

Dokumentacja Projektu

Języki Programowania Obiektowego

Elektronika i Telekomunikacja

Temat:

Baza Danych z Interfejsem Graficznym

Konrad Jura 417548

Spis treści

1. Cel projektu
2. Wymagania funkcjonalne
3. Wymagania systemowe
4. Analiza problemu
5. Opis rozwiązania
 - 5.1 Opis Klas
 - 5.2 Diagram UML
6. Etapy realizacji zadania
7. Testy
8. Obsługa aplikacji
9. Instalacja i uruchomienie przy użyciu CMake
10. Podsumowanie i wnioski
11. Bibliografia

1. Cel Projektu.

Celem projektu było zaplanowanie a następnie realizacja bazy danych z interfejsem graficznym w języku C++. Ponadto nauczenie się korzystania z biblioteki GTest oraz jej wykorzystania w środowisku Visual Studio, a również zdobycie wiedzy na temat pisania prostych CMake oraz projektowania interfejsów graficznych z użyciem frameworka QT w środowisku QT Creator.

2. Wymagania funkcjonalne.

Baza danych powinna zawierać podstawowe funkcje do obsługi rekordów, takie jak:

- Dodawanie rekordu.
- Usuwanie rekordu.
- Przeszukiwanie bazy danych w celu znalezienia rekordu.
- Zapisanie stanu bazy danych w pliku .txt.
- Wczytanie zapisanego stanu bazy.

Ponadto w interfejsie graficznym dane powinny być zbierane w tabeli, a każda funkcjonalność powinna mieć obsługujący przycisk i pole tekstowe do wprowadzania danych. Interfejs powinien być minimalistyczny oraz zapewniać łatwy dostęp do wszystkich funkcji.

Projekt musi mieć napisane i zdane testy każdej funkcji przy użyciu biblioteki Google Test.

Projekt powinien mieć skonfigurowane pliki CMakeLists.txt, które pozwolą zbudować go w danych środowiskach.

3. Wymagania Systemowe.

Do poprawnego zbudowania projektu przy użyciu CMake, należy zainstalować oprogramowanie QT Creator oraz sam Framework QT w wersji 6.0+.

Zbudowanie testów wymaga posiadania programu Visual Studio 2022+, z zainstalowaną biblioteką Google Test. Projekt musi być napisany w C++, dlatego

należy również mieć możliwość obsługi tego standardu, zalecane jak najnowszego.

4. Analiza Problemu.

W wielu instytucjach zachodzi potrzeba zarządzania danymi osobowymi lub informacjami dotyczącymi danej grupy ludzi takich jak studenci, pracownicy lub członkowie zespołów. Ręczne przechowywanie danych w arkuszach kalkulacyjnych nie jest efektywne, dlatego zachodzi potrzeba stworzenia aplikacji, która zautomatyzuje i uprości zarządzanie danymi, zapewni wygodny interfejs użytkownika i pozwoli trwale przechowywać dane w plikach, dzięki funkcji zapisu i odczytu.

Główne cele i problemy:

- Zaprojektowanie architektury aplikacji.
- Synchronizacja danych między warstwami aplikacji i interfejsu graficznego.
- Wprowadzenie obsługi operacji na danych.
- Stworzenie intuicyjnego i minimalistycznego interfejsu graficznego.
- Obsługa plików (opcja zapisu/odczytu).
- Testowanie aplikacji.

5. Opis Rozwiązania.

Uwagi wstępne:

Do wykonania projektu wykorzystano kod i wiedzę z książki "Introduction to Programming with C++ for Engineers", autor: prof. dr hab. Inż. Bogusław Cyganek.

Do warstwy logiki wykorzystano kod napisany podczas ćwiczeń Języki Programowania Obiektowego.

Do napisania kodu obsługi GUI zaadaptowano kod wygenerowany przez narzędzie ChatGPT oraz przedstawiony przez autora kursu QT z kanału Youtube ProgrammingKnowledge.

Wykorzystane technologie:

Język C++ wersja 20.

Biblioteka QT.

Biblioteka GoogleTest.

Visual Studio 2022.

QT Creator 15.0 (Community).

Aplikacja została stworzona w celu zarządzania danymi w prosty i intuicyjny sposób. Dzięki wykorzystaniu interfejsu graficznego QT, aplikacja łączy wydajność działania z przejrzystym i wygodnym interfejsem.

