**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ**

**ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Дисциплина:** Бэк-энд разработка

Отчет

# Лабораторная работа 2

Выполнил: Зайцев Кирилл   
Дмитриевич

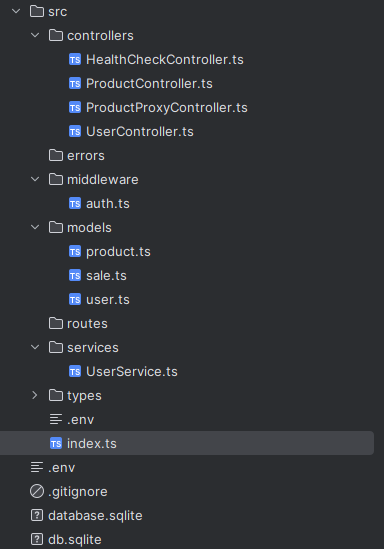
Группа K33402

Проверил: Добряков Д. И.

Санкт-Петербург

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

1. По выбранному варианту было составлено минимальное приложение с простым функционалом пользователей и товаров.



2.Создадим модели user.ts:

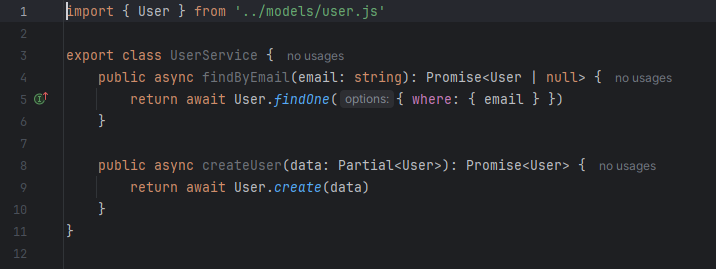
import { Table, Column, Model, Unique, PrimaryKey, AutoIncrement, Default } from 'sequelize-typescript';  
  
@Table  
export class User extends Model {  
 @Unique  
 @PrimaryKey  
 @AutoIncrement  
 @Column  
 declare id: number;  
  
 @Column  
 declare firstName: string;  
  
 @Column  
 declare lastName: string;  
  
 @Column  
 declare email: string;  
  
 @Column  
 declare password: string;  
  
 @Default(false)  
 @Column  
 declare isAdmin: boolean;  
}

sale.ts:

import {  
 Table,  
 Column,  
 Model,  
 PrimaryKey,  
 AutoIncrement,  
 ForeignKey,  
 BelongsTo  
} from 'sequelize-typescript'  
import { User } from './user.js'  
import { Product } from './product.js'  
  
@Table  
export class Sale extends Model {  
 @PrimaryKey  
 @AutoIncrement  
 @Column  
 declare id: number  
  
