Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе № 2 «Разработка RESTful API средствами Express + TypeORM + TypeScript»

по дисциплине «Бэкенд-разработка»

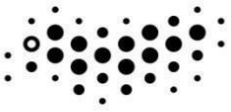
Автор: Власов М. И.

Факультет: ИКТ

Группа: K33402

Преподаватель: Добряков Д. И.

Дата: 22.04.22



УНИВЕРСИТЕТ ИТМО

Санкт-Петербург, 2022

Цель: реализовать RESTful API средствами Express + TypeORM + TypeScript для платформы для поиска и бронирования номера в отеле/квартире/хостеле.

Модели

```
@Entity({ name: "users" })
export class User {
    @PrimaryGeneratedColumn()
    id!: number

    @Column()
    @IsString()
    firstName!: string

    @Column()
    @IsString()
    lastName!: string

    @Column({ unique: true })
    @IsEmail()
    email!: string

    @Column()
    @IsString()
    password!: string

    @OneToMany(() => Booking, booking => booking.tenant)
    bookings!: Booking[]
```

```typescript
@Entity({ name: "property" })
export class Property {
    @PrimaryGeneratedColumn()
    id!: number

    @Column()
    @IsString()
    title!: string

    @Column()
    @IsString()
    description!: string

    @Column()
    @IsString()
    city!: string

    @Column()
    @IsString()
    address!: string

    @Column()
    @Min(1)
    @Max(100)
    guestLimit!: number

    @Column()
    @Min(1)
    @Max(100000)
    price!: number

    @OneToMany(() => Booking, booking => booking.property)
    bookings!: Booking[]
```

```typescript
@Entity({ name: "bookings" })
export class Booking {
    @PrimaryGeneratedColumn()
    id!: number

    @ManyToOne(() => User, user => user.bookings)
    @IsNotEmpty()
    tenant!: User

    @ManyToOne(() => Property, property => property.bookings)
    @IsNotEmpty()
    property!: Property

    @Column()
    @NotContains('error')
    checkin!: string

    @Column()
    @NotContains('error')
    checkout!: string
```

Сервисы на примере *Property*

```
class PropertyService {
    repository = dataSource.getRepository(Property);

    async create(propertyData: object) : Promise<Property> {
        try {
            const entity = Object.assign(new Property(), propertyData)
            const property = await this.repository.save(entity)
            return property
        } catch (e: any) {
            throw new Error(e)
        }
    }

    async get(id: number) : Promise<Property> {
        const property = await this.repository.findOneBy({ 'id': id })

        if (property) return property

        throw new Error(`Property with id ${id} not found`)
    }

    async update(id: number, propertyData: object) : Promise<UpdateResult> {
        try {
            const newEntity = Object.assign(new Property(), propertyData)
            const entity = await this.get(id)
            for (const field in entity) {
                if (newEntity.hasOwnProperty(field)) {
                    entity[field] = newEntity[field]
                }
            }
            return await this.repository.update({ 'id' : id }, entity)
        } catch (e: any) {
            throw new Error(e)
        }
    }
```

```
async delete(id: number) : Promise<DeleteResult> {
    try {
        return await this.repository.delete({ 'id' : id })
    } catch (e: any) {
        throw new Error(e)
    }
}

async getList() : Promise<Property[]> {
    return await this.repository.find()
}
```

