

Raport techniczny projektu

BAZA DANYCH Z

INTERFEJSEM GRAFICZNYM

Projekt wykonał:

Huber Pałka

Akademia Górniczo-Hutnicza

Elektronika i Telekomunikacja

Spis treści

| | | |
|----|----------------------------|---|
| 1. | Wstęp | Błąd! Nie zdefiniowano zakładki. |
| 2. | Obsługa Aplikacji | Błąd! Nie zdefiniowano zakładki. |
| 3. | Opis realizacji | 3 |
| 4. | Funkcjonalność | Błąd! Nie zdefiniowano zakładki. |
| 5. | Napotkane problemy | Błąd! Nie zdefiniowano zakładki. |
| 6. | Możliwości rozbudowy | 7 |
| 7. | Podsumowanie | Błąd! Nie zdefiniowano zakładki. |
| 8. | Bibliografia | 8 |

1. Wprowadzenie

Projekt "CarDatabase_HP" jest aplikacją desktopową zrealizowaną w technologii Microsoft Foundation Classes (MFC). Głównym celem aplikacji jest zarządzanie bazą danych pojazdów poprzez dodawanie, wyszukiwanie, usuwanie i edycję danych samochodów. Program ma za zadanie dostarczyć użytkownikom intuicyjny interfejs umożliwiający sprawne operowanie na plikach tekstowych, pełniących funkcję bazy danych.

2. Funkcjonalność aplikacji

Aplikacja oferuje następujące funkcje:

- **Dodawanie pojazdu** – użytkownik może wprowadzać dane pojazdu (producent, model, rok produkcji, kolor), które zostaną zapisane w pliku tekstowym cars.txt
- **Wyszukiwanie pojazdu** – użytkownik może wyszukiwać pojazd na podstawie wpisanych przez siebie kryteriów (np. producent i model), aplikacja wyświetli wszystkie pasujące wyniki
- **Usuwanie pojazdu** – użytkownik może usunąć wcześniej dodany pojazd poprzez podanie wszystkich jego danych
- **Czyszczenie bazy** – użytkownik ma możliwość całkowitego wyczyszczenia bazy danych

3. Opis realizacji

Celem projektu było stworzenie aplikacji MFC umożliwiającej zarządzanie bazą danych samochodów. W tym celu skorzystano z poniższych narzędzi oraz technologii:

- Microsoft Visual Studio 2022 – środowisko programistyczne,
- MFC (Microsoft Foundation Class) – wykorzystane do stworzenia interfejsu graficznego,
- CMake – do zarządzania budowaniem projektu,
- GitHub – do kontroli wersji i przechowywania kodu
- Google Test – do testowania kodu
- Pliki tekstowe txt – do przechowywania danych

Projekt rozpoczęto od stworzenia wszystkich elementów interfejsu MFC. Następnie przystąpiono do tworzenia każdej z funkcji.

4. Analiza problemu

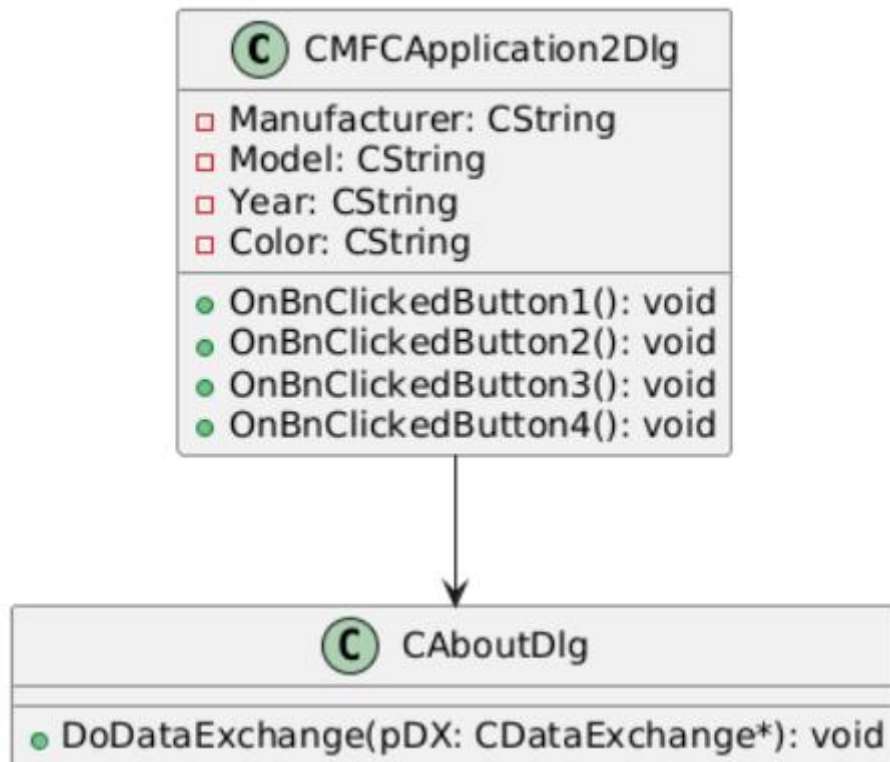
Główne wyzwania projektu obejmowały:

- **Obsługa plików tekstowych** – projekt wymagał niezawodnej obsługi operacji odczytu i zapisu plików.
- **Walidacja danych** – upewnienie się, że użytkownik wprowadza kompletne dane przed zapisem
- **Intuicyjny interfejs użytkownika** – zapewnienie prostego w obsłudze i intuicyjnego interfejsu do obsługi dla użytkownika
- **Zapewnienie możliwości stosowania wcześniej przygotowanych baz danych** – dzięki skorzystaniu z formatu .txt, program będzie rozpoznawał wcześniej przygotowaną bazę

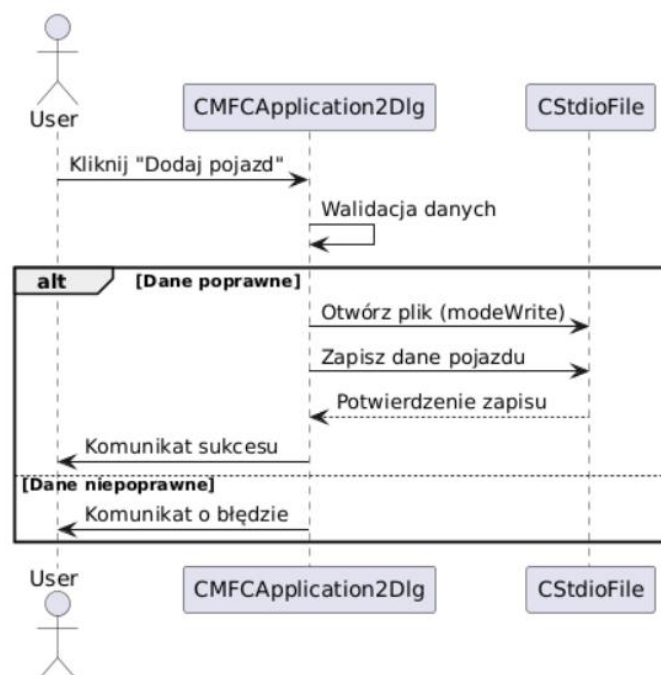
danych, pod warunkiem poprawnego formatowania tekstu wewnątrz pliku oraz odpowiedniej nazwy cars.txt

