

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №2

Выполнил:

Шалунов Андрей

Группа К3340

Проверил:

Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

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

Ход работы

1. Файл настройки переменных окружения

Вынес в *config/settings.ts* или *.env* все параметры: порт приложения, доступ к Postgres, пути к сущностям.

2. Инициализация DataSource

В *src/config/data-source.ts* создал *new DataSource({...})*.

```
import 'reflect-metadata';

import { DataSource } from 'typeorm';

import SETTINGS from '../settings';

export const AppDataSource = new DataSource({

  type: 'postgres',

  host: SETTINGS.DB_HOST,

  port: SETTINGS.DB_PORT,

  username: SETTINGS.DB_USER,

  password: SETTINGS.DB_PASSWORD,

  database: SETTINGS.DB_NAME,

  entities: [SETTINGS.DB_ENTITIES],

  logging: false,

  synchronize: true,

});
```

3. Модель пользователя

В *src/models/user.entity.ts* описал *class User* с полями *user_id*, *name*, *email*, *password*, *phone*.

```
import { Entity, PrimaryGeneratedColumn, Column, OneToMany, CreateDateColumn } from 'typeorm';

import { Property } from '../property.entity';
import { Booking } from '../booking.entity';
import { Message } from '../message.entity';

@Entity()

export class User {

    @PrimaryGeneratedColumn()

    user_id!: number;

    @Column()

    name!: string;

    @Column({unique: true})

    email!: string;

    @Column({ nullable: true })

    phone?: string;

    @Column()

    password!: string;

    @OneToMany(() => Property, p => p.owner)

    properties!: Property[];

    @OneToMany(() => Booking, b => b.renter)

    bookings!: Booking[];

    @OneToMany(() => Message, m => m.sender)

    sentMessages!: Message[];
```

```
@OneToMany(() => Message, m => m.recipient)

receivedMessages!: Message[];

}
```

4. Утилиты для пароля

В *src/utls/hash-password.ts* и *check-password.ts* реализовал хеширование и проверку пароля через *bcrypt*.

```
import bcrypt from 'bcrypt';

const hashPassword = (password: string): string => {

    return bcrypt.hashSync(password, bcrypt.genSaltSync(8));

};

export default hashPassword;
```

5. DTO и валидация

Перешел от интерфейсов к классам с декораторами *class-validator* в папке *src/dto*:

- *booking.dto.ts*,
- *message.dto.ts*,
- *photo.dto.ts*,
- *property.dto.ts*,
- *user.dto.ts*,
- *auth.dto.ts* (*RegisterDto*, *LoginDto*).

В контроллерах через *plainToInstance()* + *validateOrReject()* проверяю входящие данные.

6. Middleware

JWT-аутентификация (*auth.middleware.ts*): читает заголовок *Authorization*, проверяет токен и кладём в *req.user*.

Глобальный обработчик ошибок (*error.middleware.ts*): ловит все `next(err)` и форматирует ответ (используя `err.status`).

7. Сервисы

В *src/services/* описаны классы с `static`-методами:

- AuthService (register/login с `bcrypt` и `jsonwebtoken`)
- BookingService, MessageService, PhotoService, PropertyService, UserService (CRUD-операции через репозитории `TypeORM`)

8. Контроллеры

В *src/controllers/* реализовали классы с `static async` методами.

Каждый метод:

- Валидирует DTO.
- Вызывает сервис.

9. Роутинг

В *src/routes/* для каждого ресурса:

