Technical Report
AGH University of Science and Technology

Baza Danych z Interfejsem Graficznym

Autor : Bartłomiej Woś

Prowadzący: prof. dr hab. inż Bogusław Cyganek

16.01.2023 r. - ver. 01

Akademia Górniczo-Hutnicza

Spis treści

Lista oznaczeń

- OOP - Object-Oriented-Programming
- CMAKE - Cross-platform Make
- GUI - Graphical User Interface
- JSON - Java Script Object Notation
- MinGW - Minimalist GNU for Windows

1 Wstęp

Raport dotyczy opracowania aplikacji, która ma służyć jako baza danych. Celem tej aplikacji jest przechowywanie oraz edycja określonego zbioru informacji tekstowych i liczbowych, które zostały zapisane w odpowiednim formacie. Docelowo program ma służyć jako uporządkowany zbiór rekordów (informacji) o studentach danej uczelni. Z uwagi na fakt wykorzystania OOP, wyżej wymieniona aplikacja będzie mogła być z łatwością przeprojektowana, np. na bazę danych dla pracowników. W celu ułatwienia interakcji systemu z użytkownikiem program posiada prosty w obsłudze interfejs graficzny.

1.1 Wymagania systemowe

Podstawowe założenia projektu:

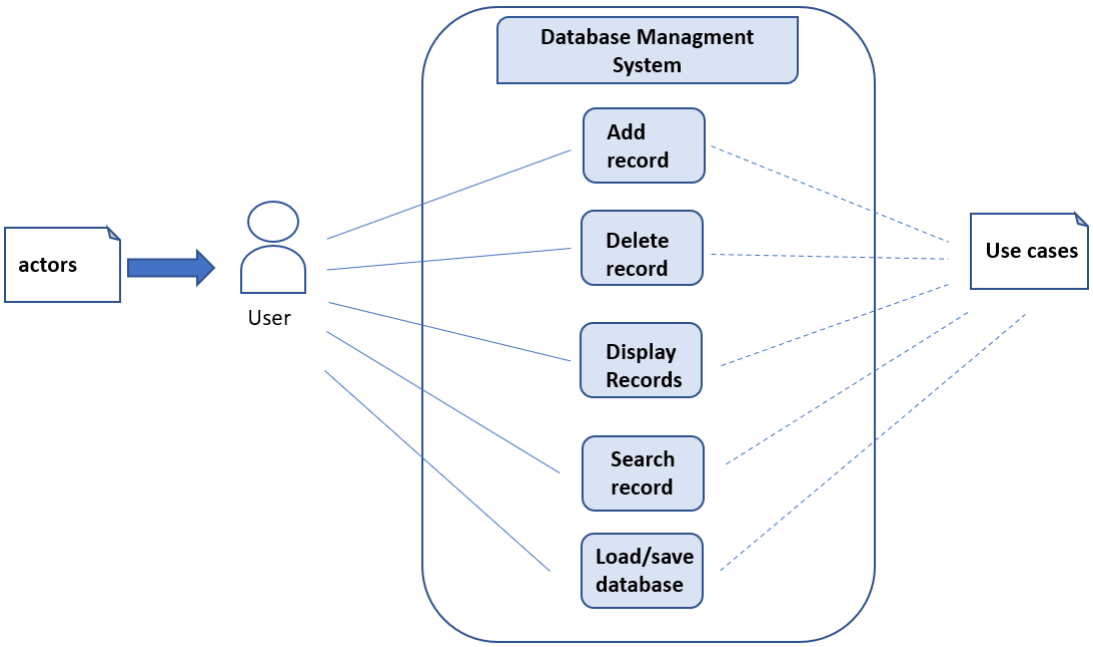
1. Możliwość pracy aplikacji na komputerach z systemem operacyjnym Microsoft Windows.
2. Wykorzystanie wieloplatformowego narzędzia do zarządzania automatycznym procesem komplilacji CMAKE, cechującego się niezależnością kompilatora od platformy sprzętowej.
3. Aplikacja ma być zaimplementowana za pomocą jezyka C++ w standardzie 20 w technologi OOP, z użyciem środowiska programistycznego Qt pozwalającego na tworzenie GUI.
4. Struktura programu powinna pozwalać na szybkie lokalizowanie błędów oraz jeśli zajdzie taka potrzeba łatwą modernizacje w celu aktualizacji aplikacji dla klienta.
5. Baza danych ma przechowywać sprecyzowane informacje o studentach. Są to:
   - numer ID
   - imię
   - nazwisko
   - wiek
   - wydział na którym studiuje dana osoba
   - Kierunek studiów
   - aktualny semestr
   - adres mailowy
   - typ studiów (stacjonarny lub zaoczny)
6. Program powinien oferować możliwość odczytu danych zapisanych w ściśle określonym formacie np. json lub txt.
7. Oprogramowanie powinno być zdatne do dodawania rekordów zawierających uporządkowane informacje.
8. Możliwość usunięcia wybranej pozycji zarejestrowanej w bazie danych, na podstawie wybranego unikalnego parametru.
9. Umożliwienie użytkownikowi wyświetlania wszystkich zgromadzonych rekordów, oraz wyszukania konkretnej pozycji zależnej od unikalnego parametru.
10. Automatyczne generowanie wybranych składowych rekordu (w tym przypadku adres e-mail).
11. Finalnie aplikacja powinna pozwalać na zapis zgromadzonych informacji w zoorganizowany sposób do pliku zgodnego z rozszerzeniem opisanym w pkt. 4.
12. Wszystkie powyższe funkcjonalności powinny być dostępne w łatwym w obsłudze interfejsie graficznym

