Разработка веб-приложения для загрузки данных в БД PostgreSQL на языке Python с использованием веб-фреймворка Flask

Описание программы

Разработанный загрузчик данных представляет собой веб-приложение для создания и заполнения таблиц в БД PostgreSQL. Исходные данные представлены в виде csv-файлов. Результатом работы программы является БД PostgreSQL, заполненная таблицами и данными в них.

В работе были использованы инструменты:

Docker, СУБД PostgreSQL через отдельный Docker образ, DBeaver для работы с БД, Pycharm для разработки программного кода, веб-фреймворк Flask, языки Python, HTML, SQL.

Исходные данные:

Исходные данные хранятся в виде файлов в формате .csv и размещены в открытом доступе в облачном хранилище. Скачать данные можно по ссылке:

https://drive.google.com/drive/folders/1xvuoSyLWTtQfDBjUancJyzCUltc9WeQJ?usp=drive_link

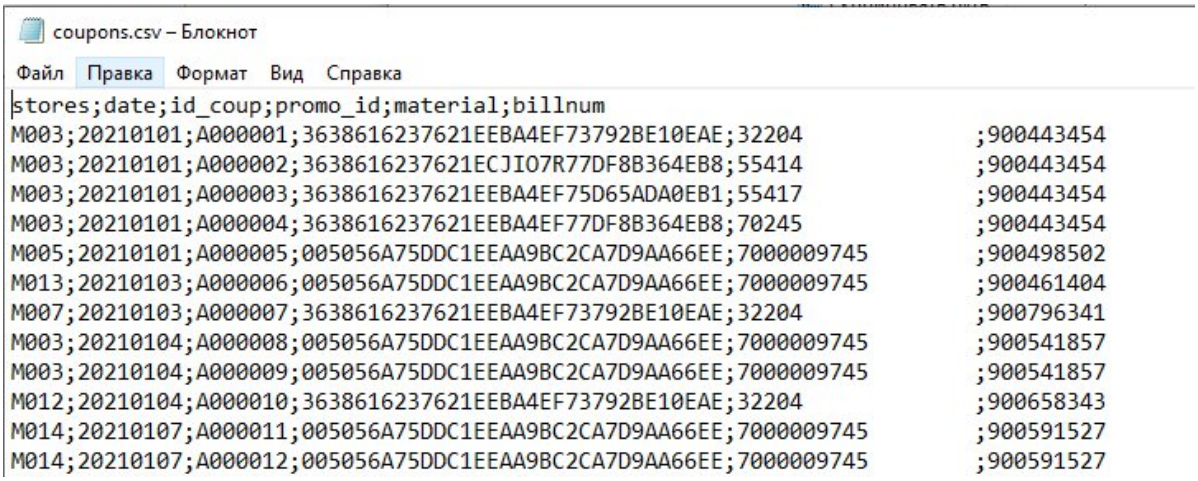


Рисунок 1 – Пример исходного файла .csv

База данных «stores_bd»

Для создания БД развернули СУБД PostgreSQL из Docker-контейнера, настроили учетную запись и сетевой доступ при помощи DBeaver, создали пустую базу данных stores_bd.

