

Dokumentacja projektu

Program wypożyczalni
samochodów z podziałem na
interfejs klienta i
pracownika

Projekt został zmieszczony w serwisie
GitHub pod adresem:

[https://github.com/klonx681/car_rental
1.git](https://github.com/klonx681/car_rental.git)

Autorzy projektu:

Grzegorz Listwan

Krzysztof Pacura

Michał Lis

Autorzy dokumentacji:

Michał Lis

Grzegorz Listwan
Krzysztof Pacura

1. Wykorzystane technologie

Środowiska programistyczne:

- Visual Studio Community 2019
- Qt Creator 4.0.2

Język programowania:

- C++ z biblioteką Qt 5.7.0

System kontroli wersji:

- Git

2. Założenia projektu

Stworzenie graficznego programu dla fikcyjnej wypożyczalni samochodowej umożliwiającego klientom (jako osoba prywatna lub firma) zdalne wyszukiwanie, przeglądanie oraz rezerwowanie pojazdów z katalogu, a także z możliwością zalogowania się pracowników firmy w celach zarządzania katalogiem.

3. Cele

1. Stworzenie graficznego interfejsu z możliwością zalogowania się dla pracownika.
2. Dodanie możliwości wyszukania samochodu spośród katalogu według określonych kryteriów lub wyświetlenie jego całości
3. Implementacja funkcjonalności pozwalającej na zarezerwowanie wyszukanego pojazdu
4. Wyświetlanie w postaci tabel zawartości plików tekstowych, oraz ich modyfikacja

4. Funkcjonalność

4.1 Wyszukiwanie pojazdów z katalogu

Po otwarciu programu użytkownik ma możliwość zawęzić wyniki wyszukiwania według pięciu kategorii:

- rodzaju pojazdu
- marki producenta
- modelu
- rodzaju paliwa
- roku produkcji



Czego szukasz?

Wybierz rodzaj pojazdu:	Marka:	Model:
<input type="text" value="Osobowy"/>	<input type="text" value="Toyota"/>	<input type="text" value="Sprinter Trueno AE86"/>
Rodzaj paliwa:	Rok produkcji:	
<input type="text" value="Benzyna"/>	<input type="text" value="1986"/>	

Użytkownik może też zamiast wyszukiwania wyświetlić kompletny katalog dostępnych pojazdów przy

użyciu przycisku „Wyświetl wszystkie”.

4.2 Wybór pojazdu i składanie rezerwacji

Po wyszukaniu użytkownik zobaczy tabelę z wynikami wyszukiwania. Pod tabelą znajduje się formularz do rezerwacji pojazdu. Użytkownik ma możliwość złożenia rezerwacji jako osoba prywatna:

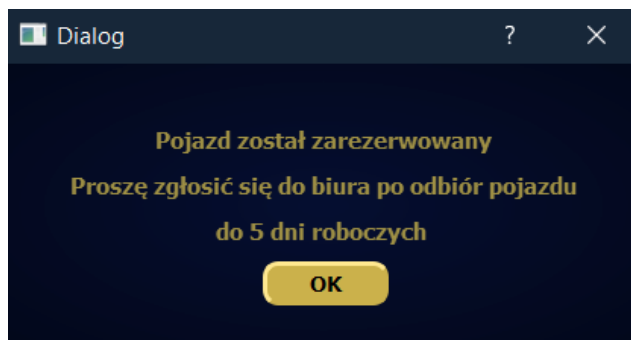
Wybierz: <input type="radio"/> Osoba prywatna <input checked="" type="radio"/> Firma Numer rejestracyjny: <input type="text" value="Numer rejestrac..."/>	Imię: <input type="text" value="Imię"/> Nazwisko: <input type="text" value="Nazwisko"/>	E-mail <input type="text" value="email@xyz.ai"/> Telefon: <input type="text" value="777222111"/>
<input type="button" value="OK"/> <input type="button" value="Cancel"/>		

lub firma:

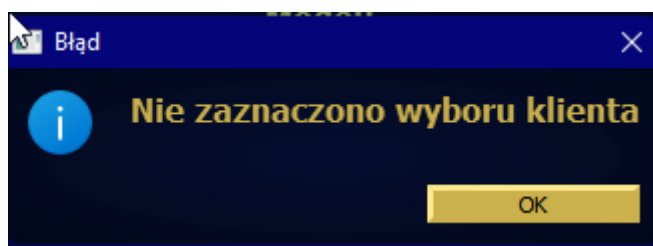
Wybierz: <input checked="" type="radio"/> Osoba prywatna <input type="radio"/> Firma Numer rejestracyjny: <input type="text" value="Numer rejestrac..."/>	Nazwa firmy <input type="text" value="Nazwa firmy"/> E-mail <input type="text" value="email@xyz.ai"/> Telefon: <input type="text" value="777222111"/>
<input type="button" value="OK"/> <input type="button" value="Cancel"/>	

Po podaniu danych osobowych/firmowych oraz numeru rejestracyjnego pożądanego samochodu kliknięcie w przycisk „OK” spowoduje pojawienie

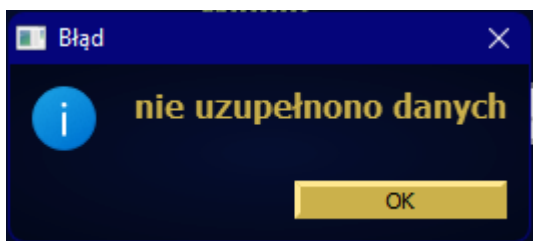
się komunikatu że wybrany samochód został pomyślnie zarezerwowany.



W przypadku braku wybrania rodzaju klienta otrzymujemy następujący komunikat.



W przypadku podania niepełnych danych otrzymujemy komunikat.



4.3 Interfejs pracownika

Po zalogowaniu się przez pracownika dostępne jest pięć opcji:

- dodaj pojazd
- usuń pojazd

- zarezerwowane
- wypożyczone
- wszystkie

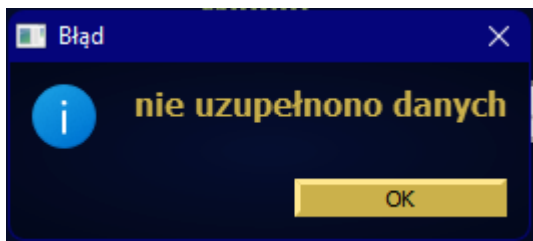
Panel „dodaj pojazd” pozwala pracownikowi dodać do katalogu nowy pojazd i zdefiniować jego parametry i cenę wypożyczenia za dzień.

