Projekt Bazy Danych

Autor: *Krystian Ruszczak*
Akademia Górniczo-Hutnicza

Spis treści

---

1. Wstęp

Celem projektu jest stworzenie aplikacji bazy danych, która umożliwia przechowywanie informacji o wybranych obiektach, w tym przypadku będą to osoby. Projekt jest realizowany z wykorzystaniem technik programowania obiektowego w celu zapewnienia przejrzystości kodu i możliwości łatwej rozbudowy. Aplikacja składa się z dwóch głównych modułów: silnika bazy danych oraz interfejsu użytkownika (GUI), stworzonego przy użyciu biblioteki Qt.

2. Analiza problemu

Struktura danych

Podstawowym problemem jest opracowanie struktury bazy danych, która będzie mogła przechowywać informacje. Wymaga się, aby dane były przechowywane w sposób uporządkowany, umożliwiając szybki dostęp do rekordów oraz operacje takie jak dodawanie, wyszukiwanie, edytowanie i usuwanie. Struktury danych muszą być skalowalne i wydajne, aby obsłużyć rosnącą liczbę rekordów. Dodatkowo, należy każdemu z rekordów nadać unikalny w skali całej bazy danych identyfikator, np. numer, umożliwiający rozróżnianie rekordów o podobnych danych.

Interfejs użytkownika

Kolejnym wyzwaniem jest zaprojektowanie prostego i intuicyjnego interfejsu użytkownika (GUI), który pozwoli na prostą i efektywną obsługę bazy danych. Interfejs powinien być przejrzysty i przyjazny dla użytkownika. Szczególną uwagę należy zwrócić na czytelność informacji w tabelach oraz nawigację po interfejsie.

Zapis i odczyt danych

Kluczowym elementem jest zapewnienie trwałego przechowywania danych na dysku. Wybrano format JSON ze względu na jego popularność, powszechne wykorzystanie i szerokie wsparcie. Ważnym wyzwaniem jest zapewnienie poprawności danych przy zapisie i minimalizacja ryzyka ich utraty w przypadku awarii.

Możliwość rozbudowy

Projekt powinien zostać zaprojektowany z myślą o przyszłej rozbudowie. Przykładowo, w przyszłości projekt będzie można rozszerzyć o funkcje importu oraz eksportu danych do innych formatów (np. CSV), czy integrację z zewnętrznymi systemami.

3. Funkcjonalności

Aplikacja zapewnia następującą funkcjonalność:

1. **Dodawanie rekordów** – umożliwia użytkownikowi wprowadzenie nowych rekordów osób do bazy. Dane przechowywane w rekordzie to:
    - Imię,
    - Nazwisko,
    - Numer ID (wyjątkowy dla każdego rekordu),
    - Wiek

   oraz adres zamieszkania, na który składają się:
    - miejscowość,
    - ulica,
    - numer domu,
    - kraj.
2. **Usuwanie rekordów** – możliwość usuwania wybranych przez użytkownika danych z bazy.
3. **Edytowanie rekordów** - możliwość edycji poszczególnych danych dla wybranego rekordu, przykładowo użytkownik może zmienić imię, wiek lub adres osoby w bazie.
4. **Wyszukiwanie** – funkcja przeszukiwania bazy danych według określonych kryteriów. Pozwala użytkownikowi na szybkie znalezienie konkretnego rekordu.
5. **Wyświetlanie wyników** – prezentacja rekordów w tabelach.

	Imię	Nazwisko	Numer ID	Wiek	Miejscowość	Ulica	Numer domu	Kraj
1.	Adam	Kowalski	1	31	Kraków	Floriańska	9	Polska
2.	Marian	Nowak	2	44	Warszawa	Techników	9	Polska

6. **Zapisywanie i odczytywanie z dysku** – możliwość eksportu i importu danych z pliku w formacie JSON. Program będzie automatycznie tworzył nowy plik np. “Baza_danych.json”, w którym będzie zapisywał wszystkie rekordy.

```
{
        "adres": {
                "kraj": "Polska",
                "miejscowosc": "Kraków",
                "numer_domu": 11,
                "ulica": "Floriańska"
        },
        "id": 1,
        "imie": "Adam",
        "nazwisko": "Kowalski",
        "wiek": 31
},
```


