Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет инфокоммуникационных технологий

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

по теме: Разработка интерфейсов для выполнения CRUD-операций над базой данных средствами PHP
по дисциплине: Проектирование и реализация баз данных

Специальность:
09.03.03 Мобильные и сетевые технологии

Проверил:
Говорова М.М. _____
Дата: «__» _____ 20__г.
Оценка _____

Выполнил:
студент группы К3241
Кривошапкина А.С.

Санкт-Петербург 2021 г

1. Цель работы

Овладеть практическими навыками разработки форм для вставки, выборки и редактирования данных

2. Практическое задание

1. Изучить функции открытия соединения к базе данных средствами PHP.
2. Изучить основные функции для создания php-скрипта (на базе видео-уроков 1-7).
3. Создать сайт с использованиеv базовых возможностей PHP (в соответствии с содержанием видео-уроков 1-7).

3. Выполнение

I. **Наименование БД:** «Таксопарк»

II. **Схема логической модели базы данных, сгенерированная в Generate ERD**

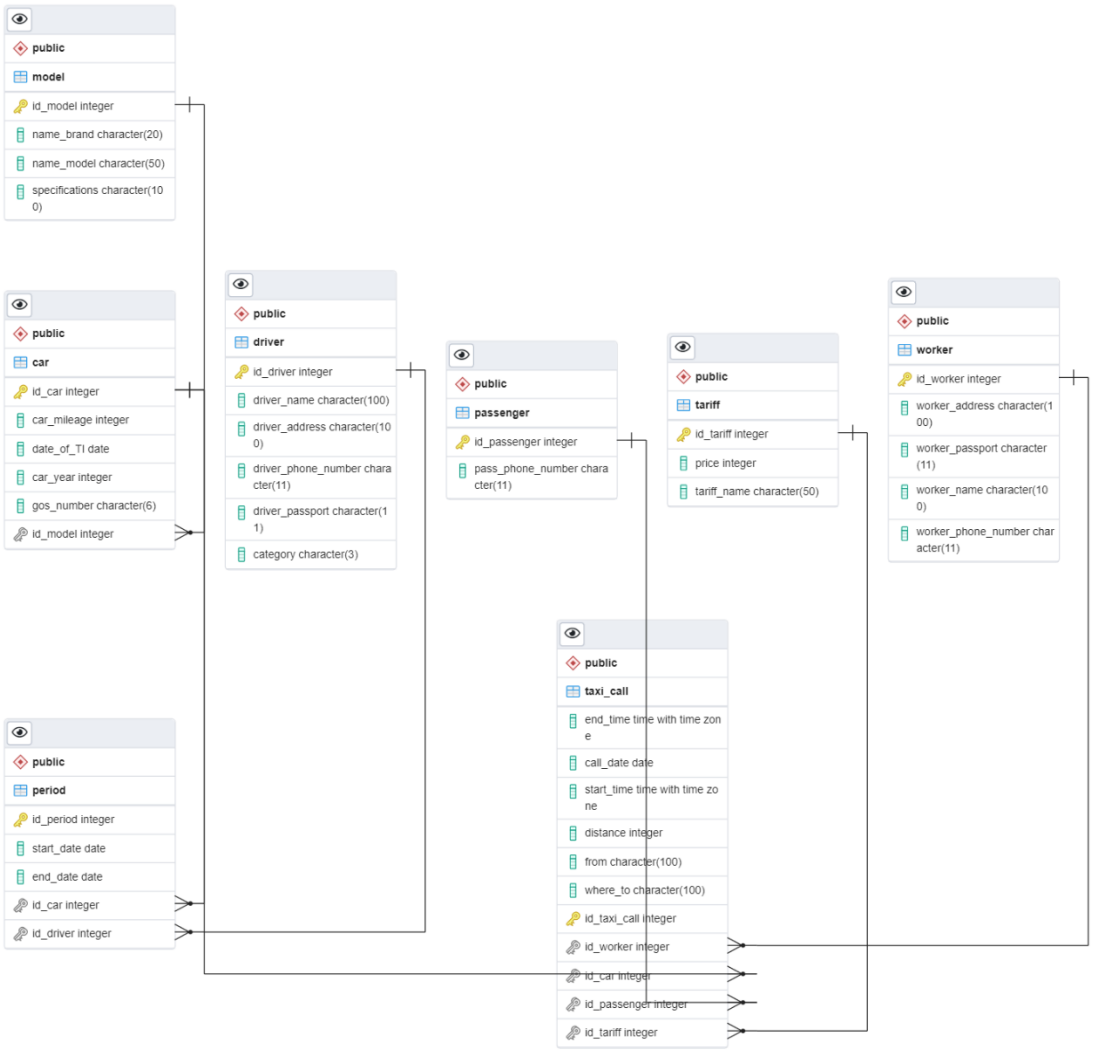

Рис. 1 – Схема логической модели базы данных «Таксопарк»

