Dokumentacja do projektu

**Car rental system**

z przedmiotu

Programowanie obiektowe

Elektronika – 2 rok

Jakub Lasak

Piątek 11:20

Prowadzący: Rafał Frączek

24.01.2023

1. **Opis projektu**

System wypożyczalni samochodów, zawierający opcje dostępu zarówno dla osoby prowadzącej (administrator) oraz dla klienta.

Możliwości administratora:

* Dodawanie pojazdów
* Usuwanie pojazdów
* Wgląd do bazy pojazdów
* Wgląd do bazy klientów
* Wgląd do bazy zawartych transakcji

Opcje klienta:

* Utworzenie indywidualnego konta w serwisie
* Możliwość zalogowania się do tego konta w dowolnej chwili
* Wypożyczanie pojazdów
* Zwrot wypożyczonych pojazdów

1. **Project description**

Car rental system, containing access options for both the operator (administrator) and the client.

Admin capabilities:

• Adding vehicles

• Deleting vehicles

• View to the vehicle database

• Insight into the customer database

• Access to the database of concluded transactions

Customer Options:

• Creating an individual account on the site

• The ability to log into this account at any time

• Vehicle rental

• Return of rental vehicles

1. **Instrukcja obsługi:**
2. Menu główne

W menu głównym opcje wybieramy za pomocą cyfr 1-4, poszczególne cyfry wywołują odpowiednie funkcje.

1 – Admin

2 – Customer

3 – Car base

4 - Exit

1. Poziom dostępu administratora

Po wybraniu opcji 1 w menu głównym zostaniemy przeniesieni do menu w którym możemy zalogować się do systemu z poziomu administratora, po wprowadzeniu odpowiednich danych użytkownik uzyskuje dostęp do opcji administratora. Może dodawać i usuwać pojazdy z bazy a także ma wgląd do bazy wypożyczeń.

1. Poziom dostępu klienta

Po wybraniu numeru 2 użytkownik przenosi się do menu w którym może zalogować się do istniejącego konta lub założyć nowe. Po wybraniu opcji „zarejestruj się” program poprosi o wprowadzenie danych użytkownika a po wpisaniu ich przeniesie użytkownika do poprzedniego menu. Jeżeli natomiast klient wybierze opcje aby się zalogować zostanie poproszony o wprowadzenie danych swojego konta. Loginem jest indywidualny kod ID a hasło jest wybierane przez użytkownika w trakcje rejestracji. Także indywidualny kod ID przypisany do konta jest podawany po pomyślnej rejestracji konta.

Po zalogowaniu użytkownik może wypożyczyć dostępne samochody, zwrócić już wypożyczone a także wyświetlić bazę danych. Może także wyświetlić swoje dane.

1. **Kompilacja:**

Projekt kompiluje się standardowo. Działa na każdym systemie operacyjnym.

1. **Pliki źródłowe:**

* Car.h, Car.cpp – deklaracja i implementacja klasy Car
* Customer.h , Customer.cpp – deklaracja i implementacja klasy Customer
* GlobalFun.h , GlobalFun.cpp – zbiór metod dostępnych we wszystkich plikach
* Rent.h , Rent.cpp – deklaracja i implementacja kasy Rent
* CarBase.h , CarBase.cpp – deklaracja i implementacja klasy CarBase
* CustomerBase.h , CustomerBase.cpp – deklaracja i implementacja klasy CustomerBase
* RentBase.h , RentBase.cpp – deklaracja i implementacja klasy RentBase
* Screen.h , Screen.cpp – zbiór metod dostępnych we wszystkich plikach

1. **Zależności:**

W projekcie zostały wykorzystane standardowe, wbudowane biblioteki.

