Использование docker-compose

Теперь нам надо попробовать создать окружение которое позволит связать frontend + backend

- Создаем новую ветку в репозитории с именем hw-19_docker-compose
- Создаем в корне репозитория docker-compose.yml

```
version: "3"
services:
    frontend:
        build: ./frontend
        image: <YourAccountDockerHub>/hillel-frontend:0.0.1
        ports:
            - "8080:80"
        networks:
            - public
    backend:
        build: ./backend
        image: <YourAccountDockerHub>/hillel-backend:0.0.1
        depends_on:
            - mongo
        environment:
            PORT: '8081'
            NODE_ENV: 'production'
            MONGO_DB_URI: 'mongodb://mongo/conduit'
            SECRET: 'secret'
        healthcheck:
            test: ["CMD", "curl", "-f", "http://backend:8081/api/status"]
            interval: 10m
            timeout: 10s
            retries: 3
        ports:
            - 8081:8081
        restart: on-failure
        networks:
            - public
            - private
    mongo:
        image: mongo:latest
        expose:
            - 27017
        restart: on-failure
        volumes:
            - data:/data/db
        networks:
            - private

volumes:
    data:
networks:
    private:
    public:
```

- Попробуем собрать локальное окружение

```
docker-compose up -d
```

- Проверим доступность frontend по ссылке http://localhost:8080
- Пройдите регистрацию и опубликуйте свой первый пост
- Происходит ошибка, которую может наблюдать в консоли разработчика любого браузера, так как frontend ссылается на backend с URL https://conduit.productionready.io/api он же https://api.realworld.io/ что является demo окружение по-умолчанию от разработчиков данного проекта и там возможно будут у Вас проблемы.

Сделаем возможность работать с использование ENV vars для того что бы определить свой API URL на backend сервис

- Исправим в файле ./frontend/src/agent.js следующий блок const

const API_ROOT = 'https://conduit.productionready.io/api';

```
//const API_ROOT = 'https://conduit.productionready.io/api';
const API_ROOT = process.env.REACT_APP_BACKEND_URL || 'https://conduit.productionready.io/api';
```

- В качестве альтернативы не будем собирать frontend, а будет его запускать из под npm start
- Переименуем файл ./frontend/Dockerfile в ./frontend/Dockerfile.previous
- Создадим новый ./frontend/Dockerfile со следующим содержимым

```
FROM node:12
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app

COPY package.json /usr/src/app/
RUN npm install && npm cache clean -f
COPY . /usr/src/app

CMD [ "npm", "start" ]
```

- В описание сервиса frontend в файле docker-compose.yml добавим ENV переменную и поменяем порт на котором работает frontend приложение, будет это выглядить следующим образом

```
    frontend:
        build: ./frontend
        image: hilleldevops102021/frontend:0.0.2
        ports:
            - "8080:4100"
        networks:
            - public
        environment:
            - REACT_APP_BACKEND_URL=http://localhost:8081/api
```

- Пересоберем окружение

```
docker-compose up --build -d
```

- Создадим пользователя http://localhost:8080/register, иначе в режиме npm start будем ловить ошибки (реквизиты не проверяются, может быть ложная почта)

Если наше решение работает и Вы создали пользователя + создание поста проходит без ошибок, проверим БД на наличие сооответствующих записей.

- Попробуем проверить как хранятся наши данные в БД, для этого поднимем инструмент работы с MongoDB, создаем docker-compose.dev.yml

```
version: "3"
services:

    admin-mongo:
        image: adicom/admin-mongo
        ports:
            - 8082:8082
        environment:
            PORT: 8082
            DB_HOST: mongo
        networks:
            - public
            - private
```

- Расширим наше локальное окружение данным контейнером

```
docker-compose -f docker-compose.yml -f docker-compose.dev.yml up -d --build
```

- Попробуем обратится на веб-интерфейс данного решения и настроить соединения с MongoDB которая в изолированной сети http://localhost:8082

```
connection name: mongo
connection string: mongodb://mongo
```

- Изучите структуру БД и добавьте в описание ДЗ скриншот или выборку из коллекций articles и users.

Результатом домашнего задания будет

- Обновленный образ frontend в DockerHub со следующим номеров версии в качестве :tag
- docker-compose up -d должен позволить создать локальное окружение, что я буду делать в качестве проверки ДЗ
- Созданный Pull Request из ветки hw-19_docker-compose в основную ветку main (или в другую определенную основной в настройках репозитория)

В качестве НЕОБЯЗАТЕЛЬНОГО дополнительного задания

- Поянисте почему мы используем в качестве API URL адрес localhost:8081 а не backend:8081

Раздел "Мотивация"

- У вас все получится!!!
- Задание является фундаментом для последующих, делаем до победного!)