III. CRUD

Для выполнения задания были использованы: PostgreSQL, Python, HTML

- Create

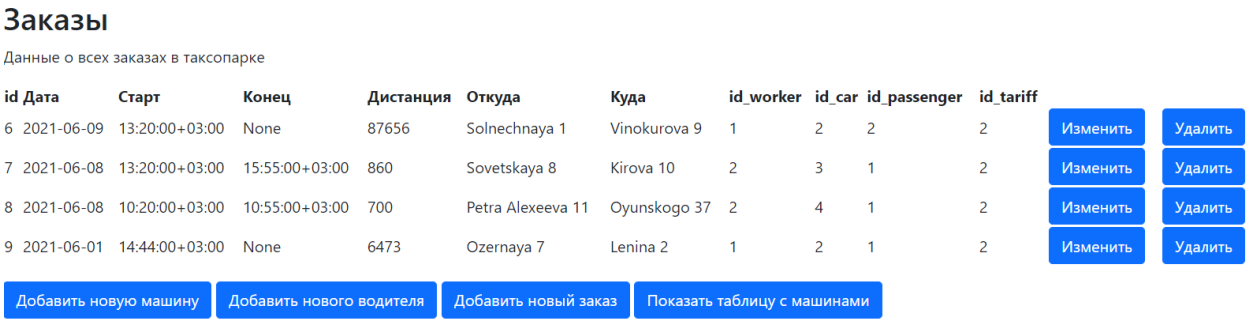

Рис. 2 – Интерфейс начальной страницы с данными таблицы taxi_call

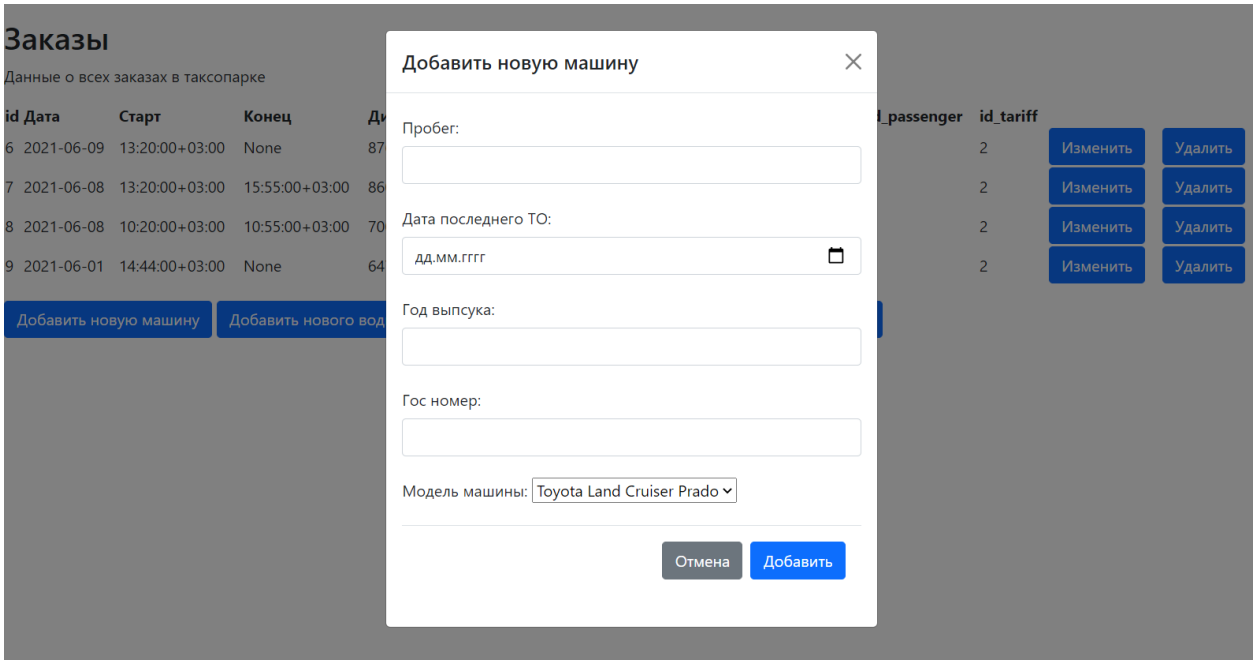

Рис. 3 –Интерфейс добавления новой машины

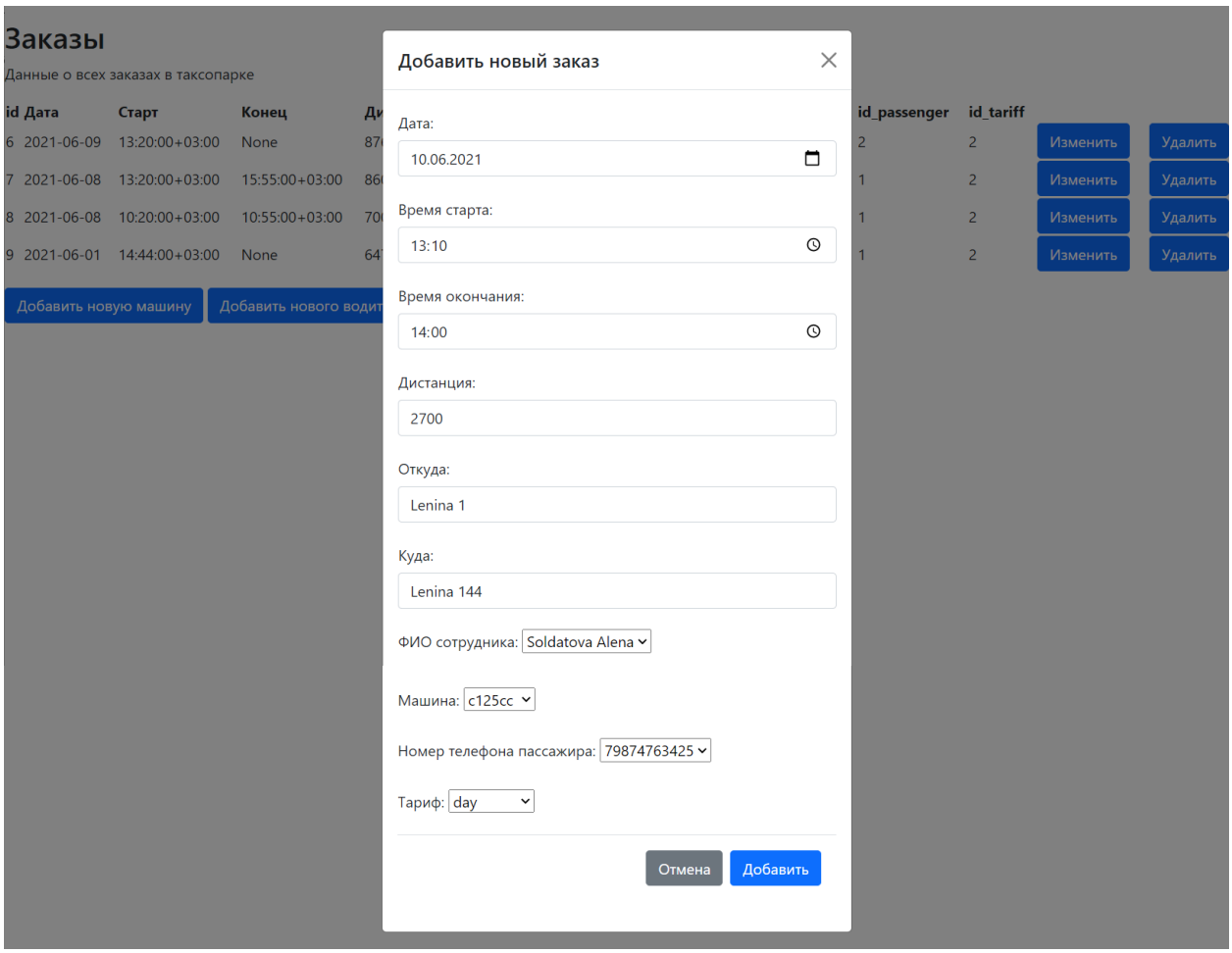

Рис. 4 –Интерфейс добавления нового заказа