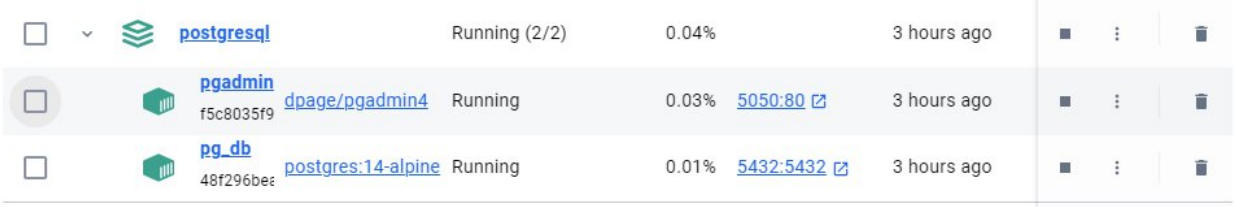


Рисунок 2 – Развернули СУБД PostgreSQL из Docker-контейнера

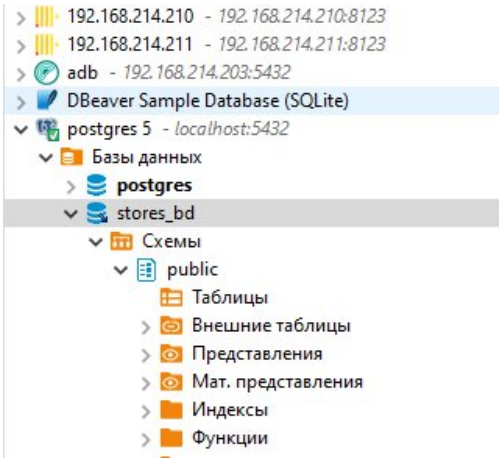


Рисунок 3 – Пустая БД stores_bd в DBeaver

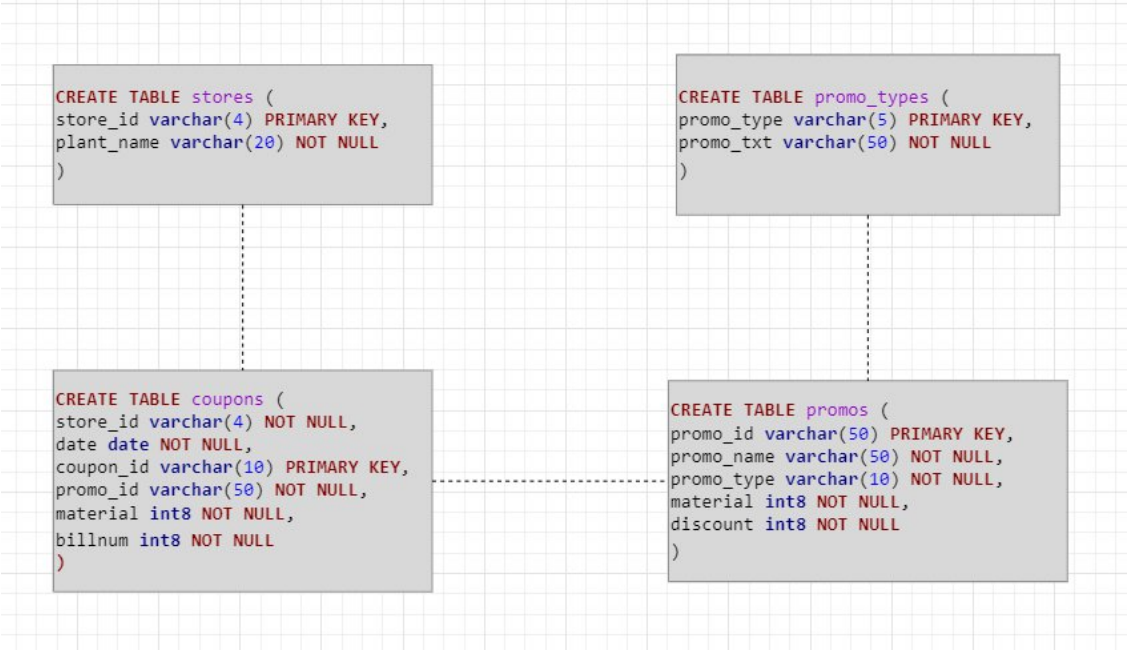


Рисунок 4 – Схема БД stores_bd

База данных stores_bd после заполнения должна включать в себя таблицы:
stores – данные о магазинах сети,
coupons – примененные купоны с промоакциями,
promos – данные о промоакциях сети,
promo_types – типы промоакций.

Программный код

Программа была реализована в среде разработки PyCharm с использованием Flask - веб-фреймворка для Python.

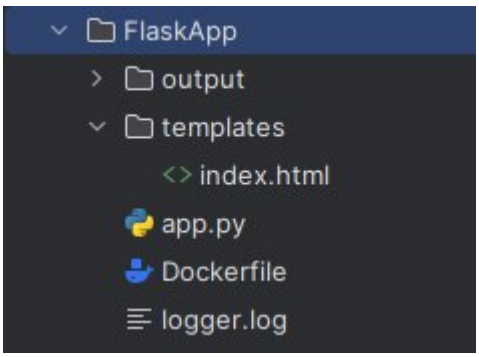


Рисунок 5- Структура приложения

Было развернуто окружение для Python с требуемыми библиотеками:

psycopg2 – для взаимодействия с PostgreSQL
flask – библиотека фреймворка Flask для создания веб-приложений
logging – для логирования
webbrowser – контроллер веб-браузера
urllib.request – для работы с URL-адресами