Jako strukturę bazy danych zdecydowano wybrać “Przechowywanie danych o studencie na uczelni”. Głównym elementem będzie tu obiekt Student, który posiada dane:

- Identyfikator Studenta
- Imię
- Nazwisko
- Semestr

Semestr został realizowany jako osobna klasa, która ma:

- Identyfikator semestru
- Numer semestru (1 lub 2).
- Numer roku

Część GUI została zaprojektowana tak, aby wszystkie dane były wyświetlane w tabeli. Każda funkcja ma przypisany przycisk, który ją wywołuje. Efekty widać właśnie na tabeli z danymi (między innymi dodanie rekordu, zapis, odczyt).

5.1 Opis klas.

Klasa Term (reprezentuje semestr w bazie danych):

Atrybuty – termId, termNr, termYear

Metody – gettery, settery, przeciążenie operatorów strumieniowych <<, >>.

Klasa Student (reprezentuje studenta):

Atrybuty – studentId, studentName, studentSurname, studentTerm

Metody – gettery, settery, przeciążenie operatorów strumieniowych <<, >>.

Klasa DataBase (przechowuje i zarządza studentami):

Atrybuty – wektor obiektów typu Student - vector<Student> students.

Metody – zarządzanie obiektami: add_record, remove_record, find_record. Zapis i odczyt plików: save_file, load_file.

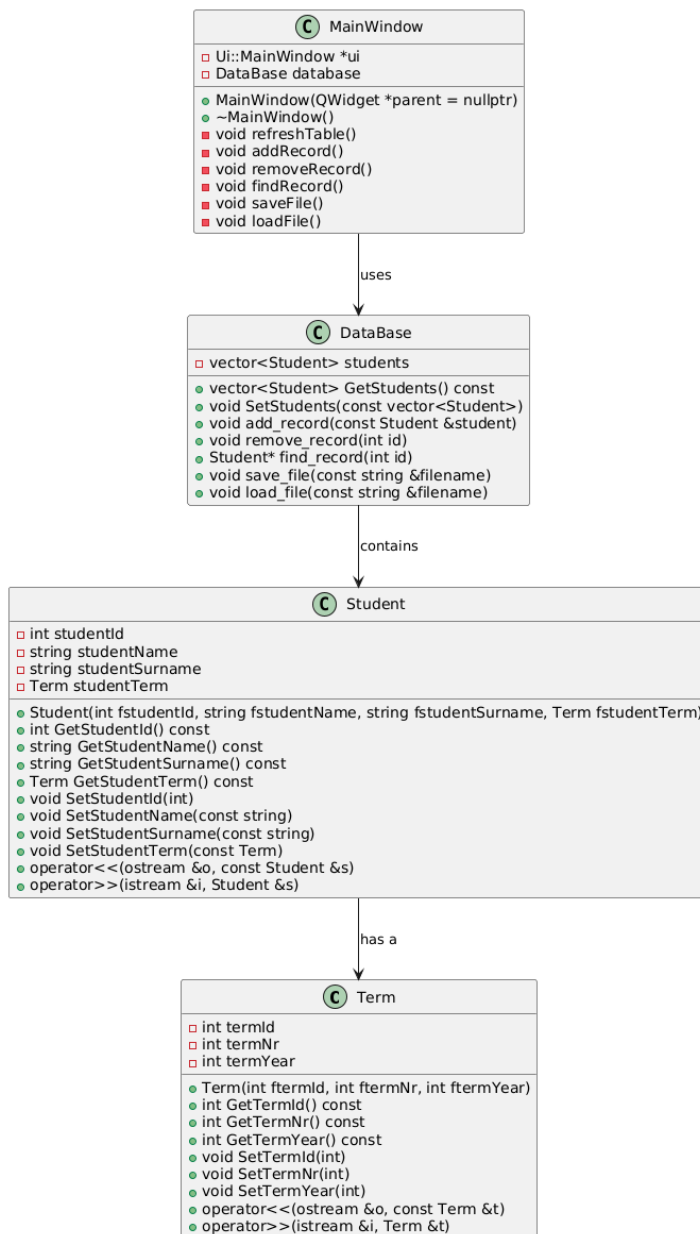
Klasa MainWindow (odpowiada za interfejs graficzny GUI oraz obsługę zdarzeń):

Atrybuty – database – baza danych używana przez GUI, komponenty QT takie jak QTableWidgetItem, QLineEdit, QPushButton.

Metody – obsługa zdarzeń GUI: addRecord, removeRecord, findRecord, saveFile, loadFile. Obsługuje również funkcję odświeżania tabeli refreshTable.