The screenshot shows a form titled "dodaj pojazd" (add vehicle) with the following fields:

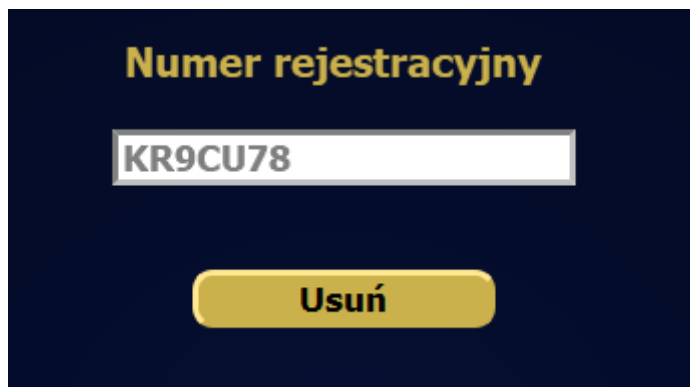
- Rodzaj pojazdu** (Vehicle Type): A dropdown menu with "Osobowy" (Personal) selected.
- Marka** (Brand): A text input field containing "Marka".
- Model**: A text input field containing "Model".
- Nr rejestracyjny** (Registration Number): A text input field containing "KR9CU78".
- Rok produkcji** (Year of Production): A text input field containing "2020".
- Nr VIN**: A text input field containing "VIN".
- Rodzaj paliwa** (Fuel Type): A dropdown menu with "Benzyna" (Petrol) selected.
- Liczba drzwi** (Number of Doors): A dropdown menu.
- Liczba miejsc** (Number of Seats): A text input field containing "Liczba miejsc".
- Cena za dzień** (Daily Price): A text input field containing "Cena".

At the bottom center of the form is a yellow button labeled "Dodaj" (Add).

W przypadku nie uzupełnienia wszystkich danych ukazuje się następujące okno.



Panel „usuń pojazd” służy do usuwania pojazdu z katalogu przed podanie odpowiedniego numeru rejestracyjnego.



Panel „usuń pojazd” służy do usuwania pojazdu z katalogu przed podanie odpowiedniego numeru rejestracyjnego.

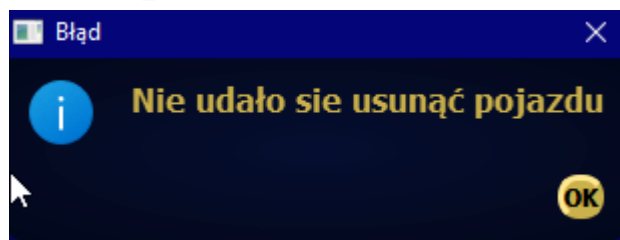
Numer rejestracyjny

KR9CU78

Usuń

Jeśli podamy błędny numer

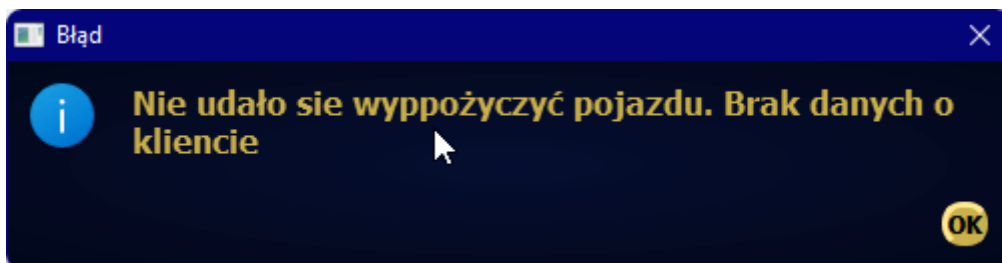
rejestracyjny



Panel „zarezerwowane” pozwala zobaczyć listę wszystkich złożonych rezerwacji oraz sfinalizować proces wypożyczenia pojazdu.

The screenshot shows a web interface for vehicle rental. At the top, there is a large empty box for a list of reservations. Below it, a dark blue header contains the text "Wybierz:" followed by two radio buttons: "osoba prywatna" (selected) and "firma". To the right is a label "Nr rejestracyjny" above a text input field. Further right is a yellow button labeled "Wypożycz". Below the header is a form with two rows of five input fields each. The first row has labels: "Imię", "Nazwisko", "Email", "Telefon", and "PESEL". The second row has labels: "Nr dowodu os.", "Ulica", "Nr domu", "Kod pocztowy", and "Miasto".

Jeśli nie podamy wymaganych danych



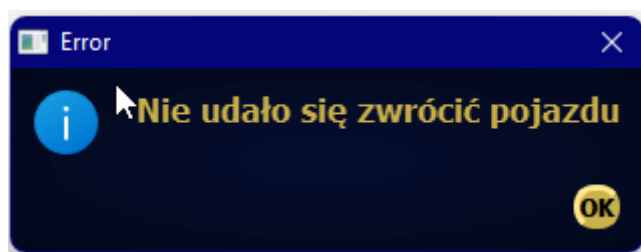
Panel „wypożyczone” pokazuje wszystkie obecnie wypożyczone pojazdy oraz daje opcje zwrócenia samochodu z powrotem przed podanie numeru rejestracyjnego pojazdu do zwrotu.

Rodzaj pojazdu	Marka	Model	Rodzaj paliwa	VIN	Nr rejestracyjny	Liczba miejsc	Rok produkcji
----------------	-------	-------	---------------	-----	------------------	---------------	---------------

Numer rejestracyjny

Zwróć pojazd

Po podaniu błędnego numeru rejestracyjnego



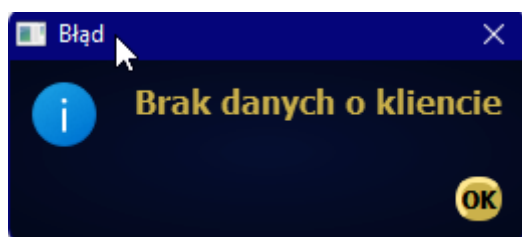
Ostatni panel „wszystkie” wyświetla wszystkie pojazdy znajdujące się w katalogu firmy oraz tak jak panel ma możliwość wypożyczania pojazdów.