2 Funkcjonalność

Wszystkie funkcjonalności zaimplementowano zgodnie z wymaganiami systemowymi.

- Aplikacja daje możliwość zarządzania ściśle określonymi informacjami dotyczących studentów sprecyzowanych w sekcji 1.1 pkt.5.
- Istnieje możliwość Dodania oraz usuwania Rekordów w długich interwałach czasowych ,ponieważ program obsługuje odczyt oraz zapis do pliku. Zdecydowano się na rozszerzenie .json ze względu na uporządkowaną strukturę danych znajdującą się w pliku. Więcej informacji na temat tego formatu znajduję się w sekcji 4.
- Oprogramowanie umożliwia również wyszukiwanie konkretnych pozycji na podstawie podanego numeru ID, wyświetlanie całej bazy danych wraz z liczbą aktualnie znajdujących się w niej studentów oraz automatyczną generacje adresów e-mail.
- Wszystkie funkcjonalności są dostępne w przejrzystym i łatwym w obsłudze GUI (szczegółowe informacje znajdują się w sekcji 7). Detale techniczne co do sposobu realizacji poszczególnych funkcjonalności znajdują się w sekcji 3 oraz 4. Aplikacje wzbogacono również o różne komunikaty tekstowe potwierdzające zmiany, odczyty oraz zapisy do plików.

  Poniżej przedstawiono diagram funkcjonalności jakie oferuje aplikacja.



Rys 1. - use case diagram.

3 Analiza problemu

Bazę danych można zdefiniować jako zbiór danych zgromadzonych według określonej systematyki i metodyki. Przebieg takiego postępowania przedstawiono za pomocą tabeli 1 w celu łatwiejszego zrozumienia tematu.

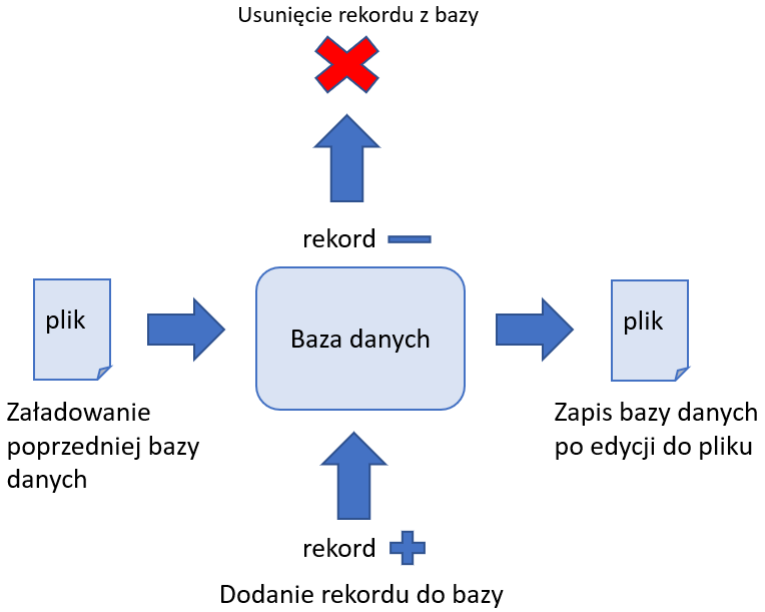
Nr. of student	Student 1	Student 2
ID	690	790
Name	Jan	Marta
Surname	Kowalski	Nowak
Age	22	19
Faculty	WIEiT	EAIiIB
Field	EiT	AiR
Semester	5	2
E-mail	jkowalski690@student.agh.edu.pl	mnowak790@student.agh.edu.pl
Mode of Study	Part-Time	Full-Time

Tabela 1. - wizualizacja pierwszych 2 rekordów zarejestrowanych w bazie danych.

