Санкт-Петербургский Национальный
Исследовательский Университет
Информационных технологий, механики и оптики

Домашняя работа

Разработка приложения

Выполнил: Смирнов
Игорь Иванович
Группа № К3121
Проверила: Казанова
Полина Петровна

Санкт-Петербург
2023

1 Цель работы

Создать программное обеспечение учета грузового транспорта для Автотранспортного отдела логистической компании.

2 Задачи

Данное программное обеспечение должно выполнять следующие задачи:

- Возможность добавлять/удалять грузовой транспорт;
- Возможность просматривать весь доступный транспорт;
- Возможность просматривать грузовой транспорт по грузоподъемности;
- Возможность просматривать свободный грузовой транспорт;
- Возможность вносить заявку на перевоз груза по указанным габаритам;
- Возможность подобрать и забронировать транспорт;
- Возможность просматривать занятый грузовой транспорт;
- Возможность сохранения данных в базу данных.

3 Ход работы

3.1 Общая информация

Данное программное обеспечение написано на языке Python с применением ООП. Приложение состоит из 4 файлов кода, среди которых исполняемым файлом является InterFace.py, а файл BDadmins.py выполняется только один раз для базового заполнения базы данных, и более чем 10 различных классов. Связь классов представлена на рисунке 1.

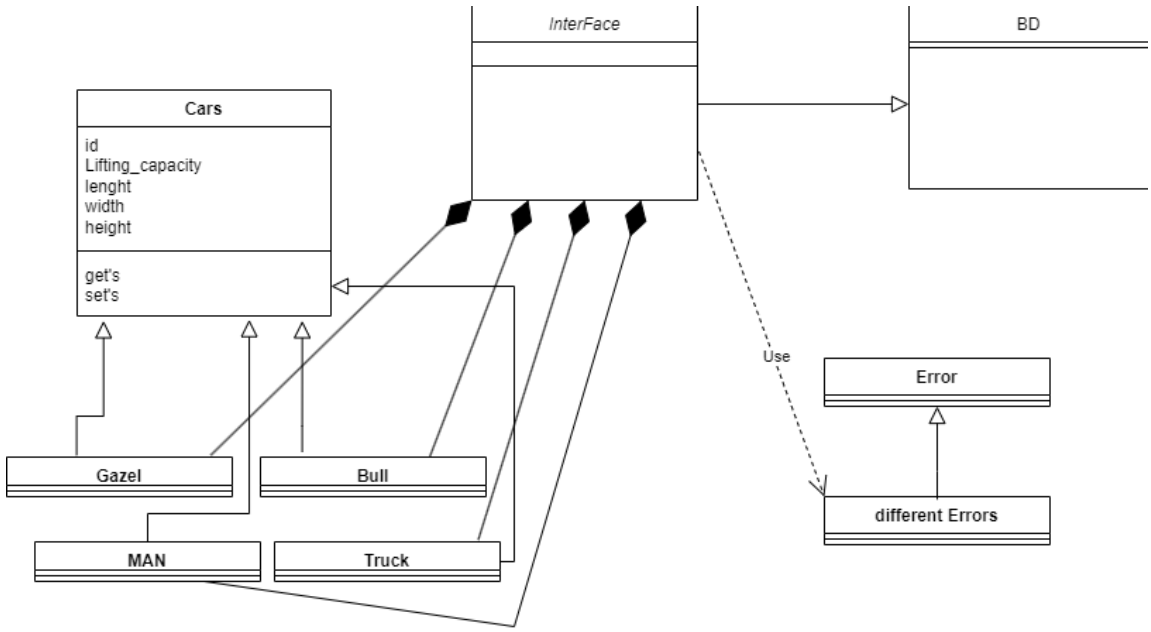

Рисунок 1. - Диаграмма классов

В ходе работы были задействованы библиотеки:

1) tkinter - создание графического интерфейса;

2) datetime - работа с датами;

3) sqlite3 - работа с базами данных.

3.2 База данных

СУБД для базы данных является SQLite3. Сама база данных состоит из 3 таблиц:

1) tUsers - информация об аккаунтах. Имеет 3 столбца: login, password, admin

2) tCars - информация о имеющихся авомобилях. Имеет 7 столбцов: Name, id, Length, Width, Height, Lifting_capacity, Busy.

3) tOrders - информация об активных заказах. Имеет 4 столбца: Name_of_customer, id_car, cargo, end_datetime.

Связь таблиц представлена на рисунке 2.

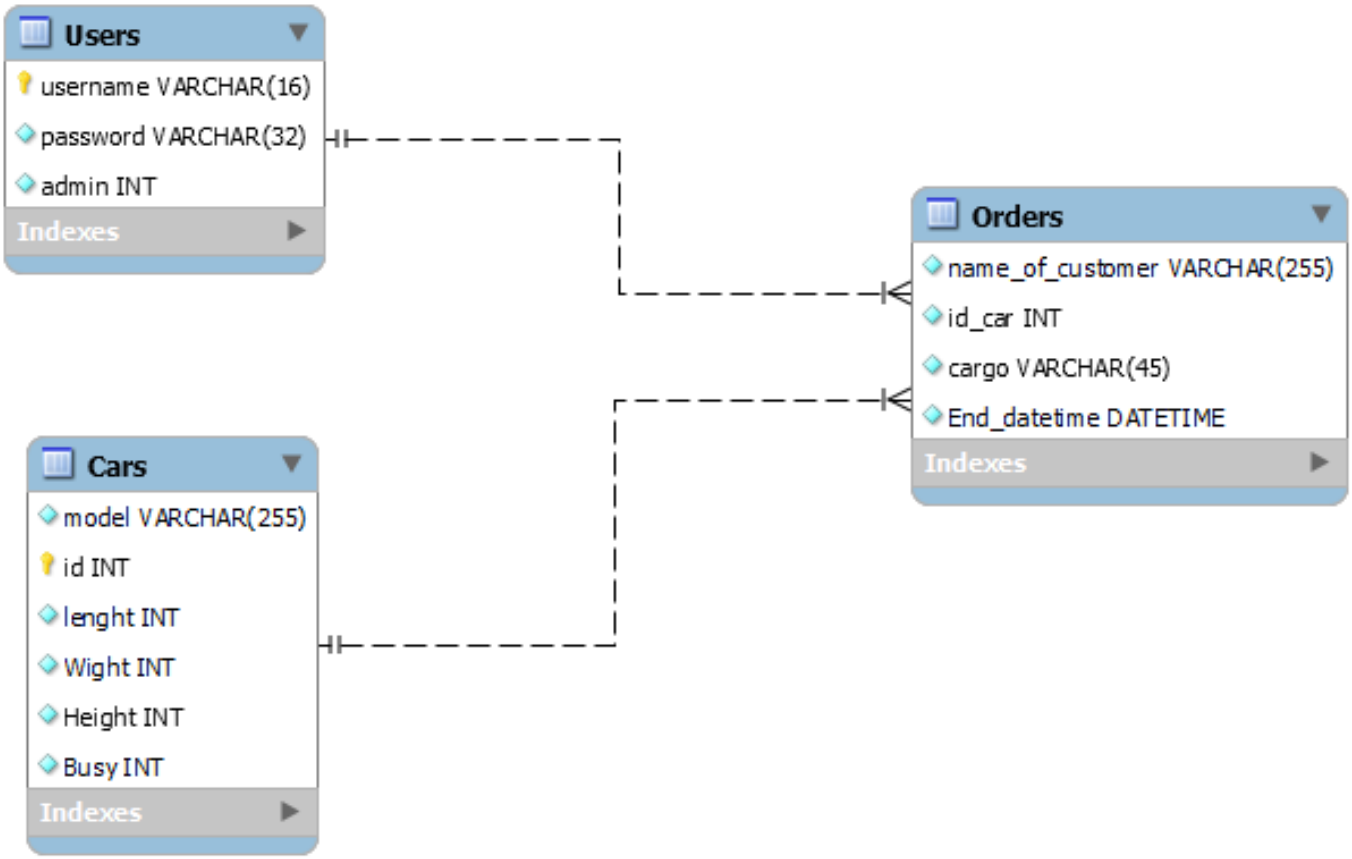

Рисунок 2. - Схема базы данных

3.3 Cars

В файле Cars находятся 5 классов. Класс Car - базовый класс и дочерние классы Gazel, Bull, MAN и Truck. Они отвечают за информацию об автомобилях при их добавлении в базу. Класс Car имеет конструктор, указывающий базовые параметры грузоподьемности, длины, ширины и высоты машины, а также методы get и set для доступа к этим параметрам. Дочерние классы пользуются этими методами, но имеют собственный get_name, чтобы

возвращать название модели. Код данного файла представлен на рисунке 3, рисунке 4, рисунке 5 и рисунке 6.

```python
class Car: #Базовый класс
    def __init__(self, id):
        self.__id = id #если будет больше одного экземпляра одной машины
        self.__lifting_capacity = 1 #кг
        self.__length = 1  # см
        self.__width = 1 #см
        self.__height = 1 #см
    #get
    2 usages (1 dynamic)
    def get_LeftCap(self):
        return self.__lifting_capacity

    2 usages (1 dynamic)
    def get_id(self):
        return self.__id

    2 usages (1 dynamic)
    def get_width(self):
        return self.__width

    2 usages (1 dynamic)
    def get_length(self):
        return self.__length

    2 usages (1 dynamic)
    def get_height(self):
        return self.__height
```

Рисунок 3. - Конструктор класса Car и методы get

```python
#set
4 usages
def set_LeftCap(self, lifting_capacity):
    self.__lifting_capacity = lifting_capacity

4 usages
def set_width(self, width):
    self.__width = width

4 usages
def set_length(self, length):
    self.__length = length

4 usages
def set_height(self, height):
    self.__height = height
```

Рисунок 4. - методы set

```python
class Gazel(Car):
    def __init__(self, id):
        super().__init__(id)
        self.__name = 'ГАЗ-3302 «Газель»'
        self.__lifting_capacity = self.set_LeftCap(2000)  # кг
        self.__length = self.set_length(300)  # см
        self.__width = self.set_width(200)  # см
        self.__height = self.set_height(170)  # см

    54 usages (53 dynamic)
    def get_name(self):
        return self.__name

1 usage
class Bull(Car):
    def __init__(self,id):
        super().__init__(id)
        self.__name = 'ЗИЛ-5301 (Бычок)'
        self.__lifting_capacity = self.set_LeftCap(3000)  # кг
        self.__length = self.set_length(420)  # см
        self.__width = self.set_width(200)  # см
        self.__height = self.set_height(200)  # см

    54 usages (53 dynamic)
    def get_name(self):
        return self.__name
```

