekanatu anuluje operację usunięcia nauczyciela. 1b. System przechodzi do stanu początkowego, bez usuwania nauczyciela.

Diagram 8: Usunięcie nauczyciela

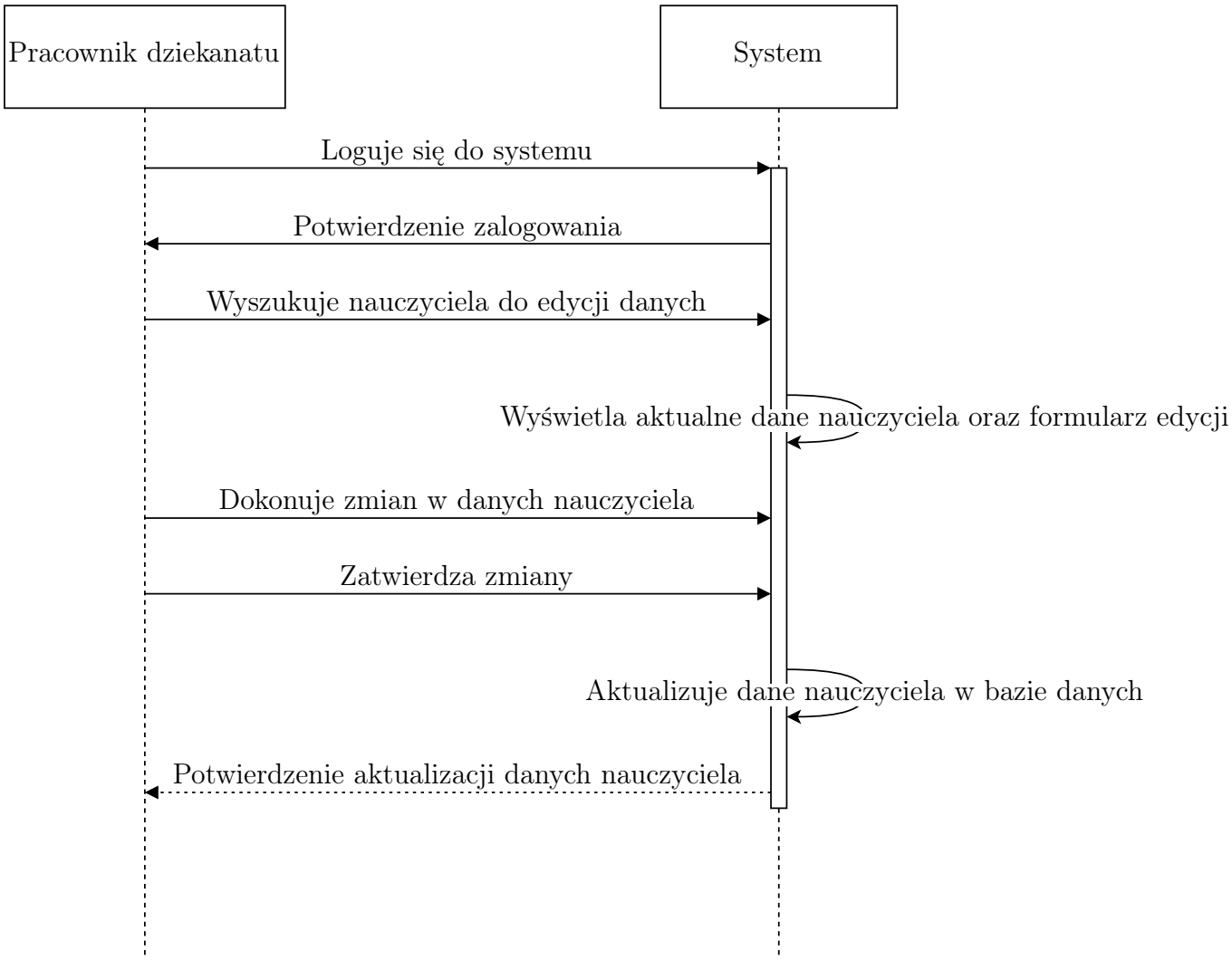


2.3.12 Edycja danych nauczyciela

Pozwala na zmianę danych nauczyciela w systemie.

Nazwa przypadku	**Edycja danych nauczyciela**
Krótki opis	Pozwala na zmianę danych nauczyciela w systemie.
Aktor biorący udział	Dziekanat
Warunki początkowe	Istnienie nauczyciela w systemie
Warunki końcowe	Zmiana danych nauczyciela w systemie
Główny ciąg zdarzeń	1. Pracownik dziekanatu loguje się do systemu. 2. Pracownik dziekanatu wyszukuje nauczyciela, którego dane chce zmienić. 3. System wyświetla aktualne dane nauczyciela oraz formularz edycji. 4. Pracownik dziekanatu dokonuje zmian w danych nauczyciela. 5. Pracownik dziekanatu zatwierdza zmiany. 6. System aktualizuje dane nauczyciela w bazie danych.
Alternatywne ciągi zdarzeń	1a. Pracownik dziekanatu anuluje operację edycji danych nauczyciela. 1b. System przechodzi do stanu początkowego, bez dokonywania zmian w danych nauczyciela.

Diagram 9: Edycja danych nauczyciela



2.3.13 Wyszukiwanie studenta

Umożliwia wyszukanie informacji o studentach.

Nazwa przypadku	**Wyszukiwanie studenta**
Krótki opis	Umożliwia wyszukanie informacji o studentach.
Aktor biorący udział	Dziekanat
Warunki początkowe	Brak
Warunki końcowe	Wyświetlenie informacji o znalezionym studencie lub informacja o braku wyników
Główny ciąg zdarzeń	1. Pracownik dziekanatu loguje się do systemu. 2. Pracownik dziekanatu wybiera opcję wyszukiwania studenta. 3. System wyświetla formularz wyszukiwania studenta. 4. Pracownik dziekanatu wprowadza kryteria wyszukiwania (np. imię, nazwisko, numer indeksu). 5. System wyszukuje studentów spełniających podane kryteria. 6. System wyświetla listę znalezionych studentów lub informację o braku wyników.
Alternatywne ciągi zdarzeń	Brak

2.3.14 Wyszukiwanie nauczyciela

Umożliwia wyszukanie informacji o nauczycielach.

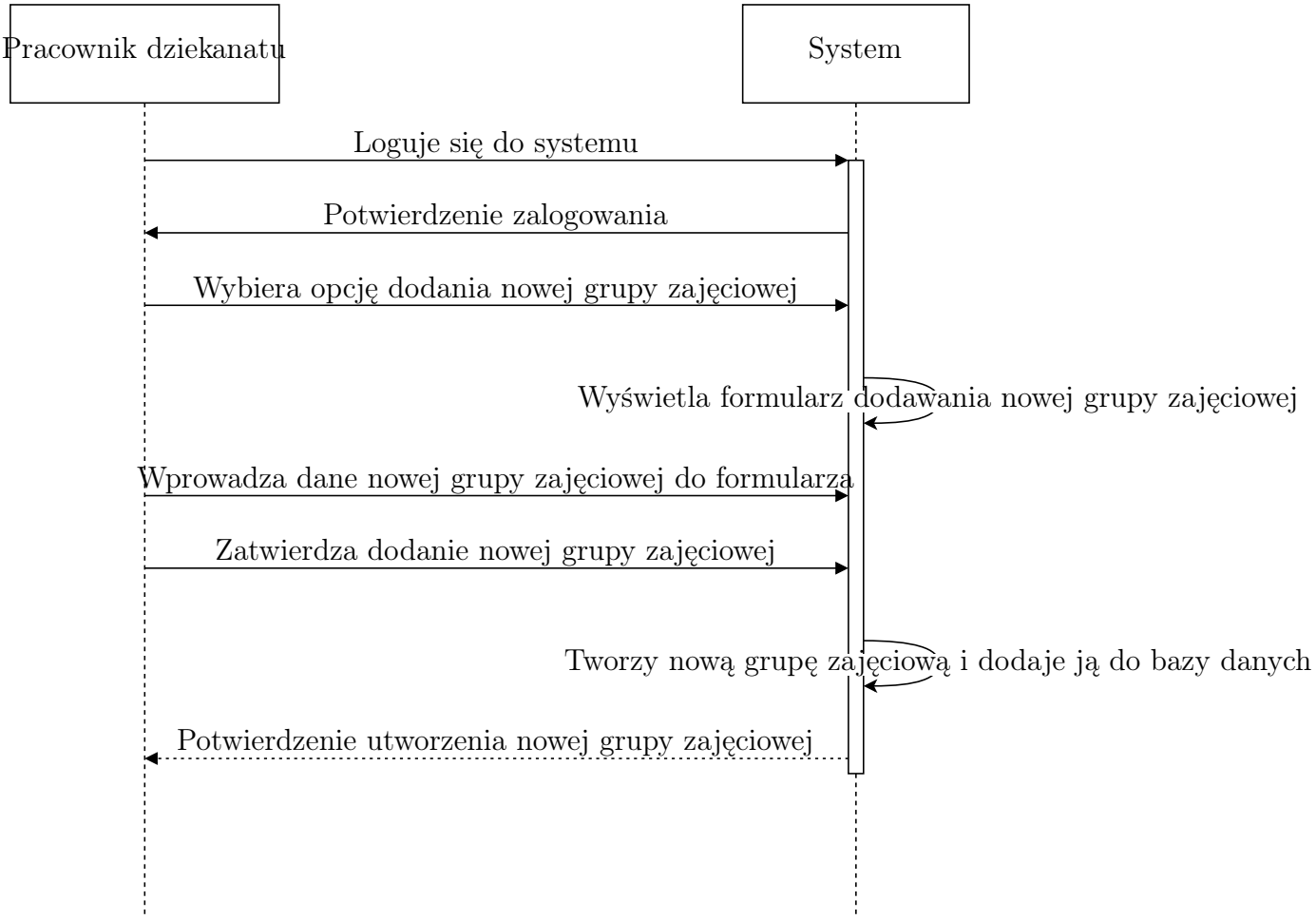
Nazwa przypadku	**Wyszukiwanie nauczyciela**
Krótki opis	Umożliwia wyszukanie informacji o nauczycielach.
Aktor biorący udział	Dziekanat
Warunki początkowe	Brak
Warunki końcowe	Wyświetlenie informacji o znalezionym nauczycielu lub informacja o braku wyników
Główny ciąg zdarzeń	1. Pracownik dziekanatu loguje się do systemu. 2. Pracownik dziekanatu wybiera opcję wyszukiwania nauczyciela. 3. System wyświetla formularz wyszukiwania nauczyciela. 4. Pracownik dziekanatu wprowadza kryteria wyszukiwania (np. imię, nazwisko, tytuł naukowy). 5. System wyszukuje nauczycieli spełniających podane kryteria. 6. System wyświetla listę znalezionych nauczycieli lub informację o braku wyników.
Alternatywne ciągi zdarzeń	Brak

2.3.15 Dodanie grupy zajęciowej

Pozwala na utworzenie nowej grupy zajęciowej.

Nazwa przypadku	**Dodanie grupy zajęciowej**
Krótki opis	Pozwala na utworzenie nowej grupy zajęciowej.
Aktor biorący udział	Dziekanat
Warunki początkowe	Brak
Warunki końcowe	Utworzenie nowej grupy zajęciowej
Główny ciąg zdarzeń	1. Pracownik dziekanatu loguje się do systemu. 2. Pracownik dziekanatu wybiera opcję dodania nowej grupy zajęciowej. 3. System wyświetla formularz dodawania nowej grupy zajęciowej. 4. Pracownik dziekanatu wprowadza dane nowej grupy zajęciowej do formularza (np. nazwa grupy, numer grupy, nauczyciel prowadzący). 5. Pracownik dziekanatu zatwierdza dodanie nowej grupy zajęciowej. 6. System tworzy nową grupę zajęciową i dodaje ją do bazy danych.
Alternatywne ciągi zdarzeń	1a. Pracownik dziekanatu anuluje operację dodania nowej grupy zajęciowej. 1b. System przechodzi do stanu początkowego, bez dodawania nowej grupy zajęciowej.

Diagram 10: Dodanie grupy zajęciowej

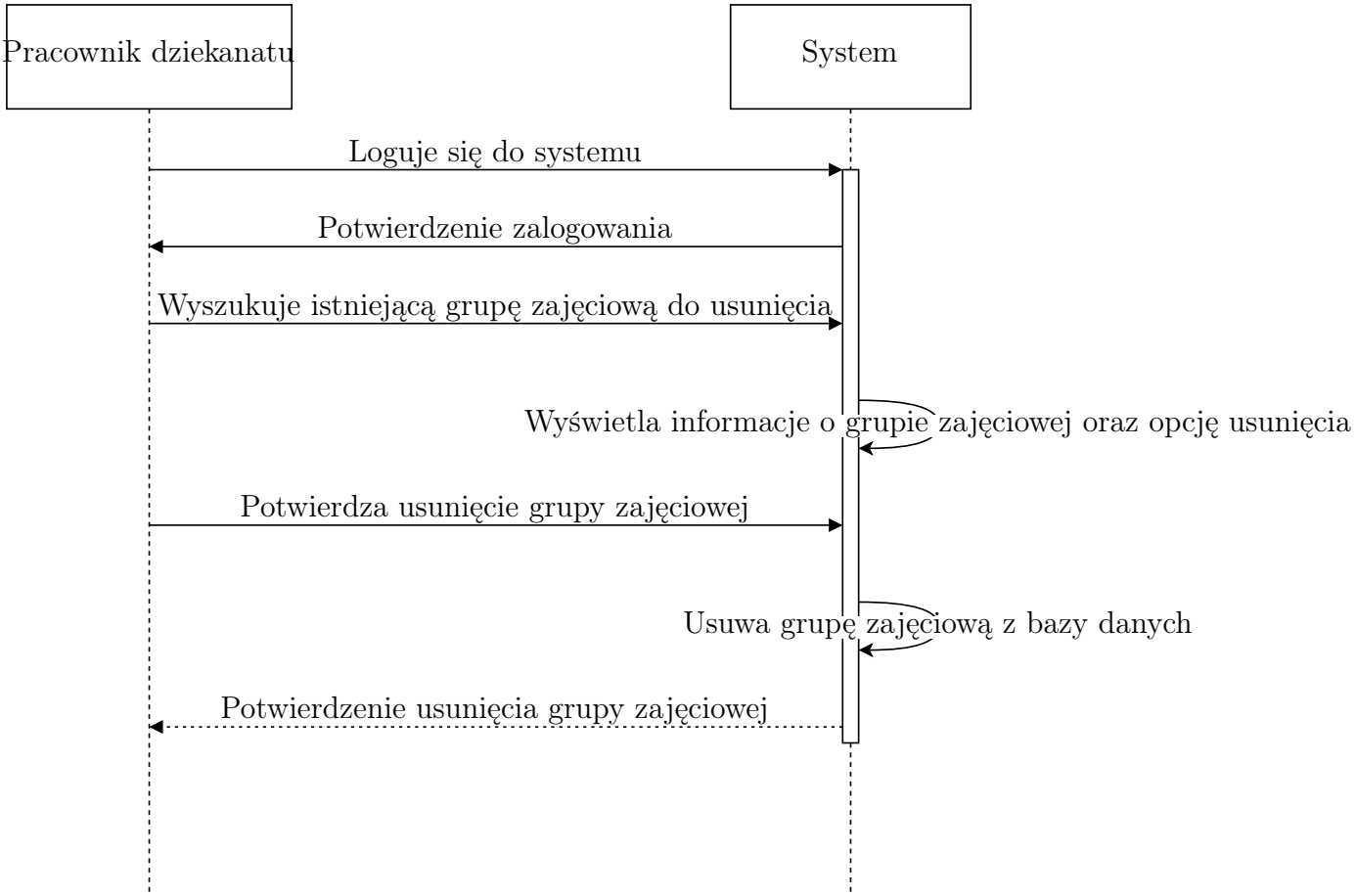


2.3.16 Usunięcie grupy zajęciowej

Umożliwia usunięcie istniejącej grupy zajęciowej.

