

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа №2

Выполнил:

Шалунов Андрей

Группа К3340

Проверил:

Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

Реализовать все модели данных, спроектированные в рамках ДЗ1

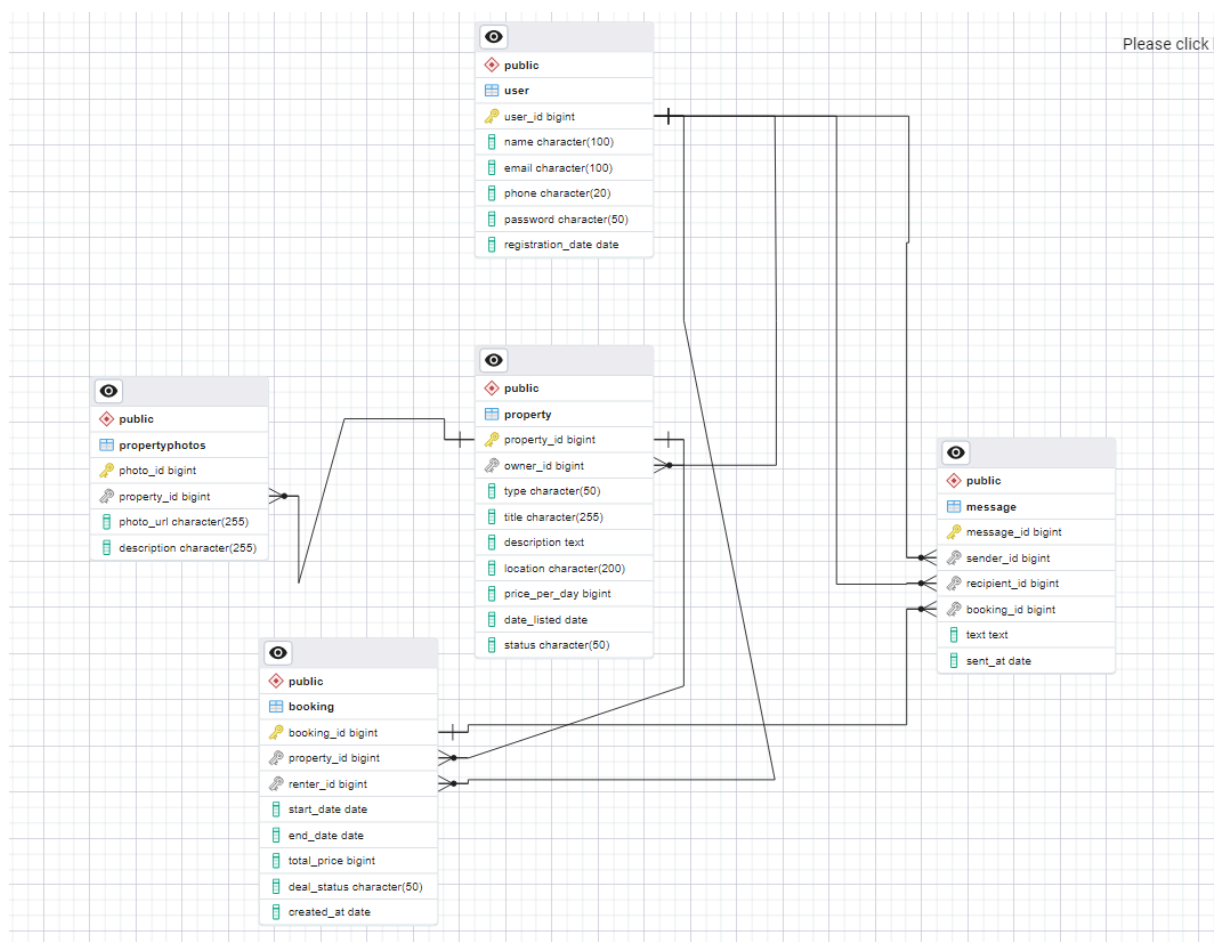
Реализовать набор из CRUD-методов для работы с моделями данных средствами Express + TypeScript

Реализовать API-эндпоинт для получения пользователя по id/email

Ход работы

В прошлом домашнем задании был взят вариант №3.

