

**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
        }
    })
    count!: number;

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

POST localhost:8000/auth/register

Status: 201 Created Size: 183 Bytes Time: 76 ms

JSON Content

```

1 {
2   "email": "baa@mail.com",
3   "password": "123"
4 }

```

Response

```

1 {
2   "id": 1,
3   "email": "baa@mail.com",
4   "password": "$2b$08$hc2Lj9z4V43KyKM156P3fuPfwGmLxR4HnLxKAEtB3cJv8U2ehTKtW",
5   "updatedAt": "2023-04-29T07:11:06.903Z",
6   "createdAt": "2023-04-29T07:11:06.903Z"
7 }

```

/auth/login

POST localhost:8000/auth/login

Status: 200 OK Size: 133 Bytes Time: 29 ms

JSON Content

```

1 {
2   "email": "baa@mail.com",
3   "password": "123"
4 }

```

Response

```

1 {
2   "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiaWF0IjoxNjgyNzUyMjkyfQ.0zkelTqK58ltUR9fxJG9iAe0WuuVg6A3V4PRnJyH4Ak"
3 }

```

POST /good

POST localhost:8000/good

Status: 200 OK Size: 126 Bytes Time: 22 ms

JSON Content

```

1 {
2   "name": "Cryn",
3   "count": 5,
4   "price": 500
5 }

```

Response

```

1 {
2   "id": 1,
3   "name": "Cryn",
4   "count": 5,
5   "price": 500,
6   "updatedAt": "2023-04-29T07:14:15.287Z",
7   "createdAt": "2023-04-29T07:14:15.287Z"
8 }

```

POST /sell

POSTlocalhost:8000/sellSend

QueryHeaders 2AuthBody 1TestsPre Run

JSONXMLTextFormForm-encodeGraphQLBinary

JSON ContentFormat

```
1 {
2   "name": "Стул",
3   "count": 2
4 }
5
```

Status: 200 OKSize: 120 BytesTime: 28 ms

ResponseHeaders 6CookiesResultsDocs

```
1 {
2   "id": 1,
3   "count": 2,
4   "price": 1000,
5   "goodId": 1,
6   "updatedAt": "2023-04-29T07:15:03.110Z",
7   "createdAt": "2023-04-29T07:15:03.110Z"
8 }
```

GET /good

GETlocalhost:8000/goodSend

QueryHeaders 2AuthBody 1TestsPre Run

JSONXMLTextFormForm-encodeGraphQLBinary

JSON ContentFormat

```
1 {
2   "name": "Стул",
3   "count": 2
4 }
5
```

Status: 200 OKSize: 128 BytesTime: 6 ms

ResponseHeaders 6CookiesResultsDocs

```
1 [
2   {
3     "id": 1,
4     "name": "Стул",
5     "count": 3,
6     "price": 500,
7     "createdAt": "2023-04-29T07:14:15.287Z",
8     "updatedAt": "2023-04-29T07:15:03.099Z"
9   }
10 ]
```

PATCH /sell/:id

PATCHlocalhost:8000/sell/1Send

QueryHeaders 2AuthBody 1TestsPre Run

JSONXMLTextFormForm-encodeGraphQLBinary

JSON ContentFormat

```
1 {
2   "name": "Стул",
3   "count": 4
4 }
5
```

Status: 200 OKSize: 120 BytesTime: 38 ms

ResponseHeaders 6CookiesResultsDocs

```
1 {
2   "id": 1,
3   "goodId": 1,
4   "count": 4,
5   "price": 2000,
6   "createdAt": "2023-04-29T07:15:03.110Z",
7   "updatedAt": "2023-04-29T07:16:58.309Z"
8 }
```

GET /good/:id

GETlocalhost:8000/good/1Send

QueryHeaders 2AuthBody 1TestsPre Run

JSONXMLTextFormForm-encodeGraphQLBinary

JSON ContentFormat

```
1 {
2   "name": "Стул",
3   "count": 4
4 }
5
```

Status: 200 OKSize: 126 BytesTime: 5 ms

ResponseHeaders 6CookiesResultsDocs

```
1 {
2   "id": 1,
3   "name": "Стул",
4   "count": 1,
5   "price": 500,
6   "createdAt": "2023-04-29T07:14:15.287Z",
7   "updatedAt": "2023-04-29T07:16:58.292Z"
8 }
```

GET /sell/earning?name=name

GETlocalhost:8000/sell/earning?name=СтулSend

QueryHeaders 2AuthBody 1TestsPre Run

Query Parameters

☒ nameСтул

Status: 200 OKSize: 30 BytesTime: 8 ms

ResponseHeaders 6CookiesResult

```
1 {
2   "total earning by good": 2000
3 }
```

GET /sell

GETlocalhost:8000/sellSend

QueryHeaders 2AuthBody 1TestsPre Run

Query Parameters

parameter

value

Status: 200 OKSize: 363 BytesTime: 10 ms

ResponseHeaders 6CookiesResultsDocs

```
1 [{
2   {
3     "id": 1,
4     "goodId": 1,
5     "count": 4,
6     "price": 2000,
7     "createdAt": "2023-04-29T07:15:03.110Z",
8     "updatedAt": "2023-04-29T07:16:58.309Z"
9   },
10  {
11    "id": 2,
12    "goodId": 1,
13    "count": 1,
14    "price": 500,
15    "createdAt": "2023-04-29T07:20:08.539Z",
16    "updatedAt": "2023-04-29T07:20:08.539Z"
17  },
18  {
19    "id": 3,
20    "goodId": 2,
21    "count": 2,
22    "price": 2000,
23    "createdAt": "2023-04-29T07:21:11.796Z",
24    "updatedAt": "2023-04-29T07:21:11.796Z"
25  }
26 }]
```

GET /sell/earning

GETlocalhost:8000/sell/earningSend

QueryHeaders 2AuthBody 1TestsPre Run

Query Parameters

parameter

value

Status: 200 OKSize: 30 BytesTime: 5 ms

ResponseHeaders 6CookiesResults

```
1 {
2   "total earning by good": 4500
3 }
```

GET /sell/count?name=name

GETlocalhost:8000/sell/count?name=СтулSend

QueryHeaders 2AuthBody 1TestsPre Run

Query Parameters

name

Стул

Status: 200 OKSize: 20 BytesTime: 9 ms

ResponseHeaders 6CookiesResults

```
1 {
2   "count of goods": 5
3 }
```

POST /staff

POSTlocalhost:8000/staffSend

QueryHeaders 2AuthBody 1TestsPre Run

JSONXMLTextFormForm-encodeGraphQLBinary

JSON Content

1 {
2 "firstName": "Иван",
3 "lastName": "Иванов",
4 "position": "Водитель",
5 "salary": 50000
6 }
7 }

Status: 200 OKSize: 180 BytesTime: 21 ms

ResponseHeaders 6CookiesResultsDocs

```
1 {
2   "id": 1,
3   "firstName": "Иван",
4   "lastName": "Иванов",
5   "position": "Водитель",
6   "salary": 50000,
7   "updatedAt": "2023-04-29T07:25:49.320Z",
8   "createdAt": "2023-04-29T07:25:49.320Z"
9 }
```

Вывод

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