- Подключил `authMiddleware` к защищённым эндпоинтам.
- В маршрутах передаем прямо методы контроллера

```
import { Router } from "express";

import { AuthController } from '../controllers/auth.controller';

import { authMiddleware } from '../middlewares/auth.middleware';

const router = Router();

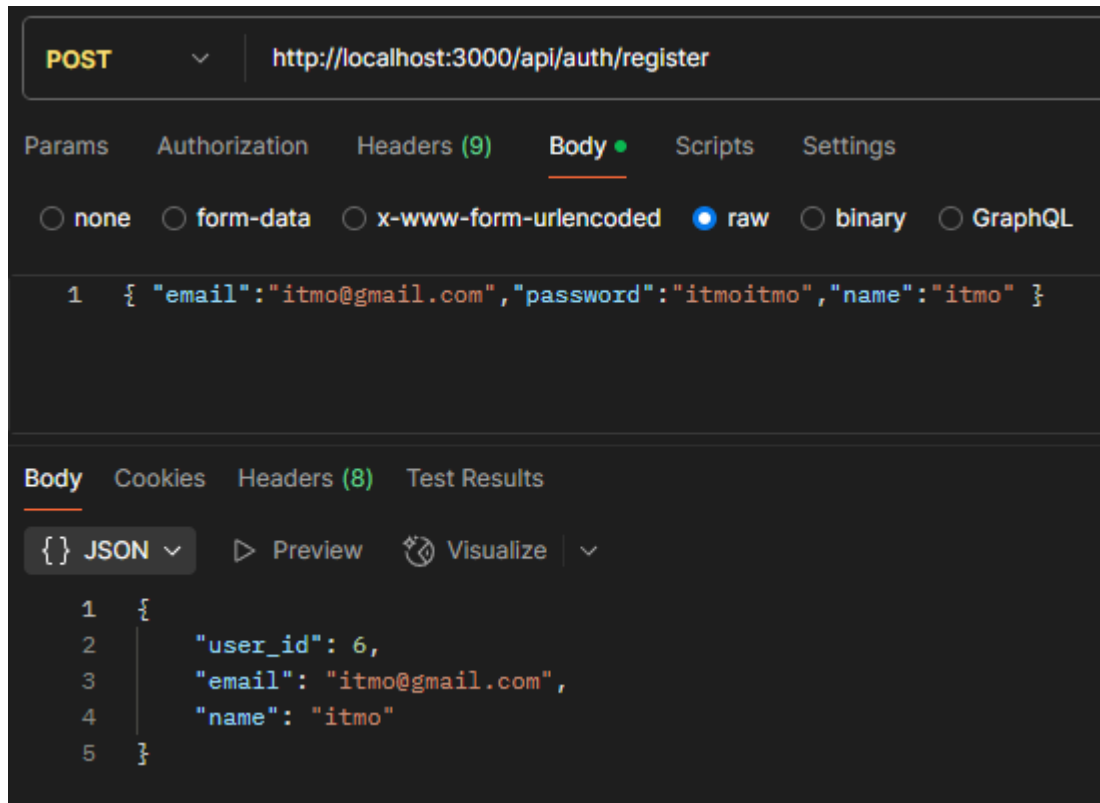
router.post('/register', AuthController.register)

router.post('/login', AuthController.login)
```