Контроллеры на примере *Property*

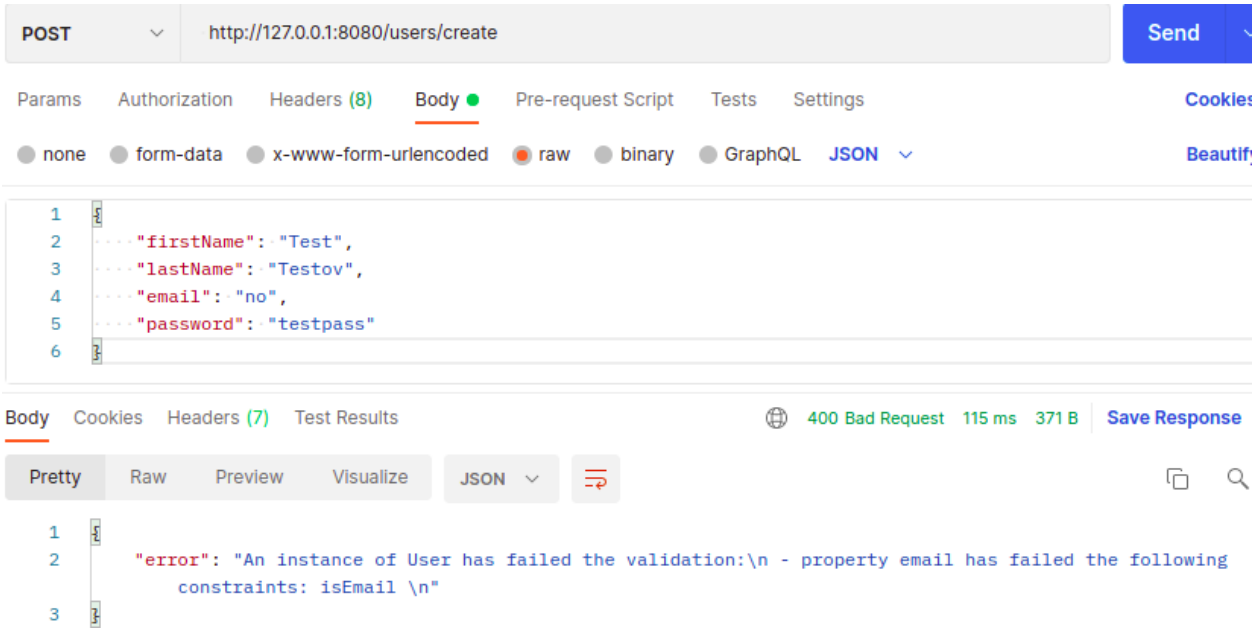
```
class PropertyController {
    private propertyService: PropertyService

    constructor() {
        this.propertyService = new PropertyService()
    }

    create = async (request: any, response: any) => {
        try {
            const { body } = request

            const res: Property | Error = await this.propertyService.create(body)

            response.status(201).send(res)
        } catch (error: any) {
            response.status(400).send({ "error": error.message })
        }
    }

    retrieve = async (request: any, response: any) => {
        try {
            const res: Property | Error = await this.propertyService.get(Number(request.params.id))

            response.send(res)
        } catch (error: any) {
            response.status(404).send({ "error": error.message })
        }
    }
```