	Rodzaj pojazdu	Marka	Model	Rodzaj paliwa	VIN	Nr rejestracyjny	Liczba miejsc	Rok ^
1	Osobowy	Kia	Niro	Hybryda	WF0HGHWPJH...	KR4UK09	5	2020
2	Osobowy	Kia	Sportage	Diesel	WF0HGHYJH7...	KR2UK11	5	2021
3	Osobowy	Toyota	Yaris	Hybryda	PUGHGHYJH7...	KR6UK14	5	2019
4	Dostawczy	Ford	Tranzit	Benzyna	PUGHGHYKK7...	KR6UC14	5	2020
5	Dostawczy	Fiat	Ducato	Benzyna	PUGHGUCUGK7...	KR6RJ45	5	2022
6	Dostawczy	Fiat	Ducato	Diesel	PUGH76CUGK7...	KR7RJ08	5	2022

Wybierz: o osoba prywatna o firma **Nr rejestracyjny** **Wypożycz**

Imię	Nazwisko	Email	Telefon	PESEL
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Nr dowodu os.	Ulica	Nr domu	Kod pocztowy	Miasto
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

W ostatnim panelu w przypadku podania niepełnych danych otrzymujemy analogiczną informację jak w zakładce „zarezerwowane”



5. Implementacja problemu w kodzie C++

5.1 Klasa MainWindow z głównym oknem aplikacji.

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QMessageBox>
QFile raportFile("../baza\\raport.csv");
MainWindow::MainWindow(QWidget* parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}
```

Linia 1-3 Złączanie bibliotek i plików nagłówkowych klas.

Linia 4 deklarowanie pliku z raportem

Linia 5- 14 deklaracja konstruktora klasy

```

18 ▼ MainWindow::~MainWindow()
19 {
20     //po zamknięciu aplikacji generownie raportu
21     if (raportFile.open(QIODevice::Append)) {
22         QTextStream stream(&raportFile);
23         QString text ="zalogowano ";
24         text+= Staf::currentStaf.getName();
25         text+= " dnia ";
26         text+= Staf::currentStaf.getStart().toString();
27         text+= "\n dodano pojazdy:\n";
28     for (auto element:Car::addedCars)
29     {
30         if(element.getBodyType().compare(""))
31             text+= element.toString()+"\n";
32     }
33     for (auto element:Motorcycle::addedMotors)
34     {
35         if(element.getBodyType().compare(""))
36             text+= element.toString()+"\n";
37     }
38     text+="\n usunięto pojazdy \n";
39     for (auto element:Car::removedCars)
40     {
41         if(element.getBodyType().compare(""))
42             text+= element.toString()+"\n";
43     }
44     for (auto element:Motorcycle::removedMotors)
45     {
46         if(element.getBodyType().compare(""))
47             text+= element.toString()+"\n";
48     }
49 }
50
51
52

```

```

52     }
53     text+="\n wypożyczono pojazdy \n";
54     for (auto element:Car::rentedCars)
55     {
56         if(element.getBodyType().compare(""))
57             text+= element.toString()+"\n";
58     }
59     for (auto element:Motorcycle::rentedMotors)
60     {
61         if(element.getBodyType().compare(""))
62             text+= element.toString()+"\n";
63     }
64     text+="\n zwrócono pojazdy \n";
65     for (auto element:Car::returnedCars)
66     {
67         if(element.getBodyType().compare(""))
68             text+= element.toString()+"\n";
69     }
70     for (auto element:Motorcycle::returnedMotors)
71     {
72         if(element.getBodyType().compare(""))
73             text+= element.toString()+"\n";
74     }
75     text+="\n wylogowano "+ Staf::currentStaf.toString();
76     stream<< text;
77     raportFile.close();
78     }
79     delete ui;
80 }
81
82
83
84
85
86

```

Linie od 18 – 85 deklaracja dekonstruktora klasy.
W dekonstruktorze zawarte są instrukcje generujące plik raportu

```

88 void MainWindow::on_actionZamknij_triggered()
89 {
90     QApplication::quit();
91 }
92
93
94 void MainWindow::on_actionO_bibliotece_triggered()
95 {
96     QApplication::aboutQt();
97 }
98
99
100 void MainWindow::on_actionO_programie_triggered()
101 {
102     QMessageBox::information(this, "O programie", "Program do obsługi wypożyczalni pojazdów.\n"
103         "Wykonany na zaliczenie przedmiotu Podstawy programowania.\n\n"
104         "Autorzy:\n"
105         "Michał Lis\n"
106         "Grzegorz Listwan\n"
107         "Krzysztof Pacura\n\n"
108         "ver. 1.00\n"
109         "Copyright © 2022\n");
110 }
111
112
113 void MainWindow::on_login_clicked()
114 {
115     Login login;
116     login.setModal(true);
117     login.exec();
118 }
119
120

```

Linie 88-91 slot przycisku zamknij w górnym menu aplikacji. Wywołanie funkcji zamykającej program.

Linie 94 - 97 slot obsługujący opcje „O bibliotece” z menu aplikacji. Wywołanie funkcji aboutQt która wyświetla informacje o użytej bibliotece.

Linie 100-111 slot obsługujący opcje „O programie” z menu aplikacji. Zawiera wywołanie QMessageBox z informacjami o autorach i nazwie programu.

Linie 113 – 118 obsługa przycisku „zaloguj” wywołanie okna klasy Login z panelem do logowania

```
121 void MainWindow::on_pushButton_clicked()
122 {
123     QMessageBox::information(this, "in progres", "work in progres");
124 }
125 void MainWindow::on_pushButton_2_clicked()
126 {
127     Booking booking;
128     booking.setModal(true);
129     booking.exec();
130 }
```

Linie 121 – 124 obsługa przycisku „Szukaj” – funkcja w trakcie realizacji

Linie 125 – 130 obsługa przycisku wyświetl wszystkie. Wywołanie okna z klasy Booking

5.2 Klasa Car

```
1  #include "car.h"
2  //inicjalizowanie statycznych list
3  QList<Car> Car::addedCars=QList<Car>();
4  QList<Car> Car::removedCars=QList<Car>();
5  QList<Car> Car::rentedCars=QList<Car>();
6  QList<Car> Car::returnedCars=QList<Car>();
7
8  //przeciążanie konstruktorów
9  Car::Car()
10 {
11     this->flag = true;
12     this->returned= true;
13 }
```

Linie 3 – 6 inicjalizowanie list statycznych potrzebnych do generowania raportu

Linie 9 – 13 podstawowy konstruktor