Основные таблицы:



Было создано 5 сущностей

User

```

БР1.1 > Шалунов Андрей > homeworks > hw2 > rental-service > src > models > TS Users.ts > User > phone
1  import { Entity, PrimaryGeneratedColumn, Column, OneToMany, CreateDateColumn } from 'typeorm';
2  import { Property } from './Property';
3  import { Booking } from './Booking';
4  import { Message } from './Message';
5
6  @Entity()
7  export class User {
8      @PrimaryGeneratedColumn()
9      user_id!: number;
10
11     @Column()
12     name!: string;
13
14     @Column({ unique: true })
15     email!: string;
16
17     @Column({ nullable: true })
18     phone?: string;
19
20     @Column()
21     password!: string;
22
23     @OneToMany(() => Property, p => p.owner)
24     properties!: Property[];
25
26     @OneToMany(() => Booking, b => b.renter)
27     bookings!: Booking[];
28
29     @OneToMany(() => Message, m => m.sender)
30     sentMessages!: Message[];
31
32     @OneToMany(() => Message, m => m.recipient)
33     receivedMessages!: Message[];
34 }

```

Property

```

БР1.1 > Шалунов Андрей > homeworks > hw2 > rental-service > src > models > TS Property.ts > Property
1  import { Entity, PrimaryGeneratedColumn, Column,ManyToOne, OneToMany, CreateDateColumn } from 'typeorm';
2  import { User } from './User';
3  import { PropertyPhoto } from './PropertyPhoto';
4  import { Booking } from './Booking';
5
6  @Entity()
7  export class Property {
8      @PrimaryGeneratedColumn()
9      property_id!:number;
10
11      @ManyToOne(() => User, u => u.properties)
12      owner!: User;
13
14      @Column()
15      type!:string;
16
17      @Column()
18      title!:string;
19
20      @Column('text')
21      description!:string;
22
23      @Column()
24      location!:string;
25
26      @Column('decimal')
27      price_per_day!:string;
28
29      @CreateDateColumn()
30      listed_at!:Date;
31
32      @Column()
33      status!:string;
34
35      @OneToMany(() => PropertyPhoto, p => p.property)
36      photos!: PropertyPhoto[];
37
38      @OneToMany(() => Booking, b => b.property)
39      bookings!: Booking[];
40  }

```

Booking

```

БР1.1 > Шалунов Андрей > homeworks > hw2 > rental-service > src > models > TS Booking.ts > Booking > deal_status
1  import { Entity, PrimaryGeneratedColumn, Column, ManyToOne, CreateDateColumn } from 'typeorm';
2  import { Property } from './Property';
3  import { User } from './User';
4
5  @Entity()
6  export class Booking {
7      @PrimaryGeneratedColumn()
8      booking_id!: number;
9
10     @ManyToOne(() => Property, p => p.bookings)
11     property!: Property;
12
13     @ManyToOne(() => User, u => u.bookings)
14     renter!: User;
15
16     @Column()
17     start_at!: Date;
18
19     @Column()
20     end_at!: Date;
21
22     @Column('decimal')
23     total_price!: number;
24
25     @Column()
26     deal_status!: string;
27
28     @CreateDateColumn()
29     created_at!: Date;
30 }

```

Message

```

БР1.1 > Шалунов Андрей > homeworks > hw2 > rental-service > src > models > TS Message.ts > Message > sent_at
1  import { Entity, PrimaryGeneratedColumn, Column, ManyToOne, CreateDateColumn } from 'typeorm';
2  import { User } from './User';
3  import { Booking } from './Booking';
4
5  @Entity()
6  export class Message {
7      @PrimaryGeneratedColumn()
8      message_id!: number;
9
10     @ManyToOne(() => User, u => u.sentMessages)
11     sender!: User;
12
13     @ManyToOne(() => User, u => u.receivedMessages)
14     recipient!: User;
15
16     @ManyToOne(() => Booking, b => b.property)
17     booking!: Booking;
18
19     @Column('text')
20     text!: string;
21
22     @CreateDateColumn()
23     sent_at!: Date;
24 }