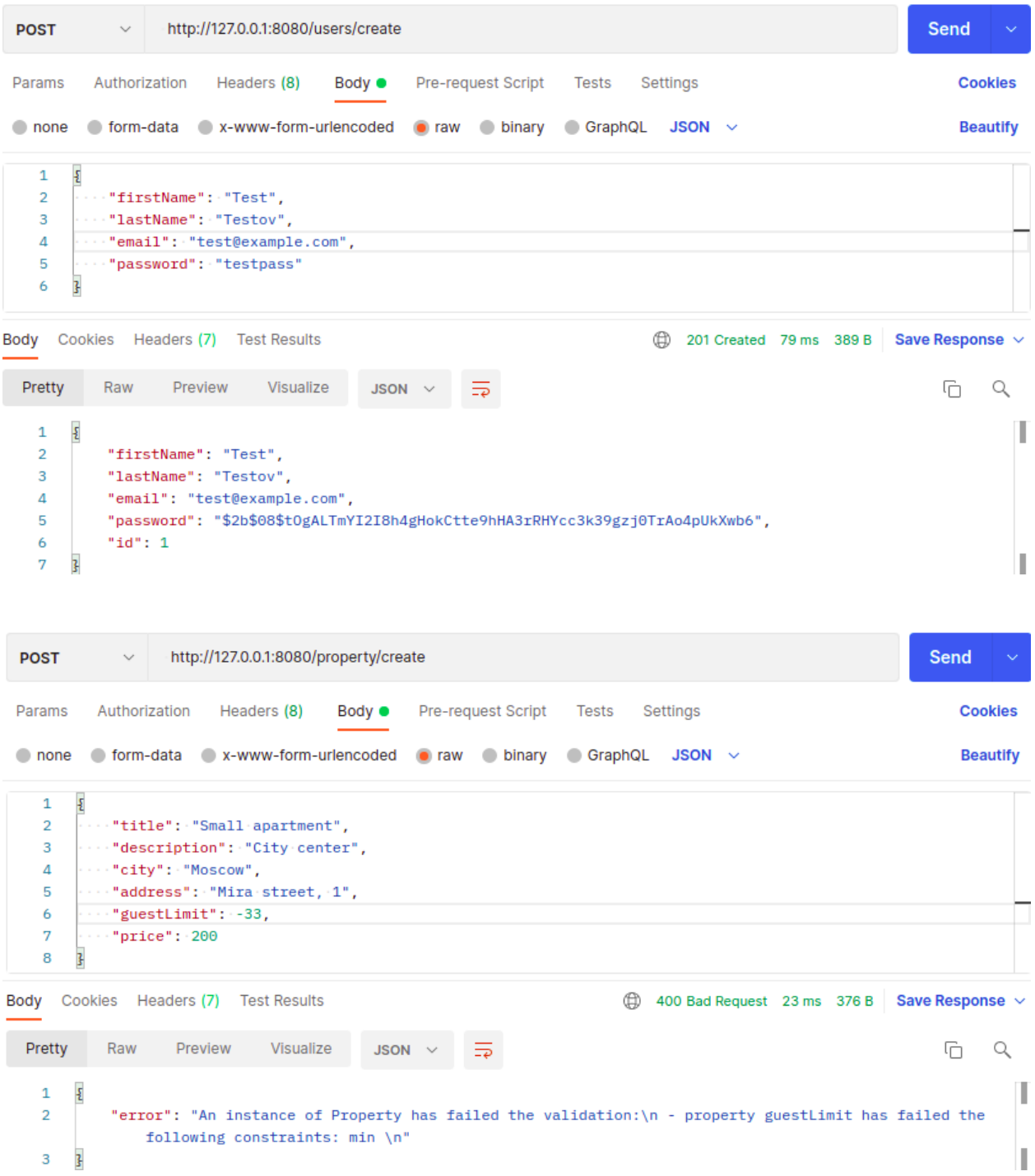
```
update = async (request: any, response: any) => {
    try {
        const { body } = request

        const res: UpdateResult | Error = await this.propertyService.update(Number(request.params.id), body)

        response.send(res)
    } catch (error: any) {
        response.status(404).send({ "error": error.message })
    }
}

delete = async (request: any, response: any) => {
    try {
        const res: DeleteResult | Error = await this.propertyService.delete(Number(request.params.id))

        response.send(res)
    } catch (error: any) {
        response.status(404).send({ "error": error.message })
    }
}

getList = async (request: any, response: any) => {
    try {
        const res: Property[] | Error = await this.propertyService.getList()

        response.send(res)
    } catch (error: any) {
        response.status(404).send({ "error": error.message })
    }
}
```

