САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

ЛР 5 - Миграция написанного API на микросервисную архитектуру

Выполнил:
Сергеев Виктор
K3441

Проверил:
Добряков Д. И.

Санкт-Петербург

2026 г.

Задача

- выделить самостоятельные модули в вашем приложении;
- провести разделение своего API на микросервисы (минимум, их должно быть 3);
- настроить сетевое взаимодействие между микросервисами;
- реализовать Dockerfile для каждого сервиса;
- написать общий docker-compose.yml.

Ход работы

В приложении были выделены следующие самостоятельные модули:

- сервис работы с пользователем - информация о нём + авторизация;
- сервис работы с рецептами - хранение рецептов, ингредиентов, тегов и шагов;
- социальный сервис - лайки и комментарии;
- gateway - точка входа, перенаправляя запросы от клиента в разные сервисы.

Был создан новый проект, в котором будет содержаться миросервисное приложение. В проекте было создано 4 директории для каждого сервиса и создано окружения со всеми библиотеками. Также, каждый сервис должен собственную базу данных, поэтому в был создан докер-композ, в котором поднимается 3 контейнера с postgres и в каждой создаётся своя база данных.

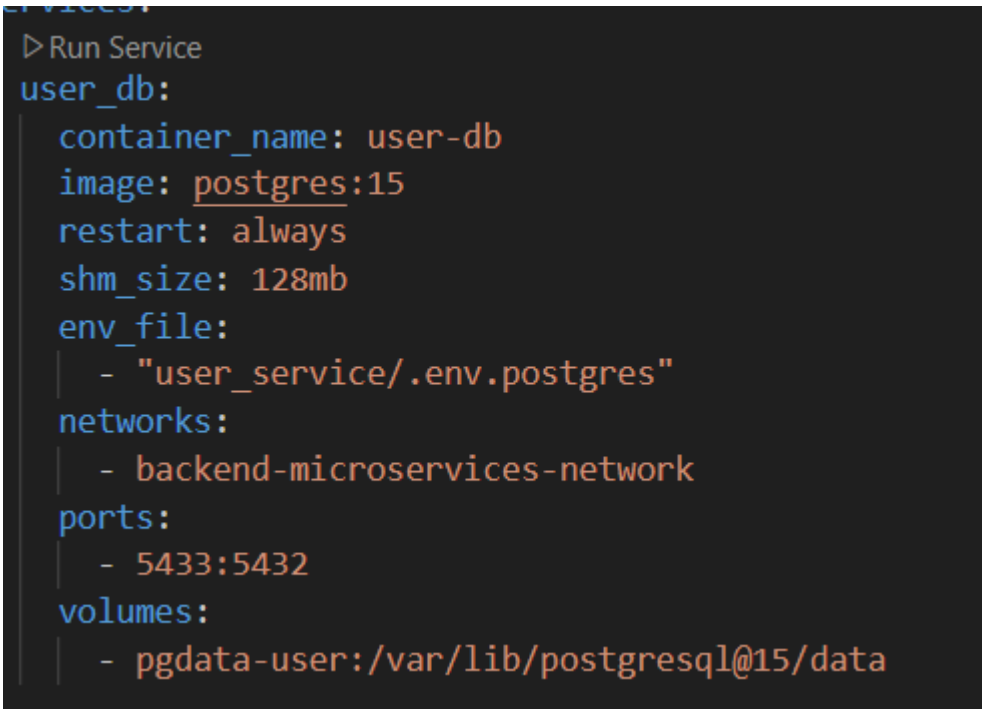

Рисунок 1 - Пример контейнера с базой данных для сервиса пользователей

Поскольку теперь приложение разделено, как и хранилище данных, требуется поменять модели и дто, чтобы модели не хранили другие модели, о которых в рамках своего сервиса они не знают. Вместо этого в моделях будут хранится ключи на эти модели в явном виде.

Рисунок 2 - Модель комментария без зависимостей

Остальные модули остались без изменений - проект просто был разделён на несколько частей, но сохранил практически всю свою функциональность.

Чтобы упаковать каждый сервис в контейнер, были написаны Dockerfile-ы для каждого из них. Логика Dockerfile-а следующая:

- берётся образ ноды, который будет выступать загрузчиком зависимостей, в который загружаются package*.json файлы и запускается загрузка библиотек
- берётся образ ноды, в котором уже будет поднимать приложение, в него копируются из загрузчика установленные зависимости, копируются исходники приложения и оно запускается

