

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа 2

Выполнил: Рашевский
Вячеслав Васильевич

Группа К33402

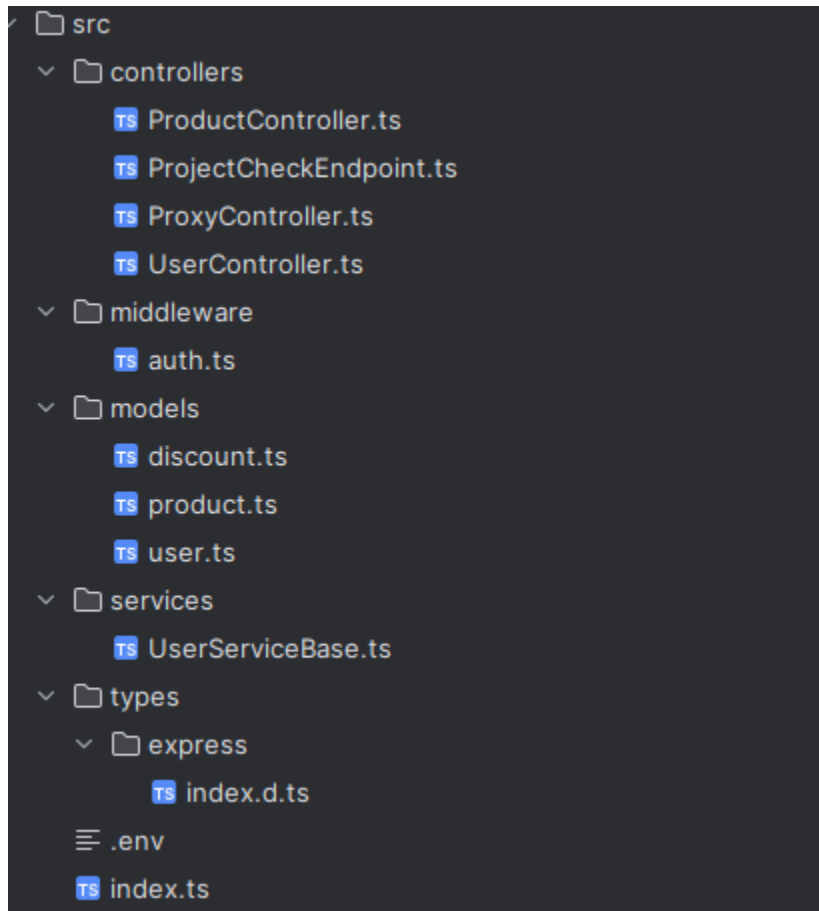
Проверил:
Добряков Д. И.

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:

```
user.ts x
1  import { Table, Column, Model, Unique, PrimaryKey, AutoIncrement, Default } from 'sequelize-typescript';
2
3  @Table Show usages
4  export class User extends Model {
5      @Unique
6      @PrimaryKey
7      @AutoIncrement
8      @Column
9      declare id: number;
10
11      @Column
12      declare firstName: string;
13
14      @Column
15      declare lastName: string;
16
17      @Column
18      declare email: string;
19
20      @Column
21      declare password: string;
22
23      @Default( value: false)
24      @Column
25      declare isAdmin: boolean;
26  }
27
```

discount.ts:

```
TS user.ts    TS discount.ts ×
1  import {
2      Table,
3      Column,
4      Model,
5      PrimaryKey,
6      AutoIncrement,
7      ForeignKey,
8      BelongsTo
9  } from 'sequelize-typescript'
10 import { User } from './user.js'
11 import { Product } from './product.js'
12
13 @Table Show usages
14 export class Discount extends Model {
15     @PrimaryKey
16     @AutoIncrement
17     @Column
18     declare id: number
19
20     @ForeignKey(relatedClassGetter: () => User)
21     @Column
22     declare userId: number
23
24     @ForeignKey(relatedClassGetter: () => Product)
25     @Column
26     declare productId: number
27
28     @Column
29     declare quantity: number
30
31     @Column
32     declare totalPrice: number
33
34     @BelongsTo(associatedClassGetter: () => User)
35     user: User
36
37     @BelongsTo(associatedClassGetter: () => Product)
38     product: Product
39 }
40
```

product.ts:

```
ts user.ts    ts discount.ts    ts product.ts x
1  import {
2      Table,
3      Column,
4      Model,
5      PrimaryKey,
6      AutoIncrement,
7      Default,
8  } from 'sequelize-typescript'
9
10 @Table  Show usages
11 export class Product extends Model {
12     @PrimaryKey
13     @AutoIncrement
14     @Column
15     declare id: number
16
17     @Column
18     declare name: string
19
20     @Column
21     declare category: string
22
23     @Column
24     declare price: number
25
26     @Default(value: 0)
27     @Column
28     declare stock: number
29
30     @Default(value: false)
31     @Column
32     declare discount: number
33 }
34
```

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

```
user.ts  discount.ts  product.ts  auth.ts  UserServiceBase.ts x
1  import { User } from '../models/user.js'
2
3  export class UserServiceBase { no usages
4      public async findByEmail(email: string): Promise<User | null> { no usages
5          return await User.findOne( options: { where: { email } })
6      }
7
8      public async createUser(data: Partial<User>): Promise<User> { no usages
9          return await User.create(data)
10     }
11 }
12
```

