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

2. 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

3. Instrukcja obsługi:

1) 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
2) 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.
3) 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.

4. Kompilacja:

Projekt kompiluje się standardowo. Działa na każdym systemie operacyjnym.

5. 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

6. Zależności:

W projekcie zostały wykorzystane standardowe, wbudowane biblioteki.

7. 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
```

8. Zasoby:

- CarBase.txt

  Lina:

  I. Tablica rejestracyjna
  II. Klasa
  III. Typ
  IV. Marka
  V. Model
  VI. Kolor
  VII. Rodzaj paliwa
  VIII. Skrzynia biegów

IX. Rok produkcji
X. Silnik
XI. Konie mechaniczne
XII. Ilość miejsc
XIII. Koszt za 24 godziny
XIV. Dostępność

W kolejnych liniach dane wpisywane są w podanej wyżej kolejności.

- CustomerBase.txt
  I. ID
  II. Hasło
  III. Imię
  IV. Nazwisko
  V. Wiek
  VI. Mail
  VII. Numer telefonu

  W kolejnych liniach dane wpisywane są w podanej wyżej kolejności.

- RentBase.txt
  I. ID
  II. Hasło
  III. Imię
  IV. Nazwisko
  V. Wiek
  VI. Mail
  VII. Numer telefonu
  VIII. Ilość wypożyczonych samochodów
  IX. Tablica rejestracyjna
  X. Klasa
  XI. Typ
  XII. Marka
  XIII. Model
  XIV. Kolor
  XV. Rodzaj paliwa
  XVI. Skrzynia biegów
  XVII. Rok produkcji

XVIII. Silnik
XIX. Konie mechaniczne
XX. Ilość miejsc
XXI. Koszt za 24 godziny
XXII. 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

9. Dalszy rozwój i ulepszenia:

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