Nazwa przypadku	**Usunięcie grupy zajęciowej**
Krótki opis	Umożliwia usunięcie istniejącej grupy zajęciowej.
Aktor biorący udział	Dziekanat
Warunki początkowe	Istnienie grupy zajęciowej w systemie
Warunki końcowe	Usunięcie grupy zajęciowej
Główny ciąg zdarzeń	1. Pracownik dziekanatu loguje się do systemu. 2. Pracownik dziekanatu wyszukuje istniejącą grupę zajęciową, którą chce usunąć. 3. System wyświetla informacje o grupie zajęciowej oraz opcję usunięcia. 4. Pracownik dziekanatu potwierdza usunięcie grupy zajęciowej. 5. System usuwa grupę zajęciową z bazy danych.
Alternatywne ciągi zdarzeń	1a. Pracownik dziekanatu anuluje operację usunięcia grupy zajęciowej. 1b. System przechodzi do stanu początkowego, bez usuwania grupy zajęciowej.

Diagram 11: Usunięcie grupy zajęciowej

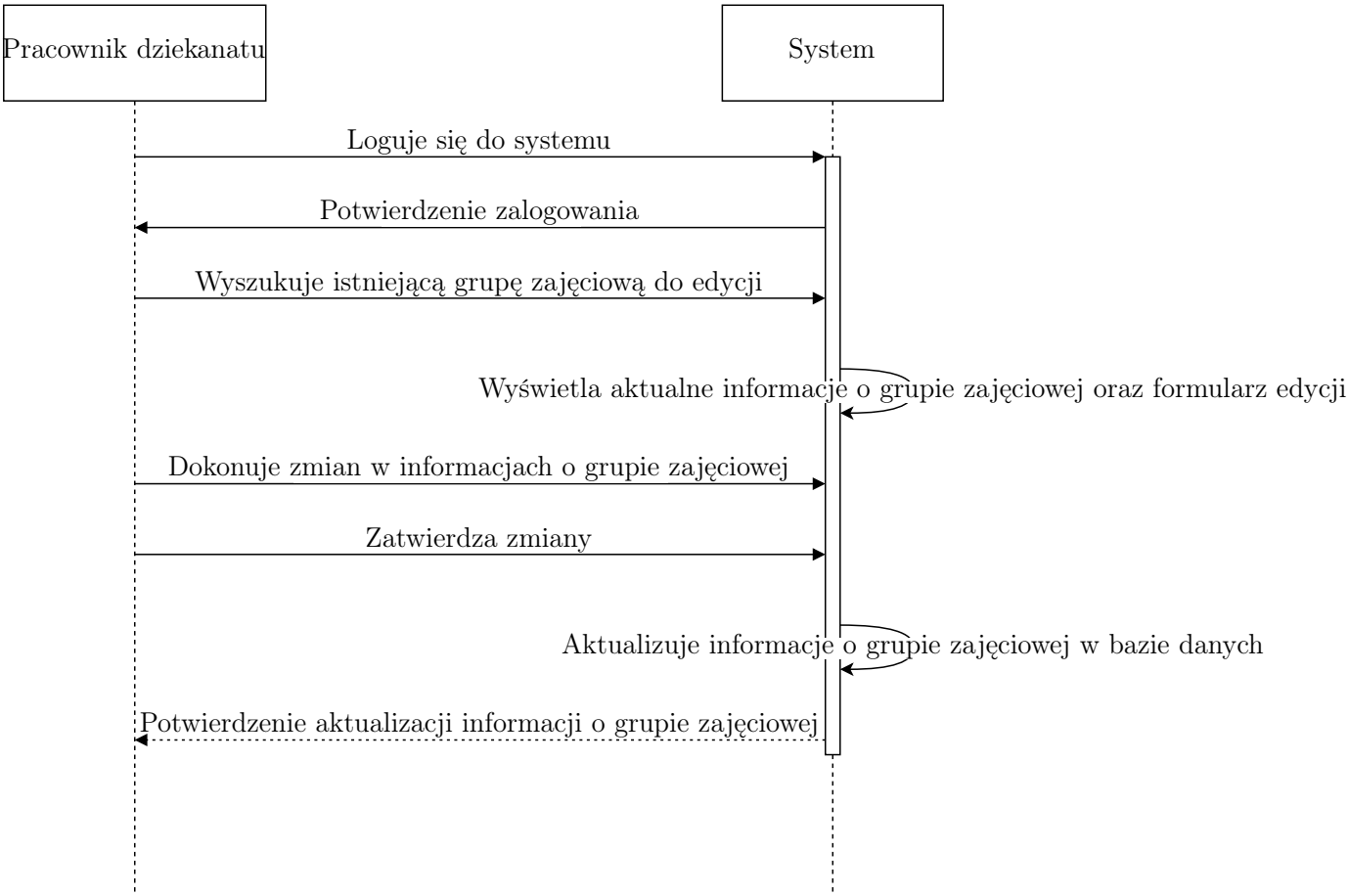


2.3.17 Edycja grupy zajęciowej

Pozwala na zmianę informacji o grupie zajęciowej.

Nazwa przypadku	**Edycja grupy zajęciowej**
Krótki opis	Pozwala na zmianę informacji o grupie zajęciowej.
Aktor biorący udział	Dziekanat
Warunki początkowe	Istnienie grupy zajęciowej w systemie
Warunki końcowe	Zmiana informacji o grupie zajęciowej
Główny ciąg zdarzeń	1. Pracownik dziekanatu loguje się do systemu. 2. Pracownik dziekanatu wyszukuje istniejącą grupę zajęciową, którą chce edytować. 3. System wyświetla aktualne informacje o grupie zajęciowej oraz formularz edycji. 4. Pracownik dziekanatu dokonuje zmian w informacjach o grupie zajęciowej. 5. Pracownik dziekanatu zatwierdza zmiany. 6. System aktualizuje informacje o grupie zajęciowej w bazie danych.
Alternatywne ciągi zdarzeń	1a. Pracownik dziekanatu anuluje operację edycji grupy zajęciowej. 1b. System przechodzi do stanu początkowego, bez dokonywania zmian w informacjach o grupie zajęciowej.

Diagram 12: Edycja grupy zajęciowej

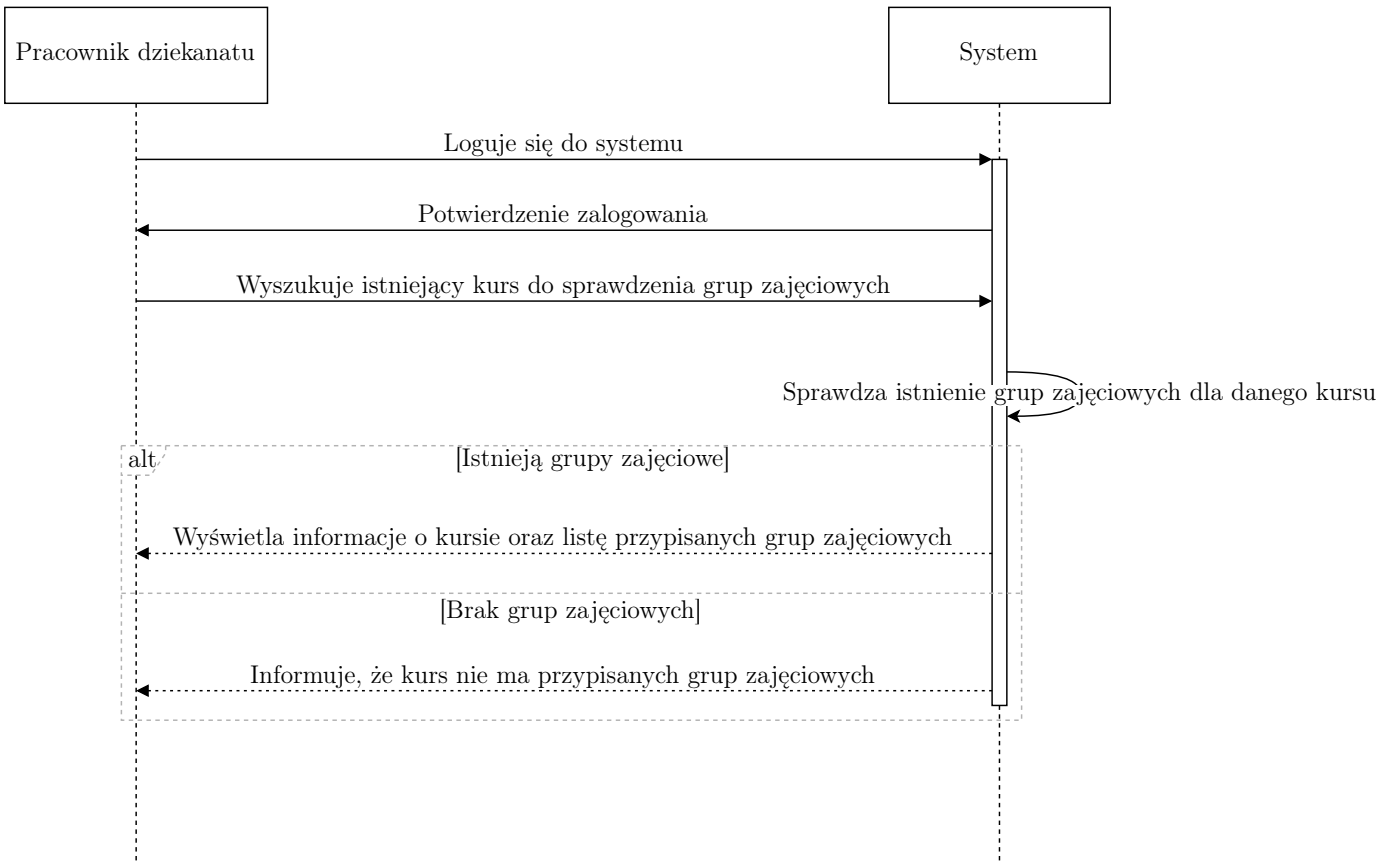


2.3.18 Sprawdzenie czy kurs posiada grupy zajęciowe

Umożliwia sprawdzenie, czy dany kurs ma przypisane grupy zajęciowe.

Nazwa przypadku	**Sprawdzenie czy kurs posiada grupy zajęciowe**
Krótki opis	Umożliwia sprawdzenie, czy dany kurs ma przypisane grupy zajęciowe.
Aktor biorący udział	Dziekanat
Warunki początkowe	Istnienie kursu w systemie
Warunki końcowe	Wyświetlenie informacji o grupach zajęciowych lub brak grup zajęciowych dla kursu
Główny ciąg zdarzeń	1. Pracownik dziekanatu loguje się do systemu. 2. Pracownik dziekanatu wyszukuje istniejący kurs, którego grupy zajęciowe chce sprawdzić. 3. System wyświetla informacje o kursie oraz listę przypisanych grup zajęciowych (jeśli istnieją).
Alternatywne ciągi zdarzeń	1a. System informuje, że kurs nie ma przypisanych grup zajęciowych.

Diagram 13: Sprawdzenie czy kurs posiada grupy zajęciowe

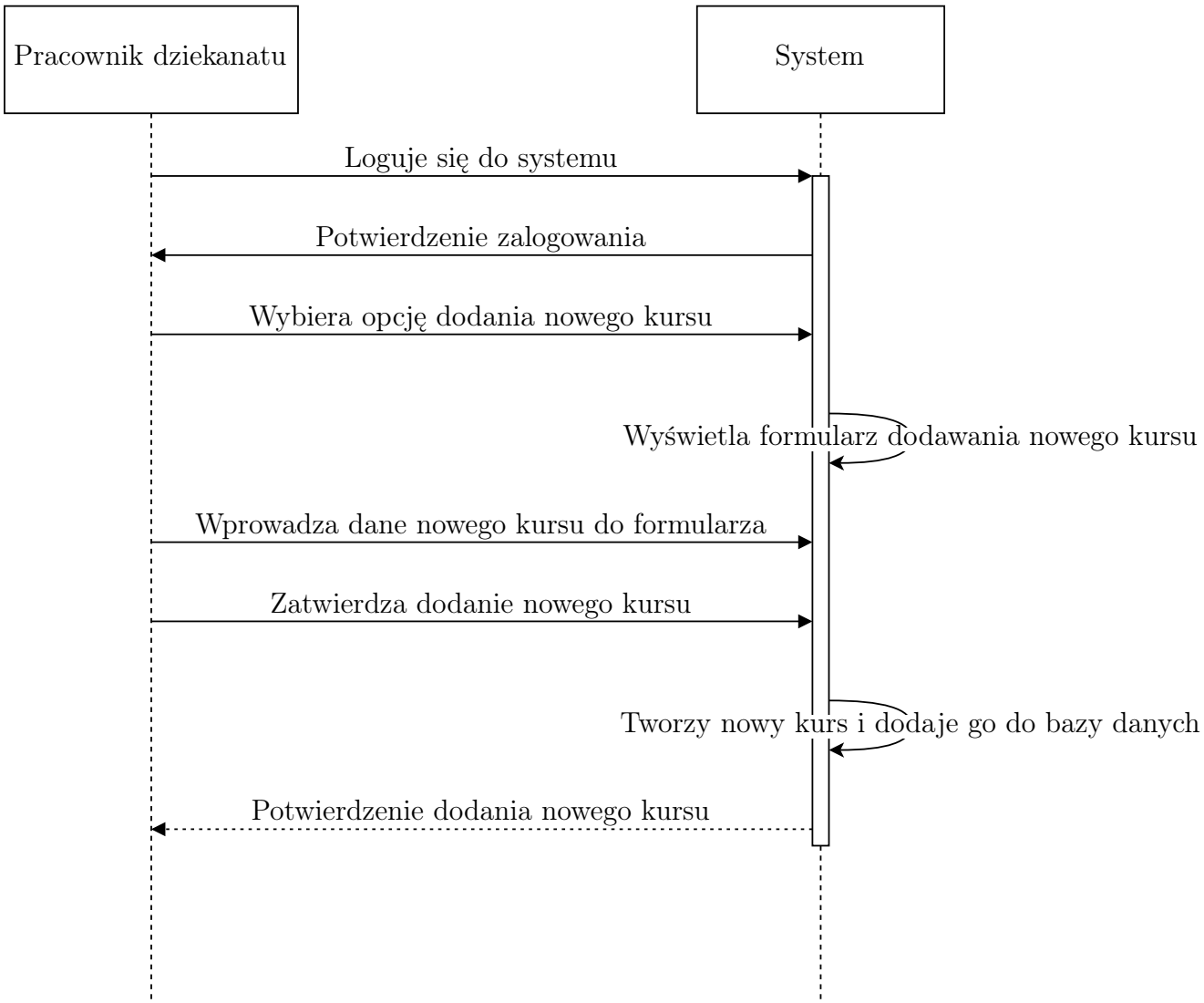


2.3.19 Dodanie kursu

Pozwala na utworzenie nowego kursu w systemie.

