САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа № 3: Микросервисы.

Выполнил:

Безруков Андрей
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.

Вариант: онлайн-магазин Cute online shop API

Ход работы

В микросервис я решил выделить функционал учета товара: подсчет товаров по id, подсчет товаров по категории, возможная выручка всех товаров, по категориям, по id. Мною была исключена модель работника и соответствующие контроллеры и сервисы, т.к. они не задействованы в данном микросервисе.

Изменения в роутинге routes/product.ts:

```
36  router
37      .route('/count')
38      .get(authMiddleware.authenticate, controller.countProds)
39
40  router
41      .route('/count/category')
42      .get(authMiddleware.authenticate, controller.countProdsByCategory)
43
44  router
45      .route('/total')
46      .get(authMiddleware.authenticate, controller.totalEarn)
47
48  router
49      .route('/total/all')
50      .get(authMiddleware.authenticate, controller.totalEarnAll)
51
52  router
53      .route('/total/category')
54      .get(authMiddleware.authenticate, controller.totalEarnByCategory)
```

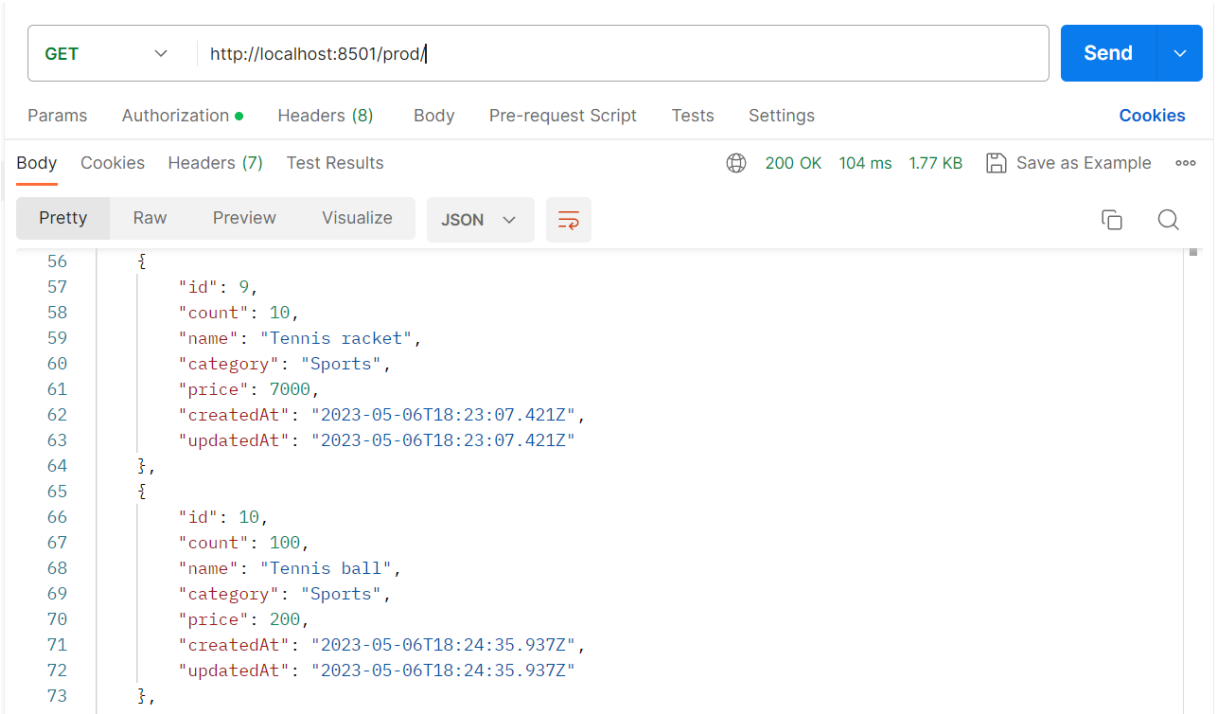
Изменения в сервисах services/product.ts:

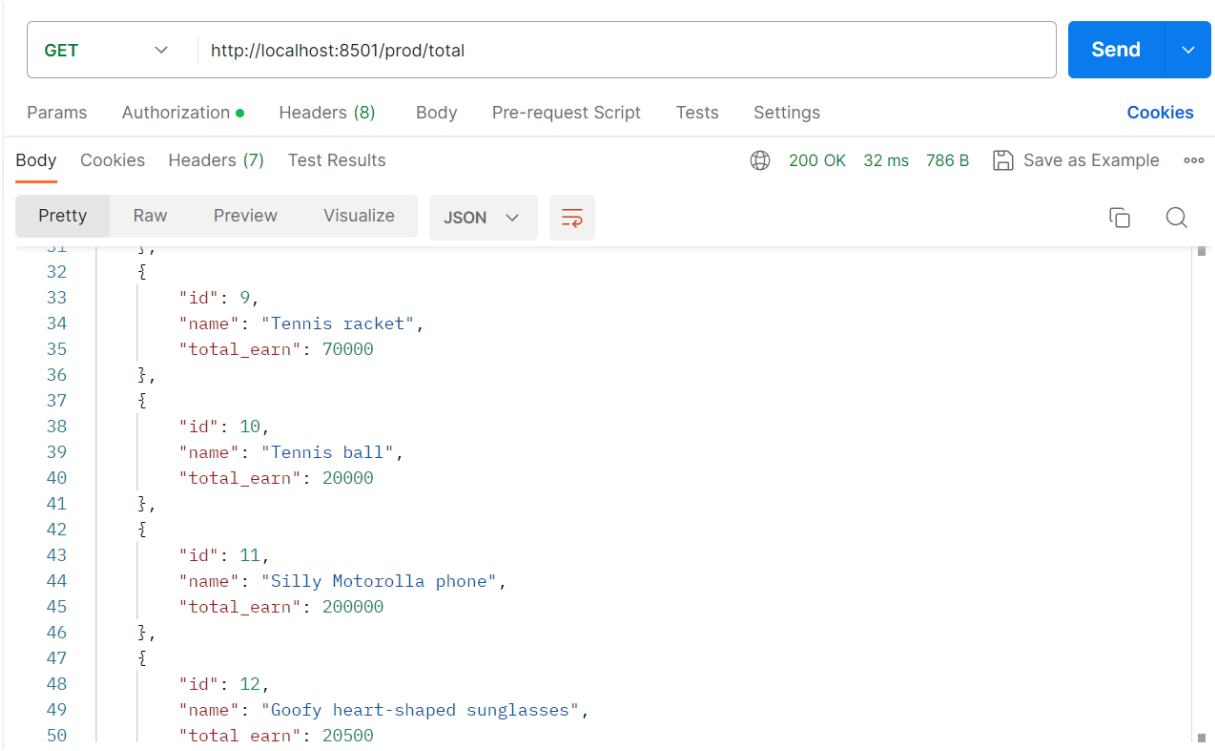
```
77      async countProdsByCategory (): Promise<any> {
78              const table = await prodRepository.findAll({
79              attributes: ['category', [sequelize.fn('SUM', sequelize.col('count')), 'total_count']],
80              group: ['category']
81          });
82          if (table) return table
83
84  }
85
86      async totalEarn (): Promise<any> {
87              const table = await prodRepository.findAll({
88              attributes: ['id','name', [sequelize.literal('price * count'), 'total_earn']],
89              group: ['id']
90          });
91          if (table) return table
92
93  }
94      async totalEarnAll (): Promise<any> {
95              const table = await prodRepository.findAll({
96              attributes: [[sequelize.fn('SUM', sequelize.literal('price * count')), 'total_earn']],
97          });
98          if (table) return table
99
100         }
101
102     async totalEarnByCategory (): Promise<any> {
103             const table = await prodRepository.findAll({
104             attributes: ['category', [sequelize.fn('SUM', sequelize.literal('price * count')), 'total
105             group: ['category']
106         });
```

