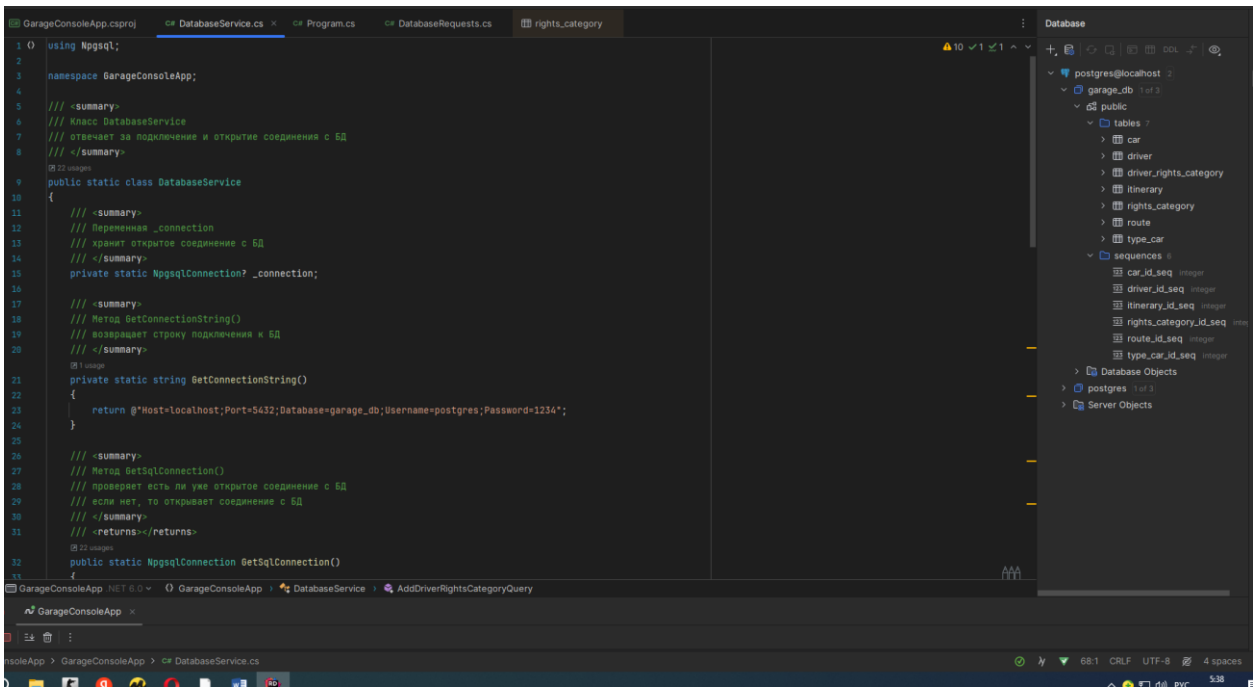
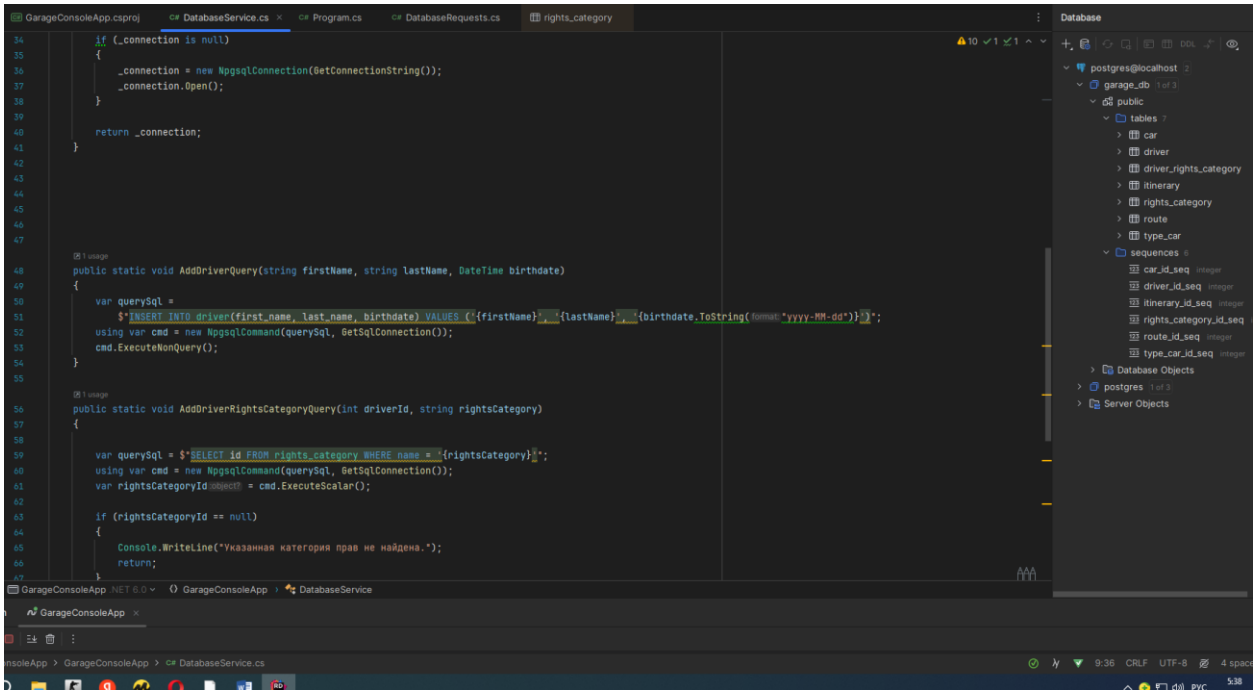
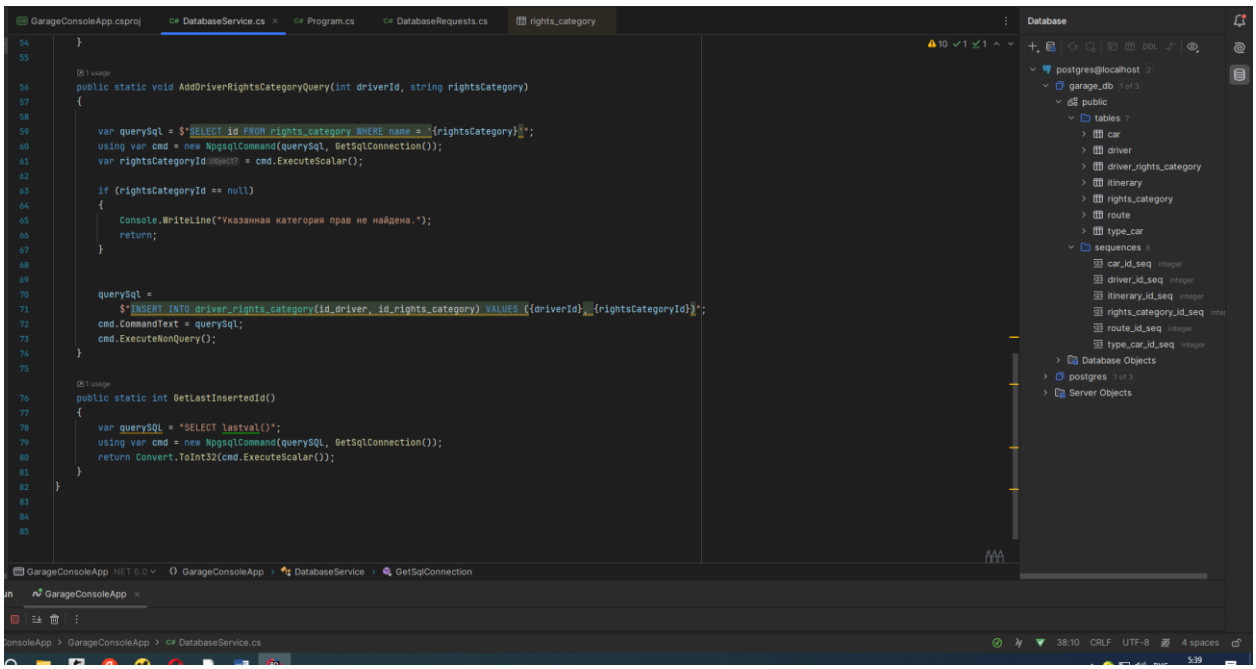
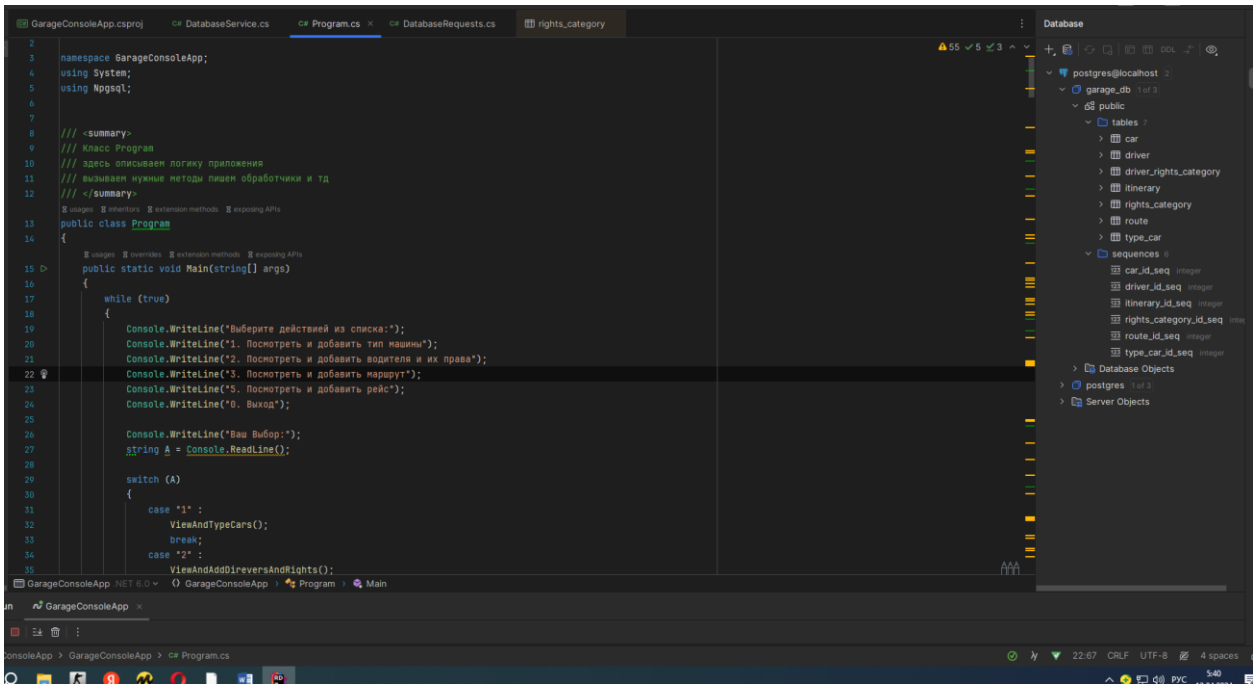

GarageConsoleApp.csproj
DatabaseService.cs
Program.cs
DatabaseRequests.cs
rights_category
Database
using Npgsql;
namespace GarageConsoleApp;
/// <summary>
/// Класс DatabaseService
/// отвечает за подключение и открытие соединения с БД
/// </summary>
22 usages
public static class DatabaseService
{
/// <summary>
/// Переменная _connection
/// хранит открытое соединение с БД
/// </summary>
private static NpgsqlConnection? _connection;
/// <summary>
/// Метод GetConnectionString()
/// возвращает строку подключения к БД
/// </summary>
1 usage
private static string GetConnectionString()
{
return @"Host=localhost;Port=5432;Database=garage_db;Username=postgres;Password=1234";
}
/// <summary>
/// Метод GetSqlConnection()
/// проверяет есть ли уже открытое соединение с БД
/// если нет, то открывает соединение с БД
/// </summary>
/// <returns></returns>
22 usages
public static NpgsqlConnection GetSqlConnection()
postgres@localhost
garage_db
public
tables
car
driver
driver_rights_category
itinerary
rights_category
route
type_car
sequences
car_id_seq
driver_id_seq
itinerary_id_seq
rights_category_id_seq
route_id_seq
type_car_id_seq
Database Objects
postgres
Server Objects
GarageConsoleApp
68:1
CRLF
UTF-8
4 spaces

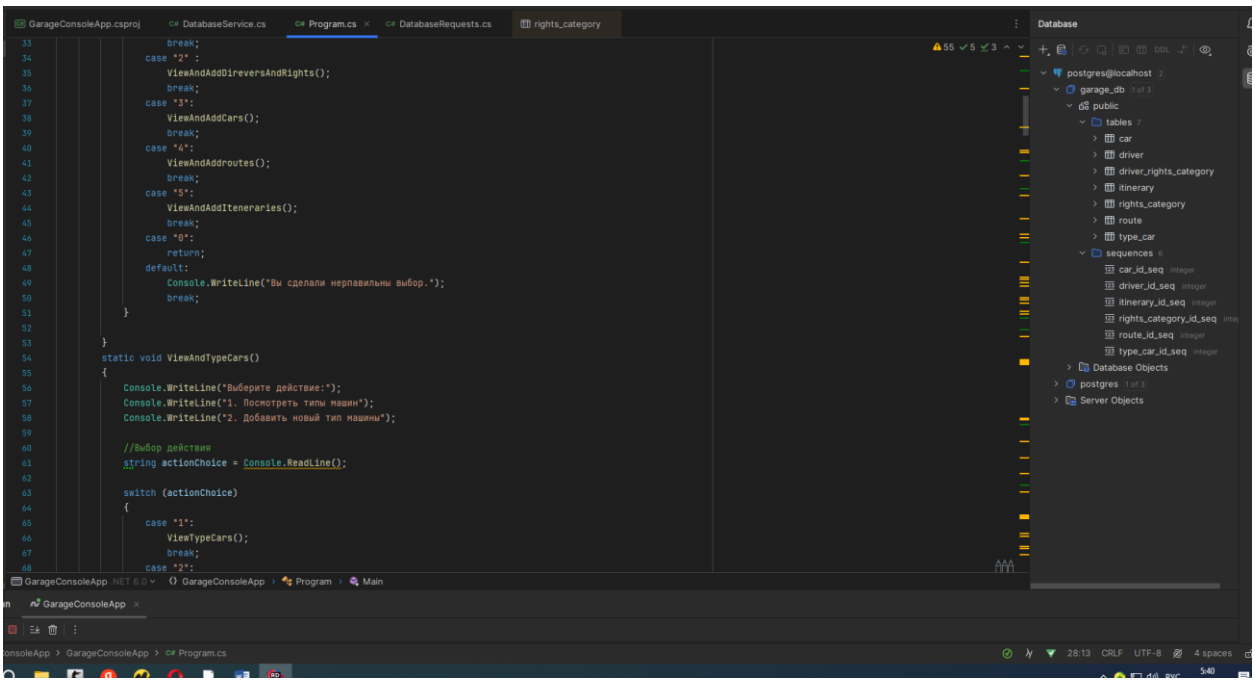

GarageConsoleApp.csproj
DatabaseService.cs
Program.cs
DatabaseRequests.cs
rights_category
Database
if (_connection is null)
{
_connection = new NpgsqlConnection(GetConnectionString());
_connection.Open();
}
return _connection;
}
1 usage
public static void AddDriverQuery(string firstName, string lastName, DateTime birthdate)
{
var querySql =
$"INSERT INTO driver(first_name, last_name, birthdate) VALUES ('{firstName}', '{lastName}', '{birthdate.ToString(format: "yyyy-MM-dd")}')";
using var cmd = new NpgsqlCommand(querySql, GetSqlConnection());
cmd.ExecuteNonQuery();
}
1 usage
public static void AddDriverRightsCategoryQuery(int driverId, string rightsCategory)
{
var querySql = $"SELECT id FROM rights_category WHERE name = '{rightsCategory}'";
using var cmd = new NpgsqlCommand(querySql, GetSqlConnection());
var rightsCategoryId = cmd.ExecuteScalar();
if (rightsCategoryId == null)
{
Console.WriteLine("Указанная категория прав не найдена.");
return;
}
postgres@localhost
garage_db
public
tables
car
driver
driver_rights_category
itinerary
rights_category
route
type_car
sequences
car_id_seq
driver_id_seq
itinerary_id_seq
rights_category_id_seq
route_id_seq
type_car_id_seq
Database Objects
postgres
Server Objects
GarageConsoleApp
9:36
CRLF
UTF-8