5.2. Diagram UML opisujący relacje między klasami.

Diagram wykonano za pomocą internetowego generatora PlantUML.



Ad. 1. Diagram klas

6. Etapy realizacji zadania.

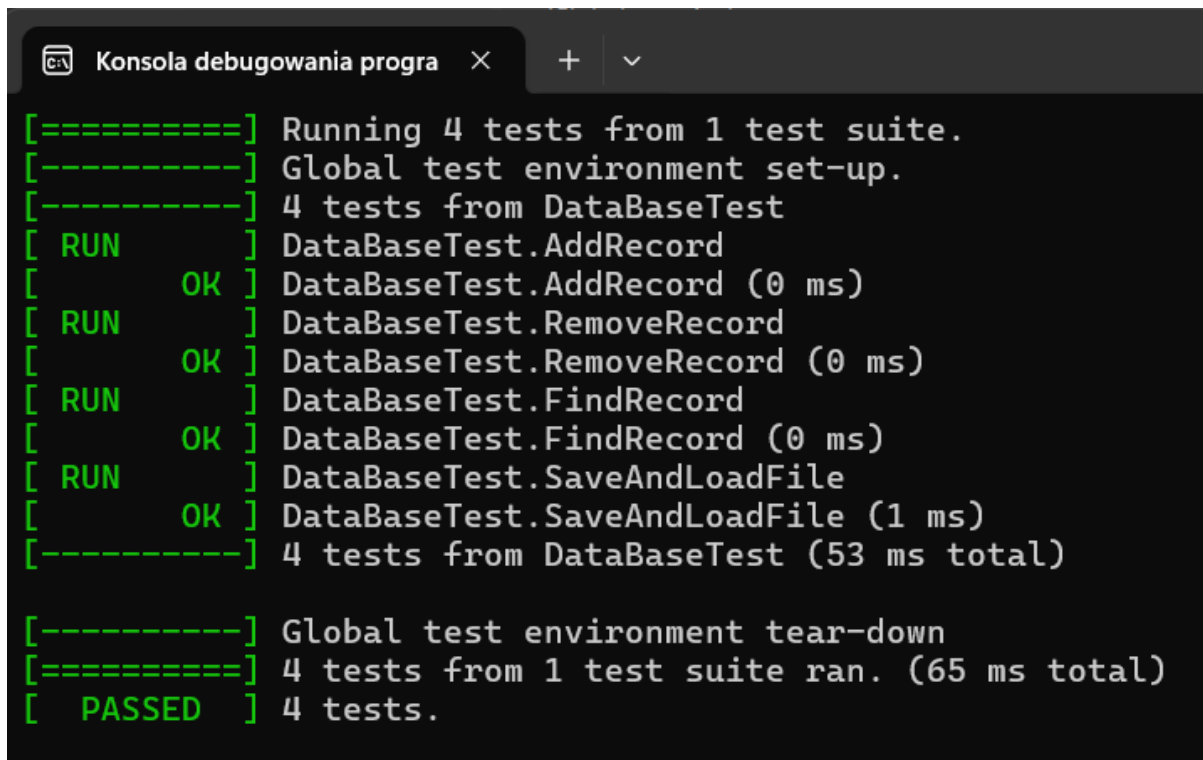
- Zaprojektowanie rozwiązania.
- Napisanie kodu warstwy logiki.
- Przetestowanie kodu warstwy logiki.
- Zapoznanie się z frameworkiem QT.

- Napisanie interfejsu graficznego kompatybilnego z warstwą logiki.
- Potwierdzenie działania finalnej wersji.

7. Testy

Testy wykonano z użyciem biblioteki Google Test w środowisku Visual Studio.

