САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №2

Выполнил:

Буданцев Артём

К3333

Проверил:
Добряков Д. И.

Санкт-Петербург

2023 г.

Задача

Нужно реализовать RESTful API средствами express + typescript (используя ранее написанный boilerplate).

Реализованы API для администратора интернет-магазина:

- Вход
- Регистрация
- Учёт товара на складе
- Графики по продажам тех или иных товаров, по общей выручке предприятия
- Управление сотрудниками

Ход работы

1. Реализация моделей

Admin (администратор)

```
@Table
class Admin extends Model {
    @Unique
    @Column
    email!: string

    @AllowNull(false)
    @Column
    password!: string

    @BeforeCreate
    @BeforeUpdate
    static generatePasswordHash(instance: Admin) {
        const { password } = instance

        if (instance.changed('password')) {
            instance.password = hashPassword(password)
        }
    }
}
```

Good (товар)

```
@Table
class Good extends Model {
    @Unique
    @Column
    name!: string;

    @Default(0)
    @Column({
        type: DataType.INTEGER,
        validate: {
            isInt: true
        }

    })
    @Min(0, {message: 'Count must be greater than 0'})
    count!: number;

    @AllowNull(false)
    @Min(0, {message: 'Count must be greater than 0'})
    @Column({
        type: DataType.FLOAT
    })
    price!: number;

}
```

Sell (продажа)

```
@Table
class Sell extends Model {
    @AutoIncrement
    @PrimaryKey
    @Column(DataType.INTEGER)
    id!: number;


    @ForeignKey(() => Good)
    @Column
    goodId!: number;

    @Default(1)
    @Min(1, {message: 'Count must be greater than 0'})
    @Column
    count!: number;

    @Column({
        type: DataType.FLOAT
    })
    price!: number;
}
```

Staff (сотрудник)

```
@Table
class Staff extends Model {
    @AllowNull(false)
    @Column
    firstName!: string;

    @AllowNull(false)
    @Column
    lastName!: string;

    @Column
    position!: string;

    @Min(0, {message: 'Count must be greater than 0'})
    @Column({
        type: DataType.FLOAT
    })
    salary!: number;
}
```

2. Реализация Routes на примере Good

```
router.route('/')
    .get(controller.get)

router.route('/:id')
    .get(controller.getById)

router.route('/')
    .post(controller.post)

router.route('/:id')
    .patch(controller.patch)

router.route("/:id")
    .delete(controller.delete)
```

3. Реализация контроллера на примере Good

```
get = async (req: express.Request, res: express.Response) => {

    res.type("json")

    const goods = await this.goodService.get();
    res.send(goods)
}

getById = async (request: any, response: any) => {
    try {
        const good = await this.goodService.getById(
            Number(request.params.id)
        )
        response.send(good)
    } catch (error: any) {
        response.status(404).send({ "error": error.message })
    }
}
```

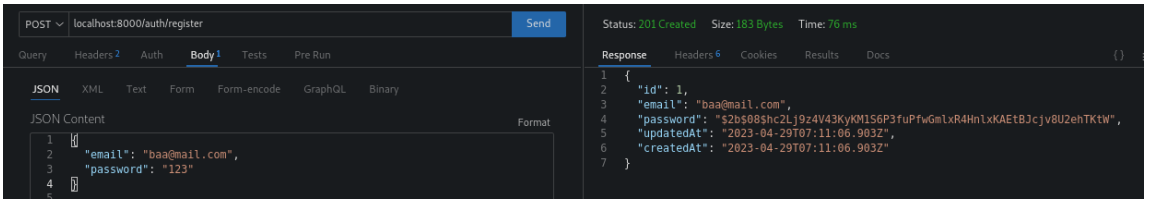
```
post = async (req: express.Request, response: express.Response) => {
    response.type("json")
    const { body } = req
    try {
        const good = await this.goodService.create(body)
        response.send(good)
    } catch (error: any) {
        response.status(404).send({ "error": error.message })
    }
};
```

```
patch = async (req: express.Request, response: express.Response) => {
    response.type("json")
    const { body } = req;
    const { id } = req.params

    try {
        const good = await this.goodService.update(Number(id), body)
        response.status(200).send(good)
    } catch (error: any) {
        response.status(404).send({ "error": error.message })
    }
};
```

