# Список файлов проекта

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

import psycopg2  
from psycopg2 import pool  
from flask import Flask, request, jsonify, send\_from\_directory  
from telegram import Update, InlineKeyboardButton, InlineKeyboardMarkup  
from telegram.ext import Application, CommandHandler, ConversationHandler, MessageHandler, ContextTypes  
from telegram.ext import filters  
import asyncio  
import os  
from threading import Thread  
  
# Настройки Flask  
app = Flask(\_\_name\_\_, static\_folder='static')  
  
# Настройки Telegram Bot  
TOKEN = os.getenv('TELEGRAM\_TOKEN', '7661263528:AAHB4LKirWI6Xtw\_MRIgrzQPqq22Xz-\_AUI')  
application = Application.builder().token(TOKEN).build()  
  
# URL для Mini App (замените на ваш ngrok-адрес)  
MINI\_APP\_URL = "https://fa66-88-204-232-102.ngrok-free.app/miniapp" # Замените на ваш HTTPS-адрес от ngrok  
  
# Настройки PostgreSQL  
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')  
}  
  
# Пул соединений для PostgreSQL  
db\_pool = psycopg2.pool.SimpleConnectionPool(1, 20, \*\*DB\_CONFIG)  
  
# Состояния для регистрации  
ROLE, CITY, CATEGORY = range(3)  
  
  
# Инициализация базы данных  
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, name TEXT NOT NULL,   
 role TEXT NOT NULL, city INTEGER REFERENCES cities(id),   
 category INTEGER REFERENCES categories(id), rating REAL DEFAULT 0,   
 completed\_orders INTEGER DEFAULT 0)''')  
 cursor.execute('''CREATE TABLE IF NOT EXISTS orders (  
 id SERIAL PRIMARY KEY, description TEXT NOT NULL, category INTEGER REFERENCES categories(id),   
 city INTEGER REFERENCES cities(id), price REAL NOT NULL, deadline TEXT NOT NULL,   
 status TEXT NOT NULL, customer\_id INTEGER REFERENCES users(id),   
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP)''')  
 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)  
  
  
# Flask маршруты  
@app.route('/')  
def serve\_index():  
 return send\_from\_directory('static', 'index.html')  
  
  
@app.route('/miniapp')  
def serve\_miniapp():  
 return send\_from\_directory('static', 'miniapp.html')  
  
  
@app.route('/create\_order', methods=['POST'])  
async def create\_order():  
 data = request.json  
 telegram\_id = data.get('telegram\_id')  
 conn = db\_pool.getconn()  
 cursor = conn.cursor()  
 cursor.execute("SELECT id FROM users WHERE telegram\_id = %s", (telegram\_id,))  
 user = cursor.fetchone()  
 if not user:  
 db\_pool.putconn(conn)  
 return jsonify({"error": "Пользователь не зарегистрирован"}), 403  
 customer\_id = user[0]  
 cursor.execute('''  
 INSERT INTO orders (description, category, city, price, deadline, status, customer\_id)  
 VALUES (%s, %s, %s, %s, %s, 'открыт', %s) RETURNING id  
 ''', (data['description'], data['category'], data['city'], data['price'], data['deadline'], customer\_id))  
 order\_id = cursor.fetchone()[0]  
 conn.commit()  
 cursor.execute("SELECT telegram\_id FROM users WHERE role = 'Исполнитель' AND category = %s AND city = %s",  
 (data['category'], data['city']))  
 executors = cursor.fetchall()  
 db\_pool.putconn(conn)  
 for executor in executors:  
 await application.bot.send\_message(chat\_id=executor[0],  
 text=f"Новый заказ #{order\_id}: {data['description']}\nЦена: {data['price']} тг\nСрок: {data['deadline']}")  
 return jsonify({"order\_id": order\_id})  
  
  
# Telegram Bot  
async def start(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> int:  
 keyboard = [[InlineKeyboardButton("Создать заказ в Mini App", web\_app={"url": MINI\_APP\_URL})]]  
 reply\_markup = InlineKeyboardMarkup(keyboard)  
 await update.message.reply\_text("Добро пожаловать! Выберите роль или создайте заказ:", 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("Пожалуйста, выберите: Заказчик или Исполнитель")  
 return ROLE  
 context.user\_data['role'] = role  
 conn = db\_pool.getconn()  
 cursor = conn.cursor()  
 cursor.execute("SELECT id, name FROM cities")  
 cities = cursor.fetchall()  
 db\_pool.putconn(conn)  
 city\_list = "\n".join([f"{c[0]}. {c[1]}" for c in cities])  
 await update.message.reply\_text(f"Выберите город (введите номер):\n{city\_list}")  
 return CITY  
  
  
async def set\_city(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> int:  
 try:  
 city\_id = int(update.message.text)  
 conn = db\_pool.getconn()  
 cursor = conn.cursor()  
 cursor.execute("SELECT id FROM cities WHERE id = %s", (city\_id,))  
 if not cursor.fetchone():  
 await update.message.reply\_text("Неверный номер города, попробуйте снова")  
 db\_pool.putconn(conn)  
 return CITY  
 context.user\_data['city'] = city\_id  
 db\_pool.putconn(conn)  
 if context.user\_data['role'] == "Исполнитель":  
 conn = db\_pool.getconn()  
 cursor = conn.cursor()  
 cursor.execute("SELECT id, name FROM categories")  
 categories = cursor.fetchall()  
 db\_pool.putconn(conn)  
 category\_list = "\n".join([f"{c[0]}. {c[1]}" for c in categories])  
 await update.message.reply\_text(f"Выберите категорию (введите номер):\n{category\_list}")  
 return CATEGORY  
 else:  
 return await save\_user(update, context)  
 except ValueError:  
 await update.message.reply\_text("Введите номер города")  
 return CITY  
  
