Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет инфокоммуникационных технологий

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 2

по теме: Разработка RESTful API
по дисциплине: Бэк-энд разработка

Специальность:
09.03.03 Мобильные и сетевые технологии

Выполнила:

Малютина Е.А., К33402

Проверил:

Добряков Д.И.

Санкт-Петербург,

2024

Задание.

По выбранному варианту необходимо будет реализовать RESTful API средствами express + typescript (используя ранее написанный boilerplate).

Был выбран вариант интернет-магазина.

Ход работы.

По выбранному варианту было реализовано приложение с добавлением, чтением и обновлением товара, а также регистрацией и авторизацией пользователя с помощью пары логин-пароль и JWT-токена. Данные хранятся в SQLite базе данных. Структура проекта представлена ниже (Рисунок 1).

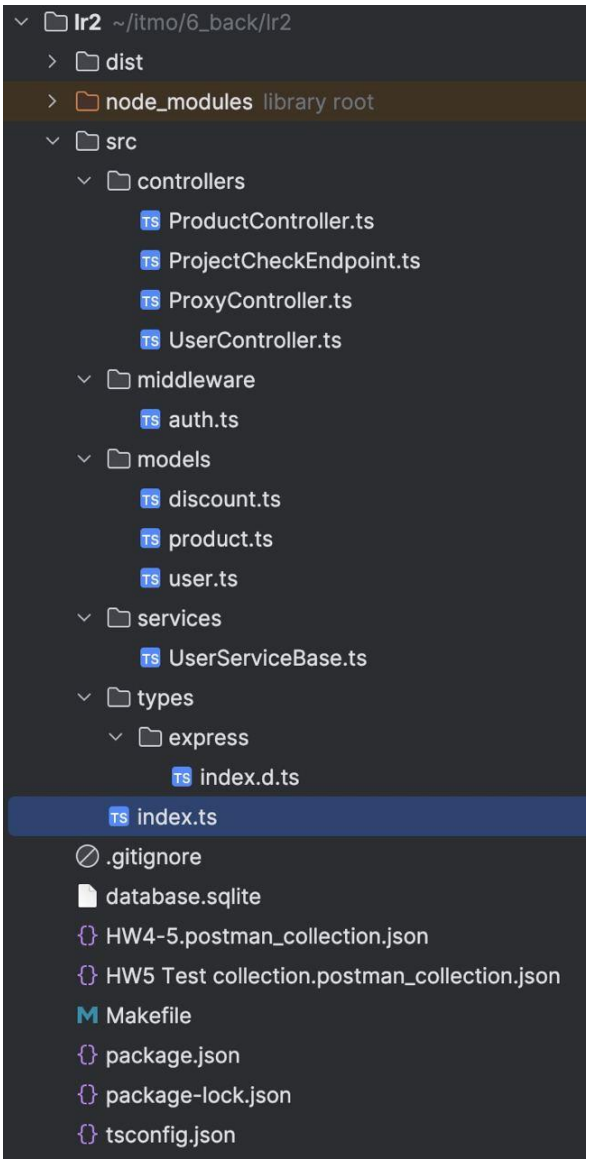

Рисунок 1 – Структура проекта

Написанные модели представлены ниже (Рисунок 2).

discount.ts

```ts
import {Table, Column, Model, PrimaryKey,
    AutoIncrement, ForeignKey, BelongsTo}
    from 'sequelize-typescript'
import {User} from './user.js'
import {Product} from './product.js'

2 usages
@Table
export class Discount extends Model {
    @PrimaryKey
    @AutoIncrement
    @Column
    declare id: number

    @ForeignKey( relatedClassGetter: () => User)
    @Column
    declare userId: number

    @ForeignKey( relatedClassGetter: () => Product)
    @Column
    declare productId: number

    @Column
    declare quantity: number

    @Column
    declare totalPrice: number

    @BelongsTo( associatedClassGetter: () => User)
    user: User

    @BelongsTo( associatedClassGetter: () => Product)
    product: Product
}
```

