

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

**Отчет**

по лабораторной работе № 3 «Реализация серверной части на django rest.  
Документирование API»

по дисциплине «**Web-программирование**»

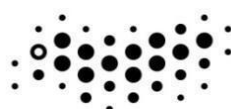
Автор: Власов М. И.

Факультет: ИКТ

Группа: K33402

Преподаватель: Говоров А. И.

Дата: 11.12.21

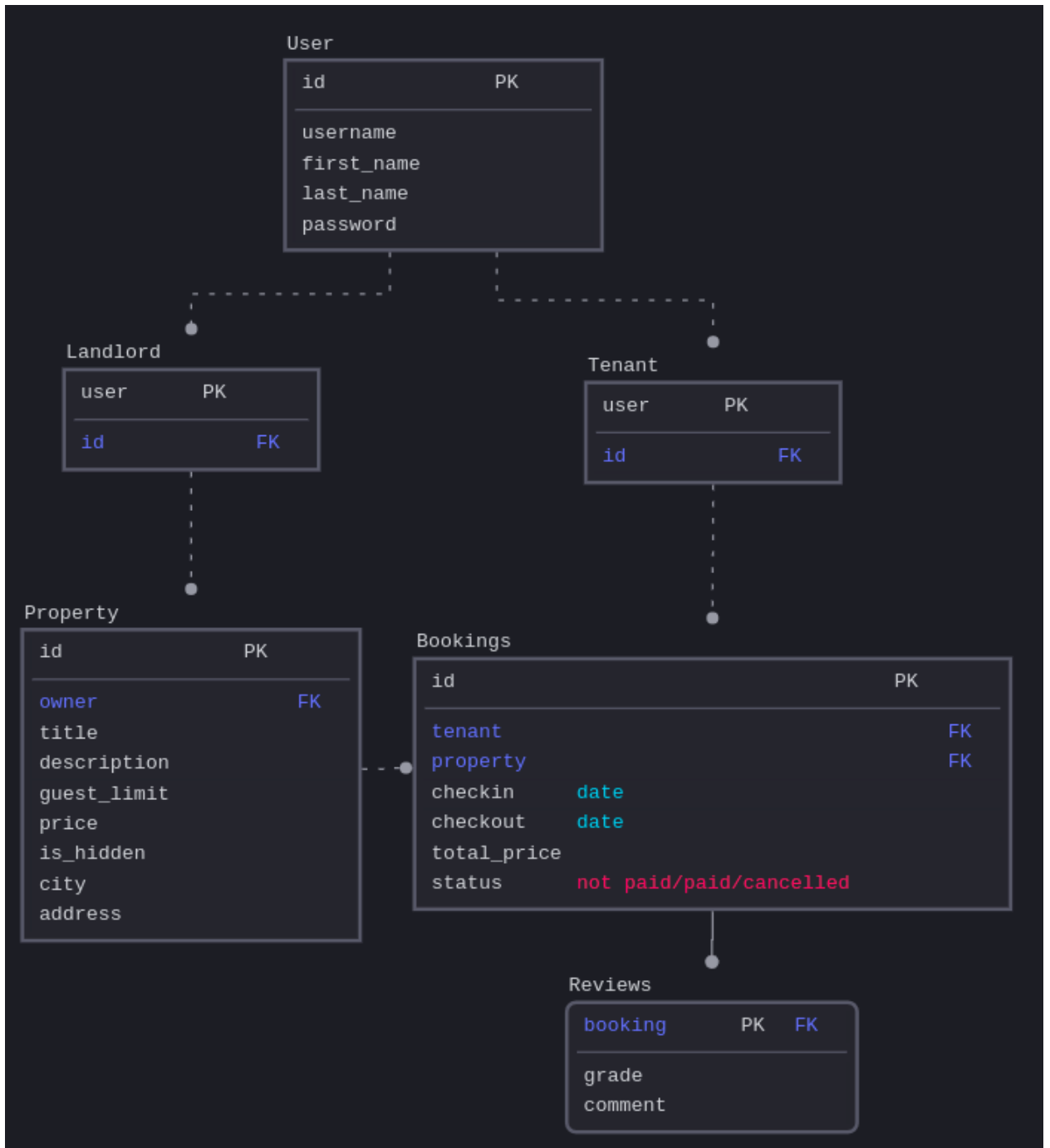


**УНИВЕРСИТЕТ ИТМО**

Санкт-Петербург, 2021

**Цель работы:** овладеть практическими навыками реализации серверной части (backend) приложений средствами Django REST framework.

