# Список файлов проекта

## Файл: C:/Users/User/PycharmProjects/service\_with\_mini\_app\app.py

import asyncio  
from threading import Thread  
from flask import Flask  
from telegram.ext import Application  
from bot.handlers import setup\_handlers  
from api.routes import init\_api  
from models.database import init\_db  
from config import TOKEN, MINI\_APP\_URL  
  
app = Flask(\_\_name\_\_, static\_folder='static')  
application = Application.builder().token(TOKEN).build()  
  
init\_db()  
setup\_handlers(application)  
init\_api(app, application) # Передаем application как аргумент  
  
def run\_flask():  
 app.run(debug=True, host='0.0.0.0', port=5000, use\_reloader=False)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 flask\_thread = Thread(target=run\_flask)  
 flask\_thread.start()  
 asyncio.run(application.run\_polling())

## Файл: C:/Users/User/PycharmProjects/service\_with\_mini\_app\config.py

# config.py  
MINI\_APP\_URL = "https://fa66-88-204-232-102.ngrok-free.app/miniapp"  
TOKEN = "7661263528:AAHB4LKirWI6Xtw\_MRIgrzQPqq22Xz-\_AUI" # Можно также перенести сюда

## Файл: C:/Users/User/PycharmProjects/service\_with\_mini\_app\docker-compose.yaml

version: '3.8'  
  
services:  
 app:  
 build: .  
 ports:  
 - "5000:5000" # Пробрасываем порт Flask наружу  
 environment:  
 - DB\_HOST=db # Имя сервиса PostgreSQL в Docker Compose  
 - DB\_NAME=service\_bot\_db  
 - DB\_USER=postgres  
 - DB\_PASSWORD=your\_password # Укажите ваш пароль  
 - DB\_PORT=5432  
 depends\_on:  
 - db  
 volumes:  
 - .:/app # Монтируем код для разработки  
  
 db:  
 image: postgres:16 # Официальный образ PostgreSQL  
 environment:  
 - POSTGRES\_DB=service\_bot\_db  
 - POSTGRES\_USER=postgres  
 - POSTGRES\_PASSWORD=your\_password # Укажите тот же пароль, что выше  
 ports:  
 - "5432:5432" # Пробрасываем порт PostgreSQL наружу (опционально)  
 volumes:  
 - pgdata:/var/lib/postgresql/data # Сохраняем данные базы между перезапусками  
  
volumes:  
 pgdata:

## Файл: C:/Users/User/PycharmProjects/service\_with\_mini\_app\Dockerfile

# Базовый образ Python  
FROM python:3.9-slim  
  
# Установка рабочей директории  
WORKDIR /app  
  
# Копирование зависимостей  
COPY requirements.txt .  
  
# Установка зависимостей  
RUN pip install --no-cache-dir -r requirements.txt  
  
# Копирование всего проекта  
COPY . .  
  
# Команда для запуска приложения  
CMD ["python", "app.py"]

## Файл: C:/Users/User/PycharmProjects/service\_with\_mini\_app\export\_to\_word.py

from docx import Document  
from docx.oxml.ns import qn  
from docx.enum.style import WD\_STYLE\_TYPE  
from docx.shared import Pt  
import os  
from datetime import datetime  
  
# Путь к папке с вашими файлами  
#folder\_path = "C:/Users/Nurba/PycharmProjects/service\_with\_mini\_app" # на работе папка  
folder\_path = "C:/Users/User/PycharmProjects/service\_with\_mini\_app" # домашняя папка  
  
# Получаем текущую дату в формате дд.мм  
current\_date = datetime.now().strftime("%d.%m")  
output\_file = f"all\_project\_files\_{current\_date}.docx" # Имя выходного файла с датой  
  
# Список допустимых расширений файлов  
ALLOWED\_EXTENSIONS = {'.html', '.css', '.py', '.yaml', '.yml'}  
ALLOWED\_FILENAMES = {'Dockerfile'} # Точное имя файла без расширения  
  
# Создаём новый документ Word  
doc = Document()  
  
# Добавляем стиль "Code", если его нет  
styles = doc.styles  
if "Code" not in styles:  
 style = styles.add\_style("Code", WD\_STYLE\_TYPE.PARAGRAPH)  
 style.font.name = "Courier New" # Моноширинный шрифт для кода  
 style.font.size = Pt(10) # Размер шрифта 10pt  
 style.paragraph\_format.space\_after = Pt(6) # Отступ после абзаца 6pt  
  
