Практическая работа 8
Расчет пути

```
import flet as ft
import requests
import sqlite3
from datetime import datetime

def get_coordinates(city):
    try:
        geo = requests.get(f'https://geocoding-api.open-meteo.com/v1/search?
name={city}&count=1', timeout=10).json()
        if geo.get('results'):
            return geo['results'][0]['longitude'], geo['results'][0]['latitude']
    except:
        pass
    return None, None

def get_route(lon1, lat1, lon2, lat2):
    try:
        route = requests.get(f'http://router.project-osrm.org/route/v1/driving/{lon1},
{lat1};{lon2},{lat2}?overview=false', timeout=10).json()
        if route.get('routes'):
            return route['routes'][0]['distance'] / 1000
    except:
        pass
    return None

def get_hint(city1, city2, lon1=None, lat1=None, lon2=None, lat2=None):
    if lon1 is None or lat1 is None:
        lon1, lat1 = get_coordinates(city1)
    if lon2 is None or lat2 is None:
        lon2, lat2 = get_coordinates(city2)
    if lon1 is None or lon2 is None:
        return None
    distance = get_route(lon1, lat1, lon2, lat2)
    if distance is None:
        return None
    return distance, city1, city2

def init_db():
```

```
    conn = sqlite3.connect('history.db')
    cursor = conn.cursor()
    cursor.execute('''CREATE TABLE IF NOT EXISTS history
                   (id INTEGER PRIMARY KEY AUTOINCREMENT,
                    from_point TEXT,
                    to_point TEXT,
                    distance REAL,
                    transport TEXT,
                    price REAL,
                    timestamp TEXT)''')
    conn.commit()
    conn.close()

def save_to_db(from_point, to_point, distance, transport, price):
    conn = sqlite3.connect('history.db')
    cursor = conn.cursor()
    cursor.execute('INSERT INTO history (from_point, to_point, distance, transport,
price, timestamp) VALUES (?, ?, ?, ?, ?, ?)',
                   (from_point, to_point, distance, transport, price,
datetime.now().strftime('%Y-%m-%d %H:%M:%S')))
    conn.commit()
    conn.close()

def load_history():
    conn = sqlite3.connect('history.db')
    cursor = conn.cursor()
    cursor.execute('SELECT id, from_point, to_point, distance, transport, price,
timestamp FROM history ORDER BY id DESC LIMIT 10')
    rows = cursor.fetchall()
    conn.close()
    return rows

def main(page: ft.Page):
    def on_radio_change(e):
        if radio_group.value == "city":
            input_1.visible = True
            input_2.visible = True
            lat1.visible = False
            lon1.visible = False
            lat2.visible = False
            lon2.visible = False
        else:
            input_1.visible = False
```

```
            input_2.visible = False
            lat1.visible = True
            lon1.visible = True
            lat2.visible = True
            lon2.visible = True
        page.update()

    def show_history_item(from_point, to_point, distance, transport, price,
timestamp):
        dlg = ft.AlertDialog(
            title=ft.Text("Запись из истории"),
            content=ft.Text(
                f"Пункт отправления: {from_point}\n"
                f"Пункт назначения: {to_point}\n"