```
14  Car::Car(QString bodyType, QString brand, QString model, QString fuel, QSt
15  {
16      this->doors = doors;
17      this->flag = false;
18  }
19  Car::Car(QString bodyType, QString brand, QString model, QString fuel, QSt
20      QString lastName,
21      QString email,
22      QString phone
23  ) : Vehicle(bodyType, brand, model, fuel, vin, regNum, sites, year, price,
24      lastName,
25      email,
26  {
27      this->doors = doors;
28      this->flag = false;
29      this->returned= false;
30  }
31  }
32
33  Car::Car(QString bodyType, QString brand, QString model, QString fuel, QSt
34      QString email,
35      QString phone ) :Vehicle(bodyType, brand, model, fuel, vin, regNum, si
36      email,
37      phone
38  {
39      this->doors = doors;
40      this->flag = false;
41      this->returned= false;
42  }
43  }
44
```

```

45 Car::Car(QString bodyType, QString brand, QString model, QString fuel, QString vin, QString regNum, QS
46     QString lastName,
47     QString email,
48     QString phone,
49     QString pesel,
50     QString IDnumber,
51     QString street,
52     QString homeNumber,
53     QString postCode,
54     QString city) : Vehicle(bodyType, brand, model, fuel, vin, regNum, sites, year, price, status, fir
55         lastName,
56         email,
57         phone,
58         pesel,
59         IDnumber,
60         street,
61         homeNumber,
62         postCode,
63     city)
64 {
65     this->doors = doors;
66     this->returned= false;
67 }
68 Car::Car(QString bodyType, QString brand, QString model, QString fuel, QString vin, QString regNum, QS
69     QString email,
70     QString phone,
71     QString NIP,
72     QString street,
73     QString localNumber,
74     QString postCode,
75     QString city) :Vehicle(bodyType, brand, model, fuel, vin, regNum, sites, year, price, status, name
76     email,
77     phone,
78     NIP,
79     street.

```

```

68 Car::Car(QString bodyType, QString brand, QString model, QString fuel, QString vin, QString regNum, QString
69     QString email,
70     QString phone,
71     QString NIP,
72     QString street,
73     QString localNumber,
74     QString postCode,
75     QString city) :Vehicle(bodyType, brand, model, fuel, vin, regNum, sites, year, price, status, name,
76     email,
77     phone,
78     NIP,
79     street,
80     localNumber,
81     postCode,
82     city)
83 {
84     this->doors = doors;
85     this->returned= false;
86 }

```

Linie 14 - 86 deklaracje przeciążonych konstruktorów i przypisywanie wartości do pull obiektów

```
87  ▼ QString Car::getBodyType()
88  {
89      return bodyType;
90  }
91  ▼ QString Car::getBrand()
92  {
93      return brand;
94  }
95  ▼ QString Car::getModel()
96  {
97      return model;
98  }
99  ▼ QString Car::getFuel()
100 {
101     return fuel;
102 }
103 ▼ QString Car::getVin()
104 {
105     return vin;
106 }
107 ▼ QString Car::getRegNum()
108 {
109     return regNum;
110 }
111 ▼ QString Car::getSites()
112 {
113     return sites;
114 }
115 ▼ QString Car::getYear()
116 {
117     return year;
118 }
119 ▼ QString Car::getPrice()
120 {
121     return price;
122 }
```

```

123  ✓ QString Car::getStatus()
124  {
125      return status;
126  }
127  ✓ QString Car::getDoors()
128  {
129      return doors;
130  }
131  ✓ bool Car::getFlag() {
132      return flag;
133  }
134  ✓ QString Car::getFirstName() {
135      return firstName;
136  }
137  ✓ QString Car::getLastName() {
138      return lastName;
139  }
140  ✓ QString Car::getEmail() {
141      return email;
142  }
143  ✓ QString Car::getPhone() {
144      return phone;
145  }
146  ✓ QString Car::getPesel() {
147      return pesel;
148  }
149  ✓ QString Car::getIDnumber() {
150      return IDnumber;
151  }
152  ✓ QString Car::getStreet() {
153      return street;
154  }
155  ✓ QString Car::getHomeNumber() {
156      return homeNumber;
157  }

158  ✓ QString Car::getPostCode() {
159      return postCode;
160  }
161  ✓ QString Car::getCity() {
162      return city;
163  }
164  ✓ QString Car::getName() {
165      return name;
166  }
167  ✓ QString Car::getNIP() {
168      return NIP;
169  }
170  ✓ QString Car::getLocalNumber() {
171      return localNumber;
172  }
173  ✓ bool Car::getReturned() {
174      return returned;
175  }

```

Linie 87 – 175 tworzenie getterów

```

176 //przeciążanie funkcji toString
177 QString Car::toString(bool isReturn) {
178     QString info = this->bodyType + ";" + this->brand + ";" + this->model + ";" + this->fuel + ";" + this->
179     this->regNum + ";" + this->sites + ";" + this->year + ";" + this->doors + ";" + this->price + ";" +
180     this->lastName+";"+
181     this->name+";"+
182     this->email+";"+
183     this->phone+";"+
184     this->pesel+";"+
185     this->IDnumber+";"+
186     this->NIP+";"+
187     this->street+";"+
188     this->homeNumber+";"+
189     this->localNumber+";"+
190     this->postCode+";"+
191     this->city+";";
192     return info;
193 }
194 QString Car::toString() {
195     QString info = this->bodyType + ";" + this->brand + ";" + this->model + ";" + this->fuel + ";" + this->
196     this->regNum + ";" + this->sites + ";" + this->year + ";" + this->doors + ";" + this->price + ";" +
197     return info;
198 }
199 Car::~Car() {
200 }
201 }
202

```

Linie 177 - 198 przeciążone funkcje zwracające QStringa

Linia 199-201 destruktork

5.3 Klasa Client