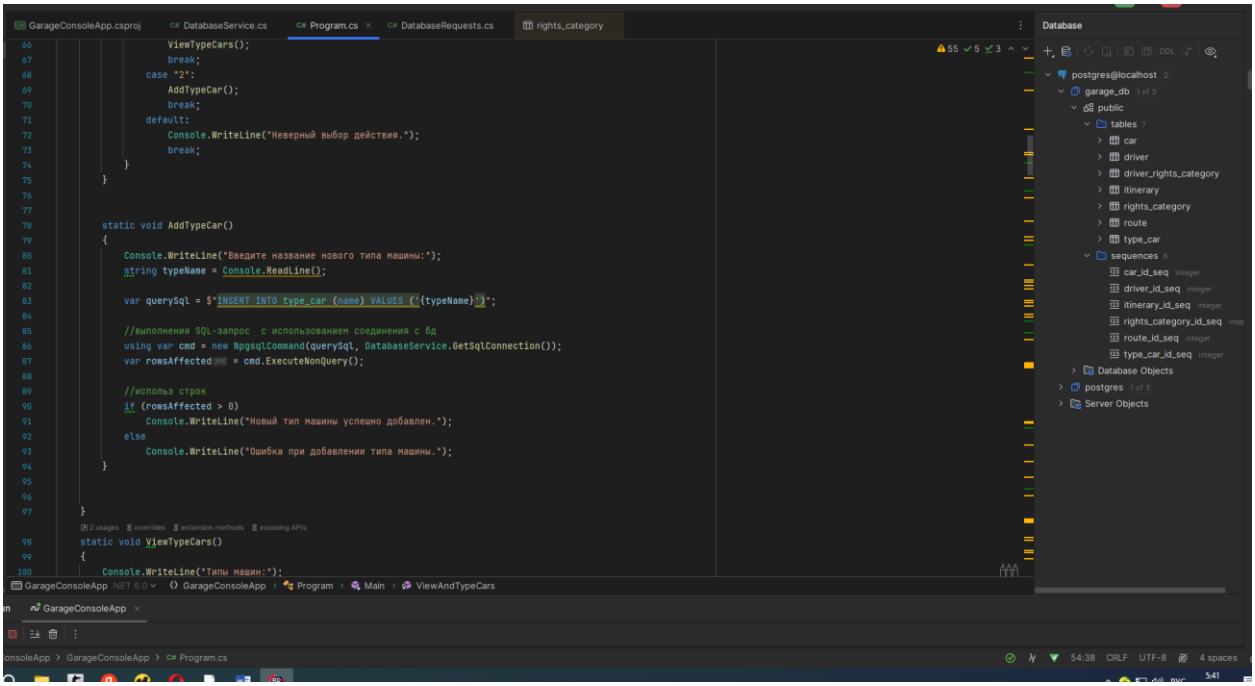


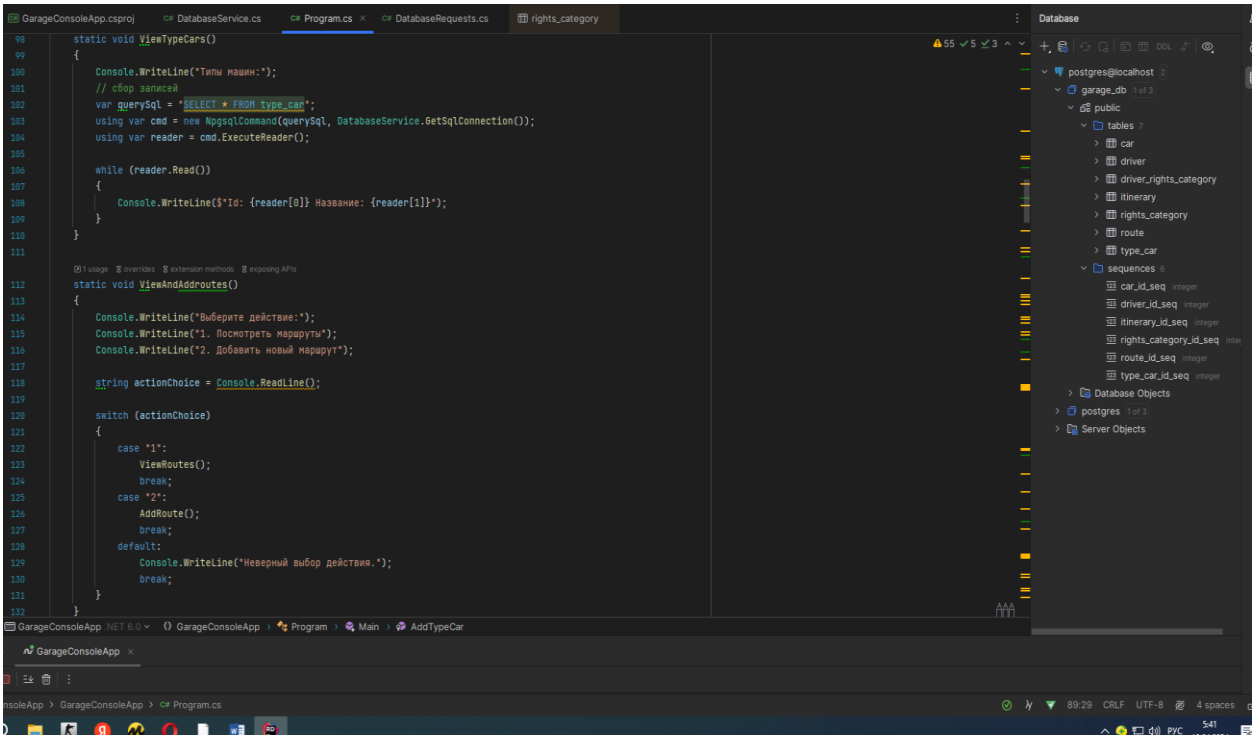
```
    }

    1 usage
    public static void AddDriverRightsCategoryQuery(int driverId, string rightsCategory)
    {

        var querySql = $"SELECT id FROM rights_category WHERE name = '{rightsCategory}'";
        using var cmd = new NpgsqlCommand(querySql, GetSqlConnection());
        var rightsCategoryId object? = cmd.ExecuteScalar();

        if (rightsCategoryId == null)
        {
            Console.WriteLine("Указанная категория прав не найдена.");
            return;
        }


        querySql =
            $"INSERT INTO driver_rights_category(id_driver, id_rights_category) VALUES ({driverId},{rightsCategoryId})";
        cmd.CommandText = querySql;
        cmd.ExecuteNonQuery();
    }

    1 usage
    public static int GetLastInsertedId()
    {
        var querySQL = "SELECT lastval()";
        using var cmd = new NpgsqlCommand(querySQL, GetSqlConnection());
        return Convert.ToInt32(cmd.ExecuteScalar());
    }
}
```



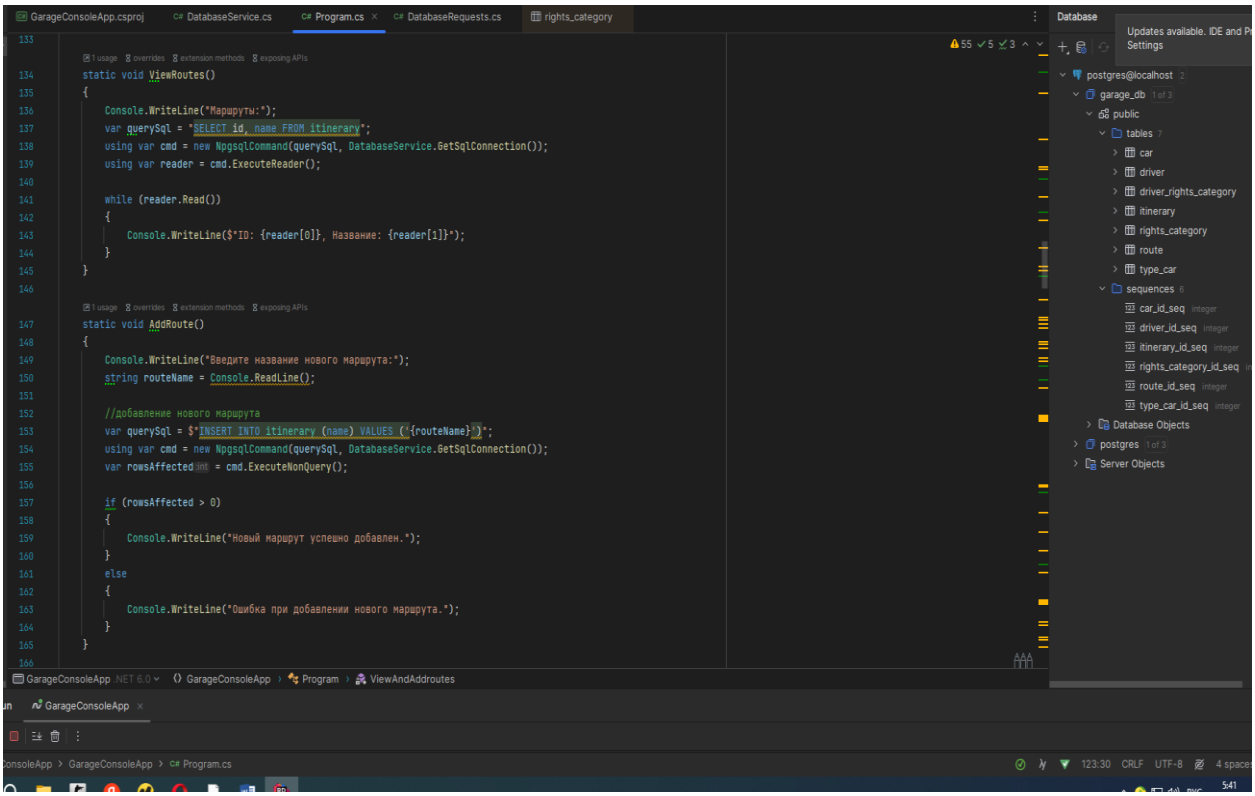
```
namespace GarageConsoleApp;
using System;
using Npgsql;


/// <summary>
/// Класс Program
/// здесь описываем логику приложения
/// вызываем нужные методы пишем обработчики и тд
/// </summary>
usages  inheritors  extension methods  exposing APIs
public class Program
{
    usages  overrides  extension methods  exposing APIs
    public static void Main(string[] args)
    {
        while (true)
        {
            Console.WriteLine("Выберите действиев из списка:");
            Console.WriteLine("1. Посмотреть и добавить тип машины");
            Console.WriteLine("2. Посмотреть и добавить водителя и их права");
            Console.WriteLine("3. Посмотреть и добавить маршрут");
            Console.WriteLine("5. Посмотреть и добавить рейс");
            Console.WriteLine("0. Выход");

            Console.WriteLine("Ваш Выбор:");
            string A = Console.ReadLine();

            switch (A)
            {
                case "1" :
                    ViewAndTypeCars();
                    break;
                case "2" :
                    ViewAndAddDireversAndRights();
```


GarageConsoleApp.csproj
DatabaseService.cs
Program.cs
DatabaseRequests.cs
rights_category
Database
break;
case "2" :
ViewAndAddDireversAndRights();
break;
case "3":
ViewAndAddCars();
break;
case "4":
ViewAndAddroutes();
break;
case "5":
ViewAndAddIteneraries();
break;
case "0":
return;
default:
Console.WriteLine("Вы сделали нерпавильны выбор.");
break;
static void ViewAndTypeCars()
Console.WriteLine("Выберите действие:");
Console.WriteLine("1. Посмотреть типы машин");
Console.WriteLine("2. Добавить новый тип машины");
//Выбор действия
string actionChoice = Console.ReadLine();
switch (actionChoice)
case "1":
ViewTypeCars();
case "2":
postgres@localhost
garage_db
public
tables
car
driver
driver_rights_category
itinerary
rights_category
route
type_car
sequences
car_id_seq
driver_id_seq
itinerary_id_seq
rights_category_id_seq
route_id_seq
type_car_id_seq
Database Objects
postgres
Server Objects
GarageConsoleApp
Program
Main
28:13
CRLF
UTF-8
4 spaces