Основной файл проекта app.py:

```
#!/usr/bin/python
# -*- coding: cp1251 -*-
import psycopg2
from flask import Flask, render_template, render_template_string
import logging
import webbrowser
import urllib.request

logger = logging.getLogger()
logging.basicConfig(filename='logger.log', level=logging.ERROR,
                    format=f'%(asctime)s %(levelname)s %(name)s: %(message)s')


app = Flask(__name__)

@app.route('/')
@app.route('/index')
def main():
    return render_template('index.html', encoding='utf-8')

def load_stores(file, conn, cur):
    cur.execute("""DROP TABLE IF EXISTS stores""")
    cur.execute("""
        CREATE TABLE IF NOT EXISTS stores(
        store_id varchar(4) PRIMARY KEY,
        store_name varchar(20) NOT NULL
    )
    """)
    conn.commit()
```

```
    with open(file, 'r', encoding='utf-8') as f:
        next(f)  # Skip the header row
        cur.copy_from(f, 'stores', sep=';')
    conn.commit()

def load_coupons(file, conn, cur):
    cur.execute("""DROP TABLE IF EXISTS coupons""")
    cur.execute("""
        CREATE TABLE IF NOT EXISTS coupons(
        store_id varchar(4) NOT NULL,
        date date NOT NULL,
        coupon_id varchar(10) PRIMARY KEY,
        promo_id varchar(50) NOT NULL,
        material int8 NOT NULL,
        billnum int8 NOT NULL
    )
    """)
    conn.commit()

    with open(file, 'r', encoding='utf-8') as f:
        next(f)  # Skip the header row
        cur.copy_from(f, 'coupons', sep=';')
    conn.commit()

def load_promos(file, conn, cur):
    cur.execute("""DROP TABLE IF EXISTS promos""")
    cur.execute("""
        CREATE TABLE IF NOT EXISTS promos(
        promo_id varchar(50) PRIMARY KEY,
        promo_name varchar(50) NOT NULL,
        promo_type varchar(10) NOT NULL,
        material int8 NOT NULL,
        discount int8 NOT NULL
        )
    """)
    conn.commit()

    with open(file, 'r', encoding='utf-8') as f:
        next(f)  # Skip the header row
        cur.copy_from(f, 'promos', sep=';')
    conn.commit()

def load_promo_types(file, conn, cur):
    cur.execute("""DROP TABLE IF EXISTS promo_types""")
    cur.execute("""
        CREATE TABLE IF NOT EXISTS promo_types(
        promo_type varchar PRIMARY KEY,
        promo_txt varchar NOT NULL
        )
    """)
    conn.commit()
```

```
    with open(file, 'r', encoding='utf-8') as f:
        next(f)  # Skip the header row
        cur.copy_from(f, 'promo_types', sep=';')
    conn.commit()

@app.route('/stores')
def stores():
    stores_file = 'stores.csv'
    stores_url = 'https://drive.google.com/uc?export=download&id=1QdFHOfPwaXP4Tcsl4qC8q_9zg2zLDPrL'
    return load(stores_url, stores_file)


@app.route('/coupons')
def coupons():
    coupons_file = 'coupons.csv'
    coupons_url = 'https://drive.google.com/uc?export=download&id=1K7NJeR1CJgXwDFykQ7tjND4ZZcOXL65S'
    return load(coupons_url, coupons_file)

@app.route('/promos')
def promos():
    promos_file = 'promos.csv'
    promos_url = 'https://drive.google.com/uc?export=download&id=15S7GkKpOcm3174jS_QFfHAdqAbvpZbFY'
    return load(promos_url, promos_file)

@app.route('/promo_types')
def promo_types():
    pt_file = 'promo_types.csv'
    pt_url = 'https://drive.google.com/uc?export=download&id=1qenV8oCW14AKQuYHPTGPoDSu-XAqfn1k'
    return load(pt_url, pt_file)

def load(url, file):
    try:
        conn = psycopg2.connect("host=localhost port=5432 dbname=stores_bd user=admin password=admin")
        cur = conn.cursor()
        urllib.request.urlretrieve(url, file)
        match file:
            case "stores.csv":
                load_stores(file, conn, cur)
            case "coupons.csv":
                load_coupons(file, conn, cur)
            case "promos.csv":
                load_promos(file, conn, cur)
            case "promo_types.csv":
                load_promo_types(file, conn, cur)
        logger.info('Success!')
        return render_template_string('<h2 align="center">Данные загружены успешно.</br>'
                                      '<a href="/index">Вернуться на главную</a></h2>')

    except Exception as e:
        logger.error('Error! Exception %s', e)
        return render_template_string('<h2 align="center">Данные не загружены.</br>'
                                      '<a href="/index">Вернуться на главную</a></h2>')
```

```
if __name__ == "__main__":
    webbrowser.open('http://127.0.0.1:5000')
    app.run(debug=False)
```

