Задание для создания микросервиса:

Для запуска нашего сервиса в Docker ему необходим Dockerfile, описывающий необходимое окружение для сборки/запуска вашего приложения. Давайте создадим такой Dockerfile для нашего сервиса. Dockerfile должен состоять из build и run образов аналогично лекции.

Задание для создания микросервиса:

Давайте напишем сервис для интеграционных тестов, который подключится к нужным контейнерам, соберет наше приложение и проведет над ним серию тестов. Это часто используемый прием, для разгрузки разработчиков и избегания необходимости поднимать все руками. Тесты будут заключаться в поднятии приложения в докер контейнере и запросе различных эндпоинтов с указанием ожидаемых результатов. Для приложения я предлагаю использовать файл конфигурации yaml для указание тестов и ожидаемых результатов, в виде:

```
ContainerName: mycontainer
Environment:
  - DEBUG=1
  - 'DB=${db:network_address}'
Dependencies:
  - name: db
    environment:
      - DEBUG=1
    image: imagename
Dockerfile: ./Dockerfile
Tests:
  - ResponseContains: test
    ExpectedCode: 200
    Name: simple query
    QueryType: GET
    URL: /list
  - ExpectedCode: 200
    Name: add info
    Query: '{"insert":1}'
    QueryType: POST
```

```
    URL: /add
```

- **ContainerName** (обязательное) определит имя нашего контейнера который будет запущен в докере
- **Dockerfile** (обязательное) определит файл из которого мы будем строить наш image (имя image будет складываться из имя_контейнера-рандомное число).
- **Dependencies** (опционально) определят дополнительные контейнеры для запуска, например с базами данных и позволит вгрузить туда env значения, например, пароль для базы данных.
- **Environment** (опционально) определит переменные окружения для вашего контейнера и позволит передать, например, адрес БД. Сейчас для простоты мы будем поддерживать только передачу ip адреса БД через поле network_address.
- **Tests** (обязательное) определит массив тестов для запуска. Каждый тест содержит в себе следующие поля:
    - **Name** - (обязательное) имя теста
    - **URL** - (обязательное) http путь по которому обращается наш тест
    - **QueryType** - (обязательное) тип запроса (GET, POST, PUT, DELETE)
    - **Query** - (опциональное) тело запроса, если его нет, то в тело запроса не попадает ничего
    - **ExpectedCode** - (опциональное) ожидаемый код ответа от теста
    - **ResponseContains** - (опциональное) тело ответа должно содержать эту строку

Приложение должно включать в себя валидатор, который должен отдавать осмысленные ошибки. Из джентльменского набора

- Неверный YAML нельзя считать YAML
- Отсутствует файл конфигурации
- Не заполнены какие либо обязательные поля
- Не получается связаться с докером

Если все получилось то команда docker-tester start -f нашфайл запустит контейнеры, проведет тесты и сгенерирует файл report.txt с результатами тестов в виде

ИмяТеста ок

Или

ИмяТеста fail (ExpectedCode 200 got 400)

По тесту на новой строке. Если файл уже был создан, то старый удаляется и появляется новый.