САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа 4

Выполнили:

Никитин Павел

Группа

К33402

Проверил:
Добряков Д. И.

Санкт-Петербург

2024 г.

Задача

Необходимо упаковать ваше приложение в docker-контейнеры и обеспечить сетевое взаимодействие между различными частями вашего приложения, а также настроить общение микросервисов между собой посредством RabbitMQ. Делать это можно как с помощью docker-compose так и с помощью docker swarm. При разумном использовании swirl вы получите дополнительные баллы.

Ход работы

Был использован docker-compose с образами posgres, prisma studio, rabbitmq а также образами полученными из микросервисов

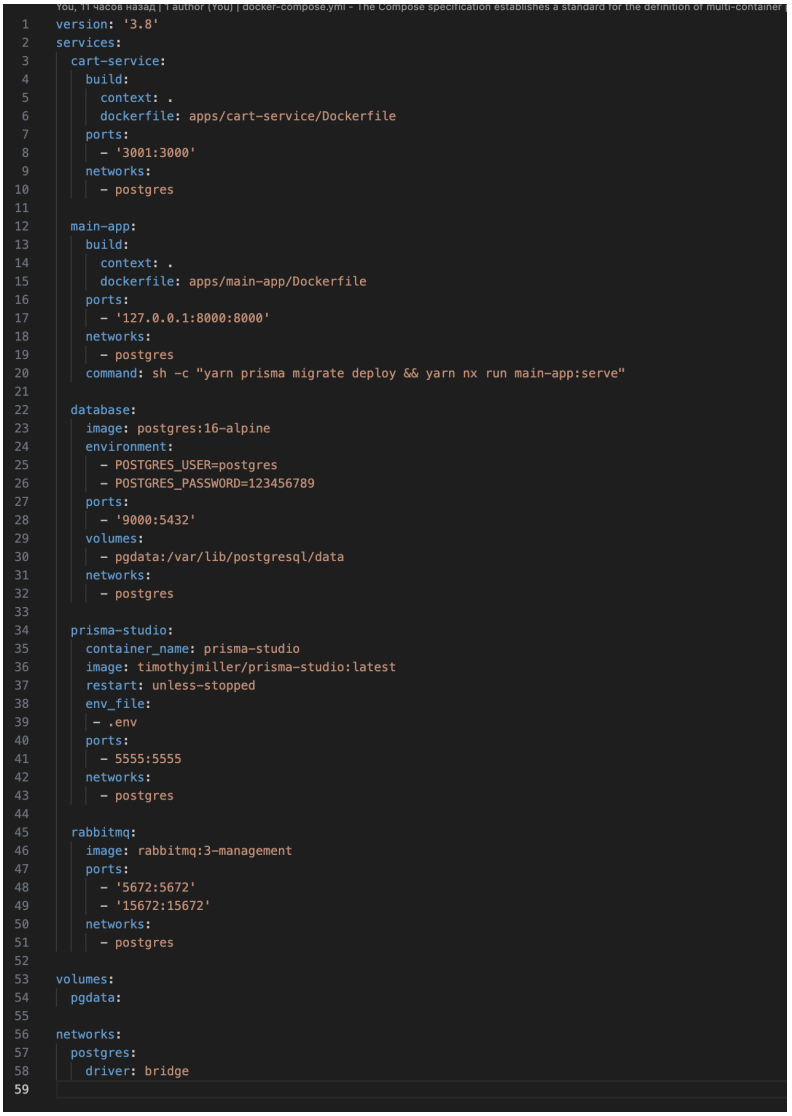

Рисунок 1 - docker compose

Мы объединили все контейнеры в 1 сеть чтобы им было проще общаться
Подключились через pgAdmin к базе данных запущенной в контейнере