Рисунок 5. - классы Gazel и Bull

```python
1 usage
class MAN(Car):
    def __init__(self,id):
        super().__init__(id)
        self.__name = 'MAN-10'
        self.__lifting_capacity = self.set_LeftCap(10000)  # кг
        self.__length = self.set_length(600)  # см
        self.__width = self.set_width(245)  # см
        self.__height = self.set_height(230)  # см

    54 usages (53 dynamic)
    def get_name(self):
        return self.__name

1 usage
class Truck(Car):
    def __init__(self,id):
        super().__init__(id)
        self.__name = 'Фура Mercedes-Benz Actros'
        self.__lifting_capacity = self.set_LeftCap(20000)  # кг
        self.__length = self.set_length(1360)  # см
        self.__width = self.set_width(246)  # см
        self.__height = self.set_height(250)  # см

    54 usages (53 dynamic)
    def get_name(self):
        return self.__name
```

Рисунок 6. - классы MAN и Truck

3.4 BDadmins

Данный файл заполняет базу данных базовыми зачениями и исполняется отдельно от остальных файлов. Код данного файла представлен на рисунке 7 и рисунке 8. Наполнение базы данных для трех таблиц представлены на рисунке 9, рисунке 10 и рисунке 11.

```
import sqlite3
import datetime
try:
    connection = sqlite3.connect('AllBD.db')
    cur = connection.cursor()
    cur.execute("'CREATE TABLE IF NOT EXISTS tUsers(Name PRIMARY KEY, Password, Admin)'")
    cur.execute('CREATE TABLE IF NOT EXISTS tCars(Name, id, Length, Width, Height, Lifting_capacity, Busy)')
    cur.execute('CREATE TABLE IF NOT EXISTS tOrders(Name_of_customer, id_car, cargo, end_datetime, FOREIGN KEY(Name_of_customer) REFERENCES tUsers(Name), '
                'FOREIGN KEY(id_car) REFERENCES tCars(id))')
    cur.execute("""INSERT INTO tUsers VALUES
    ('IgorSmir', 368813, 1),
    ('VeraShal', 367733, 1),
    ('NatashaPetr', 368646, 1)""")
    new_item = ('Kirill', 630456, 0)
    cur.execute("""INSERT INTO tUsers VALUES(?, ?, ?)""", new_item)
    cur.execute("""INSERT INTO tCars VALUES
    ('MAN-10', 111111, 600, 245, 230, 10000, 0),
    ('MAN-10', 111112, 600, 245, 230, 10000, 1),
    ('ГАЗ-3302 «Газель»', 111113, 300, 200, 170, 2000, 0),
    ('ГАЗ-3302 «Газель»', 111115, 300, 200, 170, 2000, 1),
    ('ЗИЛ-5301 (Бычок)', 111116, 420, 200, 200, 3000, 0),
    ('ЗИЛ-5301 (Бычок)', 111117, 420, 200, 200, 3000, 0),
    ('ЗИЛ-5301 (Бычок)', 111118, 420, 200, 200, 3000, 1),
    ('Фура Mercedes-Benz Actros', 111119, 1360, 246, 250, 20000, 0)
    """)
```

Рисунок 7. - Код BDadmins. часть 1

```
    new_item = ('Глеб', 2320, 0)
    cur.execute("""INSERT INTO tUsers VALUES(?, ?, ?)""", new_item)
    new_orders = [('IgorSmir', 111112, 'Кирпичи', datetime.datetime(2023, 6, 6, 14, 10, 0)),
    ('VeraShal', 111115, 'Молоко', datetime.datetime(2023, 6, 7, 10, 0, 0)),
    ('Kirill', 111118, 'Шифер', datetime.datetime(2023, 6, 6, 12, 30, 00))]
    cur.executemany("""INSERT INTO tOrders VALUES (?,?,?,?)""", new_orders)
    cur.close()
finally:
    if (connection):
        connection.commit()
        connection.close()
        print('Соединение закрыто')
```

Рисунок 8. - Код BDadmins. часть 2

	Name	Password	Admin
	Фильтр	Фильтр	Фильтр
1	IgorSmir	368813	1
2	VeraShal	367733	1
3	NatashaPetr	368646	1
4	Kirill	630456	0
5	Глеб	2320	0

Рисунок 9. - Содержимоей таблицы tUsers

	Name	id	Length	Width	Height	Lifting_capacity	Busy
	Фильтр	Фил...	Фильтр	Фил...	Фильтр	Фильтр	Фи...
1	MAN-10	111111	600	245	230	10000	0
2	MAN-10	111112	600	245	230	10000	1
3	ГАЗ-3302 «Газель»	111113	300	200	170	2000	0
4	ГАЗ-3302 «Газель»	111115	300	200	170	2000	1
5	ЗИЛ-5301 (Бычок)	111116	420	200	200	3000	0
6	ЗИЛ-5301 (Бычок)	111117	420	200	200	3000	0
7	ЗИЛ-5301 (Бычок)	111118	420	200	200	3000	1
8	Фура Mercedes-Benz Actros	111119	1360	246	250	20000	0

Рисунок 10. - Содержимоей таблицы tCars

	Name_of_customer	id_car	cargo	end_datetime
	Фильтр	Филь...	Фильтр	Фильтр
1	IgorSmir	111112	Кирпичи	2023-06-06 14:10:00
2	VeraShal	111115	Молоко	2023-06-07 10:00:00
3	Kirill	111118	Шифер	2023-06-06 12:30:00

Рисунок 11. - Содержимоей таблицы tOrders

3.5 Exceptions

В данном файле содержится класс Error, являющийся наследником класса Exception, а также множество его дочерних классов. Класс Error и часть его дочерних классов представлены на рисунке 12.

```
class Error(Exception):
    pass



#исключение некорректного ввода
2 usages
class IncorrectInputExcError(Error):
    '''Некорректный ввод'''
    pass

class NoDataInBDError(Error):
    '''Таких данных нет в базе данных пользователей'''
    pass

2 usages
class DublicateidError(Error):
    '''Указанный id уже существует'''
    pass

2 usages
class DublicatenameError(Error):
    '''Одинаковое имя пользователя при регистрации'''
    pass
```

Рисунок 12. - класс Error и дочерние классы

3.6 InterFace

В данном файле содержатся два самых главных класса проекта:

1) BD - отвечает за работу с базами данных;

2) InterFace - отвечает за страницы интерфейса. Является дочерним классом для класса BD.

За начало работы программы отвечает класс InterFace. Поэтому создается объект этого класса и вызывается метод start_app, который создает окно и вызывает метод start_page, который покажется стартовое окно. Данные методы показаны на рисунке 13.

```
    def start_page(self):
        canvas = Canvas(root, height=700, width=700)
        canvas.pack()
        global frame
        frame = Frame(root, bg='blue')
        frame.place(relwidth=1, relheight=1)
        title = Label(frame, text='Truck manager 2023', bg='red', font=30)
        title.place(relx=0.4, rely=0.05, width=300)
        title = Label(frame, text='Добро пожаловать', bg='blue', font=30)
        title.place(relx=0.4, rely=0.1, width=300)
        btn = Button(frame, text='Войти', bg='white', command=lambda: (self.log_in_page0(), self.delete_by_time()))
        btn.place(relx=0.45, rely=0.2, width=150)
        btn2 = Button(frame, text='Войти как администратор', bg='white', command=lambda: (self.log_in_page1(), self.delete_by_time()))
        btn2.place(relx=0.45, rely=0.25, width=150)
        btn3 = Button(frame, text='Зарегистрироваться', bg='white', command=lambda: (self.log_in_page2(), self.delete_by_time()))
        btn3.place(relx=0.45, rely=0.3, width=150)
        btn4 = Button(frame, text='Выйти', bg='white', command=exit)
        btn4.place(relx=0.45, rely=0.35, width=150)
    1 usage
    def start_app(self):
        global root
        root = Tk()
        root.attributes('-fullscreen', True)
        root.title('Грузоперевозки')
        # root.geometry('1024x960')
        self.start_page()
        root.mainloop()
v = InterFace()
v.start_app()
```

Рисунок 13. - методы start_page и start_app

Метод Start_page вызывает фрейм (все страницы приложения - фреймы с различным наполнением), на котором находится заглавие и 4 кнопки (рисунок 14):

1) Войти - Вход в аккаунт;

2) Войти как администратор - вход в аккаунт администратора;

3) Зарегистрироваться - регистарация нового аккаунта;

4) Выйти - выход из программы.