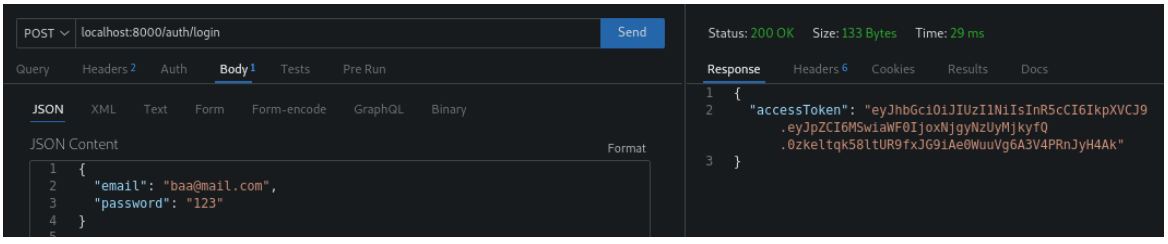
```
delete = async (req: express.Request, response: express.Response) => {
    response.type("json");
    const { id } = req.params;

    try {
        await this.goodService.delete(Number(id))
        response.status(204).send()
    } catch (error: any) {
        response.status(404).send({ "error": error.message })
    }
};
```

4. Примеры запросов

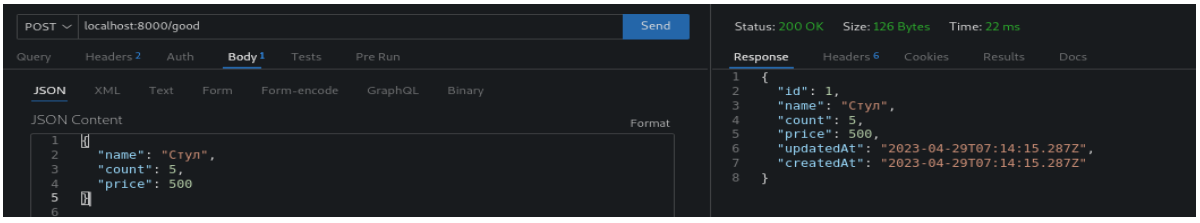
/auth/register



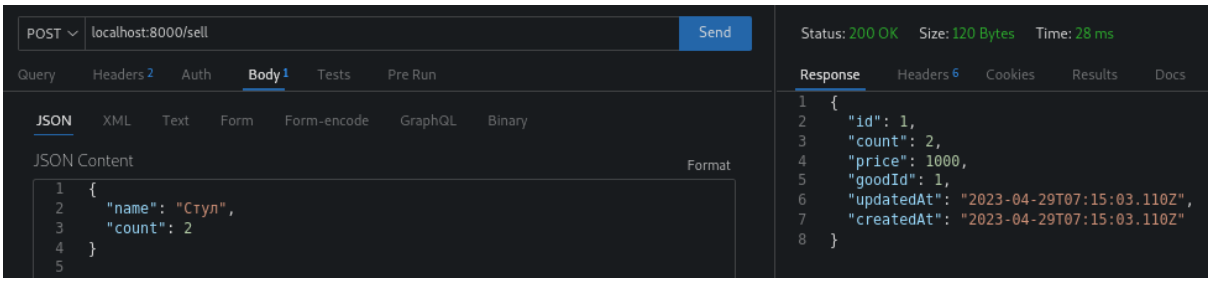
/auth/login



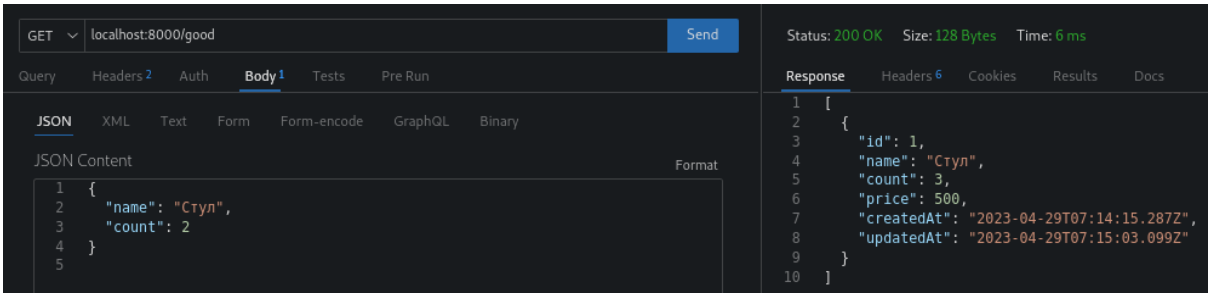
POST /good



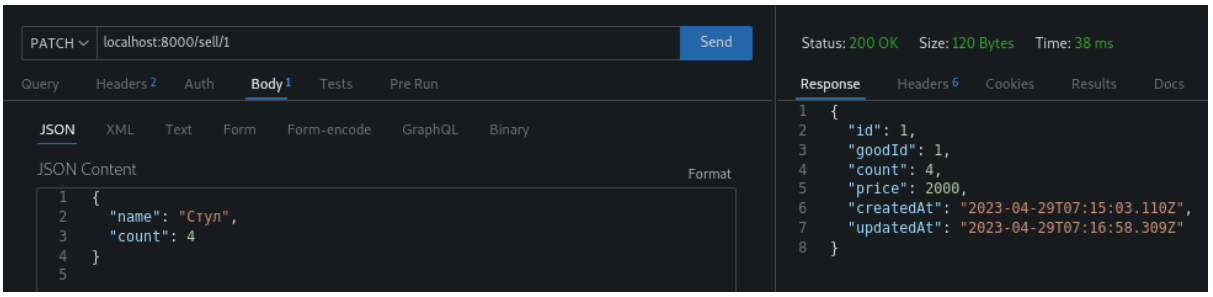
POST /sell



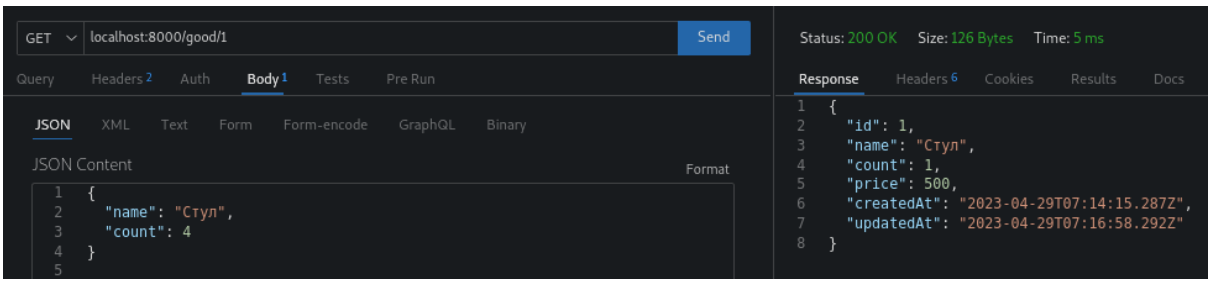
GET /good



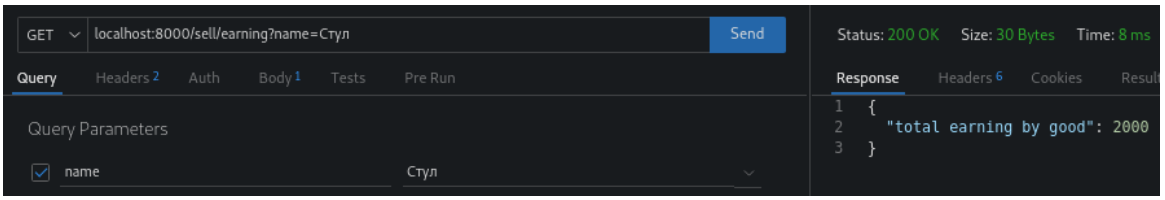
PATCH /sell/:id



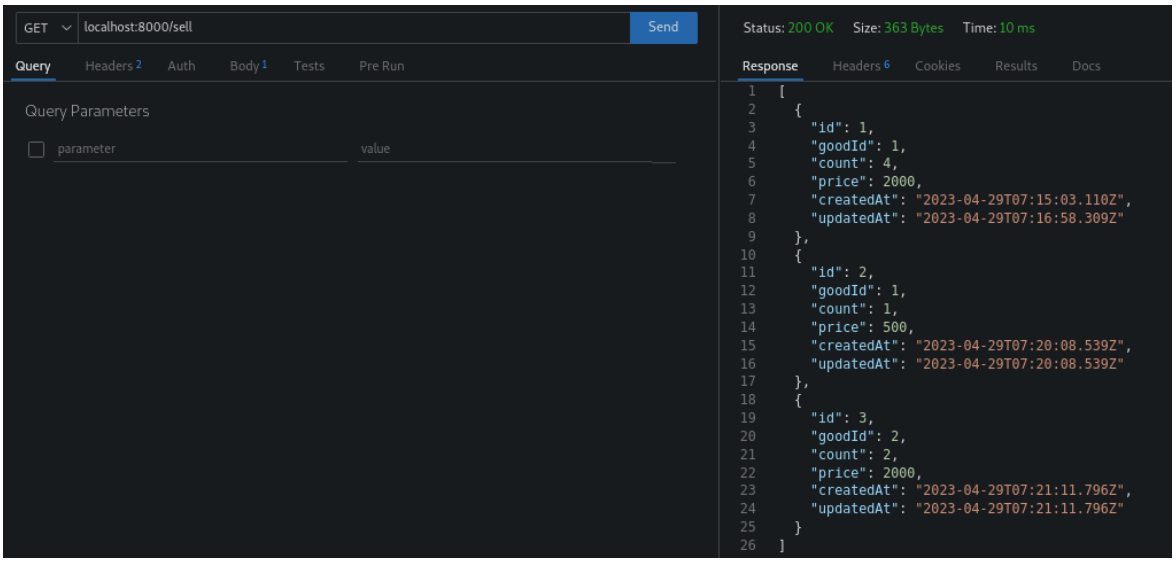