                f"Расстояние: {distance:.1f} км\n"
                f"Тип транспорта: {transport}\n"
                f"Цена: {price:.1f} рублей\n"
                f"Дата: {timestamp}"
            ),
            on_dismiss=lambda e: page.close(dlg)
        )
        page.open(dlg)

    def update_history_table():
        history_data = load_history()
        history_list.controls = []
        for row in history_data:
            row_id, from_point, to_point, distance, transport, price, timestamp = row
            item = ft.ListTile(
                title=ft.Text(f"{from_point} → {to_point}"),
                subtitle=ft.Text(f"{distance:.1f} км | {transport} | {price:.1f} руб |
{timestamp}"),
                on_click=lambda e, fp=from_point, tp=to_point, d=distance, t=transport,
p=price, ts=timestamp: show_history_item(fp, tp, d, t, p, ts),
                bgcolor="#333333" if row_id % 2 == 0 else None
            )
            history_list.controls.append(item)
        page.update()

    def calculate(e):
        transport_type = dropdown.value
        if transport_type == 'car':
            cost = 4
```

```
            transport_name = 'Автомобиль'
        elif transport_type == 'bike':
            cost = 1
            transport_name = 'Мотоцикл'
        else:
            cost = 5
            transport_name = 'Грузовик'

        if radio_group.value == "city":
            city1 = (input_1.value or "").strip()
            city2 = (input_2.value or "").strip()
            if not city1 or not city2:
                dlg = ft.AlertDialog(
                    title=ft.Text("Ошибка"),
                    content=ft.Text("Введите названия городов"),
                    on_dismiss=lambda e: page.close(dlg)
                )
                page.open(dlg)
                return
            result = get_hint(city1, city2)
            if result:
                distance, c1, c2 = result
                from_point = c1 if c1 else city1
                to_point = c2 if c2 else city2
            else:
                dlg = ft.AlertDialog(
                    title=ft.Text("Ошибка"),
                    content=ft.Text("Не удалось найти маршрут между городами"),
                    on_dismiss=lambda e: page.close(dlg)
                )
                page.open(dlg)
                return
        else:
            try:
                lon1_val = float(lon1.value) if lon1.value else None
                lat1_val = float(lat1.value) if lat1.value else None
                lon2_val = float(lon2.value) if lon2.value else None
                lat2_val = float(lat2.value) if lat2.value else None
                if lon1_val is not None and lat1_val is not None and lon2_val is not None
and lat2_val is not None:
                    distance = get_route(lon1_val, lat1_val, lon2_val, lat2_val)
                    if distance is None:
                        dlg = ft.AlertDialog(
```

```
                    title=ft.Text("Ошибка"),
                    content=ft.Text("Не удалось построить маршрут"),
                    on_dismiss=lambda e: page.close(dlg)
                )
                page.open(dlg)