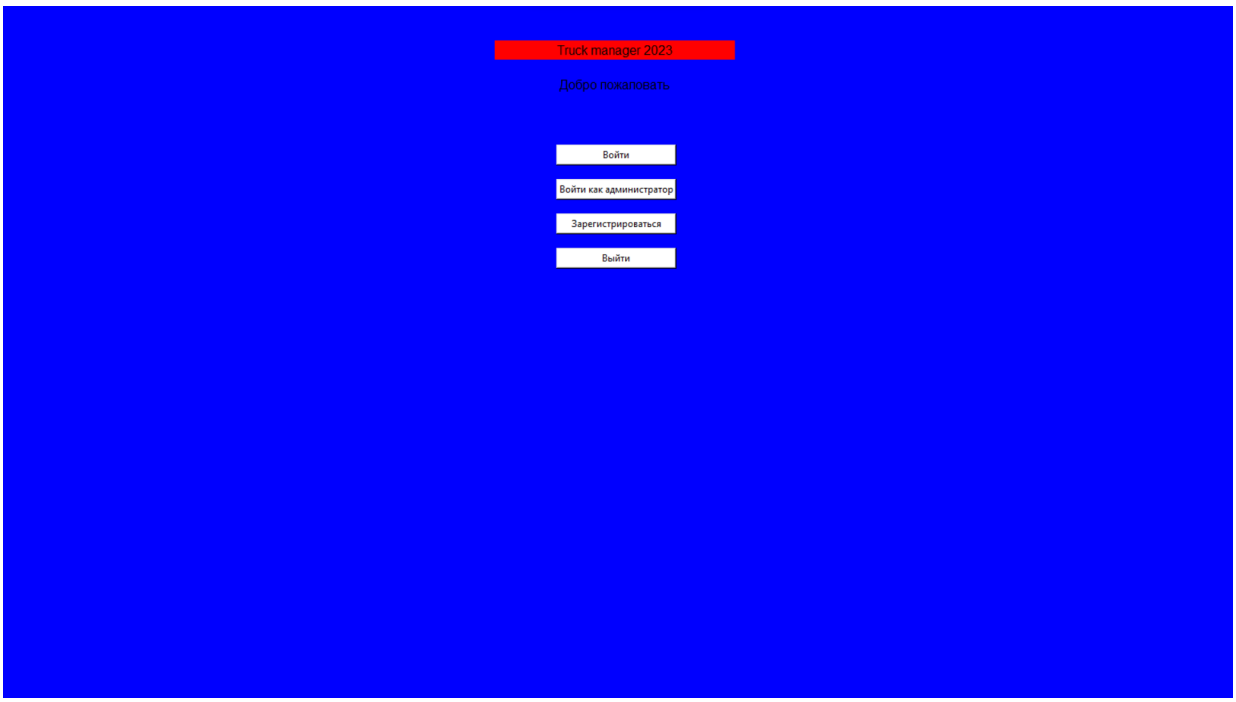

Рисунок 14. - Начальный экран

Первые 3 кнопки вызывают методы log_in_page0, log_in_page1, log_in_page2. Первые 2 метода открывают одинаковые по структуре окна (рисунок 20), но отличаются в параметрах при вызове метода BD_check по нажатию кнопки (рисунок 15, рисунок 16, рисунок 17). Метод log_in_page2 имеет небольшое отличие в коде (рисунок 18) и интерфейсе(рисунок 21), а также вызывает другой метод BD_update (рисунок 19).

```
def log_in_page0(self): #Обычный вход
    frame.destroy()
    log_frame = Frame(root, bg='blue')
    log_frame.place(relwidth=1, relheight=1)
    #Добавить на все страницы кроме главной
    exit_btn = Button(log_frame, text='X', bg='red', fg='white', command=exit, font=200)
    exit_btn.place(relx=0.98, rely=0, height=30, width=40)

    title1 = Label(log_frame, text='Введите логин и пароль', bg='red', font=30, width=300)
    title1.place(relx=0.4, rely=0.05, width=300)

    title_login = Label(log_frame, text='Логин:', bg='blue', font=30, fg='white')
    title_login.place(relx=0.4, rely=0.1, width=130)
    self.LoginInput = Entry(log_frame, bg='white')
    self.LoginInput.place(relx=0.47, rely=0.11)

    title_password = Label(log_frame, text='Пароль:', bg='blue', font=30, fg='white')
    title_password.place(relx=0.4, rely=0.18, width=130)
    PasswordInput = Entry(log_frame, bg='white', show='*')
    PasswordInput.place(relx=0.47, rely=0.19)

    self.name=self.LoginInput.get()
    btn_in = Button(log_frame, text='Зайти', bg='white', command=lambda: (self.BD_check(self.LoginInput.get(), PasswordInput.get(), 0, log_frame), self.delete_by_time()))
    btn_in.place(relx=0.45, rely=0.25, width=150)
    back_button = Button(log_frame, text='Вернуться в главное меню', bg='white',
                         command=lambda: (log_frame.destroy(), self.delete_by_time(), self.start_page()))
    back_button.place(relx=0.45, rely=0.3, width=150)
```

Рисунок 15. - метод log_in_page0

```
def log_in_page1(self): #администрация вход
    frame.destroy()
    log_frame = Frame(root, bg='blue')
    log_frame.place(relwidth=1, relheight=1)

    exit_btn = Button(log_frame, text='X', bg='red', fg='white', command=exit, font=200)
    exit_btn.place(relx=0.98, rely=0, height=30, width=40)


    title1 = Label(log_frame, text='Введите свои данные', bg='red', font=30)
    title1.place(relx=0.4, rely=0.05, width=300)
    title_login = Label(log_frame, text='Логин:', bg='blue', font=30, fg='white')
    title_login.place(relx=0.4, rely=0.1, width=130)
    self.LoginInput = Entry(log_frame, bg='white')
    self.LoginInput.place(relx=0.47, rely=0.1)

    title_password = Label(log_frame, text='Пароль:', bg='blue', font=30, fg='white')
    title_password.place(relx=0.4, rely=0.18, width=130)
    PasswordInput = Entry(log_frame, bg='white', show='*')
    PasswordInput.place(relx=0.47, rely=0.18)

    self.name = self.LoginInput.get()
    btn_in1 = Button(log_frame, text='Зайти', bg='white', command=lambda: (self.BD_check(self.LoginInput.get(), PasswordInput.get(), 1, log_frame), self.delete_by_time()))
    btn_in1.place(relx=0.45, rely=0.25, width=150)
    back_button = Button(log_frame, text='Вернуться в главное меню', bg='white',
                         command=lambda: (log_frame.destroy(), self.delete_by_time(), self.start_page()))
    back_button.place(relx=0.45, rely=0.3, width=150)
```

Рисунок 16. - метод log_in_page1

```
2 usages
def BD_check(self, name, password, admin, frame): #Проверка нахождения аккаунта в БД
    conection = sqlite3.connect('AllBD.db')
    cur = conection.cursor()
    users = cur.execute("""SELECT * FROM tUsers""")
    for i in users:
        if name == (i[0]) and str(password) == str(i[1]) and admin == (i[2]):
            cur.close()
            v = InterFace()
            if admin == 1:
                v.admin_page(name)
            else:
                v.main_page(name)
    cur.close()
    conection.close()
    ErrorLabel = Label(frame, text='Неверный логин или пароль', bg='blue', fg='red', font=20, width=30)
    ErrorLabel.place(relx=0.395, rely=0.22)
```

Рисунок 17. - метод BD_check

```
def log_in_page2(self): #регистрация
    frame.destroy()
    log_frame = Frame(root, bg='blue')
    log_frame.place(relwidth=1, relheight=1)

    exit_btn = Button(log_frame, text='X', bg='red', fg='white', command=exit, font=200)
    exit_btn.place(relx=0.98, rely=0, height=30, width=40)

    title1 = Label(log_frame, text='Введите свои данные', bg='red', font=30)
    title1.place(relx=0.4, rely=0.05, width=300)
    title_login = Label(log_frame, text='Логин:', bg='blue', font=30, fg='white')
    title_login.place(relx=0.4, rely=0.1, width=130)
    self.LoginInput = Entry(log_frame, bg='white')
    self.LoginInput.place(relx=0.47, rely=0.1)

    title_password = Label(log_frame, text='Пароль:', bg='blue', font=30, fg='white')
    title_password.place(relx=0.4, rely=0.18, width=130)
    PasswordInput = Entry(log_frame, bg='white', show='*')
    PasswordInput.place(relx=0.47, rely=0.18)

    btn_in2 = Button(log_frame, text='Зарегистрироваться', bg='white', command=lambda: (self.BD_update(self.LoginInput.get(),
                                                                                         PasswordInput.get(), 0, log_frame), self.delete_by_time()))
    btn_in2.place(relx=0.45, rely=0.25, width=150)
    back_button = Button(log_frame, text='Вернуться в главное меню', bg='white',
                         command=lambda: (log_frame.destroy(), self.delete_by_time(), self.start_page()))
    back_button.place(relx=0.45, rely=0.3, width=150)
```

Рисунок 18. - метод log_in_page2

```python
def BD_update(self, name, password, admin, frame): #Запись данных нового аккаунта
    try:
        if password.isnumeric() == False:
            raise Exceptions.IncorrectInputExcError
        user_list = self.view_BD(1)
        print(user_list)
        for i in user_list:
            if i[0] == name:
                raise Exceptions.DublicatenameError
        connection2 = sqlite3.connect('AllBD.db')
        cur = connection2.cursor()
        new_item = (name, int(password), admin)
        cur.execute("""INSERT INTO tUsers Values (?, ?, ?)""", new_item)
        cur.close()
        Succes_Label = Label(frame, text='Регистрация прошла успешно', bg='blue', fg='white', font=20, width=300)
        Succes_Label.place(relx=0.395, rely=0.22, width=300)
    except Exceptions.DublicatenameError:
        ErrorLabel = Label(frame, text='Данное имя уже занято', bg='blue', fg='red', font=20, width=300)
        ErrorLabel.place(relx=0.395, rely=0.22)
    except sqlite3.Error as error:
        print('Ошибка при подключение SQLite', error)
    except Exceptions.IncorrectInputExcError:
        ErrorLabel = Label(frame, text='Пароль должен содержать только цифры', bg='blue', fg='red', font=20, width=30)
        ErrorLabel.place(relx=0.395, rely=0.22, width=300)
    finally:
        if (connection2):
            connection2.commit()
            connection2.close()
```