# Добавляем заголовок документа  
doc.add\_heading("Список файлов проекта", level=1)  
  
# Проходим по всем файлам в папке и её подпапках  
for root, dirs, files in os.walk(folder\_path):  
 # Пропускаем папки \_\_pycache\_\_ и venv  
 if "\_\_pycache\_\_" in dirs:  
 dirs.remove("\_\_pycache\_\_")  
 if "venv" in dirs:  
 dirs.remove("venv")  
  
 for file in files:  
 # Проверяем расширение файла или точное имя  
 file\_extension = os.path.splitext(file)[1].lower() # Получаем расширение файла  
 file\_name = file # Полное имя файла  
  
 if file\_extension in ALLOWED\_EXTENSIONS or file\_name in ALLOWED\_FILENAMES:  
 file\_path = os.path.join(root, file)  
 doc.add\_heading(f"Файл: {file\_path}", level=2)  
  
 # Читаем содержимое файла  
 try:  
 with open(file\_path, "r", encoding="utf-8") as f:  
 content = f.read()  
 doc.add\_paragraph(content, style="Code") # Добавляем содержимое с форматированием  
 except Exception as e:  
 doc.add\_paragraph(f"Ошибка чтения файла {file\_path}: {e}", style="Normal")  
  
 doc.add\_page\_break() # Разделяем файлы новой страницей  
  
# Сохраняем документ  
doc.save(output\_file)  
print(f"Файлы сохранены в {output\_file}")

## Файл: C:/Users/User/PycharmProjects/service\_with\_mini\_app\api\routes.py

from flask import send\_from\_directory, request, jsonify, redirect, url\_for  
from models.entities import User, Order  
from models.database import db\_pool  
  
def init\_api(app, telegram\_app):  
 app.db\_pool = db\_pool  
  
 @app.route('/')  
 def serve\_index():  
 return redirect(url\_for('serve\_miniapp')) # Перенаправление на /miniapp  
  
 @app.route('/miniapp')  
 def serve\_miniapp():  
 return send\_from\_directory('static', 'miniapp.html')  
  
 @app.route('/register', methods=['POST'])  
 async def register():  
 data = request.json  
 telegram\_id = data['telegram\_id']  
 username = data['username']  
 role = data['role']  
 city\_id = data['city\_id']  
 category\_ids = data.get('category\_ids', [])  
 user\_id = User.create(telegram\_id, username, role, city\_id)  
 for cat\_id in category\_ids:  
 User.add\_category(user\_id, cat\_id)  
 return jsonify({"message": "Регистрация завершена", "user\_id": user\_id})  
  
 @app.route('/create\_order', methods=['POST'])  
 async def create\_order():  
 data = request.json  
 order\_id = await Order.create(data['title'], data['description'], data['category\_id'],  
 data['city\_id'], data['start\_date'], data['deadline'],  
 data['customer\_id'])  
 conn = app.db\_pool.getconn()  
 cursor = conn.cursor()  
 cursor.execute(  
 "SELECT telegram\_id FROM users u JOIN user\_categories uc ON u.id = uc.user\_id WHERE uc.category\_id = %s AND u.city\_id = %s AND u.role = 'Исполнитель'",  
 (data['category\_id'], data['city\_id']))  
 executors = cursor.fetchall()  
 db\_pool.putconn(conn)  
 for executor in executors:  
 await telegram\_app.bot.send\_message(chat\_id=executor[0], text=f"Новый заказ #{order\_id}: {data['title']}")  
 return jsonify({"order\_id": order\_id})  
  
 @app.route('/update\_profile', methods=['POST'])  
 async def update\_profile():  
 data = request.json  
 telegram\_id = data['telegram\_id']  
 city\_id = data['city\_id']  
 category\_ids = data.get('category\_ids', [])  
 User.update\_profile(telegram\_id, city\_id, category\_ids)  
 return jsonify({"message": "Профиль обновлён"})  
  
 @app.route('/get\_profile', methods=['POST'])  
 async def get\_profile():  
 data = request.json  
 telegram\_id = data['telegram\_id']  
 profile = User.get\_profile(telegram\_id)  
 if profile:  
 return jsonify(profile)  
 return jsonify({"error": "Пользователь не найден"}), 404

## Файл: C:/Users/User/PycharmProjects/service\_with\_mini\_app\api\\_\_init\_\_.py

## Файл: C:/Users/User/PycharmProjects/service\_with\_mini\_app\bot\admin.py

from telegram import Update  
from telegram.ext import CommandHandler, ContextTypes  
from models.entities import User, City, Category  
  