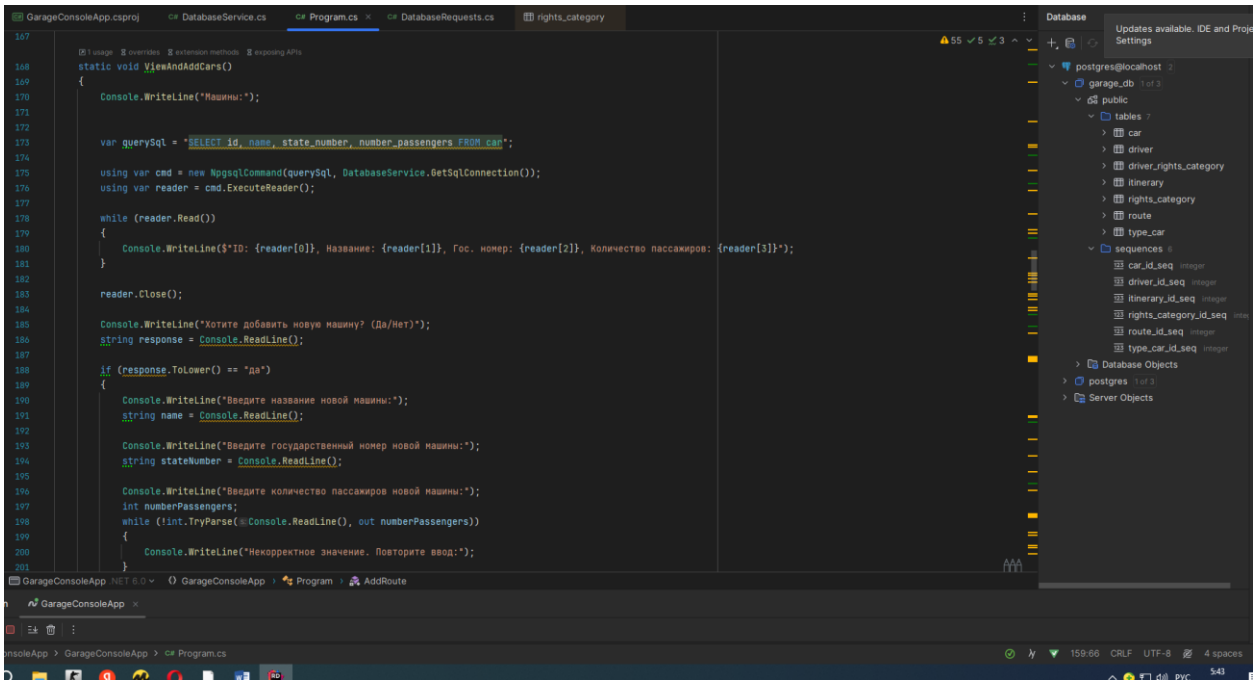

GarageConsoleApp.csproj
DatabaseService.cs
Program.cs
DatabaseRequests.cs
rights_category
Database
ViewTypeCars();
case "2":
AddTypeCar();
default:
Console.WriteLine("Неверный выбор действия.");
static void AddTypeCar()
Console.WriteLine("Введите название нового типа машины:");
string typeName = Console.ReadLine();
var querySql = $"INSERT INTO type_car (name) VALUES ('{typeName}')";
//выполнения SQL-запрос с использованием соединения с бд
using var cmd = new NpgsqlCommand(querySql, DatabaseService.GetSqlConnection());
var rowsAffected = cmd.ExecuteNonQuery();
//использ строк
if (rowsAffected > 0)
Console.WriteLine("Новый тип машины успешно добавлен.");
else
Console.WriteLine("Ошибка при добавлении типа машины.");
2 usages overrides extension methods exposing APIs
static void ViewTypeCars()
Console.WriteLine("Типы машин:");
postgres@localhost
garage_db
public
tables
car
driver
driver_rights_category
itinerary
rights_category
route
type_car
sequences
car_id_seq
driver_id_seq
itinerary_id_seq
rights_category_id_seq
route_id_seq
type_car_id_seq
Database Objects
postgres
Server Objects
GarageConsoleApp
Program
Main
ViewAndTypeCars
54:38
CRLF
UTF-8
4 spaces



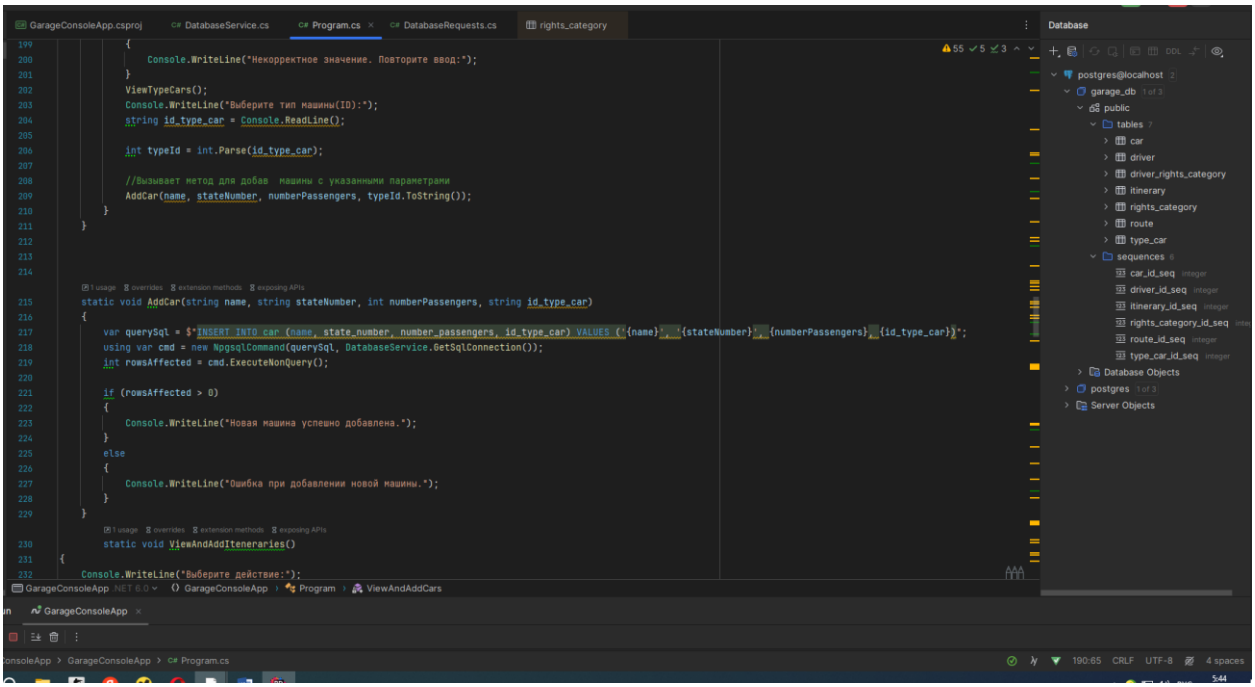
```
static void ViewTypeCars()
{
    Console.WriteLine("Типы машин:");
    // сбор записей
    var querySql = "SELECT * FROM type_car";
    using var cmd = new NpgsqlCommand(querySql, DatabaseService.GetSqlConnection());
    using var reader = cmd.ExecuteReader();

    while (reader.Read())
    {
        Console.WriteLine($"Id: {reader[0]} Название: {reader[1]}");
    }
}

1 usage  overrides  extension methods  exposing APIs
static void ViewAndAddroutes()
{
    Console.WriteLine("Выберите действие:");
    Console.WriteLine("1. Посмотреть маршруты");
    Console.WriteLine("2. Добавить новый маршрут");

    string actionChoice = Console.ReadLine();

    switch (actionChoice)
    {
        case "1":
            ViewRoutes();
            break;
        case "2":
            AddRoute();
            break;
        default:
            Console.WriteLine("Неверный выбор действия.");
            break;
    }
}
```



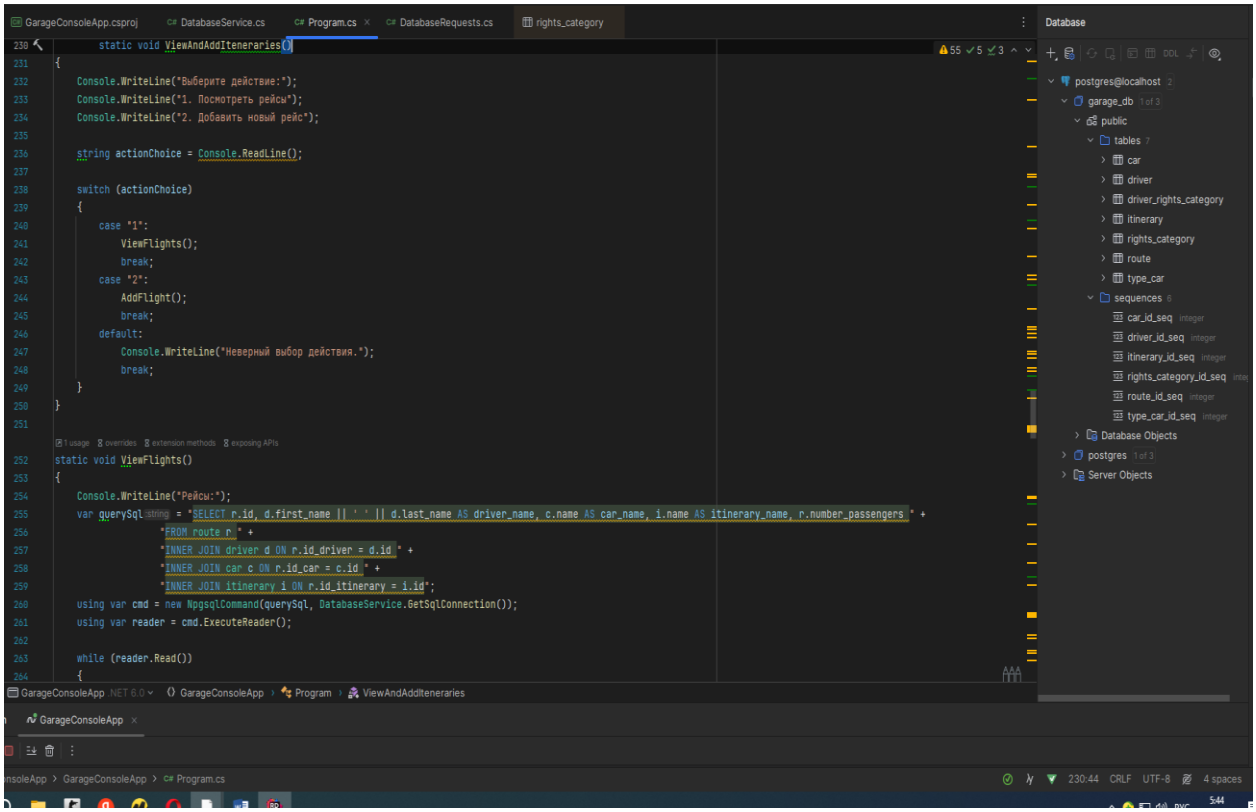
```
1 usage  overrides  extension methods  exposing APIs
static void ViewRoutes()
{
    Console.WriteLine("Маршруты:");
    var querySql = "SELECT id, name FROM itinerary";
    using var cmd = new NpgsqlCommand(querySql, DatabaseService.GetSqlConnection());
    using var reader = cmd.ExecuteReader();

    while (reader.Read())
    {
        Console.WriteLine($"ID: {reader[0]}, Название: {reader[1]}");
    }
}

1 usage  overrides  extension methods  exposing APIs
static void AddRoute()
{
    Console.WriteLine("Введите название нового маршрута:");
    string routeName = Console.ReadLine();

    //добавление нового маршрута
    var querySql = $"INSERT INTO itinerary (name) VALUES ('{routeName}')";
    using var cmd = new NpgsqlCommand(querySql, DatabaseService.GetSqlConnection());
    var rowsAffected = cmd.ExecuteNonQuery();

    if (rowsAffected > 0)
    {
        Console.WriteLine("Новый маршрут успешно добавлен.");
    }
    else
    {
        Console.WriteLine("Ошибка при добавлении нового маршрута.");
    }
}
```



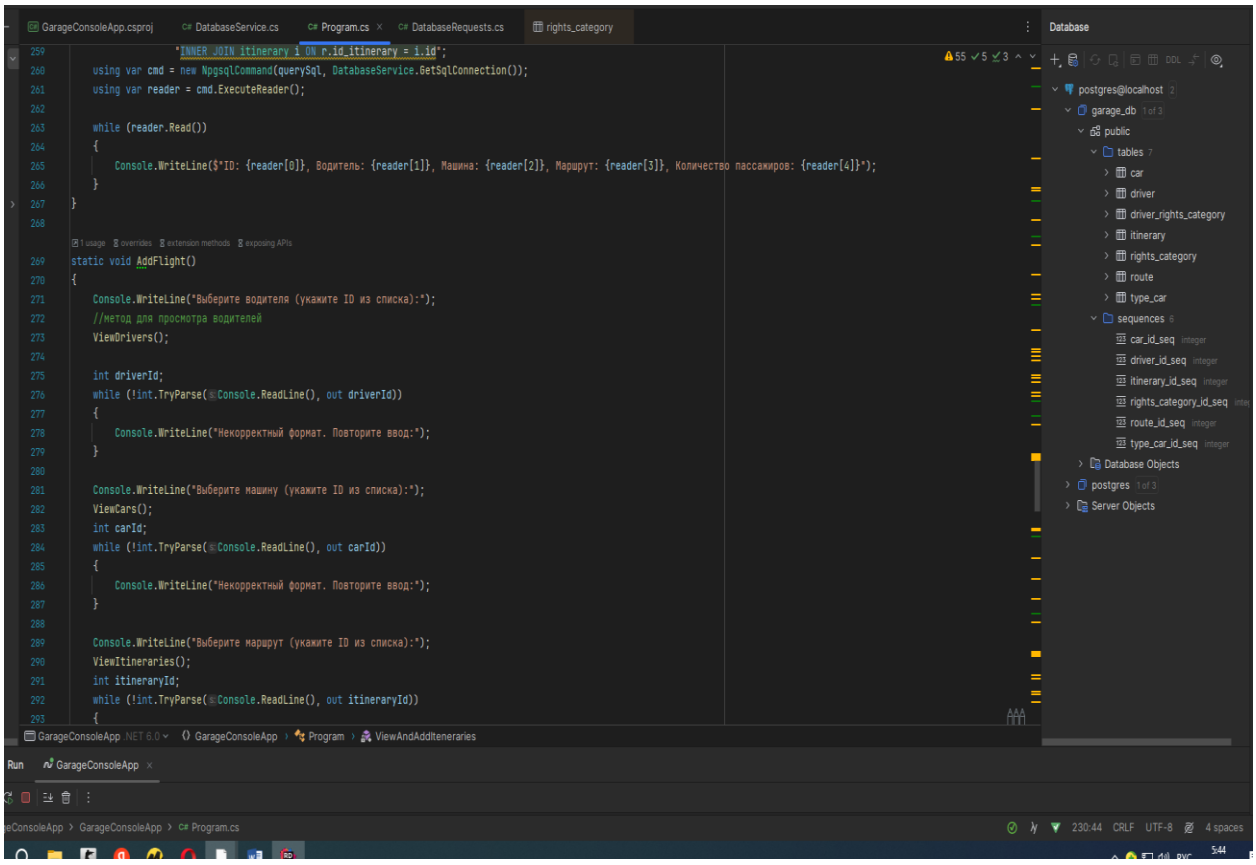
```
static void ViewAndAddCars()
{
    Console.WriteLine("Машины:");


    var querySql = "SELECT id, name, state_number, number_passengers FROM car";

    using var cmd = new NpgsqlCommand(querySql, DatabaseService.GetSqlConnection());
    using var reader = cmd.ExecuteReader();

    while (reader.Read())
    {
        Console.WriteLine($"ID: {reader[0]}, Название: {reader[1]}, Гос. номер: {reader[2]}, Количество пассажиров: {reader[3]}");
    }

    reader.Close();

    Console.WriteLine("Хотите добавить новую машину? (Да/Нет)");
    string response = Console.ReadLine();

    if (response.ToLower() == "да")
    {
        Console.WriteLine("Введите название новой машины:");
        string name = Console.ReadLine();

        Console.WriteLine("Введите государственный номер новой машины:");
        string stateNumber = Console.ReadLine();

        Console.WriteLine("Введите количество пассажиров новой машины:");
        int numberPassengers;
        while (!int.TryParse(Console.ReadLine(), out numberPassengers))
        {
            Console.WriteLine("Некорректное значение. Повторите ввод:");
        }
```