GET /good/:id



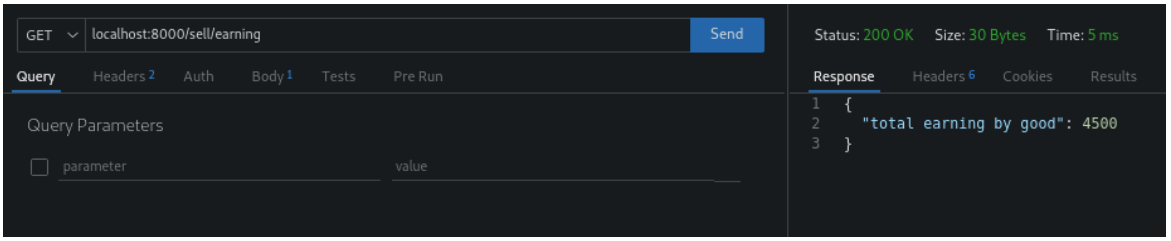
GET /sell/earning?name=name



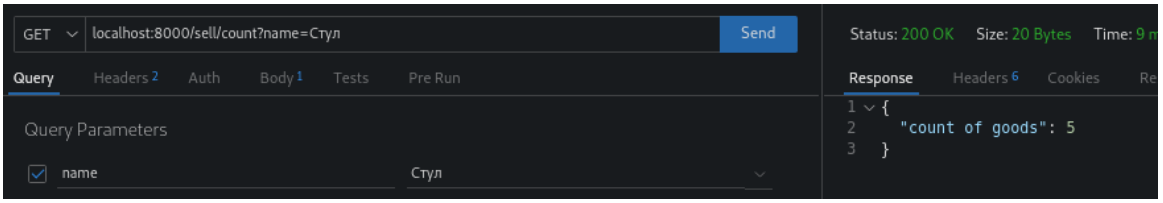
GET /sell



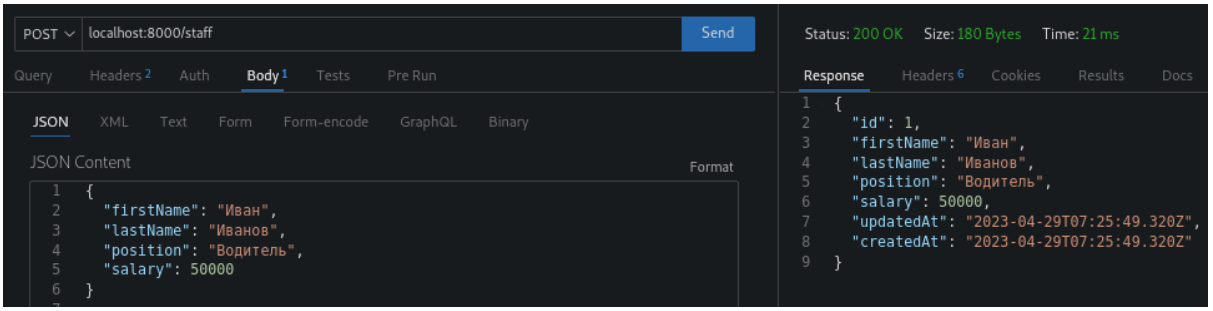
GET /sell/earning



GET /sell/count?name=name



POST /staff



Вывод

В ходе работы было реализовано RESTful API средствами express + typescript для платформы администратора интернет-магазина.