Routers на примере *Property*

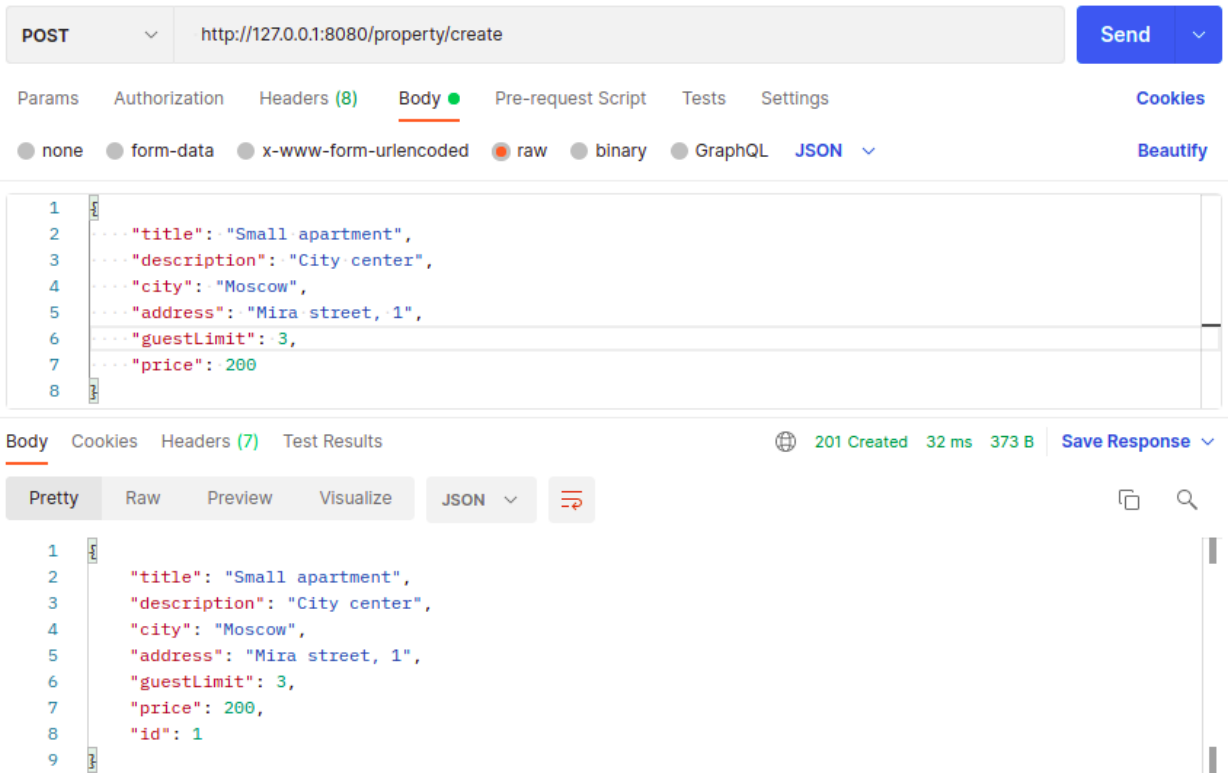
```
const router: express.Router = express.Router()

const controller: PropertyController = new PropertyController()

router.route("/create").post(controller.create)

router.route("/get/:id").get(controller.retrieve)

router.route("/update/:id").post(controller.update)

router.route("/delete/:id").post(controller.delete)

router.route("/list").get(controller.getList)
```

