

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа 2
Создание RESTful API

Выполнила:

Хорошкеева Ксения

Группа К33402

Проверил:

Добряков Д. И.

Санкт-Петербург

2022 г.

Задача

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

Вариант 3: Платформа для поиска и бронирования номера в отеле/квартире/хостеле (<https://airbnb.com>)

- Вход
- Регистрация
- Страница бронирований пользователя
- Страница для поиска номера с возможностью выбора города, времени заселения, количеству гостей

Ход работы

Модель отеля:

```
1  import ...
3
4  @DefaultScope( scopeGetter: () => ({
5    attributes: ['id', 'name', 'city', 'stars']
6  }))
7  @Scopes( scopesGetter: () => ({
8    full: {
9      attributes: ['id', 'name', 'city', 'stars'],
10     include: [{
11       model: Room,
12       attributes: ['id', 'name', 'capacity', 'price']
13     }],
14     nest: true
15   }
16 })
```

```

17  @Table
18  class Hotel extends Model {
19      @AllowNull( allowNull: false)
20      @Column
21      name: string;
22
23      @AllowNull( allowNull: false)
24      @Column
25      city: string;
26
27      @AllowNull( allowNull: false)
28      @Column
29      stars: number;
30
31      @HasMany( associatedClassGetter: () => Room)
32      rooms: Room[];
33  }
34
35  export default Hotel;

```

Модель номера:

```

1  import ...
2
3
4
5  @DefaultScope( scopeGetter: () => ({
6      attributes: ['id', 'name', 'capacity', 'price']
7  }))
8  @Scopes( scopesGetter: () => ({
9      full: {
10         attributes: ['id', 'name', 'capacity', 'price'],
11         include: [{
12             model: Hotel,
13             attributes: ['id', 'name', 'city', 'stars']
14         }],
15         nest: true
16     }
17  }))

```

```

18   @Table
19   class Room extends Model {
20       @AllowNull( allowNull: false)
21       @Column
22       name: string;
23
24       @AllowNull( allowNull: false)
25       @Column
26       capacity: number;
27
28       @AllowNull( allowNull: false)
29       @Column
30       price: number;
31
32       @ForeignKey( relatedClassGetter: () => Hotel)
33       hotelId: number;
34
35       @BelongsTo( associatedClassGetter: () => Hotel)
36       hotel: Hotel;
37   }
38
39   export default Room;

```

Модель бронирования:

```

1   import ...
16
17   @DefaultScope( scopeGetter: () => ({
18       ↑ attributes: ['id', 'startDate', 'endDate'],
19       ↑ include: [{
20       ↑     model: Room,
21       ↑     attributes: ['id', 'name', 'capacity', 'price'],
22       ↑     include: [{
23       ↑       model: Hotel,
24       ↑       attributes: ['id', 'name', 'city', 'stars']
25       ↑     }],
26       nest: true
27     }],
28     nest: true
29   })))

```

```

30   @Table
31   class Booking extends Model {
32     @IsDate
33     @AllowNull( allowNull: false)
34     @Column(DataTypes.DATEONLY)
35     startDate: any;
36
37     @IsDate
38     @AllowNull( allowNull: false)
39     @Column(DataTypes.DATEONLY)
40     endDate: any;
41
42     @ForeignKey( relatedClassGetter: () => User)
43     @Column
44     userId: number;
45
46     @ForeignKey( relatedClassGetter: () => Room)
47     @Column
48     roomId: number;
49
50     @BelongsTo( associatedClassGetter: () => Room)
51     room: Room;

```

```

52
53     @BeforeValidate
54     static validateDates(instance: Booking) {
55       // Проверка правильности дат
56       if (instance.startDate >= instance.endDate) {
57         throw new Error('Дата начала должна быть раньше даты конца');
58       }
59     }
60   }
61
62   export default Booking;