```

1 #include "client.h"
2 #include <iostream>
3 using namespace std;
4 Client::Client()
5 {
6 }
7 }
8 Client::Client(QString email, QString phone)
9 {
10     this->email = email;
11     this->phone = phone;
12 }
13 Client::Client(QString email, QString phone, QString street, QString post_code, QString city)
14 {
15     this->email = email;
16     this->phone = phone;
17     this->street = street;
18     this->postCode = post_code;
19     this->city = city;
20 }
21 }
22 Client::~Client() {
23 }
24 }
25

```

Klasa Client zawiera 3 konstruktory oraz destruktork jest to klasa po której dziedziczą klasy Company i Person

5.4 Klasa Booking

```
#include "booking.h"
#include "ui_booking.h"
#include <QMessageBox>
// definicja plików bazy danych
QFile vehiclesFile("../baza/vehicles.csv");
QFile bookFile("../baza/bookvehicle.csv");
//definicja list
QList<Car> clientCars;
QList<Motorcycle> clientMotors;
//definicja zmiennych pomocniczych
Car bookCar;
Motorcycle bookMotor;
Person person;
Company company;
//funkcja rezerwująca pojazdy
bool Booking::bookVehicle(QString regNum) {
    bookCar = Car();
    bookMotor = Motorcycle();
    int index = 0;
    bool flag = false;
    //inkrementacja po elementach listy
    for (auto element : clientCars) {

        if (!element.getRegNum().compare(regNum)) { //porównanie dwóch
stringów
            flag = true;
            break;
        }
        else {
            index++;
        }
    }
    if (flag) { //zwracanie zarezerwowanego pojazdu i usuwanie go z listy
        bookCar = clientCars.at(index);
        clientCars.removeAt(index);
    }

    else {
        index = 0;
        for (auto element : clientMotors) { //powtarzamy to co wcześniej
tylko dla motorów

            if (!element.getRegNum().compare(regNum)) {
                flag = true;
                break;
            }
            else {
                index++;
            }
        }

        if (flag) {
            bookMotor = clientMotors.at(index);
            clientMotors.removeAt(index);
        }
    }
}
```

```

        return flag;
    }
    // funkcja odczytująca dane z pliku
    bool Booking::readsFile() {
        // czyszczenie list
        clientCars.clear();
        clientMotors.clear();
        //sprawdzanie czy plik istnieje
        if (vehiclesFile.exists())
        {
            //otwarcie pliku moethodą czytaj wpis
            if (vehiclesFile.open(QIODevice::ReadWrite |
QIODevice::Text)) {
                QTextStream in(&vehiclesFile);
                //wsprawdzam czy plik się nie skończył
                while (!in.atEnd()) {
                    QString string = in.readLine(); //odczytujemy pojedyncza
linie
                    QStringList list = string.split(';',
QString::SkipEmptyParts); //rozdzielamy wczytaną linie po ;
                    if (list.length() == 11) {
                        //tworzymy nowy samochód
                        Car car = Car(list.at(0), list.at(1),
list.at(2), list.at(3), list.at(4), list.at(5), list.at(6), list.at(7),
list.at(8), list.at(9), list.at(10));
                        clientCars.append(car); //dodajemy do listy
                    }
                    else if (list.length() == 10) {
                        Motorcycle motor =
Motorcycle(list.at(0), list.at(1), list.at(2), list.at(3), list.at(4),
list.at(5), list.at(6), list.at(7), list.at(8), list.at(9));
                        clientMotors.append(motor);
                    }
                }
                vehiclesFile.close(); //zamknięcie pliku
                return true;
            }
            else {
                return false;
            }
        }
        else
        {
            return false;
        }
    }
    //wyświetlanie wszystkich pojazdów
    void Booking::displaysVehicles() {
        int rows = clientCars.length() + clientMotors.length(); //zliczamy
liczbe wierszy
        ui->bookVehicles->setRowCount(rows); //ustawiamy liczbe wierszy w
tabeli UI
        int index = 0;
        for (auto element : clientCars) {
            QTableWidgetItem* item; //tworzymy wskaźnik na item w tabeli UI
            for (int j = 0; j < ui->bookVehicles->columnCount(); j++) {
                item = new QTableWidgetItem; //ustawiamy wskaźnik na nowy item
w tabeli UI
                //wyświetlamy dane w odpowiedniej kolumnie
                switch (j) {
                    case 0:

```



```

        item->setText(element.getBodyType());
        break;
    case 1:
        item->setText(element.getBrand());
        break;
    case 2:
        item->setText(element.getModel());
        break;
    case 3:
        item->setText(element.getFuel());
        break;
    case 4:
        item->setText(element.getVin());
        break;
    case 5:
        item->setText(element.getRegNum());
        break;
    case 6:
        item->setText(element.getSites());
        break;
    case 7:
        item->setText(element.getYear());
        break;
    case 8:
        item->setText(element.getDoors());
        break;
    case 9:
        item->setText(element.getPrice());
        break;
    case 10:
        item->setText("Dostępny");
        break;
    default:
        break;
}

    ui->bookVehicles->setItem(index, j, item); // ustawiamy item w
konkretnym miejscu tabeli UI

    }
    index++;
}
// -----
-----
for (auto element : clientMotors) {
    QTableWidgetItem* item;
    for (int j = 0; j < ui->bookVehicles->columnCount(); j++) {
        item = new QTableWidgetItem;
        switch (j) {
            case 0:
                item->setText(element.getBodyType());
                break;
            case 1:
                item->setText(element.getBrand());
                break;
            case 2:
                item->setText(element.getModel());
                break;
            case 3:
                item->setText(element.getFuel());

```

```

        break;
    case 4:
        item->setText(element.getVin());
        break;
    case 5:
        item->setText(element.getRegNum());
        break;
    case 6:
        item->setText(element.getSites());
        break;
    case 7:
        item->setText(element.getYear());
        break;
    case 8:
        item->setText("");
        break;
    case 9:
        item->setText(element.getPrice());
        break;
    case 10:
        item->setText("Dostępny");
        break;
    default:
        break;
}

ui->bookVehicles->setItem(index, j, item);

