САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа 3

Выполнили:

Никитин Павел

Группа

K33402

Проверил: Добряков Д. И.

Санкт-Петербург

2024 г.

Задача

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

Ход работы

Для этой задачи, было решено выделить сервис корзины из основного приложения. Поэтому была полностью переделана структура. Репозиторий проекта превратился в моно репозиторий содержащий сразу несколько сервисов для удобства в качестве утилиты был использован пх.

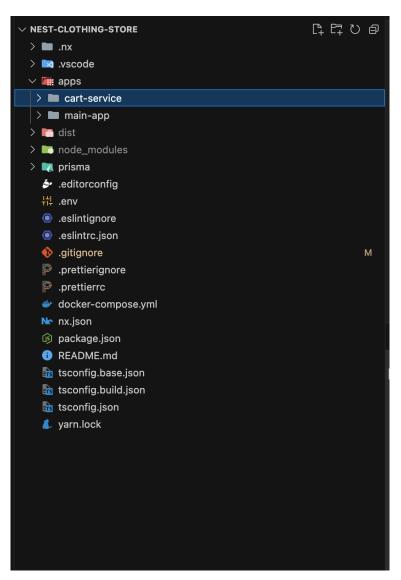


Рисунок 1 - новая структура проекта

Теперь каждый микросервис имеет свой project.json, который описывает как правильно собирать проект

```
"name": "cart-service",
"$schema": "../../node_modules/nx/schemas/project-schema.json",
"sourceRoot": "apps/cart-service/src",
"projectType": "application",
"tags": [],
"targets": {
  "build": {
    "executor": "@nx/webpack:webpack",
    "outputs": ["{options.outputPath}"],
    "defaultConfiguration": "production",
    "options": {
     "target": "node",
     "compiler": "tsc",
     "outputPath": "dist/apps/cart-service",
     "main": "apps/cart-service/src/main.ts",
     "tsConfig": "apps/cart-service/tsconfig.app.json",
     "webpackConfig": "apps/cart-service/webpack.config.ts"
    "configurations": {
     "development": {},
      "production": {
       "optimization": true,
       "inspect": false
  "serve": {
    "executor": "@nx/js:node",
    "defaultConfiguration": "development",
    "options": {
     "buildTarget": "cart-service:build"
    "configurations": {
      "development": {
       "buildTarget": "cart-service:build:development"
     "production": {
       "buildTarget": "cart-service:build:production"
  "docker-build": {
    "executor": "nx:run-commands",
    "options": {
     "command": "docker build -t cart-service:latest -f apps/cart-service/Dockerfile ."
```

Рисунок 2 - project.json мокросервиса

```
cart.service.ts ×
NEST-CLOTHING-STORE
                                                                                                                  You we fincularly import of Inaction (Tool) import { Injectable, Inject} from '@nestjs/common'; import { ClientProxy } from '@nestjs/microservices'; import { ProductInCartCreateDto, ProductInCartUpdateDto } from './dto/productInCart.dto';
      src 🖝
                                                                                                                    gunjectable()
export class CartService {
    constructor(@Inject('CART_SERVICE') private readonly client: ClientProxy) {}
         cart
                                                                                                                    getUserCart(userId: number) {
    return this.client.send({ cmd: 'getUserCart' }, { userId });
}
         cart.controller.ts
cart.module.ts
          cart.service.ts
                                                                                                                     addProductToCart(userId: number, productDto: ProductInCartCreateDto) {
   return this.client.send({ cmd: 'addProductToCart' }, { userId, productDto });
}
         category
       > m discount
       > product
                                                                                                                    removeProductFromCart(productInCartId: number) {
   return this.client.send({ cad: 'removeProductFromCart' }, { productInCartId });
}
       > user
                                                                                                                     updateProductInCart(productInCartId: number, productDto: ProductInCartUpdateDto) {
   return this.client.send({ cmd: 'updateProductInCart' }, { productInCartId, productDto });
        eslintrc.ison
        Dockerfile
                                                                                                                      createCart(userId: number) {
    return this.client.send({ cmd: 'createCart' }, { userId });
}
       tsconfig.app.json
       webpack.config.ts
```

Рисунок 3 - реализация cartService в главном приложении

Если раньше мы в сервисе обращались к базе данных то теперь просим получить эту информацию от другого микросервиса.

```
| Miles | Cart Controllers | Car
```

Рисунок 4 - контроллер в микросервисе корзины

```
ротладка
"scripts": {
    "main-app:serve": "yarn nx run main-app:serve",
    "cart-service:serve": "yarn nx run cart-service:serve",
    "dev": "yarn nx run-many -t serve -p cart-service main-app --output-style stream"
},
```

Рисунок 5 - скрипты для запуска проекта

Ссылка на результат -

https://github.com/pavel-nikitin-2022/nest-clothing-store/tree/pavelnikitin/feature/microservices

Вывод

В ходе этой работы я научился писать микросервисы на nestjs, хоть в идеале это должны быть разные репозитории с разными базами данных, текущая архитектура намного лучше предыдущей и позволяет легко вносить изменения в проект.