```

PropertyPhoto

```

БП1.1 > Шалунов Андрей > homeworks > hw2 > rental-service > src > models > TS PropertyPhoto.ts > PropertyPhoto
1  import { Entity, PrimaryGeneratedColumn, Column, ManyToOne } from 'typeorm';
2  import { Property } from '../Property';
3
4  @Entity()
5  export class PropertyPhoto {
6      @PrimaryGeneratedColumn()
7      photo_id!: number;
8
9      @ManyToOne(() => Property, p => p.photos)
10     property!: Property;
11
12     @Column()
13     photo_url!: string;
14
15     @Column({nullable: true})
16     description?: string;
17 }

```

Реализация CRUD-сервисов и контроллеров

Для каждой сущности был реализован сервис:

- *create<Model>()*
- *getAll<Models>()*
- *get<Model>ById()*
- *update<Model>()*
- *delete<Model>()*

Контроллеры обрабатывают HTTP-запросы и передают данные в сервисы.

Маршруты

Для каждой сущности добавлены маршруты:

- POST /<entity>
- GET /<entity>
- GET /<entity>/:id
- PUT /<entity>/:id
- DELETE /<entity>/:id

Дополнительно:

GET /users/by-email?email=... — получение пользователя по email.

Полный код для роутера Юзера:

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

```
import * as controller from '../controllers/UserController';

const router = Router();

router.post('/', controller.create);

router.get('/', controller.findAll);

router.get('/by-email', controller.findByEmail);

router.get('/:id', controller.findOne);

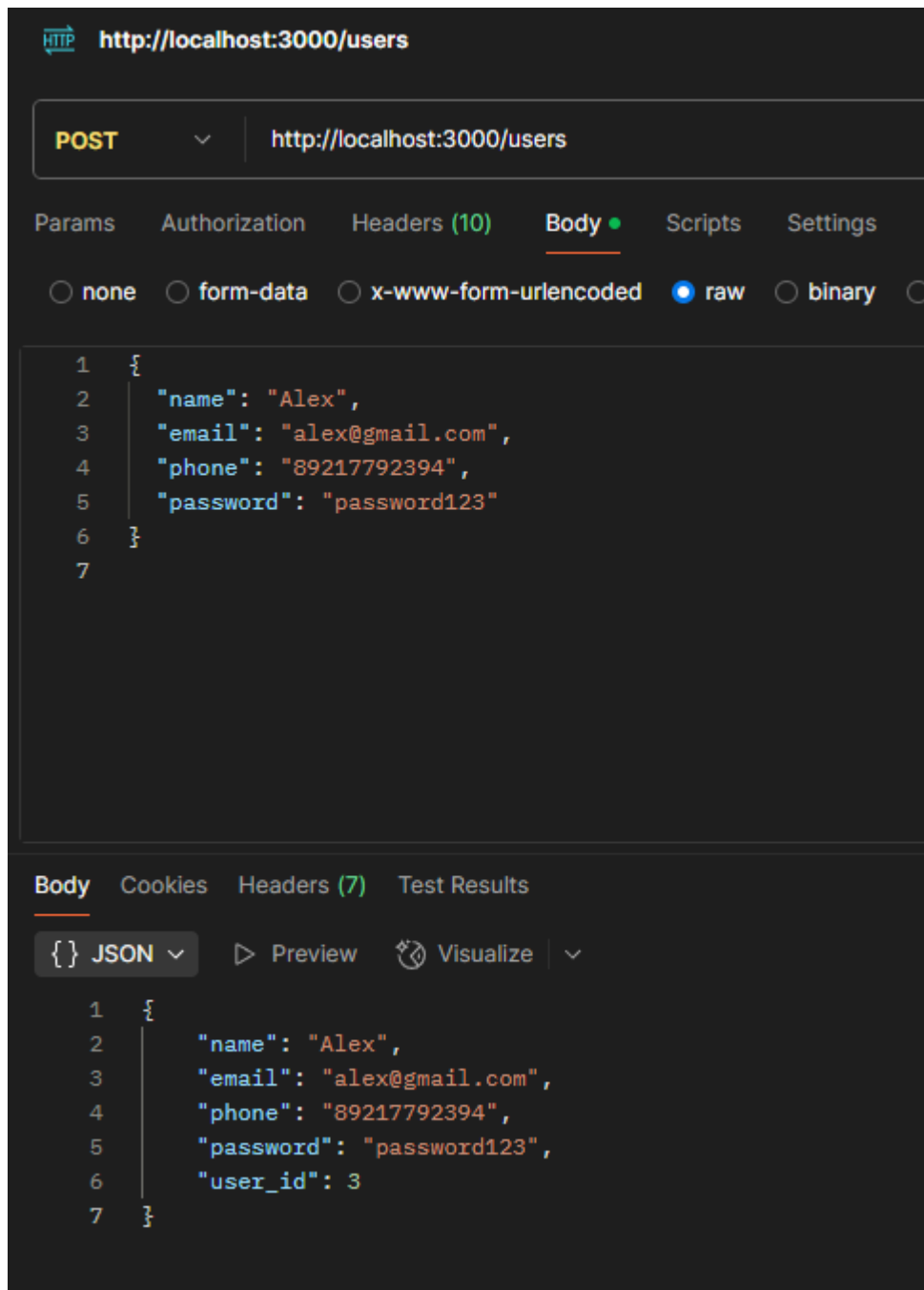
router.put('/:id', controller.update);

router.delete('/:id', controller.remove);

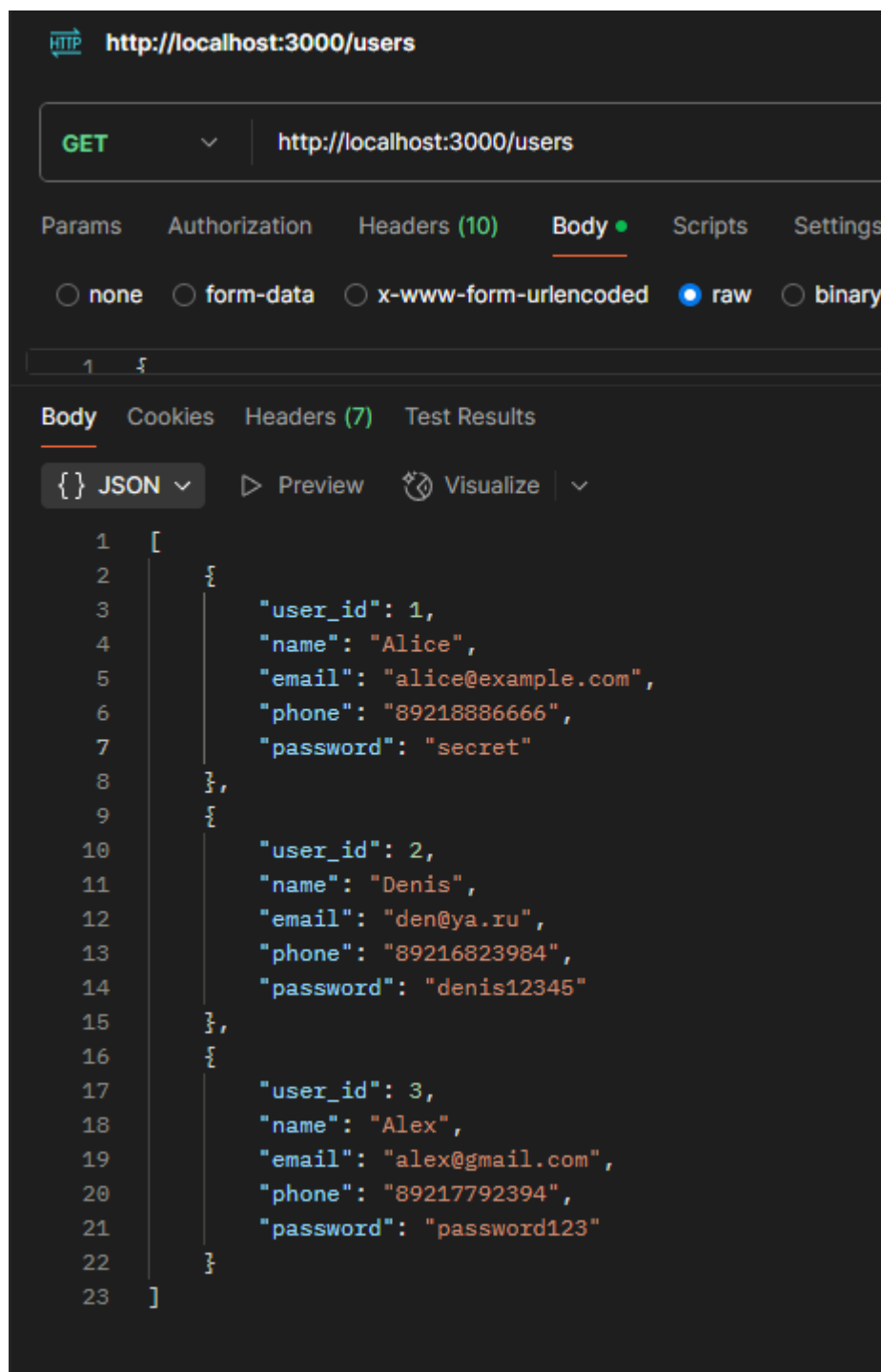
export default router;
```

