

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ  
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Дисциплина:** Бэк-энд разработка

Отчет

Лабораторная работа 1  
Создание boilerplate

Выполнила:

Хорошкеева К. А.

Группа К33402

Проверил:

Добряков Д. И.

Санкт-Петербург

2022 г.

## Задача

Нужно написать свой boilerplate на express + sequelize / TypeORM + typescript.

Должно быть явное разделение на:

- модели
- контроллеры
- роуты
- сервисы для работы с моделями (реализуем паттерн “репозиторий”)

## Ход работы

Модель пользователя:

```
1  import ...
15
16  @DefaultScope( scopeGetter: () => ({
17    attributes: ['id', 'name', 'email']
18  }))
19  @Scopes( scopesGetter: () => ({
20    full: {
21      attributes: ['id', 'name', 'email', 'password']
22    }
23  }))
24  @Table
25  class User extends Model {
26    @AllowNull( allowNull: false)
27    @Column
28    name: string;
29
30    @IsEmail
31    @AllowNull( allowNull: false)
32    @Unique
33    @Column
34    email: string;
```

```

36     @AllowNull(allowNull: false)
37     @Column
38     password: string;
39
40     @BeforeCreate
41     @BeforeUpdate
42     static generatePasswordHash(instance: User) {
43         // Хэширование пароля
44         const {password} = instance;
45
46         if (instance.changed( key: 'password')) {
47             instance.password = hashPassword(password);
48         }
49     }

```

```

50
51     @BeforeValidate
52     static checkEmailChanged(instance: User) {
53         // Проверка что пользователь не пытается изменить почту
54         if (instance.isNewRecord) {
55             return;
56         }
57         if (instance.changed( key: 'email')) {
58             throw new Error('Нельзя менять электронную почту уже созданного пользователя');
59         }
60     }
61 }
62
63 export default User;

```

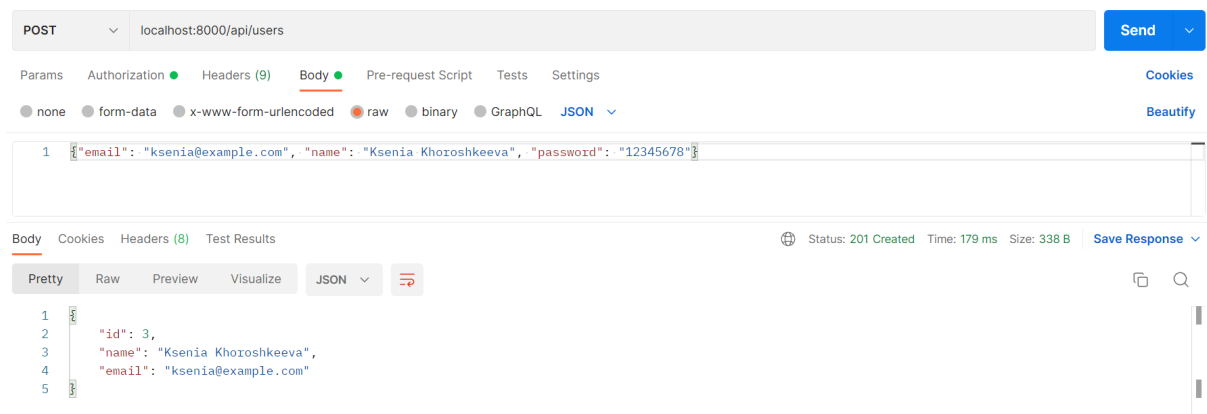
Модель refresh токена:

```

1  import ...
3
4  @Table
5  class RefreshToken extends Model {
6      @Unique
7      @AllowNull(allowNull: false)
8      @Column
9      token: string;
10
11      @ForeignKey(relatedClassGetter: () => User)
12      @Column
13      userId: number;
14  }
15
16  export default RefreshToken;

```

## Создание пользователя:



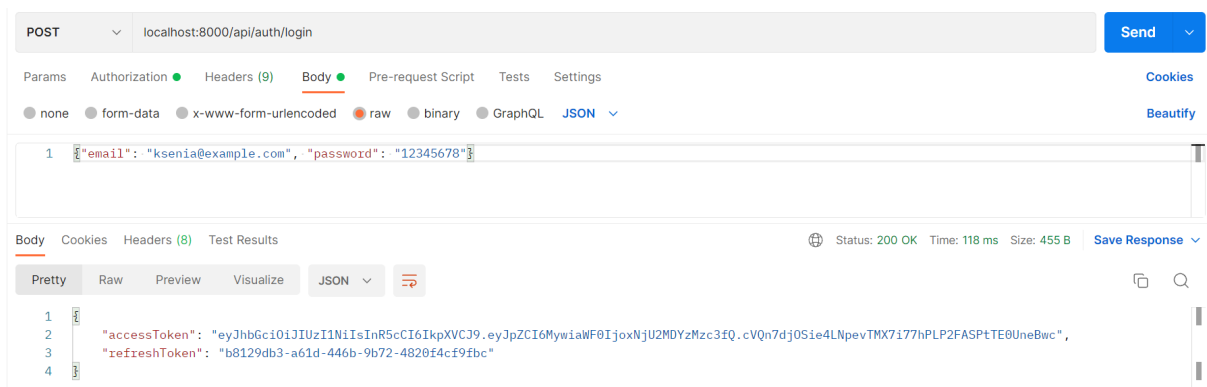
POST localhost:8000/api/users

Body: `{"email": "ksenia@example.com", "name": "Ksenia Khoroshkeeva", "password": "12345678"}`

Status: 201 Created Time: 179 ms Size: 338 B

Body: `{
 "id": 3,
 "name": "Ksenia Khoroshkeeva",
 "email": "ksenia@example.com"
}`

## Вход:



POST localhost:8000/api/auth/login

Body: `{"email": "ksenia@example.com", "password": "12345678"}`

Status: 200 OK Time: 118 ms Size: 455 B

Body: `{
 "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MywiaWF0IjoxNjU2MDYzMzc3fQ.cVQn7dJ0S1e4LNpevTMX7i77hPLP2FASPtTE0UeBwc",
 "refreshToken": "b8129db3-a61d-446b-9b72-4826f4cf9fbc"
}`

## Получение профиля:

GET localhost:8000/api/users/profile Send

Params Authorization Headers (7) **Body** Pre-request Script Tests Settings Cookies

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

This request does not have a body

Body Cookies Headers (8) Test Results Status: 200 OK Time: 27 ms Size: 333 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 3,
3   "name": "Ksenia Khoroshkeeva",
4   "email": "ksenia@example.com"
5 }
```

## Изменение профиля:

PATCH localhost:8000/api/users/profile Send

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings Cookies

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

```
1 { "name": "Neksenia Nekhoroshkeeva", "password": "87654321" }
```

Body Cookies Headers (8) Test Results Status: 200 OK Time: 156 ms Size: 337 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 3,
3   "name": "Neksenia Nekhoroshkeeva",
4   "email": "ksenia@example.com"
5 }
```

## Вывод

В ходе выполнения лабораторной работы я создала boilerplate приложения на express.js с моделью пользователя и функционалом авторизации