Рисунок 19. - метод BD_update

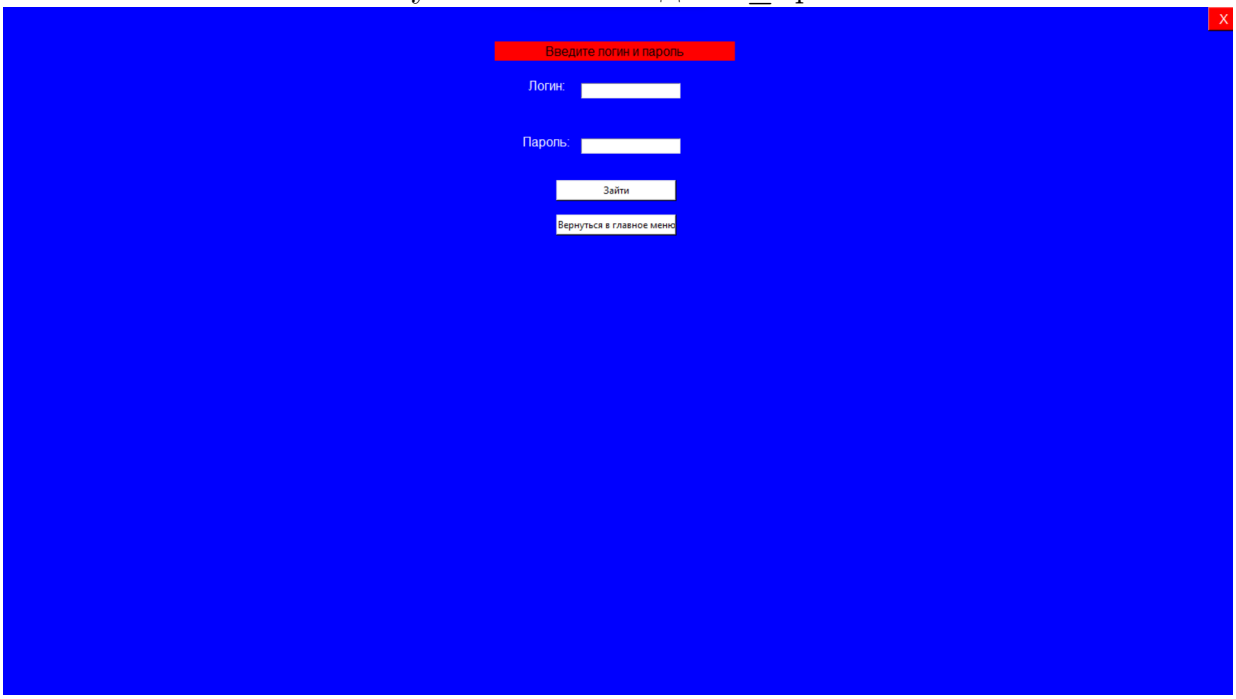

Рисунок 20. - Страница входа

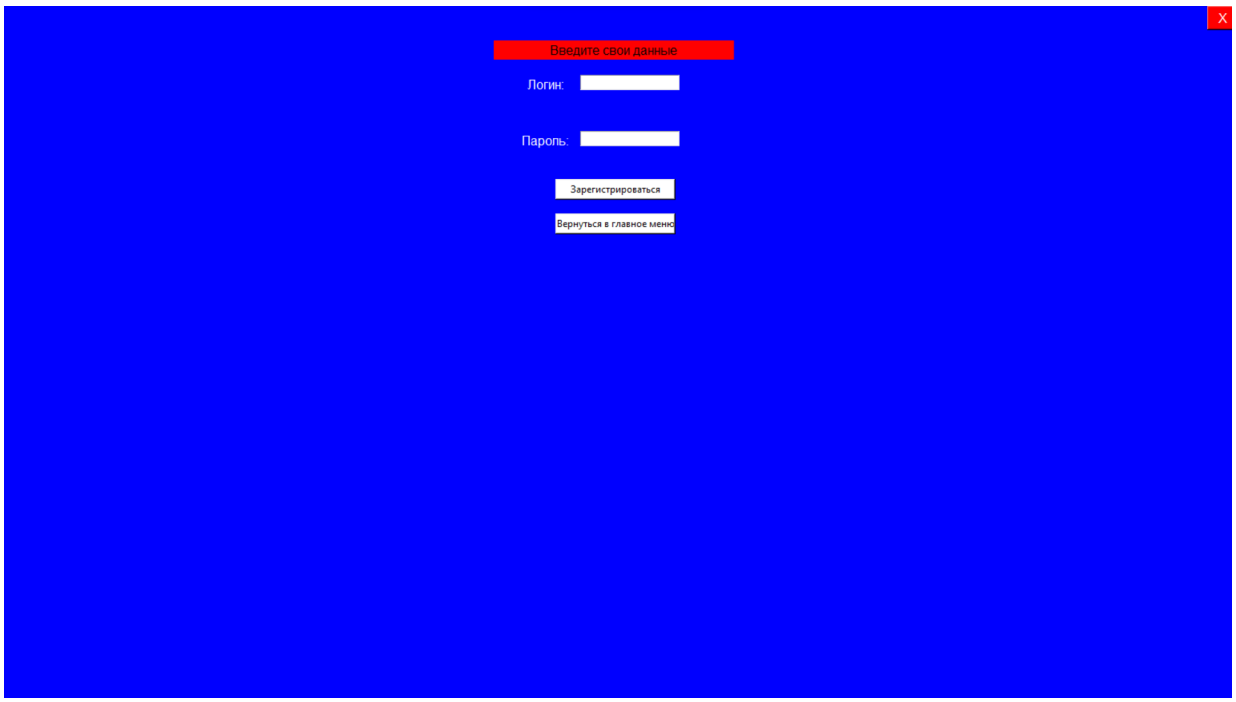

Рисунок 21. - Страница регистрации

Красная кнопка в верхнем правом углу завершает работу программы. Она есть на страницах, на которых нет кнопки "Выход". Кнопка "Вернуться в главное меню"возвращает стартовый экран.

Программа выполняет проверки на соответствие введенных данных с данными из базы данных. Также существует проверка на некорректный ввод, например, пароль должен быть только числовым значением.

При заходе обычного пользователя и администратора открываются разные окна (не хватает опций добавления и удаления автомобилей у обычного пользователя), но далее рассматриваться все будет с позиции администратора, так как он обладает всеми возможностями. Метод реализует фрейм с кнопками, которые ведут к исполнению возможностей, обозначенных в задачах. Код представлен на рисунке 22 и рисунке 23 Интерфейс представлен на рисунке 24.

```
def admin_page(self, name):
    self.name =name
    admin_Frame = Frame(root, bg='blue')
    admin_Frame.place(relwidth=1, relheight=1)


    main_title = Label(admin_Frame, text='Главное меню', bg='red', font=30)
    main_title.place(relx=0.4, rely=0.1, width=300)

    admbtn1 = Button(admin_Frame, text='Добавить автомобиль', bg='white', command=lambda: (self.add_new_car_page(), self.delete_by_time()))
    admbtn1.place(relx=0.41, rely=0.2, width=270)

    admbtn2 = Button(admin_Frame, text='Удалить автомобиль', bg='white', command=lambda: (self.delete_car_page(), self.delete_by_time()))
    admbtn2.place(relx=0.41, rely=0.25, width=270)

    mnbtn1 = Button(admin_Frame, text='Просмотреть весь доступный транспорт', bg='white', command=lambda: (self.view_all_cars_page(), self.delete_by_time()))
    mnbtn1.place(relx=0.41, rely=0.3, width=270)

    mnbtn2 = Button(admin_Frame, text='Просмотреть весь свободный транспорт', bg='white', command=lambda: (self.view_free_cars_page(), self.delete_by_time()))
    mnbtn2.place(relx=0.41, rely=0.35, width=270)

    mnbtn22 = Button(admin_Frame, text='Просмотреть весь занятый транспорт', bg='white',
                     command=lambda: (self.view_busy_cars_page(), self.delete_by_time()))
    mnbtn22.place(relx=0.41, rely=0.40, width=270)
```

Рисунок 22. - Код метода admin_menu. Часть 1

```
    mnbtn3 = Button(admin_Frame, text='Просмотреть транспорт по грузоподъёмности', bg='white', command=lambda: (self.sort_models_page(), self.delete_by_time()))
    mnbtn3.place(relx=0.41, rely=0.45, width=270)

    mnbtn4 = Button(admin_Frame, text='Создать новую заявку на перевоз', bg='white', command=lambda_: (self.new_offer_page(), self.delete_by_time()))
    mnbtn4.place(relx=0.41, rely=0.5, width=270)

    mnbtn5 = Button(admin_Frame, text='Подобрать и забронировать транспорт', bg='white', command=lambda: (self.parametres_page(), self.delete_by_time()))
    mnbtn5.place(relx=0.41, rely=0.55, width=270)

    mnbtn6 = Button(admin_Frame, text='Выйти из аккаунта', bg='white',
                    command=lambda: (self.start_page(), admin_Frame.destroy(), self.delete_by_time()))
    mnbtn6.place(relx=0.41, rely=0.6, width=270)

    mnbtn7 = Button(admin_Frame, text='Выйти из системы', bg='white', command=exit)
    mnbtn7.place(relx=0.41, rely=0.65, width=270)
```

Рисунок 22. - Код метода admin_menu. Часть 2

Рисунок 24. - Меню администратора

В течение кода в параметре command у кнопок фигурирует метод delete_by_time. Этот метод отвечает за удаление данных о заказе, когда подойдет время его окончания. Код этого метода и вспомогательного к нему на рисунке 25.

```
def delete_by_time(self):
    try:
        connection6 = sqlite3.connect('AllBD.db')
        list_of_offers = self.view_BD(3)
        for i in range(len(list_of_offers)):
            if self.normal_vid(datetime.datetime.now()) >= self.normal_vid(list_of_offers[i][-1]):
                cur = connection6.cursor()
                cur.execute("""DELETE FROM tOrders WHERE end_datetime=?""",(list_of_offers[i][-1],))
                cur.close()
                cur1 = connection6.cursor()
                cur1.execute("""UPDATE tCars set Busy=0 where id=?""", (int(list_of_offers[i][1]),))
                cur1.close()
    except sqlite3.Error as error:
        print('Ошибка проверки заказов', error)
    finally:
        if (connection6):
            print('Появился новый свободный автомобиль')
            connection6.commit()
            connection6.close()

2 usages
def normal_vid(self, time):
    time = str(time)
    stroka = ''
    for i in time:
        if i in '1234567890':
            stroka += i
    return stroka
```

Рисунок 25. - Методы delete_by_time и normal_vid

Метод add_new_car_page (код - рисунок 26) открывает страницу, на которой можно выбрать модель автомобиля и дать ему id (рисунок 27). Кнопка "Добавить автомобиль"вызывает метод car_bd(рисунок 28 и рисунок 29), который сохраняет новый автомобиль или уведомляет об ошибке, если пользователь ввел id, который уже существует или ввел не только цифры.

```
def add_new_car_page(self): #Добавление нового транспорта
    add_car_frame = Frame(root, bg='blue')
    add_car_frame.place(relwidth=1, relheight=1)

    exit_btn = Button(add_car_frame, text='X', bg='red', fg='white', command=exit, font=200)
    exit_btn.place(relx=0.98, rely=0, height=30, width=40)

    acf_title = Label(add_car_frame, text='Добавление нового автомобиля', bg='red', font=30)
    acf_title.place(relx=0.4, rely=0.1, width=300)

    var = StringVar()
    combobox = ttk.Combobox(add_car_frame, textvariable=var)
    cars = ['ГАЗ-3302 «Газель»', 'ЗИЛ-5301 (Бычок)', 'MAN-10', 'Фура Mercedes-Benz Actros']
    combobox['values'] = cars
    combobox['state'] = 'readonly'
    combobox.place(relx=0.35, rely=0.2, width=500)

    id_label = Label(add_car_frame, text='id автомобиля:', bg='blue', fg='white', font=30)
    id_label.place(relx=0.4, rely=0.25, width=150)
    id_for_new_car = Entry(add_car_frame, bg='white')
    id_for_new_car.place(relx_=_0.51, rely=0.26)

    add_button = Button(add_car_frame, text='Добавить автомобиль', bg='white', command=lambda: (self.car_bd(combobox.get(), id_for_new_car.get(), add_car_frame),
                                                                                                 self.delete_by_time()))
    add_button.place(relx_=_0.45, rely=0.30, width=150)
    back_btn = Button(add_car_frame, text='Назад', bg='white', command=lambda: (add_car_frame.destroy(), self.delete_by_time()))
    back_btn.place(relx_=_0.45, rely=0.35, width=150)
```