Nazwa przypadku	**Dodanie kursu**
Krótki opis	Pozwala na utworzenie nowego kursu w systemie.
Aktor biorący udział	Dziekanat
Warunki początkowe	Brak
Warunki końcowe	Utworzenie nowego kursu w systemie
Główny ciąg zdarzeń	1. Pracownik dziekanatu loguje się do systemu. 2. Pracownik dziekanatu wybiera opcję dodania nowego kursu. 3. System wyświetla formularz dodawania nowego kursu. 4. Pracownik dziekanatu wprowadza dane nowego kursu do formularza (np. nazwa kursu, kod kursu, prowadzący). 5. Pracownik dziekanatu zatwierdza dodanie nowego kursu. 6. System tworzy nowy kurs i dodaje go do bazy danych.
Alternatywne ciągi zdarzeń	1a. Pracownik dziekanatu anuluje operację dodania nowego kursu. 1b. System przechodzi do stanu początkowego, bez dodawania nowego kursu.

Diagram 14: Dodanie kursu

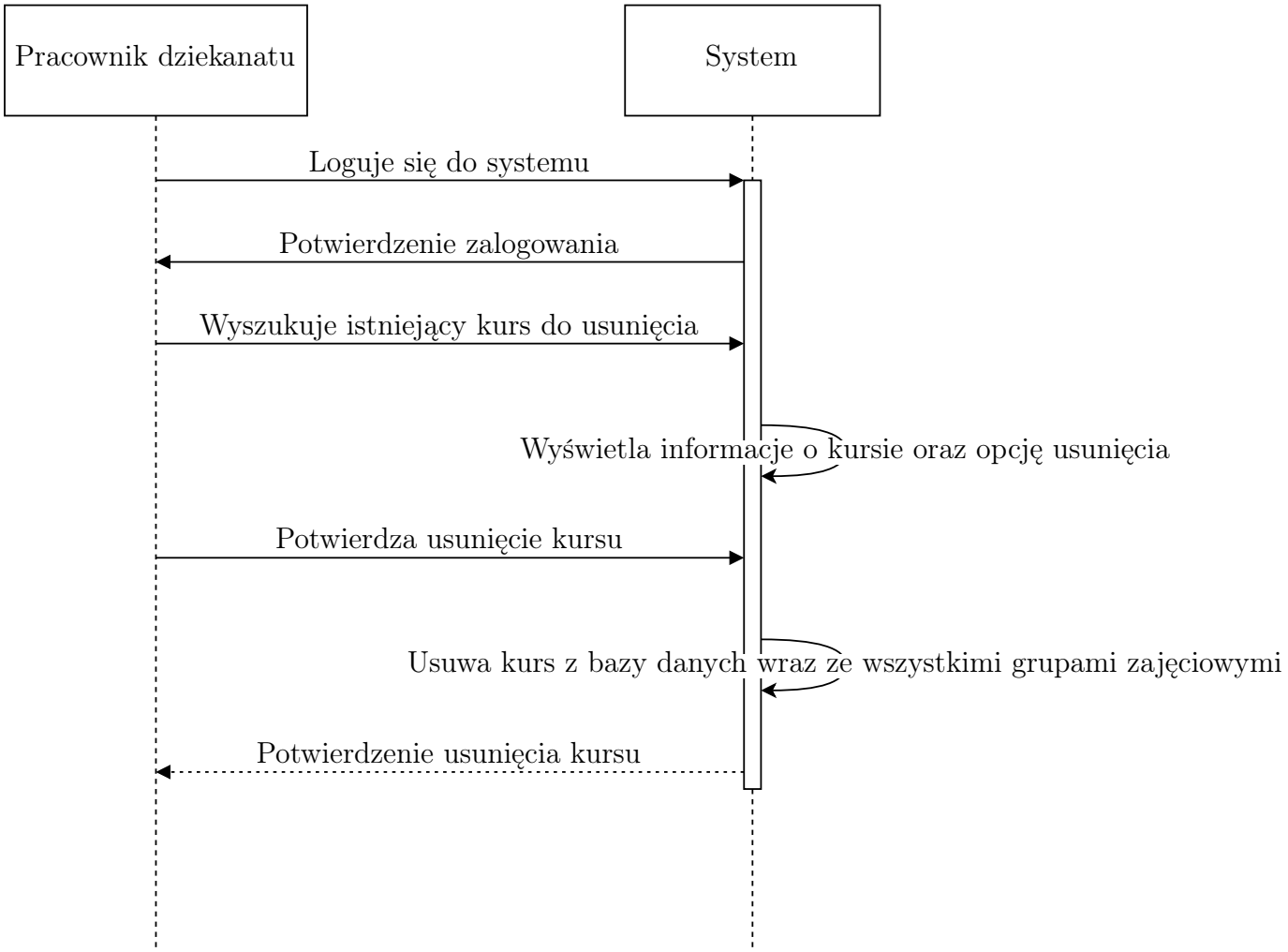


2.3.20 Usunięcie kursu

Umożliwia usunięcie kursu z systemu.

Nazwa przypadku	**Usunięcie kursu**
Krótki opis	Umożliwia usunięcie kursu z systemu.
Aktor biorący udział	Dziekanat
Warunki początkowe	Istnienie kursu w systemie
Warunki końcowe	Usunięcie kursu z systemu
Główny ciąg zdarzeń	1. Pracownik dziekanatu loguje się do systemu. 2. Pracownik dziekanatu wyszukuje istniejący kurs, który chce usunąć. 3. System wyświetla informacje o kursie oraz opcję usunięcia. 4. Pracownik dziekanatu potwierdza usunięcie kursu. 5. System usuwa kurs z bazy danych wraz ze wszystkimi grupami zajęciowymi.
Alternatywne ciągi zdarzeń	1a. Pracownik dziekanatu anuluje operację usunięcia kursu. 1b. System przechodzi do stanu początkowego, bez usuwania kursu.

Diagram 15: Usunięcie kursu

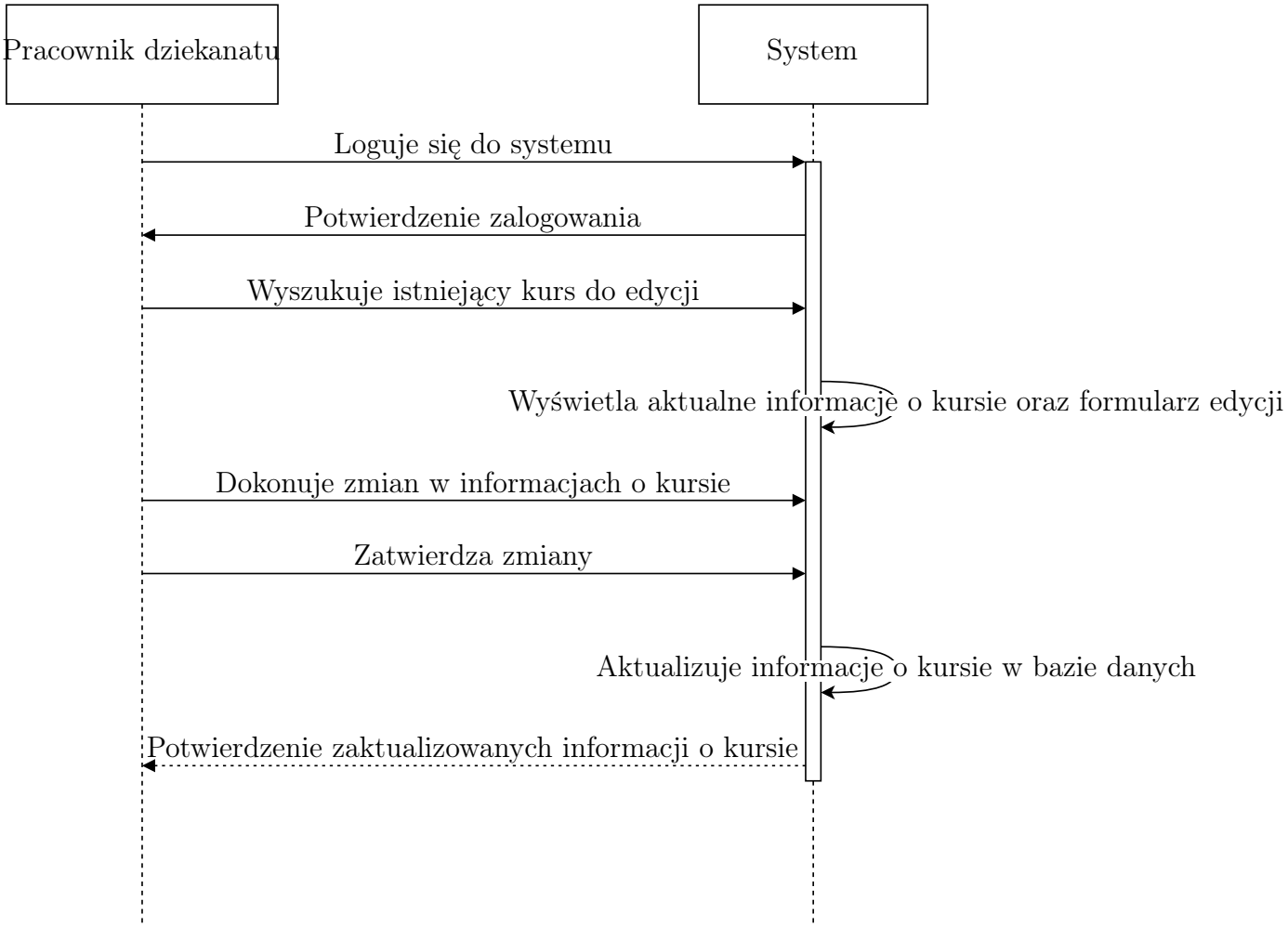


2.3.21 Edycja kursu

Pozwala na zmianę informacji o kursie.

Nazwa przypadku	**Edycja kursu**
Krótki opis	Pozwala na zmianę informacji o kursie.
Aktor biorący udział	Dziekanat
Warunki początkowe	Istnienie kursu w systemie
Warunki końcowe	Zmiana informacji o kursie
Główny ciąg zdarzeń	1. Pracownik dziekanatu loguje się do systemu. 2. Pracownik dziekanatu wyszukuje istniejący kurs, który chce edytować. 3. System wyświetla aktualne informacje o kursie oraz formularz edycji. 4. Pracownik dziekanatu dokonuje zmian w informacjach o kursie. 5. Pracownik dziekanatu zatwierdza zmiany. 6. System aktualizuje informacje o kursie w bazie danych.
Alternatywne ciągi zdarzeń	1a. Pracownik dziekanatu anuluje operację edycji kursu. 1b. System przechodzi do stanu początkowego, bez dokonywania zmian w informacjach o kursie.

Diagram 16: Edycja kursu

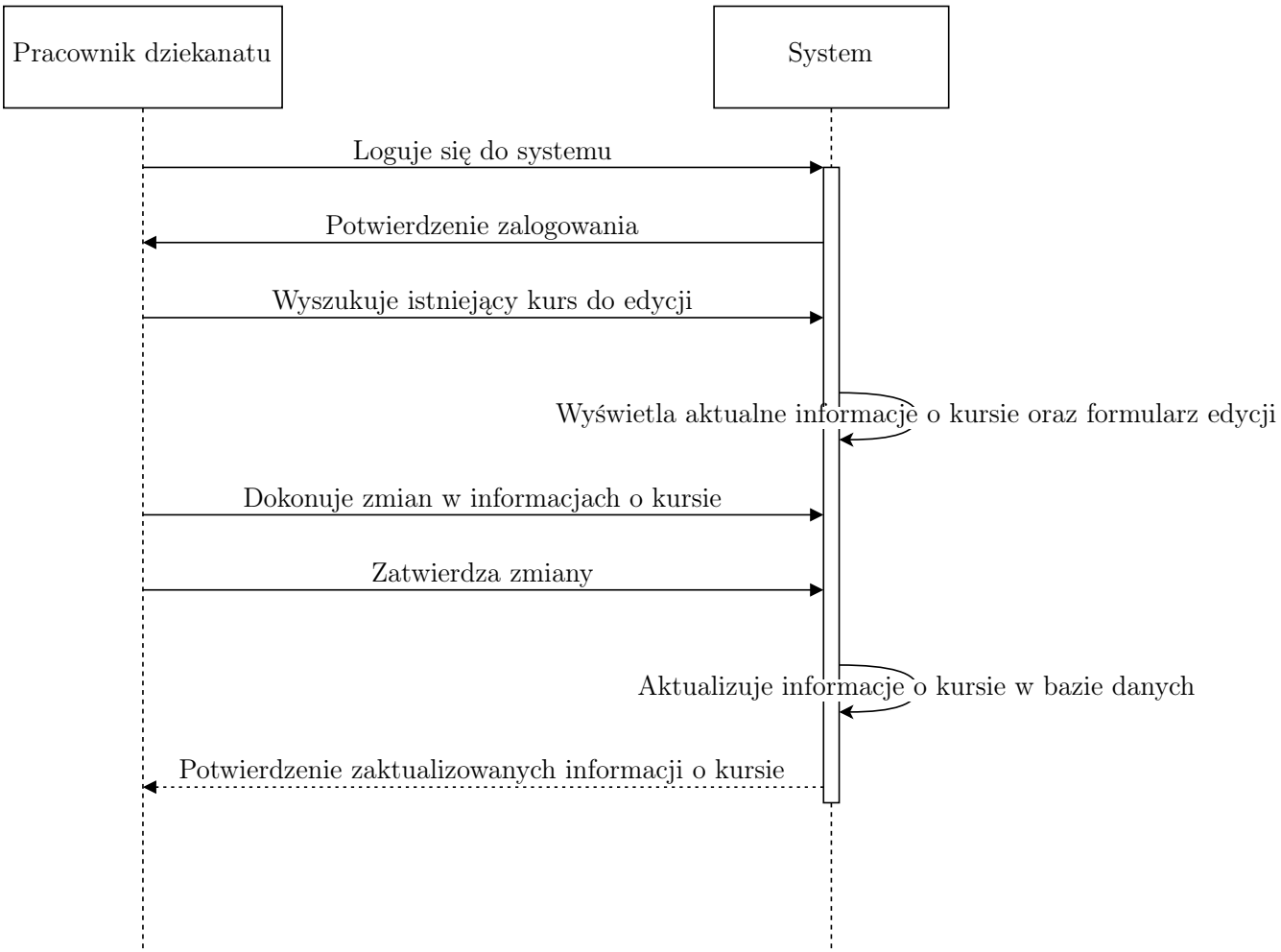


2.3.22 Wyszukiwanie kursu

Umożliwia wyszukanie informacji o kursach.