Przykładowa struktura zapisu danych w formacie JSON

4. Projekt techniczny

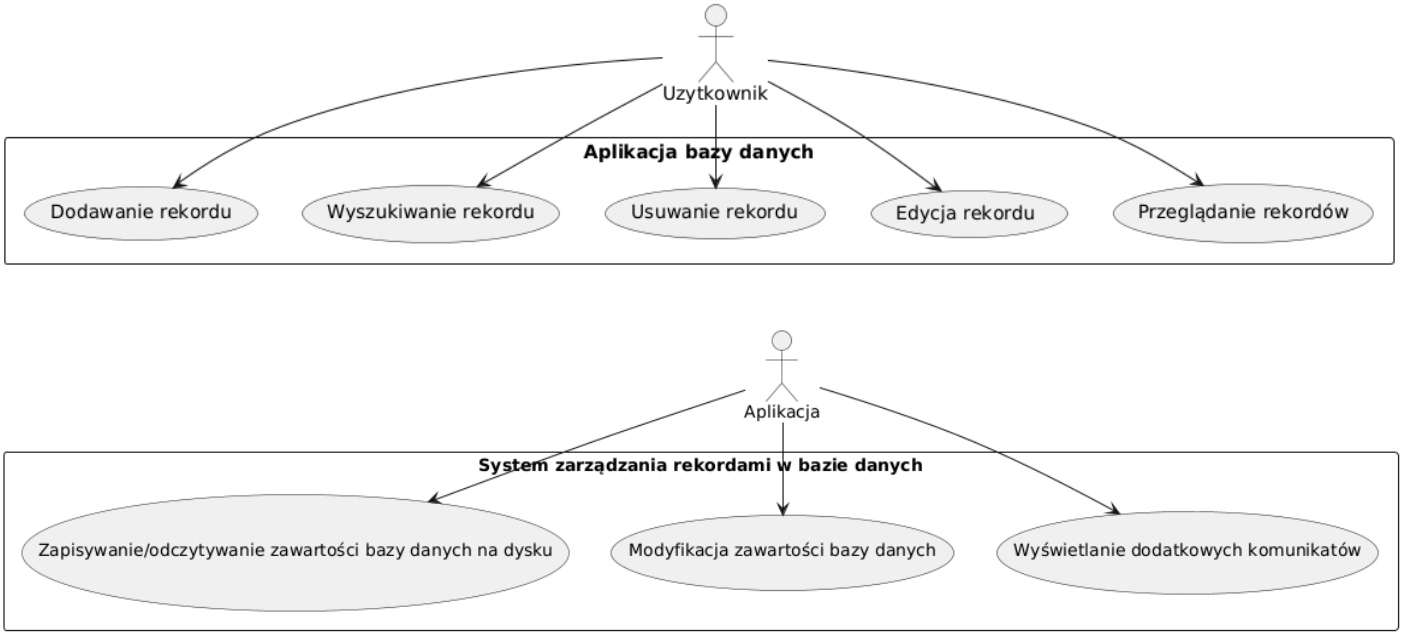
Opis klas:

- Adress – klasa reprezentująca obiekt (adres) w klasie Person, zawierająca dane wchodzące w skład adresu dla pojedynczego rekordu.
- Person – klasa reprezentująca pojedynczy obiekt (osoba) w bazie danych, zawierająca dane dla pojedynczego rekordu osoby.
- DataBase – klasa zarządzająca rekordami, zawierająca metody dodawania, usuwania, edytowania, wyszukiwania, zapisu i odczytu danych na dysku.
- MainWindow – klasa odpowiedzialna za budowanie i zarządzanie interfejsem graficznym.
- Dialog - klasa wchodząca w skład interfejsu graficznego, zarządzająca dodatkowym oknem dialogowym.