Na podstawie tabeli 1 widzimy, że jednym z kluczowych elementów jest przechowywanie informacji w czytelny i uporządkowany sposób. Zatem pierwszym problemem wynikającym z analizy będzie odpowiednie składowanie informacji, które należą do danej osoby. Ponadto aby baza danych spełnianiała swoje podstawowe funkcję musi istnieć możliwość jej edycji tzn. dodawania oraz usuwania rekordów. Program musi być także niezależny od interwałów czasowych jego uruchamiania, wprowadzone dane muszą być gdzieś zapisywane oraz odczytywane podczas późniejszego użycia aplikacji. Z początkowej fazy projektowania wynikają następujące problemy:

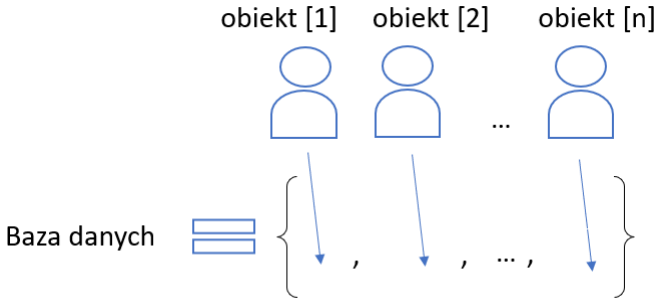
- przechowywanie informacji dla poszczególnych osób
- Odczyt z pliku
- dodanie rekordu
- usunięcię rekordu
- zapis do pliku

Poniżej przedstawiono prosty schemat blokowy obejmujący wyżej wymienione funkcjonalności.



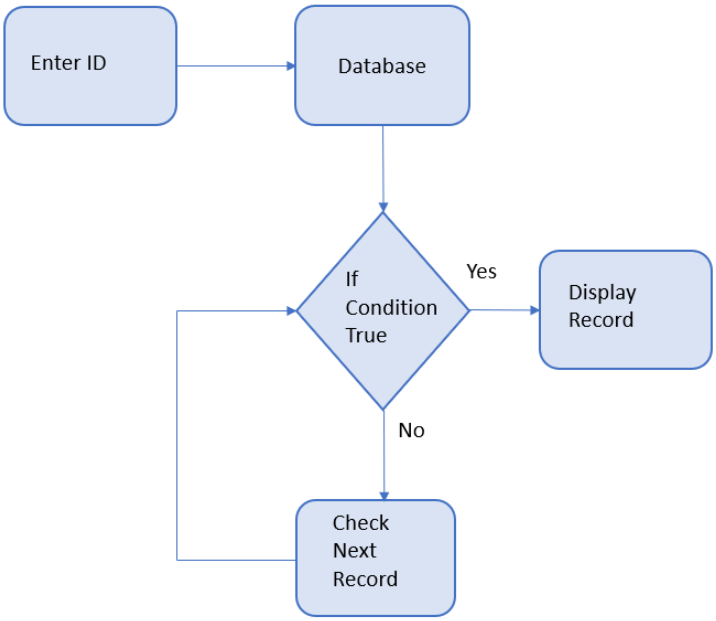
Rys. 2. - Uproszczony schemat bazy danych obejmujący podstawowe operacje.

Uporządkowane składowanie rekordów może być zrealizowane przy wykorzystaniu OOP. Możemy wtedy przedchowywać informację dla każdej z osób jako obiekt. Całą bazę bazę danych można zrealizować jako wektor takich obiektów co zostało pokazane na rys. 2.



Rys. 3. - Baza danych jako wektor obiektów

Kolejnym problemem poddanym analizie jest wyszukiwanie oraz wyświetlanie rekordu. Każda wpisana osoba posiada unikalny numer ID, w związku z tym szukanie powinno odbywać sie właśnie na podstawie tego parametru. Poniżej przedstawiono jeden ze sposobów na wykonanie takiej funkcjonalności.



Rys. 4. - Przykładowy schemat blokowy funkcji realizującej wyszukiwanie rekordu.

Analogicznie można zrealizować usuwanie rekordu, zamiast wyświetlania.

Automatyczne generowanie adresu e-mail może zostać zrealizowane za pomocą trywialnego algorytmu. Mimo wszystko został on umieszczony poniżej w celu możliwości weryfikacji czy działa poprawnie.

$$e - mail = name_1 + surname + ID + "@student.agh.edu.pl" \tag{1}$$

gdzie:
$name_1$ - pierwsza litera imienia
$surname$ - nazwisko (skonwertowane do małych liter)
ID - unikalny numer studenta

Z uwagi na fakt, że generowany adres mailowy zawiera numer ID, nie wystąpi możliwość żeby 2 różne osoby posiadały ten sam adres e-mail.