Nazwa przypadku	**Wyszukiwanie kursu**
Krótki opis	Umożliwia wyszukanie informacji o kursach.
Aktor biorący udział	Dziekanat
Warunki początkowe	Istnienie kursów w systemie
Warunki końcowe	Wyświetlenie informacji o znalezionych kursach
Główny ciąg zdarzeń	1. Pracownik dziekanatu loguje się do systemu. 2. Pracownik dziekanatu wprowadza kryteria wyszukiwania (np. nazwa kursu, kod kursu, prowadzący). 3. System przeszukuje bazę danych w poszukiwaniu kursów spełniających podane kryteria. 4. System wyświetla listę znalezionych kursów.
Alternatywne ciągi zdarzeń	1a. System informuje, że nie znaleziono kursów spełniających podane kryteria.

Diagram 17: Wyszukiwanie kursu

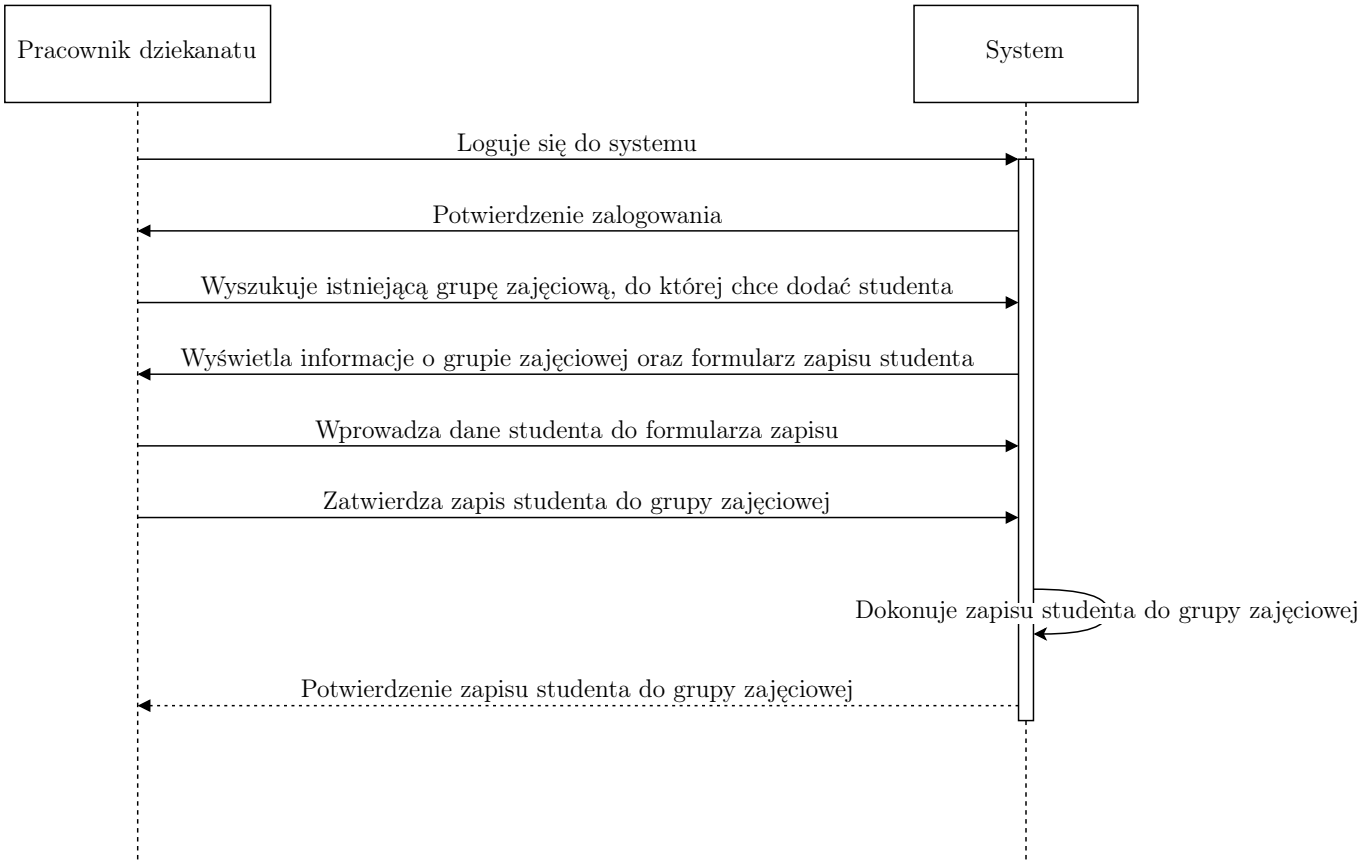


2.3.23 Administracyjny zapis do grupy

Umożliwia zapisanie studenta do grupy zajęciowej przez administratora.

Nazwa przypadku	**Administracyjny zapis do grupy**
Krótki opis	Umożliwia zapisanie studenta do grupy zajęciowej przez administratora.
Aktor biorący udział	Dziekanat
Warunki początkowe	Istnienie grupy zajęciowej w systemie
Warunki końcowe	Zapisanie studenta do grupy zajęciowej
Główny ciąg zdarzeń	1. Pracownik dziekanatu loguje się do systemu. 2. Pracownik dziekanatu wyszukuje istniejącą grupę zajęciową, do której chce dodać studenta. 3. System wyświetla informacje o grupie zajęciowej oraz formularz zapisu studenta. 4. Pracownik dziekanatu wprowadza dane studenta do formularza zapisu. 5. Pracownik dziekanatu zatwierdza zapis studenta do grupy zajęciowej. 6. System dokonuje zapisu studenta do grupy zajęciowej.
Alternatywne ciągi zdarzeń	1a. Pracownik dziekanatu anuluje operację zapisu studenta do grupy zajęciowej. 1b. System przechodzi do stanu początkowego, bez dokonania zapisu studenta.

Diagram 18: Administracyjny zapis do grupy

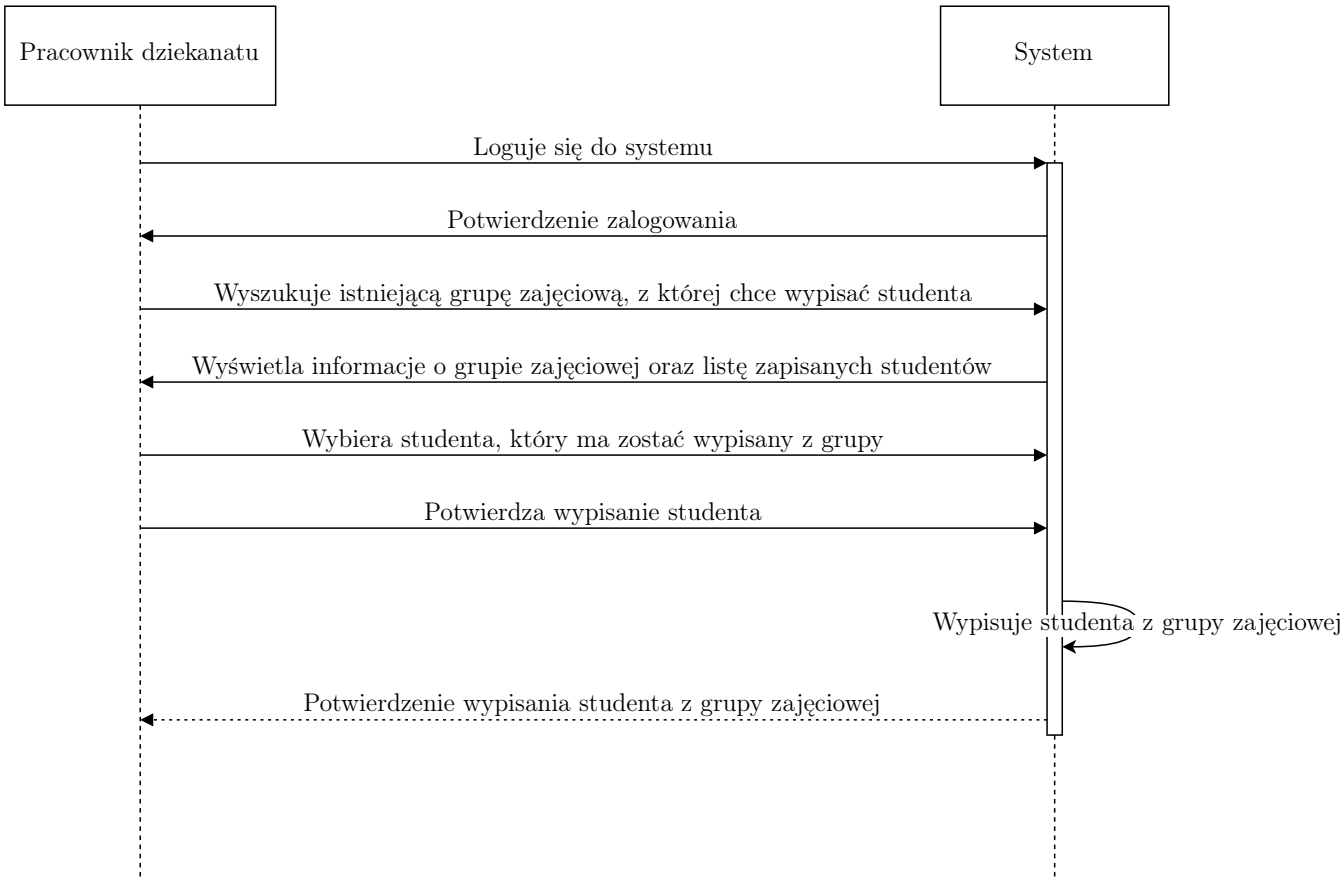


2.3.24 Administracyjny wypis z grupy

Umożliwia wypisanie studenta z grupy zajęciowej przez administratora.

Nazwa przypadku	**Administracyjny wypis z grupy**
Krótki opis	Umożliwia wypisanie studenta z grupy zajęciowej przez administratora.
Aktor biorący udział	Dziekanat
Warunki początkowe	Istnienie grupy zajęciowej w systemie oraz zapisanie studenta do tej grupy
Warunki końcowe	Wypisanie studenta z grupy zajęciowej
Główny ciąg zdarzeń	1. Pracownik dziekanatu loguje się do systemu. 2. Pracownik dziekanatu wyszukuje istniejącą grupę zajęciową, z której chce wypisać studenta. 3. System wyświetla informacje o grupie zajęciowej oraz listę zapisanych studentów. 4. Pracownik dziekanatu wybiera studenta, który ma zostać wypisany z grupy. 5. Pracownik dziekanatu potwierdza wypisanie studenta. 6. System wypisuje studenta z grupy zajęciowej.
Alternatywne ciągi zdarzeń	1a. Pracownik dziekanatu anuluje operację wypisania studenta z grupy zajęciowej. 1b. System przechodzi do stanu początkowego, bez dokonania wypisania studenta.

Diagram 19: Administracyjny wypis z grupy

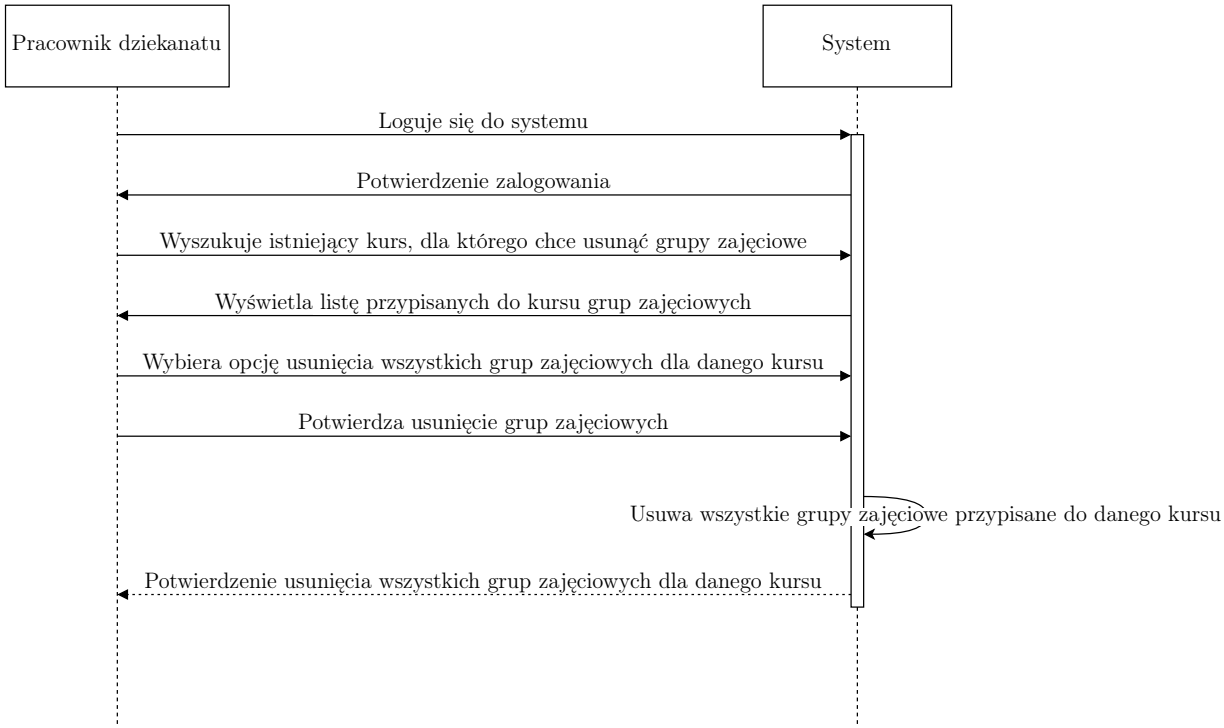


2.3.25 Usunięcie wszystkich grup zajęciowych danego kursu

Pozwala na usunięcie wszystkich grup przypisanych do konkretnego kursu.