}
index++;
}

}

Booking::Booking(QWidget* parent) : //deklaracja konstruktora klasy Booking
    QDialog(parent),
    ui(new Ui::Booking)
{
    ui->setupUi(this);
    ui->reg_num->setPlaceholderText("Numer rejestracyjny");
    ui->name->setPlaceholderText("Imię");
    ui->last_name->setPlaceholderText("Nazwisko");
    ui->email->setPlaceholderText("email@xyz.ai");
    ui->phone->setPlaceholderText("777222111");
    ui->company_email->setPlaceholderText("email@xyz.ai");
    ui->company_phone->setPlaceholderText("777222111");
    ui->company_name->setPlaceholderText("Nazwa firmy");
    ui->person_box->setVisible(false);
    ui->company_box->setVisible(false);
    ui->bookVehicles->setColumnCount(11); //ustawiamy ilość kolumn w tabeli
    UI
    QStringList titles;
    titles << "Rodzaj pojazdu" << "Marka" << "Model" << "Rodzaj paliwa"
    << " VIN" << " Nr rejestracyjny " << "Liczba miejsc" << "Rok produkcji" <<
    "Liczba drzwi" << "Cena " << "Status";
    ui->bookVehicles->setHorizontalHeaderLabels(titles); // nadajemy
    etykiety kolumną
    if (readsFile()) {
        displaysVehicles();
    }
}

```

```

        else {
            QMessageBox::information(this, "Error", "Błąd odczytu danych z
pliku");
        }
    }
Booking::~Booking()
{
    delete ui;
}

void Booking::on_buttonBox_accepted()
{
    bool dataFlag=false;
    person=Person();
    company=Company();
    QString regNum = ui->reg_num->text();
    if(ui->person->isChecked()) // sprawdzamy czy radiobutton został
ustawiony
    {
        //pobiermy dane z UI
        QString firstName= ui->name->text();
        QString lastName= ui->last_name->text();
        QString email= ui->email->text();
        QString phone= ui->phone->text();

        if((firstName.compare("")&&lastName.compare("")&&email.compare("")&&phone.co
mpare("")))
        {
            dataFlag=true;
            person=Person(firstName,lastName,email,phone);
        }else
            QMessageBox::information(this, "Błąd", "nie uzupełniono danych");
    } else if(ui->company->isChecked())
    {
        QString name= ui->company_name->text();
        QString email= ui->company_email->text();
        QString phone= ui->company_phone->text();
        if((name.compare("")&&email.compare("")&&phone.compare("")))
        {
            dataFlag=true;
            company=Company(name,email,phone);
        }else
            QMessageBox::information(this, "Błąd", "nie uzupełniono danych");
    } else
    {
        QMessageBox::information(this, "Błąd", "Nie zaznaczono wyboru
klienta");
    }
    if (dataFlag) {
        if (bookVehicle(regNum))
        {
            if (bookFile.open(QIODevice::Append)) { // otwarcie pliku metoda
dopisz na końcu
                QTextStream stream(&bookFile);
                QString text = "";
                if(!bookCar.getFlag())
                    text+=bookCar.toString();
            }
        }
    }
}

```

```

        if(!bookMotor.getFlag())
            text+=bookMotor.toString();
        if (!person.getFlag())
            text+=person.toString()+"\n";
        if (!company.getFlag())
            text+=company.toString()+"\n";
        stream<< text; // dopisanie danych do pliku
        bookFile.close();

    }
    if (vehiclesFile.open(QIODevice::WriteOnly)) { // otwarcie pliku
metoda nadpisz
        QTextStream stream(&vehiclesFile);
        for (auto element : clientCars) {
            stream << element.toString() + "\n";
        }
        for (auto element : clientMotors) {
            stream << element.toString() + "\n";
        }

        vehiclesFile.close();

    }
    if (readsFile()) {
        displaysVehicles();
        //stworzenie nowego okna i jego wywołanie
        Zarezerwowany book;
        book.setModal(true);
        book.exec();
    }
    else {
        QMessageBox::information(this, "Error", "Błąd odczytu danych z
pliku");
    }
    } else {
        QMessageBox::information(this, "Błąd", "Nie udało zarezerwować
pojazdu");
    }
}
}

void Booking::on_person_clicked()
{
    ui->person_box->setVisible(true);
    ui->company_box->setVisible(false);
}

void Booking::on_company_clicked()
{
    ui->company_box->setVisible(true);
    ui->person_box->setVisible(false);
}

```

5.5 Klasa Company

```
#include "company.h"

Company::Company()
{
    this->flag = true;
}

Company::Company(QString name, QString email, QString phone) :Client(email,
phone) {
    this->name = name;
    this->flag = false;
}

Company::Company(QString name, QString email, QString phone, QString nip,
QString street, QString locNum, QString post_code, QString city) :
Client(email, phone, street, post_code, city)
{
    this->name = name;
    this->nip = nip;
    this->locNum = locNum;
    this->flag = false;
}
}
```

Powyżej znajdują się przeciążone konstruktory

```
QString Company::getName()
{
    return name;
}

QString Company::getNip()
{
    return nip;
}

QString Company::getLocNum() {
    return locNum;
}

QString Company::getEmail()
{
    return email;
}

QString Company::getPhone()
{
    return phone;
}

QString Company::getStreet() {
    return street;
}

QString Company::getPostCode() {
    return postCode;
}

QString Company::getCity() {
    return city;
}
}
```

Powyższe linie to deklarowanie geterów.

```

QString Company::toString() {
    return this->name + ";" + this->email + ";" + this->phone + ";" +
    this->nip + ";" + this->street + ";" + this->locNum + ";" + this->postCode
    + ";" + this->city + ";";
}

```

Funkcja zwracająca QStringa

```

bool Company::getFlag() {
    return flag;
}

```

```

Company::~~Company() {}

```

5.6 Klasa Login

```

#include "login.h"
#include "ui_login.h"

```

```

using namespace std;
QList<Staf> stafs;

```

```

Login::Login(QWidget* parent) :
    QDialog(parent),
    ui(new Ui::Login)
{
    ui->setupUi(this);
    ui->login->setPlaceholderText("Login");
    ui->password->setPlaceholderText("Hasło");
    ui->password->setEchoMode(QLineEdit::Password);
}

```

Pobieranie danych od użytkownika

```

QFile file;
file.setFileName("../baza/workers.csv");
if (file.exists()) {
    if (file.open(QIODevice::ReadOnly | QIODevice::Text)) {
        QTextStream in(&file);
        while (!in.atEnd()) {
            QString string = in.readLine();
            QStringList list = string.split(';');
            QString::SkipEmptyParts);
            Staf staf = Staf(list.at(0), list.at(1));
            stafs.append(staf);
        }
        file.close();
    }
}
else

```