5. Diagramy UML

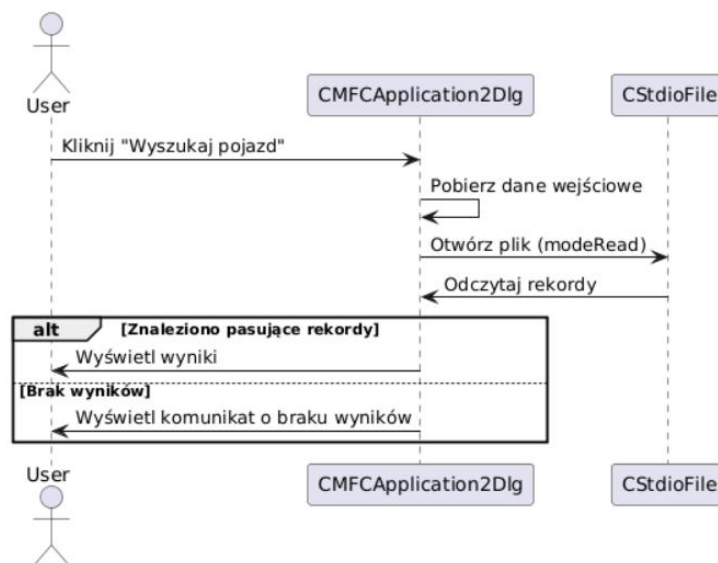
1. Diagram klas



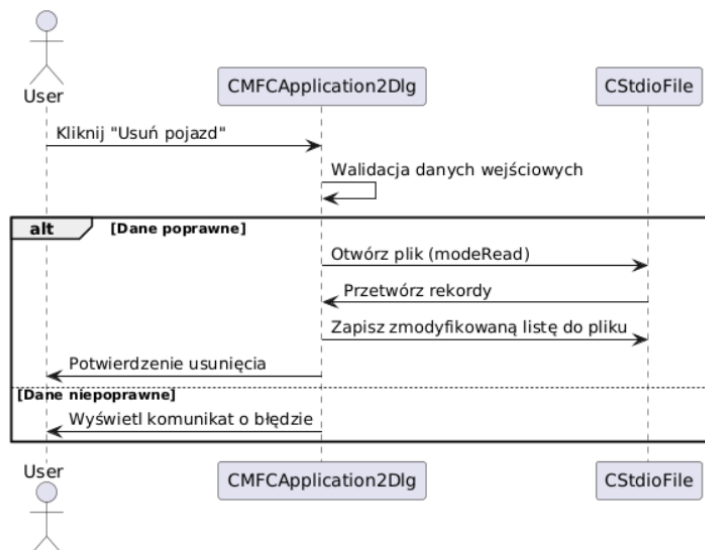
2. Diagram sekwencji (dodawanie pojazdu)



3. Diagram sekwencji (wyszukiwanie pojazdu)



4. Diagram sekwencji (usuwanie pojazdu)



6. Implementacja

Kod źródłowy aplikacji składa się z następujących elementów:

1. CMFCAplication2Dlg: Główna klasa odpowiadająca za interfejs użytkownika
2. CStdioFile: Klasa wykorzystana do obsługi plików tekstowych
3. Funkcje zdarzeń: Implementacje funkcji odpowiadających za poszczególne funkcjonalności (np. OnBnClickedButton1)

Przykładowy kod implementacji dodawania pojazdu:

```

void CMFCAplication2Dlg::OnBnClickedButton1()
{
    UpdateData(TRUE);

    if (Manufacturer.IsEmpty() || Model.IsEmpty() || Year.IsEmpty() || Color.IsEmpty()) {

```

```

        AfxMessageBox(_T("All fields must be filled before adding data!"));

        return;
    }

    CStdioFile file;

    CString filePath = _T("cars.txt");

    if (file.Open(filePath, CFile::modeCreate | CFile::modeNoTruncate | CFile::modeWrite)) {

        file.SeekToEnd();

        CString carData;

        carData.Format(_T("Manufacturer: %s, Model: %s, Year: %s, Color: %s\n"),

            Manufacturer, Model, Year, Color);

        file.WriteString(carData);

        file.Close();

        AfxMessageBox(_T("Car data has been saved!"));

        Manufacturer = _T("");

        Model = _T("");

        Year = _T("");

        Color = _T("");

        UpdateData(FALSE);

    } else {

        AfxMessageBox(_T("ERROR: Unable to access the database!"));

    }

}

```

7. Napotkane problemy

W trakcie tworzenia aplikacji napotkano różne problemy. Jednym z nich było poprawne tworzenie oraz zapisywanie plików. Błąd został naprawiony poprzez użycie odpowiednich funkcji CFile:: w funkcji open. Pojawił się także problem ze zmianą nazwy projektu. Niestety ze względu na mnogość występowania nazwy projektu w plikach aplikacji, zmienione zostały tylko podstawowe nazwy, tak aby nie utracić działania aplikacji.