Nazwa przypadku	**Usunięcie wszystkich grup zajęciowych danego kursu**
Krótki opis	Pozwala na usunięcie wszystkich grup przypisanych do konkretnego kursu.
Aktor biorący udział	Dziekanat
Warunki początkowe	Istnienie kursu w systemie oraz przypisane do niego grupy zajęciowe
Warunki końcowe	Usunięcie wszystkich grup zajęciowych danego kursu
Główny ciąg zdarzeń	1. Pracownik dziekanatu loguje się do systemu. 2. Pracownik dziekanatu wyszukuje istniejący kurs, dla którego chce usunąć grupy zajęciowe. 3. System wyświetla listę przypisanych do kursu grup zajęciowych. 4. Pracownik dziekanatu wybiera opcję usunięcia wszystkich grup zajęciowych dla danego kursu. 5. Pracownik dziekanatu potwierdza usunięcie grup zajęciowych. 6. System usuwa wszystkie grupy zajęciowe przypisane do danego kursu.
Alternatywne ciągi zdarzeń	1a. Pracownik dziekanatu anuluje operację usunięcia grup zajęciowych. 1b. System przechodzi do stanu początkowego, bez usuwania grup zajęciowych.

Diagram 20: Usunięcie wszystkich grup zajęciowych danego kursu



2.3.26 Dodanie oceny dla studenta

Umożliwia dodanie nowej oceny dla studenta.

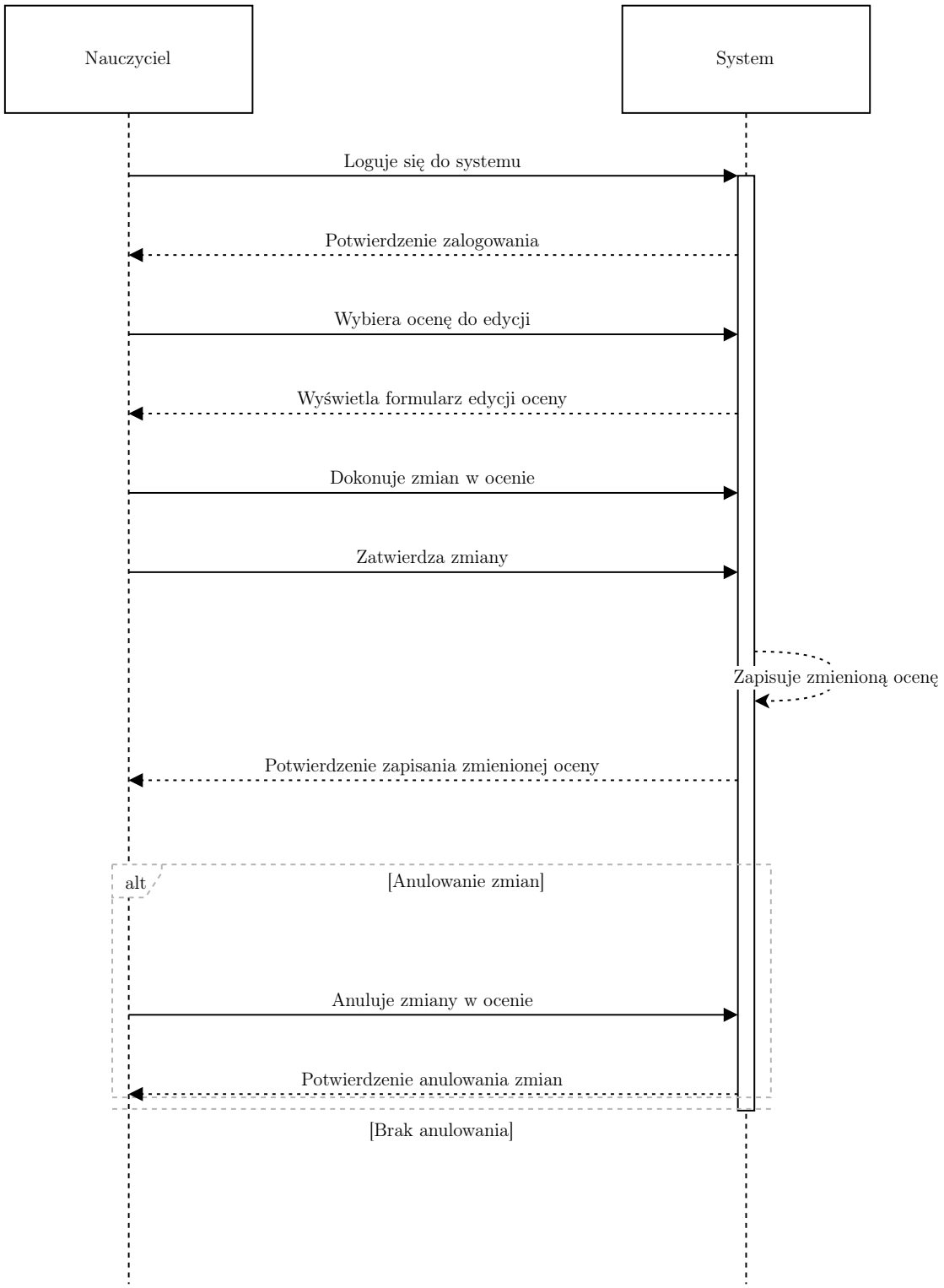
Nazwa przypadku	**Dodanie oceny dla studenta**
Krótki opis	Umożliwia dodanie nowej oceny dla studenta.
Aktor biorący udział	Nauczyciel
Warunki początkowe	Istnienie studenta i kursu, dla którego oceniana jest praca
Warunki końcowe	Dodanie nowej oceny dla studenta
Główny ciąg zdarzeń	1. Nauczyciel loguje się do systemu. 2. Nauczyciel wybiera odpowiedniego studenta oraz kurs, dla którego chce dodać ocenę. 3. System wyświetla formularz dodawania nowej oceny. 4. Nauczyciel wprowadza ocenę oraz ewentualny komentarz. 5. Nauczyciel zatwierdza dodanie oceny. 6. System zapisuje nową ocenę dla studenta.
Alternatywne ciągi zdarzeń	Brak

2.3.27 Edycja oceny dla studenta

Pozwala na zmianę istniejącej oceny studenta.

Nazwa przypadku	**Edycja oceny dla studenta**
Krótki opis	Pozwala na zmianę istniejącej oceny studenta.
Aktor biorący udział	Nauczyciel
Warunki początkowe	Istnienie oceny studenta w systemie
Warunki końcowe	Zmiana istniejącej oceny dla studenta
Główny ciąg zdarzeń	1. Nauczyciel loguje się do systemu. 2. Nauczyciel wybiera ocenę, którą chce edytować. 3. System wyświetla formularz edycji oceny. 4. Nauczyciel dokonuje zmian w ocenie. 5. Nauczyciel zatwierdza zmiany. 6. System zapisuje zmienioną ocenę.
Alternatywne ciągi zdarzeń	1a. Nauczyciel anuluje zmiany w ocenie. 1b. System przechodzi do stanu początkowego, bez dokonania zmian w ocenie.

Diagram 21: Edycja oceny studenta

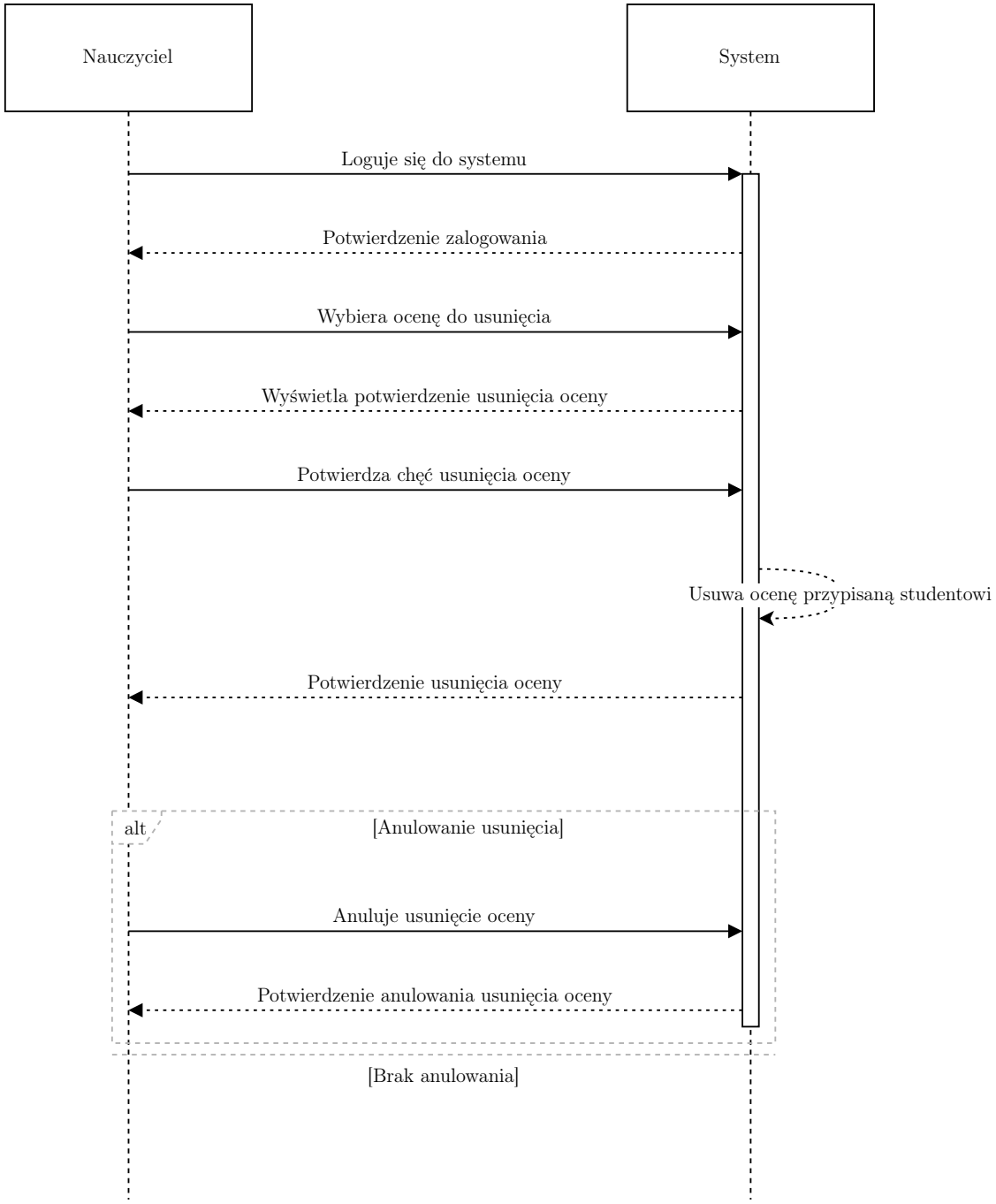


2.3.28 Usunięcie oceny dla studenta

Umożliwia usunięcie oceny przypisanej studentowi.

Nazwa przypadku	**Usunięcie oceny dla studenta**
Krótki opis	Umożliwia usunięcie oceny przypisanej studentowi.
Aktor biorący udział	Nauczyciel
Warunki początkowe	Istnienie oceny studenta w systemie
Warunki końcowe	Usunięcie oceny dla studenta
Główny ciąg zdarzeń	1. Nauczyciel loguje się do systemu. 2. Nauczyciel wybiera ocenę, którą chce usunąć. 3. System wyświetla potwierdzenie usunięcia oceny. 4. Nauczyciel potwierdza chęć usunięcia oceny. 5. System usuwa ocenę przypisaną studentowi.
Alternatywne ciągi zdarzeń	1a. Nauczyciel anuluje usunięcie oceny. 1b. System przechodzi do stanu początkowego, bez usunięcia oceny.

Diagram 22: Usunięcie oceny studenta



2.3.29 Przeglądanie kursów prowadzonych przez nauczyciela

Umożliwia nauczycielowi przeglądanie listy kursów, które prowadzi.

Nazwa przypadku	**Przeglądanie kursów prowadzonych przez nauczyciela**
Krótki opis	Umożliwia nauczycielowi przeglądanie listy kursów, które prowadzi.
Aktor biorący udział	Nauczyciel
Warunki początkowe	Zalogowanie się nauczyciela do systemu
Warunki końcowe	Wyświetlenie listy kursów prowadzonych przez nauczyciela
Główny ciąg zdarzeń	1. Nauczyciel loguje się do systemu. 2. System identyfikuje nauczyciela i jego przypisane kursy. 3. System wyświetla listę kursów prowadzonych przez nauczyciela.
Alternatywne ciągi zdarzeń	Brak

3 Architektura Systemu

3.1 Opis Architekury Systemu

3.1.1 Użytkownicy Systemu

- **Student** - Użytkownik systemu, który ma możliwość rejestracji na kursy, przeglądania dostępnych kursów, przeglądania ocen.
- **Dziekanat** - Pracownicy dziekanatu, którzy zarządzają administracyjną częścią systemu, w tym zarządzaniem kursami, grupami i użytkownikami.
- **Nauczyciel** - Użytkownik, który ma dostęp do funkcji związanych z nauczaniem.

3.1.2 Interfejsy Webowe

System oferuje trzy główne interfejsy webowe, z których każdy jest dostosowany do potrzeb różnych typów użytkowników:

- **Interfejs Studenta** - Umożliwia studentom dostęp do funkcji takich jak rejestracja na kursy, przeglądanie dostępnych kursów, przeglądanie ocen.
- **Interfejs Dziekanatu** - Umożliwia pracownikom dziekanatu zarządzanie użytkownikami, grupami i kursami.
- **Interfejs Nauczyciela** - Umożliwia nauczycielom zarządzanie, ocenianie studentów, monitorowanie obecności studentów.

3.1.3 Serwer Webowy Java

- **Serwer Aplikacji** - Kluczowy komponent, który obsługuje wszystkie żądania przychodzące z interfejsów webowych. Serwer aplikacji przetwarza żądania, zarządza sesjami użytkowników, autoryzacją i przekazuje je do logiki biznesowej.