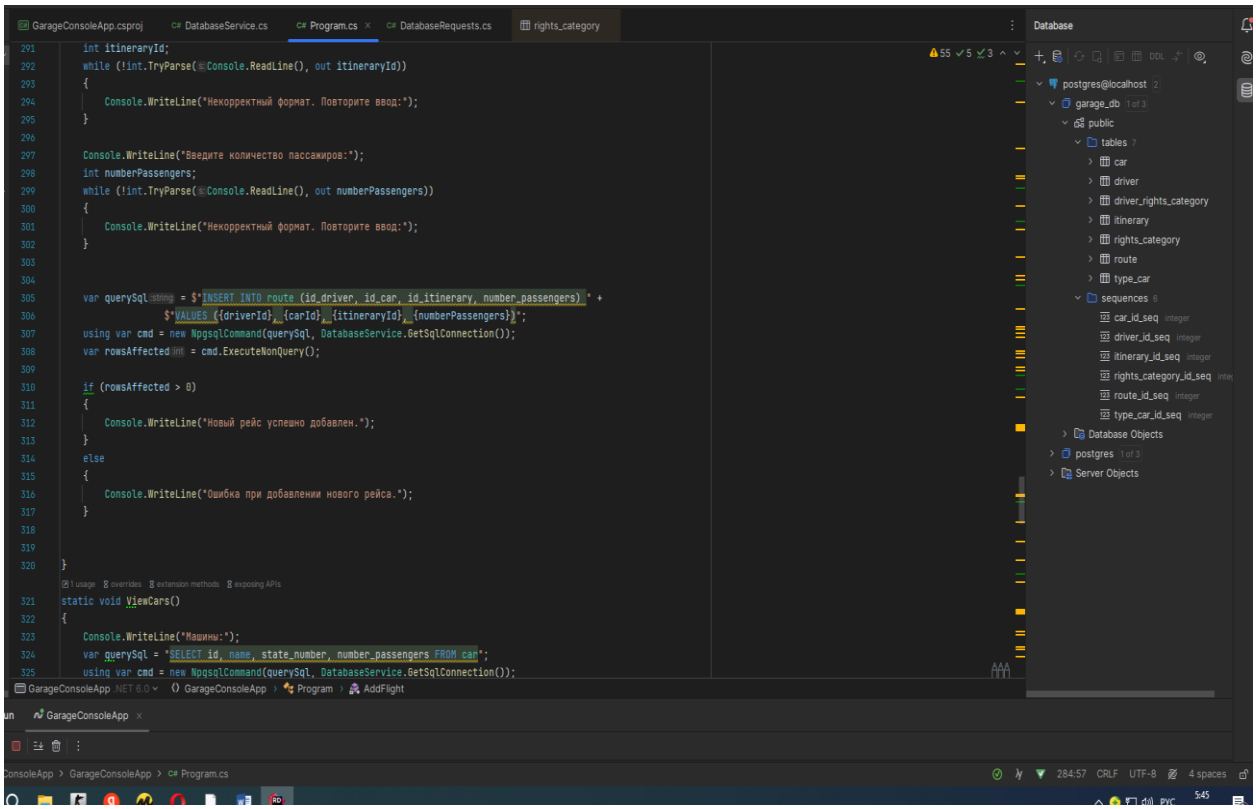
```
            {
                Console.WriteLine("Некорректное значение. Повторите ввод:");
            }
            ViewTypeCars();
            Console.WriteLine("Выберите тип машины(ID):");
            string id_type_car = Console.ReadLine();

            int typeId = int.Parse(id_type_car);

            //Вызывает метод для добав  машины с указанными параметрами
            AddCar(name, stateNumber, numberPassengers, typeId.ToString());
        }
    }


    static void AddCar(string name, string stateNumber, int numberPassengers, string id_type_car)
    {
        var querySql = $"INSERT INTO car (name, state_number, number_passengers, id_type_car) VALUES ('{name}', '{stateNumber}', {numberPassengers}, {id_type_car})";
        using var cmd = new NpgsqlCommand(querySql, DatabaseService.GetSqlConnection());
        int rowsAffected = cmd.ExecuteNonQuery();

        if (rowsAffected > 0)
        {
            Console.WriteLine("Новая машина успешно добавлена.");
        }
        else
        {
            Console.WriteLine("Ошибка при добавлении новой машины.");
        }
    }
        static void ViewAndAddIteneraries()
{
    Console.WriteLine("Выберите действие:");
```



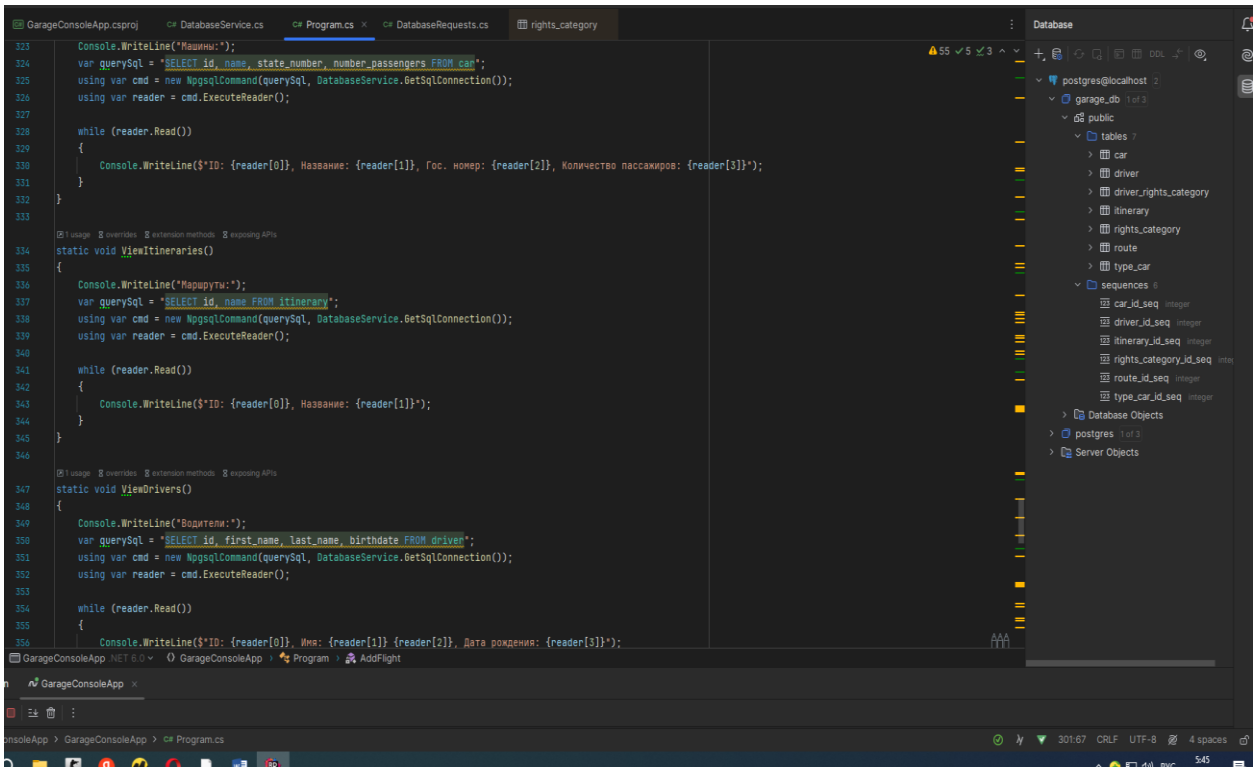
```
        static void ViewAndAddIteneraries()
{
    Console.WriteLine("Выберите действие:");
    Console.WriteLine("1. Посмотреть рейсы");
    Console.WriteLine("2. Добавить новый рейс");

    string actionChoice = Console.ReadLine();

    switch (actionChoice)
    {
        case "1":
            ViewFlights();
            break;
        case "2":
            AddFlight();
            break;
        default:
            Console.WriteLine("Неверный выбор действия.");
            break;
    }
}

1 usage  overrides  extension methods  exposing APIs
static void ViewFlights()
{
    Console.WriteLine("Рейсы:");
    var querySql = "SELECT r.id, d.first_name || ' ' || d.last_name AS driver_name, c.name AS car_name, i.name AS itinerary_name, r.number_passengers " +
                   "FROM route r " +
                   "INNER JOIN driver d ON r.id_driver = d.id " +
                   "INNER JOIN car c ON r.id_car = c.id " +
                   "INNER JOIN itinerary i ON r.id_itinerary = i.id";
    using var cmd = new NpgsqlCommand(querySql, DatabaseService.GetSqlConnection());
    using var reader = cmd.ExecuteReader();

    while (reader.Read())
    {
```



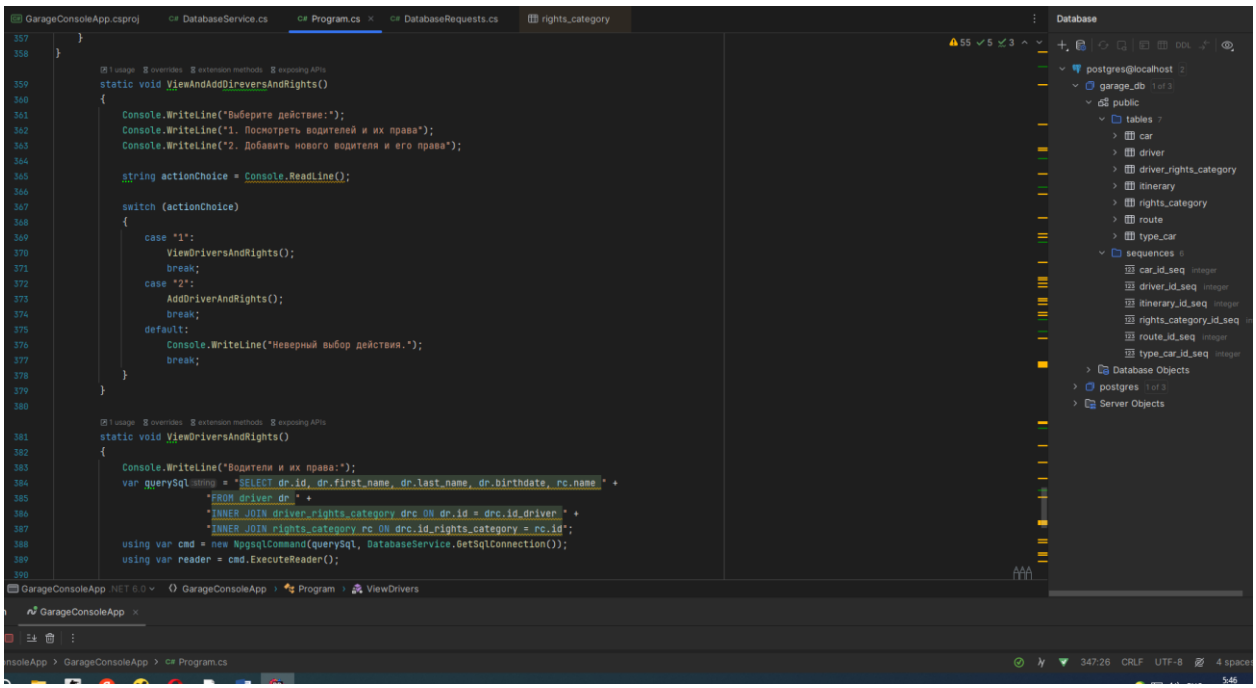
```
                   "INNER JOIN itinerary i ON r.id_itinerary = i.id";
    using var cmd = new NpgsqlCommand(querySql, DatabaseService.GetSqlConnection());
    using var reader = cmd.ExecuteReader();

    while (reader.Read())
    {
        Console.WriteLine($"ID: {reader[0]}, Водитель: {reader[1]}, Машина: {reader[2]}, Маршрут: {reader[3]}, Количество пассажиров: {reader[4]}");
    }
}

1 usage  overrides  extension methods  exposing APIs
static void AddFlight()
{
    Console.WriteLine("Выберите водителя (укажите ID из списка):");
    //метод для просмотра водителей
    ViewDrivers();

    int driverId;
    while (!int.TryParse(Console.ReadLine(), out driverId))
    {
        Console.WriteLine("Некорректный формат. Повторите ввод:");
    }

    Console.WriteLine("Выберите машину (укажите ID из списка):");
    ViewCars();
    int carId;
    while (!int.TryParse(Console.ReadLine(), out carId))
    {
        Console.WriteLine("Некорректный формат. Повторите ввод:");
    }

    Console.WriteLine("Выберите маршрут (укажите ID из списка):");
    ViewItineraries();
    int itineraryId;
    while (!int.TryParse(Console.ReadLine(), out itineraryId))
    {
```



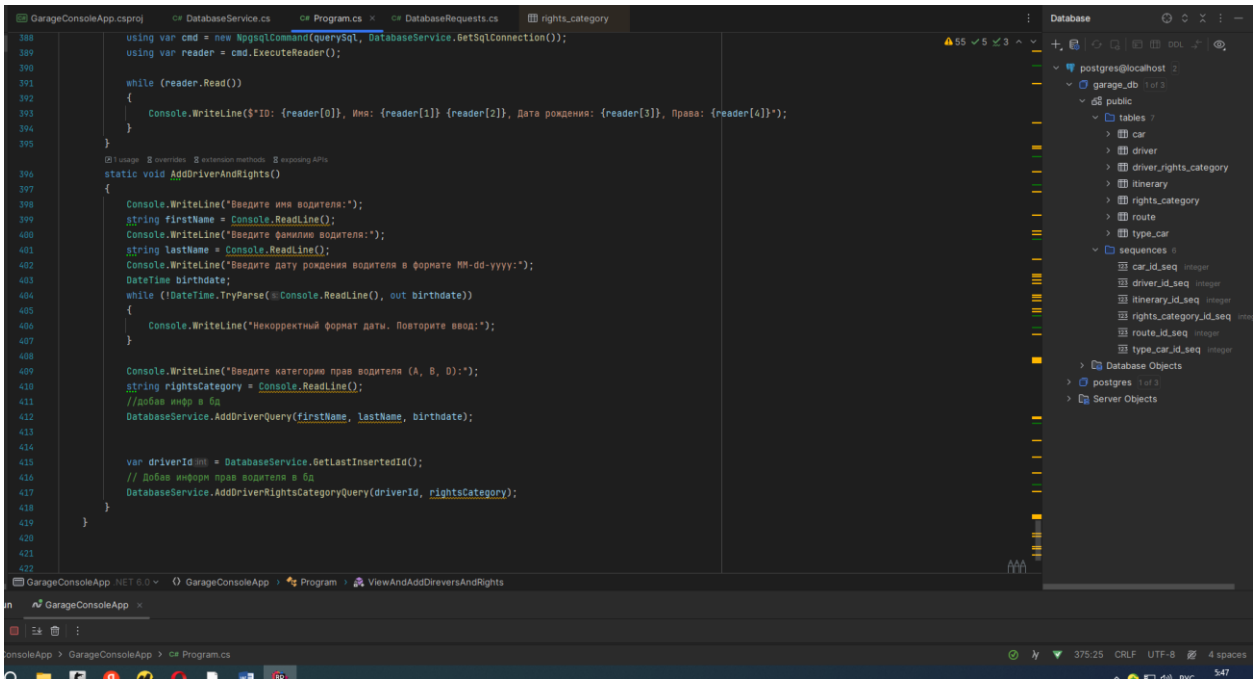
```
    int itineraryId;
    while (!int.TryParse(Console.ReadLine(), out itineraryId))
    {
        Console.WriteLine("Некорректный формат. Повторите ввод:");
    }

    Console.WriteLine("Введите количество пассажиров:");
    int numberPassengers;
    while (!int.TryParse(Console.ReadLine(), out numberPassengers))
    {
        Console.WriteLine("Некорректный формат. Повторите ввод:");
    }


    var querySql = $"INSERT INTO route (id_driver, id_car, id_itinerary, number_passengers) " +
                   $"VALUES ({driverId}, {carId}, {itineraryId}, {numberPassengers})";
    using var cmd = new NpgsqlCommand(querySql, DatabaseService.GetSqlConnection());
    var rowsAffected = cmd.ExecuteNonQuery();

    if (rowsAffected > 0)
    {
        Console.WriteLine("Новый рейс успешно добавлен.");
    }
    else
    {
        Console.WriteLine("Ошибка при добавлении нового рейса.");
    }


}
static void ViewCars()
{
    Console.WriteLine("Машины:");
    var querySql = "SELECT id, name, state_number, number_passengers FROM car";
    using var cmd = new NpgsqlCommand(querySql, DatabaseService.GetSqlConnection());
```