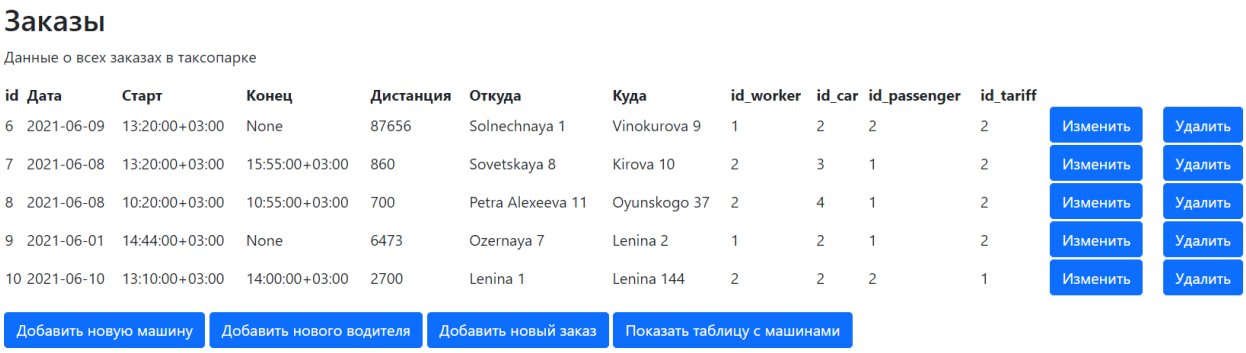

Рис. 5 –Результат добавления нового заказа

- Read

Заказы

Данные о всех заказах в таксопарке

id	Дата	Старт	Конец	Дистанция	Откуда	Куда	id_worker	id_car	id_passenger	id_tariff		
6	2021-06-09	13:20:00+03:00	None	87656	Solnechnaya 1	Vinokurova 9	1	2	2	2	Изменить	Удалить
7	2021-06-08	13:20:00+03:00	15:55:00+03:00	860	Sovetskaya 8	Kirova 10	2	3	1	2	Изменить	Удалить
8	2021-06-08	10:20:00+03:00	10:55:00+03:00	700	Petra Alexeeva 11	Oyunskogo 37	2	4	1	2	Изменить	Удалить
9	2021-06-01	14:44:00+03:00	None	6473	Ozernaya 7	Lenina 2	1	2	1	2	Изменить	Удалить

Добавить новую машину | Добавить нового водителя | Добавить новый заказ | Показать таблицу с машинами

Рис. 6 –Отобразим данные таблицы car с помощью кнопки «Показать таблицу с машинами»