В директории проекта находится папка templates, которая содержит шаблоны с расширением .html. Главная страница — index.html:

```
{% block body %}

<div class="title"><div align="center">
    <h1>Загрузчик данных</h1></div>
</div>

<div align="center">
    <form action="" method="post">
        <p>
       <a class="btn" href="/stores" target="_blank">Загрузить таблицу магазинов</a>
          <br>
       <a class="btn" href="/coupons" target="_blank">Загрузить таблицу купонов</a>
          <br>
       <a class="btn" href="/promos" target="_blank">Загрузить таблицу промоакций</a>
          <br>
       <a class="btn" href="/promo_types" target="_blank">Загрузить таблицу типов промоакций</a>
    </p>
    </form>
</div>

{% endblock %}
```

Преобразовали проект в единый исполняемый файл .exe с помощью auto-py-to-exe.

Работа приложения

Для запуска приложения нужно запустить app.exe, после этого автоматически открывается url http://localhost:5000 в браузере.

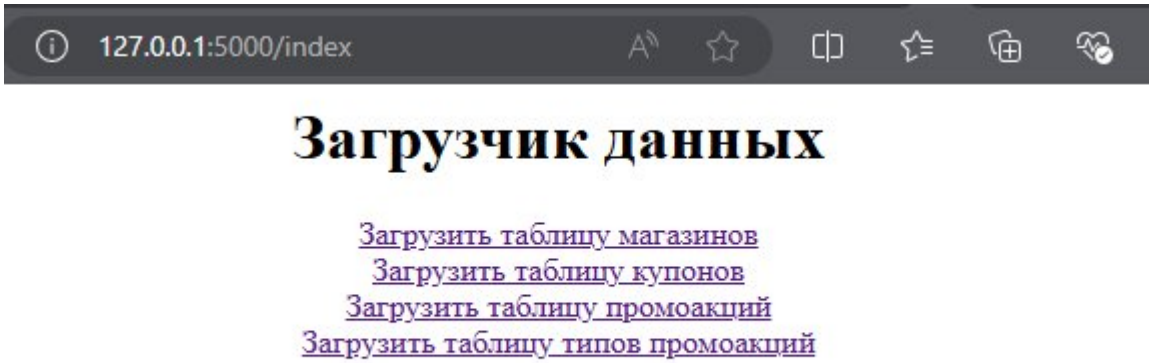


Рисунок 6 – Главная страница веб-приложения

Главная страница содержит кнопки для загрузки данных, при нажатии которых в корневую папку проекта скачивается соответствующий файл из открытого источника. Затем строится соответствующая таблица и загружаются в нее данные из файла csv.

Каждая кнопка соответствует одной из таблиц БД. Если таблица с таким названием уже существует в базе, она удаляется и перезаписывается.

После выполнения функции загрузки данных получаем сообщение о результате.