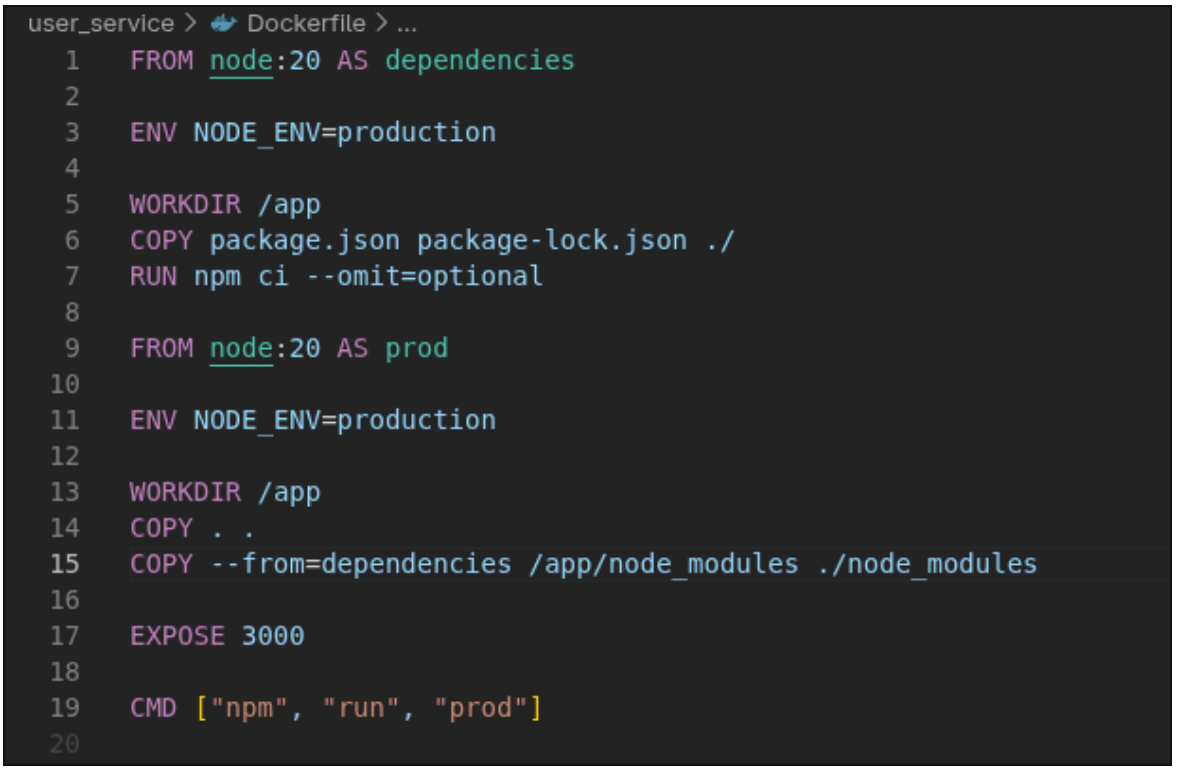

Рисунок 3 - Пример Dockerfile для сервиса пользователей

Для каждого сервиса Dockerfile практически идентичны.

Далее требовалось подключить эти сервисы через docker-compose. Сперва была создана внутренняя сеть docker-а:

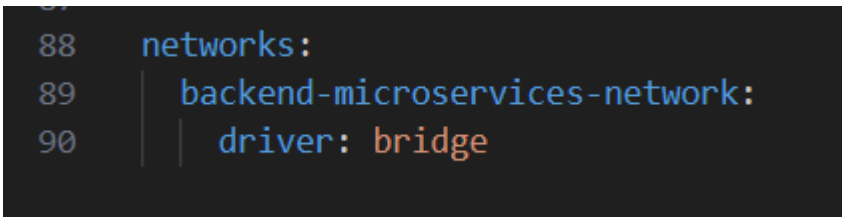

Рисунок 4 - Сеть докера для приложения

Теперь подключаются контейнеризированные сервисы:

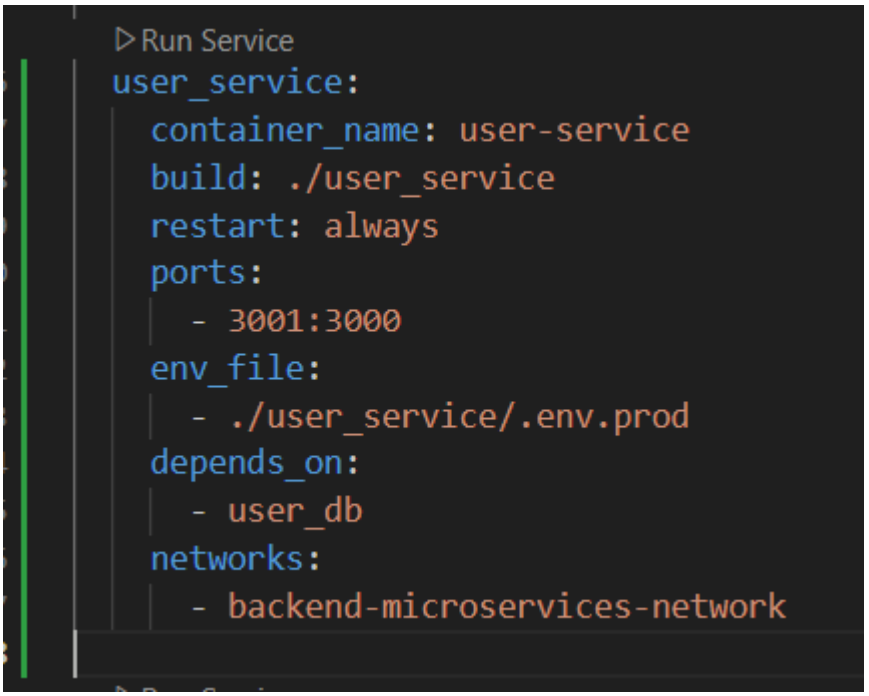

Рисунок 5 - Пример подключения сервиса в компоузе

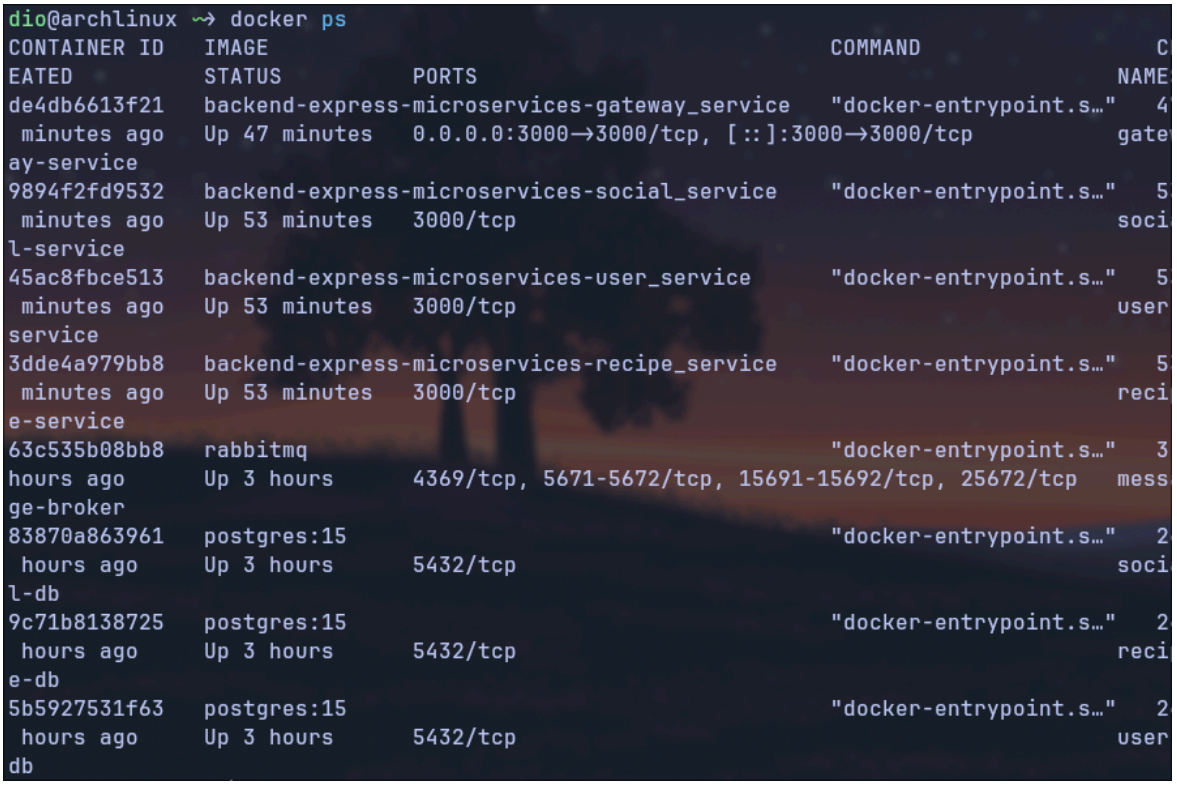

Рисунок 6 - Список запущенных контейнеров

Вывод

В процессе работы ранее написанное приложение было разделено на самостоятельные модули, которые были разделены в отдельные микросервисы. Каждый микросервис представляет собой отдельное приложение на node JS.

В процессе работы ранее написанные микросервисы приложения были упакованы в контейнеры с помощью средств docker. Был написан общий компоуз файл, который поднимает всё приложение вместе с базами данных. В компоуз файле была настроена внутренняя сеть докера для осуществления сетевого взаимодействия между сервисами.