Тестирование с помощью Postman

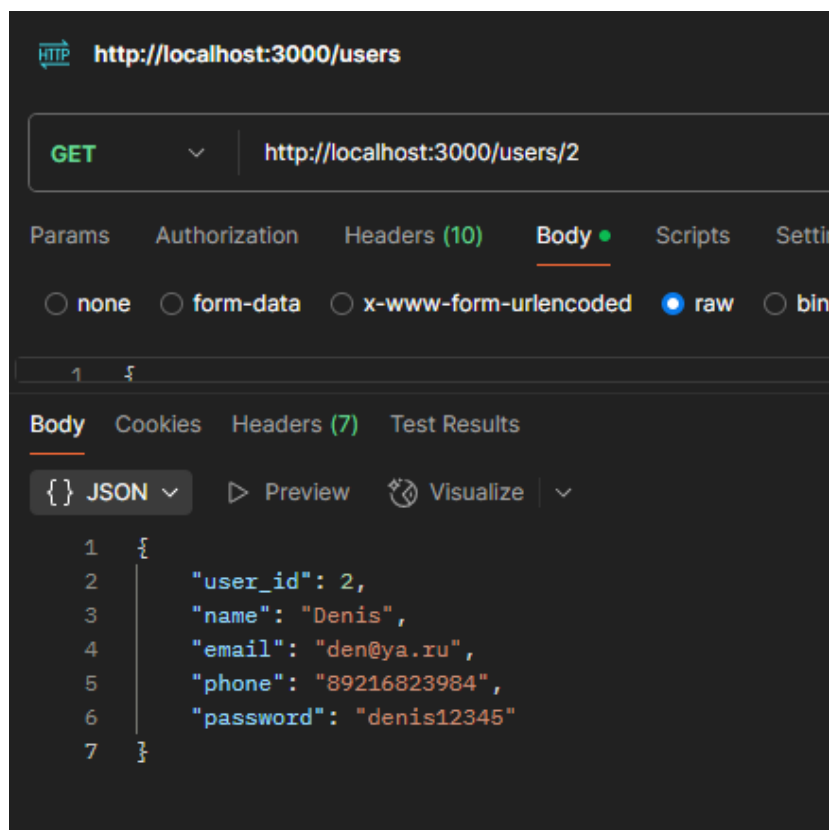
Создадим нового пользователя



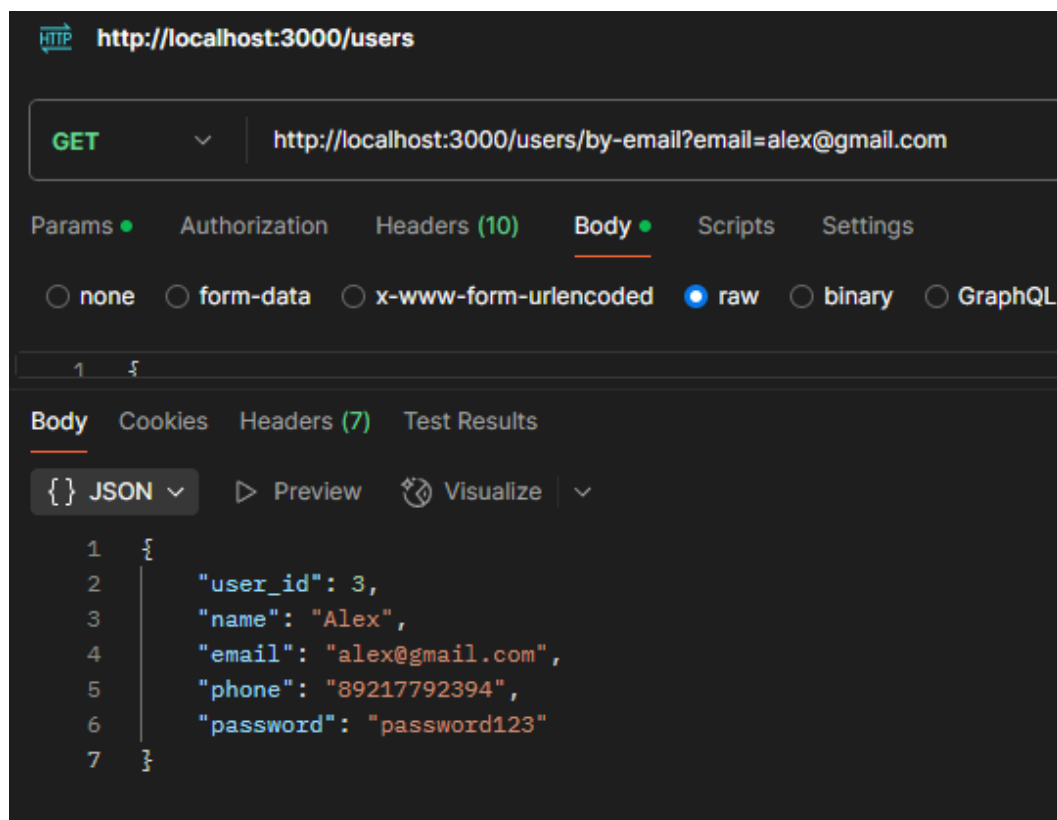
Теперь выведем информацию о всех юзерах (до этого уже создавалось двое)