```

        {
            QMessageBox::information(this, "Błąd", "Błąd");
        }
    }
}

```

Odczyt danych z pliku workers.csv

```

Login::~Login()
{
    delete ui;
}

void Login::on_buttonBox_accepted()
{
    bool flag = false;
    QString name = ui->login->text();
    QString password = ui->password->text();
    for (auto element : stafs) {

        if (!(element.getName().compare(name) &&
element.getPassword().compare(password))) {
            Staf::currentStaf=element;
            flag = true;
            break;
        }
        else {
            flag = false;
        }
    }
}

```

Porównywanie czy podane dane są poprawne

```

    if (flag) {
        Worker worker;
        worker.setModal(true);
        worker.exec();
    }
    else {
        QMessageBox::information(this, "Błąd", "Błędny login lub
hasło");
    }
}

```

Wywołanie okna z klasy Worker

5.7 Klasa Main

```

#include "mainwindow.h"

#include <QApplication>
#include <QLocale>
#include <QTranslator>

int main(int argc, char* argv[])
{
    QApplication a(argc, argv); //tworzenie okna aplikacji
}

```

```

        QTranslator translator;
        const QStringList uiLanguages = QLocale::system().uiLanguages();
        for (const QString& locale : uiLanguages) {
            const QString baseName = "Car_rental_" +
                QLocale(locale).name();
            if (translator.load(":/i18n/" + baseName)) {
                a.installTranslator(&translator);
                break;
            }
        }
        MainWindow w;
        w.show();
        return a.exec();
    }

```

Klasa odpowiedzialna za postawienie gotowej aplikacji tu buduje się aplikacja

5.8 Klasy Motorcycle

```

#include "motorcycle.h"
QList<Motorcycle> Motorcycle::addedMotors=QList<Motorcycle>();
QList<Motorcycle> Motorcycle::removedMotors=QList<Motorcycle>();
QList<Motorcycle> Motorcycle::rentedMotors=QList<Motorcycle>();
QList<Motorcycle> Motorcycle::returnedMotors=QList<Motorcycle>();

```

Statyczny QListy do generacji raportu

```

Motorcycle::Motorcycle()
{
    this->flag = true;
    this->returned= true;
}
Motorcycle::Motorcycle(QString bodyType, QString brand, QString model,
    QString fuel, QString vin, QString regNum, QString sites, QString year,
    QString price, QString status) :Vehicle(bodyType, brand, model, fuel, vin,
    regNum, sites, year, price, status)
{
    this->flag = false;
}
Motorcycle::Motorcycle(QString bodyType, QString brand, QString model,
    QString fuel, QString vin, QString regNum, QString sites, QString year,
    QString price, QString status, QString firstName,
    QString lastName,
    QString email,
    QString phone) : Vehicle(bodyType, brand, model, fuel, vin, regNum,
    sites, year, price, status, firstName,
    lastName,
    email,
    phone
    )
{

```



```

this->returned= false;
    this->flag = false;
}
Motorcycle::Motorcycle(QString bodyType, QString brand, QString model,
    QString fuel, QString vin, QString regNum, QString sites, QString year,
    QString price, QString status, QString name,
        QString email,
        QString phone) :Vehicle(bodyType, brand, model, fuel, vin, regNum,
sites, year, price, status, name,
        email,
        phone
    )
{
this->returned= false;
    this->flag = false;
}
Motorcycle::Motorcycle(QString bodyType, QString brand, QString model,
    QString fuel, QString vin, QString regNum, QString sites, QString year,
    QString price, QString status, QString firstName,
        QString lastName,
        QString email,
        QString phone,
        QString pesel,
        QString IDnumber,
        QString street,
        QString homeNumber,
        QString postCode,
        QString city) : Vehicle(bodyType, brand, model, fuel, vin, regNum,
sites, year, price, status, firstName,
        lastName,
        email,
        phone,
        pesel,
        IDnumber,
        street,
        homeNumber,
        postCode,
        city)
{
this->returned= false;
}
Motorcycle::Motorcycle(QString bodyType, QString brand, QString model,
    QString fuel, QString vin, QString regNum, QString sites, QString year,
    QString price, QString status, QString name,
        QString email,
        QString phone,
        QString NIP,
        QString street,
        QString localNumber,
        QString postCode,
        QString city) :Vehicle(bodyType, brand, model, fuel, vin, regNum,
sites, year, price, status, name,
        email,
        phone,
        NIP,
        street,
        localNumber,
        postCode,
        city)
{
this->returned= false;
}

```

```
}
```

Przeciążone konstruktory z przypisaniem danych

```
QString Motorcycle::getBodyType()
{
    return bodyType;
}
QString Motorcycle::getBrand()
{
    return brand;
}
QString Motorcycle::getModel()
{
    return model;
}
QString Motorcycle::getFuel()
{
    return fuel;
}
QString Motorcycle::getVin()
{
    return vin;
}
QString Motorcycle::getRegNum()
{
    return regNum;
}
QString Motorcycle::getSites()
{
    return sites;
}
QString Motorcycle::getYear()
{
    return year;
}
QString Motorcycle::getPrice()
{
    return price;
}
QString Motorcycle::getStatus()
{
    return status;
}
bool Motorcycle::getFlag() {
    return flag;
}
QString Motorcycle::getFirstName() {
    return firstName;
}
QString Motorcycle::getLastName() {
    return lastName;
}
QString Motorcycle::getEmail() {
    return email;
}
QString Motorcycle::getPhone() {
    return phone;
}
QString Motorcycle::getPesel() {
    return pesel;
}
```

```

}
QString Motorcycle::getIDnumber() {
    return IDnumber;
}
QString Motorcycle::getStreet() {
    return street;
}
QString Motorcycle::getHomeNumber() {
    return homeNumber;
}
QString Motorcycle::getPostCode() {
    return postCode;
}
QString Motorcycle::getCity() {
    return city;
}
QString Motorcycle::getName() {
    return name;
}
QString Motorcycle::getNIP() {
    return NIP;
}
QString Motorcycle::getLocalNumber() {
    return localNumber;
}
bool Motorcycle::getReturned() {
    return returned;
}

```

Funkcje getery do zwracania wartości pól obiektu