ADMIN\_ID = "YOUR\_TELEGRAM\_ID" # Замените на ваш Telegram ID  
  
async def admin\_panel(update: Update, context: ContextTypes.DEFAULT\_TYPE):  
 if str(update.message.from\_user.id) != ADMIN\_ID:  
 await update.message.reply\_text("Доступ запрещён.")  
 return  
 await update.message.reply\_text("Админ-панель: /add\_city, /del\_city, /add\_category, /del\_category, /del\_user")  
  
async def add\_city(update: Update, context: ContextTypes.DEFAULT\_TYPE):  
 if str(update.message.from\_user.id) != ADMIN\_ID:  
 return  
 city\_name = " ".join(context.args)  
 City.create(city\_name)  
 await update.message.reply\_text(f"Город {city\_name} добавлен.")  
  
# Другие админ-команды аналогично...  
  
def setup\_admin\_handlers(application):  
 application.add\_handler(CommandHandler("admin", admin\_panel))  
 application.add\_handler(CommandHandler("add\_city", add\_city))  
 # Добавьте остальные команды: del\_city, add\_category, etc.

## Файл: C:/Users/User/PycharmProjects/service\_with\_mini\_app\bot\handlers.py

from telegram import Update, InlineKeyboardButton, InlineKeyboardMarkup  
from telegram.ext import CommandHandler, ConversationHandler, MessageHandler, ContextTypes, filters  
from models.entities import User  
from config import MINI\_APP\_URL # Импорт из config  
  
ROLE, CITY, CATEGORY = range(3)  
  
async def start(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> int:  
 keyboard = [[InlineKeyboardButton("Регистрация и профиль в Mini App", web\_app={"url": MINI\_APP\_URL})]]  
 reply\_markup = InlineKeyboardMarkup(keyboard)  
 await update.message.reply\_text("Добро пожаловать! Зарегистрируйтесь или управляйте профилем через Mini App:", reply\_markup=reply\_markup)  
 return ROLE  
  
async def set\_role(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> int:  
 role = update.message.text  
 if role not in ["Заказчик", "Исполнитель"]:  
 await update.message.reply\_text("Выберите: Заказчик или Исполнитель")  
 return ROLE  
 context.user\_data['role'] = role  
 await update.message.reply\_text("Регистрация продолжится в Mini App.")  
 return ConversationHandler.END  
  
def setup\_handlers(application):  
 conv\_handler = ConversationHandler(  
 entry\_points=[CommandHandler('start', start)],  
 states={  
 ROLE: [MessageHandler(filters.TEXT & ~filters.COMMAND, set\_role)],  
 },  
 fallbacks=[]  
 )  
 application.add\_handler(conv\_handler)

## Файл: C:/Users/User/PycharmProjects/service\_with\_mini\_app\bot\\_\_init\_\_.py

## Файл: C:/Users/User/PycharmProjects/service\_with\_mini\_app\models\database.py

import psycopg2  
from psycopg2 import pool  
import os  
  
DB\_CONFIG = {  
 'dbname': os.getenv('DB\_NAME', 'service\_bot\_db'),  
 'user': os.getenv('DB\_USER', 'postgres'),  
 'password': os.getenv('DB\_PASSWORD', 'your\_password'),  
 'host': os.getenv('DB\_HOST', 'localhost'),  
 'port': os.getenv('DB\_PORT', '5432')  
}  
  
db\_pool = psycopg2.pool.SimpleConnectionPool(1, 20, \*\*DB\_CONFIG)  
  