Заказы

Данные о всех заказах в таксопарке

id	Дата	Старт	Конец	Дистанция	Откуда	Куда	id_worker	id_car	id_passenger	id_tariff		
6	2021-06-09	13:20:00+03:00	None	87656	Solnechnaya 1	Vinokurova 9	1	2	2	2	Изменить	Удалить
7	2021-06-08	13:20:00+03:00	15:55:00+03:00	860	Sovetskaya 8	Kirova 10	2	3	1	2	Изменить	Удалить
8	2021-06-08	10:20:00+03:00	10:55:00+03:00	700	Petra Alexeeva 11	Oyunskogo 37	2	4	1	2	Изменить	Удалить
9	2021-06-01	14:44:00+03:00	None	6473	Ozernaya 7	Lenina 2	1	2	1	2	Изменить	Удалить
10	2021-06-10	13:10:00+03:00	14:00:00+03:00	2700	Lenina 1	Lenina 144	2	2	2	1	Изменить	Удалить

Добавить новую машину | Добавить нового водителя | Добавить новый заказ | Скрыть таблицу с машинами

Машины

Данные о всех машинах в таксопарке

id	Пробег	Дата ТО	Год	Гос номер	Номер модели		
1	45679	2021-06-01	2018	a576kc	3	Изменить	Удалить
2	579000	2020-01-15	2018	c125cc	1	Изменить	Удалить
3	593870	2021-03-27	2020	k983at	2	Изменить	Удалить
4	73626	2021-02-01	2018	a676ka	4	Изменить	Удалить
5	73539	2021-02-01	2018	a783ka	4	Изменить	Удалить

