

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №4

Выполнил:

Таначев Егор

Группа К33412

Проверил:

Добряков Д. И.

Санкт-Петербург

2023 г.

Задача

Упаковать наше приложение в docker-контейнеры и обеспечить сетевое взаимодействие между различными частями вашего приложения. Делать это можно как с помощью docker-compose, так и с помощью docker swarm.

Ход работы

Создадим файл docker-compose.yml, как показано на Рисунке 1, который будет определять нашу инфраструктуру, состоящую из нескольких контейнеров: User и Market. В файле docker-compose.yml мы указываем, какие контейнеры мы хотим запустить, как они должны взаимодействовать и какие порты следует открыть для доступа к сервисам.



```
docker-compose.yml
1  # Версия
2  version: "3.9"
3
4  services:
5    # Сервис Пользователя
6    user:
7      container_name: user
8      build:
9        context: ./user
10     ports:
11       - "1111:1111"
12     command: npm run start
13   # Сервис Интернет-магазина
14   market:
15     container_name: market
16     build:
17       context: ./market
18     ports:
19       - "2222:2222"
20     command: npm run start
21     depends_on:
22       - user
```

Рисунок 1 – Файл docker-compose.yml

Для каждой части приложения создадим Dockerfile, который определяет, как упаковать эту часть в контейнер. Dockerfile содержит

инструкции по установке зависимостей, копированию файлов и настройке окружения. Dockerfile показан на Рисунках 2-3.

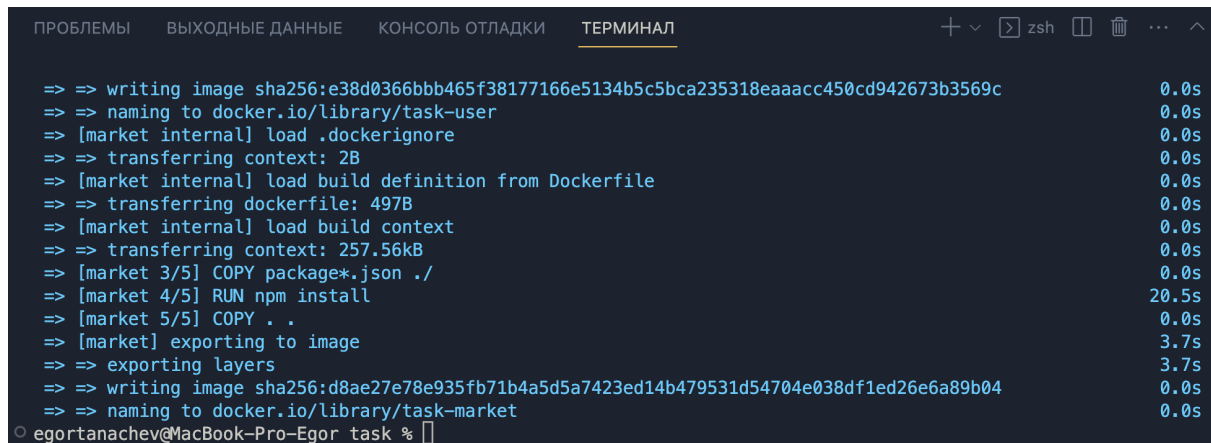
```
user > Dockerfile
1  # На каком образе работает докер контейнер
2  FROM node:16-alpine
3
4  # Рабочая директория
5  WORKDIR /app
6
7  # Копирование package.json и package-lock.json
8  COPY package*.json ./
9
10 # Установка зависимостей
11 RUN npm install
12
13 # Копирование всех файлов
14 COPY . .
15
16 # Прослушивание порта
17 EXPOSE 1111
18
19 # Запуск скрипта
20 CMD [ "npm", "start" ]
```

Рисунок 2 – Dockerfile в User

```
market > Dockerfile
1  # На каком образе работает докер контейнер
2  FROM node:16-alpine
3
4  # Рабочая директория
5  WORKDIR /app
6
7  # Копирование package.json и package-lock.json
8  COPY package*.json ./
9
10 # Установка зависимостей
11 RUN npm install
12
13 # Копирование всех файлов
14 COPY . .
15
16 # Прослушивание порта
17 EXPOSE 2222
18
19 # Запуск скрипта
20 CMD [ "npm", "start" ]
```

Рисунок 3 – Dockerfile в Market

Теперь соберем наш проект с помощью команды `docker-compose build`, как показано на Рисунке 4.