product.ts

```ts
import {Table, Column, Model, PrimaryKey,
    AutoIncrement, Default}
    from 'sequelize-typescript'

5+ usages
@Table
export class Product extends Model {
    @PrimaryKey
    @AutoIncrement
    @Column
    declare id: number

    @Column
    declare name: string

    @Column
    declare category: string

    @Column
    declare price: number

    @Default( value: 0)
    @Column
    declare stock: number

    @Default( value: false)
    @Column
    declare discount: number
}
```

user.ts

```ts
import {Table, Column, Model, Unique, PrimaryKey,
    AutoIncrement, Default} from 'sequelize-typescript';

5+ usages
@Table
export class User extends Model {
    @Unique
    @PrimaryKey
    @AutoIncrement
    @Column
    declare id: number;

    @Column
    declare firstName: string;

    @Column
    declare lastName: string;

    @Column
    declare email: string;

    @Column
    declare password: string;

    @Default( value: false)
    @Column
    declare isAdmin: boolean;
}
```

Рисунок 2 – Модели

Также осуществлена проверка токена (Рисунок 3).

auth.ts

```ts
import {NextFunction, Request, Response} from 'express'
import jwt from 'jsonwebtoken'

3 usages
export const authMiddleware = (req: Request, res: Response, next: NextFunction): void => {
    const token : string  = req.headers.authorization?.split( separator: ' ')[1]
    if (!token) {
        res.status( code: 401).json( body: { error: `Unauthorized: token is '${token}'` })
        return
    }

    try {
        req.user = jwt.verify(token, 'SECRET_KEY') as { id: number, isAdmin: boolean }
        next()
    } catch (error) {
        res.status( code: 401).json( body: { error: `Invalid token: token is '${token}'` })
    }
}
```

Рисунок 3 – auth.ts

Остальные операции также были написаны в соответствующих директориях и файлах (Рисунок 4).

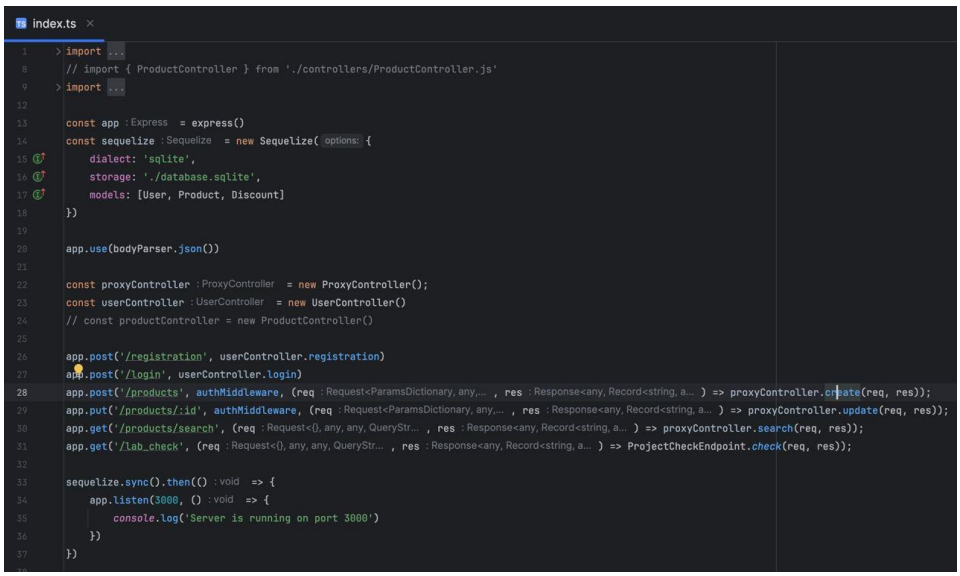

Рисунок 4 – Остальная логика приложения

Также написанное приложение было протестировано с помощью запросов в Postman (Рисунки 5-7).