## Модель базы данных



## Реализация API

Мы создали API для всех наших таблиц. Рассмотрим для примера таблицу *User*

Для начала получим список пользователей (*ListView*)

```
GET /user/list/
```

**HTTP 200 OK**

**Allow:** GET, HEAD, OPTIONS

**Content-Type:** application/json

**Vary:** Accept

```
[
  {
    "id": 1,
    "username": "admin",
    "first_name": "",
    "last_name": ""
  },
  {
    "id": 3,
    "username": "passtest",
    "first_name": "Pass",
    "last_name": "Test"
  },
  {
    "id": 4,
    "username": "tenant",
    "first_name": "Te",
    "last_name": "Nant"
  },
  {
    "id": 5,
    "username": "nate",
    "first_name": "Te",
    "last_name": "Na"
  },
  {
    "id": 6,
    "username": "onemore",
    "first_name": "One",
    "last_name": "More"
  },
  {
    "id": 7,
    "username": "todelete"
```

Теперь посмотрим информацию о конкретном пользователе (*RetrieveView*)

```
GET /user/6

HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
  "id": 6,
  "username": "onemore",
  "first_name": "One",
  "last_name": "More"
}
```

Так выглядит окно для создания нового пользователя (*CreateView*)

GET /user/create/

HTTP 405 Method Not Allowed  
Allow: POST, OPTIONS  
Content-Type: application/json  
Vary: Accept  

```
{
  "detail": "Method \"GET\" not allowed."
}
```

Raw data

HTML form

Username

Required. 150 characters or fewer. Letters, digits and @/./+/-/\_ only.

First name

Last name

Password

POST

## Давайте обновим пользователя с номером 3 (*UpdateView*)

GET /user/update/3

```
HTTP 200 OK
Allow: GET, PUT, PATCH, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
  "id": 3,
  "username": "passtest",
  "first_name": "Pass",
  "last_name": "Test",
  "password": "pbkdf2_sha256$320000$W44A0rc5gzb8jF7oiDo0vX$jrvS1kcbE+LPRT/UTFDD0hg7RcJu5KSr2XuooW01Is="
}
```

Raw data HTML form

Username   
Required. 150 characters or fewer. Letters, digits and @/./+/-/\_ only.

First name

Last name

Password

PUT

## Посмотрим на результат (видим, что изменился шифр пароля)

PUT /user/update/3

```
HTTP 200 OK
Allow: GET, PUT, PATCH, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
  "id": 3,
  "username": "passtest",
  "first_name": "Pass",
  "last_name": "Test",
  "password": "pbkdf2_sha256$320000$Sub2Hsc1LafC9urFSECdqjb$St3ZxmsKj1+kmDNSWU/YVT0fNFsSA9hLxDma55h/RkQ="
}
```

## Удалим пользователя с ником todelete (*DeleteView*)

User Destroy

Are you sure you want to delete this User Destroy?

Cancel Delete

DELETE OPTIONS GET ▾

GET /user/delete/7

```
HTTP 200 OK
Allow: GET, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
  "id": 7,
  "username": "todelete",
  "first_name": "To",
  "last_name": "Delete"
}
```

Убедимся, что пользователь успешно удалён

```
GET /user/7
```

**HTTP 404 Not Found**

**Allow:** GET, HEAD, OPTIONS

**Content-Type:** application/json

**Vary:** Accept

```
{
  "detail": "Not found."
}
```

Перейдём к таблице *Property*. Обратим внимание, что некоторые поля имеют вложенность больше единицы

```
GET /property/list/
```

**HTTP 200 OK**

**Allow:** GET, HEAD, OPTIONS

**Content-Type:** application/json

**Vary:** Accept

```
[
  {
    "id": 1,
    "owner": {
      "user": {
        "id": 3,
        "username": "passtest",
        "first_name": "Pass",
        "last_name": "Test"
      }
    },
    "city": "Saint Petersburg",
    "address": "Belorusskaya street, 6",
    "title": "Dorm",
    "description": "No",
    "guest_limit": 3,
    "price": 10,
    "is_hidden": false
  },
  {
    "id": 2,
    "owner": {
      "user": {
        "id": 1,
        "username": "admin",
        "first_name": "Admin",
        "last_name": "Adminov"
      }
    },
    "city": "Moscow",
    "address": "Lenina street, 1",
    "title": "Apartment",
    "description": "Very good",
    "guest_limit": 5,
    "price": 111,
    "is_hidden": false
  }
]
```

## Забронируем жильё

Tenant	One More	▼
Property	Moscow   Apartment	▼
Checkin	12/30/2021	📅
Checkout	01/03/2022	📅
Status	Unpaid	▼

POST

Убедимся, что появилась соответствующая запись. Заметим, что автоматически была подсчитана общая стоимость на основе даты заезда, выезда и цены за ночь

```
{
  "tenant": {
    "user": {
      "id": 6,
      "username": "onemore",
      "first_name": "One",
      "last_name": "More"
    }
  },
  "property": {
    "id": 2,
    "owner": {
      "user": {
        "id": 1,
        "username": "admin",
        "first_name": "Admin",
        "last_name": "Adminov"
      }
    },
    "city": "Moscow",
    "address": "Lenina street, 1",
    "title": "Apartment",
    "description": "Very good",
    "guest_limit": 5,
    "price": 111,
    "is_hidden": false
  },
  "checkin": "2021-12-30",
  "checkout": "2022-01-03",
  "total_price": 444,
  "status": 1
}
```