Рис. 6 –Отображение данных таблицы car

- Update

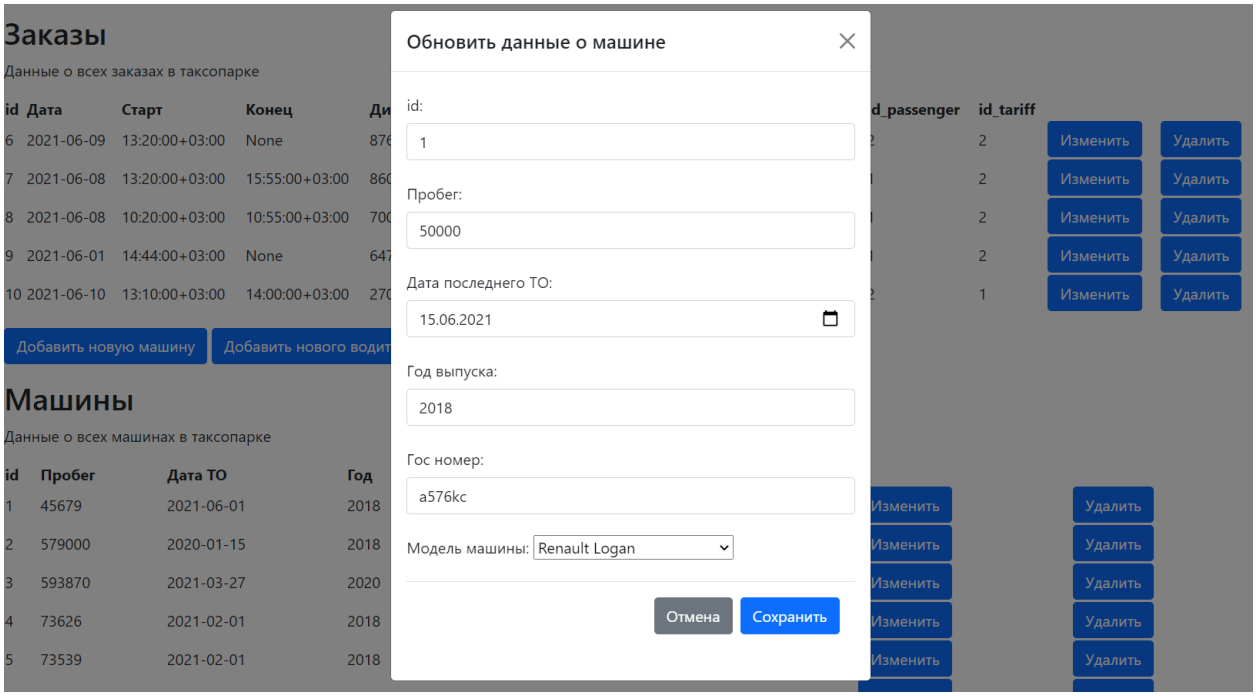

Рис. 7 –Интерфейс изменения данных машины

Рис. 8 –Результат изменения данных машины

- Delete

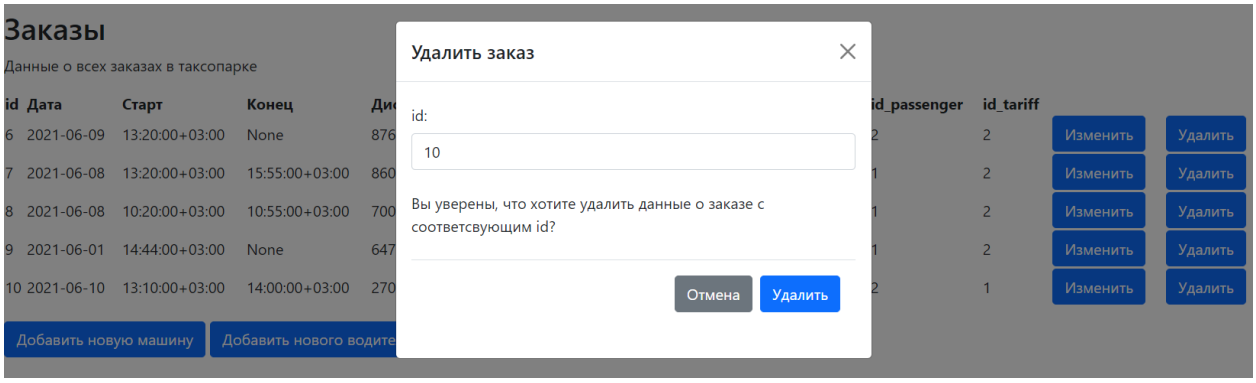

Рис. 7 –Интерфейс удаления данных заказа

Рис. 7 –Результат удаления данных заказа

- Листинг

```
from flask import Flask, redirect
from flask import render_template
from flask import request
import psycopg2

# Подключение к серверу
t_host = "127.0.0.1"
t_port = "5432"
t_dbname = "postgres"
    t_name_user = "postgres"
    t_password = "190101"
    db_conn = psycopg2.connect(host=t_host, port=t_port, dbname=t_dbname, user=t_name_user, password=t_password)
    db_cursor = db_conn.cursor()

    app = Flask(__name__)


    # Создание необходимых функций для работы с БД
    def create_driver(id, name, address, number, passport, category):
        s = f'''INSERT INTO public.driver(
        id_driver, driver_name, driver_address, driver_phone_number, driver_passport, category)
            VALUES ({id}, \'{name}\', \'{address}\', \'{number}\', \'{passport}\', \'{category}\');'''
        db_cursor.execute(s)

    def create_car(id, mileage, ti, year, gos, id_model):
        s = f'''INSERT INTO public.car(
            id_car, car_mileage, "date_of_TI", car_year, gos_number, id_model)
            VALUES ({id}, {mileage}, \'{ti}\', {year}, \'{gos}\', {id_model});'''
        db_cursor.execute(s)

    def create_taxi_call(end, date, start, distance, from_, where, id, id_worker, id_car, id_passenger, id_tariff):
        s = f'''INSERT INTO public.taxi_call(
            end_time, call_date, start_time, distance, "from", where_to, id_taxi_call, id_worker, id_car, id_passenger, id_tariff)
            VALUES (\'{end}\', \'{date}\', \'{start}\', {distance}, \'{from_}\', \'{where}\', {id}, {id_worker}, {id_car}, {id_passenger}, {id_tariff});'''
        db_cursor.execute(s)

    def select_driver(t_schema):
        s = f'''SELECT *
            FROM public.driver;'''
        db_cursor.execute(s)
        return db_cursor.fetchall()

    def select_passenger(t_schema):
        s = f'''SELECT *
            FROM public.passenger;'''
        db_cursor.execute(s)
        return db_cursor.fetchall()