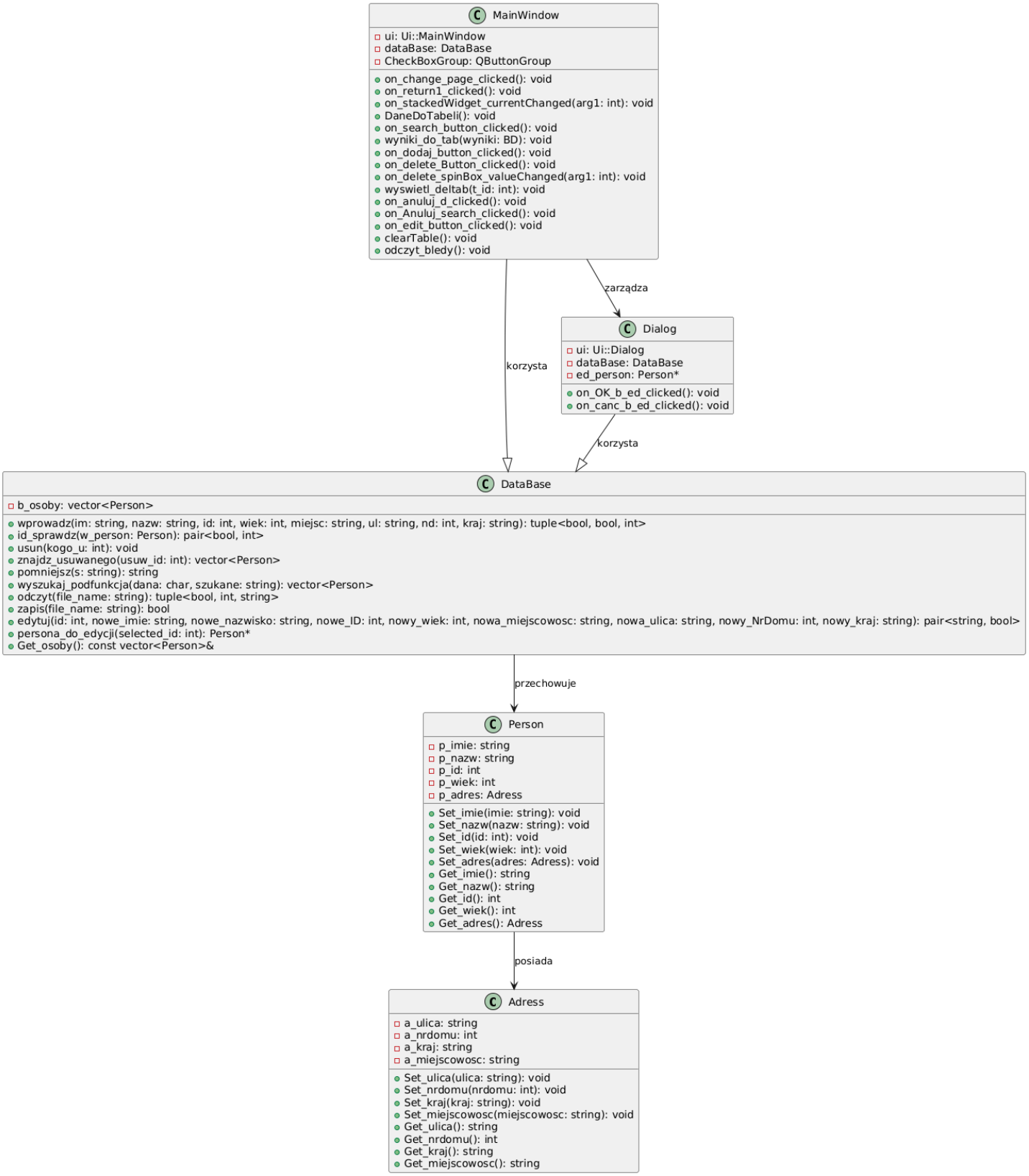
Poniżej zamieszczono diagramy UML typu:

- diagram przypadków użycia,
- diagram klas,
- diagramy aktywności,
- diagram sekwencji.

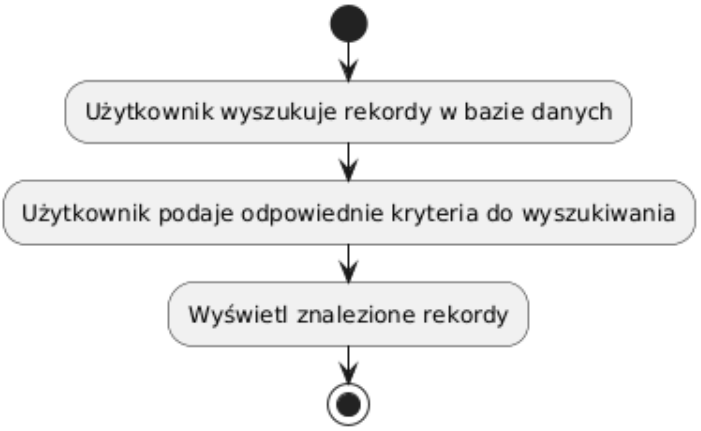
Zamieszczone poniżej diagramy wykonano przy użyciu strony PlantUML.