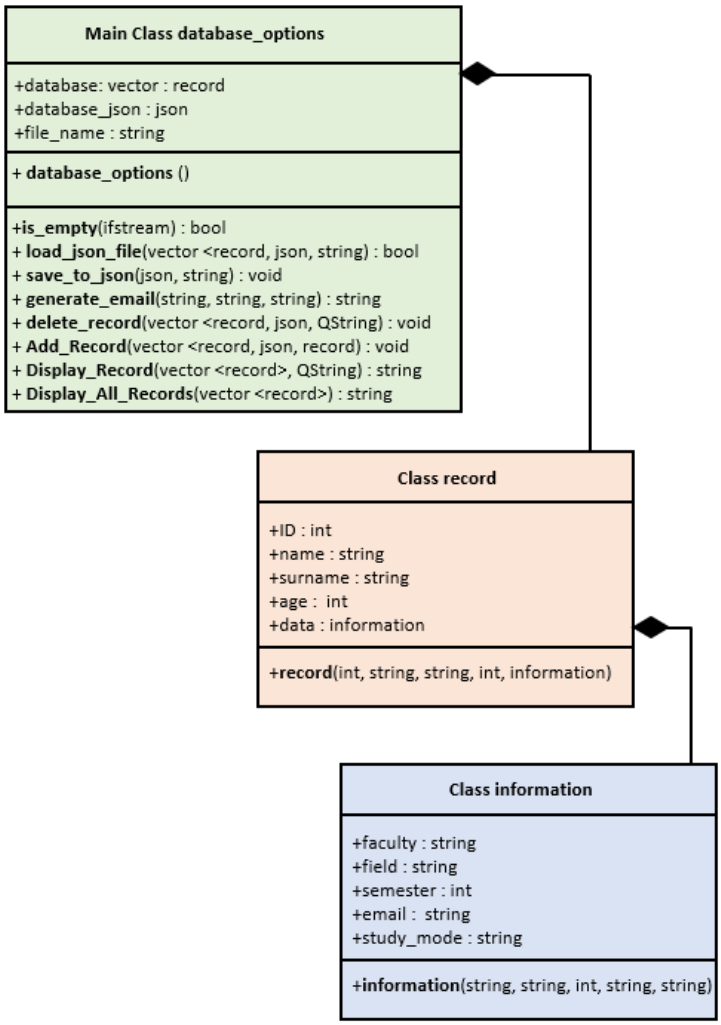
Zapis i odczyt do pliku można zrealizować na podstawie dołączonej klasy która przechowuje wszystkie metody i atrybuty operujące na plikach (np. ofstream)
.

4 Projekt techniczny

4.1 Parę uwag wstępnych

Jednym z wymagań systemowych było stworzenie aplikacji za pomocą technik OOP. Z tego powodu wszystkie informacje należące do bazy danych są przechowywane właśnie jako obiekty na które składa się system klas. Cała baza danych została zamodelowana jako wektor takich obiektów (sekcja 3 rys. 3). Wykorzystano również sposoby na realizowanie poszczególnych funkcjonalności zawarte w analizie problemu (sekcja 3). Kasowanie i wyszukiwanie rekordów odbywa się na podstawie schematu dołączonego jako rys.4, a adres e-mail jest generowany zgodnie z równaniem (1). Wszystkie funkcje służące do zarządzania bazą danych zostały zaimplementowane jako metody klasy głównej. Pozwala to na przejrzystość i czytelność kodu. Struktura klasy głównej została ukazana w sekcji poniżej.

4.2 Przykład - main class hierarchy



Rys. 5. - Hierarchia klas.

Na rys. 5 został przedstawiony układ klas występujący w aplikacji,
gdzie:
+ - plus oznacza typ public
◆— - strzałka zakończona rombem, agregacja (systuacja gdy jedna klasa zawiera obiekty innej))

Warto dodać że każda klasa posiada osobne pliki tj. plik nagłówkowy oraz .cpp, zwiększa to przejrzystość kodu. Poniżej zamieszczono ich krótki opis:

Main Class Database - Klasa główna służąca do zarządzania całą bazą danych ,zawiera metody za pomocą których wykonywana jest wiekszość funkcjonalności całej aplikacji oraz domyślny pusty konstruktor. Posiada 3 atrybuty są to:

- database - wektor obiektów typu Record zawierający wszystkie informacje na temat studenttów i służący za model bazy danych.
- database_json - obiekt typu json zawierający dane wchodzące w skład bazy, istnieje on wyłącznie po to aby zapisać go do pliku ze względu na wybrany format.
- file_name - zmienna typu string przechowująca ścieżke do bazy danych znajdującej się na komputerze.

