|  |  |
| --- | --- |
|  | **AKADEMIA GÓRNICZO-HUTNICZA** |

Dokumentacja do projektu

**System rezerwacji kolejowej**

z przedmiotu

**Języki programowania obiektowego**

Elektronika I

*Tomasz Mróz, Jakub Domagała*

Piątek 16:15

prowadzący: Rafał Frączek

09.06.2022

# Opis projektu

Założeniem projektu jest stworzenie programu do rezerwacji biletów linii kolejowych na dowolnej wybranej trasie.

# Project description

The main idea of the project is to create a program for booking railway tickets on any chosen route.

# Instrukcja użytkownika

Działanie programu rozpoczyna okno logowania. W oknie logowania użytkownik musi się zarejestrować jeżeli nie ma założonego wcześniej konta. Następnie zostajemy przeniesieni do okna głównego w którym wyświetlana zostaje tabela z dostępnymi połączeniami, naciśnięcie na jedną z podanych linii ukaże obok opcje rezerwacji biletu. W zakładce menu możemy przejść do okna „Dodaj Połączenie” w którym możemy dodać linię kolejową gdzie podajemy miasta , które chcemy aby miały połączenie między sobą oraz ilość miejsc w pociągu. W zakładce Moje bilety możemy sprawdzić bilety które rezerwowaliśmy.

# Kompilacja

Do kompilacji programu wymagane jest posiadanie na komputerze biblioteki Qt 6.4.2 oraz wtyczki Qt tool do programu Microsoft visual studio 2022. Projekt uruchomić można wchodząc w zakładkę rozszerzenie -> Qt tool -> otworzyć plik .pro

# Pliki źródłowe

Projekt składa się z następujących plików źródłowych:

* *mainwindow.h*, *mainwindow.cpp, mainwindow.ui* – deklaracja, implementacja klasy oraz szata graficzna głównego okna programu
* *loginwindow.h, loginwindow.cpp, loginwindow.ui* – deklaracja, implementacja klasy oraz szata graficzna okna rejestracji i logowania
* *addtrainwindow.h, addtrainwindow.cpp, addtrainwindow.ui* – deklaracja, implementacja klasy oraz szata graficzna okna dodawania połączeń
* *dbmanager.h, dbmanager.cpp* – deklaracja oraz implementacja baza danych pociągów
* *mytickets.h, mytickets.cpp, mytickets.ui* – deklaracja, implementacja klasy oraz szata graficzna okna wyświetlania swoich biletów
* *registerdb.h, registerdb.cpp* – deklaracja oraz implementacja bazy danych rejestracji
* *user.h, user.cpp* – deklaracja oraz instancje zmiennych globalnych
* *userdb.h, userdb.cpp* – deklaracja oraz implementacja bazy danych biletów a użytkowników.

# Zależności

W projekcie wykorzystano następujące dodatkowe biblioteki:

* SQLite – silnik bazodanowy SQL. Strona internetowa: <https://sqlite.org/>.
* Qt – zestaw przenośnych bibliotek i narzędzi programistycznych dedykowanych dla języków C++, QML i Java, umożliwiających tworzenie oprawy graficznej aplikacji. <https://www.qt.io/?hsLang=en>

# Opis klas

W projekcie utworzono następujące klasy:

* MainWindow – klasa reprezentująca główne okno pliku
  + void AddTrainToScroll(int id) – dodaje linie kolejową do listy
  + void updateTable() – metoda odświeżająca tabele połączeń
* AddTrainWindow – klasa zawierająca konstruktor i destruktor, służy do tworzenia linii kolejowych
* LoginWindow – klasa zawierająca konstruktor i destruktor odpowiada za logowanie się do aplikacji
* UserDb – klasa zawierająca metody umożliwiające działanie bazy danych
  + bool isOpen() const – metoda sprawdzająca czy baza danych jest otwarta
  + bool createTable() – metoda tworząca tabele w bazie danych
  + bool userAdd(const QString& userName, const QString& code, int tickets) – metoda umożliwająca dodania danych do bazy danych
  + bool userExists(const QString& username) const – metoda sprawdzająca istnienie danych dla podanego użytkownika
  + void UsersRecords(QString username) – metoda zwracająca dane dla danego użytkownika
* RegisterDb – klasa zawierająca metody umożliwiające działanie bazy danych
  + bool isOpen() const – metoda sprawdzająca czy baza danych jest otwarta
  + bool createTable() – metoda tworząca tabele w bazie danych
  + bool userAdd(const QString& userName, const QString& password) – metoda umożliwająca dodania danych do bazy danych
  + bool userExists(const QString& username) const – metoda sprawdzająca istnienie danych dla podanego użytkownika
  + QString userPassword(QString username) – metoda zwracająca hasło danego użytkownika
* MyTickets – klasa obsługująca okno „moje bilety”
  + void drawScrollSArea(QVBoxLayout \*lay) – metoda wyświetlająca listę biletów
* DbManager – klasa zawierająca metody umożliwiające działanie bazy danych
  + bool isOpen() const – metoda sprawdzająca czy baza danych jest otwarta
  + bool createTable() – metoda tworząca tabele w bazie danych
  + bool addTrain(const QString& code, const QString& code, const QString &to, int seats) – metoda umożliwająca dodania danych do bazy danych
  + bool trainExists(const QString& code) const – metoda sprawdzająca istnienie danych dla podanego użytkownika
  + bool removeTrain(const QString& code) – metoda umożliwiająca usuwanie danych
  + void printAllTrains() const – metoda wyświetlająca wszystkie dane bazy danych
  + QString printCodeToTableByID(int id) – metoda zwracająca kod
  + QString printFromToTableByID(int id) – metoda zwracająca cel
  + QString printToToTableByID(int id) – metoda zwracająca miejsce wyjazdu
  + int printSeatsToTableByID(int id) – metoda zwracająca liczbę miejsc
  + bool removeAllTrain() – metoda usuwająca wszystkie dane
  + void updateSeatsNumber(int id, int seats) – metoda nadpisująca liczbę miejsc
  + int printIDByCode(QString code) – metoda zwracająca id
  + int countAllTrains() – metoda zliczająca ilość danych w bazie danych

# Zasoby

* database.db – baza danych do której wysyłamy wszystkie dane, korzystamy z jednej bazy danych i tworzymy w niej listy do przechowywania danych.

# Dalszy rozwój i ulepszenia

Usprawnienie działalności okna „Moje bilety”, poprawa estetyczna oraz usprawnienie działania okna „logowania”, usuwanie linii kolejowych, możliwość zwrotu/reklamacji/ rezygnacji z rezerwacji biletów, usunięcie pociągu z listy po wyczerpaniu się miejsc dostępnych. Ogólne poprawy, optymalizacja aplikacji.

# Inne

brak