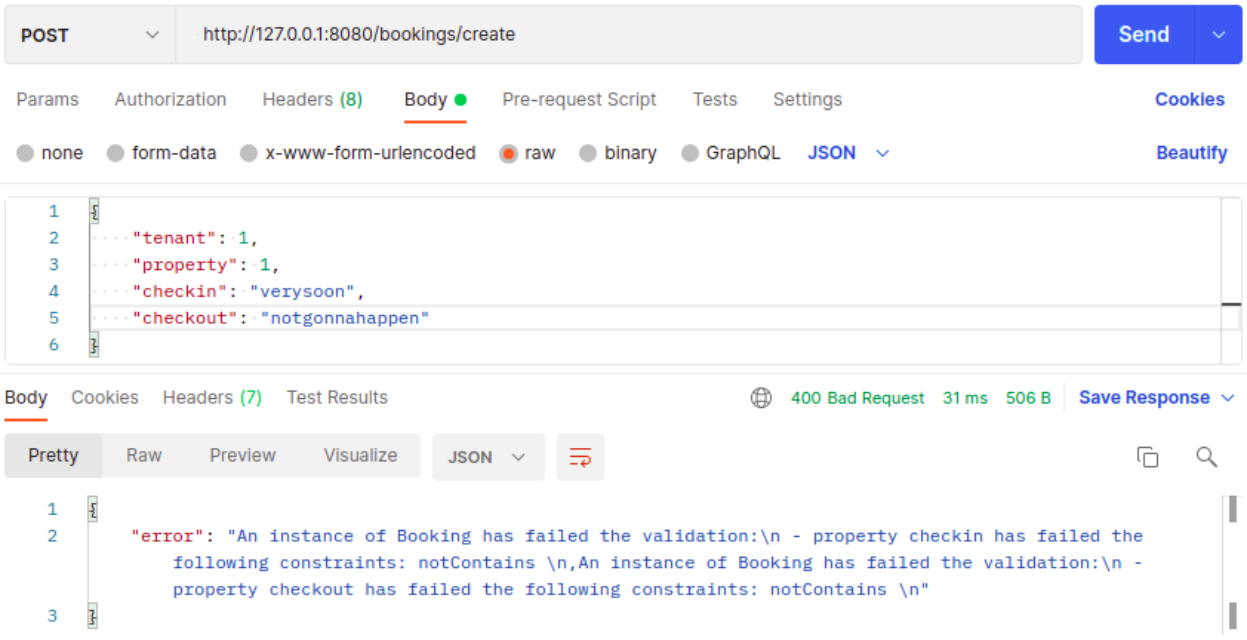
Примеры запросов

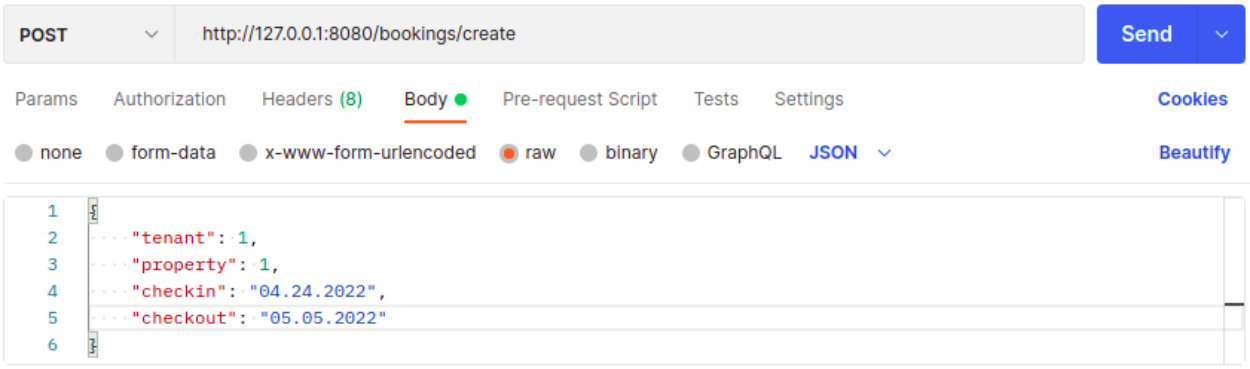


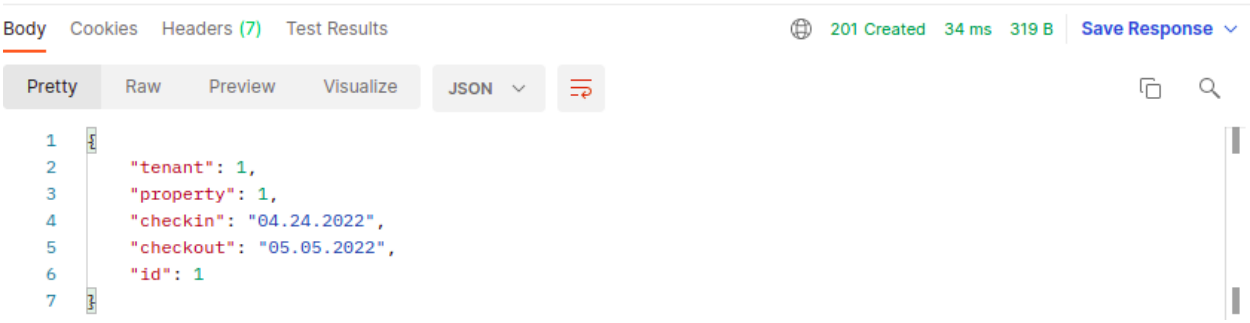

POST
http://127.0.0.1:8080/users/create
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"firstName": "Test",
"lastName": "Testov",
"email": "test@example.com",
"password": "testpass"
}
Body
Cookies
Headers (7)
Test Results
201 Created
79 ms
389 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"firstName": "Test",
"lastName": "Testov",
"email": "test@example.com",
"password": "$2b$08$tOgALTmYI2I8h4gHokCtte9hHA3rRHYcc3k39gzj0TrAo4pUkXwb6",
"id": 1
}