Jak zostało wspomniane wyżej klasa zawiera metody w których zdefiniowana jest funkcjonalność aplikacji. Z rys. 5 jasno wynika ich cel i przeznaczenie w związku z tym zdecydowano się opisać tylko tą która nie została wspomniana podczas całego dokumentu. Mianowicie:

- is_empty(ifstream) - metoda przyjmuje deskryptor pliku typu ifstream i sprawdza czy plik nie jest pusty, jest ona używana w funkcji załadowania bazy danych i jeśli plik jest pusty na jej podstawie jest zwracany komunikat, a sam odczyt nie jest wykonywany.

Class Record - Klasa zawierająca wszystkie informacje które należą do każdej z osób znajdującej się w bazie danych. Posiada 5 atrybutów:

- ID - zmienna typu int zawierająca unikalny numer ID
- name - zmienna typu string zawierająca imię
- surname - zmienna typu string zawierająca nazwisko
- surname - zmienna typu int zawierająca wiek
- data - obiekt typu Information przechowujący informacje takie jak kierunek studiów, email itd.

Klasa posiada konstruktor inicjalizujący.

Class Information - Klasa przechowuje dane danej osoby związane z uczelnią. Posiada 5 atrybutów oraz konstruktor inicjalizujący domyślnymi wartościami. W skład pól klasy wchodzą:

- faculty - zmienna typu string zawierająca nazwe wydziału na którym dana osoba studiuje
- field - zmienna typu string określająca wybrany kierunek studiów
- semester - zmienna typu int określająca aktualny semestr na którym przebywa student
- email - zmienna typu string zawierająca adres mailowy
- study_mode - zmienna typu string informująca o trybie studiów

Oprócz klas opisanych wyżej środowisko Qt Creator wygenerowało również klase za pomocą której obsługiwane jest okno aplikacji - **MainWindow**. Możemy tworzyć jej metody za pomocą dodawania kolejnych przycisków które mają spełniać określone funkcjonalności. Działanie systemu opiera się na korzystaniu z metod zdefiniowanych w klasie **Database** właśnie w przyciskach odpowiądających za daną funkcję. Dzięki temu uzyskujemy łatwą obsługę błędu, czytelną i spójną strukturę kodu oraz możliwość dalszej rozbudowy.

5 Opis Realizacji

Wykorzystane narzędzia - Aplikacja docelowo została napisana na komputery z systemem Windows.Użyte do jej zrealizowania biblioteki umieszczono poniżej. Wykorzystano oferany przez środowisko Qt Creator kompilator **MinGW**, a jako narzędzie do kontroli kompilacji wybrano **CMAKE**.

