Aplikacja do zarządzania szkołą językową – projekt MAS

Bartłomiej Białkowski s27335

Spis treści

Spis obrazków

Cel systemu

Celem systemu jest wsparcie działania szkoły językowej w celu skomputeryzowania systemu zarządzania szkołą językową.

Wymagania użytkownika

System będzie przechowywał informację dotyczące uczniów i pracowników(w szczególności Lektorów, Kierowników i Administratorów). Studenci będą przypisywani do grup, które będą uczone przez Lektorów. Grupy mogą realizować programy. Administratorzy zarządzają budynkami, w których są pokoje. Pokoje są zajmowane przez lekcję w konkretnych godzinach.

System będzie używany przez:

- Kierowników
- Lektorów
- Uczniów

Wymagania funkcjonalne:

Kierownik może:

- Dodawać grupy, uczniów i lektorów
- Wyświetlać wszystkie grupy, uczniów, lektorów i plan lekcji
- Tworzyć nowe lekcje

Lektor może:

- Wyświetlać plan lekcji
- Dodawać tematy do lekcji

Uczeń może:

- Wyświetlać plan lekcji

Wymagania niefunkcjonalne

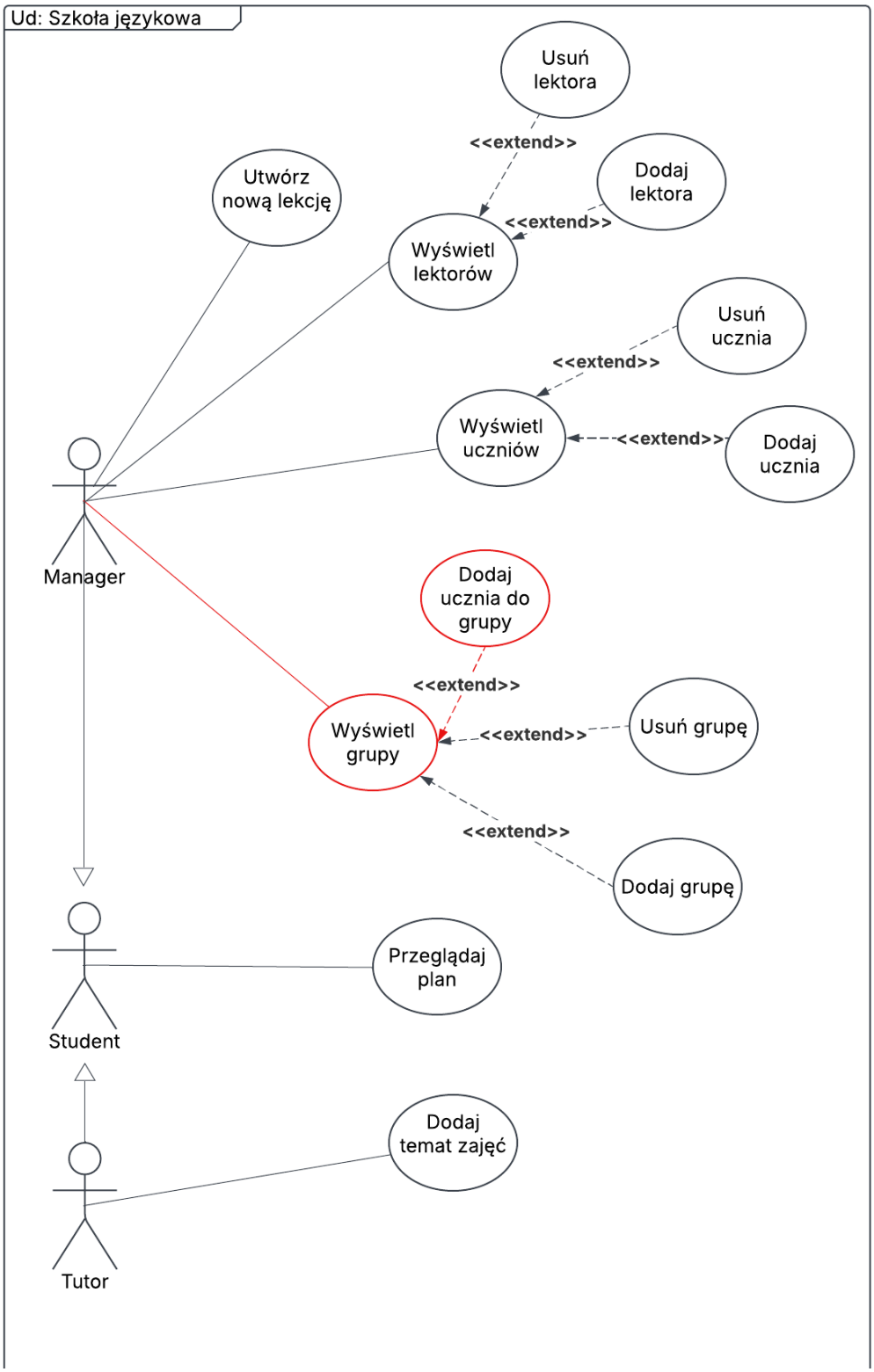
Cecha: Łatwość użytkowania