1. **Opis klas:**

* main.cpp – główny plik zawierający funkcję main oraz inne
* void main\_menu() – funkcja odpowiadająca za generację menu głównego
* void m1\_admin() – generuje menu logowania do poziomu administrator
* void m2\_admin() – wyświetla menu po zalogowaniu jako administrator z dostępnymi opcjami
* void adding\_car() – procedura dodawania samochodu do bazy
* void delating\_car() – procedura usuwania pojazdu z bazy
* void m1\_user() – menu logowania do poziomu użytkownika
* void loggin\_in() – procedura logowania do konta użytkownika
* void user\_menu– menu użytkownika po zalogowaniu z dostępnymi opcjami
* void registration(RentBase& r\_base, CustomerBase& c\_base) – procedura rejestracji nowego konta użytkownika
* bool car\_available(shared\_ptr<Car>& abc) – sprawdza czy samochód jest dostępny
* Car – reprezentuje samochód
* Pola klasy:
* string c\_plate{};
* string c\_class{};
* string c\_type{};
* string c\_brand{};
* string c\_model{};
* string c\_color{};
* string c\_fuel{};
* string c\_transmission{};
* string c\_year{};
* string c\_engine{};
* string c\_hp{};
* string c\_seats{};
* string c\_price{};
* bool c\_borrowed;
* Gettery i setery
* Konstruktor domyślny
* Customer – reprezentuje użytkownika
* Pola:
* string u\_fname{};
* string u\_lname{};
* string u\_age{};
* string u\_mail{};
* string u\_phone\_number{};
* string u\_id{};
* string u\_password{};
* Gettery i setery
* Konstruktor domyślny
* Rent – reprezentuje wypożyczenie
* Pola:
* vector<shared\_ptr<Car>> rent\_cars;
* shared\_ptr<Customer> rent\_cus;
* Metody:
* shared\_ptr<Car>& get\_one\_rent\_car(int i){return rent\_cars.at(i);} – zwraca wybrany samochód z listy
* vector<shared\_ptr<Car>>& get\_rent\_cars(); - zwraca całą listę samochodów
* shared\_ptr<Customer>& get\_rent\_cus(); - zwraca użytkownika
* void set\_rent\_cars(vector<shared\_ptr<Car>>& cars);
* void set\_rent\_cus(shared\_ptr<Customer>& cus);
* void remove\_car(int num); - usuwa pojazd
* void show\_rent(); - wyświetla obiekt klasy rent
* CarBase – przechowuje obiekty klasy Car, baza pojazdów
* Pola:
* vector<shared\_ptr<Car>> base;
* Metody:
* CarBase(); - konstruktor domyślny
* void add\_car(shared\_ptr<Car>& car); - dodaje samochód do listy
* void delate\_car(string num\_plate); - usuwa samochód z listy
* void upload\_base(); - pobiera wartości z pliku i wpisuje je do bazy
* void update\_base(); - wpisuje do pliku zawartość bazy
* void show\_base(); - wyświetla bazę
* shared\_ptr<Car>& get\_car(int i){return base.at(i);} – zwraca wybrany pojazd z listy
* size\_t get\_base\_size(){return base.size();} – zwraca rozmiar bazy
* CustomerBase – przechowuje obiekty klasy Customer
* Pola:
* vector<shared\_ptr<Customer>> cusBase;
* Metody:
* CustomerBase(); - konstruktor domyślny
* void add\_customer(shared\_ptr<Customer>& customer); - dodaje nowego użytkownika
* void show\_cus\_base(); - wyświetla bazę użytkowników
* void upload\_cus\_base(); - pobiera z pliku i wpisuje je do bazy
* shared\_ptr<Customer> & get\_customer(int i){return cusBase.at(i);} – zwraca wybranego użytkownika
* RentBase – przechowuje obiekty klasy Rent
* Pola:
* vector<shared\_ptr<Rent>> rent\_base;
* Metody:
* RentBase(); - konstruktor domyślny
* void set\_rent\_base(vector<shared\_ptr<Rent>> base); - ustawia bazę
* void add\_rent(shared\_ptr<Rent>& rent); - dodaje nowe wypożyczenie
* void upload\_rent\_base(); - pobiera wartości z pliku i wpisuje do bazy
* void update\_rent\_base(); - zapisuje wartości z bazy do pliku
* void show\_whole\_base(); - wyświetla zawartość bazy
* shared\_ptr<Rent>& get\_one\_rent(int i) { return rent\_base.at(i); } – zwraca jedno wypożyczenie
* size\_t get\_base\_size() { return rent\_base.size(); } – zwraca rozmiar bazy
* GlobalFun.cpp
* extern bool is\_file\_empty(fstream& file); - zwraca true gdy plik jest pusty
* extern bool str\_to\_bool(string& line); - zamienia string na bool
* extern bool isNumber(const string& str); - sprawdza czy string jest liczbą
* extern bool check\_option(string choice, vector<string> options); - sprawdza czy wpisana wartość znajduję się w zbiorze dostępnych wyborów
* Screen.cpp
* extern void fullscreen(); - przełącza okno w tryb pełnoekranowy
* extern const string currentDataTime(); - zwraca obecną godzinę i datę
* extern void gotoxy(int x, int y); - przenosi kursor w punkt (x,y)
* extern void Boarder(); - wyświetla ramkę
* extern void art(); - wyświetla napis
* extern void main\_menu\_load(); - ekran ładowania
* extern void pre\_login(); - ekran ładowania
* extern void updating\_data\_base(); - ekran ładowania
* extern void pre\_logout(); - ekran ładowania
* extern void welcome(); - napis powitalny
* extern void exit\_imag(); - napis pożegnalny

1. **Zasoby:**

* CarBase.txt

Lina:

1. Tablica rejestracyjna
2. Klasa
3. Typ
4. Marka
5. Model
6. Kolor
7. Rodzaj paliwa
8. Skrzynia biegów
9. Rok produkcji
10. Silnik
11. Konie mechaniczne
12. Ilość miejsc
13. Koszt za 24 godziny
14. Dostępność

W kolejnych liniach dane wpisywane są w podanej wyżej kolejności.

* CustomerBase.txt

1. ID
2. Hasło
3. Imię
4. Nazwisko
5. Wiek
6. Mail
7. Numer telefonu

W kolejnych liniach dane wpisywane są w podanej wyżej kolejności.

* RentBase.txt

1. ID
2. Hasło
3. Imię
4. Nazwisko
5. Wiek
6. Mail
7. Numer telefonu
8. Ilość wypożyczonych samochodów
9. Tablica rejestracyjna
10. Klasa
11. Typ
12. Marka
13. Model
14. Kolor
15. Rodzaj paliwa
16. Skrzynia biegów
17. Rok produkcji
18. Silnik
19. Konie mechaniczne
20. Ilość miejsc
21. Koszt za 24 godziny
22. Dostępność

W kolejnych liniach dane wpisywane są w podanej wyżej kolejności.

* Welcome.txt – obrazek powitalny
* Art.txt – obrazek z nazwą projektu
* ExitArt.txt – obrazek pożegnalny

1. **Dalszy rozwój i ulepszenia:**

* Implementacja interfejsu graficznego
* Dodanie funkcji naliczającej opłaty z tytułu wypożyczenia
* Przeniesienie bazy danych na serwer