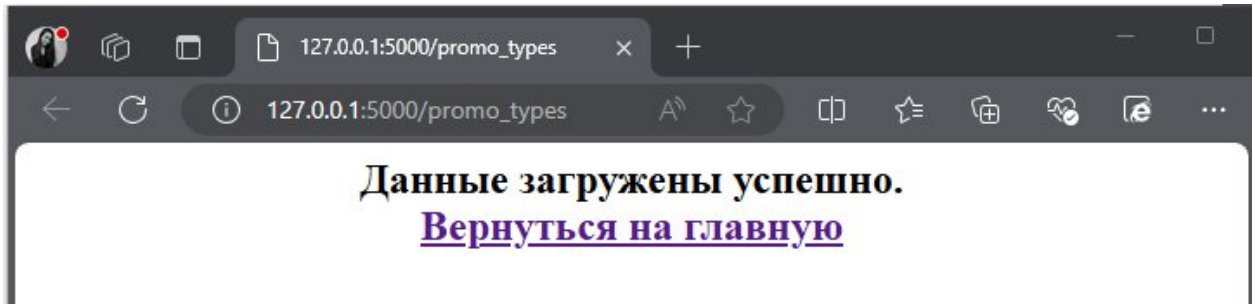


Рисунок 7 – Уведомление об успешной загрузке

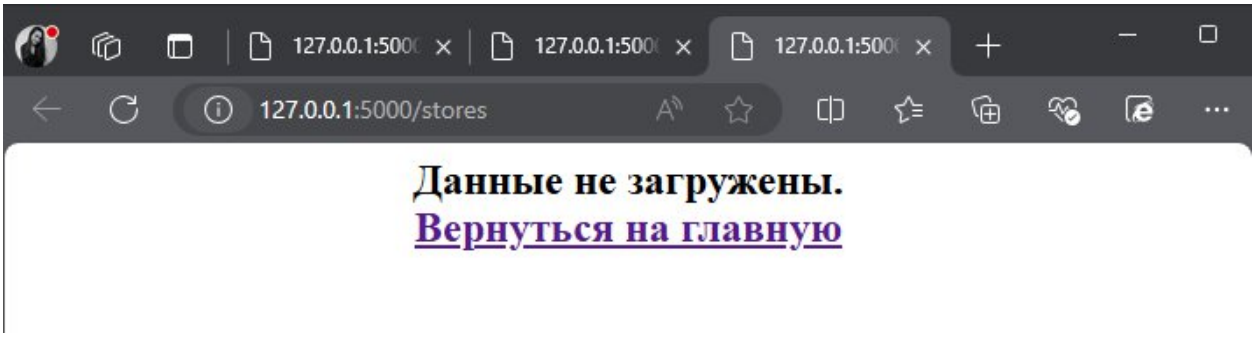


Рисунок 8 – Уведомление об ошибке

Настроена система журнализации ошибок. При возникновении ошибок, можно обратиться к журналу логов для выяснения причин.

```
2024-08-01 01:15:33,749 INFO werkzeug: ESC[31mESC[1mWARNING: This is a development server. Do not use it in a
 * Running on http://127.0.0.1:5000
2024-08-01 01:15:33,749 INFO werkzeug: ESC[33mPress CTRL+C to quitESC[0m
2024-08-01 01:15:33,861 INFO werkzeug: 127.0.0.1 - - [01/Aug/2024 01:15:33] "GET / HTTP/1.1" 200 -
2024-08-01 01:15:40,418 ERROR root: Error! Exception invalid input syntax for type bigint: "Магазин №1"
CONTEXT:  COPY stores, line 1, column store_name: "Магазин №1"

2024-08-01 01:15:40,422 INFO werkzeug: 127.0.0.1 - - [01/Aug/2024 01:15:40] "GET /stores HTTP/1.1" 200 -
```

Рисунок 9 – Журнал логов logger.log

Результатом работы приложения является заполненная таблицами и данными БД stores_bd:

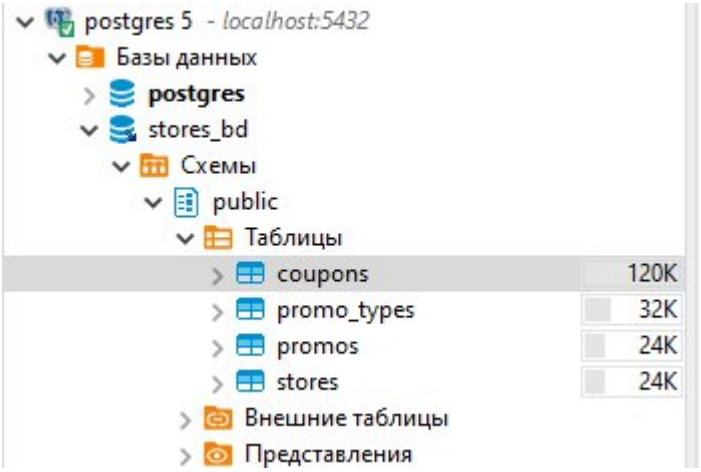
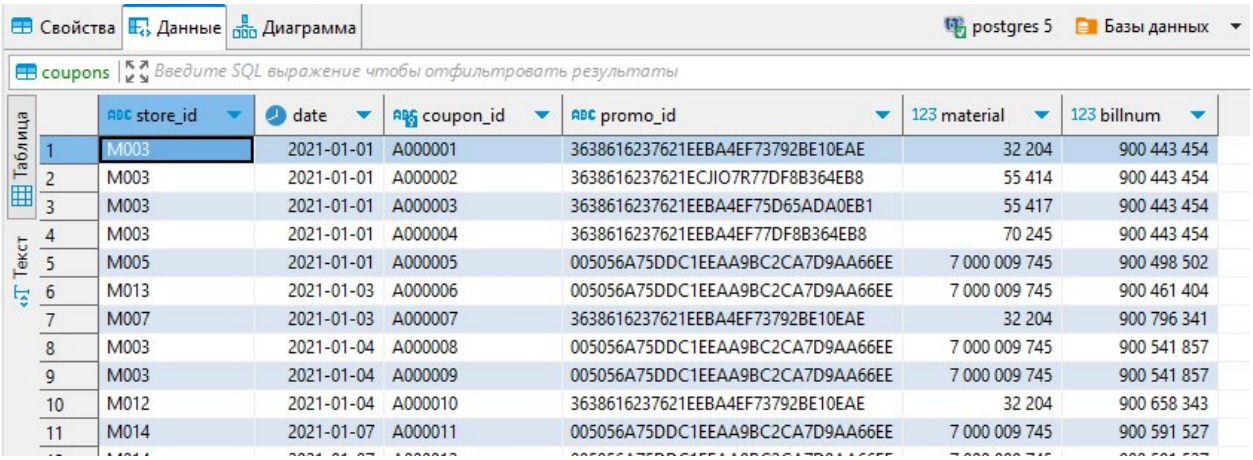