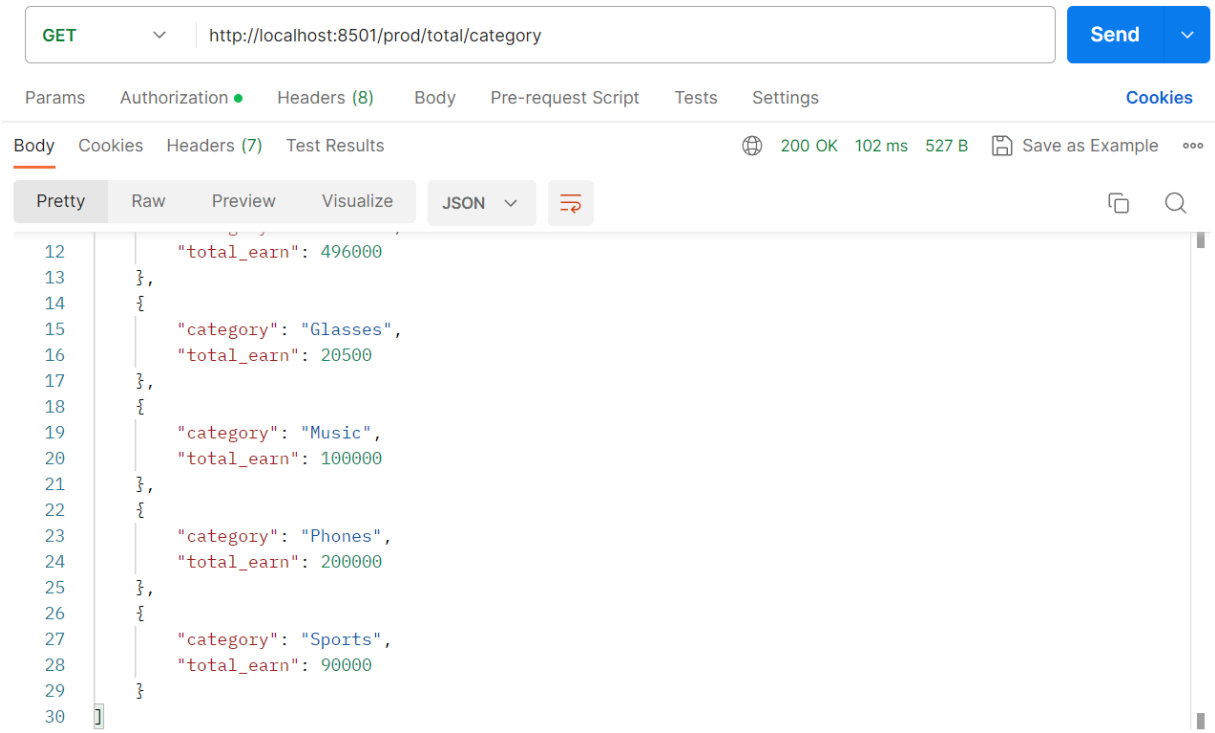
Изменения в контроллерах controllers/product.ts:

```
54    countProdsByCategory = async (request: any, response: any) => {
55        try {
56            const count = await this.productService.countProdsByCategory()
57            return response.json(count);
58        } catch (error: any) {
59            response.status(404).send({ "error": error.message })
60        }
61    }
62
63    totalEarn = async (request: any, response: any) => {
64        try {
65            const count = await this.productService.totalEarn()
66            return response.json(count);
67        } catch (error: any) {
68            response.status(404).send({ "error": error.message })
69        }
70    }
71
72    totalEarnAll = async (request: any, response: any) => {
73        try {
74            const count = await this.productService.totalEarnAll()
75            return response.json(count);
76        } catch (error: any) {
77            response.status(404).send({ "error": error.message })
78        }
79    }
80
81
82    totalEarnByCategory = async (request: any, response: any) => {
83        try {
```

Примеры работы:







Вывод

По итогам работы был успешно реализован отдельный микросервис из моего монолитного приложения, который позволяет отслеживать учет товара.