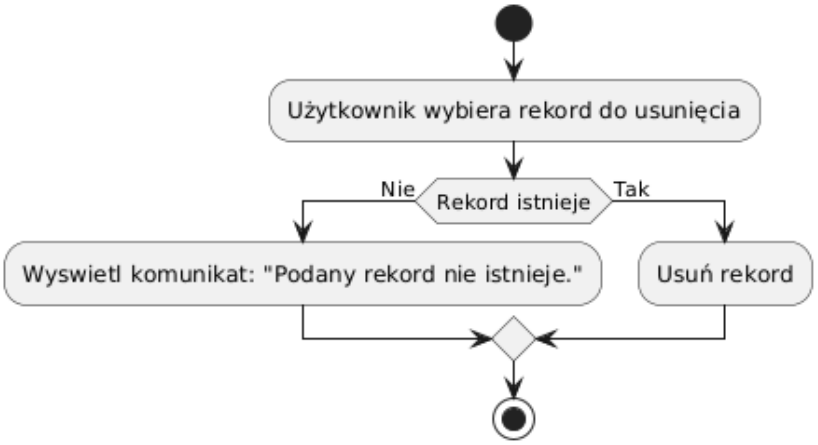
Rysunek 2. Diagramy przypadków użycia



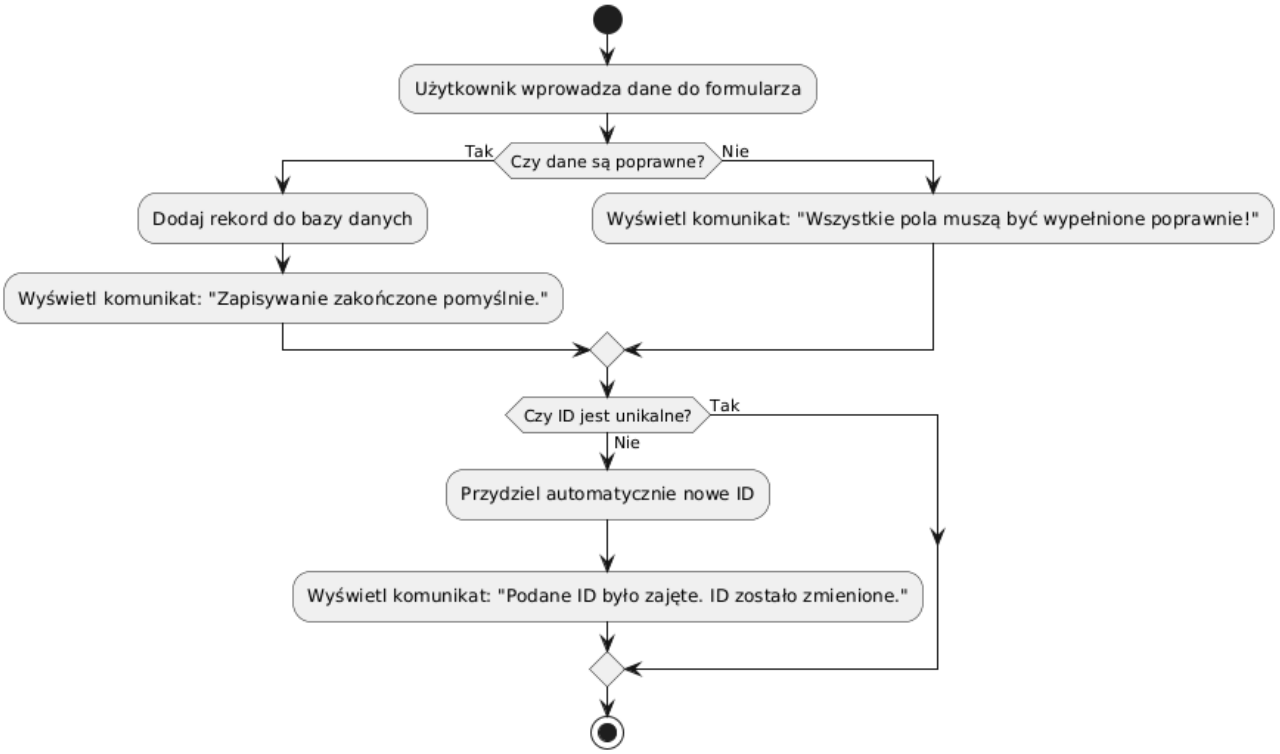
Rysunek 3. Diagram klas