Рисунок 26. - Метод add_new_car_page

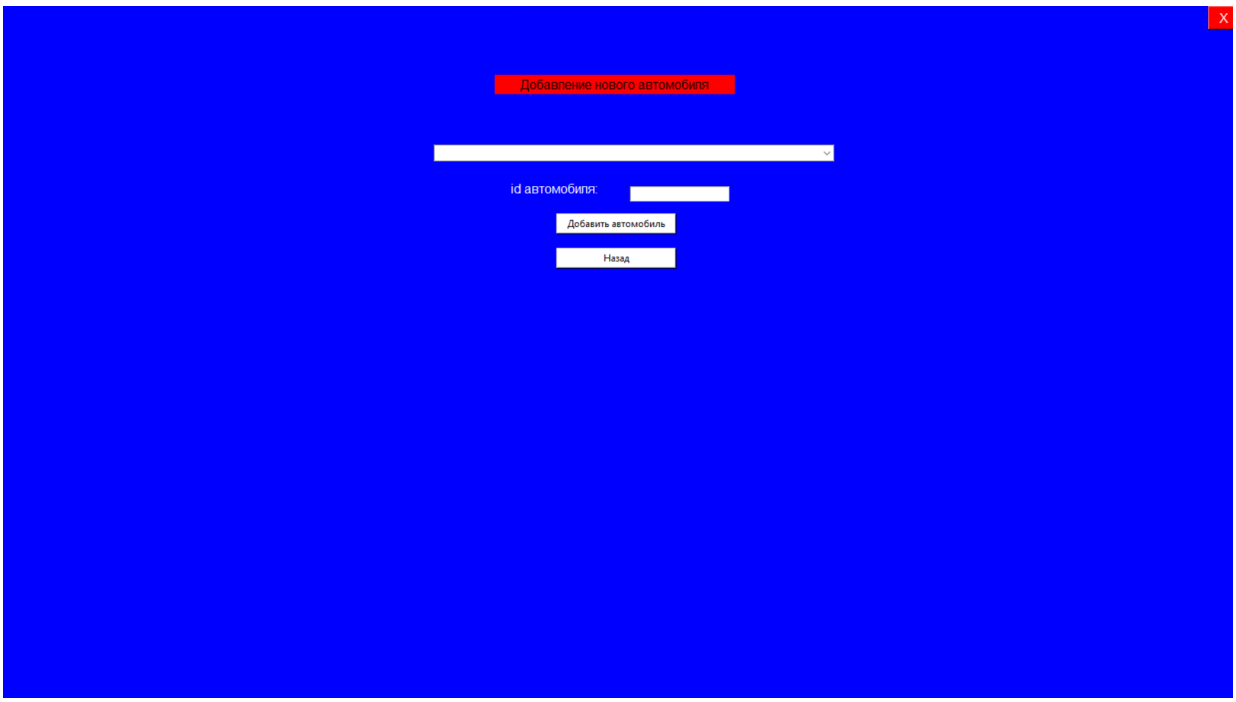

Рисунок 27. - Страница добавления автомобиля

```
def car_bd(self, model, id, frame): #Добавление новой машины в БД
    try:
        connection3 = sqlite3.connect('AllBD.db')
        cur = connection3.cursor()
        if model == 'ГАЗ-3302 «Газель»':
            car = Cars.Gazel(int(id))
        elif model == 'ЗИЛ-5301 (Бычок)':
            car = Cars.Bull(int(id))
        elif model == 'MAN-10':
            car = Cars.MAN(int(id))
        elif model == 'Фура Mercedes-Benz Actros':
            car = Cars.Truck(int(id))
        check_list = self.view_BD(2)
        for i in check_list:
            if int(i[1]) == int(id):
                raise Exceptions.DublicateidError
        new_item = (car.get_name(), car.get_id(), car.get_length(), car.get_width(), car.get_height(), car.get_LeftCap(), 0)
        cur.execute("""INSERT INTO tCars Values (?, ?, ?, ?, ?, ?, ?)""", new_item)
        cur.close()
        succes_label = Label(frame, text='Новый автомобиль успешно добавлен', bg='blue', fg='white', font=30)
        succes_label.place(relx=0.3, rely=0.4, width=500)
```

Рисунок 28. - Метод car_bd. Часть 1

```
except sqlite3.Error:
    pass
except Exceptions.DublicateidError:
    Error_label = Label(frame, text='Такой id уже существует', bg='blue', fg='red', font=30)
    Error_label.place(relx=0.4, rely=0.4, width=500)
except ValueError:
    Errorlab = Label(frame, text='Ошибка ввода! id - только числа', bg='blue', fg='red', font=30)
    Errorlab.place(relx=0.4, rely=0.4, width=500)
finally:
    if (connection3):
        print('Новый автомобиль успешно добавлен в базу')
        connection3.commit()
        connection3.close()
```

Рисунок 28. - Метод car_bd. Часть 2

Метод delete_car_page (рисунок 29) открывает страницу удаления автомобиля из базы данных (рисунок 30). На странице имеется список всех автомобилей, при выборе одного из них и нажатии на кнопку "Удалить автомобиль"вызывается метод car_del (рисунок 31) и происходит удаление ав-

томобиля из базы данных. Кнопка назад на любых страницах возвращает пользователя на одну страницу назад.

```
def delete_car_page(self):
    del_car_frame = Frame(root, bg='blue')
    del_car_frame.place(relwidth=1, relheight=1)

    exit_btn = Button(del_car_frame, text='X', bg='red', fg='white', command=exit, font=200)
    exit_btn.place(relx=0.98, rely=0, height=30, width=40)

    dcf_title = Label(del_car_frame, text='Удаление автомобилей', bg='red', font=30)
    dcf_title.place(relx=0.4, rely=0.1, width=300)
    list_of_car = self.view_BD(2)
    clear_list = []
    for i in list_of_car:
        if int(i[-1]) == 0:
            clear_list.append([i[0], i[1]])
    var1 = StringVar()
    combobox = ttk.Combobox(del_car_frame, textvariable=var1)
    combobox['values'] = clear_list
    combobox['state'] = 'readonly'
    combobox.place(relx=0.35, rely=0.2, width=500)
    del_btn = Button(del_car_frame, text='удалить автомобиль', bg='white', command=lambda: (self.car_del(combobox.get(), del_car_frame)))
    del_btn.place(relx=0.45, rely=0.30, width=150)
    back_btn = Button(del_car_frame, text='Назад', bg='white', command=lambda: (del_car_frame.destroy(), self.delete_by_time()))
    back_btn.place(relx=0.45, rely=0.35, width=150)
```

Рисунок 29. - Метод delete_car_page

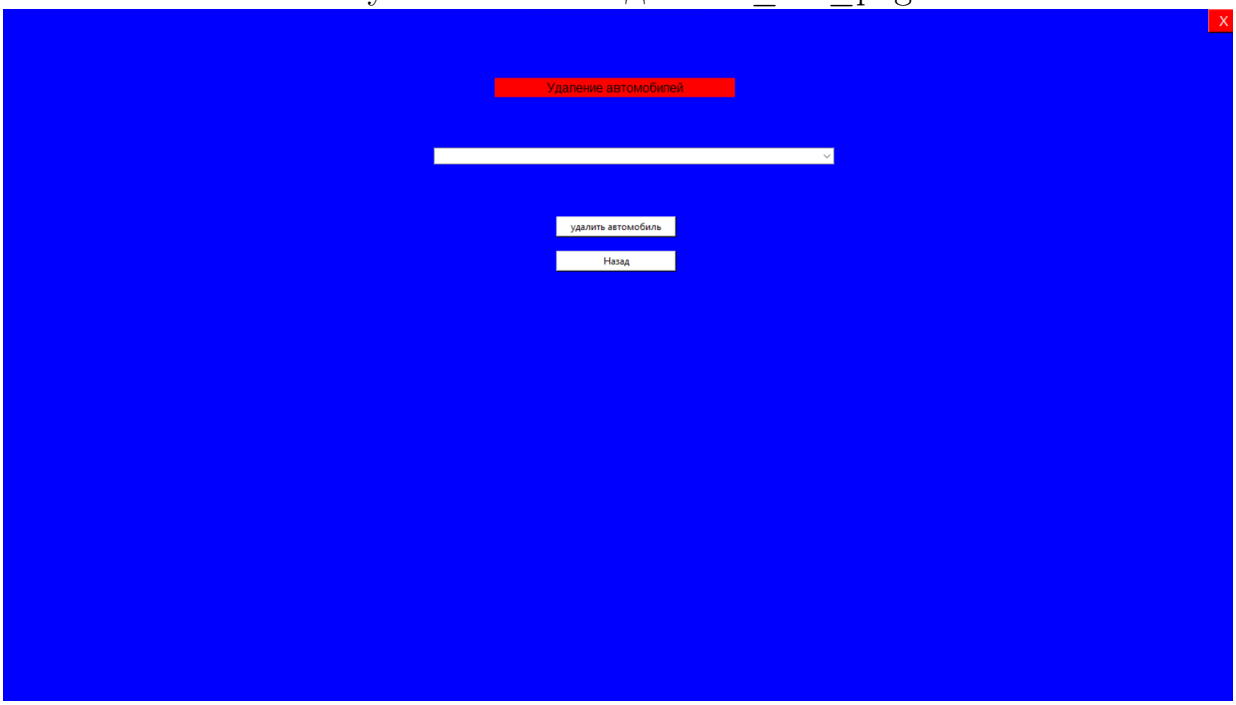

Рисунок 30. - Страница удаления автомобиля

```
1 usage
def car_del(self, name_andid_of_car, frame): #Удаление машины из базы
    try:
        id_of_car = int(name_andid_of_car[-6:])
        connection4 = sqlite3.connect('AllBD.db')
        cur = connection4.cursor()
        cur.execute("""DELETE FROM tCars WHERE id=?""", (id_of_car,))
        cur.close()
        succes_label = Label(frame, text='Новый автомобиль успешно удален', bg='blue', fg='white', font=30)
        succes_label.place(relx=0.3, rely=0.4, width=500)
    except sqlite3.Error as error:
        succes_label = Label(frame, text='Ошибка при удалении автомобиля', bg='blue', fg='red', font=30)
        succes_label.place(relx=0.3, rely=0.4, width=500)
    finally:
        if (connection4):
            print('Автомобиль успешно удален из базы')
            connection4.commit()
            connection4.close()
```

Рисунок 31. - Метод car_del

Кнопка "Просмотреть весь доступный транспорт"использует метод view_all_cars_page (рисунок 32), который открывает страницу показа всех имеющихся в базе автомобилей в виде scrollbar (рисунок 33).

```
def view_all_cars_page(self): #Просмотр всего имеющегося транспорта
    view_Frame = Frame(root, bg='blue')
    view_Frame.place(relwidth=1, relheight=1)
    vacp_title = Label(view_Frame, text='Просмотр всех автомобилей', bg='red', font=30)
    vacp_title.place(relx=0.4, rely=0.1, width=300)
    all_cars = self.view_BD(2)
    output_list = []
    for i in all_cars:
        output_list.append(str(i[0])+' ' + str(i[1]))
    cars_var = StringVar(value=output_list)
    listbox = Listbox(view_Frame,listvariable=cars_var)
    listbox.place(relx=0.35,  rely=0.2, width=500, height=300)
    scrollbar = ttk.Scrollbar(listbox,orient="vertical", command=listbox.yview)
    scrollbar.pack(side=RIGHT, fill=Y)

    listbox["yscrollcommand"] = scrollbar.set
    back_btn = Button(view_Frame, text='Назад', bg='white',
                      command=lambda: (view_Frame.destroy(), self.delete_by_time()))
    back_btn.place(relx=0.45, rely=0.7, width=150)
```