def init\_db():  
 conn = db\_pool.getconn()  
 cursor = conn.cursor()  
 cursor.execute('''CREATE TABLE IF NOT EXISTS cities (  
 id SERIAL PRIMARY KEY, name TEXT UNIQUE NOT NULL)''')  
 cursor.execute('''CREATE TABLE IF NOT EXISTS categories (  
 id SERIAL PRIMARY KEY, name TEXT UNIQUE NOT NULL)''')  
 cursor.execute('''CREATE TABLE IF NOT EXISTS users (  
 id SERIAL PRIMARY KEY, telegram\_id TEXT UNIQUE NOT NULL, username TEXT,   
 role TEXT NOT NULL, city\_id INTEGER REFERENCES cities(id),   
 rating REAL DEFAULT 0, completed\_orders INTEGER DEFAULT 0)''')  
 cursor.execute('''CREATE TABLE IF NOT EXISTS user\_categories (  
 user\_id INTEGER REFERENCES users(id), category\_id INTEGER REFERENCES categories(id),   
 PRIMARY KEY (user\_id, category\_id))''')  
 cursor.execute('''CREATE TABLE IF NOT EXISTS orders (  
 id SERIAL PRIMARY KEY, title TEXT NOT NULL, description TEXT NOT NULL,   
 category\_id INTEGER REFERENCES categories(id), city\_id INTEGER REFERENCES cities(id),   
 price REAL, deadline TEXT, start\_date TEXT, status TEXT NOT NULL,   
 customer\_id INTEGER REFERENCES users(id), executor\_id INTEGER REFERENCES users(id))''')  
 cursor.execute("INSERT INTO cities (name) VALUES ('Алматы'), ('Астана'), ('Шымкент') ON CONFLICT DO NOTHING")  
 cursor.execute("INSERT INTO categories (name) VALUES ('Сантехника'), ('IT'), ('Уборка') ON CONFLICT DO NOTHING")  
 conn.commit()  
 db\_pool.putconn(conn)

## Файл: C:/Users/User/PycharmProjects/service\_with\_mini\_app\models\entities.py

from models.database import db\_pool  
  
class User:  
 @staticmethod  
 def create(telegram\_id, username, role, city\_id):  
 conn = db\_pool.getconn()  
 cursor = conn.cursor()  
 cursor.execute("INSERT INTO users (telegram\_id, username, role, city\_id) VALUES (%s, %s, %s, %s) RETURNING id",  
 (telegram\_id, username, role, city\_id))  
 user\_id = cursor.fetchone()[0]  
 conn.commit()  
 db\_pool.putconn(conn)  
 return user\_id  
  
 @staticmethod  
 def add\_category(user\_id, category\_id):  
 conn = db\_pool.getconn()  
 cursor = conn.cursor()  
 cursor.execute("INSERT INTO user\_categories (user\_id, category\_id) VALUES (%s, %s) ON CONFLICT DO NOTHING",  
 (user\_id, category\_id))  
 conn.commit()  
 db\_pool.putconn(conn)  
  
 @staticmethod  
 def update\_profile(telegram\_id, city\_id, category\_ids):  
 conn = db\_pool.getconn()  
 cursor = conn.cursor()  
 # Обновляем город  
 cursor.execute("UPDATE users SET city\_id = %s WHERE telegram\_id = %s", (city\_id, telegram\_id))  
 # Удаляем старые категории  
 cursor.execute("DELETE FROM user\_categories WHERE user\_id = (SELECT id FROM users WHERE telegram\_id = %s)", (telegram\_id,))  
 # Добавляем новые категории  
 cursor.execute("SELECT id FROM users WHERE telegram\_id = %s", (telegram\_id,))  
 user\_id = cursor.fetchone()[0]  
 for cat\_id in category\_ids:  
 cursor.execute("INSERT INTO user\_categories (user\_id, category\_id) VALUES (%s, %s) ON CONFLICT DO NOTHING",  
 (user\_id, cat\_id))  
 conn.commit()  
 db\_pool.putconn(conn)  
  
 @staticmethod  
 def get\_profile(telegram\_id):  
 conn = db\_pool.getconn()  
 cursor = conn.cursor()  
 cursor.execute("SELECT id, role, city\_id FROM users WHERE telegram\_id = %s", (telegram\_id,))  
 user = cursor.fetchone()  
 if not user:  
 db\_pool.putconn(conn)  
 return None  
 user\_id, role, city\_id = user  
 cursor.execute("SELECT category\_id FROM user\_categories WHERE user\_id = %s", (user\_id,))  
 category\_ids = [row[0] for row in cursor.fetchall()]  
 db\_pool.putconn(conn)  
 return {"user\_id": user\_id, "role": role, "city\_id": city\_id, "category\_ids": category\_ids}  
  
class Order:  
 @staticmethod  
 async def create(title, description, category\_id, city\_id, start\_date, deadline, customer\_id):  
 conn = db\_pool.getconn()  
 cursor = conn.cursor()  
 cursor.execute('''INSERT INTO orders (title, description, category\_id, city\_id, start\_date, deadline, status, customer\_id)  
 VALUES (%s, %s, %s, %s, %s, %s, 'Открыт', %s) RETURNING id''',  
 (title, description, category\_id, city\_id, start\_date, deadline, customer\_id))  
 order\_id = cursor.fetchone()[0]  
 conn.commit()  
 db\_pool.putconn(conn)  
 return order\_id