```

QString Motorcycle::toString(bool isReturn) {
    QString info = this->bodyType + ";" + this->brand + ";" + this->model +
    ";" + this->fuel + ";" + this->vin + ";" +
        this->regNum + ";" + this->sites + ";" + this->year + ";" + this-
    >price + ";" + this->status + ";" + this->firstName + ";" +
        this->lastName + ";" +
        this->name + ";" +
        this->email + ";" +
        this->phone + ";" +
        this->pesel + ";" +
        this->IDnumber + ";" +
        this->NIP + ";" +
        this->street + ";" +
        this->homeNumber + ";" +
        this->localNumber + ";" +
        this->postCode + ";" +
        this->city + ";" +
        return info;
}
QString Motorcycle::toString() {
    QString info = this->bodyType + ";" + this->brand + ";" + this-
    >model + ";" + this->fuel + ";" + this->vin + ";" +
        this->regNum + ";" + this->sites + ";" + this->year + ";" +
    this->price + ";" + this->status + ";" +
        return info;
}

```

Przeciążone funkcje toString() zwracające QStringa

```
Motorcycle::~Motorcycle()  
{  
  
}
```

Destruktor

5.9 Klasa Person

Klasa dziedzicząca po klasie client

```
#include "person.h"  
  
Person::Person()  
{  
    this->flag = true;  
}  
Person::Person(QString name, QString surname, QString email, QString phone)  
:Client(email, phone) {  
    this->firstName = name;  
    this->surname = surname;  
    this->flag= false;  
}  
Person::Person(QString name, QString lastName, QString email, QString  
phone, QString pesel, QString idNumber, QString street, QString homeNumber,  
QString postCode, QString city)  
: Client(email, phone, street, postCode, city)  
{  
    this->firstName = name;  
    this->surname = lastName;  
    this->idNumber = idNumber;  
    this->pesel = pesel;  
    this->homeNumber = homeNumber;  
    this->flag = false;  
}
```

Przeciążone konstruktory z przypisaniem danych do pól obiektu

```

QString Person::getName()
{
    return firstName;
}
QString Person::getSurname()
{
    return surname;
}
QString Person::getIdNumber()
{
    return idNumber;
}
QString Person::getPesel()
{
    return pesel;
}
QString Person::getHomeNumber()
{
    return homeNumber;
}
QString Person::getLocNumber()
{
    return locNumber;
}

QString Person::getEmail()
{
    return email;
}

QString Person::getPhone()
{
    return phone;
}

QString Person::getStreet() {
    return street;
}
QString Person::getPostCode() {
    return postCode;
}
QString Person::getCity() {
    return city;
}

```

Funkcje getery

```

QString Person::toString() {
    return this->firstName + ";" + this->surname + ";" + this->email + ";"
+ this->phone + ";" + this->pesel + ";" + this->idNumber + ";" + this-
>street + ";" + this->homeNumber + ";" + this->postCode + ";" + this->city
+ ";";
}
bool Person::getFlag() {
    return flag;
}

```

Funkcje toString() do zwracania QStringa oraz funkcja getFlag do zwracania flagi.

```
Person::~~Person() {
}
```

Destruktor

5.10 Klasa Staf

```
#include "staf.h"
Staf Staf::currentStaf=Staf();
Staf::Staf()
{
}
Staf::Staf(QString name, QString password) {
    this->name = name;
    this->password = password;
    this->start = QDateTime::currentDateTime();
}
```

Konstruktor z przypisaniem danych do pól oraz pobranie aktualnego czasu logowania.

```
QString Staf::getName() {
    return name;
}
QString Staf::getPassword() {
    return password;
}
QDateTime Staf::getStart(){
    return start;
}

QString Staf::toString(){
    //zwracanie aktualnego czasu
    QDateTime now=QDateTime::currentDateTime();
    return " "+ this-> name+" dnia "+ now.toString("dddd, d MMMM yy
hh:mm:ss") +"\n";
}
```

Funkcja toString() zwracająca QStringa z danymi o pracowniku oraz zwracająca czas wylogowania.

```
Staf::~~Staf() {  
}
```

5.11 Klasa Vehicle

Klasa Vehicle nadrzędna klasa po której dziedziczą klasy Car i Motorcycle.

5.12 Klasa Worker

Klasa z obsługą wszystkich funkcji pracownika funkcji pracownika.

5.12 Klasa Zarezerwowany

```
#include "zarezerwowany.h"  
#include "ui_zarezerwowany.h"  
  
Zarezerwowany::Zarezerwowany(QWidget* parent) :  
    QDialog(parent),  
    ui(new Ui::Zarezerwowany)  
{  
    ui->setupUi(this);  
}  
  
Zarezerwowany::~~Zarezerwowany()  
{  
    delete ui;  
}  
  
void Zarezerwowany::on_pushButton_clicked()  
{  
    this->closeEvent(0);  
}
```

Klasa która odpowiedzialna jest za komunikację o poprawnym zarezerwowaniu pojazdu.

6. Wnioski

Wstępna koncepcja projektu zakładała wykorzystanie bazy danych MySQL. Z powodu problemów ze sterownikiem bazy danych w projekcie bazę zastąpiono plikami .csv

Nie stworzyliśmy funkcjonalności wyszukiwania pojazdów z powodu braku czasu.

Do pełni funkcjonalności projektu brakuje następujące rzeczy:

- brak walidacji wprowadzanych danych

- brak możliwości z poziomu aplikacji dodawania nowych pracowników

- brak przypiętej do menu instrukcji użytkownika programu przez użytkownika jak i pracownika

- mało komfortowa funkcja usunięcia pojazdu

- brak auto uzupełnienia danych w zakładce zarezerwowane

- brak zapamiętania klienta w bazie danych

- brak skalowania okna pod konkretne monitory

- brak pobierania daty wypożyczenia i zwrotu pojazdu na podstawie których można będzie możliwość obliczenia należnej kwoty.

Możliwe poprawy w kodzie:

- brak refaktoryzacji kodu

- wiele powtórzeń podobnych funkcji

- poprawa czytelności kodu

Nad projektem pracowaliśmy około 116h.

Projekt zawiera około 3300 linii kodu dla UI, około 495 linii kodu z nagłówkami klas oraz około 2300 linii z funkcjonalnością projektu. W sumie projekt ma około 6000

Przewidywany czas na doprowadzenie projektu do pełni funkcjonalności około 50h.

W trakcie realizacji projektu poznaliśmy bibliotekę Qt oraz pogłęбилиśmy wiedzę na temat programowania obiektowego w C++