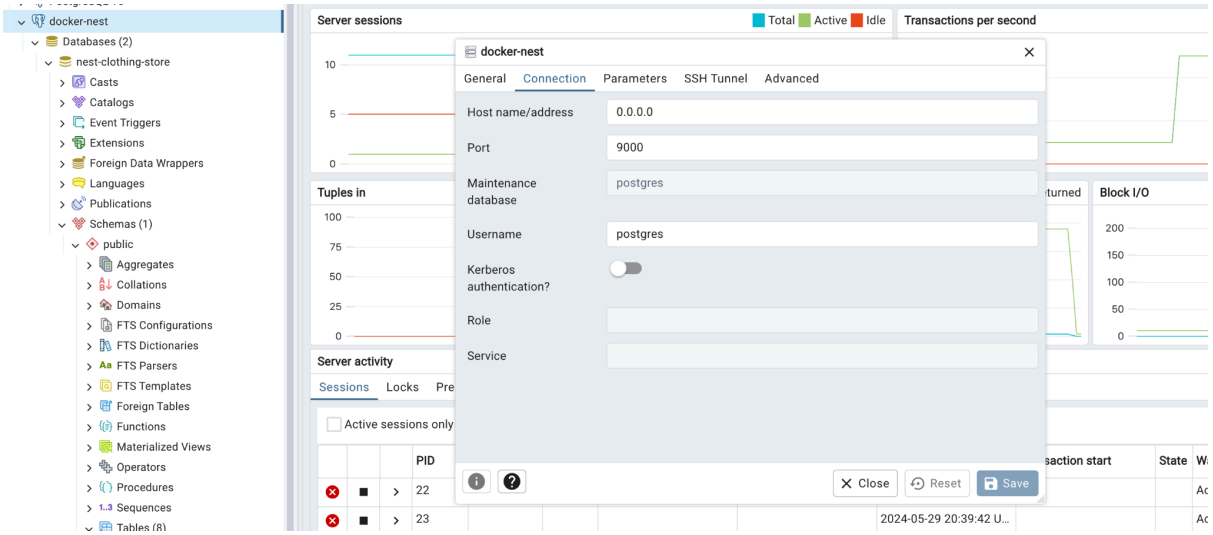

Рисунок 2 - PG admin

Также мы написали Dockerfile для каждого из сервисов

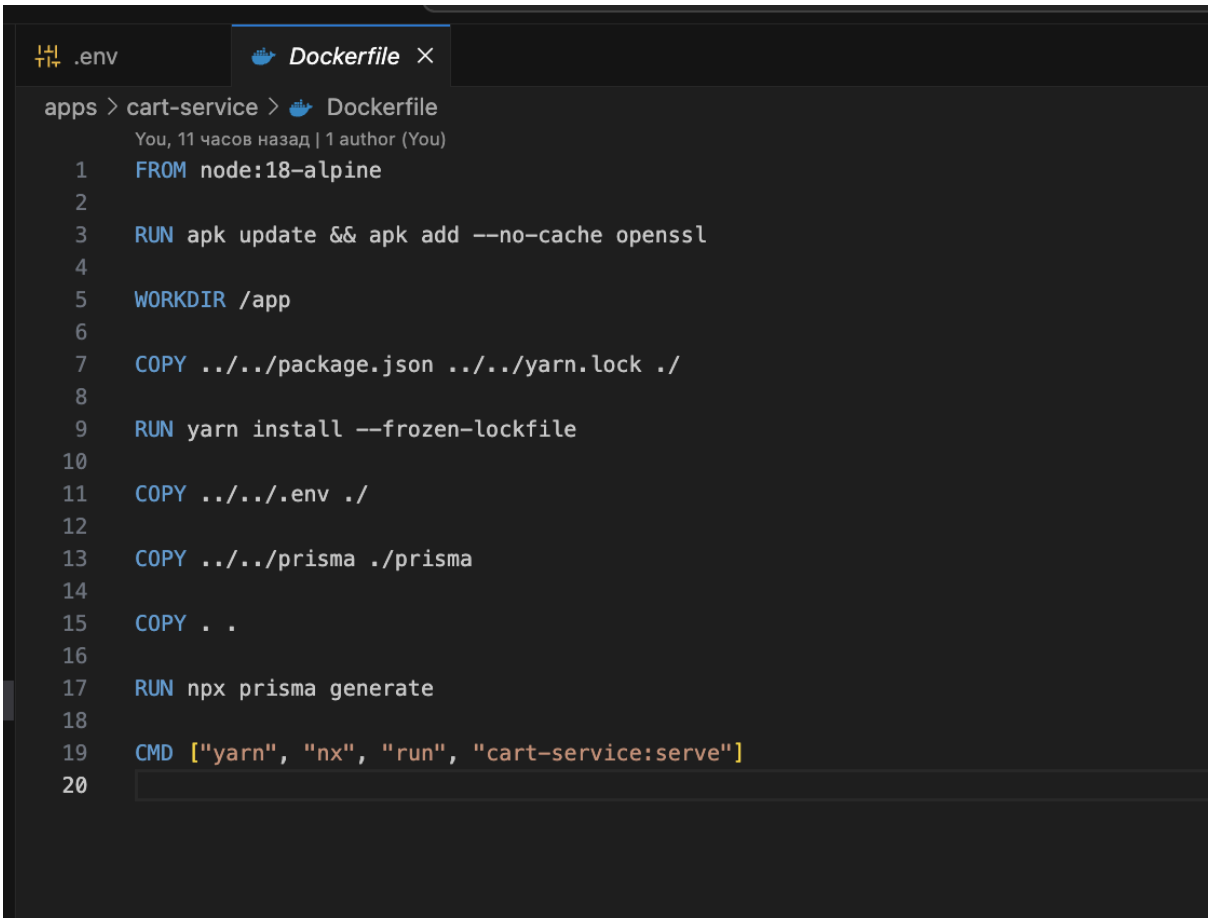

Рисунок 3 - Dockerfile для микросервисов

Добавили брокер сообщений в качестве транспорта между микросервисами:

```
async function bootstrap() {
  const app = await NestFactory.createMicroservice<MicroserviceOptions>(AppModule, {
    transport: Transport.RMQ,
    options: {
      urls: ['amqp://rabbitmq:5672'],
      queue: 'cart_service_queue',
      queueOptions: {
        durable: false,
      },
    },
  });

  await app.listen();
}
bootstrap();
```

Рисунок 4 - подключились к Rabbitmq

```
async function bootstrap() {
  const app = await NestFactory.create<NestExpressApplication>(AppModule);
  const config = app.get(ConfigService);
  const globalPrefix = 'api';

  app.useGlobalPipes(new ValidationPipe());
  app.setGlobalPrefix(globalPrefix);

  const swaggerConfig = new DocumentBuilder()
    .setTitle('nestjs-prisma-boilerplate')
    .setDescription('Store application')
    .setVersion('1.0')
    .addTag('auth')
    .addTag('user')
    .addBearerAuth()
    .build();
  const document = SwaggerModule.createDocument(app, swaggerConfig, {
    deepScanRoutes: true,
  });
  SwaggerModule.setup('api', app, document);

  app.connectMicroservice<MicroserviceOptions>({
    transport: Transport.RMQ,
    options: {
      urls: ['amqp://rabbitmq:5672'],
      queue: 'main_app_queue',
      queueOptions: {
        durable: false,
      },
    },
  });

  await app.listen(config.PORT);
  Logger.log(`🚀 Application is running on: http://localhost:${config.PORT}/${globalPrefix}`);
}

bootstrap();
```