```

```
49.     def select_car(t_schema):
50.         s = f'''SELECT *
51.             FROM public.car
52.         ORDER BY id_car;'''
53.         db_cursor.execute(s)
54.         return db_cursor.fetchall()
55.
56.     def select_model(t_schema):
57.         s = f'''SELECT *
58.             FROM public.model;'''
59.         db_cursor.execute(s)
60.         return db_cursor.fetchall()
61.
62.     def select_period(t_schema):
63.         s = f'''SELECT *
64.             FROM public.period;'''
65.         db_cursor.execute(s)
66.         return db_cursor.fetchall()
67.
68.     def select_tariff(t_schema):
69.         s = f'''SELECT *
70.             FROM public.tariff;'''
71.         db_cursor.execute(s)
72.         return db_cursor.fetchall()
73.
74.     def select_taxi_call(t_schema):
75.         s = f'''SELECT *
76.         FROM public.taxi_call;'''
77.         db_cursor.execute(s)
78.         return db_cursor.fetchall()
79.
80.     def select_worker(t_schema):
81.         s = f'''SELECT *
82.             FROM public.worker;'''
83.         db_cursor.execute(s)
84.         return db_cursor.fetchall()
85.
86.     def update_car(id, mileage, ti, year, gos, id_model):
87.         s = f'''UPDATE public.car
88.             SET car_mileage={mileage}, "date_of_TI"=\'{ti}\',
    car_year={year}, gos_number=\'{gos}\', id_model={id_model}
89.             WHERE id_car={id};'''
90.         db_cursor.execute(s)
91.
92.     def delete_car(id):
93.         s = f'''DELETE FROM public.car
94.             WHERE id_car={id};'''
95.         db_cursor.execute(s)
96.
97.     def update_taxi_call(end, date, start, distance, from_, where, id,
    id_worker, id_car, id_passenger, id_tariff):
98.         s = f'''UPDATE public.taxi_call
99.             SET end_time=\'{end}\', call_date=\'{date}\',
    start_time=\'{start}\', distance={distance}, "from"=\'{from_}\',
    where_to=\'{where}\', id_worker={id_worker}, id_car={id_car},
    id_passenger={id_passenger}, id_tariff={id_tariff}
100.             WHERE id_taxi_call={id};'''
101.         db_cursor.execute(s)
102.
```

```
103.    def delete_taxi_call(id):
104.        s = f'''DELETE FROM public.taxi_call
105.            WHERE id_taxi_call={id};'''
106.        db_cursor.execute(s)
107.
108.
109.    @app.route("/", methods=["POST","GET"])
110.    def Main():
111.
112.        t_action = request.args.get("t_action", "")
113.        t_name_table = request.args.get("t_name_table", "")
114.
115.        t_schema = "public"
116.        drivers_tables = select_driver(t_schema)
117.        cars_tables = select_car(t_schema)
118.        taxi_calls_tables = select_taxi_call(t_schema)
119.        car_models = select_model(t_schema)
120.        workers = select_worker(t_schema)
121.        cars = select_car(t_schema)
122.        passengers = select_passenger(t_schema)
123.        tariffs = select_tariff(t_schema)
124.
125.        get_driver_categories = f'''SELECT DISTINCT category
126.        FROM public.driver'''
127.        db_cursor.execute(get_driver_categories)
128.        driver_categories_tuple = db_cursor.fetchall()
129.        driver_categories = []
130.        for item in driver_categories_tuple:
131.            driver_categories.append(item[0].replace(' ', ''))
132.
133.        select_model(t_schema)
134.
135.        t_url = "http://127.0.0.1:5000/"
136.
137.        return render_template("ws.html", **locals())
138.
139.    # Добавление нового заказа
140.    @app.route("/add_taxi_call", methods=["POST","GET"])
141.    def add_taxi_call():
142.        if request.method == 'POST':
143.            get_id_request = f'''SELECT id_taxi_call
144.            FROM public.taxi_call
145.            ORDER BY id_taxi_call DESC
146.            LIMIT 1'''
147.
148.            db_cursor.execute(get_id_request)
149.
150.            id = db_cursor.fetchall()
151.            id = id[0][0] + 1
152.
153.            end = request.form.get('end')
154.            date = request.form.get('date')
155.            start = request.form.get('start')
156.            distance = request.form.get('distance')
157.            from_ = request.form.get('from_')
158.            where = request.form.get('where')
159.            id_worker = request.form.get('id_worker')
160.            id_car = request.form.get('id_car')
161.            id_passenger = request.form.get('id_passenger')
```

```
            id_tariff = request.form.get('id_tariff')

            create_taxi_call(end, date, start, distance, from_, where, id,
   id_worker, id_car, id_passenger, id_tariff)

        return redirect('/')


    # Добавление нового водителя
    @app.route("/add_driver", methods=["POST","GET"])
    def add_driver():
        if request.method == 'POST':
            get_id_request = f'''SELECT id_driver
            FROM public.driver
            ORDER BY id_driver DESC
            LIMIT 1'''

            db_cursor.execute(get_id_request)

            id = db_cursor.fetchall()
            id = id[0][0] + 1

            name = request.form.get('name')
            address = request.form.get('address')
            number = request.form.get('number')
            passport = request.form.get('passport')
            category = request.form.get('category')

            create_driver(id, name, address, number, passport, category)

        return redirect('/')


    # Добавление новой машины
    @app.route("/add_car", methods=["POST","GET"])
    def add_car():
        if request.method == 'POST':
            get_id_request = f'''SELECT id_car
            FROM public.car
            ORDER BY id_car DESC
            LIMIT 1'''

            db_cursor.execute(get_id_request)

            id = db_cursor.fetchall()
            print(id)

            id = id[0][0] + 1

            mileage = request.form.get('mileage')
            ti = request.form.get('ti')
            year = request.form.get('year')
            gos = request.form.get('gos')
            id_model = request.form.get('id_model')

            create_car(id, mileage, ti, year, gos, id_model)

        return redirect('/')