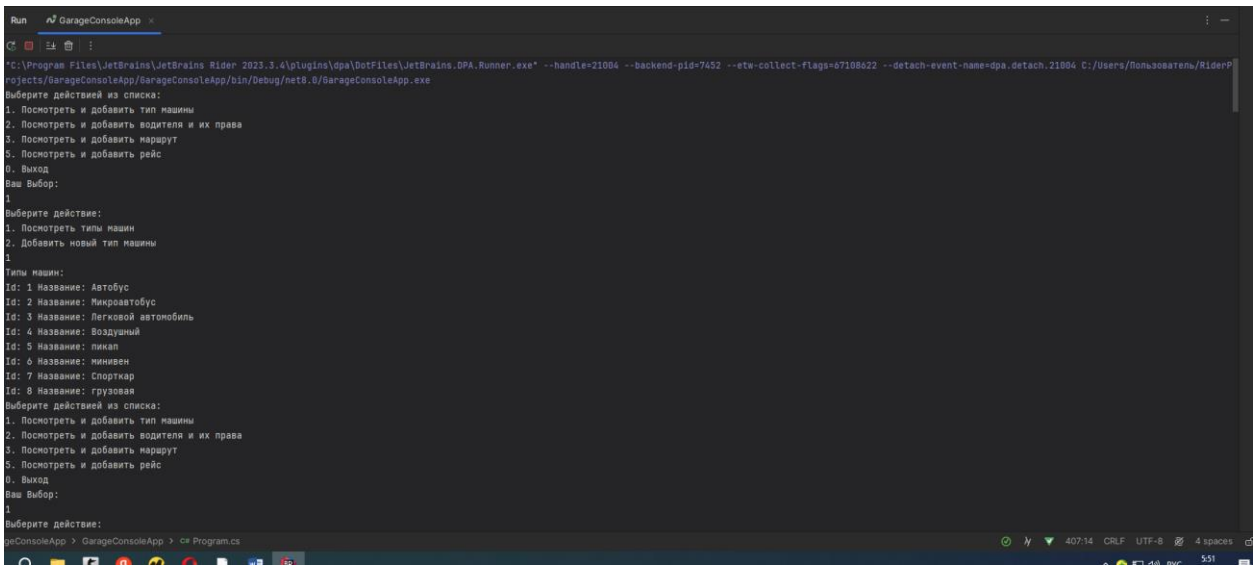
```
    Console.WriteLine("Машины:");
    var querySql = "SELECT id, name, state_number, number_passengers FROM car";
    using var cmd = new NpgsqlCommand(querySql, DatabaseService.GetSqlConnection());
    using var reader = cmd.ExecuteReader();

    while (reader.Read())
    {
        Console.WriteLine($"ID: {reader[0]}, Название: {reader[1]}, Гос. номер: {reader[2]}, Количество пассажиров: {reader[3]}");
    }
}

static void ViewItineraries()
{
    Console.WriteLine("Маршруты:");
    var querySql = "SELECT id, name FROM itinerary";
    using var cmd = new NpgsqlCommand(querySql, DatabaseService.GetSqlConnection());
    using var reader = cmd.ExecuteReader();

    while (reader.Read())
    {
        Console.WriteLine($"ID: {reader[0]}, Название: {reader[1]}");
    }
}

static void ViewDrivers()
{
    Console.WriteLine("Водители:");
    var querySql = "SELECT id, first_name, last_name, birthdate FROM driver";
    using var cmd = new NpgsqlCommand(querySql, DatabaseService.GetSqlConnection());
    using var reader = cmd.ExecuteReader();

    while (reader.Read())
    {
        Console.WriteLine($"ID: {reader[0]}, Имя: {reader[1]} {reader[2]}, Дата рождения: {reader[3]}");
```

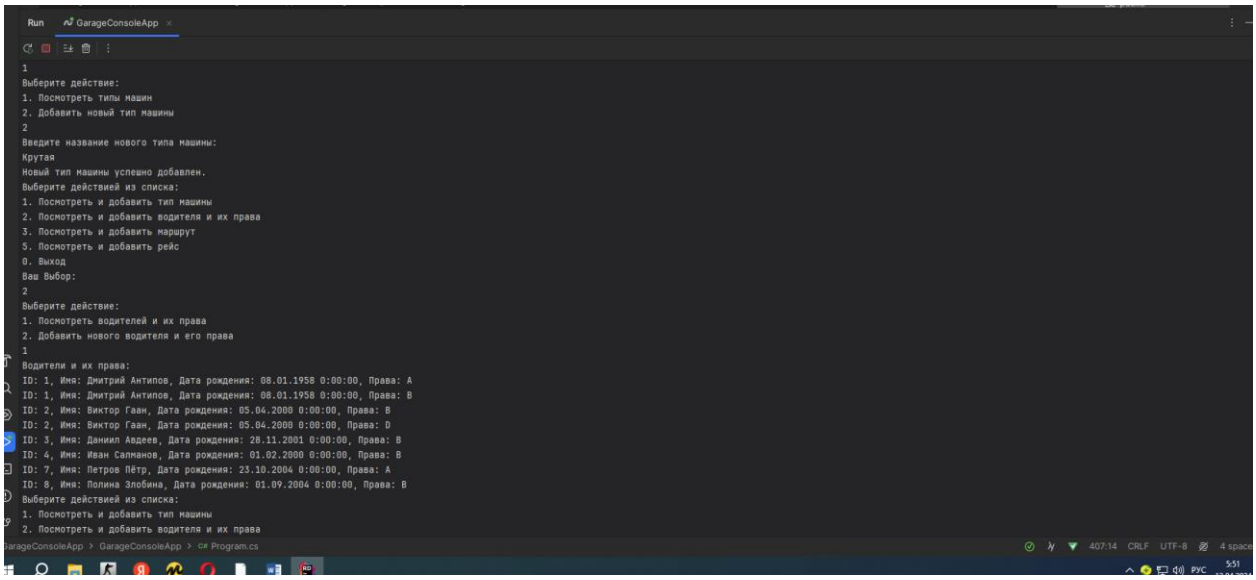


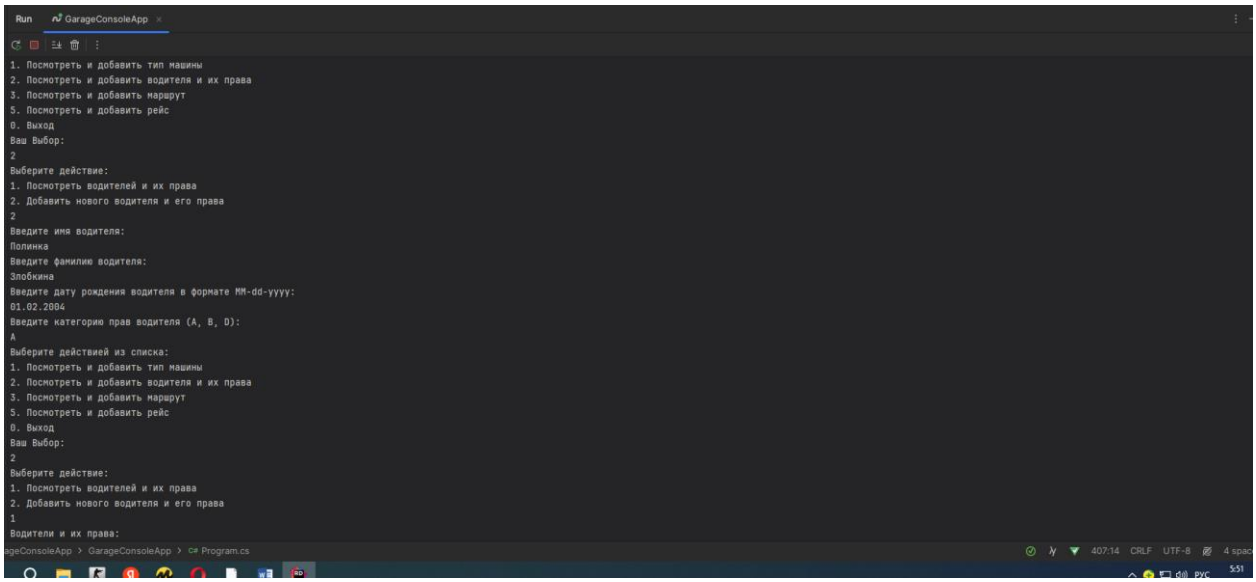
```
    }
}

1 usage  overrides  extension methods  exposing APIs
static void ViewAndAddDireversAndRights()
{
    Console.WriteLine("Выберите действие:");
    Console.WriteLine("1. Посмотреть водителей и их права");
    Console.WriteLine("2. Добавить нового водителя и его права");

    string actionChoice = Console.ReadLine();

    switch (actionChoice)
    {
        case "1":
            ViewDriversAndRights();
            break;
        case "2":
            AddDriverAndRights();
            break;
        default:
            Console.WriteLine("Неверный выбор действия.");
            break;
    }
}

1 usage  overrides  extension methods  exposing APIs
static void ViewDriversAndRights()
{
    Console.WriteLine("Водители и их права:");
    var querySql string = "SELECT dr.id, dr.first_name, dr.last_name, dr.birthdate, rc.name " +
                          "FROM driver dr " +
                          "INNER JOIN driver_rights_category drc ON dr.id = drc.id_driver " +
                          "INNER JOIN rights_category rc ON drc.id_rights_category = rc.id";
    using var cmd = new NpgsqlCommand(querySql, DatabaseService.GetSqlConnection());
    using var reader = cmd.ExecuteReader();
```

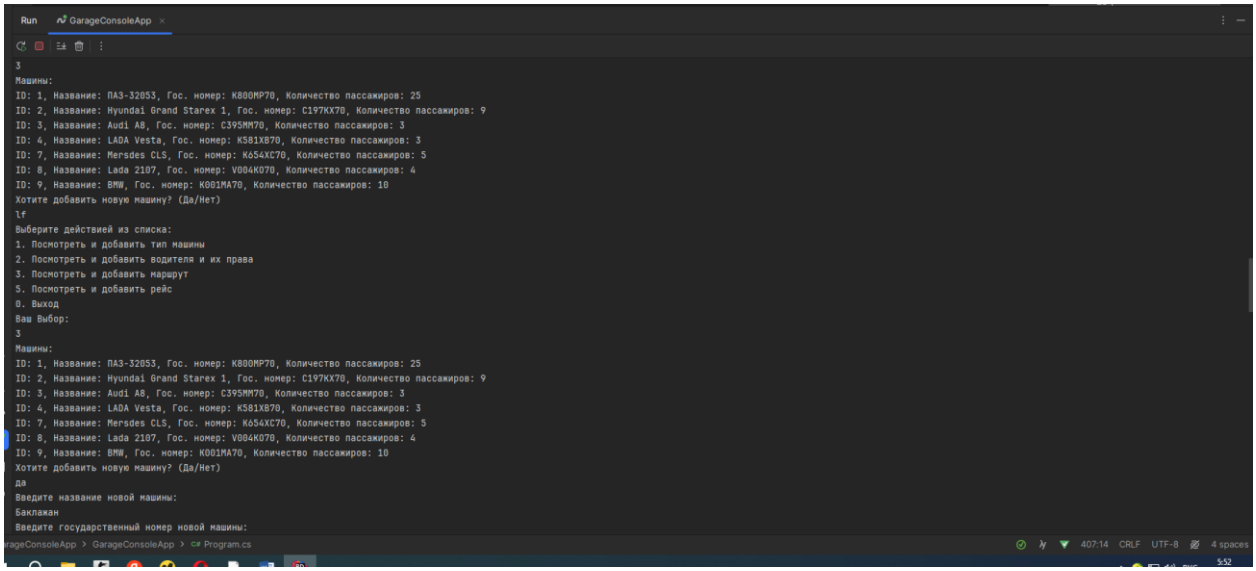


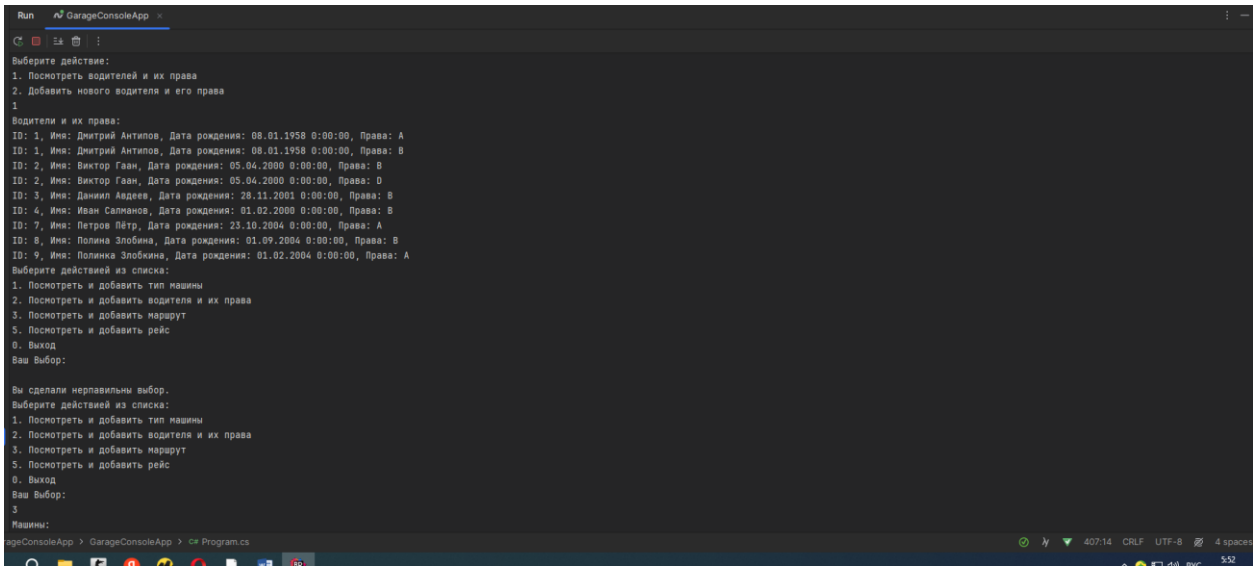
```
    using var cmd = new NpgsqlCommand(querySql, DatabaseService.GetSqlConnection());
    using var reader = cmd.ExecuteReader();

    while (reader.Read())
    {
        Console.WriteLine($"ID: {reader[0]}, Имя: {reader[1]} {reader[2]}, Дата рождения: {reader[3]}, Права: {reader[4]}");
    }
}
1 usage  overrides  extension methods  exposing APIs
static void AddDriverAndRights()
{
    Console.WriteLine("Введите имя водителя:");
    string firstName = Console.ReadLine();
    Console.WriteLine("Введите фамилию водителя:");
    string lastName = Console.ReadLine();
    Console.WriteLine("Введите дату рождения водителя в формате MM-dd-yyyy:");
    DateTime birthdate;
    while (!DateTime.TryParse(Console.ReadLine(), out birthdate))
    {
        Console.WriteLine("Некорректный формат даты. Повторите ввод:");
    }

    Console.WriteLine("Введите категорию прав водителя (A, B, D):");
    string rightsCategory = Console.ReadLine();
    //Добав инфр в бд
    DatabaseService.AddDriverQuery(firstName, lastName, birthdate);


    var driverId int = DatabaseService.GetLastInsertedId();
    // Добав информ прав водителя в бд
    DatabaseService.AddDriverRightsCategoryQuery(driverId, rightsCategory);
}
}
```