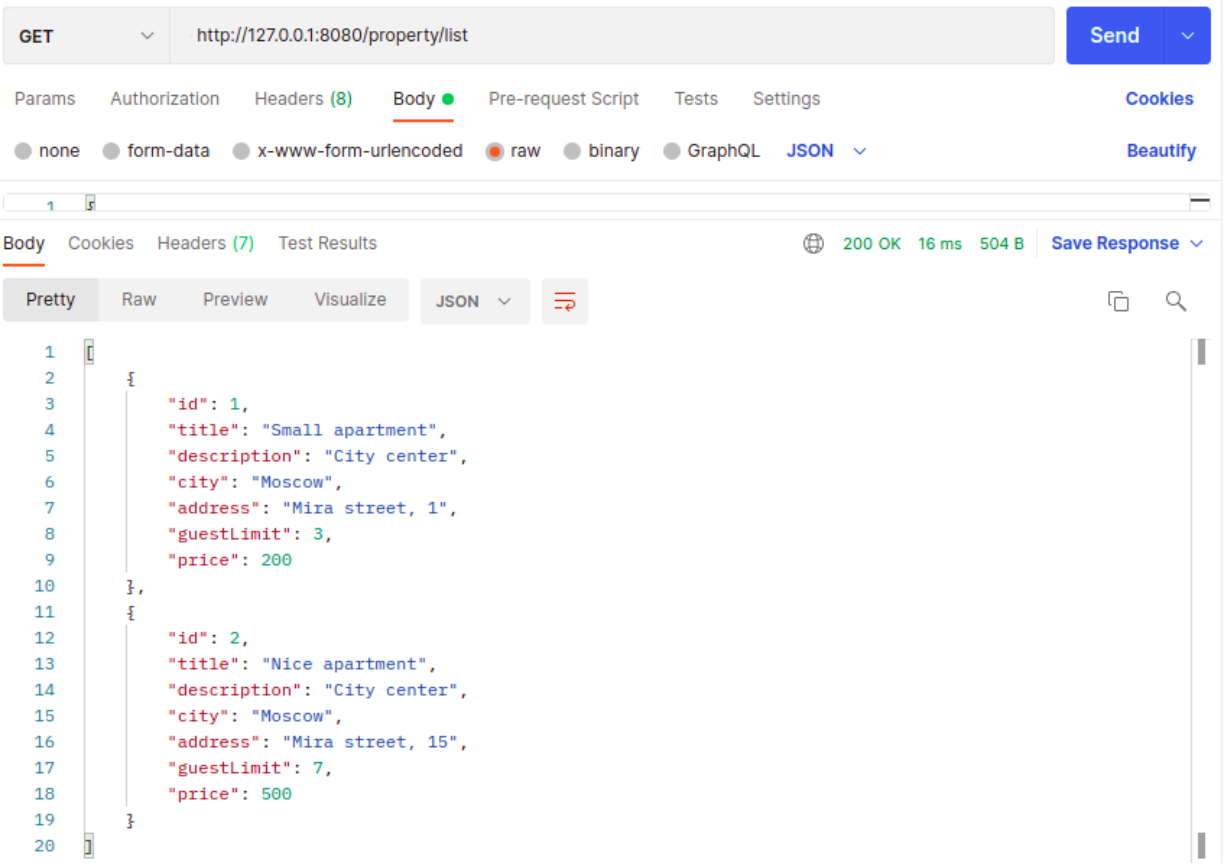
POST
http://127.0.0.1:8080/property/create
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"title": "Small apartment",
"description": "City center",
"city": "Moscow",
"address": "Mira street, 1",
"guestLimit": -33,
"price": 200
}
Body
Cookies
Headers (7)
Test Results
400 Bad Request
23 ms
376 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"error": "An instance of Property has failed the validation:\n - property guestLimit has failed the following constraints: min \n"
}