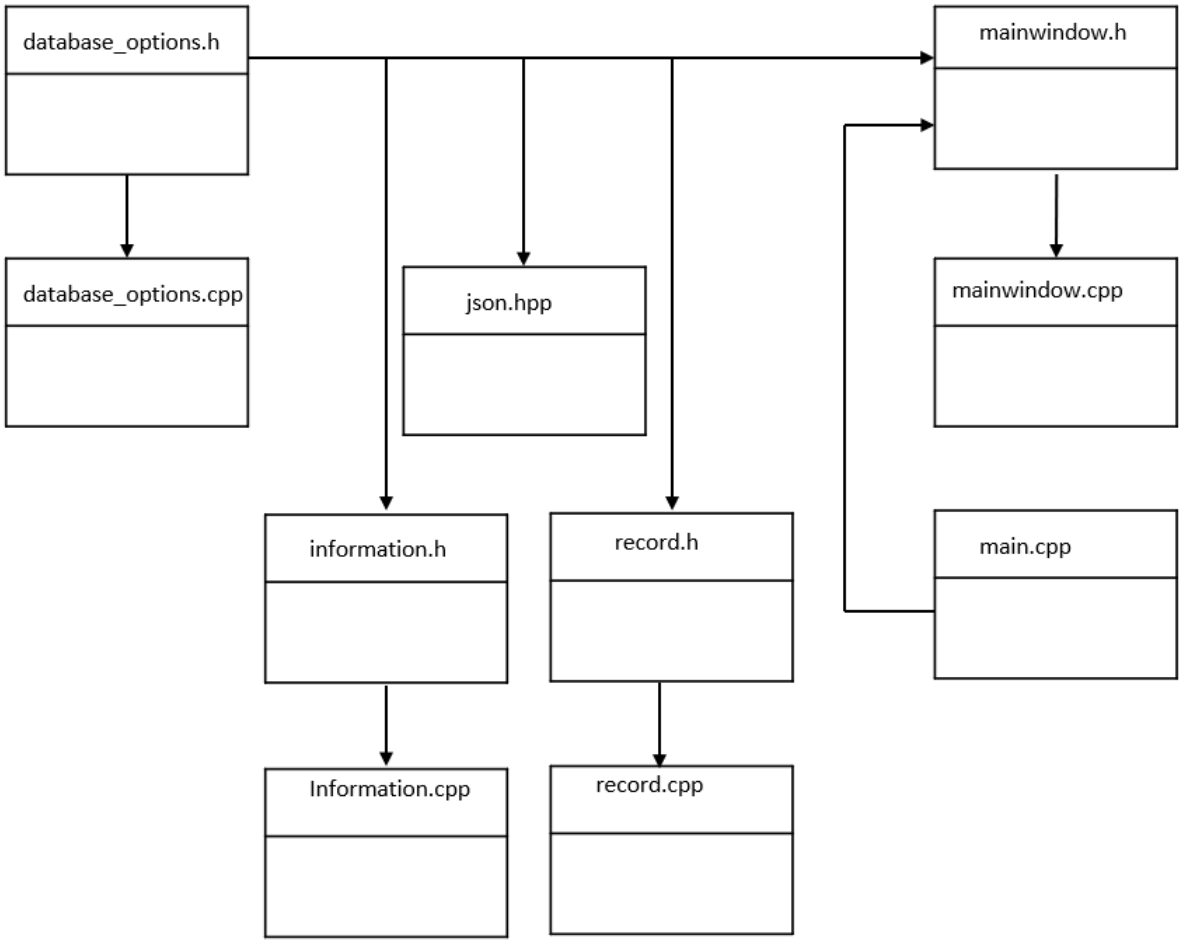
Użyte biblioteki - W celu realizacji projektu niezbędne jest dołączenie bibliotek. Standardowe biblioteki jakie zostały użyte to:

- string - operacje na łańcuchach znaków
- QString - biblioteka pozwalająca operować również na łańcucach znaków udostępniona przez Qt
- vector - kontenter zawierający dane (dynamiczna tablica)
- fstream - klasa zawierająca metody i atrybuty przeznaczone do operacji na plikach

Niestandardowe biblioteki:

- single include json.hpp - zewnętrzna biblioteka pozwalająca na tworzenie obiektów typu json. Niezbędna w celu zapisu i odczytu z pliku. Link do pobrania i jej dokumentacji umieszczono w bibliografii.

Poniżej umieszczono strukturę plików znajdującą się w projekcie.



Rys. 6. - układ plików projektowych.

6 Opis wykonanych testów

Testy wykonano na komputerze osobistym (laptop) z procesorem Intel Core i5 7 generacji, 8 GB pamięci RAM i sysytemem Windows 10. Listę buggów przedstawiono w tabeli poniżej.

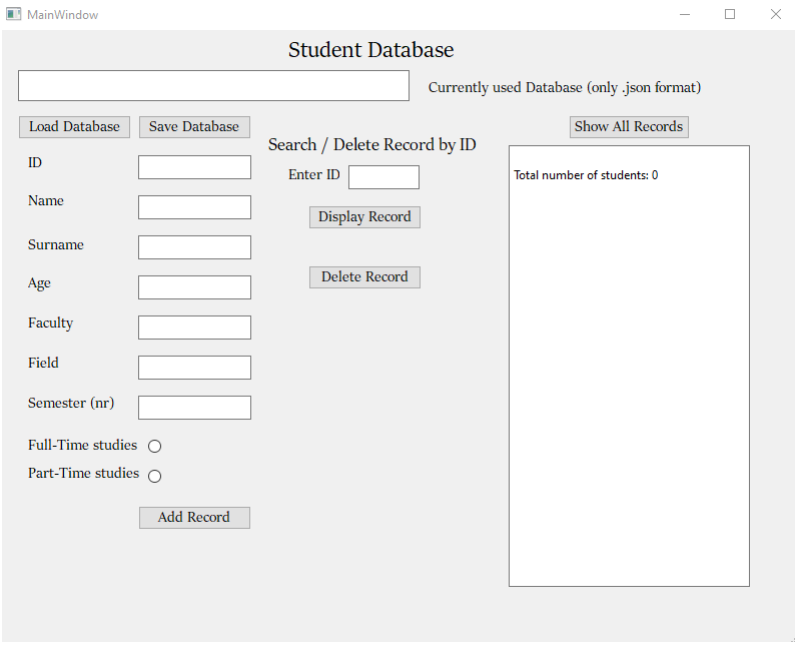
Kod Usterki	Data	Autor	Opis	Stan
0001	30.12.2022	BW	Zapis składowych rekordu do pliku w złej kolejności	Poprawiono
0002	06.01.2023	BW	Nadpisywanie zawartości bazy danych w pliku po jej odczycie i zapisie	Poprawiono
0003	12.01.2023	BW	Odczyt pustego pliku .json powoduje wyłączenie aplikacji	Poprawiono