Miara: Czas niezbędny dla przeszkolenia przyszłych użytkowników systemu: 30 min

Cecha: Spójność elementów wizualnych z misją

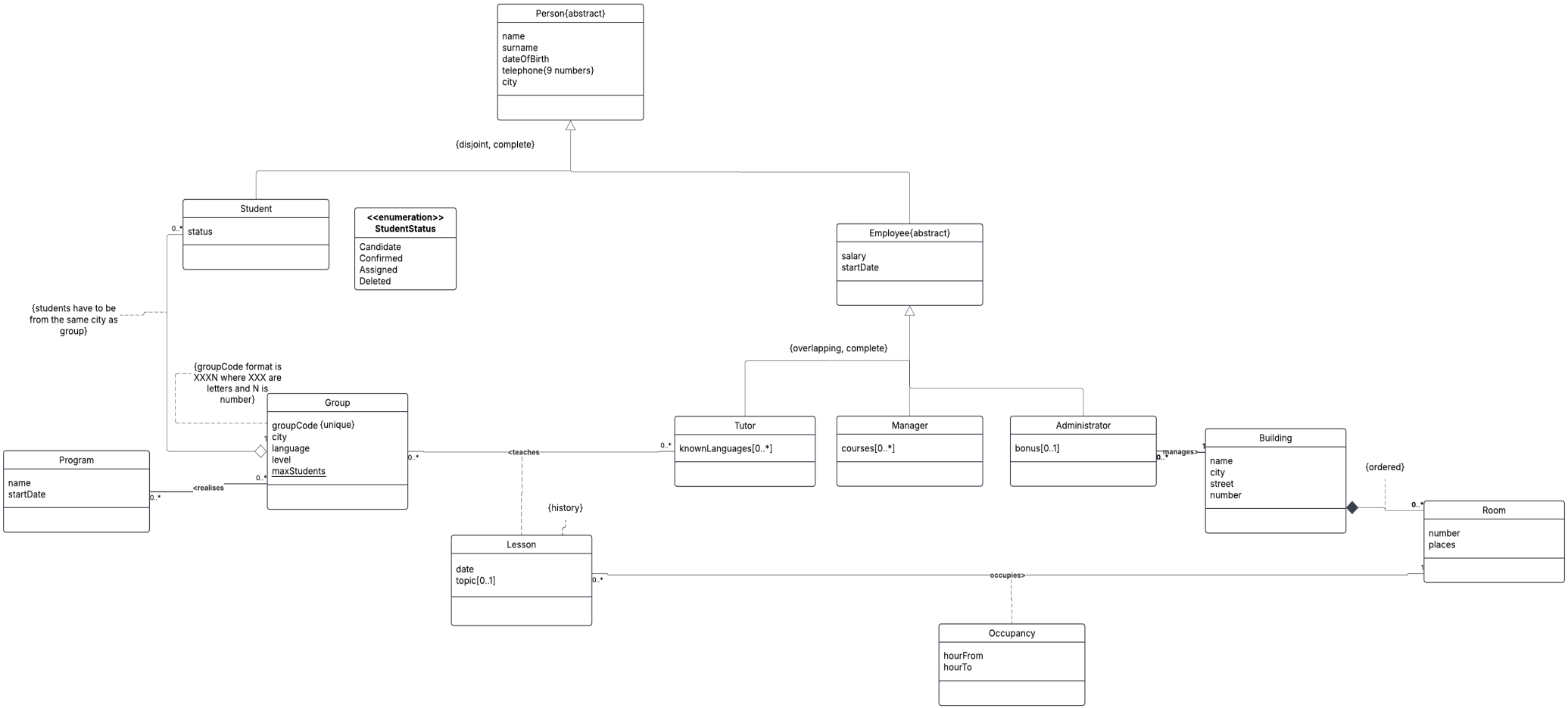
Miara: Interfejs utrzymany w kolorystyce: czerwonej, białej i niebieskiej.

Diagram przypadków użycia



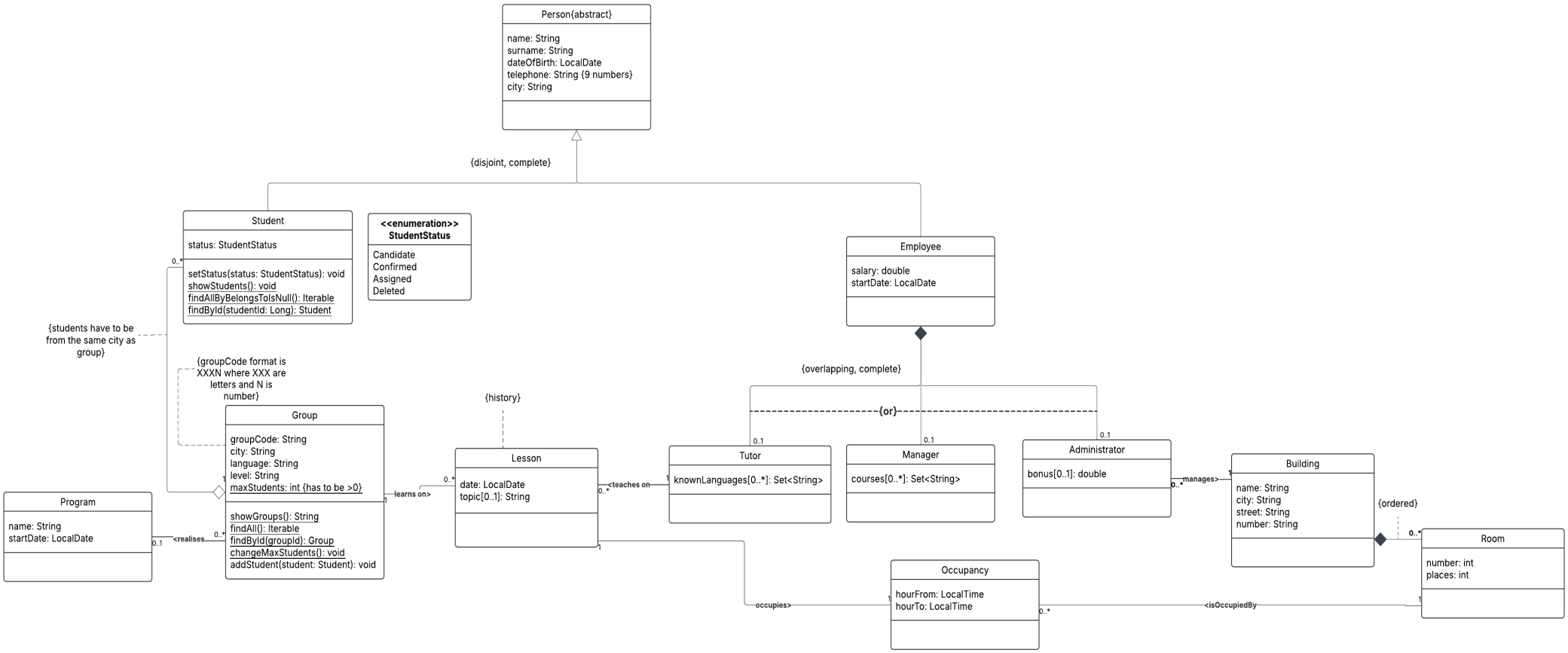
Rysunek 1 diagram przypadków użycia – na czerwono przypadek użycia z GUI

Diagram klas analityczny



Rysunek 2 Diagram klas analityczny

Diagram klas projektowy



Rysunek 3 Diagram klas projektowy

Scenariusz przypadku użycia

Nazwa przypadku użycia: Wyświetl grupy

Opis: Wyświetlenie listy grup

Aktor główny: Manager

Warunki początkowe: Jest przynajmniej jedna grupa w systemie

Wyniki końcowe: Została wyświetlona przynajmniej jedna grupa

Scenariusz:

1. Manager uruchamia przypadek użycia klikając show groups
2. System wyświetla listę grup

Nazwa przypadku użycia: Dodaj ucznia do grupy

Opis: Dodanie ucznia do grupy

Aktor główny: Manager

Warunki początkowe: Uczeń został dodany do systemu

Wyniki końcowe: Uczeń został dodany do grupy z tego samego miasta

Scenariusz:

1. Manager wybiera konkretną grupę z listy i uruchamia przypadek użycia klikając add student

2. System wyświetla listę studentów

3. Manager wybiera ucznia i klika add

4. System dodaje studenta do grupy i wyświetla komunikat o pozytywnym dodaniu

Scenariusz alternatywny:

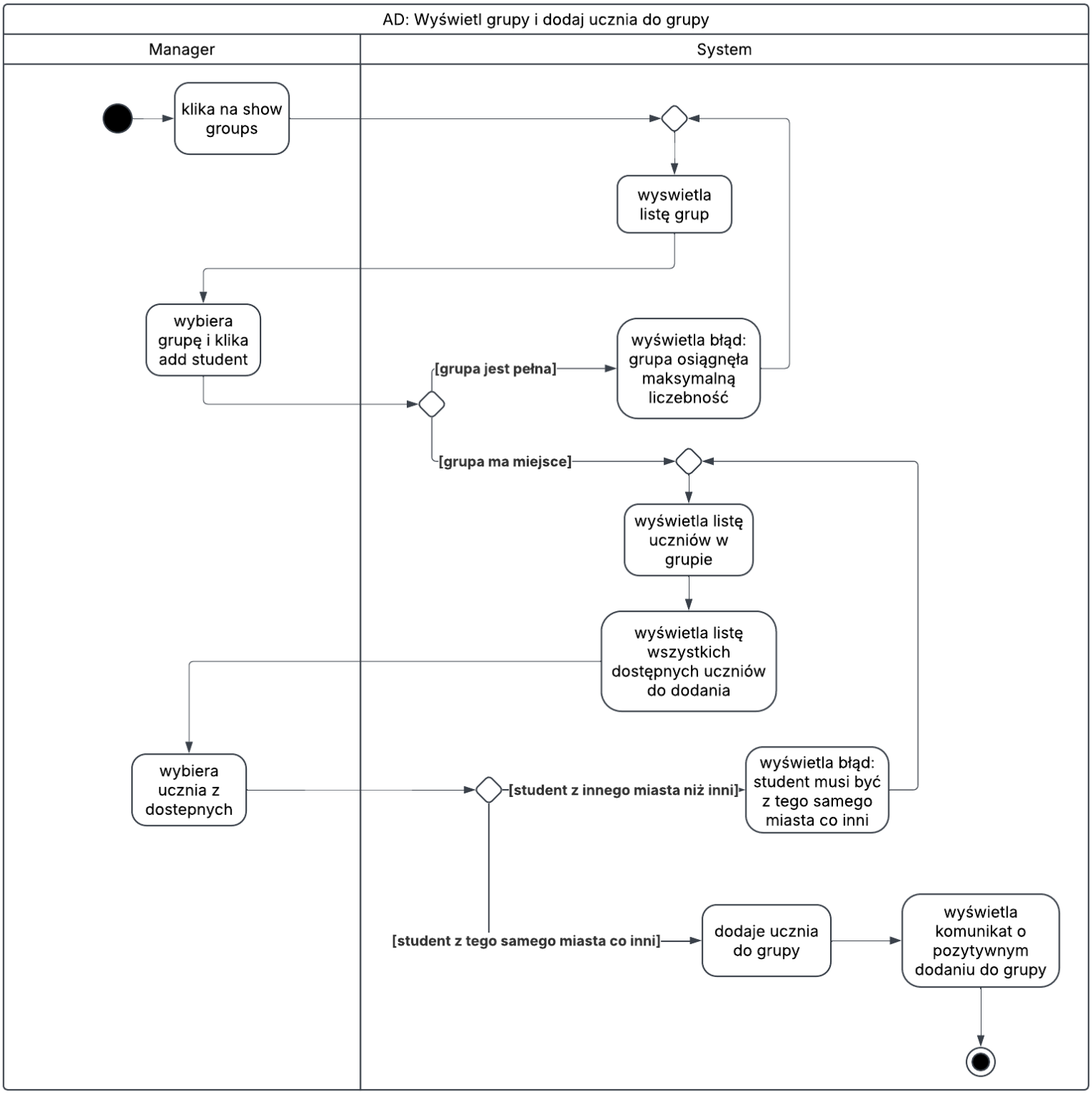
1a. Manager wybiera pełną grupę

1b. System wyświetla błąd: pełna grupa – powrót do punktu 1

3a. Manager wybiera ucznia z innego miasta niż grupa

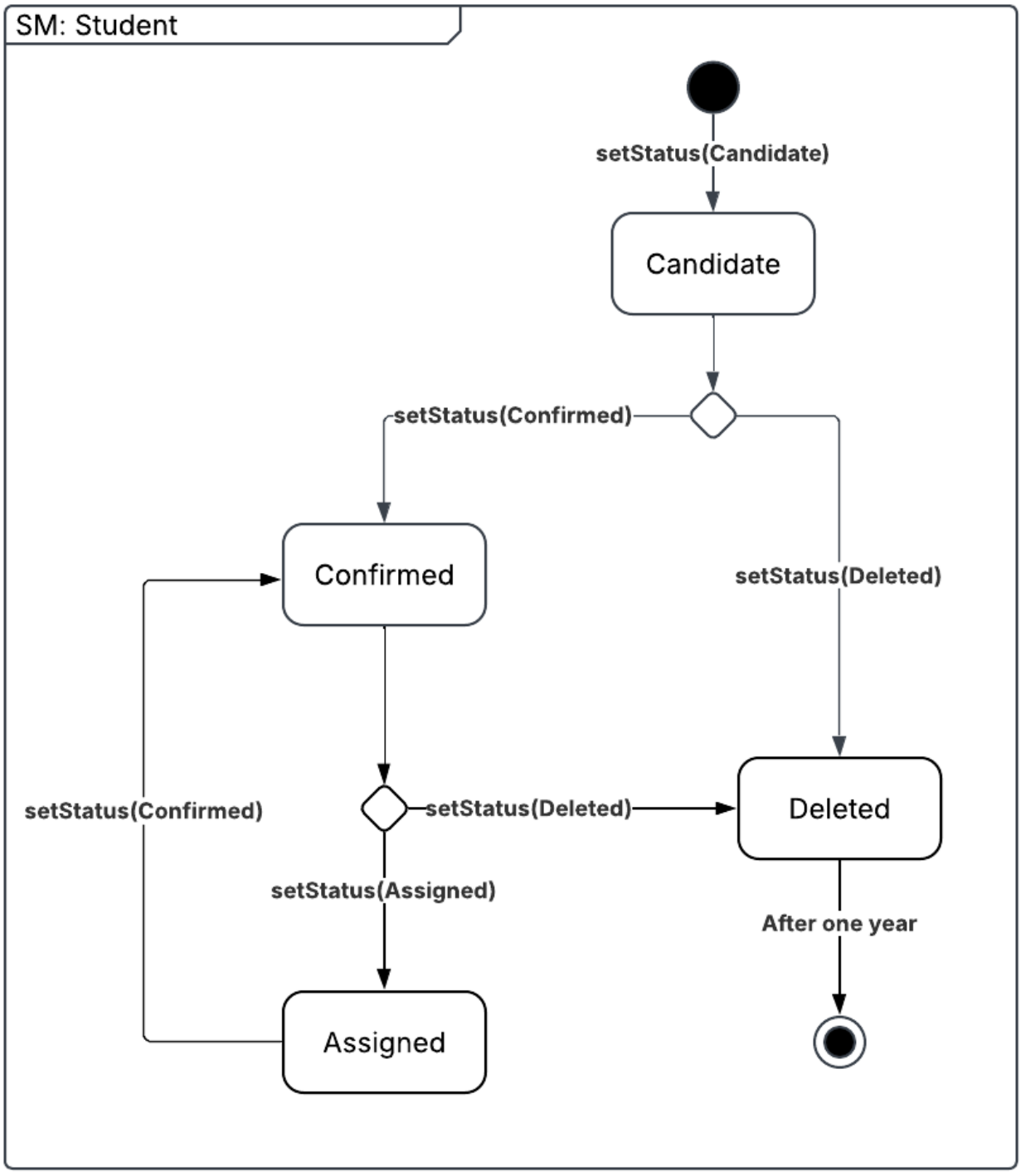
3b. System wyświetla błąd: student z innego miasta – powrót do punktu 3