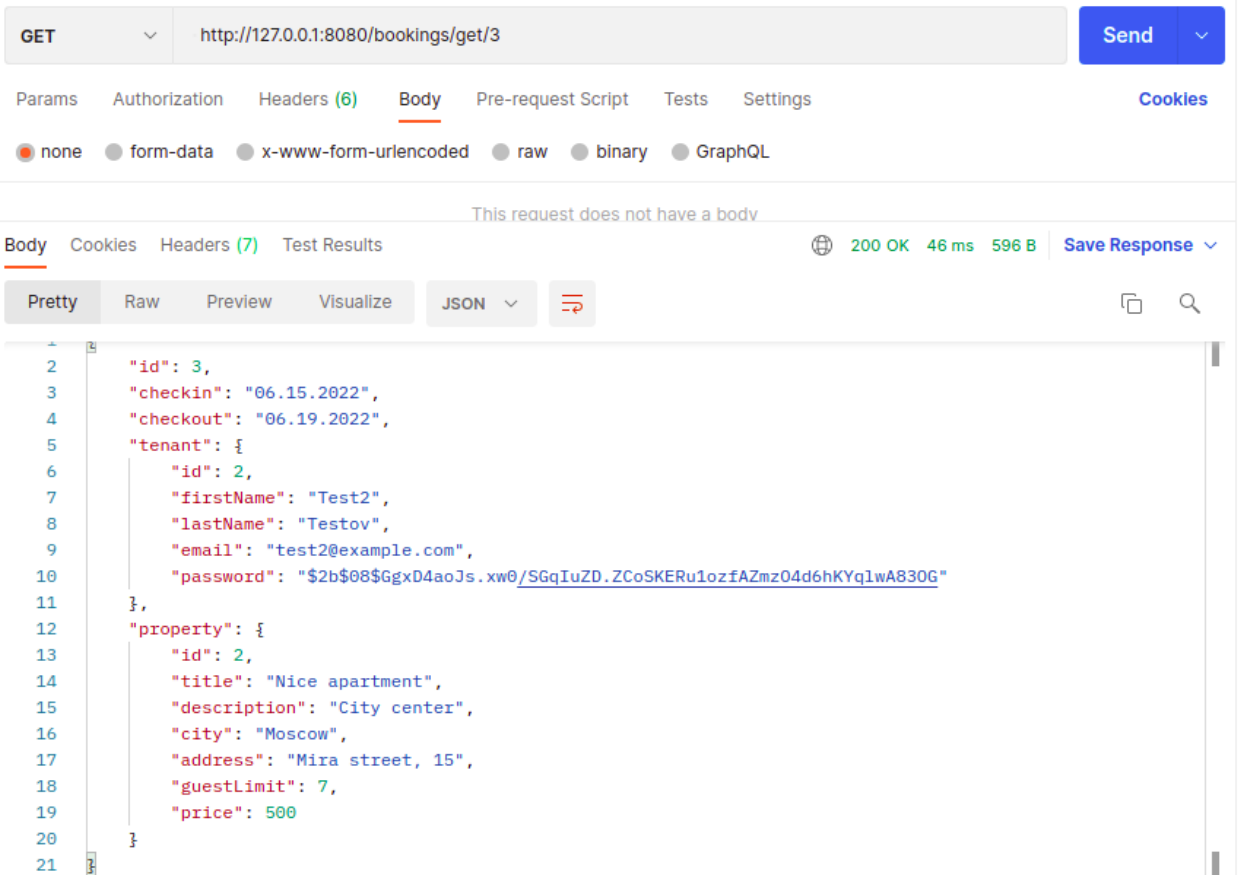

POST
http://127.0.0.1:8080/property/create
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"title": "Small apartment",
"description": "City center",
"city": "Moscow",
"address": "Mira street, 1",
"guestLimit": 3,
"price": 200
}
Body
Cookies
Headers (7)
Test Results
201 Created
32 ms
373 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"title": "Small apartment",
"description": "City center",
"city": "Moscow",
"address": "Mira street, 1",
"guestLimit": 3,
"price": 200,
"id": 1
}


POST
http://127.0.0.1:8080/bookings/create
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"tenant": 1,
"property": 1,
"checkin": "verysoon",
"checkout": "notgonnahappen"
}
Body
Cookies
Headers (7)
Test Results
400 Bad Request
31 ms
506 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"error": "An instance of Booking has failed the validation:\n - property checkin has failed the following constraints: notContains \n,An instance of Booking has failed the validation:\n - property checkout has failed the following constraints: notContains \n"
}

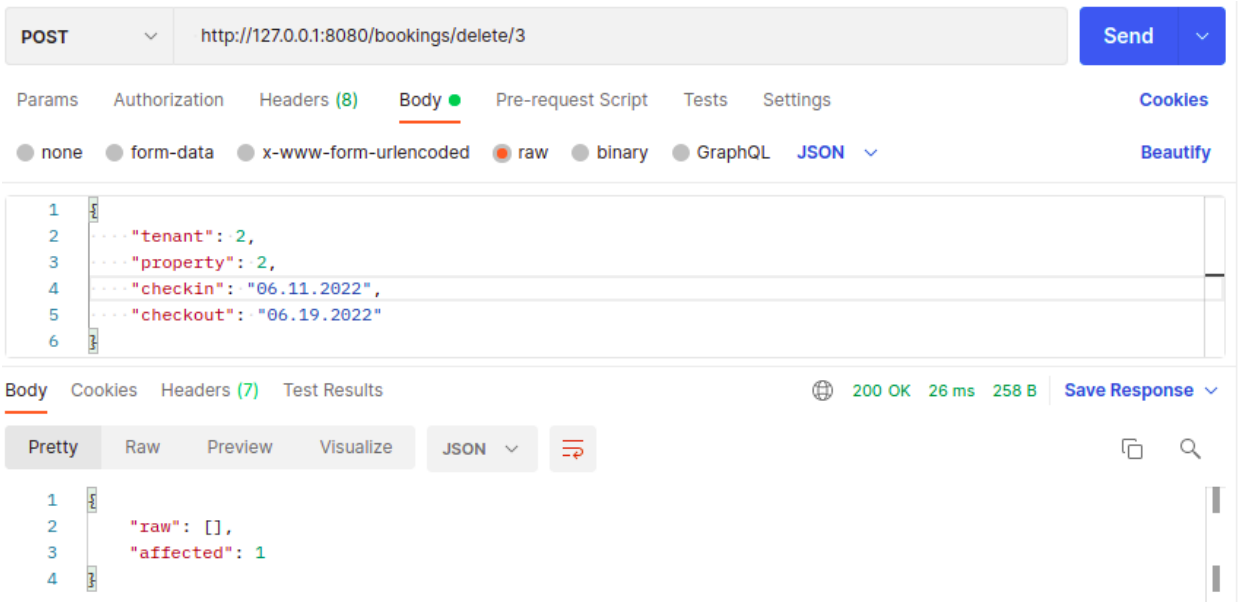

POST
http://127.0.0.1:8080/bookings/create
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"tenant": 1,
"property": 1,
"checkin": "04.24.2022",
"checkout": "05.05.2022"
}