## Файл: C:/Users/User/PycharmProjects/service\_with\_mini\_app\models\\_\_init\_\_.py

## Файл: C:/Users/User/PycharmProjects/service\_with\_mini\_app\static\miniapp.html

<!DOCTYPE html>  
<html lang="ru">  
<head>  
 <meta charset="UTF-8">  
 <title>Mini App</title>  
 <script src="https://telegram.org/js/telegram-web-app.js"></script>  
 <style>  
 body { font-family: Arial, sans-serif; padding: 20px; background: var(--tg-theme-bg-color); color: var(--tg-theme-text-color); }  
 label { display: block; margin: 10px 0 5px; }  
 input, select { width: 100%; padding: 8px; margin-bottom: 10px; background: var(--tg-theme-secondary-bg-color); color: var(--tg-theme-text-color); border: 1px solid var(--tg-theme-hint-color); }  
 button { background: var(--tg-theme-button-color); color: var(--tg-theme-button-text-color); padding: 10px; border: none; width: 100%; }  
 .section { margin-top: 20px; }  
 </style>  
</head>  
<body>  
 <div id="registerSection" class="section">  
 <h2>Регистрация</h2>  
 <form id="registerForm">  
 <label>Роль:</label>  
 <select id="role" required>  
 <option value="Заказчик">Заказчик</option>  
 <option value="Исполнитель">Исполнитель</option>  
 </select>  
 <label>Город:</label>  
 <select id="city" required>  
 <option value="1">Алматы</option>  
 <option value="2">Астана</option>  
 <option value="3">Шымкент</option>  
 </select>  
 <label>Категории (до 10):</label>  
 <select id="categories" multiple>  
 <option value="1">Сантехника</option>  
 <option value="2">IT</option>  
 <option value="3">Уборка</option>  
 </select>  
 <button type="submit">Зарегистрироваться</button>  
 </form>  
 </div>  
  
 <div id="profileSection" class="section" style="display:none;">  
 <h2>Редактировать профиль</h2>  
 <form id="profileForm">  
 <label>Город:</label>  
 <select id="profile\_city" required>  
 <option value="1">Алматы</option>  
 <option value="2">Астана</option>  
 <option value="3">Шымкент</option>  
 </select>  
 <label>Категории (до 10):</label>  
 <select id="profile\_categories" multiple>  
 <option value="1">Сантехника</option>  
 <option value="2">IT</option>  
 <option value="3">Уборка</option>  
 </select>  
 <button type="submit">Сохранить изменения</button>  
 </form>  
 </div>  
  
 <div id="orderSection" class="section" style="display:none;">  
 <h2>Создать заказ</h2>  
 <form id="orderForm">  
 <label>Название:</label>  
 <input type="text" id="title" required>  
 <label>Описание:</label>  
 <input type="text" id="description" required>  
 <label>Категория:</label>  
 <select id="order\_category" required>  
 <option value="1">Сантехника</option>  
 <option value="2">IT</option>  
 <option value="3">Уборка</option>  
 </select>  
 <label>Город:</label>  
 <select id="order\_city" required>  
 <option value="1">Алматы</option>  
 <option value="2">Астана</option>  
 <option value="3">Шымкент</option>  
 </select>  
 <label>Дата начала:</label>  
 <input type="date" id="start\_date" required>  
 <label>Срок:</label>  
 <select id="deadline" required>  
 <option value="1 час">1 час</option>  
 <option value="1 день">1 день</option>  
 <option value="1 неделя">1 неделя</option>  
 </select>  
 <button type="submit">Создать заказ</button>  
 </form>  
 </div>  
  
 <script>  
 const tg = window.Telegram.WebApp;  
 tg.ready();  
 const user = tg.initDataUnsafe.user;  
  
 // Проверка, зарегистрирован ли пользователь  
 async function checkProfile() {  
 const response = await fetch('https://fa66-88-204-232-102.ngrok-free.app/get\_profile', {  
 method: 'POST',  
 headers: { 'Content-Type': 'application/json' },  
 body: JSON.stringify({ telegram\_id: user.id })  
 });  
 const profile = await response.json();  
 if (response.ok && profile) {  
 document.getElementById('registerSection').style.display = 'none';  
 document.getElementById('profileSection').style.display = 'block';  
 document.getElementById('orderSection').style.display = 'block';  
 document.getElementById('profile\_city').value = profile.city\_id;  
 const categories = document.getElementById('profile\_categories');  
 profile.category\_ids.forEach(cat => {  
 for (let option of categories.options) {  
 if (option.value == cat) option.selected = true;  
 }  
 });  
 }  
 }  
 checkProfile();  
  