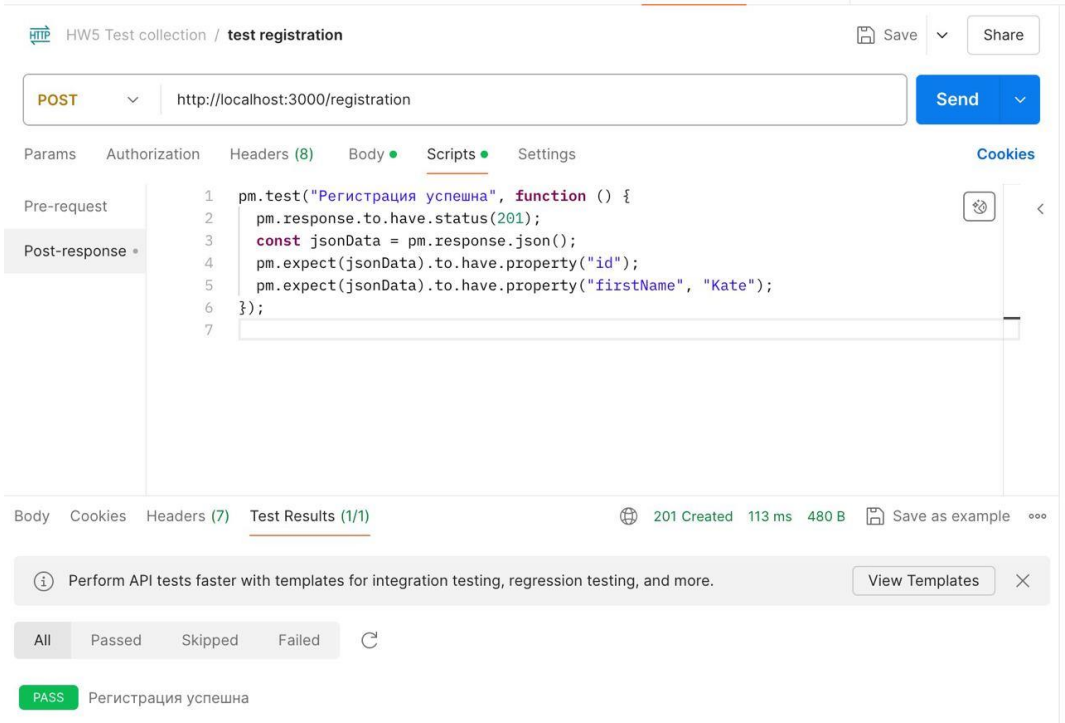

Рисунок 5 – Проверка регистрации

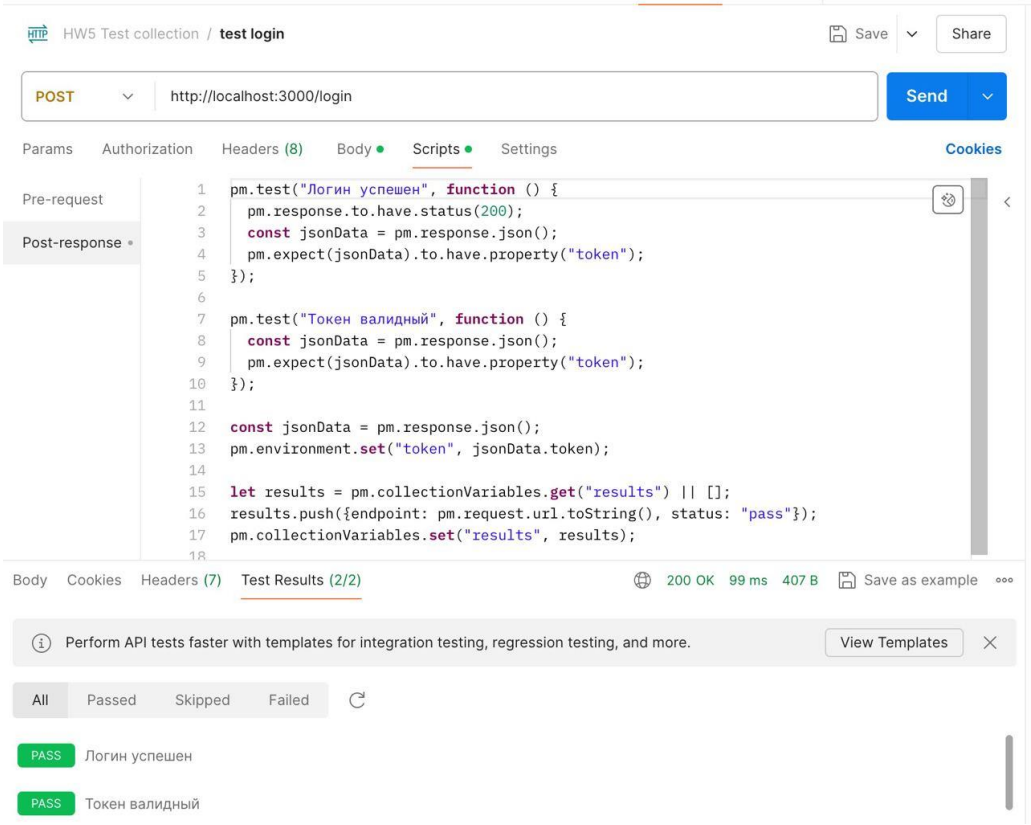

Рисунок 6 – Проверка авторизации

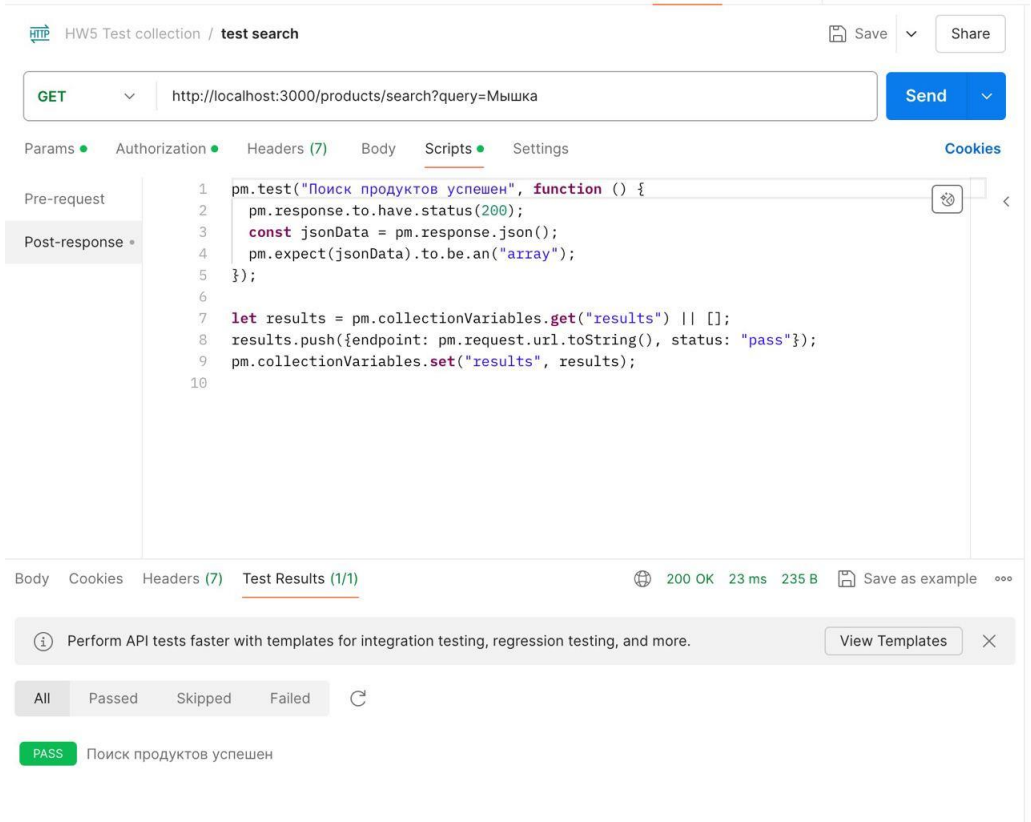

Рисунок 7 – Проверка поиска товара

Выводы.

В ходе выполнения лабораторной работы был реализован микросервис по варианту интернет-магазина с необходимыми запросами и работой с пользователем, который был основан на ранее написанном шаблоне. Благодаря использованию написанного шаблонного репозитория удалось существенно сократить время на имплементацию задания.

Сервис включает в себя регистрацию и авторизацию пользователя, проверку токена доступа, добавление товара и его обновление с поиском. Данных хранятся в базе данных SQLite, обрабатываются на сервере с помощью TypeScript.