3.1.4 Backend Java

- **Logika Biznesowa** - Centralny moduł obsługujący logikę aplikacji. Realizuje wszystkie operacje związane z funkcjonowaniem systemu, takie jak zarządzanie danymi studentów, kursów, grup, ocen i obecności. Komunikuje się bezpośrednio z bazą danych oraz poszczególnymi modułami zarządzania.

3.1.5 Moduły Zarządzania

System składa się z kilku wyspecjalizowanych modułów zarządzania, które współpracują z logiką biznesową:

- **Zarządzanie użytkownikami** - Moduł odpowiedzialny za tworzenie, aktualizowanie i usuwanie kont użytkowników (studenci, nauczyciele, pracownicy dziekanatu).
- **Zarządzanie grupami** - Moduł do zarządzania grupami studenckimi, w tym tworzenie nowych grup i przypisywanie studentów do grup.

- **Zarządzanie kursami** - Moduł umożliwiający dodawanie, modyfikowanie i usuwanie kursów oraz przypisywanie kursów do nauczycieli.
- **Rejestracja i przeglądanie** - Moduł umożliwiający studentom rejestrację na kursy oraz przeglądanie dostępnych kursów.
- **Oceny i obecność** - Moduł do zarządzania ocenami studentów oraz monitorowania ich obecności na zajęciach.
- **Dodatkowe funkcje** - Moduł obsługujący dodatkowe funkcje wspierające działanie systemu.

3.1.6 Baza Danych

- **PostgreSQL** - Relacyjna baza danych, która przechowuje wszystkie dane systemu, w tym dane użytkowników, kursów, grup, ocen, obecności. Baza danych zapewnia integralność danych oraz możliwość ich skomplikowanego przetwarzania i wyszukiwania.

3.1.7 Przepływ Danych w Systemie

1. Użytkownicy (student, dziekanat, nauczyciel) korzystają z odpowiednich interfejsów webowych do interakcji z systemem.
2. Interfejsy webowe przesyłają żądania do serwera aplikacji działającego na serwerze webowym Java. Serwer aplikacji jest odpowiedzialny za autoryzację użytkowników, zarządzanie sesjami oraz przetwarzanie żądań.
3. Serwer aplikacji przekazuje żądania do logiki biznesowej w backendzie Java, gdzie realizowane są odpowiednie operacje biznesowe.
4. Logika biznesowa współpracuje z różnymi modułami zarządzania oraz komunikuje się z bazą danych PostgreSQL, wykonując operacje zapisu, odczytu, aktualizacji i usuwania danych.
5. Wyniki operacji są zwracane przez serwer aplikacji do odpowiednich interfejsów webowych, które następnie prezentują je użytkownikom.

Cała architektura jest zaprojektowana w sposób zapewniający skalowalność, elastyczność i możliwość łatwego rozszerzania systemu o nowe funkcje w przyszłości. Dzięki użyciu języka Java oraz relacyjnej bazy danych PostgreSQL, system jest stabilny, bezpieczny i wydajny, co jest kluczowe dla jego prawidłowego funkcjonowania w środowisku akademickim.

3.2 Diagram Architektury

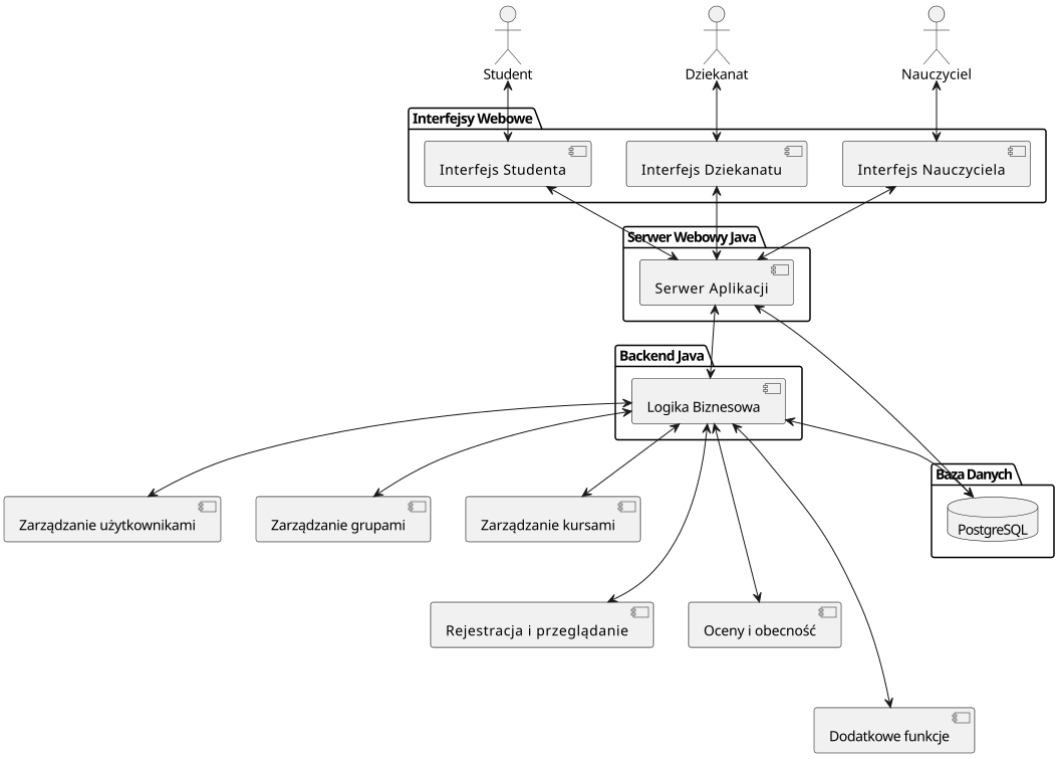


Diagram 23: Diagram architektury systemu.

4 Technologie

4.1 Frontend

- **React.js** - Biblioteka JavaScript do budowy interfejsów użytkownika. Jest to popularne narzędzie, które umożliwia tworzenie responsywnych, interaktywnych aplikacji internetowych.
- **Redux** - Biblioteka do zarządzania stanem aplikacji w połączeniu z React.js. Zapewnia jednolite zarządzanie stanem aplikacji, co jest istotne w przypadku systemu obsługującego różne rodzaje użytkowników i operacji.
- **Material-UI** - Biblioteka komponentów React zgodnych z Material Design. Umożliwia szybkie tworzenie estetycznych i intuicyjnych interfejsów użytkownika.
- **Styled Components** - Biblioteka pozwalająca na pisanie CSS wewnątrz komponentów w React.js. Zapewnia modularność i łatwiejsze zarządzanie stylami aplikacji.

4.2 Backend

- **Spring Boot** - Framework do tworzenia aplikacji w języku Java. Jest to popularne narzędzie, które zapewnia szybkie wdrożenie oraz obsługę protokołu HTTP.
- **Spring Security** - Moduł Spring Framework zapewniający wsparcie dla mechanizmów uwierzytelniania, autoryzacji i zarządzania sesjami użytkowników.
- **Spring Data JPA** - Moduł Spring Framework zapewniający abstrakcję dostępu do danych w bazie danych relacyjnej. Umożliwia efektywne zarządzanie danymi bezpośrednio z poziomu aplikacji.

4.3 Baza Danych

- **PostgreSQL** - Zaawansowany system zarządzania relacyjnymi bazami danych. Zapewnia wysoką wydajność, niezawodność i skalowalność, co jest istotne dla systemu obsługującego dużą ilość danych.

4.4 Narzędzia Deweloperskie

- **Git** - System kontroli wersji, umożliwiający efektywne zarządzanie kodem źródłowym aplikacji.
- **GitHub Actions** - Platforma do automatyzacji procesów CI/CD (Continuous Integration/Continuous Deployment) na GitHubie. Umożliwia tworzenie i uruchamianie skryptów automatyzacyjnych bezpośrednio w repozytorium.
- **Docker** - Platforma do konteneryzacji aplikacji, umożliwiająca izolację środowiska uruchomieniowego i łatwe wdrażanie aplikacji w różnych środowiskach.

5 Opis klas

1. Student: Reprezentuje studenta z atrybutami takimi jak imię, ID, informacje kontaktowe i szczegóły akademickie.
2. Group: Definiuje grupę lub klasę, do której zapisani są studenci, w tym ID grupy, nazwę i powiązane kursy.
3. Course: Zawiera szczegóły dotyczące kursu, takie jak ID kursu, nazwa, opis i punkty kredytowe.
4. Enrollment: Rejestruje zapisy studentów na kursy lub do grup, w tym ID studenta, ID kursu i datę zapisu.
5. Grade: Przechowuje oceny przyznane studentom za ich kursy, w tym ID studenta, ID kursu i datę.
6. Attendance: Śledzi frekwencję studentów na kursach lub zajęciach, w tym ID studenta, datę, status (obecny/nieobecny) i ID kursu.
7. User: Reprezentuje użytkownika systemu z atrybutami takimi jak ID, imię, nazwisko, adres e-mail i hasło.
8. Role: Definiuje role użytkowników systemu, w tym ID roli, nazwę i opis.
9. GroupRepository: Przechowuje i zarządza danymi grup w bazie danych.
10. EnrollmentRepository: Przechowuje i zarządza danymi zapisów studentów w bazie danych.
11. CourseRepository: Przechowuje i zarządza danymi kursów w bazie danych.
12. StudentRepository: Przechowuje i zarządza danymi studentów w bazie danych.
13. TeacherRepository: Przechowuje i zarządza danymi nauczycieli w bazie danych.
14. AttendanceRepository: Przechowuje i zarządza danymi frekwencji w bazie danych.
15. SecurityRepository: Przechowuje i zarządza danymi związanymi z bezpieczeństwem systemu w bazie danych.
16. RoleRepository: Przechowuje i zarządza danymi ról użytkowników w bazie danych.
17. UserRepository: Przechowuje i zarządza danymi użytkowników systemu w bazie danych.
18. GroupService: Zarządza logiką związaną z grupami.
19. EnrollmentService: Zarządza logiką związaną z zapisami studentów.
20. StudentService: Zarządza logiką związaną ze studentami.
21. TeacherService: Zarządza logiką związaną z nauczycielami.
22. CourseService: Zarządza logiką związaną z kursami.

23. GradeService: Zarządza logiką związaną z ocenami.
24. AttendanceService: Zarządza logiką związaną z frekwencją.
25. AuthenticationService: Zarządza uwierzytelnianiem użytkowników.
26. AuthorizationService: Zarządza autoryzacją użytkowników i nadawaniem uprawnień.
27. RoleService: Zarządza logiką związaną z rolami użytkowników.
28. UserService: Zarządza logiką związaną z użytkownikami systemu.
29. PasswordResetService: Zarządza procesem resetowania haseł użytkowników.
30. GroupController: Kontroler obsługujący operacje związane z grupami.
31. EnrollmentController: Kontroler obsługujący operacje związane z zapisami studentów.
32. StudentController: Kontroler obsługujący operacje związane ze studentami.
33. TeacherController: Kontroler obsługujący operacje związane z nauczycielami.
34. CourseController: Kontroler obsługujący operacje związane z kursami.
35. GradeController: Kontroler obsługujący operacje związane z ocenami.
36. AttendanceController: Kontroler obsługujący operacje związane z frekwencją.
37. SecurityController: Kontroler obsługujący operacje związane z bezpieczeństwem systemu.
38. RoleController: Kontroler obsługujący operacje związane z rolami użytkowników.
39. LoginController: Kontroler obsługujący proces logowania użytkowników.
40. RegistrationController: Kontroler obsługujący proces rejestracji nowych użytkowników.
41. PasswordResetController: Kontroler obsługujący proces resetowania haseł użytkowników.
42. GroupDTO: Obiekt transferu danych dla grup, zawierający podstawowe informacje o grupach.
43. EnrollmentDTO: Obiekt transferu danych dla zapisów studentów, zawierający podstawowe informacje o zapisach.
44. StudentDTO: Obiekt transferu danych dla studentów, zawierający podstawowe informacje o studentach.
45. TeacherDTO: Obiekt transferu danych dla nauczycieli, zawierający podstawowe informacje o nauczycielach.

46. CourseDTO: Obiekt transferu danych dla kursów, zawierający podstawowe informacje o kursach.
47. GradeDTO: Obiekt transferu danych dla ocen, zawierający podstawowe informacje o ocenach.
48. AttendanceDTO: Obiekt transferu danych dla frekwencji, zawierający podstawowe informacje o frekwencji.
49. SecurityDTO: Obiekt transferu danych dla bezpieczeństwa systemu, zawierający podstawowe informacje o bezpieczeństwie.
50. RoleDTO: Obiekt transferu danych dla ról użytkowników, zawierający podstawowe informacje o rolach.
51. UserDTO: Obiekt transferu danych dla użytkowników systemu, zawierający podstawowe informacje o użytkownikach.
52. LoginDTO: Obiekt transferu danych dla procesu logowania, zawierający podstawowe informacje o logowaniu.
53. RegistrationDTO: Obiekt transferu danych dla procesu rejestracji, zawierający podstawowe informacje o rejestracji.
54. PasswordResetDTO: Obiekt transferu danych dla procesu resetowania haseł, zawierający podstawowe informacje o resetowaniu haseł.
55. AddStudentComponent: Komponent umożliwiający dodanie nowego studenta.
56. RemoveUserModalComponent: Komponent umożliwiający usunięcie użytkownika.
57. UsersTableListComponent: Komponent wyświetlający wyniki wyszukiwania użytkowników.
58. UsersTableListItemComponent: Komponent wyświetlający pojedyńczy wiersz użytkownika.
59. StudentsListComponent: Komponent umożliwiający zarządzanie studentami.
60. TeachersListComponent: Komponent umożliwiający zarządzanie nauczycielami.
61. AddGroupComponent: Komponent umożliwiający dodanie nowej grupy.
62. RemoveGroupComponent: Komponent umożliwiający usunięcie grupy.
63. EditGroupComponent: Komponent umożliwiający edytowanie danych grupy.
64. CheckGroupsComponent: Komponent umożliwiający sprawdzanie danych grup.
65. GroupsListComponent: Komponent wyświetlający listę grup.
66. CoursesListComponent: Komponent umożliwiający zarządzanie kursami.
67. CourseTableListComponent: Komponent wyświetlający wyniki wyszukiwania kursów.