```

Seeder для заполнения базы данных отелями и номерами:

```

3 module.exports = {
4   async up (queryInterface, Sequelize) {
5     await queryInterface.bulkInsert( tableName: 'Hotels', records: [
6       {name: 'Hotel A', city: 'Paris', stars: 5, createdAt: new Date(), updatedAt: new Date()},
7       {name: 'Hotel B', city: 'London', stars: 4, createdAt: new Date(), updatedAt: new Date()},
8       {name: 'Hotel C', city: 'Paris', stars: 5, createdAt: new Date(), updatedAt: new Date()},
9       {name: 'Hotel D', city: 'London', stars: 2, createdAt: new Date(), updatedAt: new Date()},
10      {name: 'Hotel E', city: 'Paris', stars: 1, createdAt: new Date(), updatedAt: new Date()},
11      {name: 'Hotel F', city: 'Berlin', stars: 5, createdAt: new Date(), updatedAt: new Date()}
12    ]);
13    await queryInterface.bulkInsert( tableName: 'Rooms', records: [
14      {hotelId: 1, name: 'Super deluxe', capacity: 2, price: 10000, createdAt: new Date(), updatedAt: new Date()},
15      {hotelId: 1, name: 'Deluxe', capacity: 2, price: 5000, createdAt: new Date(), updatedAt: new Date()},
16      {hotelId: 2, name: 'Family room', capacity: 4, price: 3000, createdAt: new Date(), updatedAt: new Date()},
17      {hotelId: 2, name: 'Single room', capacity: 1, price: 1500, createdAt: new Date(), updatedAt: new Date()},
18      {hotelId: 3, name: 'Presidential', capacity: 2, price: 40000, createdAt: new Date(), updatedAt: new Date()},
19      {hotelId: 4, name: 'Economy', capacity: 2, price: 1000, createdAt: new Date(), updatedAt: new Date()},
20      {hotelId: 4, name: 'Bed', capacity: 1, price: 300, createdAt: new Date(), updatedAt: new Date()},
21      {hotelId: 5, name: 'Super economy', capacity: 2, price: 500, createdAt: new Date(), updatedAt: new Date()},
22      {hotelId: 6, name: 'Large room', capacity: 6, price: 7500, createdAt: new Date(), updatedAt: new Date()}
23    ]);
24  },

```

Поиск комнат по городу:

```

20   async searchByCity(city: string): Promise<Room[]> {
21     // Получение списка комнат по названию города
22     return await Room.scope( options: 'full').findAll( options: {
23       include: {
24         model: Hotel,
25         where: {
26           city: {[Op.substring]: city}
27         }
28       }
29     });
30   }

```

Поиск комнат по вместительности:

```

32   async searchByCapacity(capacity: number): Promise<Room[]> {
33     // Получение списка комнат по вместимости
34     return await Room.scope( options: 'full').findAll( options: {
35       where: {
36         capacity: {[Op.gte]: capacity}
37       }
38     });
39   }

```

Поиск комнат по городу и вместительности:

```

41   async searchByCityAndCapacity(city: string, capacity: number): Promise<Room[]> {
42       // Получение списка комнат по названию города и вместимости
43       return await Room.scope( options: 'full').findAll( options: {
44           where: {
45               capacity: {[Op.gte]: capacity}
46           },
47           include: {
48               model: Hotel,
49               where: {
50                   city: {[Op.substring]: city}
51               }
52           }
53       });
54   }

```

Контроллер для поиска:

```

21   list = async (request: any, response: any) => {
22       // Получение всех номеров с фильтрацией
23       const {city, capacity} = request.query;
24       console.log(city, capacity);
25       if (city && capacity) {
26           // Указан город и вместительность
27           response.send(await this.roomService.searchByCityAndCapacity(city, capacity));
28       } else if (city) {
29           // Указан город
30           response.send(await this.roomService.searchByCity(city));
31       } else if (capacity) {
32           // Указана вместительность
33           response.send(await this.roomService.searchByCapacity(capacity));
34       } else {
35           // Ничего не указано
36           response.send(await this.roomService.getAll());
37       }
38   }

```

Получение списка отелей:

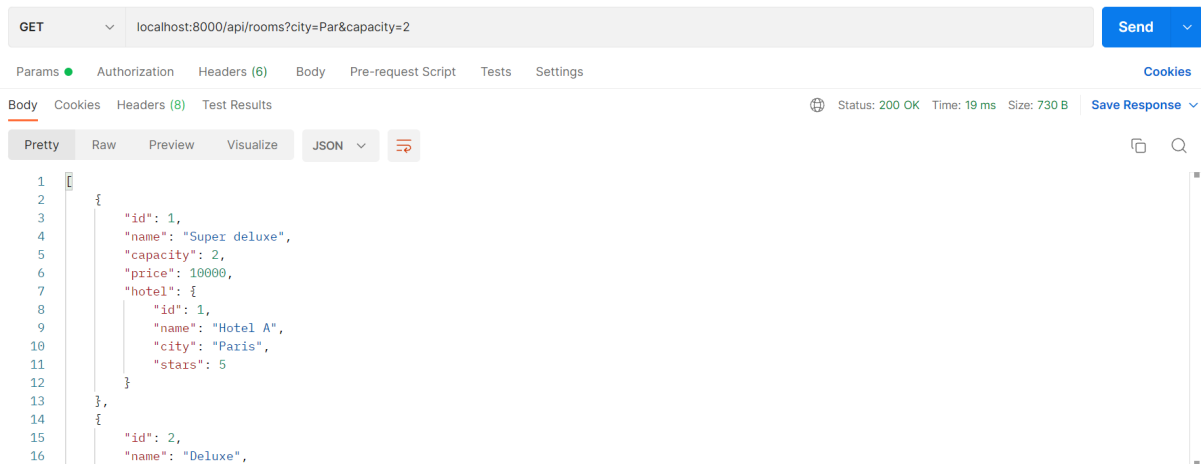
The screenshot shows a REST client interface with a GET request to `localhost:8000/api/hotels`. The response status is 200 OK, with a time of 46 ms and a size of 1.1 KB. The response body is displayed in JSON format, showing a list of hotels and their rooms.

```

1  {
2    "id": 1,
3    "name": "Hotel A",
4    "city": "Paris",
5    "stars": 5,
6    "rooms": [
7      {
8        "id": 2,
9        "name": "Deluxe",
10       "capacity": 2,
11       "price": 5000
12     },
13     {
14       "id": 1,
15       "name": "Super deluxe",

```

Получение списка номеров с поиском:



GET localhost:8000/api/rooms?city=Par&capacity=2

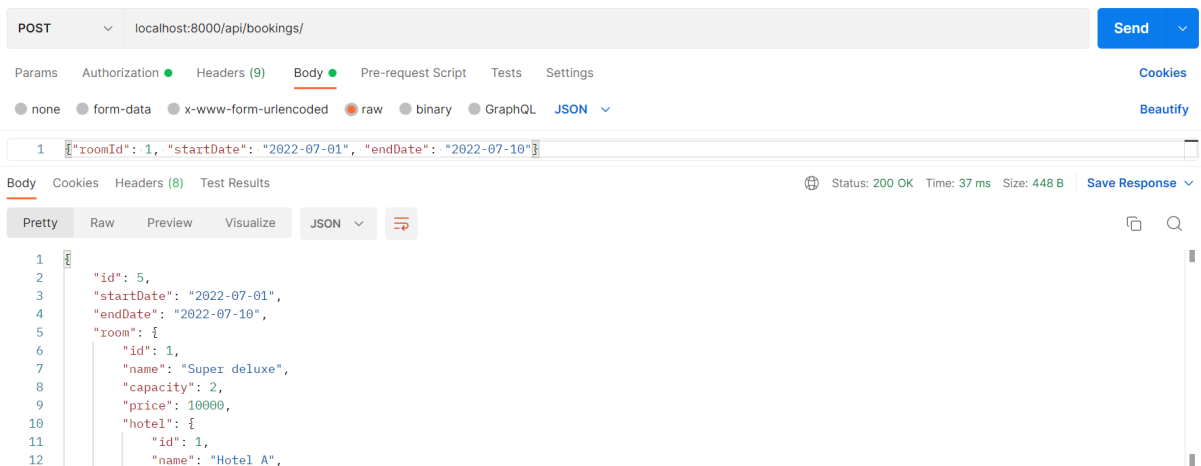
Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Body Cookies Headers (8) Test Results Status: 200 OK Time: 19 ms Size: 730 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "id": 1,
4     "name": "Super deluxe",
5     "capacity": 2,
6     "price": 10000,
7     "hotel": {
8       "id": 1,
9       "name": "Hotel A",
10      "city": "Paris",
11      "stars": 5
12    }
13  },
14  {
15    "id": 2,
16    "name": "Deluxe",
```

Создание бронирования:



POST localhost:8000/api/bookings/

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

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

1 {"roomId": 1, "startDate": "2022-07-01", "endDate": "2022-07-10"}

Body Cookies Headers (8) Test Results Status: 200 OK Time: 37 ms Size: 448 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 5,
3   "startDate": "2022-07-01",
4   "endDate": "2022-07-10",
5   "room": {
6     "id": 1,
7     "name": "Super deluxe",
8     "capacity": 2,
9     "price": 10000,
10    "hotel": {
11      "id": 1,
12      "name": "Hotel A",
```

Вывод

Я разработала RESTful API сервиса поиска и бронирования отелей с использованием фреймворка Express и ORM Sequelize