



МИНОБРНАУКИ РОССИИ

**Федеральное государственное бюджетное образовательное учреждение
высшего образования**

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий

**Кафедра Математического обеспечения и стандартизации информационных
технологий**

Отчет по самостоятельной работе №3

по дисциплине

«Технология разработки программных приложений»

Выполнили:

Студенты группы ИНБО-30-23

Кук М. В.

Антошкин В. Г.

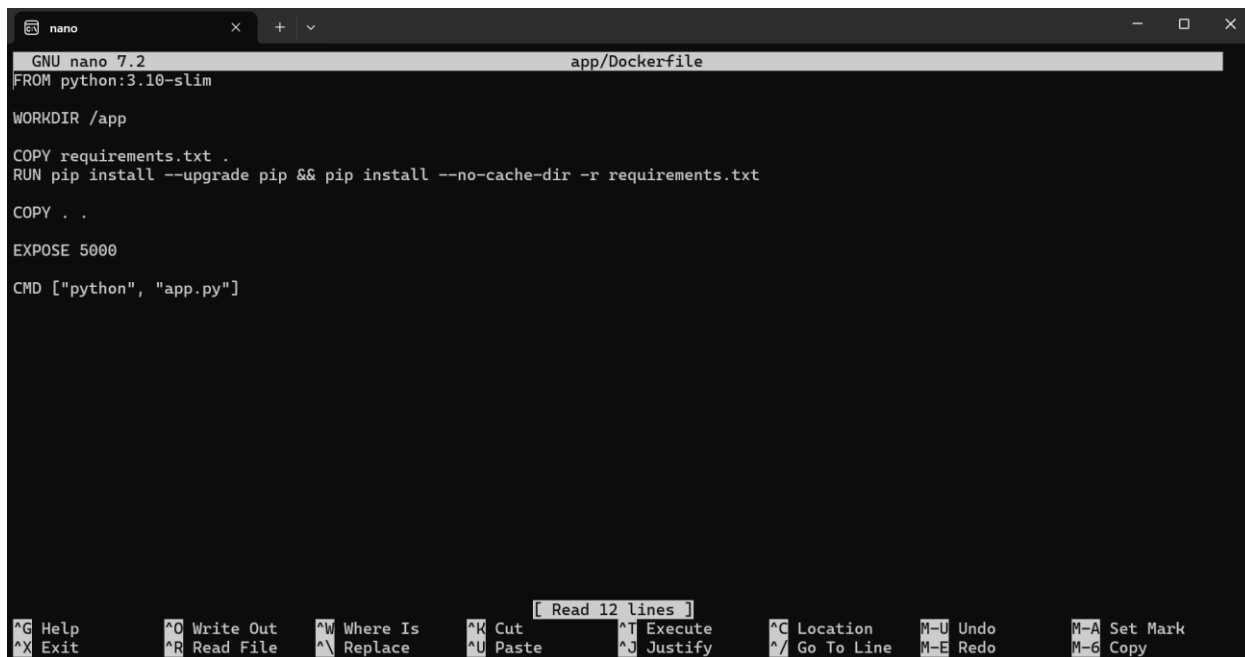
Проверил:

Преподаватель Исабекова О. А.

Москва 2025

1. Часть 3. Развертывание приложения

Создайте Dockerfile, в который запакуйте ваше приложение. Если приложению нужны внешние ресурсы (например, базы данных), то необходимые параметры для подключения приложение должно получать через переменные окружения, рис. 1. Структура проекта представлена, рис. 2.



```
GNU nano 7.2 app/Dockerfile
FROM python:3.10-slim

WORKDIR /app

COPY requirements.txt .
RUN pip install --upgrade pip && pip install --no-cache-dir -r requirements.txt

COPY . .

EXPOSE 5000

CMD ["python", "app.py"]
```

The screenshot shows a terminal window with the nano text editor open. The editor is editing a file named 'app/Dockerfile'. The content of the Dockerfile is as follows: FROM python:3.10-slim, WORKDIR /app, COPY requirements.txt ., RUN pip install --upgrade pip && pip install --no-cache-dir -r requirements.txt, COPY . ., EXPOSE 5000, and CMD ["python", "app.py"]. The nano editor's status bar at the bottom shows various keyboard shortcuts like ^G Help, ^X Exit, ^O Write Out, etc.