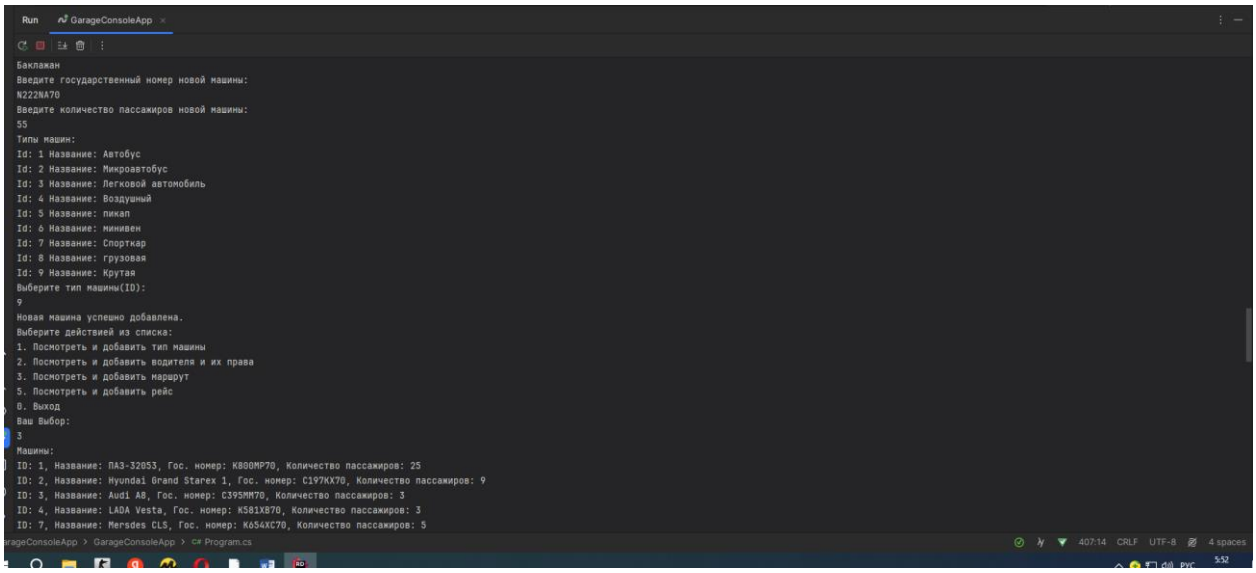

Run
GarageConsoleApp
"C:\Program Files\JetBrains\JetBrains Rider 2023.3.4\plugins\dpa\DotFiles\JetBrains.DPA.Runner.exe" --handle=21004 --backend-pid=7452 --etw-collect-flags=67108622 --detach-event-name=dpa.detach.21004 C:/Users/Пользователь/RiderProjects/GarageConsoleApp/GarageConsoleApp/bin/Debug/net8.0/GarageConsoleApp.exe
Выберите действией из списка:
1. Посмотреть и добавить тип машины
2. Посмотреть и добавить водителя и их права
3. Посмотреть и добавить маршрут
5. Посмотреть и добавить рейс
0. Выход
Ваш Выбор:
1
Выберите действие:
1. Посмотреть типы машин
2. Добавить новый тип машины
1
Типы машин:
Id: 1 Название: Автобус
Id: 2 Название: Микроавтобус
Id: 3 Название: Легковой автомобиль
Id: 4 Название: Воздушный
Id: 5 Название: пикап
Id: 6 Название: минивен
Id: 7 Название: Спорткар
Id: 8 Название: грузовая
Выберите действией из списка:
1. Посмотреть и добавить тип машины
2. Посмотреть и добавить водителя и их права
3. Посмотреть и добавить маршрут
5. Посмотреть и добавить рейс
0. Выход
Ваш Выбор:
1
Выберите действие:
GarageConsoleApp > GarageConsoleApp > Program.cs
407:14 CRLF UTF-8 4 spaces

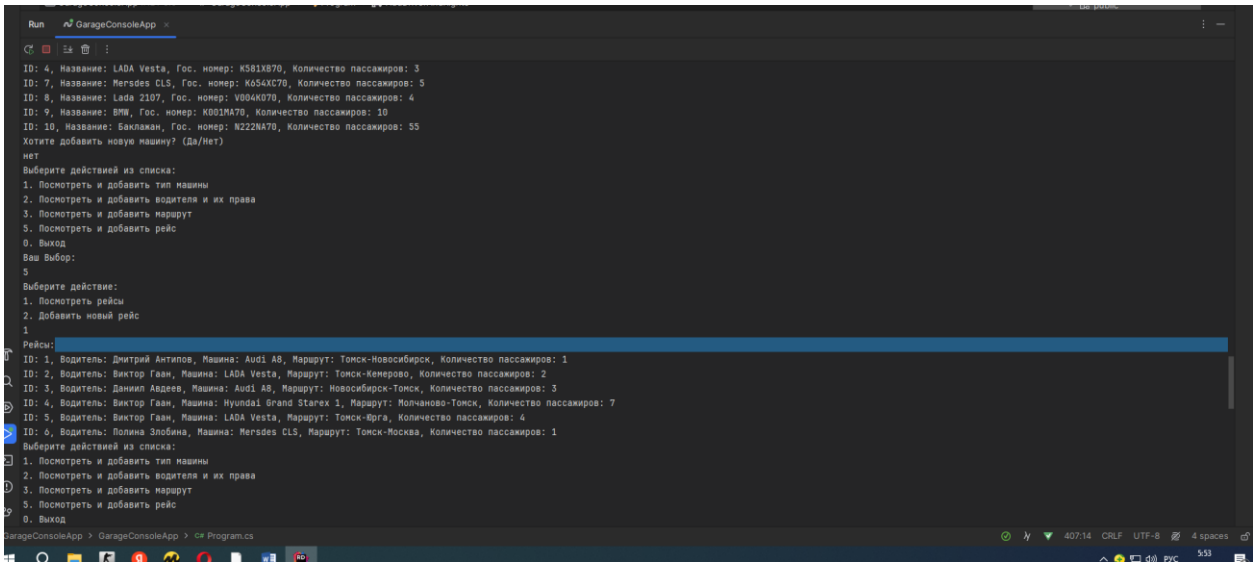

Run
GarageConsoleApp
1
Выберите действие:
1. Посмотреть типы машин
2. Добавить новый тип машины
2
Введите название нового типа машины:
Крутая
Новый тип машины успешно добавлен.
Выберите действией из списка:
1. Посмотреть и добавить тип машины
2. Посмотреть и добавить водителя и их права
3. Посмотреть и добавить маршрут
5. Посмотреть и добавить рейс
0. Выход
Ваш Выбор:
2
Выберите действие:
1. Посмотреть водителей и их права
2. Добавить нового водителя и его права
1
Водители и их права:
ID: 1, Имя: Дмитрий Антипов, Дата рождения: 08.01.1958 0:00:00, Права: A
ID: 1, Имя: Дмитрий Антипов, Дата рождения: 08.01.1958 0:00:00, Права: B
ID: 2, Имя: Виктор Гаам, Дата рождения: 05.04.2000 0:00:00, Права: B
ID: 2, Имя: Виктор Гаам, Дата рождения: 05.04.2000 0:00:00, Права: D
ID: 3, Имя: Даниил Авдеев, Дата рождения: 28.11.2001 0:00:00, Права: B
ID: 4, Имя: Иван Салманов, Дата рождения: 01.02.2000 0:00:00, Права: B
ID: 7, Имя: Петров Пётр, Дата рождения: 23.10.2004 0:00:00, Права: A
ID: 8, Имя: Полина Злобина, Дата рождения: 01.09.2004 0:00:00, Права: B
Выберите действией из списка:
1. Посмотреть и добавить тип машины
2. Посмотреть и добавить водителя и их права
GarageConsoleApp > GarageConsoleApp > Program.cs
407:14 CRLF UTF-8 4 spaces

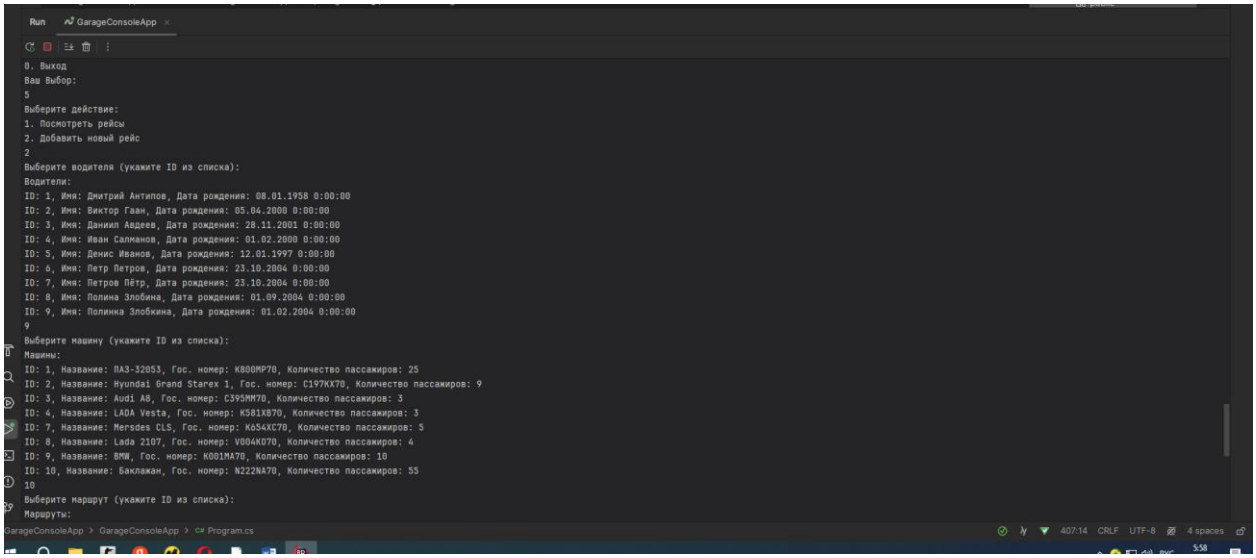

Run
GarageConsoleApp
1. Посмотреть и добавить тип машины
2. Посмотреть и добавить водителя и их права
3. Посмотреть и добавить маршрут
5. Посмотреть и добавить рейс
0. Выход
Ваш Выбор:
2
Выберите действие:
1. Посмотреть водителей и их права
2. Добавить нового водителя и его права
2
Введите имя водителя:
Полинка
Введите фамилию водителя:
Злобкина
Введите дату рождения водителя в формате MM-dd-yyyy:
01.02.2004
Введите категорию прав водителя (A, B, D):
A
Выберите действией из списка:
1. Посмотреть и добавить тип машины
2. Посмотреть и добавить водителя и их права
3. Посмотреть и добавить маршрут
5. Посмотреть и добавить рейс
0. Выход
Ваш Выбор:
2
Выберите действие:
1. Посмотреть водителей и их права
2. Добавить нового водителя и его права
1
Водители и их права:
GarageConsoleApp > GarageConsoleApp > Program.cs
407:14 CRLF UTF-8 4 spaces

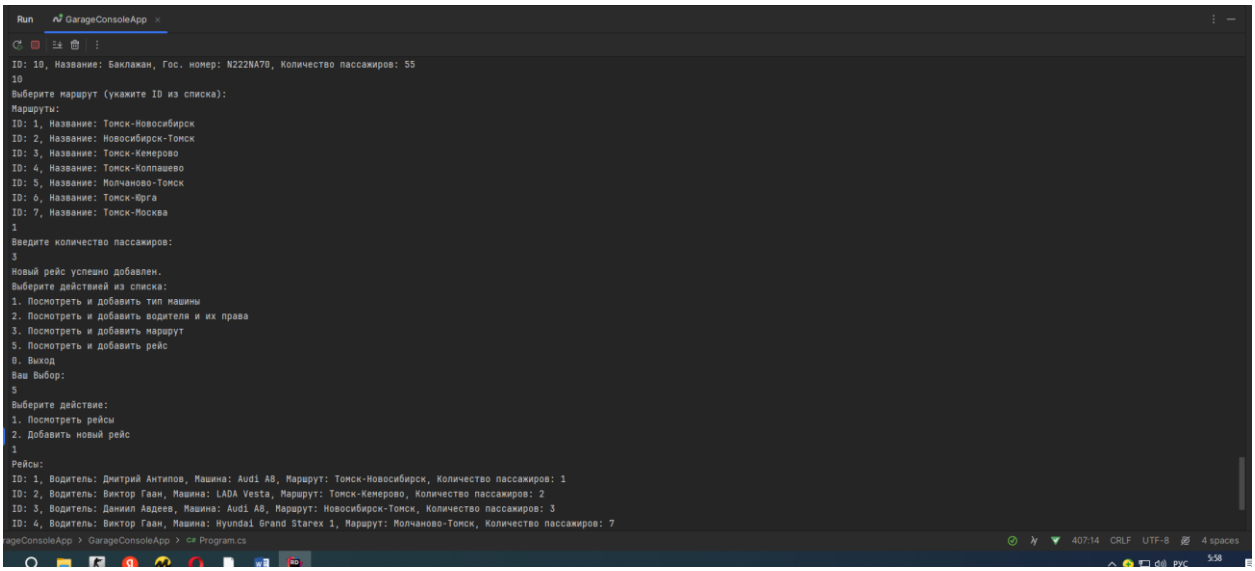

Run
GarageConsoleApp
3
Машины:
ID: 1, Название: ПАЗ-32053, Гос. номер: К800МР70, Количество пассажиров: 25
ID: 2, Название: Hyundai Grand Starex 1, Гос. номер: С197КХ70, Количество пассажиров: 9
ID: 3, Название: Audi A8, Гос. номер: С395ММ70, Количество пассажиров: 3
ID: 4, Название: LADA Vesta, Гос. номер: К581ХВ70, Количество пассажиров: 3
ID: 7, Название: Mersdes CLS, Гос. номер: К654ХС70, Количество пассажиров: 5
ID: 8, Название: Lada 2107, Гос. номер: У004КО70, Количество пассажиров: 4
ID: 9, Название: BMW, Гос. номер: К001МА70, Количество пассажиров: 10
Хотите добавить новую машину? (Да/Нет)
1f
Выберите действией из списка:
1. Посмотреть и добавить тип машины
2. Посмотреть и добавить водителя и их права
3. Посмотреть и добавить маршрут
5. Посмотреть и добавить рейс
0. Выход
Ваш Выбор:
3
Машины:
ID: 1, Название: ПАЗ-32053, Гос. номер: К800МР70, Количество пассажиров: 25
ID: 2, Название: Hyundai Grand Starex 1, Гос. номер: С197КХ70, Количество пассажиров: 9
ID: 3, Название: Audi A8, Гос. номер: С395ММ70, Количество пассажиров: 3
ID: 4, Название: LADA Vesta, Гос. номер: К581ХВ70, Количество пассажиров: 3
ID: 7, Название: Mersdes CLS, Гос. номер: К654ХС70, Количество пассажиров: 5
ID: 8, Название: Lada 2107, Гос. номер: У004КО70, Количество пассажиров: 4
ID: 9, Название: BMW, Гос. номер: К001МА70, Количество пассажиров: 10
Хотите добавить новую машину? (Да/Нет)
да
Введите название новой машины:
Баклажан
Введите государственный номер новой машины:
GarageConsoleApp > GarageConsoleApp > Program.cs
407:14 CRLF UTF-8 4 spaces