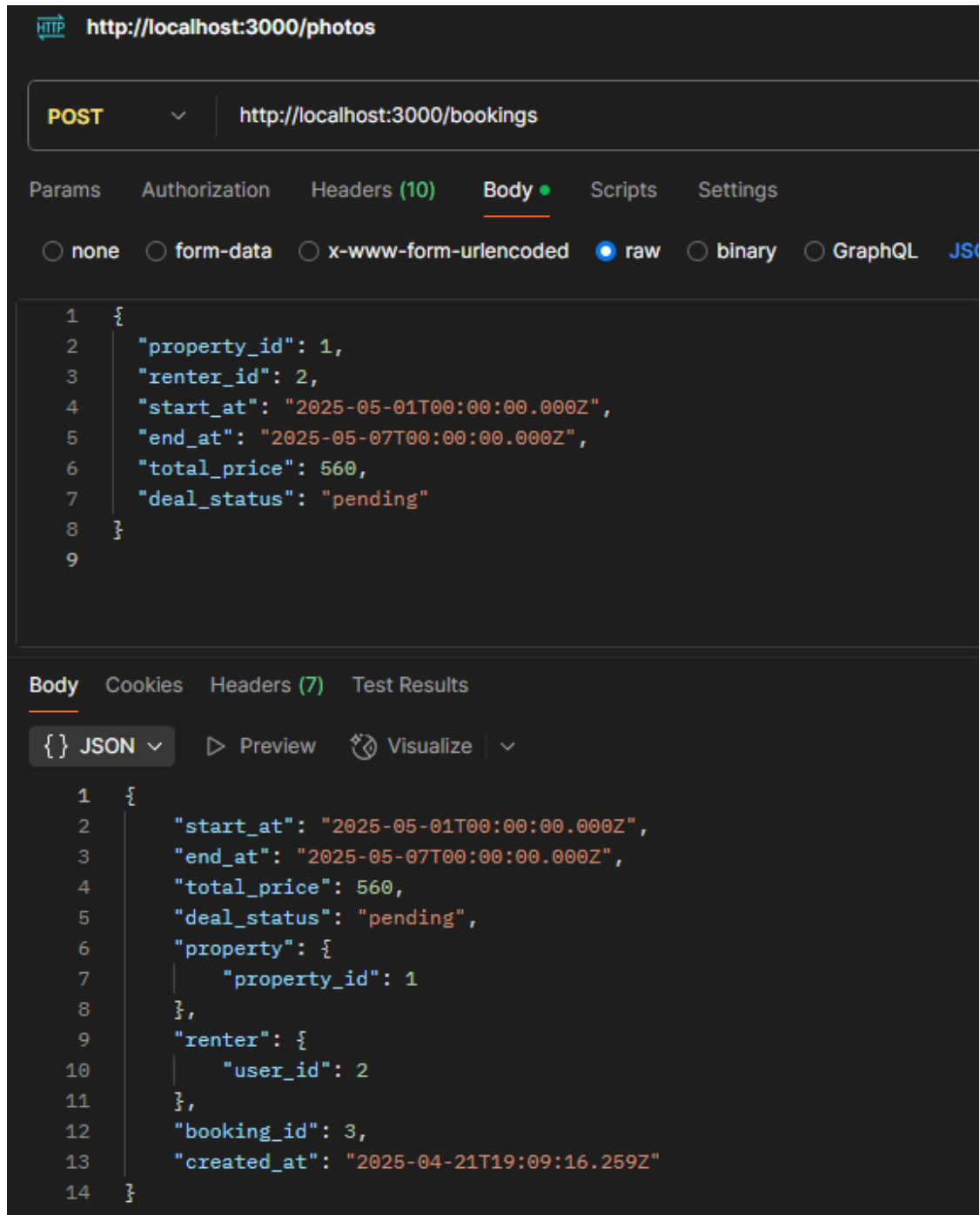
Получим информацию о пользователе с $id=2$



Теперь получим пользователя по email



Создадим бронь



The screenshot shows a REST client interface with a POST request to `http://localhost:3000/bookings`. The request body is raw JSON, and the response body is also JSON, showing the full booking record with nested property and renter information.