  
async def set\_category(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> int:  
 try:  
 category\_id = int(update.message.text)  
 conn = db\_pool.getconn()  
 cursor = conn.cursor()  
 cursor.execute("SELECT id FROM categories WHERE id = %s", (category\_id,))  
 if not cursor.fetchone():  
 await update.message.reply\_text("Неверный номер категории, попробуйте снова")  
 db\_pool.putconn(conn)  
 return CATEGORY  
 context.user\_data['category'] = category\_id  
 db\_pool.putconn(conn)  
 return await save\_user(update, context)  
 except ValueError:  
 await update.message.reply\_text("Введите номер категории")  
 return CATEGORY  
  
  
async def save\_user(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> int:  
 conn = db\_pool.getconn()  
 cursor = conn.cursor()  
 telegram\_id = str(update.message.from\_user.id)  
 name = update.message.from\_user.first\_name  
 role = context.user\_data['role']  
 city = context.user\_data['city']  
 category = context.user\_data.get('category', None)  
 cursor.execute('''  
 INSERT INTO users (telegram\_id, name, role, city, category, rating, completed\_orders)  
 VALUES (%s, %s, %s, %s, %s, 0, 0)  
 ON CONFLICT (telegram\_id) DO UPDATE  
 SET name = %s, role = %s, city = %s, category = %s, rating = 0, completed\_orders = 0  
 ''', (telegram\_id, name, role, city, category, name, role, city, category))  
 conn.commit()  
 db\_pool.putconn(conn)  
 await update.message.reply\_text(f"Регистрация завершена!\nРоль: {role}\nГород: {city}")  
 return ConversationHandler.END  
  
  
# Настройка бота  
conv\_handler = ConversationHandler(  
 entry\_points=[CommandHandler('start', start)],  
 states={  
 ROLE: [MessageHandler(filters.TEXT & ~filters.COMMAND, set\_role)],  
 CITY: [MessageHandler(filters.TEXT & ~filters.COMMAND, set\_city)],  
 CATEGORY: [MessageHandler(filters.TEXT & ~filters.COMMAND, set\_category)],  
 },  
 fallbacks=[]  
)  
  
application.add\_handler(conv\_handler)  
  
  
# Запуск Flask в отдельном потоке  
def run\_flask():  
 app.run(debug=True, port=5000, use\_reloader=False)  
  
  
# Запуск приложения  
if \_\_name\_\_ == '\_\_main\_\_':  
 init\_db()  
 flask\_thread = Thread(target=run\_flask)  
 flask\_thread.start()  
 asyncio.run(application.run\_polling())

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

<!DOCTYPE html>  
<html lang="ru">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Сервис услуг</title>  
 <link rel="stylesheet" href="styles.css">  
 <script src="https://telegram.org/js/telegram-web-app.js"></script>  
</head>  
<body>  
 <div class="container">  
 <h1>Создать заказ</h1>  
 <form id="orderForm">  
 <label for="description">Описание:</label>  
 <input type="text" id="description" required>  
 <label for="category">Категория:</label>  
 <select id="category" required>  
 <option value="1">Сантехника</option>  
 <option value="2">IT</option>  
 <option value="3">Уборка</option>  
 </select>  
 <label for="city">Город:</label>  
 <select id="city" required>  
 <option value="1">Алматы</option>  
 <option value="2">Астана</option>  
 <option value="3">Шымкент</option>  
 </select>  
 <label for="price">Цена (тг):</label>  
 <input type="number" id="price" required>  
 <label for="deadline">Срок:</label>  
 <input type="text" id="deadline" required placeholder="например, 2 дня">  
 </form>  
 <div id="message"></div>  
 </div>  
 <script src="script.js"></script>  
</body>  
</html>

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

<!DOCTYPE html>  
<html lang="ru">  
<head>  
 <meta charset="UTF-8">  
 <title>Создать заказ</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%; }  
 </style>  
</head>  
<body>  
 <h2>Создать заказ</h2>  
 <form id="orderForm">  
 <label>Описание:</label>  
 <input type="text" id="description" required>  
 <label>Категория:</label>  
 <select id="category">  
 <option value="1">Сантехника</option>  
 <option value="2">IT</option>  
 <option value="3">Уборка</option>  
 </select>  
 <label>Город:</label>  
 <select id="city">  
 <option value="1">Алматы</option>  
 <option value="2">Астана</option>  
 <option value="3">Шымкент</option>  
 </select>  
 <label>Цена (тг):</label>  
 <input type="number" id="price" required>  
 <label>Срок выполнения:</label>  
 <input type="text" id="deadline" required>  
 <button type="submit">Создать заказ</button>  
 </form>  
  
 <script>  
 const tg = window.Telegram.WebApp;  
 tg.ready();  
  
 document.getElementById('orderForm').addEventListener('submit', async (e) => {  
 e.preventDefault();  
 const data = {  
 telegram\_id: tg.initDataUnsafe.user.id,  
 description: document.getElementById('description').value,  
 category: document.getElementById('category').value,  
 city: document.getElementById('city').value,  
 price: document.getElementById('price').value,  
 deadline: document.getElementById('deadline').value  
 };  
  
 const response = await fetch('https://fa66-88-204-232-102.ngrok-free.app/create\_order', { // Замените на ваш ng Wrok-адрес  
 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/Nurba/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);  
}