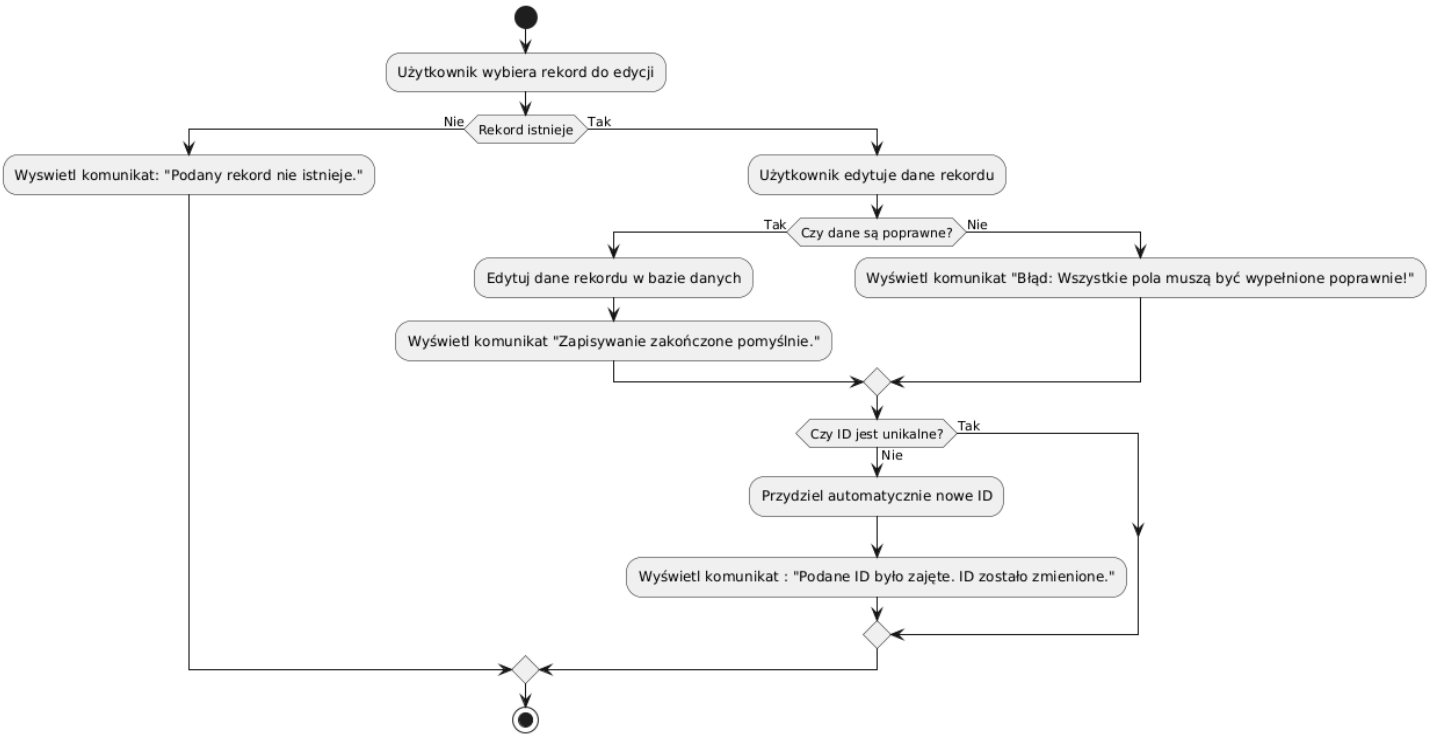
Rysynek 4. Diagram aktywności reprezentujący algorytm wyszukiwania rekordów przez uzytkownika