 @ForeignKey(() => User)  
 @Column  
 declare userId: number  
  
 @ForeignKey(() => Product)  
 @Column  
 declare productId: number  
  
 @Column  
 declare quantity: number  
  
 @Column  
 declare totalPrice: number  
  
 @BelongsTo(() => User)  
 user: User  
  
 @BelongsTo(() => Product)  
 product: Product  
}

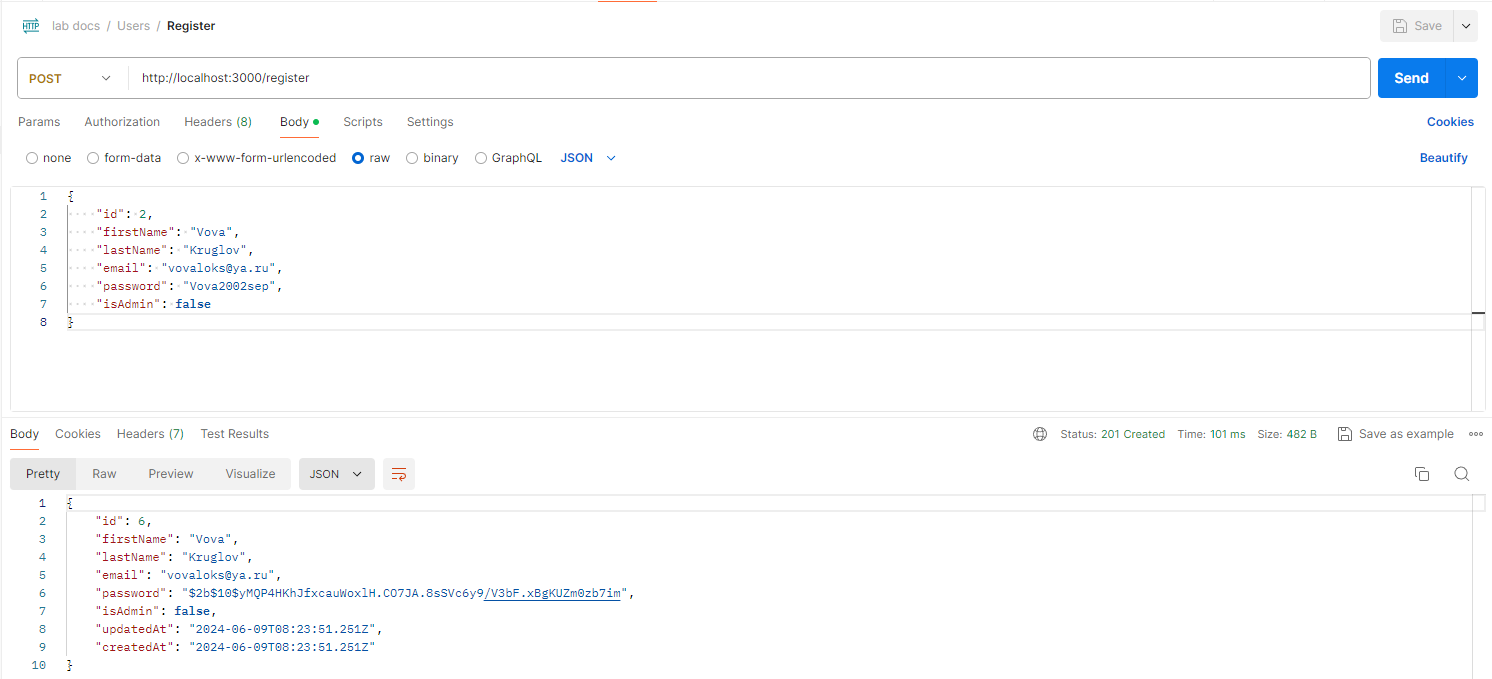
product.ts:

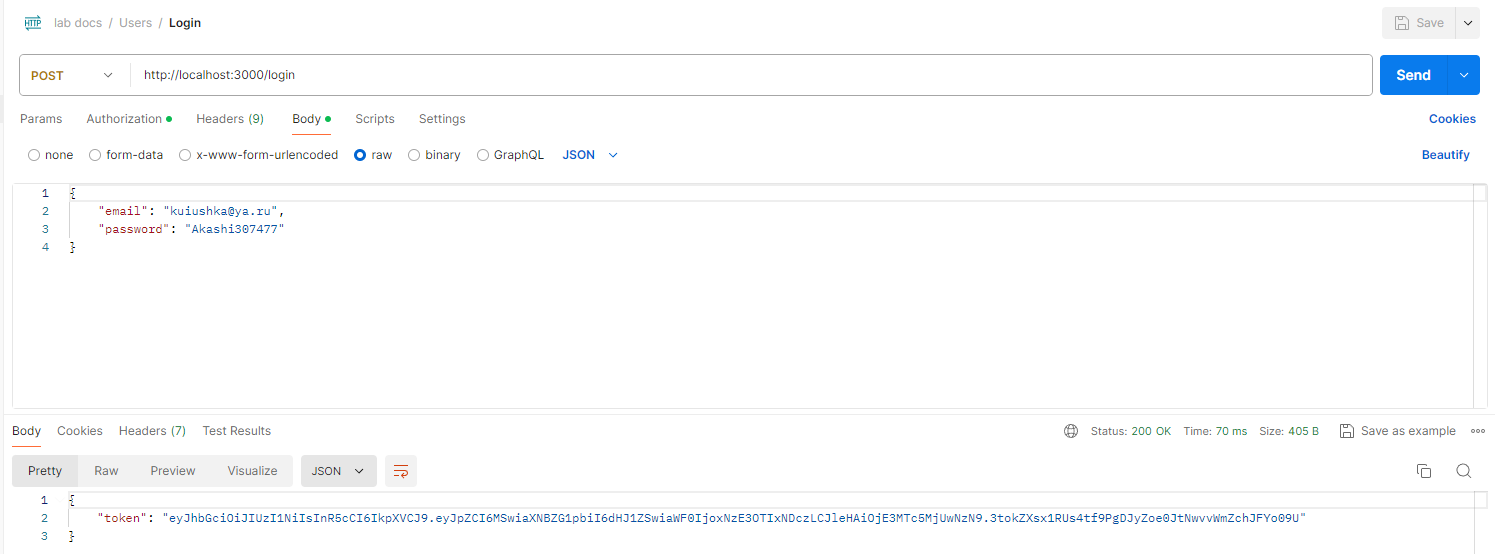
import {  
 Table,  
 Column,  
 Model,  
 PrimaryKey,  
 AutoIncrement,  
 Default,  
} from 'sequelize-typescript'  
  