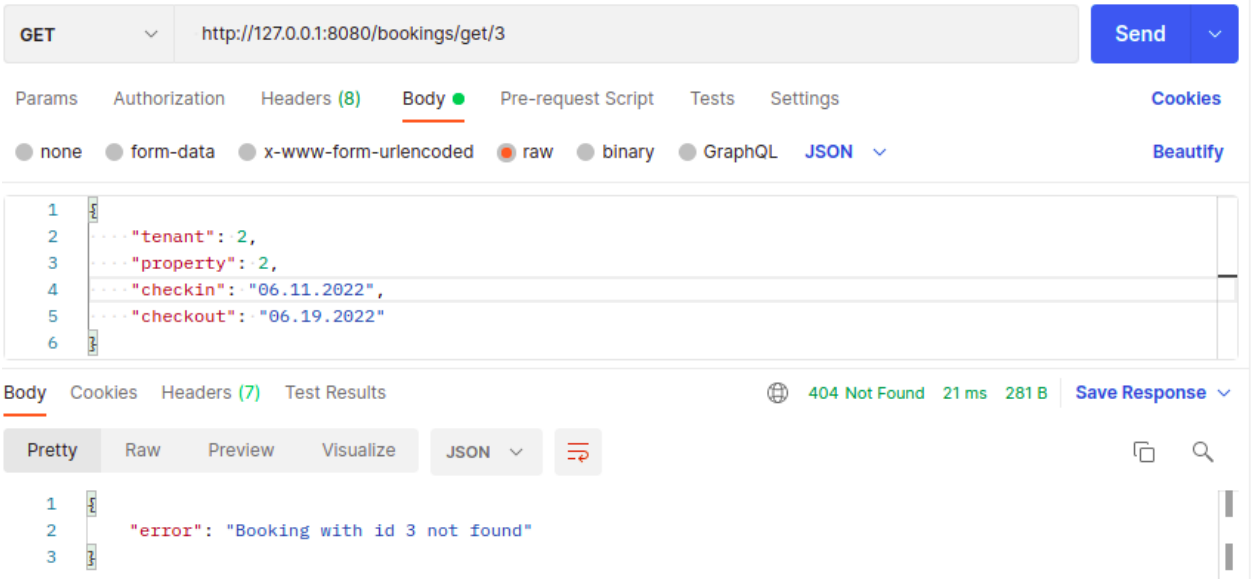

Body
Cookies
Headers (7)
Test Results
201 Created
34 ms
319 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"tenant": 1,
"property": 1,
"checkin": "04.24.2022",
"checkout": "05.05.2022",
"id": 1
}


GET
http://127.0.0.1:8080/property/list
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
Body
Cookies
Headers (7)
Test Results
200 OK
16 ms
504 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
[
{
"id": 1,
"title": "Small apartment",
"description": "City center",
"city": "Moscow",
"address": "Mira street, 1",
"guestLimit": 3,
"price": 200
},
{
"id": 2,
"title": "Nice apartment",
"description": "City center",
"city": "Moscow",
"address": "Mira street, 15",
"guestLimit": 7,
"price": 500
}
]


GET
http://127.0.0.1:8080/bookings/get/3
Send
Params
Authorization
Headers (6)
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 (7)
Test Results
200 OK
46 ms
596 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"id": 3,
"checkin": "06.15.2022",
"checkout": "06.19.2022",
"tenant": {
"id": 2,
"firstName": "Test2",
"lastName": "Testov",
"email": "test2@example.com",
"password": "$2b$08$GgxD4aoJs.xw0/SGqIuZD.ZCoSKERu1ozfAZmzO4d6hKYqlwA830G"
},
"property": {
"id": 2,
"title": "Nice apartment",
"description": "City center",
"city": "Moscow",
"address": "Mira street, 15",
"guestLimit": 7,
"price": 500
}
}


POST
http://127.0.0.1:8080/bookings/update/3
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"tenant": 2,
"property": 2,
"checkin": "06.11.2022",
"checkout": "06.19.2022"
}
Body
Cookies
Headers (7)
Test Results
200 OK
50 ms
277 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"generatedMaps": [],
"raw": [],
"affected": 1
}





Вывод: в ходе лабораторной работы мы реализовали RESTful API средствами Express + TypeORM + TypeScript для платформы для поиска и бронирования номера в отеле/квартире/хостеле.