68. CourseTableListItemComponent: Komponent wyświetlający pojedyńczy wiersz kursu.
69. CreateCourseModalComponent: Komponent umożliwiający dodanie nowego kursu.
70. RemoveCourseModalComponent: Komponent umożliwiający usunięcie kursu.
71. AdminEnrollStudentComponent: Komponent umożliwiający administratorowi zapisanie studenta na kurs.
72. AdminUnenrollStudentComponent: Komponent umożliwiający administratorowi wypisanie studenta z kursu.
73. AdminDeleteAllGroupsComponent: Komponent umożliwiający administratorowi usunięcie wszystkich grup.
74. AddGradeComponent: Komponent umożliwiający dodanie nowej oceny.
75. EditGradeComponent: Komponent umożliwiający edytowanie istniejącej oceny.
76. DeleteGradeComponent: Komponent umożliwiający usunięcie oceny.
77. ViewTeacherCoursesComponent: Komponent wyświetlający kursy prowadzone przez nauczyciela.
78. ViewTeacherCoursesList: Komponent wyświetlający listę kursów prowadzących przez nauczyciela.
79. LoginComponent: Komponent obsługujący proces logowania użytkowników.
80. RegisterComponent: Komponent obsługujący proces rejestracji nowych użytkowników.
81. PasswordResetComponent: Komponent obsługujący proces resetowania haseł użytkowników.
82. StudentsService: Zarządza logiką związaną ze studentami.
83. TeachersService: Zarządza logiką związaną z nauczycielami.
84. CoursesService: Zarządza logiką związaną z kursami.
85. EnrollmentService: Zarządza logiką związaną z zapisami studentów.
86. LoginService: Zarządza logiką związaną z procesem logowania użytkowników.
87. RegisterService: Zarządza logiką związaną z procesem rejestracji nowych użytkowników.
88. PasswordResetService: Zarządza logiką związaną z procesem resetowania haseł użytkowników.
89. AddStudentStyles: Definiuje style i wygląd komponentów związanych z dodawaniem studentów.
90. RemoveUserModalStyles: Definiuje style i wygląd komponentów związanych z usuwaniem użytkowników.

91. UsersTableListStyles: Definiuje style i wygląd komponentów związanych z wyświetlaniem użytkowników.

92. UsersTablieListItemStyles: Definiuje style i wygląd komponentów związanych z pojedyńczym wierszem użytkownika.

93. TeachersListStyles: Definiuje style i wygląd komponentów związanych z wyświetlaniem i wyszukiwaniem nauczycieli.

94. StudentsListStyles: Definiuje style i wygląd komponentów związanych z wyświetlaniem i wyszukiwaniem użytkowników.

95. AddGroupStyles: Definiuje style i wygląd komponentów związanych z dodawaniem grup.

96. RemoveGroupStyles: Definiuje style i wygląd komponentów związanych z usuwaniem grup.

97. EditGroupStyles: Definiuje style i wygląd komponentów związanych z edytowaniem danych grup.

98. CheckGroupsStyles: Definiuje style i wygląd komponentów związanych ze sprawdzaniem danych grup.

99. GroupsListStyles: Definiuje style i wygląd komponentów związanych z wyświetlaniem grup.

100. CoursesListStyles: Definiuje style i wygląd komponentów związanych z wyszukiwaniem kursów.

101. CourseTableListStyles: Definiuje style i wygląd komponentów związanych z wyświetlaniem kursów.

102. CourseTableListItemStyles: Definiuje style i wygląd komponentów związanych z pojedyńczym wierszem kursu.

103. CreateCourseModalStyles: Definiuje style i wygląd komponentów związanych z dodawaniem kursów.

104. RemoveCourseModalStyles: Definiuje style i wygląd komponentów związanych z usuwaniem kursów.

105. AdminEnrollStyles: Definiuje style i wygląd komponentów związanych z zapisywaniem studentów na kursy przez administratora.

106. AdminUnenrollStyles: Definiuje style i wygląd komponentów związanych z wypisywaniem studentów z kursów przez administratora.

107. AdminDeleteAllGroupsStyles: Definiuje style i wygląd komponentów związanych z usuwaniem wszystkich grup przez administratora.

108. AddGradeStyles: Definiuje style i wygląd komponentów związanych z dodawaniem ocen.

109. EditGradeStyles: Definiuje style i wygląd komponentów związanych z edytowaniem ocen.

110. DeleteGradeStyles: Definiuje style i wygląd komponentów związanych z usuwaniem ocen.

111. ViewAttendanceStyles: Definiuje style i wygląd komponentów związanych z wyświetlaniem frekwencji studentów.

112. ViewTeacherCoursesStyles: Definiuje style i wygląd komponentów związanych z wyświetlaniem kursów prowadzonych przez nauczycieli.

113. LoginStyles: Definiuje style i wygląd komponentów związanych z logowaniem użytkowników.

114. RegisterStyles: Definiuje style i wygląd komponentów związanych z rejestracją nowych użytkowników.

115. PasswordResetStyles: Definiuje style i wygląd komponentów związanych z resetowaniem haseł użytkowników.

116. AuthenticationService: Zarządza uwierzytelnianiem użytkowników.

117. AuthorizationService: Zarządza autoryzacją użytkowników i nadawaniem uprawnień.

118. SecurityConfig: Konfiguruje ustawienia bezpieczeństwa systemu.

119. UserDetailsService: Zarządza szczegółami użytkowników w systemie.

120. JwtTokenProvider: Dostarcza tokeny JWT do uwierzytelniania użytkowników.

121. JwtTokenFilter: Filtruje żądania HTTP w celu sprawdzenia ważności tokenów JWT.

122. SecurityController: Kontroler obsługujący operacje związane z bezpieczeństwem systemu.

123. SecurityRepository: Przechowuje i zarządza danymi związanymi z bezpieczeństwem systemu w bazie danych.

124. SecurityDTO: Obiekt transferu danych dla bezpieczeństwa systemu, zawierający podstawowe informacje o bezpieczeństwie.

125. Role: Definiuje role użytkowników systemu, w tym ID roli, nazwę i opis.

126. RoleService: Zarządza logiką związaną z rolami użytkowników.

127. RoleRepository: Przechowuje i zarządza danymi ról użytkowników w bazie danych.

128. RoleController: Kontroler obsługujący operacje związane z rolami użytkowników.

129. RoleDTO: Obiekt transferu danych dla ról użytkowników, zawierający podstawowe informacje o rolach.

130. LoginController: Kontroler obsługujący proces logowania użytkowników.

131. RegistrationController: Kontroler obsługujący proces rejestracji nowych użytkowników.

132. PasswordResetController: Kontroler obsługujący proces resetowania haseł użytkowników.

133. UserService: Zarządza logiką związaną z użytkownikami systemu.

134. UserRepository: Przechowuje i zarządza danymi użytkowników systemu w bazie danych.

135. LoginDTO: Obiekt transferu danych dla procesu logowania, zawierający podstawowe informacje o logowaniu.

136. RegistrationDTO: Obiekt transferu danych dla procesu rejestracji, zawierający podstawowe informacje o rejestracji.

137. PasswordResetDTO: Obiekt transferu danych dla procesu resetowania haseł, zawierający podstawowe informacje o resetowaniu haseł.

6 Podział na Moduły i Pakiety

6.1 Moduł: DataBase

6.1.1 Pakiet: model

Pakiet `model` w module `DataBase` zawiera klasy reprezentujące kluczowe elementy systemu zarządzania uczelnią, takie jak `Student`, `Group`, `Course`. Każda z tych klas odpowiada za przechowywanie i zarządzanie danymi dotyczącymi studentów, kursów, grup, ocen, obecności. Ponadto, pakiet obejmuje zarządzanie użytkownikami, ich rolami i uprawnieniami, co umożliwia kontrolowanie dostępu do różnych funkcji systemu.

- Student
- Group
- Course
- Enrollment
- Grade
- Attendance
- User
- Teacher
- Enrollment
- Meeting
- Role

6.1.2 Pakiet: repository

Pakiet `repository` w module `DataBase` zawiera klasy odpowiedzialne za interakcje z bazą danych, związane z poszczególnymi obiektami systemu zarządzania uczelnią. Klasy takie jak `GroupRepository`, `EnrollmentRepository` czy `CourseRepository` umożliwiają operacje CRUD (tworzenie, odczyt, aktualizacja, usuwanie) na danych przechowywanych w bazie. Pakiet ten zapewnia również mechanizmy dostępu i zarządzania danymi dotyczącymi użytkowników, ról.

- AttendanceRepository
- CourseGroupRepository
- CourseRepository
- EnrollmentRepository
- GradeRepository
- MeetingRepository

- RoleRepository
- StudentRepository
- TeacherRepository
- UserRepository

6.2 Moduł: Backend

6.2.1 Pakiet: service

Pakiet `service` w module `Backend` zawiera klasy odpowiedzialne za logikę biznesową i operacje wykonywane na danych w systemie zarządzania uczelnią. Klasy takie jak `GroupService`, `EnrollmentService` obsługują procesy związane z zarządzaniem grupami, zapisami na kursy i innymi funkcjami systemu. Dodatkowo, pakiet ten zawiera serwisy odpowiedzialne za uwierzytelnianie, autoryzację, zarządzanie rolami i uprawnieniami użytkowników, a także resetowanie haseł.

- GroupService
- UserService
- EnrollmentService
- StudentService
- TeacherService
- CourseService
- GradeService
- AttendanceService
- AuthService
- MeetingService

6.2.2 Pakiet: controller

Pakiet `controller` w module `Backend` zawiera klasy odpowiedzialne za obsługę żądań HTTP i interakcję użytkownika z systemem zarządzania uczelnią. Klasy takie jak `GroupController`, `EnrollmentController` przetwarzają żądania dotyczące zarządzania grupami, zapisami na kursy, oraz innymi aspektami systemu. Pakiet ten obejmuje również kontrolery odpowiedzialne za uwierzytelnianie, rejestrację użytkowników, zarządzanie rolami i uprawnieniami.

- AuthController
- CourseController
- EnrollmenttController
- GradeController
- GroupController
- MeetingAttendanceController
- StudentController
- TeacherController
- UserController

6.2.3 Pakiet: Payload

Pakiet `Payload` w module `Backend` zawiera klasy reprezentujące obiekty transferu danych (DTO i Response), które są używane do przesyłania danych między różnymi warstwami aplikacji. Klasy takie jak `CreateCourseDTO`, `UpdateGroupDTO` służą do przenoszenia danych dotyczących grup, zapisów i innych elementów systemu zarządzania uczelnią. Pakiet ten zapewnia również DTO dla operacji związanych z użytkownikami, takimi jak logowanie i rejestracja.

- AttendanceRecordDTO
- CreateCourseDTO
- CreateGradeDTO
- CreateGroupDTO
- MeetingDTO
- LoginDTO
- RegisterDTO
- UpdateAttendanceDTO
- UpdateCourseDTO
- UpdateGradeDTO
- UpdateGroupDTO
- UpdateTeacherDTO

6.3 Moduł: Frontend

6.3.1 Pakiet: components

Pakiet `components` w module `Frontend` zawiera różnorodne komponenty UI, które umożliwiają interakcję użytkownika z systemem zarządzania uczelnią. Komponenty takie jak, `AddStudentComponent` są odpowiedzialne za wyświetlanie list, szczegółów, formularzy oraz interakcji związanych z studentami, nauczycielami, grupami, kursami. Dodatkowo, pakiet ten zawiera komponenty odpowiedzialne za zarządzanie użytkownikami, takie jak logowanie, rejestracja i resetowanie haseł.

- AddStudentComponent
- RemoveUserModalComponent
- UsersTableListComponent
- UsersTablieListItemComponent
- TeachersListComponent
- StudentsListComponent

- AddGroupComponent
- RemoveGroupComponent
- EditGroupComponent
- CheckGroupsComponent
- GroupsListComponent
- CoursesListComponent
- CourseTableListComponent
- CourseTableListItemComponent
- CreateCourseModalComponent
- RemoveCourseModalComponent
- AdminEnrollStudentComponent
- AdminUnenrollStudentComponent
- AdminDeleteAllGroupsComponent
- AddGradeComponent
- EditGradeComponent
- DeleteGradeComponent
- ViewTeacherCoursesComponent
- ViewTeacherCoursesList
- LoginComponent
- RegisterComponent
- PasswordResetComponent

6.3.2 Pakiet: services

Pakiet `services` w module `Frontend` zawiera klasy odpowiedzialne za logikę biznesową i komunikację z backendem. Usługi takie jak `StudentService` obsługują operacje związane z danymi studentów. Dodatkowo, pakiet obejmuje frekwencją oraz obsługą użytkowników, w tym logowaniem, rejestracją i resetowaniem haseł.

- StudentsService
- TeachersService
- GroupsService
- CoursesService

- EnrollmentService
- AttendanceReportService
- LoginService
- RegisterService
- PasswordResetService

6.3.3 Pakiet: styles

Pakiet `styles` w module `Frontend` zawiera klasy odpowiedzialne za stylizację poszczególnych komponentów UI w systemie zarządzania uczelnią. Każda z klas, takich jak `AddStudentStyles`, definiuje wygląd i układ elementów interfejsu użytkownika związanych z studentami, nauczycielami, grupami, kursami oraz innymi funkcjami systemu. Pakiet ten zapewnia spójny i estetyczny wygląd aplikacji poprzez centralne zarządzanie stylami.

- AddStudentStyles
- RemoveUserModalStyles
- UsersTableListStyles
- UsersTablieListItemStyles
- TeachersListStyles
- StudentsListStyles
- AddGroupStyles
- RemoveGroupStyles
- EditGroupStyles
- CheckGroupsStyles
- GroupsListStyles
- CoursesListStyles
- CourseTableListStyles
- CourseTableListItemStyles
- CreateCourseModalStyles
- RemoveCourseModalStyles
- AdminEnrollStyles
- AdminUnenrollStyles
- AdminDeleteAllGroupsStyles

- AddGradeStyles
- EditGradeStyles
- DeleteGradeStyles
- ViewAttendanceStyles
- ViewTeacherCoursesStyles
- LoginStyles
- RegisterStyles
- PasswordResetStyles

6.4 Moduł: Security

6.4.1 Pakiet: security

Pakiet `security` w module `Security` zawiera klasy i komponenty odpowiedzialne za zapewnienie bezpieczeństwa w systemie zarządzania uczelnią. Klasy takie jak `AuthenticationService` i `AuthorizationService` obsługują uwierzytelnianie i autoryzację użytkowników. Dodatkowo, pakiet zawiera konfiguracje bezpieczeństwa (`SecurityConfig`), usługi użytkownika (`UserDetailsService`), mechanizmy zarządzania tokenami JWT (`JwtTokenProvider` i `JwtTokenFilter`), a także kontrolery (`SecurityController`), repozytoria (`SecurityRepository`) i obiekty transferu danych (`SecurityDTO`) związane z bezpieczeństwem.