Przeprowadzono test każdej funkcji. Między innymi sprawdzono, czy po dodaniu rekordu do pustej bazy danych znajduje się tam faktycznie 1 wpis, czy po usunięciu go baza danych jest pusta. CMake do przeprowadzenia testu został przepisany z książki “Introduction to Programming with C++ for Engineers”.



```

Konsola debugowania progra x + v
[=====] Running 4 tests from 1 test suite.
[-----] Global test environment set-up.
[-----] 4 tests from DataBaseTest
[ RUN ] DataBaseTest.AddRecord
[ OK ] DataBaseTest.AddRecord (0 ms)
[ RUN ] DataBaseTest.RemoveRecord
[ OK ] DataBaseTest.RemoveRecord (0 ms)
[ RUN ] DataBaseTest.FindRecord
[ OK ] DataBaseTest.FindRecord (0 ms)
[ RUN ] DataBaseTest.SaveAndLoadFile
[ OK ] DataBaseTest.SaveAndLoadFile (1 ms)
[-----] 4 tests from DataBaseTest (53 ms total)

[-----] Global test environment tear-down
[=====] 4 tests from 1 test suite ran. (65 ms total)
[ PASSED ] 4 tests.

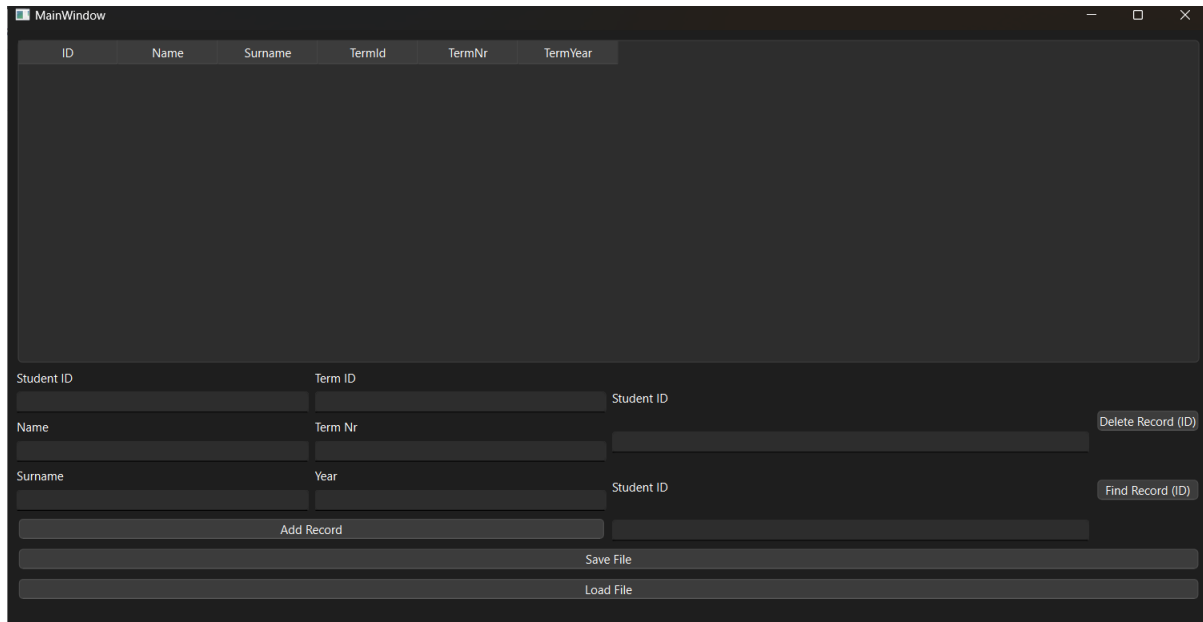
```

Ad. 2. Każdy test przebiegł pomyślnie.

8. Obsługa aplikacji

Obsługa aplikacji jest bardzo prosta. Wpisanie danych w pola tekstowe i naciśnięcie przycisku spowoduje dodanie rekordu do tabeli z danymi. Wpisanie ID studenta i naciśnięcie przycisku usuń spowoduje usunięcie rekordu z tabeli.

Wpisanie ID i naciśnięcie przycisku znajdowania studenta wyświetli informacje o danym rekordzie w tabeli. Zapis spowoduje zapisanie do pliku tekstowego obecnego stanu tabeli, a wczytanie wpisze do tabeli dane z tego pliku tekstowego.



The screenshot shows a window titled "MainWindow" with a table and several input fields. The table has columns: ID, Name, Surname, TermId, TermNr, and TermYear. Below the table, there are input fields for Student ID, Name, Surname, Term ID, Term Nr, and Year. There are also buttons for "Delete Record (ID)", "Find Record (ID)", "Add Record", "Save File", and "Load File".