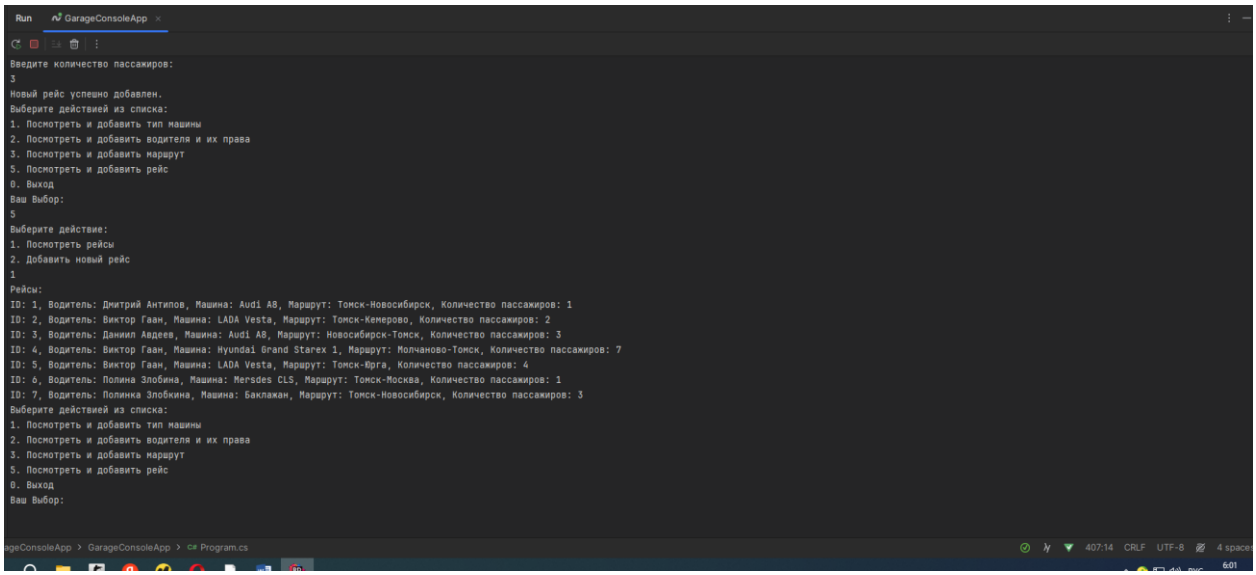

Run
GarageConsoleApp
Выберите действие:
1. Посмотреть водителей и их права
2. Добавить нового водителя и его права
1
Водители и их права:
ID: 1, Имя: Дмитрий Антипов, Дата рождения: 08.01.1958 0:00:00, Права: A
ID: 1, Имя: Дмитрий Антипов, Дата рождения: 08.01.1958 0:00:00, Права: B
ID: 2, Имя: Виктор Гаан, Дата рождения: 05.04.2000 0:00:00, Права: B
ID: 2, Имя: Виктор Гаан, Дата рождения: 05.04.2000 0:00:00, Права: D
ID: 3, Имя: Даниил Авдеев, Дата рождения: 28.11.2001 0:00:00, Права: B
ID: 4, Имя: Иван Салманов, Дата рождения: 01.02.2000 0:00:00, Права: B
ID: 7, Имя: Петров Пётр, Дата рождения: 23.10.2004 0:00:00, Права: A
ID: 8, Имя: Полина Злобина, Дата рождения: 01.09.2004 0:00:00, Права: B
ID: 9, Имя: Полинка Злобкина, Дата рождения: 01.02.2004 0:00:00, Права: A
Выберите действией из списка:
1. Посмотреть и добавить тип машины
2. Посмотреть и добавить водителя и их права
3. Посмотреть и добавить маршрут
5. Посмотреть и добавить рейс
0. Выход
Ваш Выбор:
Вы сделали неправильны выбор.
Выберите действией из списка:
1. Посмотреть и добавить тип машины
2. Посмотреть и добавить водителя и их права
3. Посмотреть и добавить маршрут
5. Посмотреть и добавить рейс
0. Выход
Ваш Выбор:
3
Машины:
GarageConsoleApp > GarageConsoleApp > Program.cs
407:14 CRLF UTF-8 4 spaces


Run
GarageConsoleApp
Баклажан
Введите государственный номер новой машины:
N222NA70
Введите количество пассажиров новой машины:
55
Типы машин:
Id: 1 Название: Автобус
Id: 2 Название: Микроавтобус
Id: 3 Название: Легковой автомобиль
Id: 4 Название: Воздушный
Id: 5 Название: пикап
Id: 6 Название: минивен
Id: 7 Название: Спорткар
Id: 8 Название: грузовая
Id: 9 Название: Крутая
Выберите тип машины(ID):
9
Новая машина успешно добавлена.
Выберите действией из списка:
1. Посмотреть и добавить тип машины
2. Посмотреть и добавить водителя и их права
3. Посмотреть и добавить маршрут
5. Посмотреть и добавить рейс
0. Выход
Ваш Выбор:
3
Машины:
ID: 1, Название: ПАЗ-32053, Гос. номер: К800МР70, Количество пассажиров: 25
ID: 2, Название: Hyundai Grand Starex 1, Гос. номер: С197КХ70, Количество пассажиров: 9
ID: 3, Название: Audi A8, Гос. номер: С395ММ70, Количество пассажиров: 3
ID: 4, Название: LADA Vesta, Гос. номер: К581ХВ70, Количество пассажиров: 3
ID: 7, Название: Mersdes CLS, Гос. номер: К654ХС70, Количество пассажиров: 5
GarageConsoleApp > GarageConsoleApp > Program.cs
407:14 CRLF UTF-8 4 spaces


Run
GarageConsoleApp
ID: 4, Название: LADA Vesta, Гос. номер: К581ХВ70, Количество пассажиров: 3
ID: 7, Название: Mersdes CLS, Гос. номер: К654ХС70, Количество пассажиров: 5
ID: 8, Название: Lada 2107, Гос. номер: У004КО70, Количество пассажиров: 4
ID: 9, Название: BMW, Гос. номер: К001МА70, Количество пассажиров: 10
ID: 10, Название: Баклажан, Гос. номер: N222NA70, Количество пассажиров: 55
Хотите добавить новую машину? (Да/Нет)
нет
Выберите действией из списка:
1. Посмотреть и добавить тип машины
2. Посмотреть и добавить водителя и их права
3. Посмотреть и добавить маршрут
5. Посмотреть и добавить рейс
0. Выход
Ваш Выбор:
5
Выберите действие:
1. Посмотреть рейсы
2. Добавить новый рейс
1
Рейсы:
ID: 1, Водитель: Дмитрий Антипов, Машина: Audi A8, Маршрут: Томск-Новосибирск, Количество пассажиров: 1
ID: 2, Водитель: Виктор Гаан, Машина: LADA Vesta, Маршрут: Томск-Кемерово, Количество пассажиров: 2
ID: 3, Водитель: Даниил Авдеев, Машина: Audi A8, Маршрут: Новосибирск-Томск, Количество пассажиров: 3
ID: 4, Водитель: Виктор Гаан, Машина: Hyundai Grand Starex 1, Маршрут: Молчаново-Томск, Количество пассажиров: 7
ID: 5, Водитель: Виктор Гаан, Машина: LADA Vesta, Маршрут: Томск-Юрга, Количество пассажиров: 4
ID: 6, Водитель: Полина Злобина, Машина: Mersdes CLS, Маршрут: Томск-Москва, Количество пассажиров: 1
Выберите действией из списка:
1. Посмотреть и добавить тип машины
2. Посмотреть и добавить водителя и их права
3. Посмотреть и добавить маршрут
5. Посмотреть и добавить рейс
0. Выход
GarageConsoleApp > GarageConsoleApp > Program.cs
407:14 CRLF UTF-8 4 spaces
5:53


Run
GarageConsoleApp
0. Выход
Ваш Выбор:
5
Выберите действие:
1. Посмотреть рейсы
2. Добавить новый рейс
2
Выберите водителя (укажите ID из списка):
Водители:
ID: 1, Имя: Дмитрий Антипов, Дата рождения: 08.01.1958 0:00:00
ID: 2, Имя: Виктор Гаан, Дата рождения: 05.04.2000 0:00:00
ID: 3, Имя: Даниил Авдеев, Дата рождения: 28.11.2001 0:00:00
ID: 4, Имя: Иван Салманов, Дата рождения: 01.02.2000 0:00:00
ID: 5, Имя: Денис Иванов, Дата рождения: 12.01.1997 0:00:00
ID: 6, Имя: Петр Петров, Дата рождения: 23.10.2004 0:00:00
ID: 7, Имя: Петров Пётр, Дата рождения: 23.10.2004 0:00:00
ID: 8, Имя: Полина Злобина, Дата рождения: 01.09.2004 0:00:00
ID: 9, Имя: Полинка Злобкина, Дата рождения: 01.02.2004 0:00:00
9
Выберите машину (укажите ID из списка):
Машины:
ID: 1, Название: ПАЗ-32053, Гос. номер: К800МР70, Количество пассажиров: 25
ID: 2, Название: Hyundai Grand Starex 1, Гос. номер: С197КХ70, Количество пассажиров: 9
ID: 3, Название: Audi A8, Гос. номер: С395ММ70, Количество пассажиров: 3
ID: 4, Название: LADA Vesta, Гос. номер: К581ХВ70, Количество пассажиров: 3
ID: 7, Название: Mersdes CLS, Гос. номер: К654ХС70, Количество пассажиров: 5
ID: 8, Название: Lada 2107, Гос. номер: У004КО70, Количество пассажиров: 4
ID: 9, Название: BMW, Гос. номер: К001МА70, Количество пассажиров: 10
ID: 10, Название: Баклажан, Гос. номер: N222NA70, Количество пассажиров: 55
10
Выберите маршрут (укажите ID из списка):
Маршруты:
GarageConsoleApp > GarageConsoleApp > Program.cs
407:14 CRLF UTF-8 4 spaces
5:58


Run
GarageConsoleApp
ID: 10, Название: Баклажан, Гос. номер: N222NA70, Количество пассажиров: 55
10
Выберите маршрут (укажите ID из списка):
Маршруты:
ID: 1, Название: Томск-Новосибирск
ID: 2, Название: Новосибирск-Томск
ID: 3, Название: Томск-Кемерово
ID: 4, Название: Томск-Колпашево
ID: 5, Название: Молчаново-Томск
ID: 6, Название: Томск-Юрга
ID: 7, Название: Томск-Москва
1
Введите количество пассажиров:
3
Новый рейс успешно добавлен.
Выберите действией из списка:
1. Посмотреть и добавить тип машины
2. Посмотреть и добавить водителя и их права
3. Посмотреть и добавить маршрут
5. Посмотреть и добавить рейс
0. Выход
Ваш Выбор:
5
Выберите действие:
1. Посмотреть рейсы
2. Добавить новый рейс
1
Рейсы:
ID: 1, Водитель: Дмитрий Антипов, Машина: Audi A8, Маршрут: Томск-Новосибирск, Количество пассажиров: 1
ID: 2, Водитель: Виктор Гаан, Машина: LADA Vesta, Маршрут: Томск-Кемерово, Количество пассажиров: 2
ID: 3, Водитель: Даниил Авдеев, Машина: Audi A8, Маршрут: Новосибирск-Томск, Количество пассажиров: 3
ID: 4, Водитель: Виктор Гаан, Машина: Hyundai Grand Starex 1, Маршрут: Молчаново-Томск, Количество пассажиров: 7
GarageConsoleApp > GarageConsoleApp > Program.cs
407:14 CRLF UTF-8 4 spaces
5:58