Рисунок 32. - Метод view_all_cars_page

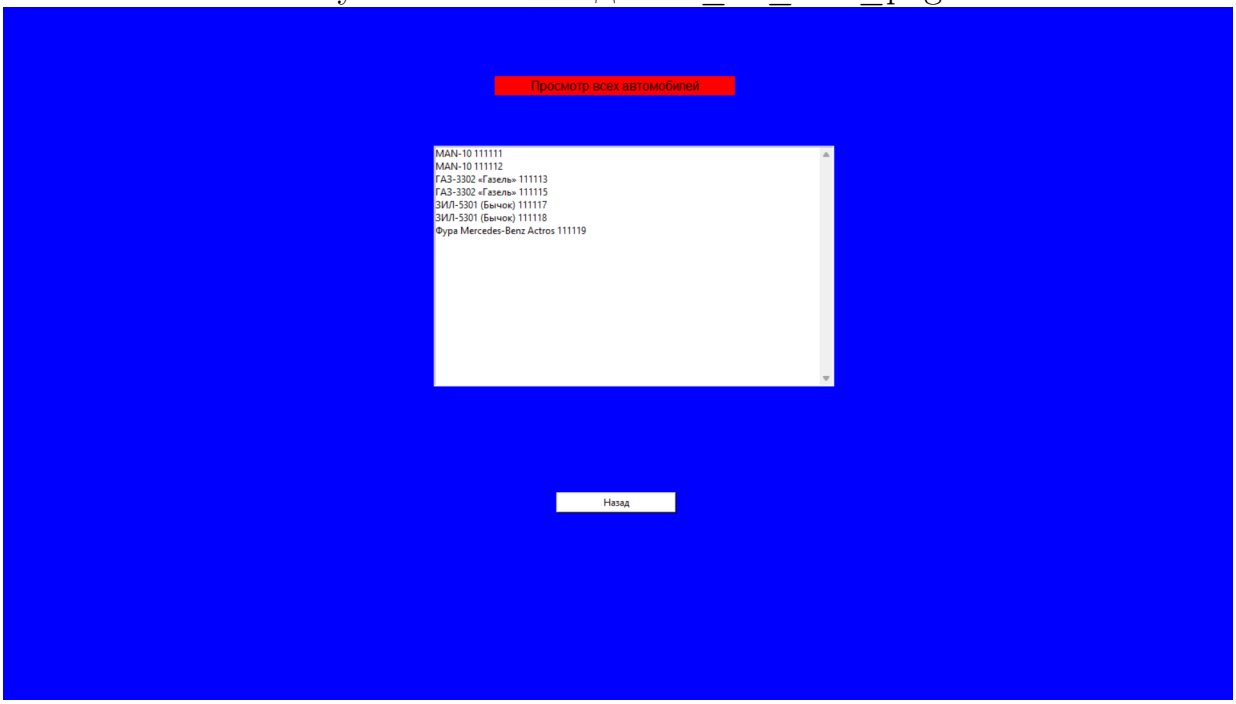

Рисунок 33. - Страница просмотра всех автомобилей

Функции "Посмотреть свободный транспорт"и "посмотреть занятый транспорт"имеют одинаковую структуру, поэтому рассматриваться будет только просмотр свободных автомобилей, а название метода для занятых будет в скобках. По натажатию кнопки вызывается метод view_free_cars_page

(view_busy_cars_page), представленный на рисунке 34. Показываемая страница представлена на рисунке 35

```
view_free_frame = Frame(root, bg='blue')
view_free_frame.place(relwidth=1, relheight=1)
exit_btn = Button(view_free_frame, text='X', bg='red', fg='white', command=exit, font=200)
exit_btn.place(relx=0.98, rely=0, height=30, width=40)
vfcp_title = Label(view_free_frame, text='Просмотр доступных автомобилей', bg='red', font=30)
vfcp_title.place(relx=0.35, rely=0.1, width=350)
list_of_free_cars = self.view_BD(2)
list_of_offers = self.view_BD(3)
free_cars = []
for i in list_of_free_cars:
    for j in list_of_offers:
        if int(i[-1]) == 1:
            continue
        if int(i[-1]) == 0 or (int(i[-1]) == 2 and j[0] == self.name and int(i[1]) == int(j[1])):
            free_cars.append(i)
free_cars = set(free_cars)
output_list = []
for i in free_cars:
    output_list.append(str(i[0])+' ' + str(i[1]))
print(output_list)
cars_var = StringVar(value=output_list)
listbox = Listbox(view_free_frame,listvariable=cars_var)
listbox.place(relx=0.32,  rely=0.2, width=500, height=300)
scrollbar = ttk.Scrollbar(listbox,orient="vertical", command=listbox.yview)
scrollbar.pack(side=RIGHT, fill=Y)
listbox["yscrollcommand"] = scrollbar.set
back_btn = Button(view_free_frame, text='Назад', bg='white',command=lambda: (view_free_frame.destroy(), self.delete_by_time()))
back_btn.place(relx=0.45, rely=0.7, width=150)
```

Рисунок 34. - Метод view_free_cars_page

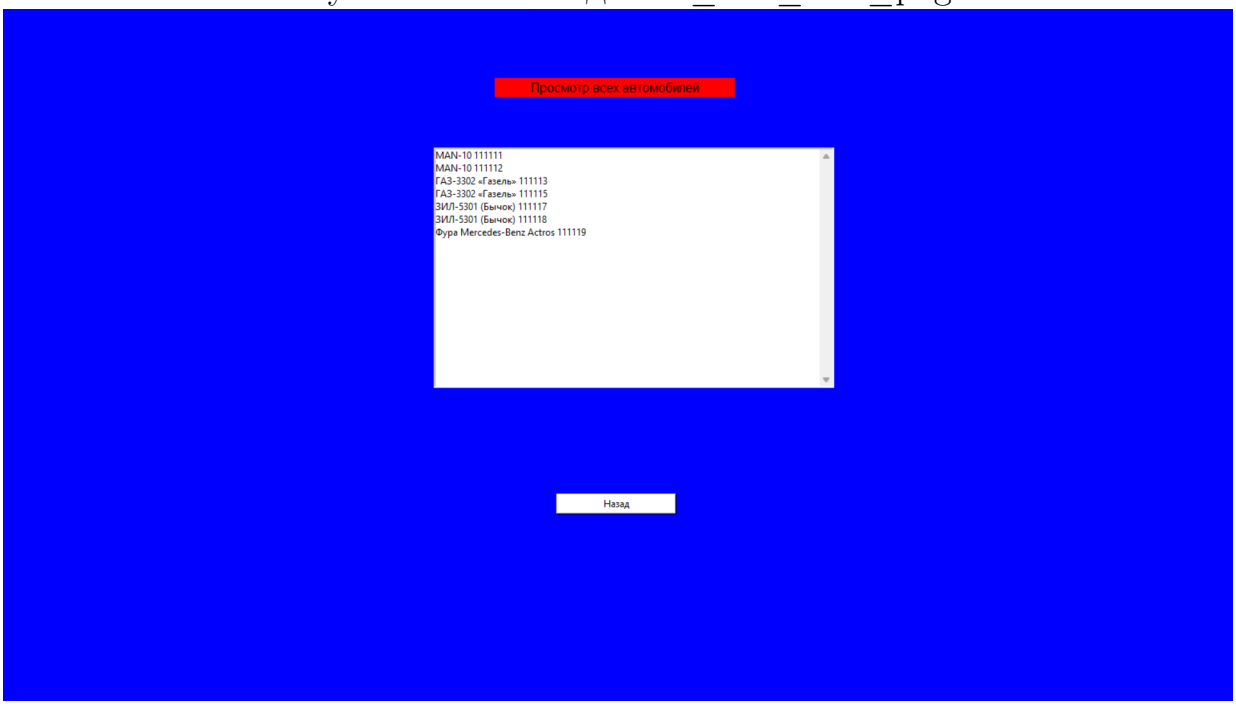

Рисунок 35. - Страница просмотра свободных автомобилей

Во всех методах, где проверяются экземпляры базы данных, все данные извлекаются из определенной таблицы с помощью метода view_BD (рисунок 36).

```
def view_BD(self, numb_of_table): #Получение всех данных из БД
    try:
        connection1 = sqlite3.connect('AllBD.db')
        cur = connection1.cursor()
        if numb_of_table == 1:
            m = cur.execute("""SELECT * FROM tUsers""")
            all_values = list(m.fetchall())
        elif numb_of_table == 2:
            m = cur.execute("""SELECT * FROM tCars""")
            all_values = list(m.fetchall())
        elif numb_of_table == 3:
            m = cur.execute("""SELECT * FROM tOrders""")
            all_values = list(m.fetchall())
        cur.close()
        return all_values
    except sqlite3.Error as error:
        print('Ошибка при получении данных из базы данных', error)
    finally:
        if (connection1):
            connection1.commit()
            connection1.close()
```

Рисунок 36. - Метод view_BD

За сортировку машин по грузоподъёмности отвечает метод sort_models_page (рисунок 37). Он выводит на экран специальную страницу с двумя списками: по убыванию и по возрастанию (риснуок 38)

```
def sort_models_page(self):
    sort_cars_frame = Frame(root, bg='blue')
    sort_cars_frame.place(relwidth=1, relheight=1)
    exit_btn = Button(sort_cars_frame, text='X', bg='red', fg='white', command=exit, font=200)
    exit_btn.place(relx=0.98, rely=0, height=30, width=40)
    acf_title = Label(sort_cars_frame, text='Автомобили по грузоподъемности', bg='red', font=30)
    acf_title.place(relx=0.4, rely=0.1, width=300)
    list_of_types_cars = ['ГАЗ-3302 «Газель»', 'ЗИЛ-5301 (Бычок)', 'MAN-10', 'Фура Mercedes-Benz Actros']
    vozr_title = Label(sort_cars_frame, text='По возрастанию', bg='blue', fg='white', font=30)
    vozr_title.place(relx=0.2, rely=0.3, width=250)
    ubiv_title = Label(sort_cars_frame, text='По убыванию', bg='blue', fg='white', font=30)
    ubiv_title.place(relx=0.6, rely=0.3, width=250)
    start_place = 0.4
    end_place = 0.55
    for i in range(len(list_of_types_cars)):
        left_title = Label(sort_cars_frame, text=list_of_types_cars[i], bg='blue', fg='white', font=30)
        left_title.place(relx=0.2, rely=start_place)
        start_place +=0.05
        right_title = Label(sort_cars_frame, text=list_of_types_cars[i], bg='blue', fg='white', font=30)
        right_title.place(relx=0.6, rely=end_place)
        end_place -= 0.05
    back_btn = Button(sort_cars_frame, text='Назад', bg='white',
                      command=lambda: (sort_cars_frame.destroy(), self.delete_by_time()))
    back_btn.place(relx=0.45, rely=0.6, width=150)
```