8. Testy

Testy działania aplikacji wykonano przy użyciu GoogleTest. Poprzez stworzenie projektu testowego obsługującego GoogleTest sprawdzono działanie dodawania, usuwania i wyszukiwania pojazdu.

```

Running main() from D:\a_work\1\s\googletest\googletest\src\gtest_main.cc
[=====] Running 3 tests from 1 test case.
[=====] Global test environment set-up.
[=====] 3 tests from CarDatabaseTests
[ RUN      ] CarDatabaseTests.AddCar
[ OK       ] CarDatabaseTests.AddCar (9 ms)
[ RUN      ] CarDatabaseTests.SearchCar
[ OK       ] CarDatabaseTests.SearchCar (2 ms)
[ RUN      ] CarDatabaseTests.DeleteCar
[ OK       ] CarDatabaseTests.DeleteCar (3 ms)
[=====] 3 tests from CarDatabaseTests (17 ms total)

[=====] Global test environment tear-down
[=====] 3 tests from 1 test case ran. (19 ms total)
[ PASSED  ] 3 tests.

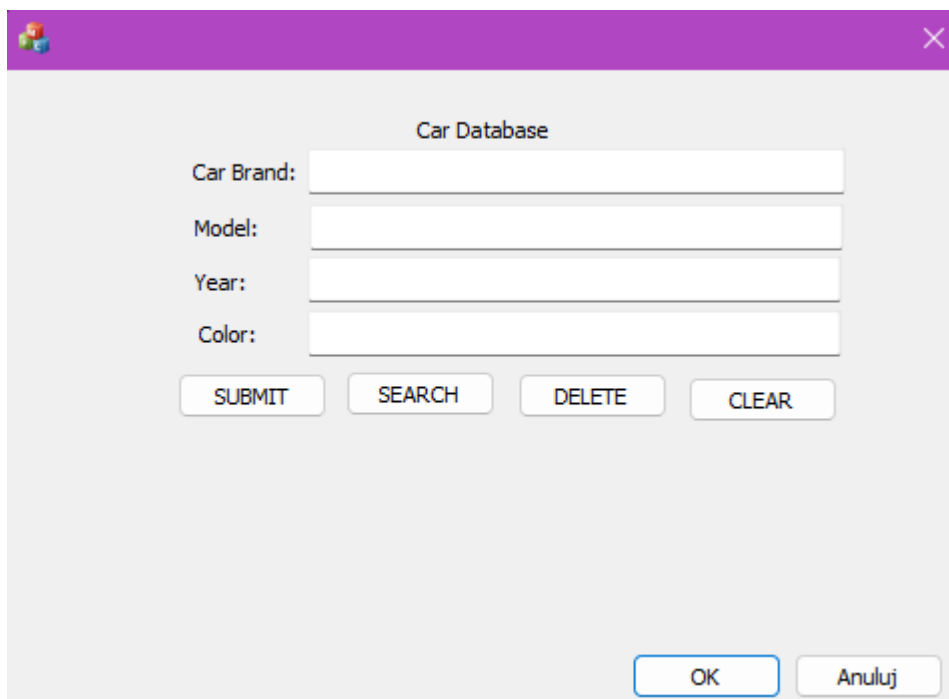
C:\Users\palka\source\repos\Database_Test\x64\Debug\Database_Test.exe (proces 23188) zakończono z kodem 0 (0x0).
Aby automatycznie zamknąć konsolę po zatrzymaniu debugowania, włącz opcję Narzędzia -> Opcje -> Debugowanie -> Automatyc
znie zamknij konsolę po zatrzymaniu debugowania.
Naciśnij dowolny klawisz, aby zamknąć to okno..|

```

9. Możliwości rozbudowy

Projekt można rozbudować o kolejne dane opisujące pojazd, jak na przykład numer rejestracyjny, właściciel czy chociażby kraj pochodzenia samochodu. Dodatkowo, można także rozbudować program poprzez dodanie funkcjonalności importowania gotowych baz danych z pliku CSV lub JSON.

10. Krótka instrukcja obsługi programu



Aby dodać pojazd do bazy danych, należy podać wszystkie dane pojazdu (producent, model, rok produkcji oraz kolor).

Aby wyszukać pojazd, można podać tylko wybrane elementy, na przykład model i rok produkcji. W przypadku nie podania żadnych danych, aplikacja wyświetli wszystkie pojazdy znajdujące się w bazie danych.

Aby usunąć pojazd z bazy danych, należy podać wszystkie jego dane. Brak danych skutkuje nie wykonaniem polecenia.

Aby wyczyścić bazę danych, wystarczy tylko wcisnąć przycisk clear.

11. Podsumowanie

Projekt można uznać za zakończony sukcesem, wszystkie założenia początkowe projektu zostały zrealizowane. Realizacja projektu umożliwiła zdobycie wiedzy z zakresu projektowania aplikacji z interfejsem wizualnym. Aplikacja działa zgodnie z założeniami, umożliwiając tworzenie użytkownikowi bazy danych z interfejsem graficznym oraz wykorzystując wymagane metody.

12. Bibliografia

<https://learn.microsoft.com/en-us/cpp/mfc/mfc-desktop-applications?view=msvc-170>

<https://learn.microsoft.com/en-us/cpp/mfc/reference/cstdiofile-class?view=msvc-170>

<https://stackoverflow.com/questions/11580748/using-cmake-for-making-a-project-which-includes-mfc>

<https://github.com/google/googletest/blob/main/googletest/README.md>

<https://www.youtube.com/watch?v=bfCYGmWolgQ>

Programowanie w języku C++ Wprowadzenie dla inżynierów – Dr hab. inż. Bogusław Cyganek