                return
            from_point = f"{lat1_val}, {lon1_val}"
            to_point = f"{lat2_val}, {lon2_val}"
        else:
            dlg = ft.AlertDialog(
                title=ft.Text("Ошибка"),
                content=ft.Text("Введите все координаты"),
                on_dismiss=lambda e: page.close(dlg)
            )
            page.open(dlg)
            return
    except ValueError:
        dlg = ft.AlertDialog(
            title=ft.Text("Ошибка"),
            content=ft.Text("Некорректные координаты"),
            on_dismiss=lambda e: page.close(dlg)
        )
        page.open(dlg)
        return

    total_price = cost * distance
    save_to_db(from_point, to_point, distance, transport_name, total_price)
    dlg = ft.AlertDialog(
        title=ft.Text("Результат"),
        content=ft.Text(
            f"Пункт отправления: {from_point}\n"
            f"Пункт назначения: {to_point}\n"
            f"Расстояние: {distance:.1f} км\n"
            f"Тип транспорта: {transport_name}\n"
            f"Цена: {total_price:.1f} рублей"
        ),
        on_dismiss=lambda e: page.close(dlg)
    )
    page.open(dlg)
    update_history_table()

def clear_fields(e):
    input_1.value = ""
```

```
    input_2.value = ""
    lat1.value = ""
    lon1.value = ""
    lat2.value = ""
    lon2.value = ""
    page.update()

page.title = 'Калькулятор расстояния'
page.horizontal_alignment = ft.CrossAxisAlignment.CENTER
page.vertical_alignment = ft.MainAxisAlignment.CENTER

header = ft.Text('Калькулятор расстояния', size=40)

radio_group = ft.RadioGroup(
    value="city",
    on_change=on_radio_change,
    content=ft.Column([
        ft.Radio(value="city", label="Города"),
        ft.Radio(value="coord", label="Координаты")
    ])
)

input_1 = ft.TextField(label='Пункт отправления (город)')
input_2 = ft.TextField(label='Пункт назначения (город)')

lat1 = ft.TextField(label='Широта 1', visible=False)
lon1 = ft.TextField(label='Долгота 1', visible=False)
lat2 = ft.TextField(label='Широта 2', visible=False)
lon2 = ft.TextField(label='Долгота 2', visible=False)

dropdown = ft.Dropdown(label='Тип транспорта', options=[
    ft.DropdownOption(key="car", text="Автомобиль 4 руб."),
    ft.DropdownOption(key="bus", text="Грузовик 6 руб."),
    ft.DropdownOption(key="bike", text="Мотоцикл 2 руб."),
])

btn = ft.ElevatedButton('Рассчитать', on_click=calculate)
btn_1 = ft.ElevatedButton('Очистить', on_click=clear_fields)

history_header = ft.Text('История поиска (нажмите для просмотра)', size=24)
history_list = ft.Column(scroll=ft.ScrollMode.AUTO, height=300)

first_block = ft.Column(controls=[
```

```
        header,
        radio_group,
        input_1, input_2,
        lat1, lon1, lat2, lon2,
        dropdown,
        ft.Row(controls=[btn, btn_1])
    ])

    history_block = ft.Column(controls=[
        history_header,
        history_list
    ])

    page.add(first_block)
    page.add(history_block)
    update_history_table()

if __name__ == "__main__":
    init_db()
    ft.app(target=main)
```

Теперь посмотрим на работу приложения на рисунке 1:

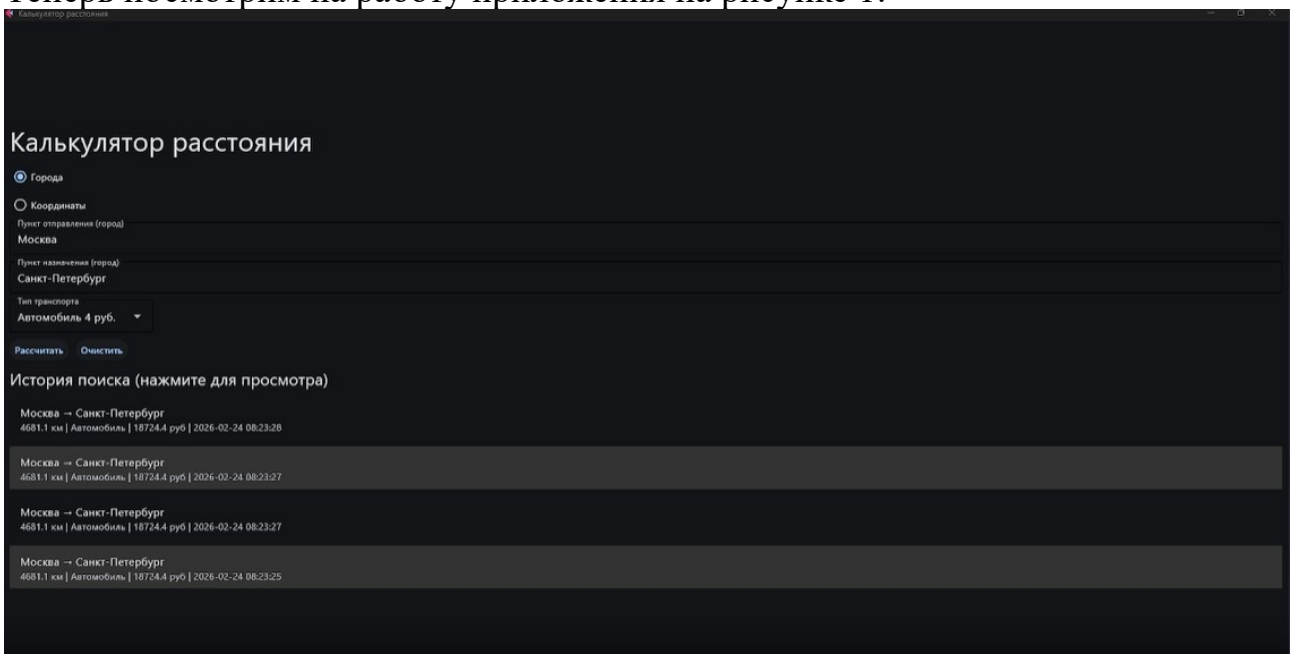

Рисунок 1 — Демонстрация работы