Рисунок 5 -Также подключили main-app

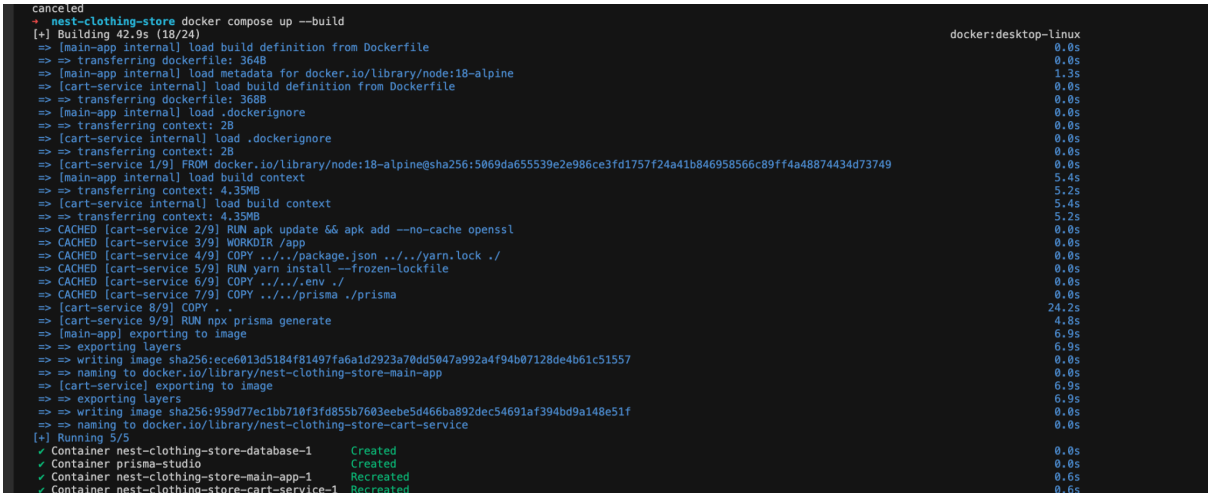

Рисунок 6 - запустили build

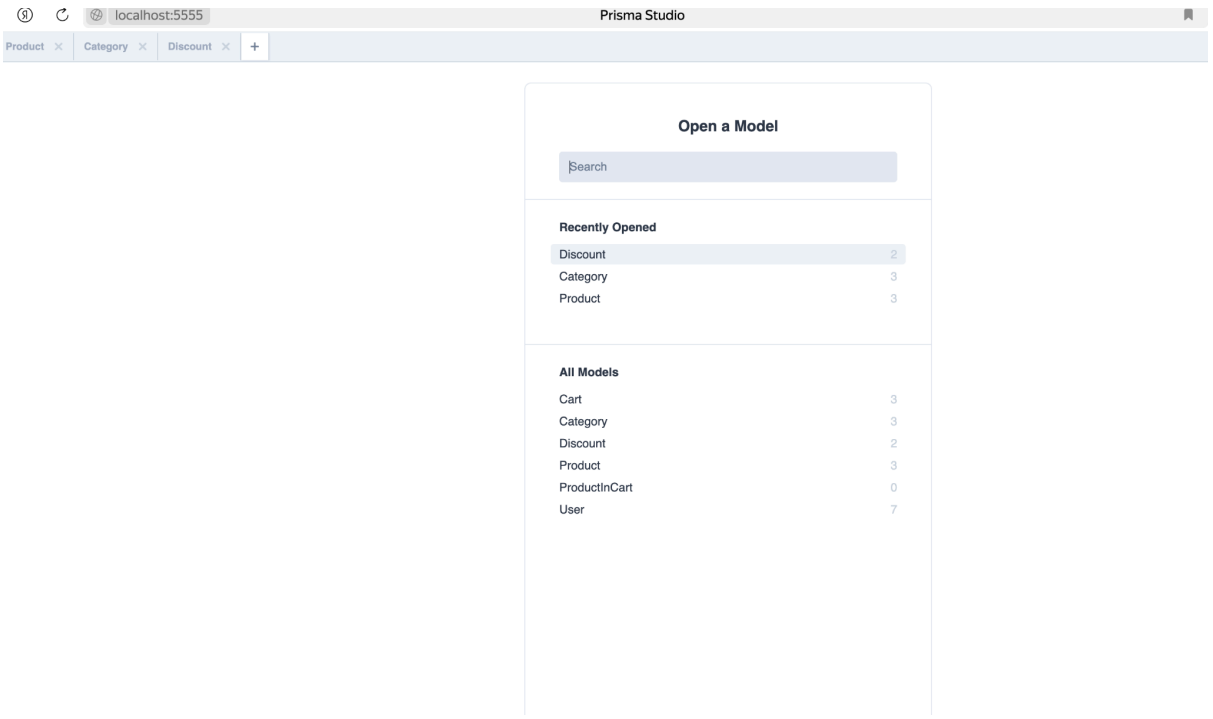

Рисунок 7 - подняли prisma

Вывод

Научился работать с docker, docker compose узнал о брокерах сообщений, смог соединить это воедино, поднять и пробросить порты. Также научился пользоваться dockerhub и искать образы.