Diagram aktywności



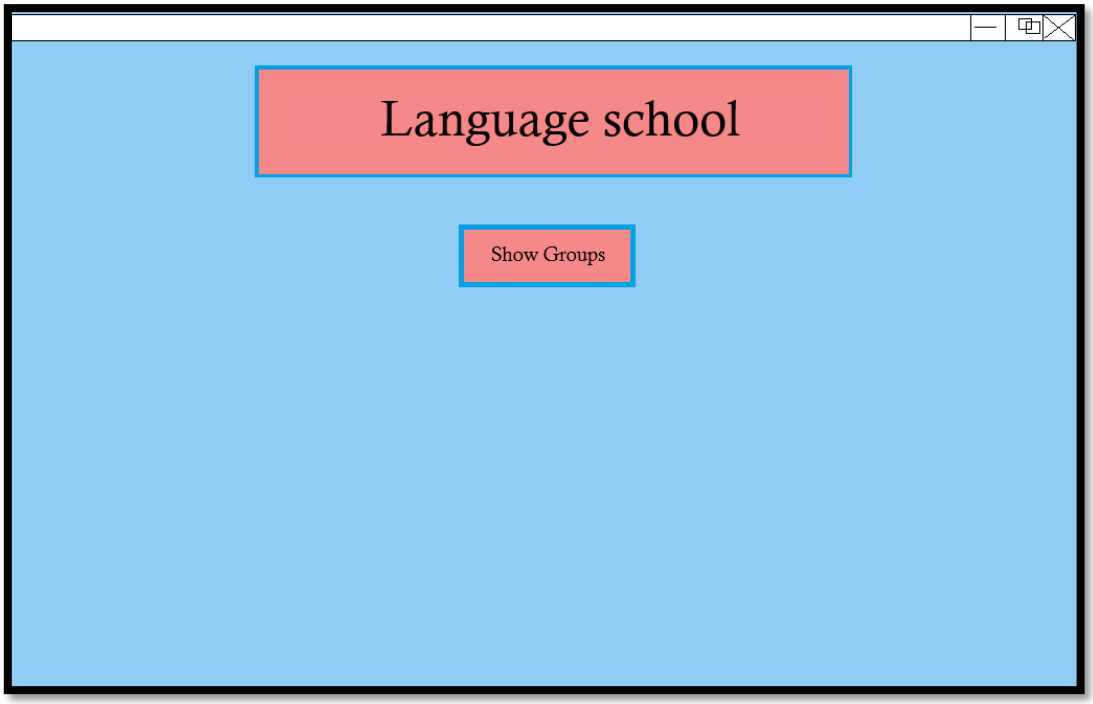
Rysunek 4 Diagram aktywności

Diagram stanu

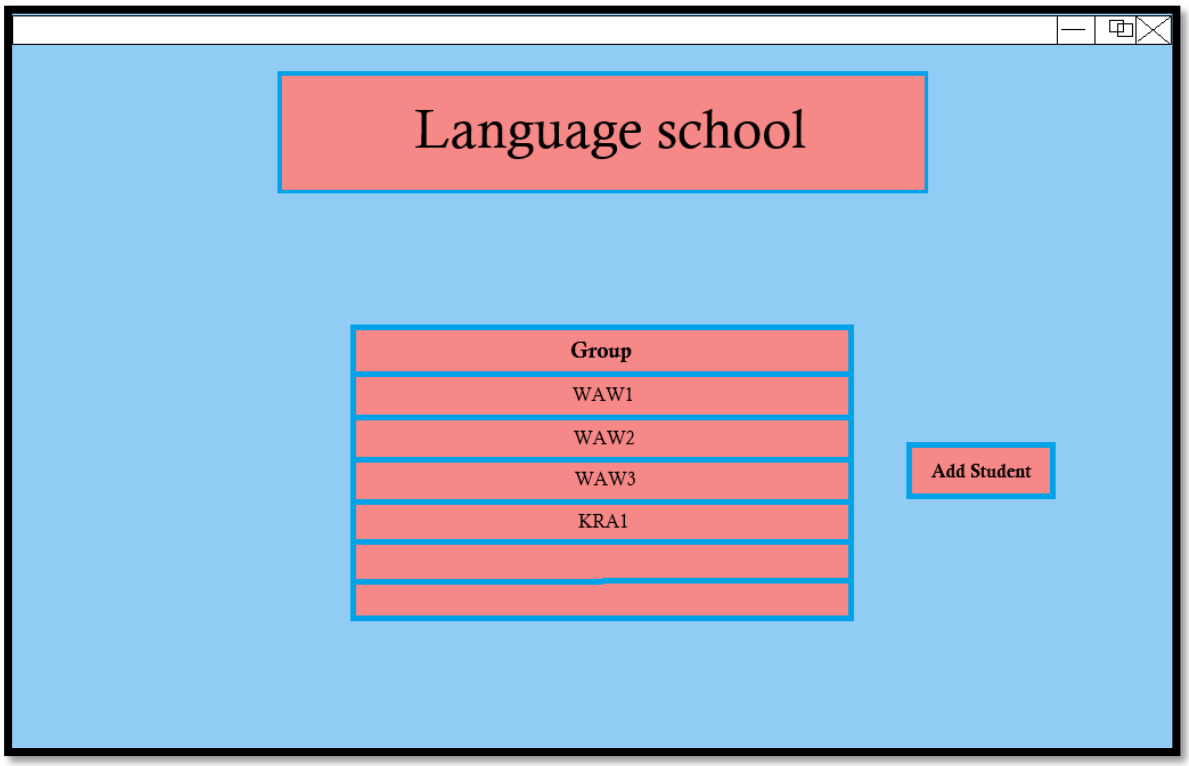


Rysunek 5 Diagram stanów