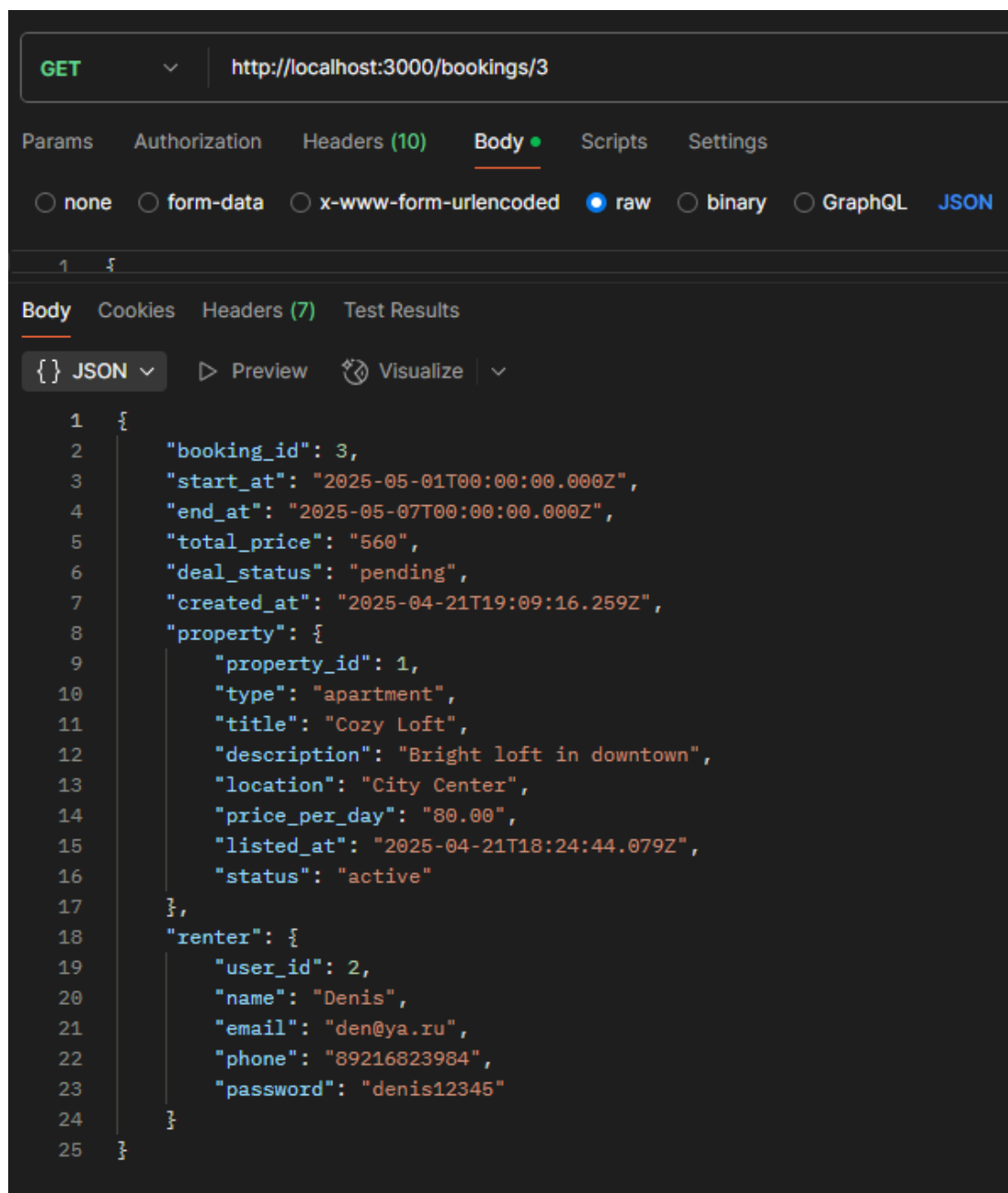
Request Body:

```
1  {
2    "property_id": 1,
3    "renter_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  }
```

Response Body:

```
1  {
2    "start_at": "2025-05-01T00:00:00.000Z",
3    "end_at": "2025-05-07T00:00:00.000Z",
4    "total_price": 560,
5    "deal_status": "pending",
6    "property": {
7      "property_id": 1
8    },
9    "renter": {
10     "user_id": 2
11   },
12   "booking_id": 3,
13   "created_at": "2025-04-21T19:09:16.259Z"
14 }
```

Теперь посмотрим полную информацию об этом бронировании



Вывод

Получилось спроектировать и создать модели с нужными связями, написать CRUD методы в сервисах и контроллерах и проверить работу API через Postman.