```

```
220.    # Обновление данных машины
221.    @app.route("/update_car", methods=["POST","GET"])
222.    def update_info_car():
223.        if request.method == 'POST':
224.            id = request.form.get('id_car')
225.            mileage = request.form.get('mileage')
226.            ti = request.form.get('ti')
227.            year = request.form.get('year')
228.            gos = request.form.get('gos')
229.            id_model = request.form.get('id_model')
230.
231.            update_car(id, mileage, ti, year, gos, id_model)
232.
233.        return redirect('/')
234.
235.
236.    # Удаление данных машины
237.    @app.route("/delete_car", methods=["POST","GET"])
238.    def delete_info_car():
239.        if request.method == 'POST':
240.            id = request.form.get('id_car')
241.
242.            delete_car(id)
243.
244.        return redirect('/')
245.
246.
247.    # Обновление данных заказа
248.    @app.route("/update_taxi_call", methods=["POST","GET"])
249.    def update_info_taxi_call():
250.        if request.method == 'POST':
251.            id = request.form.get('id_taxi_call')
252.            end = request.form.get('end')
253.            date = request.form.get('date')
254.            start = request.form.get('start')
255.            distance = request.form.get('distance')
256.            from_ = request.form.get('from_')
257.            where = request.form.get('where')
258.            id_worker = request.form.get('id_worker')
259.            id_car = request.form.get('id_car')
260.            id_passenger = request.form.get('id_passenger')
261.            id_tariff = request.form.get('id_tariff')
262.
263.            update_taxi_call(end, date, start, distance, from_, where, id,
    id_worker, id_car, id_passenger, id_tariff)
264.
265.        return redirect('/')\
266.
267.
268.    # Удаление данных заказа
269.    @app.route("/delete_taxi_call", methods=["POST","GET"])
270.    def delete_info_taxi_call():
271.        if request.method == 'POST':
272.            id = request.form.get('id_taxi_call')
273.
274.            delete_taxi_call(id)
275.
276.        return redirect('/')
277.
```

```
278.
279.    app.run(debug=True, use_reloader=False)
```

4. Выводы

В результате выполненной работы были созданы формы для вставки, редактирования, удаления и выборки данных для базы данных «Таксопарк»