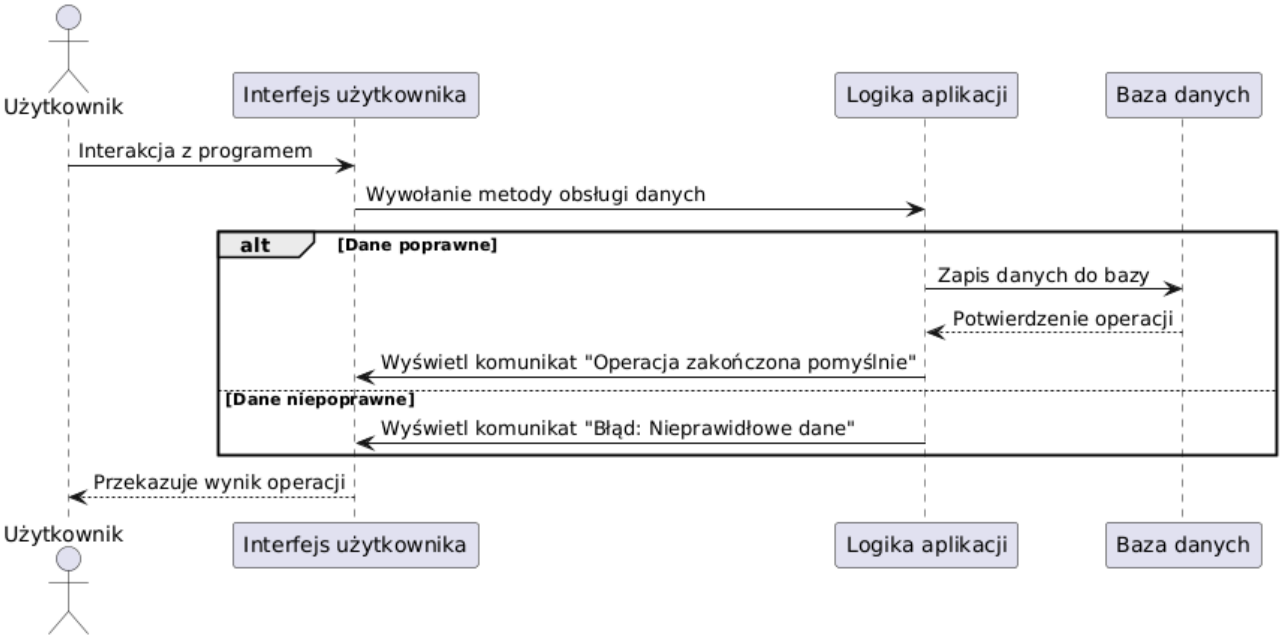
Rysynek 5. Diagram aktywności reprezentujący algorytm usuwania rekordów



Rysynek 6. Diagram aktywności reprezentujący algorytm wprowadzania nowych rekordów



Rysynek 7. Diagram aktywności reprezentujący algorytm edytowania rekordów



Rysynek 8. Diagram sekwencji reprezentujący interakcję użytkownika z systemem zarządzania bazą danych

5. Opis realizacji

Uwagi wstępne

Do wykonania projektu, podczas procesu pisania programu:

- inspirowano się lub zaadaptowano kod z książki "*Introduction to Programming with C++ for Engineers*", autor: prof. dr hab. inż. Bogusław Cyganek,
- wykorzystano bibliotekę nlohmann/json (plik **json.hpp**), autor: Niels Lohmann, udostepnioną pod adresem nlohmann/json: JSON for Modern C++ - GitHub, przy pracy z formatem JSON (na końcu dokumentu zamieszczono licencję),
- inspirowano się lub zaadaptowano kod wygenerowany przez narzędzie ChatGPT 4,
- inspirowano się lub zaadaptowano kod ze strony cppreference.com, zawierającą dokumentację i przykłady użycia dla języka C++.
- inspirowano się lub zaadaptowano kod zaprezentowany na stronie Qt for Beginners.

Wyżej wymienione źródła zostały wykorzystane przy realizacji projektu, w szczególności przy tworzeniu ogólnego zarysu kodu oraz w plikach stanowiących integralną część projektu. Kod prezentowany w każdym z wyżej wymienionych źródeł został przeanalizowany, dostosowany oraz/lub zaadaptowany poprzez np. zmianę nazw zmiennych, funkcji i dostosowanie logiki programu do specyfiki projektu.

Główne etapy implementacji:

- Utworzono klasy Adress i Person z polami przechowującymi odpowiednie dane, np. imię, nazwisko, ulica itp.
- Zaimplementowano klasę DataBase z metodami zarządzającymi rekordami klasy Person.
- Stworzeno interfejs graficzny z użyciem Qt Creator oraz powiązano go z metodami DataBase. Zaimplementowano główne okno dialogowe MainWindow z funkcjami dodawania, usuwania, edytowania i wyszukiwania rekordów. Utworzono okno dialogowe i odpowiadającą mu klasę Dialog, umożliwiającą zmianę danych wybranej osoby.
- Na potrzeby projektu powstały pliki:
    - Adress.h/Adress.cpp
    - Person.h/Person.cpp
    - DataBase.h/DataBase.cpp
    - mainwindow.h/mainwindow.cpp/mainwindow.ui
    - dialog.h/dialog.cpp/dialog.ui
    - main.cpp

Technologie:

- Język: C++ 20.
- Biblioteka GUI: Qt.
- Format zapisu danych: JSON (przy użyciu biblioteki nlohmann/json).
- Środowisko IDE: Visual Studio 2022, Qt Creator 15.0.0 Community.
- Narzędzie CMake.
- GTest (Google Test) - narzędzie do przeprowadzania testów funkcji programu.