Рисунок 1 – Dockerfile проекта



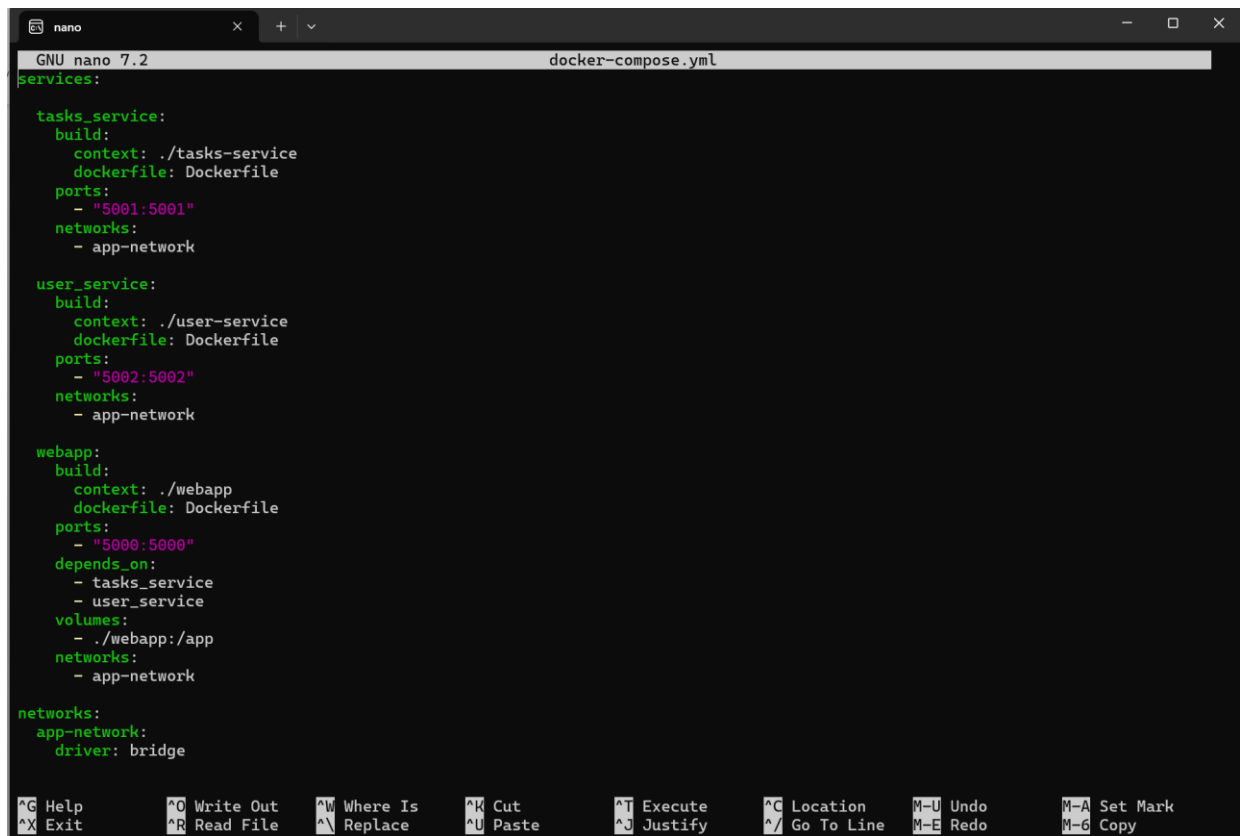
```
--> micro-project (main)tree .
.
├── docker-compose.prod.yml
├── docker-compose.yml
├── tasks-service
│   ├── Dockerfile
│   ├── requirements_serv.txt
│   └── src
│       ├── task_service.py
│       └── test_task_service.py
├── user-service
│   ├── Dockerfile
│   ├── requirements_user.txt
│   └── src
│       └── user_service.py
└── webapp
    ├── Dockerfile
    ├── requirements.txt
    └── src
        ├── app.py
        └── templates
            ├── index.html
            ├── register.html
            └── tasks.html

8 directories, 15 files
```

The screenshot shows a terminal window with the 'tree' command output. The output displays the directory structure of the project. The root directory contains 'docker-compose.prod.yml', 'docker-compose.yml', 'tasks-service', 'user-service', and 'webapp'. Each of these subdirectories contains a 'Dockerfile', a requirements file, and a 'src' directory. The 'src' directory for 'tasks-service' contains 'task_service.py' and 'test_task_service.py'. The 'src' directory for 'user-service' contains 'user_service.py'. The 'src' directory for 'webapp' contains 'app.py' and a 'templates' directory. The 'templates' directory contains 'index.html', 'register.html', and 'tasks.html'. The output ends with '8 directories, 15 files'.

Рисунок 2 – Структура проекта

Также можно сделать docker-compose.yml, рис 3.



```
GNU nano 7.2 docker-compose.yml
services:

  tasks_service:
    build:
      context: ./tasks-service
      dockerfile: Dockerfile
    ports:
      - "5001:5001"
    networks:
      - app-network

  user_service:
    build:
      context: ./user-service
      dockerfile: Dockerfile
    ports:
      - "5002:5002"
    networks:
      - app-network

  webapp:
    build:
      context: ./webapp
      dockerfile: Dockerfile
    ports:
      - "5000:5000"
    depends_on:
      - tasks_service
      - user_service
    volumes:
      - ./webapp:/app
    networks:
      - app-network

networks:
  app-network:
    driver: bridge

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   ^M-U Undo     ^M-A Set Mark
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/_ Go To Line ^M-E Redo     ^M-G Copy
```

Рисунок 3 – Файл docker-compose.yml