**Министерство науки и высшего образования Российской Федерации**
федеральное государственное автономное образовательное учреждение высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Отчет**

# «Лабораторная работа №3»

Автор: Кононов Степан

Факультет: ИКТ

Группа: К33392

Санкт-Петербург 2024

# Лабораторая работа №2

В данной работе выполнена интеграция FastAPI приложения, базы данных и парсера данных в единый Docker контейнер. Реализована возможность вызова парсера как напрямую через HTTP, так и асинхронно через очередь задач Celery и брокера сообщений Redis.

## Подзадача 1: Упаковка FastAPI приложения, базы данных и парсера данных в Docker

### Создание FastAPI приложения, базы данных и парсера данных

- **FastAPI приложение**: Создано на основе лабораторной работы номер 1.
- **База данных**: Создана на основе лабораторной работы номер 1.
- **Парсер данных**: Создан на основе лабораторной работы номер 2.

### Разработка Dockerfile

Для каждого сервиса (FastAPI приложение и парсер данных) были созданы отдельные Dockerfile. Эти файлы содержат инструкции для:

1. Указания базового образа.
2. Установки необходимых зависимостей.
3. Копирования исходных файлов в контейнер.
4. Определения команды для запуска приложения.

```
FROM python:3.10-alpine3.19

WORKDIR /lab_2

COPY . .
RUN pip3 install -r requirements.txt

CMD uvicorn main:app --host localhost --port 3001
```

```
FROM python:3.10-alpine3.19

WORKDIR /taskmanager

COPY . .
RUN pip3 install -r requirements.txt

CMD uvicorn main:app --host localhost --port 3000
```

## Создание Docker Compose файла

Создан файл `docker-compose.yml` для управления оркестром следующих сервисов:

```
version: '3.10'
services:

  taskmanager:
    container_name: taskmanager
    build:
      context: ./lab1
    depends_on:
      - db
    ports:
      - "3000:3000"
    command: uvicorn main:app --host 0.0.0.0 --port 3000
    networks:
      - backend_3
    restart: always

  lab_2:
    container_name: lab_2
    build:
      context: ./lab_2
    restart: always
    ports:
      - "3001:3001"
    command: uvicorn main:app --host 0.0.0.0 --port 3001
    depends_on:
```

```
      - redis
      - db
    networks:
      - backend_3

  celery_start:
    build:
      context: ./lab_2
    container_name: celery_start
    command: celery -A celery_start worker --loglevel=info
    restart: always
    depends_on:
      - redis
      - lab_2
      - db
    networks:
      - backend_3

  redis:
    image: redis
    ports:
      - "6379:6379"
    networks:
      - backend_3
    depends_on:
      - db

  db:
    image: postgres
    restart: always
    environment:
      - POSTGRES_PASSWORD=postgres
      - POSTGRES_USER=postgres
      - POSTGRES_DB=timedb
    volumes:
      - postgres_data:/var/lib/postgresql/data/
    ports:
      - "5432:5432"
```

```
    networks:
      - backend_3


volumes:
  postgres_data:

networks:
  backend_3:
     driver: bridge
```

# Подзадача 2: Вызов парсера из FastAPI

## Эндпоинт в FastAPI для вызова парсера

В FastAPI приложение добавлен эндпоинт, который принимает запросы с URL для парсинга:

```
from fastapi import FastAPI, HTTPException
import requests

app = FastAPI()

@app.post("/parse/")
async def parse(url: str, background_tasks: BackgroundTask
s):
    background_tasks.add_task(parse_and_save_threading, ur
l)
    return {"message": "Parse started."}


@app.get("/get_task")
def read_project(db=Depends(get_db)):
    return db.query(Page).all()
```

# Подзадача 3: Вызов парсера из FastAPI через очередь

## Установка Celery и Redis

Добавлены зависимости для Celery и Redis в проект:

```
pip install celery redis
```

### Настройка Celery

```
@celery_app.task
def parse_and_save_threading(url):
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    title = soup.title.string if soup.title else "No title"
    conn = psycopg2.connect(DATABASE_URL)
    curs = conn.cursor()
    curs.execute("INSERT INTO pages (url, title) VALUES (%
s, %s)",
                 (url, title))
    conn.commit()
    curs.close()
    conn.close()
    print(f"Threading - URL: {url}, Title: {title}")
```

## Заключение

Выполненная работа включает в себя упаковку FastAPI приложения, базы данных и парсера данных в Docker, а также реализацию вызова парсера как синхронно через HTTP, так и асинхронно через Celery и Redis. Эти решения обеспечивают гибкость, масштабируемость и улучшенную производительность приложения.