Рисунок 10 – Структура БД stores_bd



	store_id	date	coupon_id	promo_id	material	billnum
1	M003	2021-01-01	A000001	3638616237621EEBA4EF73792BE10EAE	32 204	900 443 454
2	M003	2021-01-01	A000002	3638616237621ECJIO7R77DF8B364EB8	55 414	900 443 454
3	M003	2021-01-01	A000003	3638616237621EEBA4EF75D65ADA0EB1	55 417	900 443 454
4	M003	2021-01-01	A000004	3638616237621EEBA4EF77DF8B364EB8	70 245	900 443 454
5	M005	2021-01-01	A000005	005056A75DDC1EEAA9BC2CA7D9AA66EE	7 000 009 745	900 498 502
6	M013	2021-01-03	A000006	005056A75DDC1EEAA9BC2CA7D9AA66EE	7 000 009 745	900 461 404
7	M007	2021-01-03	A000007	3638616237621EEBA4EF73792BE10EAE	32 204	900 796 341
8	M003	2021-01-04	A000008	005056A75DDC1EEAA9BC2CA7D9AA66EE	7 000 009 745	900 541 857
9	M003	2021-01-04	A000009	005056A75DDC1EEAA9BC2CA7D9AA66EE	7 000 009 745	900 541 857
10	M012	2021-01-04	A000010	3638616237621EEBA4EF73792BE10EAE	32 204	900 658 343
11	M014	2021-01-07	A000011	005056A75DDC1EEAA9BC2CA7D9AA66EE	7 000 009 745	900 591 527

Рисунок 11 – Пример заполнения. Таблица coupons

Создание Dockerfile

Был создан Dockerfile на основе образа python:3.9.19-alpine3.20

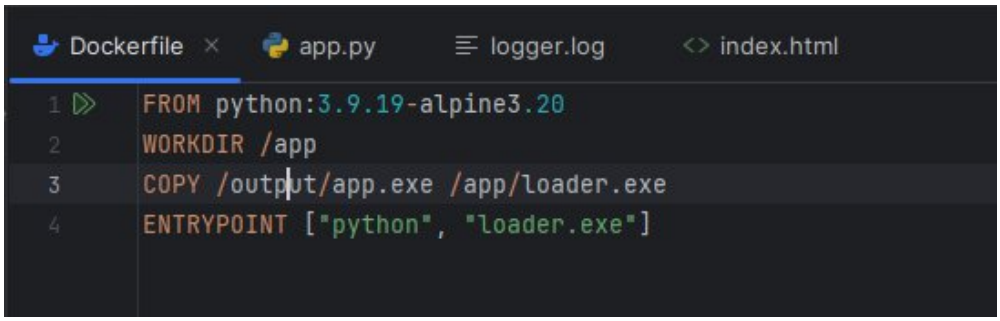


Рисунок 12 – Создание Dockerfile

Запустить не удалось, возникает ошибка

2024-08-01 01:43:21 SyntaxError: Non-UTF-8 code starting with '\x90' in file /app/test.exe on line 1, but no encoding declared; see https://peps.python.org/pep-0263/ for details

Вывод

Разработанное веб-приложение загружает исходные данные из csv-файлов в таблицы в БД PostgreSQL. Исполняемый файл app.exe прилагается.