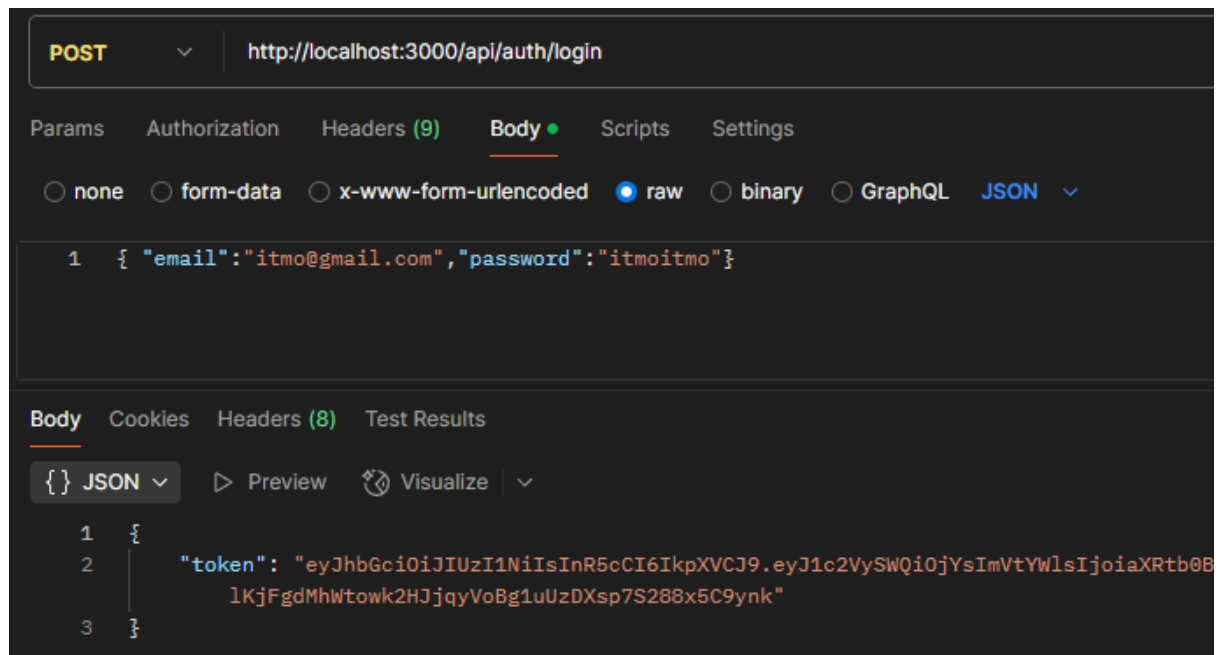
```
export default router;
```

Тестирование с помощью Postman

Регистрируем нового пользователя



Логинимся и получаем информацию о нашем токене



Теперь выведем информацию о всех юзерах

GET http://localhost:3000/api/users/

Params Authorization Headers (11) Body Scripts Settings

Query Params

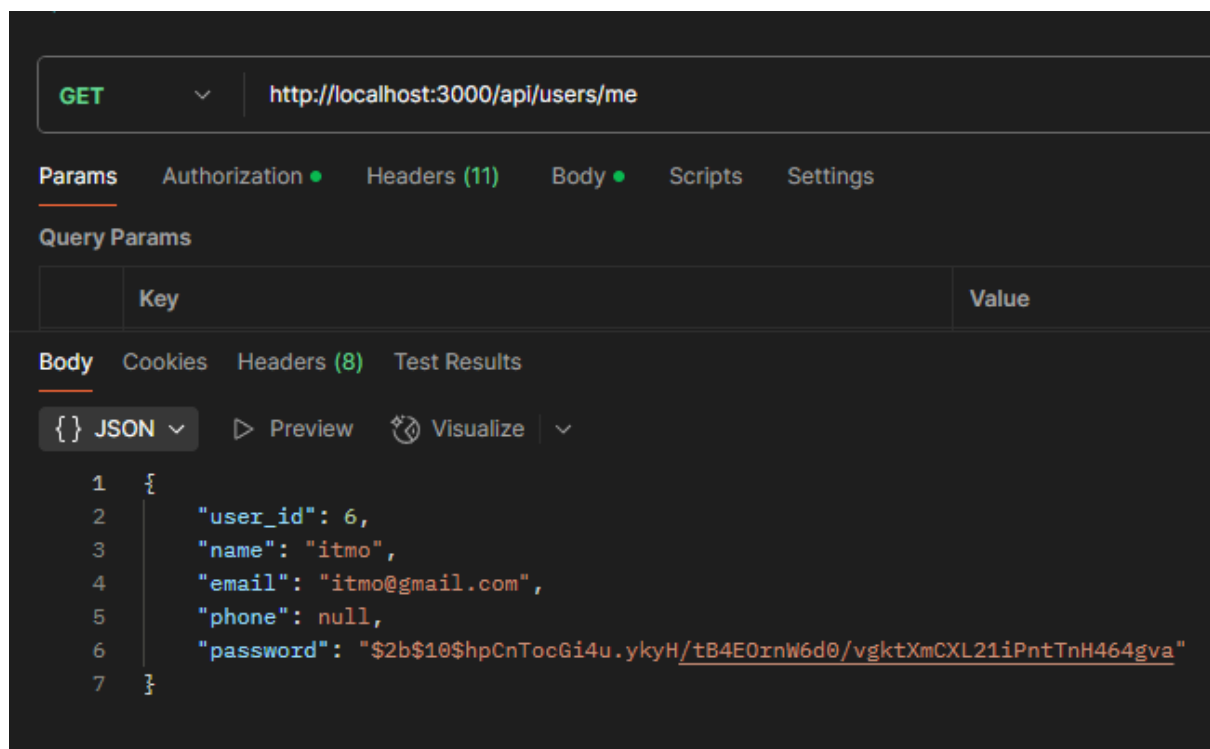
Key	Value
-----	-------

Body Cookies Headers (8) Test Results

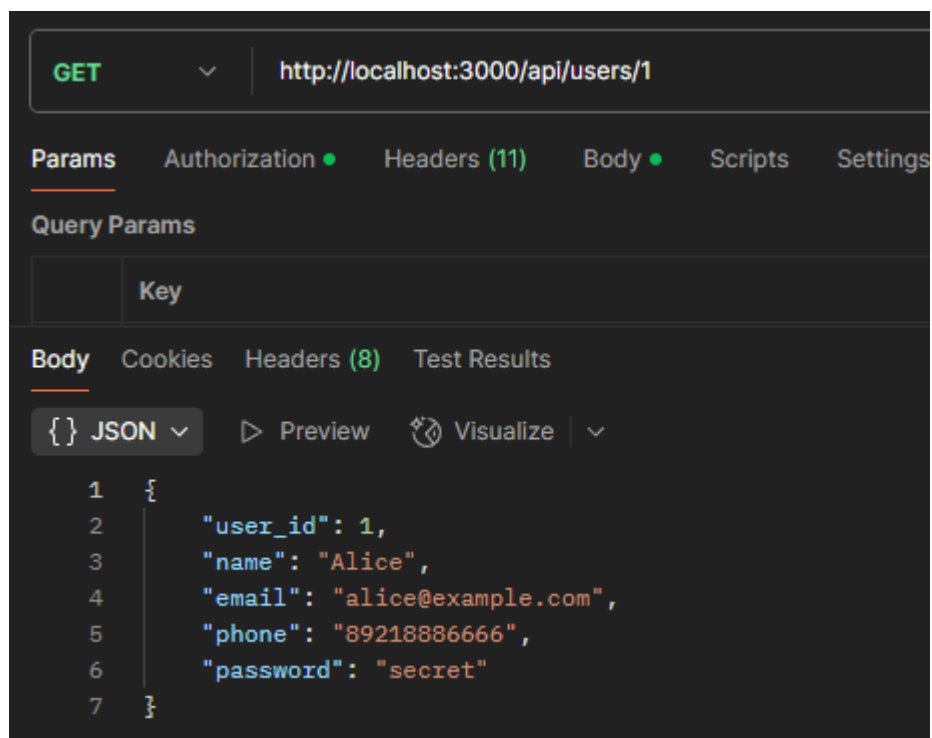
{ } JSON Preview Visualize

```
15 },
16 {
17   "user_id": 3,
18   "name": "Alex",
19   "email": "alex@gmail.com",
20   "phone": "89217792394",
21   "password": "password123"
22 },
23 {
24   "user_id": 4,
25   "name": "vova",
26   "email": "vova@gmail.com",
27   "phone": null,
28   "password": "$2b$08$uZT.KWQr93mgX7ilnS3Xf.8cwdgiGakiiT4Jiyrf.wwwXaaKTZ.oK"
29 },
30 {
31   "user_id": 5,
32   "name": "admin",
33   "email": "admin@gmail.com",
34   "phone": null,
35   "password": "$2b$08$HMiCha84hNThrSdQ63ARS.xeIKYr58i4cV3o8iUn3T/g.guEV1JHu"
36 },
37 {
38   "user_id": 6,
39   "name": "itmo",
40   "email": "itmo@gmail.com",
41   "phone": null,
42   "password": "$2b$10$hpCnTocGi4u.ykyH/tB4E0rnW6d0/vgktXmCXL21iPntTnH464gva"
43 }
44 ]
```

Выведем информацию о себе



Узнаем информацию о первом юзере



Теперь получим пользователя по email

GET ▼ <http://localhost:3000/api/users/by-email?email=admin@gmail.com>

Params ● Authorization ● Headers (11) Body ● Scripts Settings

Query Params

<input checked="" type="checkbox"/> Key	Value
---	-------

Body Cookies Headers (8) Test Results