## Подключение регистрации и авторизации по токенам

Создадим нового пользователя

POST http://127.0.0.1:8000/auth/users/ Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

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

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	username	firstuser			
<input checked="" type="checkbox"/>	password	firstpass			
	Key	Value	Description		

Body Cookies Headers (10) Test Results 201 Created 309 ms 368 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "email": "",
3   "username": "firstuser",
4   "id": 8
5 }
```

Авторизуемся и получим токен

POST http://127.0.0.1:8000/auth/token/login/ Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

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

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	username	firstuser			
<input checked="" type="checkbox"/>	password	firstpass			
	Key	Value	Description		

Body Cookies Headers (10) Test Results 200 OK 368 ms 367 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "auth_token": "8b54b0bc2328a80201941fc5a3bba0385f1177b7"
3 }
```



Получим информацию о текущем пользователе с помощью токена

GET

http://127.0.0.1:8000/auth/users/me

Send

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Settings

Cookies

Headers

6 hidden

	KEY	VALUE	DESCRIPTION		Bulk Edit	Presets
<input checked="" type="checkbox"/>	Authorization	Token 8b54b0bc2328a80201941fc5a3bba...				
	Key	Value	Description			

Body

Cookies

Headers (10)

Test Results

200 OK 14 ms 377 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

"email": "",

"id": 8,

"username": "firstuser"

Документация

В Swagger документация выглядит следующим образом

review

POST

/review/create/

review\_create\_create

GET

/review/delete/{booking}

review\_delete\_read

DELETE

/review/delete/{booking}

review\_delete\_delete

GET

/review/list/

review\_list\_list

GET

/review/update/{booking}

review\_update\_read

PUT

/review/update/{booking}

review\_update\_update

PATCH

/review/update/{booking}

review\_update\_partial\_update

GET

/review/{booking}

review\_read

# Рассмотрим метод создания отзыва подробнее

POST /review/create/

review\_create\_create

Create review

Parameters

Try it out

Name	Description
<b>data</b> * required object (body)	Example Value   Model <pre>ReviewCreateUpdate {   booking integer title: Booking   grade integer title: Grade Enum:     &gt; Array [ 5 ]   comment string title: Comment maxLength: 500 }</pre>

Responses

Response content type application/json

Code	Description
201	Example Value   Model <pre>ReviewCreateUpdate {   booking integer title: Booking   grade integer title: Grade Enum:     &gt; Array [ 5 ]   comment string title: Comment maxLength: 500 }</pre>

# Перейдём к redoc

🔍 Search...

Authentication

auth >

booking ▾

POST booking\_create\_create

GET booking\_delete\_read

DELETE booking\_delete\_delete

GET booking\_list\_list

GET booking\_update\_read

PUT booking\_update\_update

POST booking\_update\_partial\_upd...

GET booking\_read

landlord >

property >

review >

tenant >

user >

Documentation Powered by Redoc

booking\_create\_create

Create booking

AUTHORIZATIONS: Basic

REQUEST BODY SCHEMA: application/json

tenant required	string (Tenant)
property required	integer (Property)
checkin required	string <date> (Checkin)
checkout required	string <date> (Checkout)
status required	integer (Status) Enum: -1 0 1

Responses

201

RESPONSE SCHEMA: application/json

tenant required	string (Tenant)
property required	integer (Property)
checkin required	string <date> (Checkin)
checkout required	string <date> (Checkout)
status required	integer (Status) Enum: -1 0 1

POST /booking/create/

Request samples

Payload

Content type application/json

```
{
  "tenant": "string",
  "property": 0,
  "checkin": "2019-08-24",
  "checkout": "2019-08-24",
  "status": -1
}
```

Response samples

201

Content type application/json

```
{
  "tenant": "string",
  "property": 0,
  "checkin": "2019-08-24",
  "checkout": "2019-08-24",
  "status": -1
}
```

Рассмотрим параметры создания бронирования подробнее

## booking\_create\_create

Create booking

AUTHORIZATIONS: [Basic](#)

REQUEST BODY SCHEMA: application/json

tenant required	string (Tenant)
property required	integer (Property)
checkin required	string <date> (Checkin)
checkout required	string <date> (Checkout)
status required	integer (Status) Enum: <input type="radio"/> -1 <input type="radio"/> 0 <input type="radio"/> 1

Эти же параметры в формате *json*

```
Content type
application/json

Copy Expand all Collapse all

{
  "tenant": "string",
  "property": 0,
  "checkin": "2019-08-24",
  "checkout": "2019-08-24",
  "status": -1
}
```

Вывод: в ходе лабораторной работы мы овладели практическими навыками реализации серверной части (backend) приложений средствами Django REST framework.