Рисунок 37. - Метод sort_models_page

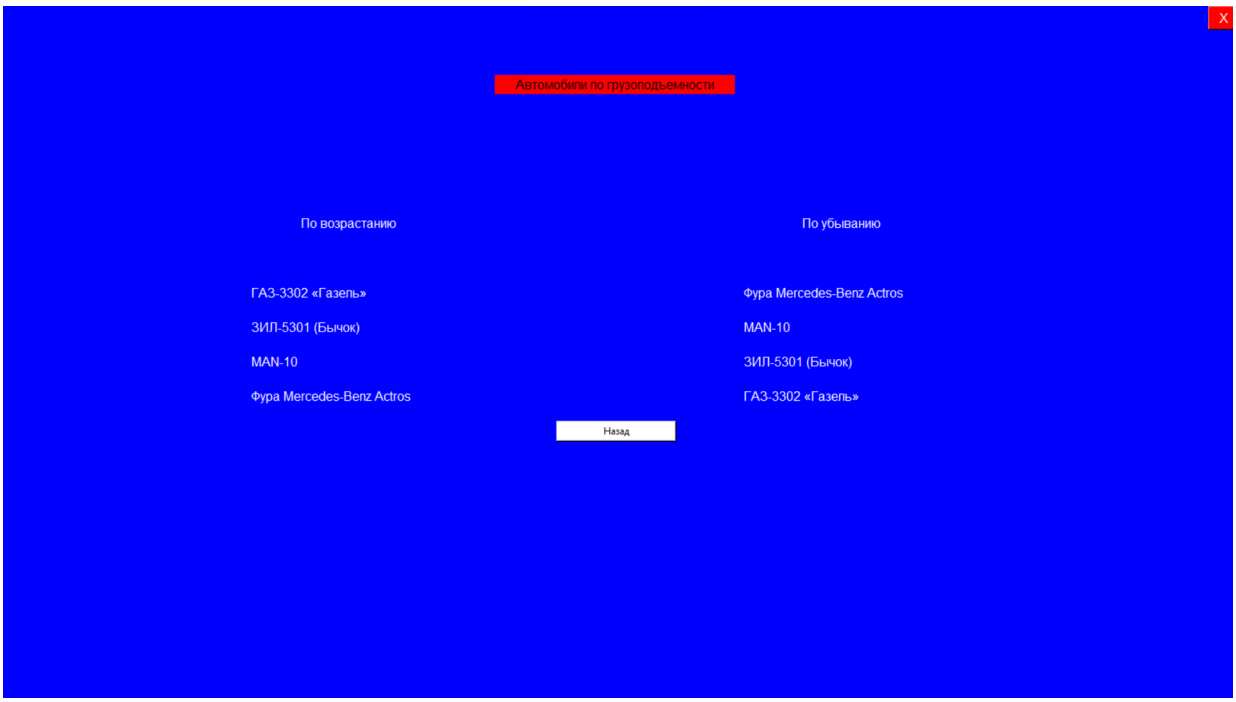

Рисунок 38. - Страница просмотра списка авомобилей по грузоподъёмности

Кнопка "Создать новую заявку на перевоз"вызывает метод new_offer_page (рисунок 39 и рисунок 40). Он открывает страницу, представленную на рисунке 41. По кнопке открывается новая страница с выбором автомобиля или уведомлением об ошибке (в случае некорректного ввода) (рисунок 42). За эти действия отвечает модуль pick_car_for_offer_page (рисунок 43 и рисунок 44). сохранение заказа происходит с помощью метода add_offer (Рисунок 45)

```
def new_offer_page(self):
    offer_frame = Frame(root, bg='blue')
    offer_frame.place(relwidth=1, relheight=1)
    exit_btn = Button(offer_frame, text='X', bg='red', fg='white', command=exit, font=200)
    exit_btn.place(relx=0.98, rely=0, height=30, width=40)
    offer_title = Label(offer_frame, text='Новый заказ', bg='red', font=30)
    offer_title.place(relx=0.4, rely=0.1, width=300)
    LeftCap_title = Label(offer_frame, text='Введите вес груза', bg='blue', font=30)
    LeftCap_title.place(relx=0.3, rely=0.2, width=300)
    offer_LeftCap = Entry(offer_frame, bg='white')
    offer_LeftCap.place(relx=0.55, rely=0.21)
    length_title = Label(offer_frame, text='Введите длину груза', bg='blue', font=30)
    length_title.place(relx=0.3, rely=0.26, width=300)
    offer_length = Entry(offer_frame, bg='white')
    offer_length.place(relx=0.55, rely=0.27)
    width_title = Label(offer_frame, text='Введите ширину груза', bg='blue', font=30)
    width_title.place(relx=0.3, rely=0.32, width=300)
    offer_width = Entry(offer_frame, bg='white')
    offer_width.place(relx=0.55, rely=0.33)
    height_title = Label(offer_frame, text='Введите высоту груза', bg='blue', font=30)
    height_title.place(relx=0.3, rely=0.38, width=300)
    offer_height = Entry(offer_frame, bg='white')
    offer_height.place(relx=0.55, rely=0.39)
    name_title = Label(offer_frame, text='Введите название груза', bg='blue', font=30)
    name_title.place(relx=0.3, rely=0.44, width=300)
    offer_name = Entry(offer_frame, bg='white')
    offer_name.place(relx=0.55, rely=0.45)
```

Рисунок 39. - Метод new_offer_page. Часть 1

```
date_title = Label(offer_frame, text='Введите срок доставки груза', bg='red', font=30)
date_title.place(relx=0.4, rely=0.1, width=300)
year_title = Label(offer_frame, text='Год:', bg='blue', fg='white', font=30)
year_title.place(relx=0.3, rely=0.5, width=300)
offer_date = Entry(offer_frame, bg='white')
offer_date.place(relx=0.55, rely=0.5)
month_title = Label(offer_frame, text='Месяц:', bg='blue', fg='white', font=30)
month_title.place(relx=0.3, rely=0.55, width=300)
offer_month = Entry(offer_frame, bg='white')
offer_month.place(relx=0.55, rely=0.55)
day_title = Label(offer_frame, text='День:', bg='blue', fg='white', font=30)
day_title.place(relx=0.3, rely=0.6, width=300)
offer_day = Entry(offer_frame, bg='white')
offer_day.place(relx=0.55, rely=0.6)
hour_title = Label(offer_frame, text='Час:', bg='blue', fg='white', font=30)
hour_title.place(relx=0.3, rely=0.65, width=300)
offer_hour = Entry(offer_frame, bg='white')
offer_hour.place(relx=0.55, rely=0.65)
minute_title = Label(offer_frame, text='Минуты:', bg='blue', fg='white', font=30)
minute_title.place(relx=0.3, rely=0.7, width=300)
offer_minute = Entry(offer_frame, bg='white')
offer_minute.place(relx=0.55, rely=0.7)
offer_button = Button(offer_frame, bg='white', text='Продолжить заказ', command=lambda: (self.pick_car_for_offer_page(offer_name.get(), offer_LeftCap.get(),
offer_length.get(),offer_width.get(),offer_height.get(), offer_date.get(), offer_month.get(), offer_day.get(),offer_hour.get(), offer_minute.get()), self.delete_by_time()))
offer_button.place(relx=0.4, rely=0.75, width=150)
back_btn = Button(offer_frame, text='Назад', bg='white',command=lambda: (offer_frame.destroy(), self.delete_by_time()))
back_btn.place(relx=0.4, rely=0.80, width=150)
```

Рисунок 40. - Метод new_offer_page. Часть 2

Рисунок 41. - Экран начала заказа

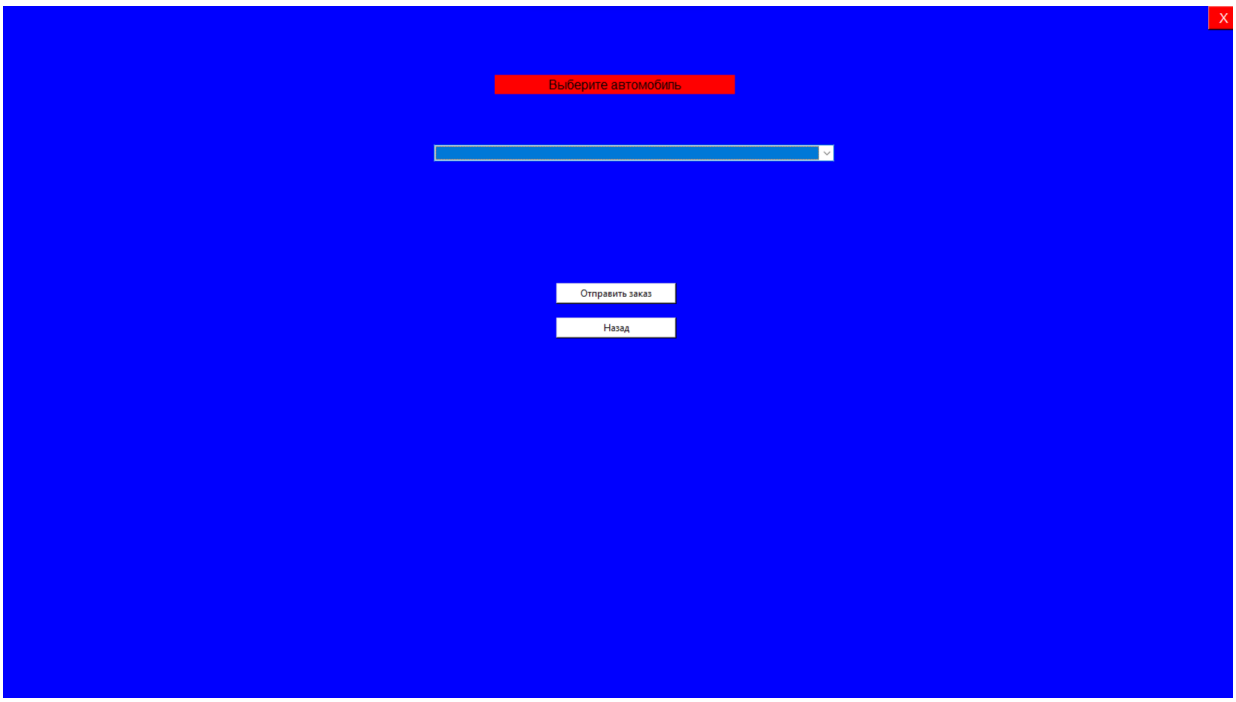

Рисунок 42. - Экран продолжения заказа

```
def BD_update(self, name, password, admin, frame): #Запись данных нового аккаунта
    try:
        if password.isnumeric() == False:
            raise Exceptions.IncorrectInputExcError
        user_list = self.view_BD(1)
        print(user_list)
        for i in user_list:
            if i[0] == name:
                raise Exceptions.DublicatenameError
        connection2 = sqlite3.connect('AllBD.db')
        cur = connection2.cursor()
        new_item = (name, int(password), admin)
        cur.execute("""INSERT INTO tUsers Values (?, ?, ?)""", new_item)
        cur.close()
        Succes_Label = Label(frame, text='Регистрация прошла успешно', bg='blue', fg='white', font=20, width=300)
        Succes_Label.place(relx=0.395, rely=0.22, width=300)
    except Exceptions.DublicatenameError:
        ErrorLabel = Label(frame, text='Данное имя уже занято', bg='blue', fg='red', font=20, width=300)
        ErrorLabel.place(relx=0.395, rely=0.22)
    except sqlite3.Error as error:
        print('Ошибка при подключение SQLite', error)
    except Exceptions.IncorrectInputExcError:
        ErrorLabel = Label(frame, text='Пароль должен содержать только цифры', bg='blue', fg='red', font=20, width=30)
        ErrorLabel.place(relx=0.395, rely=0.22, width=300)
    finally:
        if (connection2):
            connection2.commit()
            connection2.close()
```