{ } JSON ▼ ▶ Preview ↺ Visualize ▼

```
1 {
2   "user_id": 5,
3   "name": "admin",
4   "email": "admin@gmail.com",
5   "phone": null,
6   "password": "$2b$08$HMiCha84hNThrSdQ63ARS.xeIKYr58i4cV3o8iUn3T/g.guEV1JHu"
7 }
```

Выведем информацию о всех бронях

GET http://localhost:3000/api/bookings

Params Authorization Headers (11) Body Scripts Settings

Query Params

Key	Value
-----	-------

Body Cookies Headers (8) Test Results

{ } JSON Preview Visualize

```
1  [  
2    {  
3      "booking_id": 2,  
4      "start_at": "2025-05-01T00:00:00.000Z",  
5      "end_at": "2025-05-07T00:00:00.000Z",  
6      "total_price": "560",  
7      "deal_status": "pending",  
8      "created_at": "2025-04-21T18:27:28.337Z",  
9      "property": {  
10         "property_id": 1,  
11         "type": "apartment",  
12         "title": "Cozy Loft",  
13         "description": "Bright loft in downtown",  
14         "location": "City Center",  
15         "price_per_day": "80.00",  
16         "listed_at": "2025-04-21T18:24:44.079Z",  
17         "status": "active"  
18       },  
19       "renter": {  
20         "user_id": 2,  
21         "name": "Denis",  
22         "email": "den@ya.ru",  
23         "phone": "89216823984",  
24         "password": "denis12345"  
25       }  
26     },  
27     {  
28       "booking_id": 3,  
29       "start_at": "2025-05-01T00:00:00.000Z",  
30       "end_at": "2025-05-07T00:00:00.000Z",
```

Вывод

Получилось написать RESTful API на Express + TypeScript с JWT-аутентификацией, валидацией DTO. Весь код структурирован по слоям: модели, сервисы, контроллеры, роутеры.