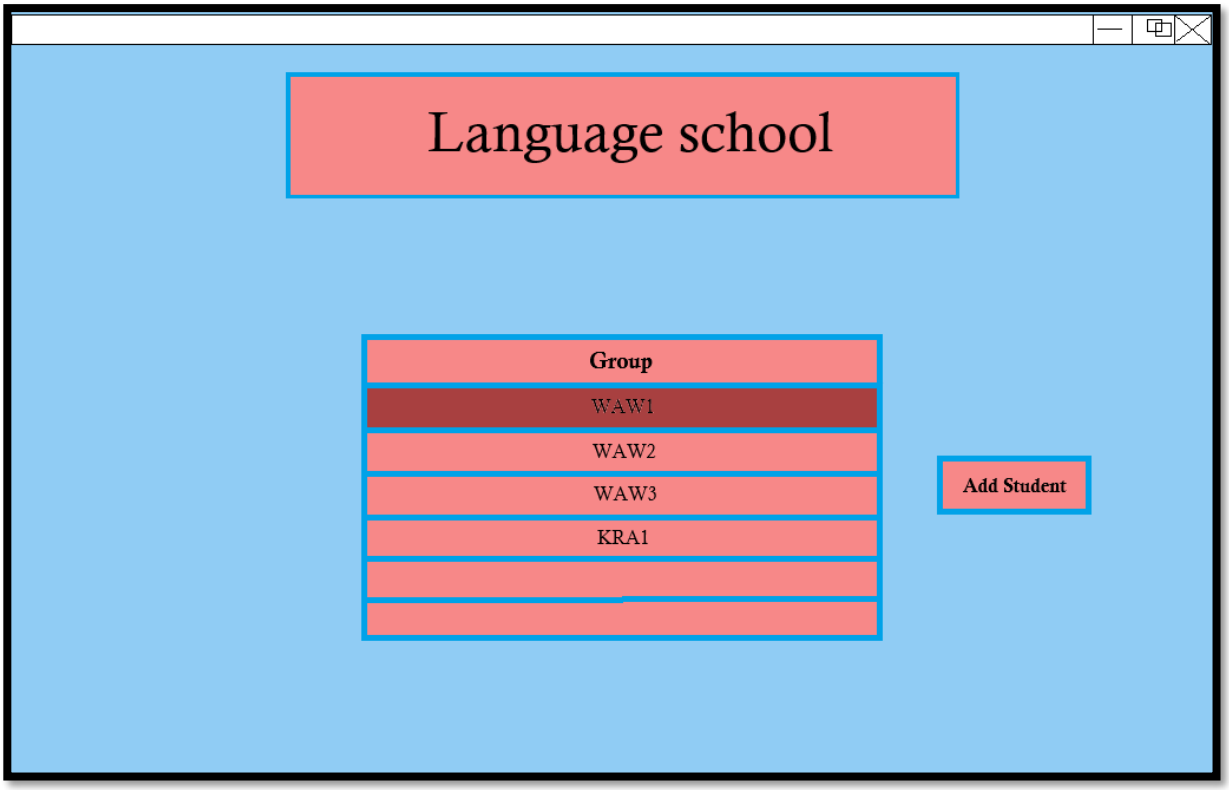
Projekt GUI



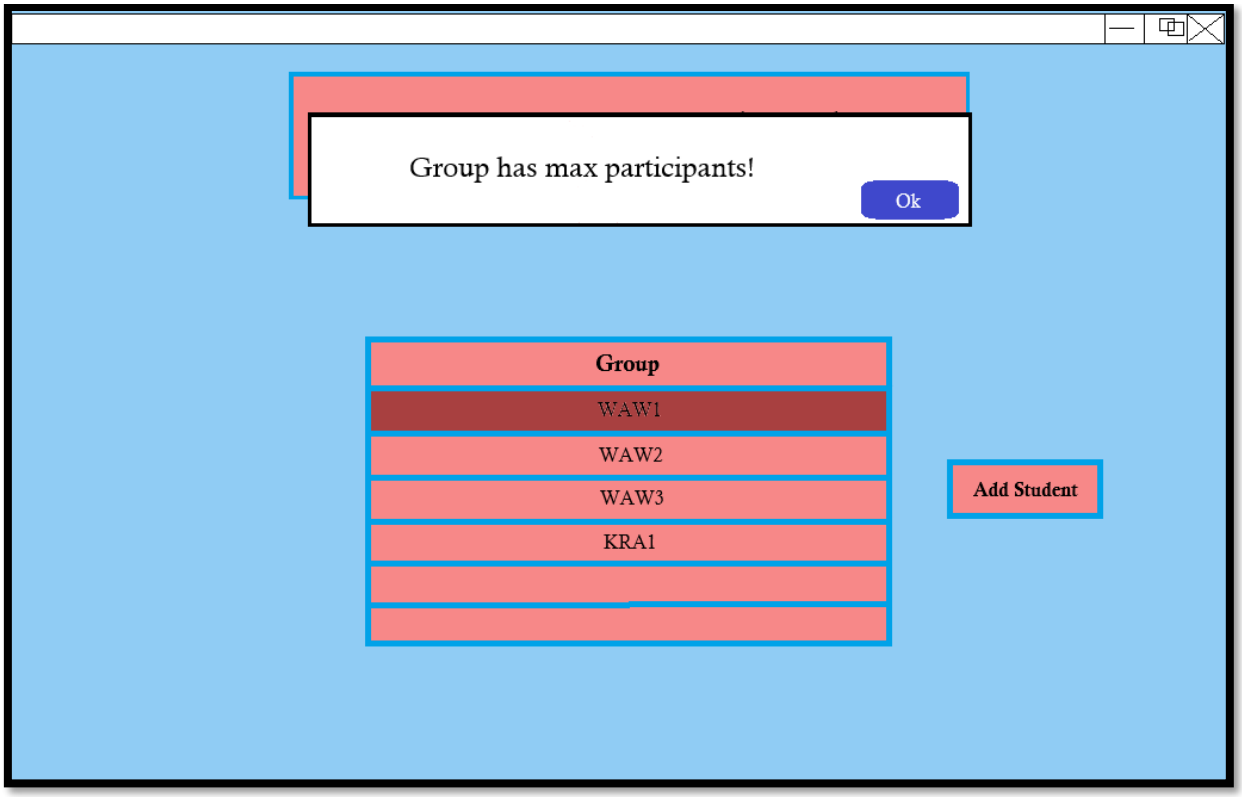
Rysunek 7 GUI Strona początkowa



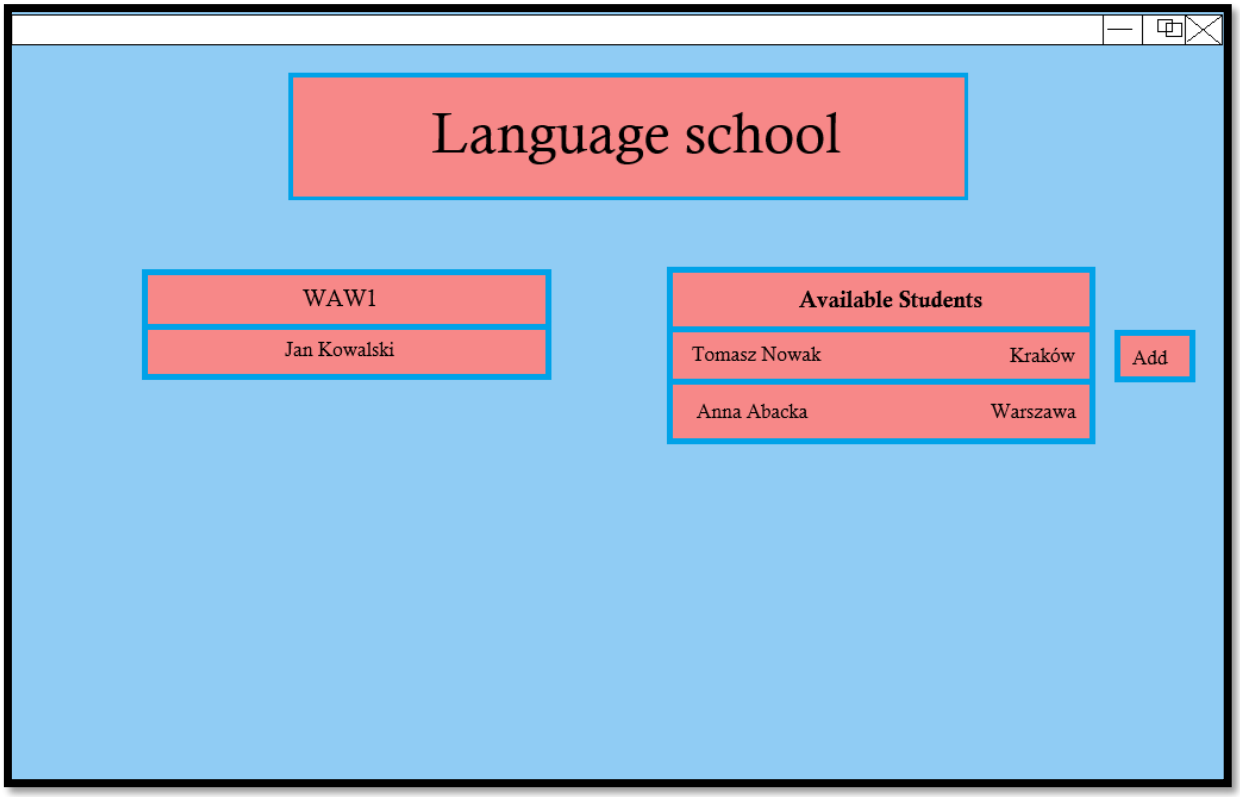