Run
GarageConsoleApp
Введите количество пассажиров:
3
Новый рейс успешно добавлен.
Выберите действией из списка:
1. Посмотреть и добавить тип машины
2. Посмотреть и добавить водителя и их права
3. Посмотреть и добавить маршрут
5. Посмотреть и добавить рейс
0. Выход
Ваш Выбор:
5
Выберите действие:
1. Посмотреть рейсы
2. Добавить новый рейс
1
Рейсы:
ID: 1, Водитель: Дмитрий Антипов, Машина: Audi A8, Маршрут: Томск-Новосибирск, Количество пассажиров: 1
ID: 2, Водитель: Виктор Гаан, Машина: LADA Vesta, Маршрут: Томск-Кемерово, Количество пассажиров: 2
ID: 3, Водитель: Даниил Авдеев, Машина: Audi A8, Маршрут: Новосибирск-Томск, Количество пассажиров: 3
ID: 4, Водитель: Виктор Гаан, Машина: Hyundai Grand Starex 1, Маршрут: Молчаново-Томск, Количество пассажиров: 7
ID: 5, Водитель: Виктор Гаан, Машина: LADA Vesta, Маршрут: Томск-Юрга, Количество пассажиров: 4
ID: 6, Водитель: Полина Злобина, Машина: Mersdes CLS, Маршрут: Томск-Москва, Количество пассажиров: 1
ID: 7, Водитель: Полинка Злобкина, Машина: Баклажан, Маршрут: Томск-Новосибирск, Количество пассажиров: 3
Выберите действией из списка:
1. Посмотреть и добавить тип машины
2. Посмотреть и добавить водителя и их права
3. Посмотреть и добавить маршрут
5. Посмотреть и добавить рейс
0. Выход
Ваш Выбор:
ageConsoleApp › GarageConsoleApp › Program.cs
407:14
CRLF
UTF-8
4 spaces
6:01

```
using Npgsql;

namespace GarageConsoleApp;

/// <summary>
/// Класс DatabaseService
/// отвечает за подключение и открытие соединения с БД
/// </summary>
public static class DatabaseService
{
    /// <summary>
    /// Переменная _connection
    /// хранит открытое соединение с БД
    /// </summary>
    private static NpgsqlConnection? _connection;

    /// <summary>
    /// Метод GetConnectionString()
    /// возвращает строку подключения к БД
    /// </summary>
    private static string GetConnectionString()
    {
        return
@"Host=localhost;Port=5432;Database=garage_db;Username=postgres;Password=1234
";
    }

    /// <summary>
    /// Метод GetSqlConnection()
    /// проверяет есть ли уже открытое соединение с БД
    /// если нет, то открывает соединение с БД
    /// </summary>
    /// <returns></returns>
    public static NpgsqlConnection GetSqlConnection()
    {
        if (_connection is null)
        {
            _connection = new NpgsqlConnection(GetConnectionString());
            _connection.Open();
        }

        return _connection;
    }







    public static void AddDriverQuery(string firstName, string lastName,
DateTime birthdate)
    {
        var querySql =
            $"INSERT INTO driver(first_name, last_name, birthdate) VALUES
('{firstName}', '{lastName}', '{birthdate.ToString("yyyy-MM-dd")}')";
        using var cmd = new NpgsqlCommand(querySql, GetSqlConnection());
        cmd.ExecuteNonQuery();
    }

    public static void AddDriverRightsCategoryQuery(int driverId, string
rightsCategory)
    {

        var querySql = $"SELECT id FROM rights_category WHERE name =
```

```
'{rightsCategory}'";
        using var cmd = new NpgsqlCommand(querySql, GetSqlConnection());
        var rightsCategoryId = cmd.ExecuteScalar();

        if (rightsCategoryId == null)
        {
            Console.WriteLine("Указанная категория прав не найдена.");
            return;
        }


        querySql =
            $"INSERT INTO driver_rights_category(id_driver,
id_rights_category) VALUES ({driverId}, {rightsCategoryId})";
        cmd.CommandText = querySql;
        cmd.ExecuteNonQuery();
    }

    public static int GetLastInsertedId()
    {
        var querySQL = "SELECT lastval()";
        using var cmd = new NpgsqlCommand(querySQL, GetSqlConnection());
        return Convert.ToInt32(cmd.ExecuteScalar());
    }
}
```

```
using Microsoft.EntityFrameworkCore.Metadata.Internal;

namespace GarageConsoleApp;
using System;
using Npgsql;


/// <summary>
/// Класс Program
/// здесь описываем логику приложения
/// вызываем нужные методы пишем обработчики и тд
/// </summary>
public class Program
{
    public static void Main(string[] args)
    {
        while (true)
        {
            Console.WriteLine("Выберите действией из списка:");
            Console.WriteLine("1. Посмотреть и добавить тип машины");
            Console.WriteLine("2. Посмотреть и добавить водителя и их
права");
            Console.WriteLine("3. Посмотреть и добавить маршрут");
            Console.WriteLine("5. Посмотреть и добавить рейс");
            Console.WriteLine("0. Выход");

            Console.WriteLine("Ваш Выбор:");
            string A = Console.ReadLine();

            switch (A)
            {
                case "1" :
                    ViewAndTypeCars();
                    break;
                case "2" :
                    ViewAndAddDireversAndRights();
                    break;
                case "3":
                    ViewAndAddCars();
                    break;
                case "4":
                    ViewAndAddroutes();
                    break;
                case "5":
                    ViewAndAddIteneraries();
                    break;
                case "0":
                    return;
                default:
                    Console.WriteLine("Вы сделали нерпавильны выбор.");
                    break;
            }

        }
        static void ViewAndTypeCars()
        {
            Console.WriteLine("Выберите действие:");
            Console.WriteLine("1. Посмотреть типы машин");
            Console.WriteLine("2. Добавить новый тип машины");

            //Выбор действия
```

```
            string actionChoice = Console.ReadLine();

            switch (actionChoice)
            {
                case "1":
                    ViewTypeCars();
                    break;
                case "2":
                    AddTypeCar();
                    break;
                default:
                    Console.WriteLine("Неверный выбор действия.");
                    break;
            }
        }


        static void AddTypeCar()
        {
            Console.WriteLine("Введите название нового типа машины:");
            string typeName = Console.ReadLine();

            var querySql = $"INSERT INTO type_car (name) VALUES
('{typeName}')";

            //выполнения SQL-запрос  с использованием соединения с бд
            using var cmd = new NpgsqlCommand(querySql,
DatabaseService.GetSqlConnection());
            var rowsAffected = cmd.ExecuteNonQuery();

            //использ строк
            if (rowsAffected > 0)
                Console.WriteLine("Новый тип машины успешно добавлен.");
            else
                Console.WriteLine("Ошибка при добавлении типа машины.");
        }


    }
    static void ViewTypeCars()
    {
        Console.WriteLine("Типы машин:");
        // сбор записей
        var querySql = "SELECT * FROM type_car";
        using var cmd = new NpgsqlCommand(querySql,
DatabaseService.GetSqlConnection());
        using var reader = cmd.ExecuteReader();

        while (reader.Read())
        {
            Console.WriteLine($"Id: {reader[0]} Название: {reader[1]}");
        }
    }

    static void ViewAndAddroutes()
    {
        Console.WriteLine("Выберите действие:");
        Console.WriteLine("1. Посмотреть маршруты");
        Console.WriteLine("2. Добавить новый маршрут");

        string actionChoice = Console.ReadLine();

        switch (actionChoice)
        {
```

```
            case "1":
                ViewRoutes();
                break;
            case "2":
                AddRoute();
                break;
            default:
                Console.WriteLine("Неверный выбор действия.");
                break;
        }
    }

    static void ViewRoutes()
    {
        Console.WriteLine("Маршруты:");
        var querySql = "SELECT id, name FROM itinerary";
        using var cmd = new NpgsqlCommand(querySql,
DatabaseService.GetSqlConnection());
        using var reader = cmd.ExecuteReader();

        while (reader.Read())
        {
            Console.WriteLine($"ID: {reader[0]}, Название: {reader[1]}");
        }
    }

    static void AddRoute()
    {
        Console.WriteLine("Введите название нового маршрута:");
        string routeName = Console.ReadLine();

        //добавление нового маршрута
        var querySql = $"INSERT INTO itinerary (name) VALUES
('{routeName}')";
        using var cmd = new NpgsqlCommand(querySql,
DatabaseService.GetSqlConnection());
        var rowsAffected = cmd.ExecuteNonQuery();

        if (rowsAffected > 0)
        {
            Console.WriteLine("Новый маршрут успешно добавлен.");
        }
        else
        {
            Console.WriteLine("Ошибка при добавлении нового маршрута.");
        }
    }


    static void ViewAndAddCars()
    {
        Console.WriteLine("Машины:");


        var querySql = "SELECT id, name, state_number, number_passengers FROM
car";

        using var cmd = new NpgsqlCommand(querySql,
DatabaseService.GetSqlConnection());
        using var reader = cmd.ExecuteReader();

        while (reader.Read())
        {
            Console.WriteLine($"ID: {reader[0]}, Название: {reader[1]}, Гос.
```

```
номер: {reader[2]}, Количество пассажиров: {reader[3]}");
        }

        reader.Close();

        Console.WriteLine("Хотите добавить новую машину? (Да/Нет)");
        string response = Console.ReadLine();

        if (response.ToLower() == "да")
        {
            Console.WriteLine("Введите название новой машины:");
            string name = Console.ReadLine();

            Console.WriteLine("Введите государственный номер новой машины:");
            string stateNumber = Console.ReadLine();

            Console.WriteLine("Введите количество пассажиров новой машины:");
            int numberPassengers;
            while (!int.TryParse(Console.ReadLine(), out numberPassengers))
            {
                Console.WriteLine("Некорректное значение. Повторите ввод:");
            }
            ViewTypeCars();
            Console.WriteLine("Выберите тип машины(ID):");
            string id_type_car = Console.ReadLine();

            int typeId = int.Parse(id_type_car);

            //Вызывает метод для добав  машины с указанными параметрами
            AddCar(name, stateNumber, numberPassengers, typeId.ToString());
        }
    }



    static void AddCar(string name, string stateNumber, int numberPassengers,
string id_type_car)
    {
        var querySql = $"INSERT INTO car (name, state_number,
number_passengers, id_type_car) VALUES ('{name}', '{stateNumber}',
{numberPassengers}, {id_type_car})";
        using var cmd = new NpgsqlCommand(querySql,
DatabaseService.GetSqlConnection());
        int rowsAffected = cmd.ExecuteNonQuery();

        if (rowsAffected > 0)
        {
            Console.WriteLine("Новая машина успешно добавлена.");
        }
        else
        {
            Console.WriteLine("Ошибка при добавлении новой машины.");
        }
    }
        static void ViewAndAddIteneraries()
{
    Console.WriteLine("Выберите действие:");
    Console.WriteLine("1. Посмотреть рейсы");
    Console.WriteLine("2. Добавить новый рейс");

    string actionChoice = Console.ReadLine();

    switch (actionChoice)
    {
```

```
        case "1":
            ViewFlights();
            break;
        case "2":
            AddFlight();
            break;
        default:
            Console.WriteLine("Неверный выбор действия.");
            break;
    }
}

static void ViewFlights()
{
    Console.WriteLine("Рейсы:");
    var querySql = "SELECT r.id, d.first_name || ' ' || d.last_name AS
driver_name, c.name AS car_name, i.name AS itinerary_name,
r.number_passengers " +
                   "FROM route r " +
                   "INNER JOIN driver d ON r.id_driver = d.id " +
                   "INNER JOIN car c ON r.id_car = c.id " +
                   "INNER JOIN itinerary i ON r.id_itinerary = i.id";
    using var cmd = new NpgsqlCommand(querySql,
DatabaseService.GetSqlConnection());
    using var reader = cmd.ExecuteReader();

    while (reader.Read())
    {
        Console.WriteLine($"ID: {reader[0]}, Водитель: {reader[1]}, Машина:
{reader[2]}, Маршрут: {reader[3]}, Количество пассажиров: {reader[4]}");
    }
}

static void AddFlight()
{
    Console.WriteLine("Выберите водителя (укажите ID из списка):");
    //метод для просмотра водителей
    ViewDrivers();

    int driverId;
    while (!int.TryParse(Console.ReadLine(), out driverId))
    {
        Console.WriteLine("Некорректный формат. Повторите ввод:");
    }

    Console.WriteLine("Выберите машину (укажите ID из списка):");
    ViewCars();
    int carId;
    while (!int.TryParse(Console.ReadLine(), out carId))
    {
        Console.WriteLine("Некорректный формат. Повторите ввод:");
    }

    Console.WriteLine("Выберите маршрут (укажите ID из списка):");
    ViewItineraries();
    int itineraryId;
    while (!int.TryParse(Console.ReadLine(), out itineraryId))
    {
        Console.WriteLine("Некорректный формат. Повторите ввод:");
    }

    Console.WriteLine("Введите количество пассажиров:");
    int numberPassengers;
    while (!int.TryParse(Console.ReadLine(), out numberPassengers))
```

```
    {
        Console.WriteLine("Некорректный формат. Повторите ввод:");
    }


    var querySql = $"INSERT INTO route (id_driver, id_car, id_itinerary,
number_passengers) " +
                   $"VALUES ({driverId}, {carId}, {itineraryId},
{numberPassengers})";
    using var cmd = new NpgsqlCommand(querySql,
DatabaseService.GetSqlConnection());
    var rowsAffected = cmd.ExecuteNonQuery();

    if (rowsAffected > 0)
    {
        Console.WriteLine("Новый рейс успешно добавлен.");
    }
    else
    {
        Console.WriteLine("Ошибка при добавлении нового рейса.");
    }


}
static void ViewCars()
{
    Console.WriteLine("Машины:");
    var querySql = "SELECT id, name, state_number, number_passengers FROM
car";
    using var cmd = new NpgsqlCommand(querySql,
DatabaseService.GetSqlConnection());
    using var reader = cmd.ExecuteReader();

    while (reader.Read())
    {
        Console.WriteLine($"ID: {reader[0]}, Название: {reader[1]}, Гос.
номер: {reader[2]}, Количество пассажиров: {reader[3]}");
    }
}

static void ViewItineraries()
{
    Console.WriteLine("Маршруты:");
    var querySql = "SELECT id, name FROM itinerary";
    using var cmd = new NpgsqlCommand(querySql,
DatabaseService.GetSqlConnection());
    using var reader = cmd.ExecuteReader();

    while (reader.Read())
    {
        Console.WriteLine($"ID: {reader[0]}, Название: {reader[1]}");
    }
}

static void ViewDrivers()
{
    Console.WriteLine("Водители:");
    var querySql = "SELECT id, first_name, last_name, birthdate FROM driver";
    using var cmd = new NpgsqlCommand(querySql,
DatabaseService.GetSqlConnection());
    using var reader = cmd.ExecuteReader();

    while (reader.Read())
    {
```

```
        Console.WriteLine($"ID: {reader[0]}, Имя: {reader[1]} {reader[2]},
Дата рождения: {reader[3]}");
    }
}
        static void ViewAndAddDireversAndRights()
        {
            Console.WriteLine("Выберите действие:");
            Console.WriteLine("1. Посмотреть водителей и их права");
            Console.WriteLine("2. Добавить нового водителя и его права");

            string actionChoice = Console.ReadLine();

            switch (actionChoice)
            {
                case "1":
                    ViewDriversAndRights();
                    break;
                case "2":
                    AddDriverAndRights();
                    break;
                default:
                    Console.WriteLine("Неверный выбор действия.");
                    break;
            }
        }

        static void ViewDriversAndRights()
        {
            Console.WriteLine("Водители и их права:");
            var querySql = "SELECT dr.id, dr.first_name, dr.last_name,
dr.birthdate, rc.name " +
                           "FROM driver dr " +
                           "INNER JOIN driver_rights_category drc ON dr.id =
drc.id_driver " +
                           "INNER JOIN rights_category rc ON
drc.id_rights_category = rc.id";
            using var cmd = new NpgsqlCommand(querySql,
DatabaseService.GetSqlConnection());
            using var reader = cmd.ExecuteReader();

            while (reader.Read())
            {
                Console.WriteLine($"ID: {reader[0]}, Имя: {reader[1]}
{reader[2]}, Дата рождения: {reader[3]}, Права: {reader[4]}");
            }
        }
        static void AddDriverAndRights()
        {
            Console.WriteLine("Введите имя водителя:");
            string firstName = Console.ReadLine();
            Console.WriteLine("Введите фамилию водителя:");
            string lastName = Console.ReadLine();
            Console.WriteLine("Введите дату рождения водителя в формате MM-
dd-yyyy:");
            DateTime birthdate;
            while (!DateTime.TryParse(Console.ReadLine(), out birthdate))
            {
                Console.WriteLine("Некорректный формат даты. Повторите
ввод:");
            }

            Console.WriteLine("Введите категорию прав водителя (A, B, D):");
            string rightsCategory = Console.ReadLine();
            //добав инфр в бд
```

```
            DatabaseService.AddDriverQuery(firstName, lastName, birthdate);


            var driverId = DatabaseService.GetLastInsertedId();
            // Добав информ прав водителя в бд
            DatabaseService.AddDriverRightsCategoryQuery(driverId,
rightsCategory);
        }
    }
```