Tabela. 2. - lista błędów wraz z ich opisem i obecnym stanem

7 Podręcznik użytkownika

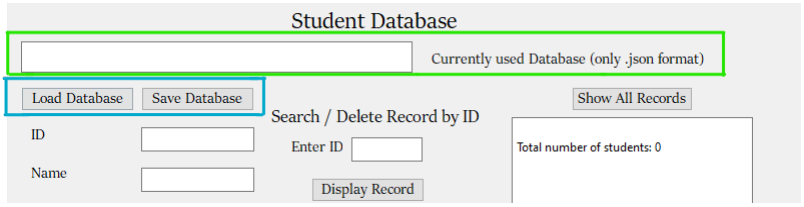
W sekcji tej umieszczono szczegółową instrukcję użytkownika.

1. Panel Główny
   Poniższe zdjęcie przedstawia GUI oferujące możliwość zarządzania całą aplikacją.



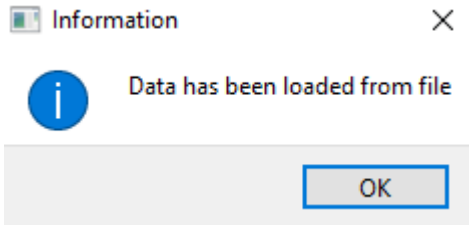
Rys. 7 . - Panel główny aplikacji

2. Odczyt i zapis
   Niebieskim kolorem oznaczono przyciski służące do wymienonych wyżej funkcjonalności.



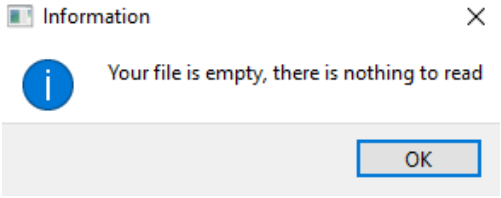
Rys. 8. - Funkcjonalność zapisu i odczytu.

Po użyciu opcji **Load Database** użytkownik otrzyma dostęp do wybrania pliku, który chce załadować. W polu zaznaczonym na zielono będzie widoczna ścieżka do aktualnie wybranego pliku. Jeśli plik zostanie wczytany poprawnie pojawi się komunikat umieszczony poniżej.



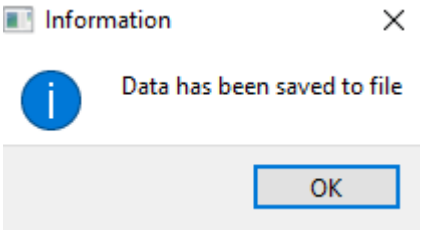
Rys.9. - Komunikat informujący o odczycie z pliku.

Jeśli plik będzie pusty (np. nowo utworzony) odczyt się nię odbędzie a program wyświetli :



Rys. 10 . - Komunikat informujący o tym że odczytany plik jest całkiem pusty

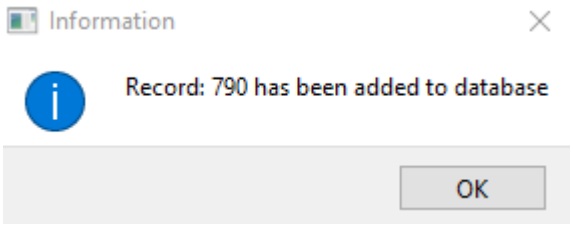
W celu zapisu bazy danych do pliku należy wcisnąć **Save Database**, następnie pojawi się informacja:



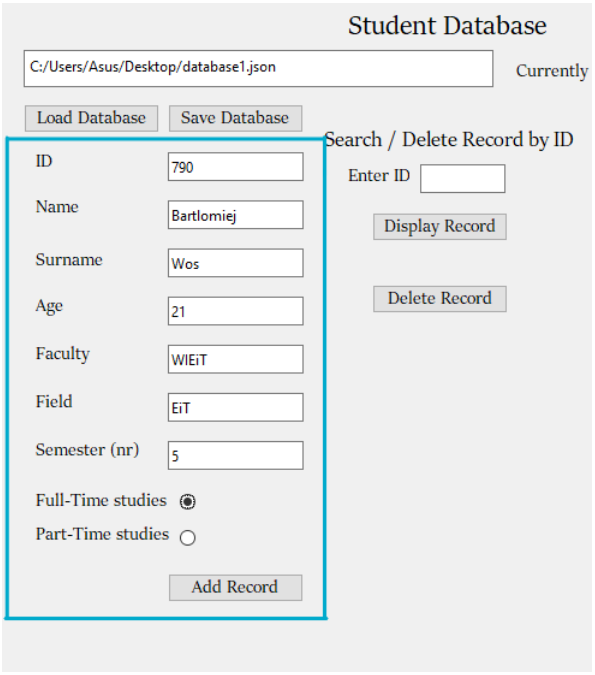
Rys. 11 . - Komunikat informujący o poprawnym zapisie do pliku

3. Dodanie rekordu
   Na niebiesko zaznaczono pole służące do wprowadzania danych wraz z możliwością wprowadzania rekordu. Nie jest możliwe wprowadzenie adresu e-mail ponieważ jest on generowany automatycznie w celu zachowania określonej

systematyczności. Przedstawiono również przykładowy zestaw danych jakie mogą być wprowadzone. Należy pamiętać o odznaczeniu na jaki tryb studiów uczęszcza dana osoba. Aby dodać rekord należy użyć przycisku **Add Record** po wykonaniu tej operacji wyświetli się komunikat przedstawiony poniżej:

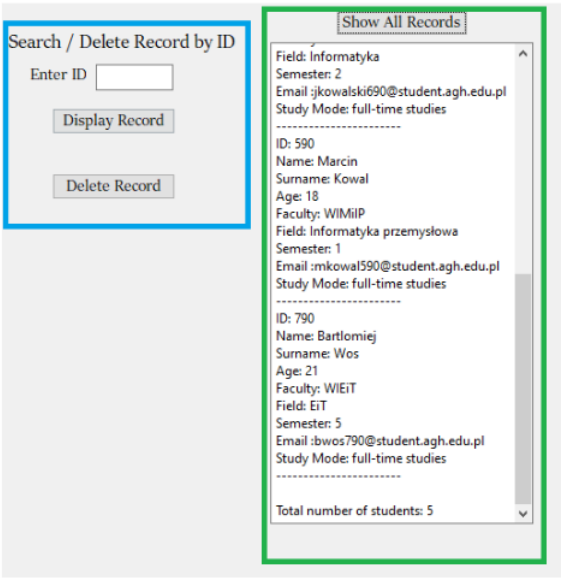


Rys. 12. - Komunikat informujący o dodaniu rekordu



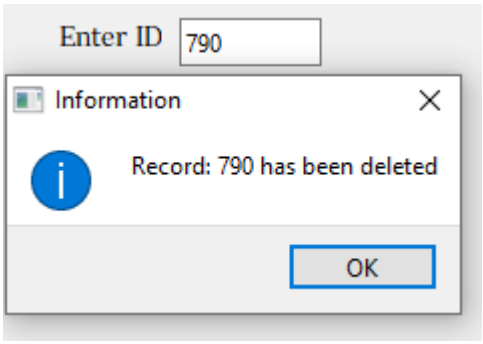
Rys. 13. - Część panelu aplikacji służąca do wprowadzania danych

4. Wyświetlanie, Kasowanie oraz wyszukiwanie rekordu
   Na zielono zaznaczono pole w którym możemy wyświetlać całą bazę danych. Otrzymujemy również informację o liczbie zapisanych osób. Możemy to zrobić za pomocą **Show All Records**.



Rys. 14. - Funkcjonalności wyświetlania, poszukiwania i kasowania rekordu

Na niebiesko zaznaczono Pole służące do wyszukiwania oraz usuwania rekordu. Obydwie te operacje bazują na parametrze ID (unikalny numer studenta). Należy go podać w polu **Enter ID**, za pomocą przycisku **Display Record** możemy wyświetlić szukany rekord w polu zaznaczonym na zielono. W celu wykasowania rekordu należy użyć **Delete Record**, następnie pojawi się informacja który rekord został wykasowany.



Rys. 15 . - Komunikat informujący o skasowaniu konkretnego rekordu.

8 Bibliografia

[1] B. Cyganek - Introduction to Programming with C++ for Engineers

[2] https://github.com/nlohmann/json

[3] Qt Creator Manual - https://doc.qt.io/qtcreator/

[4] https://wiki.qt.io/Qt_for_Beginners