@Table  
export class Product extends Model {  
 @PrimaryKey  
 @AutoIncrement  
 @Column  
 declare id: number  
  
 @Column  
 declare name: string  
  
 @Column  
 declare category: string  
  
 @Column  
 declare price: number  
  
 @Default(0)  
 @Column  
 declare stock: number  
  
 @Default(false)  
 @Column  
 declare discount: number  
}

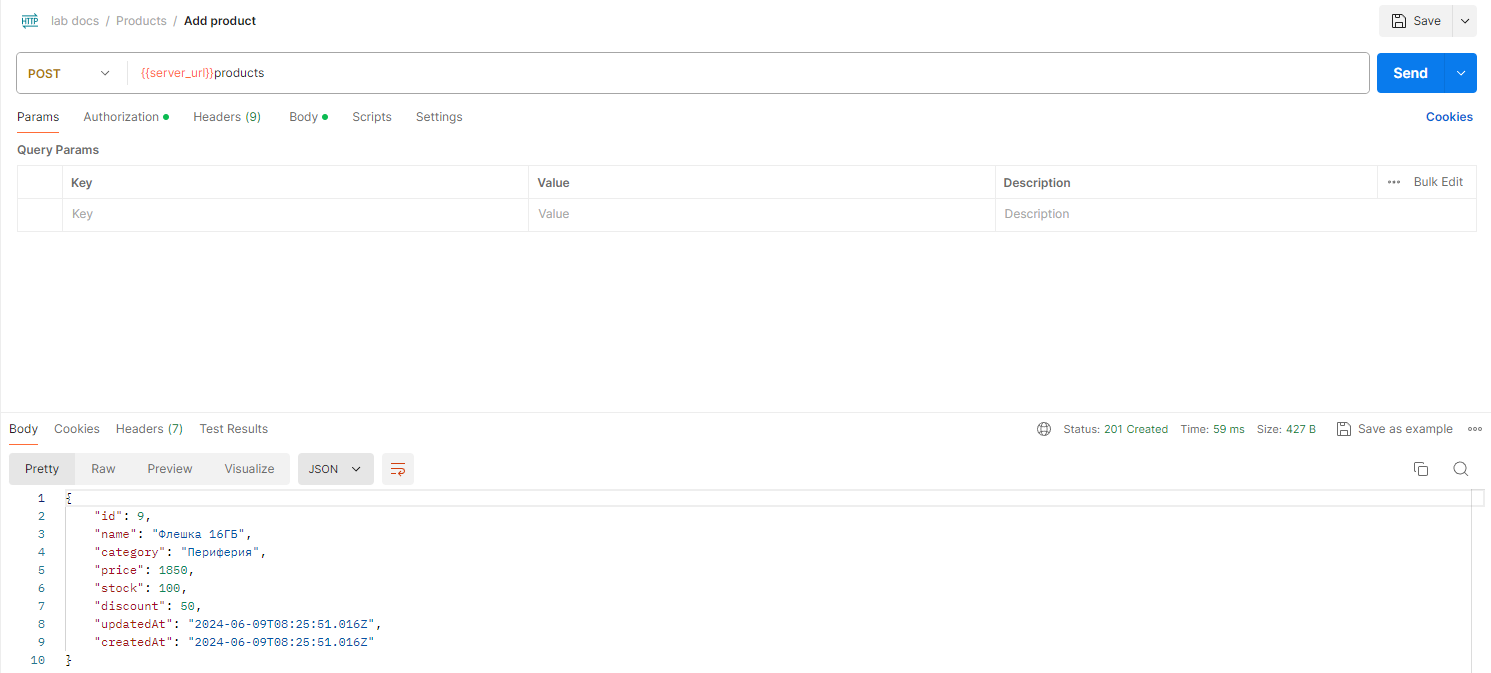
5. Создадим сервис для работы с юзером и veddleware для ограничения доступа к функционалу:  