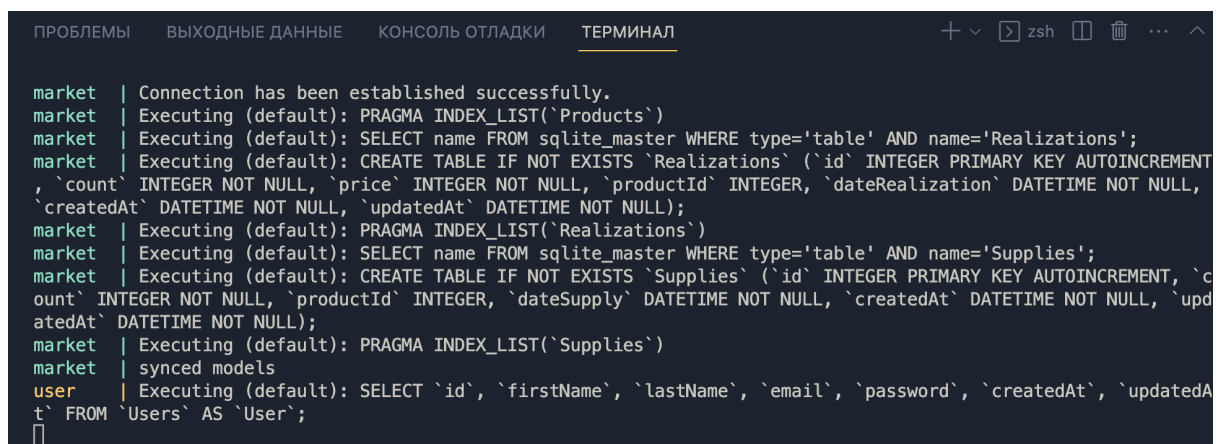


```
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ  + - [x] zsh [ ] [ ] ... ^

=> => writing image sha256:e38d0366bbb465f38177166e5134b5c5bca235318eaaacc450cd942673b3569c      0.0s
=> => naming to docker.io/library/task-user      0.0s
=> [market internal] load .dockerignore      0.0s
=> => transferring context: 2B      0.0s
=> [market internal] load build definition from Dockerfile      0.0s
=> => transferring dockerfile: 497B      0.0s
=> [market internal] load build context      0.0s
=> => transferring context: 257.56kB      0.0s
=> [market 3/5] COPY package*.json ./      0.0s
=> [market 4/5] RUN npm install      20.5s
=> [market 5/5] COPY . .      0.0s
=> [market] exporting to image      3.7s
=> => exporting layers      3.7s
=> => writing image sha256:d8ae27e78e935fb71b4a5d5a7423ed14b479531d54704e038df1ed26e6a89b04      0.0s
=> => naming to docker.io/library/task-market      0.0s
egortanachev@MacBook-Pro-Egor task %
```

Рисунок 4 – Команда `docker-compose build`

И запустим наше приложение с помощью команды `docker-compose up`, как показано на Рисунке 5.



```
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ  + - [x] zsh [ ] [ ] ... ^

market | Connection has been established successfully.
market | Executing (default): PRAGMA INDEX_LIST('Products')
market | Executing (default): SELECT name FROM sqlite_master WHERE type='table' AND name='Realizations';
market | Executing (default): CREATE TABLE IF NOT EXISTS `Realizations` (`id` INTEGER PRIMARY KEY AUTOINCREMENT, `count` INTEGER NOT NULL, `price` INTEGER NOT NULL, `productId` INTEGER, `dateRealization` DATETIME NOT NULL, `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL);
market | Executing (default): PRAGMA INDEX_LIST('Realizations')
market | Executing (default): SELECT name FROM sqlite_master WHERE type='table' AND name='Supplies';
market | Executing (default): CREATE TABLE IF NOT EXISTS `Supplies` (`id` INTEGER PRIMARY KEY AUTOINCREMENT, `count` INTEGER NOT NULL, `productId` INTEGER, `dateSupply` DATETIME NOT NULL, `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL);
market | Executing (default): PRAGMA INDEX_LIST('Supplies')
market | synced models
user    | Executing (default): SELECT `id`, `firstName`, `lastName`, `email`, `password`, `createdAt`, `updatedAt` FROM `Users` AS `User`;

```

Рисунок 5 – Команда `docker-compose up`

Вывод

В результате работы мы успешно упаковали наше приложение в Docker-контейнеры и обеспечили сетевое взаимодействие между различными его частями, используя инструмент `docker-compose`.