Auth.ts:

```
user.ts  discount.ts  product.ts  auth.ts x
1  import {NextFunction, Request, Response} from 'express'
2  import jwt from 'jsonwebtoken'
3
4  export const authMiddleware = (req: Request, res: Response, next: NextFunction): void => { Show usages
5      const token : string = req.headers.authorization?.split( separator: ' ')[1]
6      if (!token) {
7          res.status( code: 401 ).json( body: { error: 'Unauthorized' })
8          return
9      }
10
11      try {
12          req.user = jwt.verify(token, 'SECRET_KEY') as { id: number, isAdmin: boolean }
13          next()
14      } catch (error) {
15          res.status( code: 401 ).json( body: { error: 'Invalid token' })
16      }
17  }
18
19
```

Примеры выполнения запросов в postman:

registration

lab docs / Users / Register

POST http://localhost:3000/register

Params Authorization Headers (8) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "id": 2,
3   "firstName": "Vova",
4   "lastName": "Kruglov",
5   "email": "vovaloks@ya.ru",
6   "password": "Vova2002sep",
7   "isAdmin": false
8 }
```

Body Cookies Headers (7) Test Results

Status: 201 Created Time: 101 ms Size: 482 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 6,
3   "firstName": "Vova",
4   "lastName": "Kruglov",
5   "email": "vovaloks@ya.ru",
6   "password": "$2b$10$yMQ4KHJfxcawoxlH.C07JA.8sSV6y9/V3bF.xBgKUzm8zb7im",
7   "isAdmin": false,
8   "updatedAt": "2024-06-09T08:23:51.251Z",
9   "createdAt": "2024-06-09T08:23:51.251Z"
10 }
```

Login:

lab docs / Users / Login

POST http://localhost:3000/login

Params Authorization Headers (9) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "email": "kuiushka@ya.ru",
3   "password": "Akash1307477"
4 }
```

Body Cookies Headers (7) Test Results

Status: 200 OK Time: 70 ms Size: 405 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZiI6Im90dWVkaW80IiwiaWF0IjE6dHJ1ZSwiaWF0IjoxNzE3OTI4NDczLCJleHAiOiE3MTc5MjUwNzN9.3tokZXsx1RUs4tf9PgDJyZoe0JtNvvWmZehJFYe09U"
3 }
```


Post products:

lab docs / Products / Add product

POST

{{server_url}}products

Send

ParamsAuthorizationHeaders (9)BodyScriptsSettings

Query Params

Key	Value	Description	...	Bulk Edit
Key	Value	Description		

BodyCookiesHeaders (7)Test Results

Status: 201 CreatedTime: 59 msSize: 427 BSave as example

PrettyRawPreviewVisualizeJSON

```
1 {
2   "id": 9,
3   "name": "Флешка 16ГБ",
4   "category": "Периферия",
5   "price": 1850,
6   "stock": 100,
7   "discount": 50,
8   "updatedAt": "2024-06-09T08:25:51.816Z",
9   "createdAt": "2024-06-09T08:25:51.816Z"
10 }
```

Get product by id:

lab docs / Products / Get product stats by id

PUT

{{server_url}}products/8

Send

ParamsAuthorizationHeaders (8)BodyScriptsSettings

Auth Type

Bearer Token

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables.

The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization.

Token

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...

BodyCookiesHeaders (7)Test Results

Status: 200 OKTime: 14 msSize: 413 BSave as example

PrettyRawPreviewVisualizeJSON

```
1 {
2   "id": 8,
3   "name": "Не мышка",
4   "category": "Sample Category",
5   "price": 100,
6   "stock": 50,
7   "discount": 10,
8   "updatedAt": "2024-06-04T20:32:57.845Z",
9   "createdAt": "2024-06-04T20:32:57.845Z"
10 }
```

Search:

lab docs / Products / search product

OET

{{server_url}}products/search?query=Product

Send

Params Authorization Headers (6) Body Scripts Settings

Query Params

<input checked="" type="checkbox"/> Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/> query	Product			
<input type="checkbox"/> Key	Value	Description		

Body Cookies Headers (7) Test Results

Status: 200 OK Time: 7 ms Size: 1.28 KB Save as example

Pretty Raw Preview Visualize JSON

```
39      "createdAt": "2024-06-04T20:01:47.167Z",
40      "updatedAt": "2024-06-04T20:01:47.167Z"
41    },
42    {
43      "id": 5,
44      "name": "Not a Sample Product",
45      "category": "Sample Category",
46      "price": 100,
47      "stock": 50,
48      "discount": 10,
49      "createdAt": "2024-06-04T20:01:47.021Z",
50      "updatedAt": "2024-06-04T20:01:47.021Z"
51    },
52    {
53      "id": 6,
54      "name": "Simp Product",
55      "category": "Sample Category",
56      "price": 100,
57      "stock": 50,
58      "discount": 10,
59      "createdAt": "2024-06-04T20:01:53.435Z",
60      "updatedAt": "2024-06-04T20:01:53.435Z"
61    }
62  ]
```

Вывод:

В рамках этой лабораторной работы был создан сервис для интернет-магазина. В качестве основы для разработки использовался шаблон, созданный в ходе предыдущей лабораторной работы. Этот шаблон включал в себя такие технологии, как TypeScript, Sequelize и Express. Благодаря уже готовому шаблону удалось значительно сократить время на начальную настройку проекта и сосредоточиться на разработке функционала, специфичного для интернет-магазина. Сервис включает в себя функциональные возможности для управления товарами, обработки заказов, а также взаимодействия с базой данных, обеспечивая надежную и эффективную работу интернет-магазина.