Рисунок 43. - Метод pick_car_for_offer_page. Часть 1

```
def pick_car_for_offer_page(self, name_of_offer, LeftCap_offer, lenght_offer, width_offer, height_offer, date_offer, month_offer, day_offer,hour_offer, minute_offer):
    try:
        pick_car_frame = Frame(root, bg='blue')
        pick_car_frame.place(relwidth=1, relheight=1)

        exit_btn = Button(pick_car_frame, text='X', bg='red', fg='white', command=exit, font=200)
        exit_btn.place(relx=0.98, rely=0, height=30, width=40)

        pcf_title = Label(pick_car_frame, text='Выберите автомобиль', bg='red', font=30)
        pcf_title.place(relx=0.4, rely=0.1, width=300)
        good_cars = []
        list_of_cars_for_offer = self.view_BD(2)
        for i in range(len(list_of_cars_for_offer)):
            if list_of_cars_for_offer[i][-1] != 1:
                if int(LeftCap_offer) <= list_of_cars_for_offer[i][5] and int(lenght_offer) <= \
                        list_of_cars_for_offer[i][
                            2] and int(width_offer) <= list_of_cars_for_offer[i][3] and int(height_offer) <= \
                        list_of_cars_for_offer[i][4]:
                    good_cars.append([list_of_cars_for_offer[i][0], list_of_cars_for_offer[i][1]]) #модель и id

        back_btn = Button(pick_car_frame, text='Назад', bg='white',
                          command=lambda: (pick_car_frame.destroy(), self.delete_by_time()))
        back_btn.place(relx=0.45, rely=0.45, width=150)

        if int(LeftCap_offer) < 0 or int(lenght_offer) < 0 or int(width_offer) < 0 or int(height_offer) < 0:
            raise Exceptions.IncorrectcargoinfoError
```

Рисунок 44. - Метод pick_car_for_offer_page. Часть 2

Рисунок 45. - метод add_offer

Подбор машины по параметрам осуществляется с помощью метода parametres_page (рисунок 46). Он похож по своей структуре на метод new_offer_page, поэтому их страницы тоже схожи (рисунок 47). Однако здесь все реализованно в рамках одной страницы, поэтому исползуемой внутри кода метод place_good_cars (рисунок 48 и рисунок 49) не создает новую страницу, а продолжает заполнять уже существующую (рисунок 50). На странице предлагается выбрать доступный автомобиль подходящей модели, а также забронировать его. Бронирование происходит благодаря методу add_offer, как и в случае с обычным заказом.

Рисунок 46. - Метод parametres_page

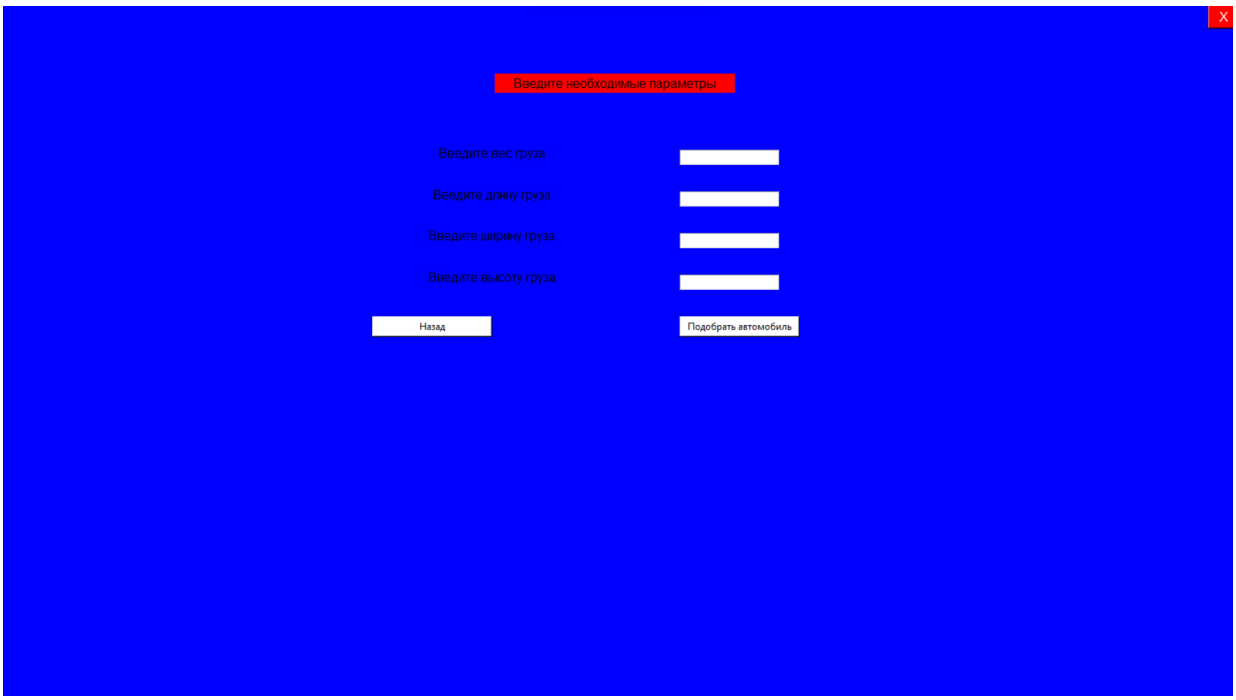

Рисунок 47. - Подбор параметров

```
def place_good_cars(self, frame, offer_LeftCap, offer_length, offer_width, offer_height):
    try:
        list_of_cars_for_check = self.view_BD(2)
        best_parametres = [10000000000000, 10000000000000, 10000000000000, 10000000000000,
                           'FFF']  # длина ширина высота тяжесть
        for i in range(len(list_of_cars_for_check)):
            if list_of_cars_for_check[i][-1] != 1:
                if offer_LeftCap <= list_of_cars_for_check[i][5] and offer_length <= list_of_cars_for_check[i][
                    2] and offer_width <= list_of_cars_for_check[i][3] \
                        and offer_height <= list_of_cars_for_check[i][4]:
                    if best_parametres[0] >= list_of_cars_for_check[i][2] and best_parametres[1] >= \
                            list_of_cars_for_check[i][3] \
                            and best_parametres[2] >= list_of_cars_for_check[i][4] and best_parametres[3] >= \
                            list_of_cars_for_check[i][5]:
                        best_parametres = [list_of_cars_for_check[i][2], list_of_cars_for_check[i][3],
                                           list_of_cars_for_check[i][4], list_of_cars_for_check[i][5],
                                           list_of_cars_for_check[i][0]]

        if best_parametres[-1] == 'FFF':
            havent_label = Label(frame, text='нет подходящих машин', bg='blue', fg='red')
            havent_label.place(relx=0.32, rely=0.5, width=500)
        else:
            if int(offer_LeftCap) < 0 or int(offer_length) < 0 or int(offer_width) < 0 or int(offer_height) < 0:
                raise Exceptions.IncorrectcargoinfoError
```

Рисунок 48. - Метод place_good_cars. Часть 1

```
    var3 = StringVar()
    good_title = Label(frame, text=f'Вам подходит автомобиль модели {best_parametres[-1]}')
    good_title.place(relx=0.32, rely=0.5, width=500)
    good_cars = []
    for i in range(len(list_of_cars_for_check)):
        if best_parametres[-1] == list_of_cars_for_check[i][0] and list_of_cars_for_check[i][-1] != 1:
            good_cars.append([list_of_cars_for_check[i][0], list_of_cars_for_check[i][1]])
    combobox = ttk.Combobox(frame, textvariable=var3)
    combobox['values'] = good_cars
    combobox['state'] = 'readonly'
    combobox.place(relx=0.32, rely=0.55, width=500)
    my_date = str(datetime.datetime.now()+datetime.timedelta(days=3))
    date_str, time_str = my_date.split(' ')[0], my_date.split(' ')[1]
    fdate = date_str.split('-')
    ftime = time_str.split(':')
    param_button = Button(frame, text='Выбрать', bg='white', command=lambda: (self.add_offer(self.name, combobox.get(), 'Бронирование',
                                                                               fdate[0], fdate[1], fdate[2], ftime[0], ftime[1], frame), self.delete_by_time()))
    param_button.place(relx=0.43, rely=0.6, width=150)
```

Рисунок 49. - Метод place_good_cars. Часть 2

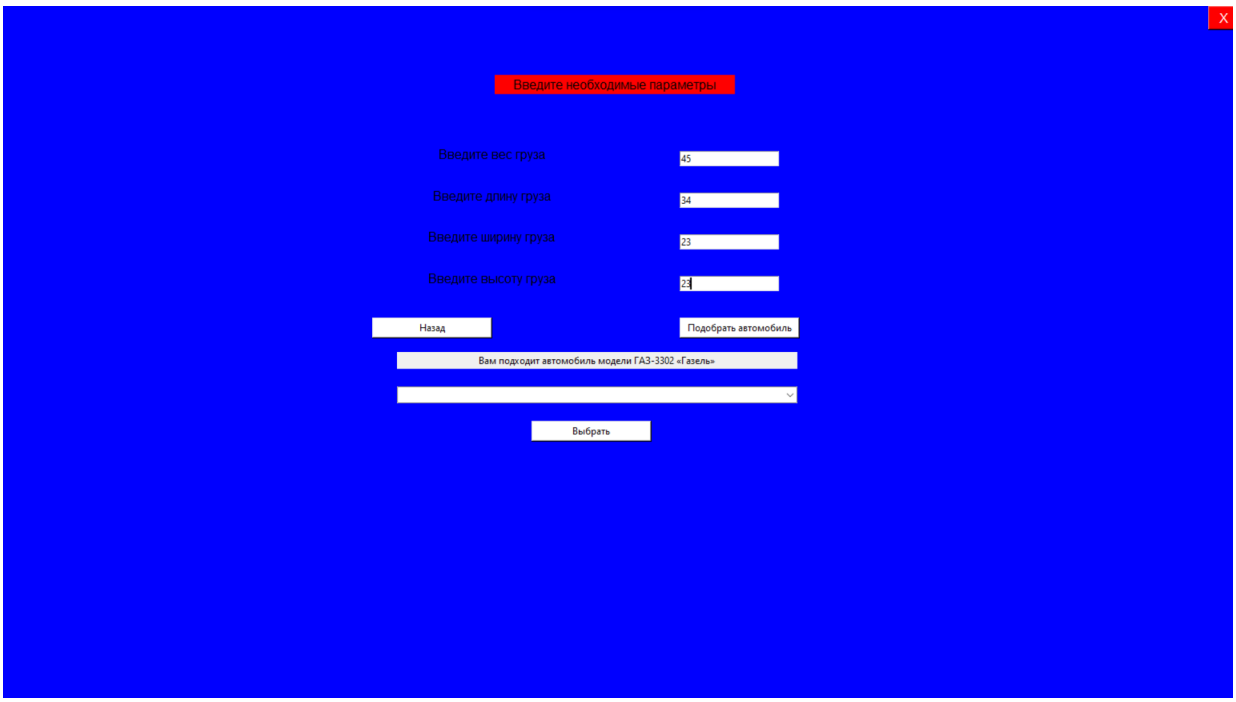

Рисунок 50. - Выбор машины под указанные параметры

В главном меню кнопка "Выйти из аккаунта"возвращает пользователя на страницу входа/регистрации, а кнопка "Выйти из системы"завершает работу программы.

ЗАКЛЮЧЕНИЕ

В данной домашней работе было реализовано программное обеспечение для учета грузового транспорта. Оно реализовано с помощью ООП, имеет возможноть сохранения данных в базу данных и имеет графический интерфейс. Также программное обеспечение справляется со всеми поставленными задачами.