Rysunek 6 GUI lista grup



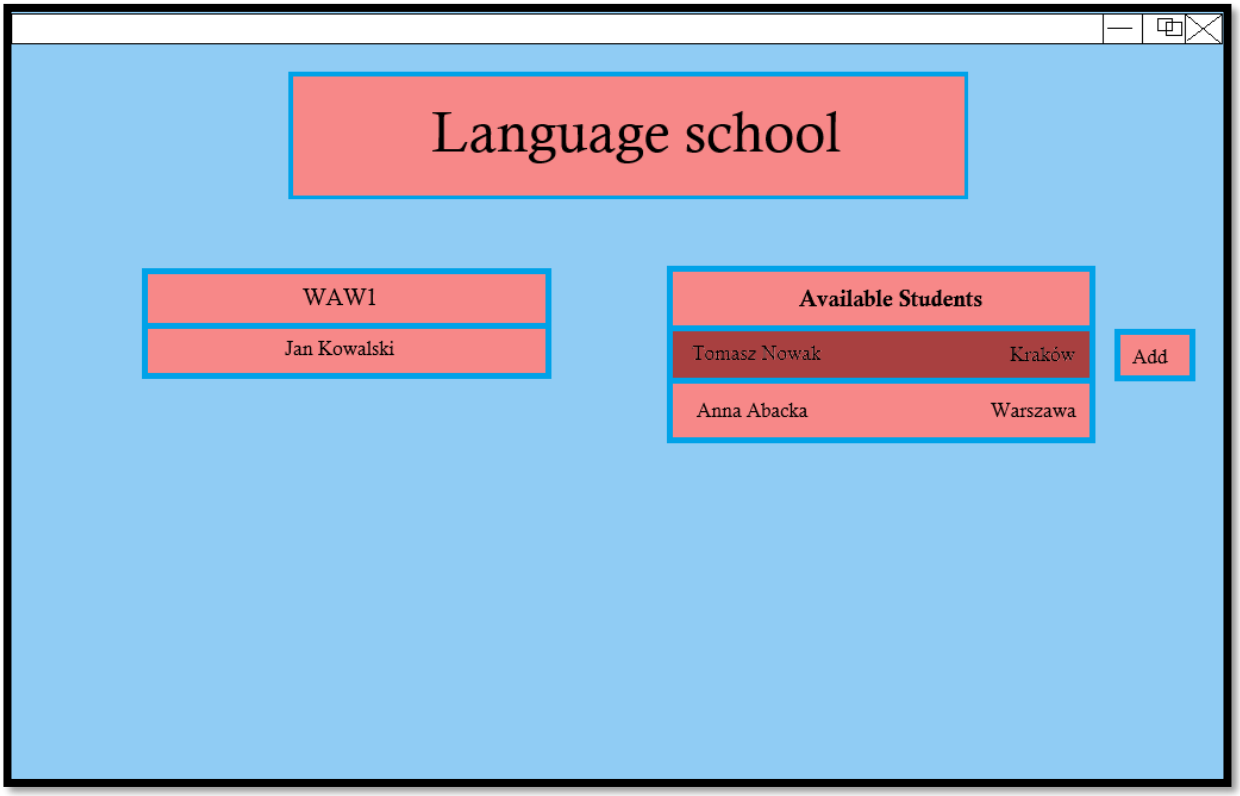
Rysunek 9 GUI Lista grup zaznaczenie



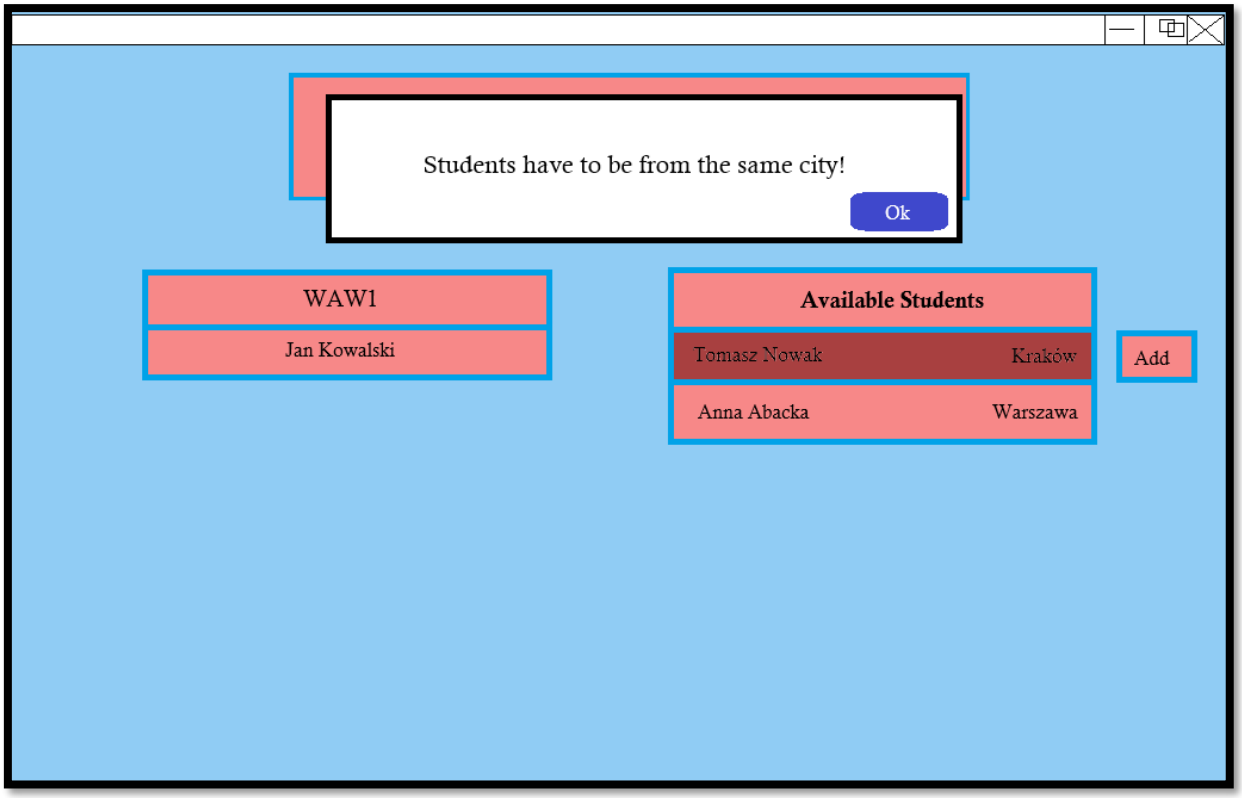
Rysunek 8 GUI Lista grup błąd pełnej grupy