6. Opis wykonanych testów

W projekcie przetestowano kluczowe funkcjonalności aplikacji za pomocą GoogleTest. Testy miały na celu zweryfikowanie poprawności działania metod klasy DataBase. Przeprowadzono następujące rodzaje testów:

- dodawanie rekordów – sprawdzono, czy można poprawnie dodać osobę do bazy danych, a dane są przechowywane w oczekiwanej formie.
- usuwanie rekordów – zweryfikowano, czy po usunięciu rekordu nie jest on dostępny w bazie danych.
- edycja rekordów – przetestowano możliwość modyfikacji danych istniejącego rekordu.
- wyszukiwanie rekordów – przetestowano wyszukiwanie osób na podstawie różnych kryteriów, takich jak imię, nazwisko, numer ID lub wiek. W testach sprawdzono, czy wielkość liter szukanej danej wpływa na wyniki.
- operacje na plikach – sprawdzono zapisywanie i odczytywanie danych z pliku, weryfikując zgodność wprowadzonych danych z zapisanym wynikiem.

```
[==========] Running 7 tests from 1 test case.
[----------] Global test environment set-up.
[----------] 7 tests from DataBaseTest
[ RUN      ] DataBaseTest.Dodaj_rekord_i_wyszukaj_imie
[       OK ] DataBaseTest.Dodaj_rekord_i_wyszukaj_imie (1 ms)
[ RUN      ] DataBaseTest.Wyszukaj_ID
[       OK ] DataBaseTest.Wyszukaj_ID (0 ms)
[ RUN      ] DataBaseTest.Wyszukaj_nazwisko
[       OK ] DataBaseTest.Wyszukaj_nazwisko (1 ms)
[ RUN      ] DataBaseTest.Wyszukaj_wiek
[       OK ] DataBaseTest.Wyszukaj_wiek (1 ms)
[ RUN      ] DataBaseTest.Usun_osobe
[       OK ] DataBaseTest.Usun_osobe (1 ms)
[ RUN      ] DataBaseTest.Edycja_osoby
[       OK ] DataBaseTest.Edycja_osoby (1 ms)
[ RUN      ] DataBaseTest.Zapis_i_odczyt
[       OK ] DataBaseTest.Zapis_i_odczyt (2 ms)
[----------] 7 tests from DataBaseTest (10 ms total)

[----------] Global test environment tear-down
[==========] 7 tests from 1 test case ran. (10 ms total)
[  PASSED  ] 7 tests.
```

Rysunek 7. Zrzut ekranu przedstawiający pomyślne wykonanie siedmiu testów.

Uwagi

Wszystkie testy przeprowadzono przy użyciu przykładowych danych. Wprowadzane dane nie używały polskich liter. Powodem był napotykany błąd, którego nie udało się rozwiązać: *unknown file: error: C++ exception with description "[json.exception.type_error.316] incomplete UTF-8 string; last byte: 0xF1" thrown in the test body.*

Jednak sama aplikacja bez problemów obsługuje polskie znaki.

7. Podręcznik użytkownika

W przypadku pierwszego uruchomienia programu lub braku zawartości w pliku przechowującym rekordy wyświetli się komunikat informacyjny.

Po uruchomieniu programu wyświetlane jest okno głównego wyboru, zawierające trzy opcje:

- **Wyświetl aktualną bazę danych**
- **Dodaj nowy rekord**
- **Edytuj lub usuń rekord**

Po wybraniu jednej z opcji i zatwierdzeniu przyciskiem „Dalej” następuje przekierowanie do odpowiedniego okna.

Wyświetl aktualną bazę danych

Po przejściu do tego okna wyświetlana jest tabela z zawartością bazy danych. W razie niepowodzenia wczytania danych, pojawia się komunikat informujący o wystąpieniu problemu wraz z jego opisem.

Dostępne jest tutaj wyszukiwanie rekordów po podaniu odpowiednich kryteriów i zatwierdzeniu przyciskiem “OK”. Wyszukiwanie jest realizowane bez uwzględniania wielkości liter.

Po kliknięciu “Anuluj”, resetowane są kryteria wyszukiwania i widok bazy danych powraca do domyślnego.

Do menu głównego przechodzimy klikając “Powrót do menu”.