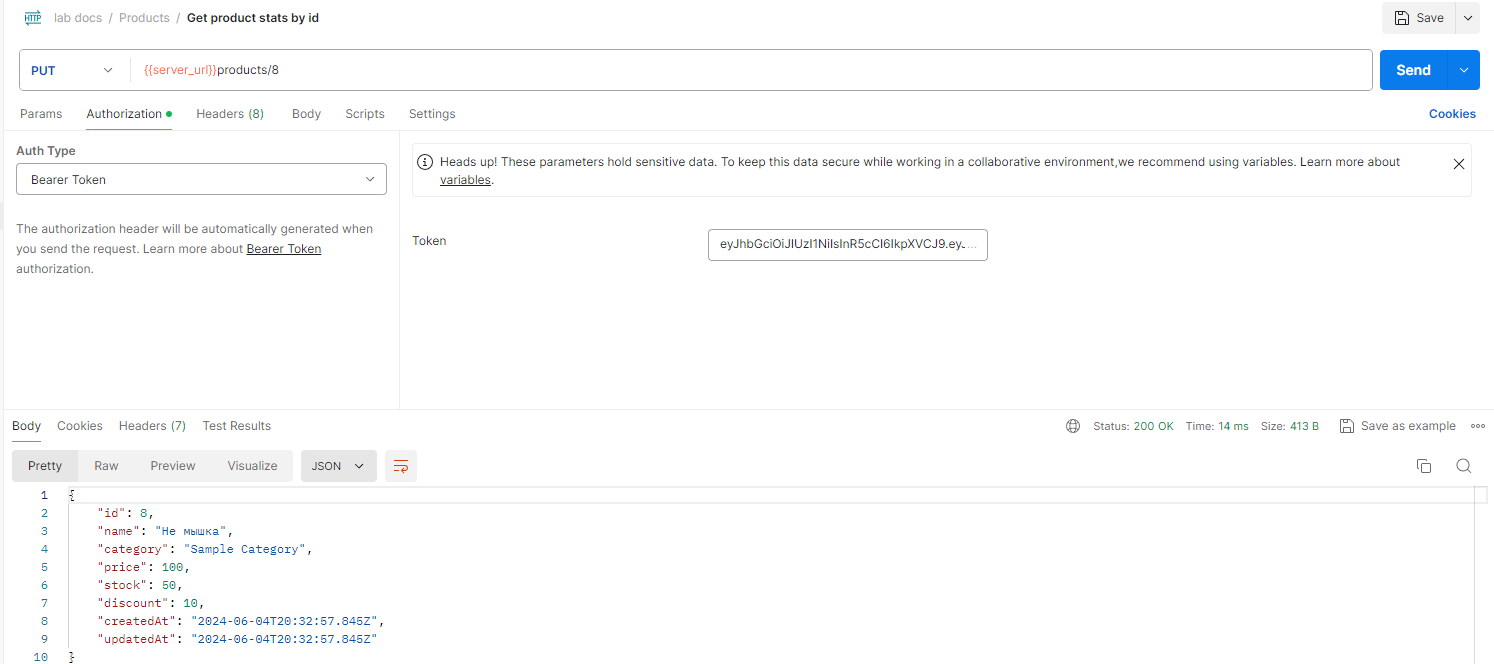
Auth.ts:

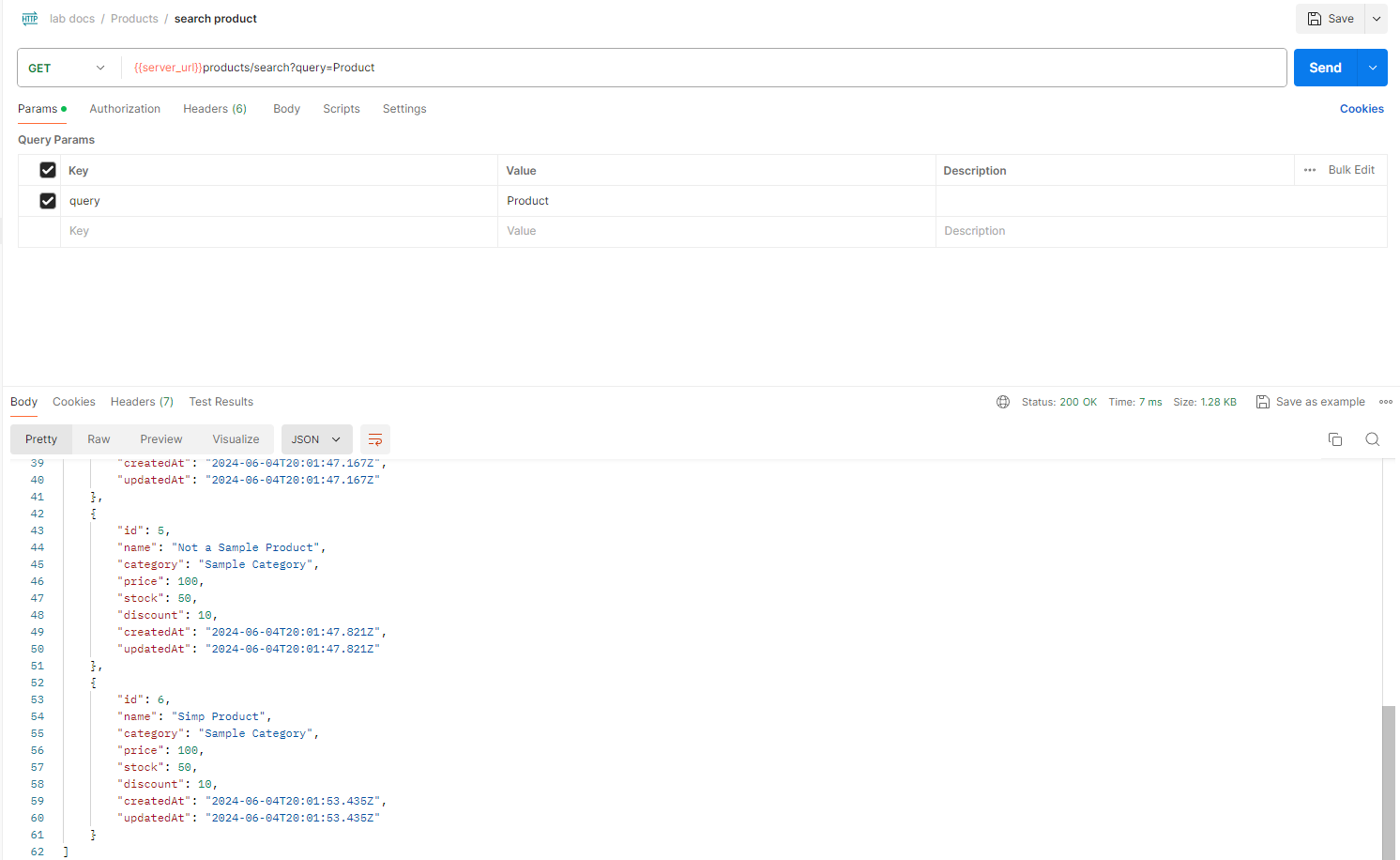
import {NextFunction, Request, Response} from 'express'  
import jwt from 'jsonwebtoken'  
  
export const authMiddleware = (req: Request, res: Response, next: NextFunction): void => {  
 const token = req.headers.authorization?.split(' ')[1]  
 if (!token) {  
 res.status(401).json({ error: 'Unauthorized' })  
 return  
 }  
  
 try {  
 req.user = jwt.verify(token, 'SECRET\_KEY') as { id: number, isAdmin: boolean }  
 next()  
 } catch (error) {  
 res.status(401).json({ error: 'Invalid token' })  
 }  
}

Примеры выполнения запросов в postman:  
register  


Login:  


Post product:  


Get product by id:  


Search:  


**Вывод:**  
В этой лабораторной работе был разработан сервис для интернет-магазина, используя шаблон, который был создан в ходе предыдущей лабораторной работы.

Начало формы

Конец формы