САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №3

Микросервисы

Выполнила:
Зайцева А. А.
Группа К33402

Проверил:
Добряков Д. И.

Санкт-Петербург

2023 г.

Задача

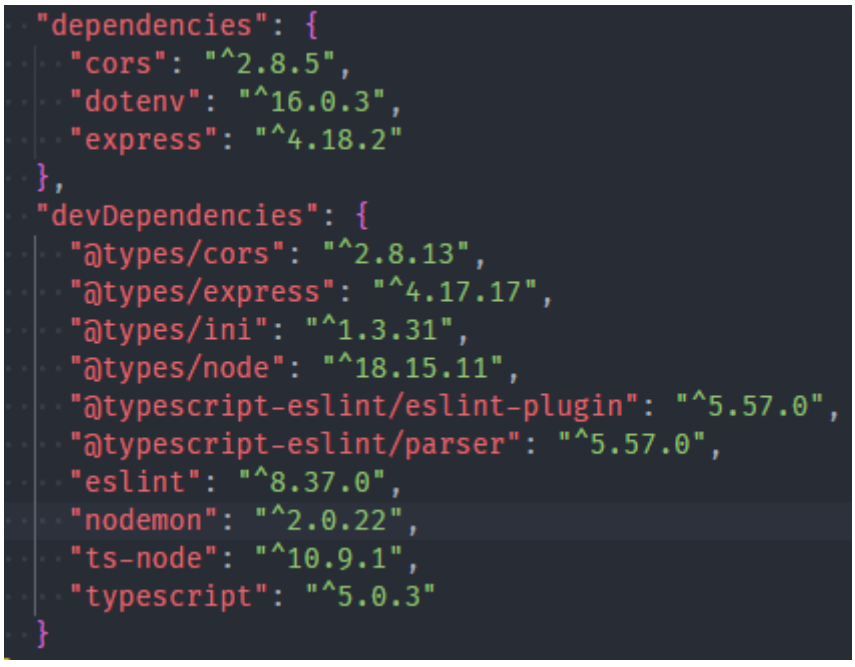
Необходимо реализовать отдельный микросервис, выполняющий какую-либо содержательную функцию из всего арсенала функций вашего приложения.

Ход работы

В отдельный микросервис был выделен функционал получения статистики по продажам интернет-магазина для построения графика. Сервис называется соответствующе – **Stat service**.

На основе проекта из лабораторной работы №2 (основного приложения) была подготовлена структура микросервиса.

В **package.json** были оставлены только необходимые зависимости - **express**, **typescript** и пакеты для развертывания, такие как **nodemon** и **ts-node**. Также для удобства доступа к конфигурации был добавлен пакет **dotenv**.



Считывание **dotenv** файла происходит при инициализации пакета **dotenv** в **index** файле микросервиса.

Микросервис предоставляет всего два эндпоинта - для получения статистики по количеству проданных единиц и по выручке.

```
router.route('/amount')
    .get(controller.getAmountStat)
router.route('/revenue')
    .get(controller.getRevenueStat)
```

Этим эндпоинтам соответствуют методы в контроллере. Каждый из методов получает список продаж от основного приложения с помощью **http** запроса. Если в запросе к микросервису был **query** параметр **product-id** - он передается в запросе к основному приложению, таким образом фильтруя продажи по товару. Для поддержки этой фильтрации был доработан метод **get** на получение списка продаж в основном приложении.

```
class StatController {
    getAmountStat = async (request: any, response: any) => {
        try {
            const productIdFilter = request.query['product-id']
                ? `?product-id=${request.query['product-id']}`
                : '';

            const salesResponse = await fetch(
                `http://${process.env.MAIN_HOST}:${process.env.MAIN_PORT}/v1/sales/sale${productIdFilter}`
            );
            const sales = await salesResponse.json() as Array<Sale>;
            const groupedSalesByDate = sales.reduce((groupedSales, sale) => {
                if (!(sale.dateOfSale in groupedSales)) {
                    return {
                        ...groupedSales,
                        [sale.dateOfSale]: sale.quantity
                    }
                } else {
                    return {
                        ...groupedSales,
                        [sale.dateOfSale]: groupedSales[sale.dateOfSale] + sale.quantity
                    }
                }
            }, {} as GroupedSalesByDate);

            const salesStat = Object.entries(groupedSalesByDate).map(([dateOfSale, quantity]) => ({
                dateOfSale,
                quantity,
            }));

            salesStat.sort((a, b) => a.dateOfSale.localeCompare(b.dateOfSale));
            response.send(salesStat)
        } catch (error: any) {
            response.status(500).send({ "error": error.message })
        }
    }
```

Список продаж далее группируется по датам продаж с помощью метода **reduce**. Результат возвращается в виде отсортированного массива.

Функционал, вынесенный в микросервис, был удален из кода основного приложения.

Для эффективного использования микросервиса его можно доработать, добавив кеширование самых популярных запросов статистики, снизив таким образом нагрузку на основное приложение.

Вывод

В ходе лабораторной работы была изучена микросервисная архитектура и реализован микросервис на базе приложения из лабораторной работы №2. Микросервисы являются эффективным решением для обеспечения отказоустойчивости, масштабирования приложения и распределения нагрузки.