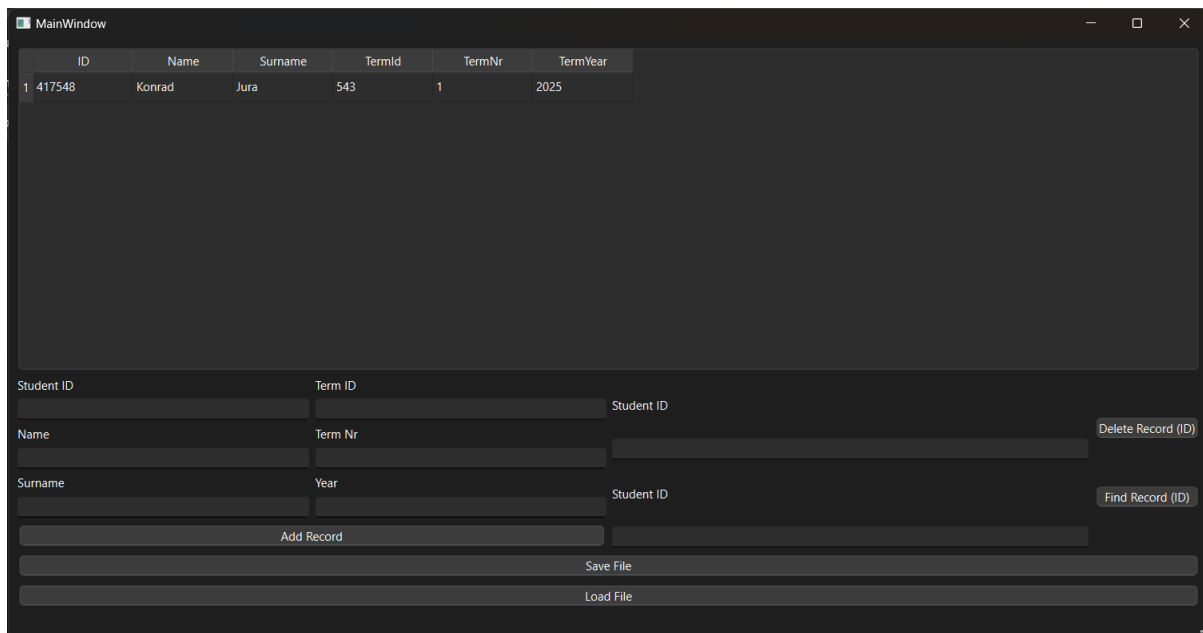
ID	Name	Surname	TermId	TermNr	TermYear
----	------	---------	--------	--------	----------

Student ID: Term ID: Student ID:

Name: Term Nr:

Surname: Year: Student ID:

Ad.3. Baza danych przed dodaniem wpisu.



The screenshot shows the same "MainWindow" window, but now the table contains one record. The input fields and buttons are the same as in the previous screenshot.

ID	Name	Surname	TermId	TermNr	TermYear	
1	417548	Konrad	Jura	543	1	2025

Student ID: Term ID: Student ID:

Name: Term Nr:

Surname: Year: Student ID:

Ad 4. Baza danych po dodaniu wpisu.

9. Instalacja i uruchomienie przy użyciu CMake.

Plik CMake aplikacji został wygenerowany przez program QT Creator przy stworzeniu projektu. Modyfikacji zostały poddane jedynie pliki, które mają być wczytywane (*database.h* oraz *database.cpp*).

Aby załadować projekt, należy w programie QT Creator wybrać “Otwórz projekt” i załadować CMake programu. Następnie wybranie “run” włączy program.

CMake do testów został napisany przez autora książki “Introduction to programming with C++ for Engineers”. Zawiera on dołączenie biblioteki Google Test do programu. Wszystkie niezbędne pliki znajdują się w folderach *src* oraz *include*. Należy stworzyć folder, w którym ma znajdować się rozwiązanie i z poziomu terminala użyć komendy “*cmake ..*”. Następnie należy otworzyć zbudowane rozwiązanie i uruchomić program.

10. Podsumowanie i wnioski.

Projekt pozwolił lepiej zrozumieć projektowanie i implementację kodu w programowaniu zorientowanym obiektowo.

Nauczono się korzystać z aplikacji GUI do kodu i zrozumiano obsługę QT. Zapoznano się z możliwościami oferowanymi przez GoogleTest.

Zweryfikowano wiedzę o CMake.

Nauczono się efektywnie zarządzać projektem.

Stworzono aplikację, która może być rozwijana w przyszłości o funkcjonalności takie jak zwiększenie bezpieczeństwa, system logowania do bazy danych, walidacja dodawanych wpisów, zapis na inne formaty niż *.txt*.

11. Bibliografia.

“Introduction to Programming with C++ for Engineers”, autor: prof. dr hab. Inż.
Bogusław Cyganek.

Fragmenty kodu QT wygenerowane przez narzędzie ChatGPT.

Kanał Youtube ProgrammingKnowledge

Cppreference.com