- AuthenticationService
- AuthorizationService
- SecurityConfig
- UserDetailsService
- JwtTokenProvider
- JwtTokenFilter
- SecurityController
- SecurityRepository
- SecurityDTO

6.4.2 Pakiet: roles

Pakiet `roles` w module `Security` zawiera klasy i komponenty odpowiedzialne za zarządzanie rolami i uprawnieniami w systemie zarządzania uczelnią. Klasy takie jak `Role` reprezentują role i uprawnienia użytkowników. Dodatkowo, pakiet zawiera serwisy (`RoleService`), repozytoria (`RoleRepository`), kontrolery (`RoleController`), oraz obiekty transferu danych (`RoleDTO`), które umożliwiają zarządzanie rolami i uprawnieniami w systemie.

- Role
- RoleService
- RoleRepository
- RoleController
- RoleDTO

6.4.3 Pakiet: user-management

Pakiet `user-management` w module `Security` zawiera klasy i komponenty odpowiedzialne za zarządzanie użytkownikami w systemie zarządzania uczelnią. Klasy takie jak `UserService` i `UserRepository` obsługują operacje związane z użytkownikami, takie jak logowanie, rejestracja i resetowanie haseł. Dodatkowo, pakiet zawiera kontrolery (`LoginController`, `RegistrationController` i `PasswordResetController`) oraz obiekty transferu danych (`LoginDTO`, `RegistrationDTO` i `PasswordResetDTO`), które umożliwiają zarządzanie danymi użytkowników i procesami autoryzacyjnymi w systemie.

- LoginController
- RegistrationController
- PasswordResetController
- UserService
- UserRepository
- LoginDTO
- RegistrationDTO
- PasswordResetDTO

7 Komunikacja między modułami

7.1 Komunikacja między modułami DataBase i Backend

7.1.1 Interfejsy Repository i Service

Moduł *DataBase* zawiera pakiety *model* i *repository*, które odpowiadają za dostęp do danych i zarządzanie nimi. Moduł *Backend* zawiera pakiet *service*, który wykorzystuje interfejsy z pakietu *repository* do operacji na danych.

- **Repository**: Każdy *repository* (np. *StudentRepository*, *CourseRepository*) zapewnia metody do pobierania, zapisywania, aktualizacji i usuwania danych w bazie danych.
- **Service**: Serwisy (np. *StudentService*, *CourseService*) wywołują metody z odpowiednich *repository*, aby wykonywać operacje biznesowe. Serwisy zapewniają logikę biznesową i zarządzają transakcjami.

7.2 Komunikacja między modułami Backend i Frontend

7.2.1 Interfejsy Controller i DTO

Moduł *Backend* zawiera pakiety *controller* i *DTO*, które są kluczowe dla komunikacji z modułem *Frontend*.

- **Controller**: Kontrolery (np. *StudentController*, *CourseController*) obsługują żądania HTTP przychodzące z interfejsu użytkownika (Frontend). Przyjmują dane, przetwarzają je i zwracają odpowiednie odpowiedzi.
- **DTO**: Obiekty transferu danych (DTO) służą do przenoszenia danych między warstwami. DTO są używane przez kontrolery do odbierania danych z Frontendu i wysyłania odpowiedzi.

7.2.2 Komponenty Frontendu

Moduł *Frontend* zawiera pakiety *components* i *services*, które komunikują się z kontrolerami w module *Backend*.

- **Components**: Komponenty (np. *StudentsListComponent*, *CourseTableListItemComponent*) reprezentują interfejs użytkownika i wyświetlają dane użytkownikowi. Wywołują metody serwisów, aby komunikować się z Backendem.
- **Services**: Serwisy (np. *StudentsService*, *CoursesService*) w module Frontend wysyłają żądania HTTP do kontrolerów w module Backend, pobierają dane i zwracają je do komponentów.

7.3 Moduł Security

Moduł *Security* zarządza autoryzacją i uwierzytelnianiem użytkowników oraz ochroną zasobów aplikacji.

- **AuthService**: Zarządza uprawnieniami użytkowników do zasobów oraz zapewnia logikę uwierzytelniania użytkowników.
- **SecurityConfig**: Konfiguruje zabezpieczenia aplikacji, takie jak filtry JWT.
- **UserDetailsService**: Ładuje dane użytkowników potrzebne do uwierzytelnienia.

Komunikacja między modułem *Security* a innymi modułami odbywa się poprzez mechanizmy uwierzytelniania i autoryzacji. Przykładowo, *SecurityController* może obsługiwać żądania logowania i rejestracji, a następnie komunikować się z *UserService* w celu zarządzania użytkownikami i rolami.

8 Kolaboracja Klas w Systemie Rejestracj Uczelni

8.1 Moduł: DataBase

8.1.1 Pakiet: model

Student:

- Z Group: **Group.addStudent(student)** i **Group.removeStudent(student)**.
- Z Course: **Course.addStudent(student)** i **Course.removeStudent(student)**.
- Z Grade: **Grade.updateGrade(value)**.
- Z Attendance: **Attendance.markAttendance(status)**.

Group:

- Z Student: **Student.enrollInCourse(course)**.
- Z Course: **Course.assignToGroup(group)**.

Course:

- Z Student: **Student.enrollInCourse(course)** i **Student.dropCourse(course)**.
- Z Group: **Group.assignCourse(course)**.
- Z Teacher: **Teacher.assignCourse(course)**.

Enrollment:

- Z Student: **Student.enrollInCourse(course)**.
- Z Course: **Course.enrollStudent(student)**.

Grade:

- Z Student: **Student.receiveGrade(grade)**.

Attendance:

- Z Student: **Student.markAttendance(attendance)**.
- Z Course: **Course.recordAttendance(attendance)**.

User:

- Z Role: **Role.assignToUser(user)**.

Role:

- Z User: **User.assignRole(role)**.

8.1.2 Pakiet: repository

StudentRepository:

- Z Student: **save(student)** i **findById(studentId)**.
- Z Group: **Group.findByStudent(student)**.

CourseRepository:

- Z Course: **save(course)** i **findById(courseId)**.
- Z Group: **Group.findByCourse(course)**.

GroupRepository:

- Z Group: **save(group)** i **findById(groupId)**.
- Z Student: **Student.findByGroup(group)**.

8.2 Moduł: Backend

8.2.1 Pakiet: service

StudentService:

- Z StudentRepository: **StudentRepository.findById(studentId)** i **StudentRepository.save(student)**.

CourseService:

- Z CourseRepository: **CourseRepository.findById(courseId)** i **CourseRepository.save(course)**.

EnrollmentService:

- Z StudentRepository: **StudentRepository.findById(studentId)** i **StudentRepository.save(student)**.
- Z CourseRepository: **CourseRepository.findById(courseId)** i **CourseRepository.save(course)**.
- Z EnrollmentRepository: **EnrollmentRepository.save(enrollment)** i **EnrollmentRepository.delete(enrollment)**.

GradeService:

- Z GradeRepository: **GradeRepository.save(grade)** i **GradeRepository.findByStudent(studentId)**.

8.2.2 Pakiet: controller

StudentController:

- Z StudentService: **StudentService.getStudentDetails(studentId)** i **StudentService.updateStudentDetails(studentId, details)**.

CourseController:

- Z CourseService: **CourseService.getCourseDetails(courseId)** i **CourseService.addCourse(course)**.

EnrollmentController:

- Z EnrollmentService: **EnrollmentService.enrollStudent(studentId, courseId)** i **EnrollmentService.unenrollStudent(studentId, courseId)**.

8.3 Moduł: Frontend

8.3.1 Pakiet: components

AddStudentComponent:

- Z StudentService: **StudentService.saveStudentDetails(studentDetails)**.

ViewStudentDetailsComponent:

- Z StudentService: **StudentService.fetchStudentDetails(studentId)**.

CourseListComponent:

- Z CourseService: **CourseService.getAllCourses()**.

EnrollmentComponent:

- Z EnrollmentService: **EnrollmentService.enrollStudent(studentId, courseId)**.

8.3.2 Pakiet: services

StudentService:

- Z Backend: **fetchStudentDetails(id)** i **saveStudentDetails(details)**.

CourseService:

- Z Backend: **fetchCourseDetails(id)** i **saveCourseDetails(details)**.

EnrollmentService:

- Z Backend: **enrollStudent(studentId, courseId)** i **unenrollStudent(studentId, courseId)**.

8.4 Moduł: Security

8.4.1 Pakiet: security

AuthenticationService:

- Z UserRepository: **UserRepository.findByUsername(username)**.

AuthorizationService:

- Z User: **User.hasPermission(permission)**.

8.4.2 Pakiet: roles

RoleService:

- Z RoleRepository: **RoleRepository.save(role)**
  i **RoleRepository.findById(roleId)**.

8.4.3 Pakiet: user-management

LoginController:

- Z AuthenticationService: **AuthenticationService.login(credentials)**
  i **AuthenticationService.logout()**.

RegistrationController:

- Z UserService: **UserService.registerUser(userDetails)**.

9 Współpraca między komponentami

9.1 EnrollmentService

Interakcje:

- Z StudentRepository do wyszukiwania
  i zapisywania studentów.
- Z CourseRepository do wyszukiwania
  i zapisywania kursów.
- Z EnrollmentRepository do zapisywania
  i usuwania zapisów.

Proces:

Kiedy student zapisuje się na kurs, EnrollmentService wywołuje metody w StudentRepository
i CourseRepository, aby zaktualizować odpowiednie dane. Następnie tworzy nowy zapis w EnrollmentRepository.

9.2 StudentController

Interakcje:

- Z StudentService do pobierania danych studenta.

Proces:

Kiedy klient wysyła żądanie dotyczące studenta, StudentController przekazuje to żądanie do StudentService, który następnie komunikuje się z StudentRepository, aby pobrać lub zaktualizować dane studenta.

9.3 Frontend Components

Interakcje:

- Z StudentsService do pobierania i zapisywania danych studenta.
- Z CoursesService do pobierania danych kursów.
- Z EnrollmentService do zapisania studentów na kursy.

Proces:

Komponenty frontendowe, takie jak AddStudentComponent i ViewStudentDetailsComponent, używają serwisów do komunikacj i z backendem, aby pobrać lub zapisać odpowiednie dane.

9.4 AuthenticationService

Interakcje:

- Z UserRepository do wyszukiwania użytkowników na podstawie nazwy użytkownika.

Proces:

Kiedy użytkownik próbuje się zalogować, AuthenticationService używa UserRepository, aby znaleźć użytkownika i sprawdzić poprawność danych logowania.

10 Podział Zadań - Etap I

Backend

Kacper Machnik: Warstwa Serwisowa (Logika Biznesowa)

- Implementacja **interfejsów i klas Serwisów** (‘xxxService‘, ‘xxxServiceImpl‘) w pakiecie ‘service‘.
- Implementacja głównej **logiki biznesowej** aplikacji (operacje CRUD, walidacja biznesowa).
- Implementacja logiki **wyszukiwania i filtrowania** danych (np. budowanie ‘Specification‘). Implementacja mapowania między DTO a encjami oraz między encjami a obiektami Response.
- Rzucanie **niestandardowych wyjątków** biznesowych i obsługa błędów.
- Implementacja (z Michałem) definicji **niestandardowych zapytań** (‘@Query‘) w repozytoriach dla optymalizacji (np. ‘JOIN FETCH‘).

Piotr Waluszek: Bezpieczeństwo i Zarządzanie Tożsamością

- Implementacja i konfiguracja **Spring Security** (uwierzytelnianie, np. JWT lub sesje).
- Implementacja ‘UserDetailsService‘ i zarządzanie hasłami.
- Implementacja **rejestracji użytkowników** (‘POST /api/users/register‘, ‘UserService‘) - w tym tworzenie profili Student/Teacher.
- Implementacja logowania (‘AuthController‘, ‘AuthService‘).
- Definicja i implementacja **niestandardowych serwisów bezpieczeństwa** (‘xxxSecurityService‘) do logiki ‘@PreAuthorize‘.
- Zarządzanie rolami (‘Role‘, ‘RoleType‘) i ich inicjalizacja.
- Stosowanie adnotacji **‘@PreAuthorize‘** do zabezpieczania metod API (z Kacprem).

Michał Krzempek: Warstwa API i Model Danych

- Definicja **encji JPA** (‘@Entity‘) w pakiecie ‘model‘.
- Definicja obiektów **DTO** (wejściowe) i **Response** (wyjściowe) w pakiecie ‘payload‘.
- Implementacja wszystkich **Kontrolerów REST** (‘@RestController‘) w pakiecie ‘controller‘.
- Definiowanie ścieżek API, obsługa parametrów, ciał żądań/odpowiedzi.
- Podstawowa **walidacja DTO** (‘@Valid‘ i adnotacje).

- Definicja **interfejsów Repozytoriów** (‘JpaRepository‘) w pakiecie ‘repository‘, w tym sygnatur metod oraz zapytań SQL (’@Query’).

Frontend

Patryk Przybysz: Komponenty Domenowe i Modele Danych

- Implementacja głównych komponentów React dla poszczególnych domen (students, teachers, users).
- Definicja i utrzymanie interfejsów/typów **TypeScript** dla wszystkich domen.
- Integracja i wykorzystanie wspólnych komponentów w tworzonych widokach domenowych.
- Implementacja logiki wyświetlania danych pobranych ze stanu aplikacji (Redux).
- Współpraca z Tomkiem przy integracji danych z API i Reduxa w komponentach.
- Podstawowe testy jednostkowe/komponentowe dla tworzonych widoków.

Artur Sojka: Layout i Wspólne Komponenty UI (Shared)

- Implementacja i utrzymanie głównego **układu aplikacji** (layoutu).
- Tworzenie i utrzymanie **współdzielonych, reużywalnych komponentów UI** (formularze, tabele, modale, obsługa ładowania/błędów).
- Implementacja **globalnych styli** i zapewnienie spójności wizualnej
- Dbanie o responsywność (RWD) aplikacji.
- Współpraca z Patrykiem i Tomkiem w celu zapewnienia, że komponenty domenowe używają wspólnych elementów UI.

Tomasz Madeja: Zarządzanie Stanem (Redux), API i Logika Aplikacji

- Konfiguracja i zarządzanie **store’m Reduxa**.
- Tworzenie i utrzymanie **slice’ów Reduxa** dla poszczególnych domen (akcje, reducery, selektory).
- Implementacja logiki **komunikacji z API backendu**.
- Obsługa stanu ładowania danych i błędów API.
- Implementacja **komponentów związanych z autoryzacją** (Login, Register) oraz integracja z Redux/API.
- Tworzenie i utrzymanie **niestandardowych hooków** Reacta.

- Integracja logiki Reduxa i wywołań API z komponentami tworzonymi przez Patryka.
- Zarządzanie routingiem aplikacji.