 // Регистрация  
 document.getElementById('registerForm').addEventListener('submit', async (e) => {  
 e.preventDefault();  
 const data = {  
 telegram\_id: user.id,  
 username: user.username,  
 role: document.getElementById('role').value,  
 city\_id: document.getElementById('city').value,  
 category\_ids: Array.from(document.getElementById('categories').selectedOptions).map(opt => opt.value)  
 };  
 const response = await fetch('https://fa66-88-204-232-102.ngrok-free.app/register', {  
 method: 'POST',  
 headers: { 'Content-Type': 'application/json' },  
 body: JSON.stringify(data)  
 });  
 const result = await response.json();  
 if (response.ok) {  
 tg.showAlert('Регистрация завершена!');  
 document.getElementById('registerSection').style.display = 'none';  
 document.getElementById('profileSection').style.display = 'block';  
 document.getElementById('orderSection').style.display = 'block';  
 } else {  
 tg.showAlert('Ошибка: ' + result.error);  
 }  
 });  
  
 // Редактирование профиля  
 document.getElementById('profileForm').addEventListener('submit', async (e) => {  
 e.preventDefault();  
 const data = {  
 telegram\_id: user.id,  
 city\_id: document.getElementById('profile\_city').value,  
 category\_ids: Array.from(document.getElementById('profile\_categories').selectedOptions).map(opt => opt.value)  
 };  
 const response = await fetch('https://fa66-88-204-232-102.ngrok-free.app/update\_profile', {  
 method: 'POST',  
 headers: { 'Content-Type': 'application/json' },  
 body: JSON.stringify(data)  
 });  
 const result = await response.json();  
 if (response.ok) {  
 tg.showAlert('Профиль обновлён!');  
 } else {  
 tg.showAlert('Ошибка: ' + result.error);  
 }  
 });  
  
 // Создание заказа  
 document.getElementById('orderForm').addEventListener('submit', async (e) => {  
 e.preventDefault();  
 const data = {  
 title: document.getElementById('title').value,  
 description: document.getElementById('description').value,  
 category\_id: document.getElementById('order\_category').value,  
 city\_id: document.getElementById('order\_city').value,  
 start\_date: document.getElementById('start\_date').value,  
 deadline: document.getElementById('deadline').value,  
 customer\_id: user.id  
 };  
 const response = await fetch('https://fa66-88-204-232-102.ngrok-free.app/create\_order', {  
 method: 'POST',  
 headers: { 'Content-Type': 'application/json' },  
 body: JSON.stringify(data)  
 });  
 const result = await response.json();  
 if (response.ok) {  
 tg.showAlert(`Заказ #${result.order\_id} создан!`);  
 tg.close();  
 } else {  
 tg.showAlert('Ошибка: ' + result.error);  
 }  
 });  
 </script>  
</body>  
</html>

## Файл: C:/Users/User/PycharmProjects/service\_with\_mini\_app\static\static.html

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<body>  
  
</body>  
</html>

## Файл: C:/Users/User/PycharmProjects/service\_with\_mini\_app\static\styles.css

body {  
 font-family: Arial, sans-serif;  
 background-color: var(--tg-theme-bg-color, #ffffff);  
 color: var(--tg-theme-text-color, #000000);  
 margin: 0;  
 padding: 20px;  
}  
  
.container {  
 max-width: 600px;  
 margin: 0 auto;  
}  
  
h1 {  
 text-align: center;  
 color: var(--tg-theme-hint-color, #888888);  
}  
  
form {  
 display: flex;  
 flex-direction: column;  
 gap: 15px;  
}  
  
label {  
 font-weight: bold;  
}  
  
input, select {  
 padding: 10px;  
 font-size: 16px;  
 border: 1px solid var(--tg-theme-hint-color, #cccccc);  
 border-radius: 5px;  
 background-color: var(--tg-theme-secondary-bg-color, #f0f0f0);  
 color: var(--tg-theme-text-color, #000000);  
}  
  
#message {  
 margin-top: 20px;  
 text-align: center;  
 color: var(--tg-theme-link-color, #007bff);  
}