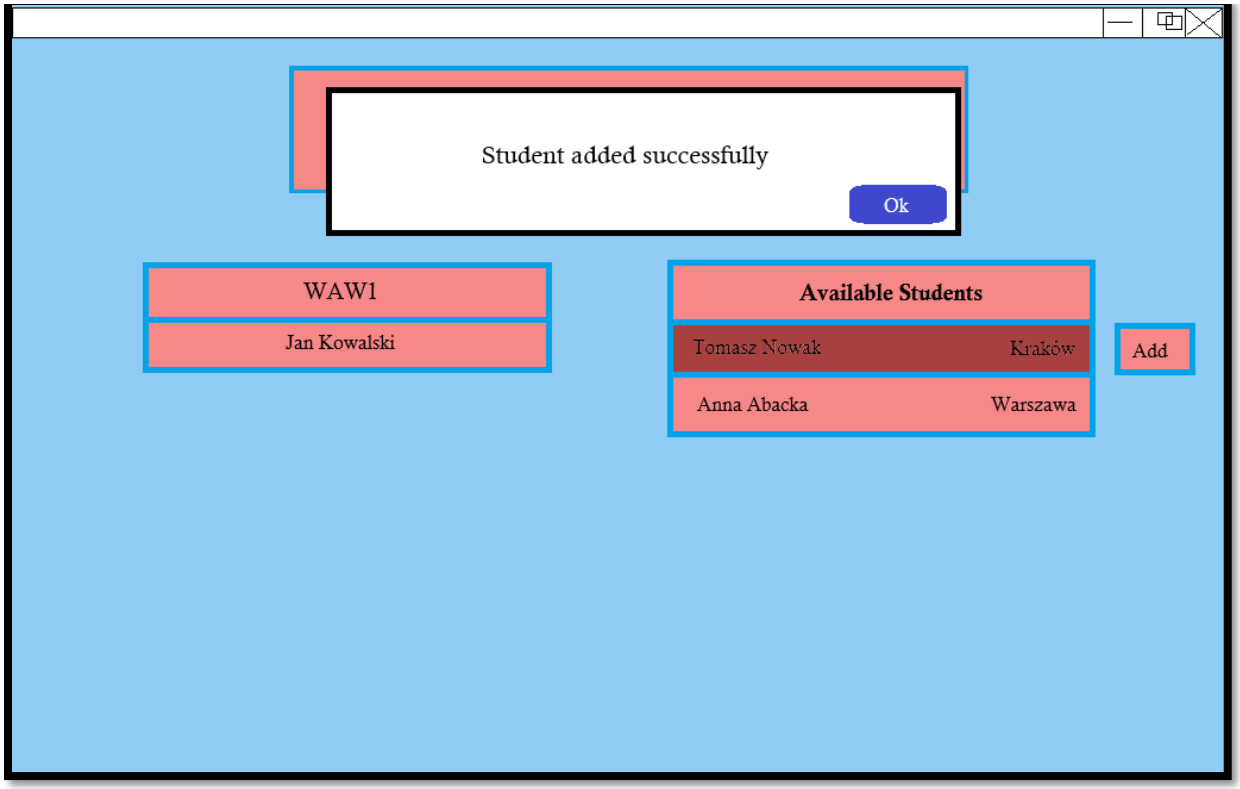
Rysunek 10 GUI Lista Studentów



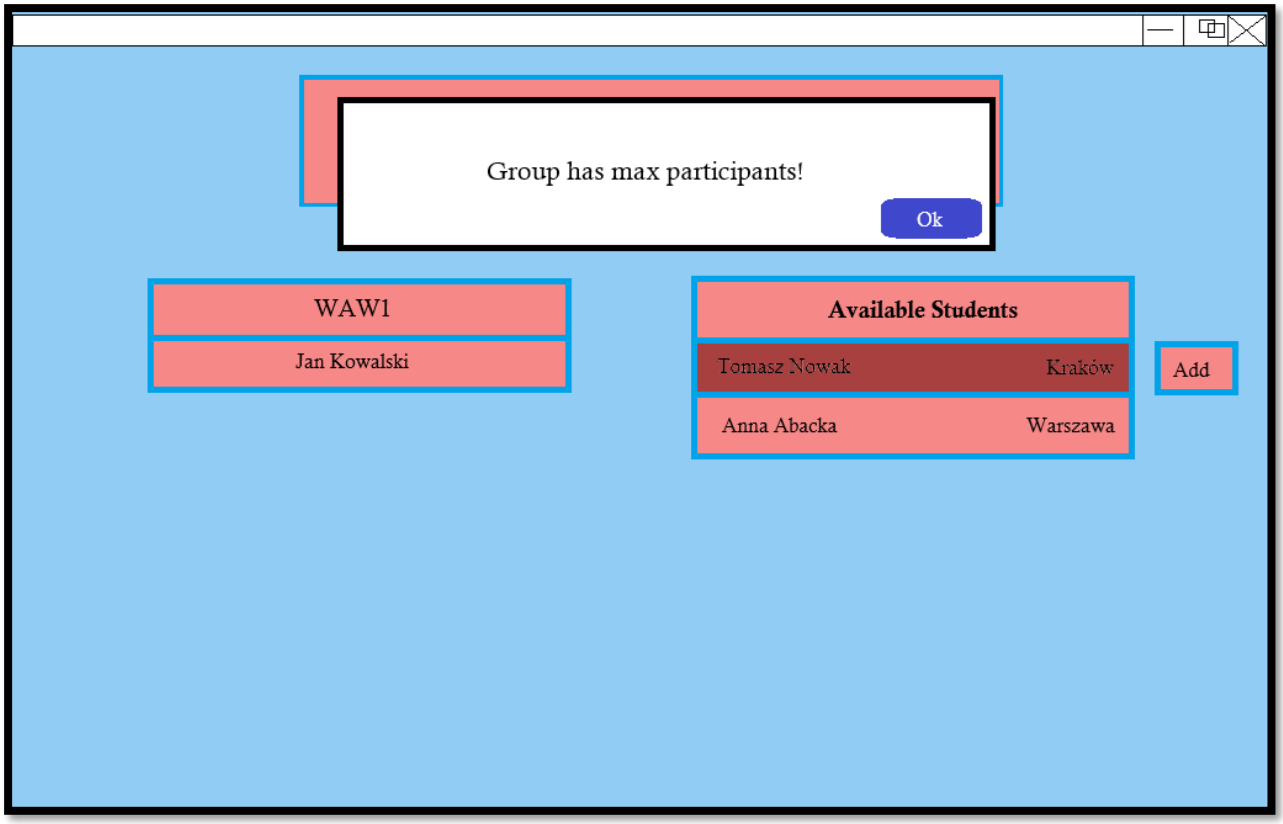
Rysunek 11 GUI Lista studentów zaznaczenie



Rysunek 12 GUI Lista studentów błąd złego miasta



Rysunek 13 GUI Lista studentów pomyślne dodanie



Rysunek 14 GUI Lista studentów błąd pełnej grupy

Omówienie decyzji projektowych i skutków analizy dynamicznej

Użyte technologie

Aplikacja jest zaimplementowana przy użyciu Javy, w szczególności we frameworku Spring Boot. Aplikacja korzysta z bazy danych H2 i narzędzia ORM Hibernate. GUI jest zaimplementowane przy użyciu Thymeleaf. Wybrałem te technologie ponieważ, uczę się Javy od początku studiów i dobrze się w nich czuję.

Omówienie decyzji projektowych

Asocjacja z atrybutem – klasy Group – Tutor oraz Lesson - Room

Z racji, że języki programowania nie wspierają tworzenia bezpośrednio asocjacji z atrybutem, zmieniłem atrybuty na klasy asocjacyjne, które połączyłem asocjacjami z klasami.

Dziedziczenie overlapping Employee

Postanowiłem, że dziedziczenie overlapping będzie zaimplementowane przy użyciu kompozycji, dlatego na diagramie zamiast dziedziczenie powstała kompozycja z klasami Tutor, Manager, Administrator. Dokonałem takiego wyboru, ponieważ kompozycja pozwala na lepszą kontrolę asocjacji poszczególnych podklas.

Atrybuty klasowe

Z racji, że bazy danych nie wspierają atrybutów statycznych, atrybut maxStudents będzie przechowywany w pamięci programu

Ograniczenie sorted w asocjacji Building – Room

Pokoje będą sortowane po numerach

Skutki analizy dynamicznej

Klasa Student

Podczas tworzenia diagramu stanu, który dotyczy klasy Student, zaistniała potrzeba dodania metody zmieniającej stan ucznia – setStatus(StudentStatus)

Podczas tworzenia diagramu aktywności dodałem metody umożliwiające realizację przypadku użycia. showStudents() wyświetla uczniów i korzysta z metod findAllByBelongsToIsNull() która wyszukuje uczniów bez przydzielonych grup oraz findById(groupId) która zwraca listę studentów danej grupy.

Klasa Group

Podczas tworzenia diagramu aktywności, który między innymi dotyczy klasy Group postanowiłem dodać metodę zmieniającą maksymalną liczbę uczniów w grupie changeMaxStudents()

Dodatkowo dodałem metody potrzebne do GUI: showGroups(), która wyświetla listę grup i korzysta z metody findAll() która zwraca wszystkie grupy oraz addStudent(student), która dodaje studenta do grupy.