Dodaj nowy rekord

W tym oknie wyświetlane są pola formularza przeznaczone do wprowadzenia danych nowego rekordu. W przypadku wprowadzenia istniejącego już numeru ID (który powinien być uniklany w skali całej bazy danych), program automatycznie przypisze najmniejszy dostępny numer.

Po zatwierdzeniu przyciskiem “Dodaj”, nowy rekord zapisywany jest w bazie danych. Kliknięcie “Anuluj” skutkuje wyczyszczeniem zawartości formularza.

Do menu głównego przechodzimy klikając “Powrót do menu”.

Edytuj lub usuń rekord

W tym oknie należy podać numer ID rekordu, którego dane mają zostać usunięte

lub edytowane. Po podaniu numeru, wyswietlą się dane osoby o wskazanym ID. W przypadku, gdy rekord o wskazanym numerze ID nie istnieje, wyświetlany jest komunikat informujący o jego braku.

Po kliknięciu “Usuń” dane są usuwane z bazy.

Po kliknięciu “Edytuj” wyskakuje dodatkowe okno z formularzem. Pola formularza są domyślnie wypełnione aktualnymi danymi rekordu, co pozwala na ich szybkie i wygodne modyfikowanie. Po wprowadzeniu nowych danych i zatwierdzeniu przyciskiem “OK”, następuje nadpisanie danych. Po kliknięciu “Anuluj” operacja edycji jest przerywana. W przypadku zmiany numeru ID na numer już zajęty, program automatycznie zmieni numer na najmniejszy dostępny - pojawi się informujące o tym komunikat.

Do menu głównego przechodzimy klikając “Powrót do menu”.

Zamykanie programu

W celu zamknięcia programu należy kliknąć “Zakończ program” lub krzyżyk w prawym górnym rogu.

8. Przykładowa instalacja i uruchomienie programów

Proces przeprowadzony na komputerze z systemem operacyjnym **Windows 10**.

Wymagane są zainstalowane biblioteki Qt oraz dodanie do zmiennej środowiskowej ‘Path’ ścieżek (…)\Qt\(wersja)\msvc()\bin, (...)\CMake\bin i (...)\Visual Studio IDE\MSBuild\Current\Bin.

- Wszystkie niezbędne pliki znajdują się w folderze */Database*.
- Po pobraniu folderu utworzono folder np. ‘*build*’.
- Przy pomocy *Windows PowerShell* z poziomu utworzonego folderu wywołano komendę: ‘*cmake (ścieżka do folderu Database)/DataBase*’.
- Następnie wywołano komendę ‘*msbuild DataBase.sln /p:Configuration=Release*’.
- W folderze build powinien zostać utworzony folder *Release*. Przechodzimy do folderu i uruchamiamy plik *DataBaseApp.exe*. Program bazy danych powinien się uruchomić.

Alternatywnie:

Wymagane są zainstalowane biblioteki Qt.

- Wszystkie niezbędne pliki znajdują się w folderze */Database*.
- Po pobraniu folderu utworzono folder np. ‘*build*’.
- Przy pomocy *Windows PowerShell* z poziomu utworzonego folderu wywołano komendę: ‘*cmake -DCMAKE_PREFIX_PATH="(pełna scieżka do folderu z Qt)\Qt\6.8.1\msvc2022_64" (ścieżka do folderu Database)\Database*’.
- Następnie wywołano komendę: ‘*cmake --build . --config Release*’ oraz po przejściu do folderu *\Release*: ‘*(ścieżka do folderu z Qt)\Qt\6.8.1\msvc2022_64\bin\windeployqt.exe .\DatabaseApp.exe*’.

Proces przeprowadzony na maszynie wirtualnej z systemem operacyjnym **Ubuntu**.

Wymagane są zainstalowane biblioteki Qt.

- Po pobraniu folderu należy utworzyć folder np. ‘*build*’.
- Przechodzimy do utworzonego folderu i z jego poziomu wywołujemy komendę: ‘*cmake (ścieżka do folderu Database)/Database*’.
- Następnie wywołujemy komendę ‘*make*’.
- W folderze build powinien zostać utworzony folder */gui*. Przechodzimy do folderu *gui* i uruchamiamy plik *Database* komendą ‘*./Database*’. Program bazy danych powinien się uruchomić.

9. Lincencje

Biblioteka **nlohmann/json** jest udostępniona na licencji MIT. Poniżej zamieszczony jest tekst licencji MIT: