

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИТМО

Отчет по домашней работе №3 по курсу “Бэкенд разработка”

Выполнили:

Бахарева М. А., К3342

Привалов К.А., К3342

Проверил:

Добряков Д.И.

Санкт-Петербург

2025 г.

1. Задание:

- реализовать автодокументирование средствами swagger;
- реализовать документацию API средствами Postman.

2. Ход работы

Документирование Swagger

Инициализируем проект и устанавливаем зависимости

С помощью нескольких команд выполняем инициализацию и установку необходимых библиотек:

```
mariabakhareva@Marias-MacBook-Air-2 lr2 % npm install swagger-ui-express swagger-jsdoc

up to date, audited 337 packages in 1s

44 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Инициализируем swagger в [app.ts](#)

```
import express from 'express';
import helmet from 'helmet';
import morgan from 'morgan';
import routes from './routes';
import errorHandler from './middleware/errorHandler';
import swaggerJsdoc from 'swagger-jsdoc';
import * as swaggerUi from 'swagger-ui-express';

const app = express();

app.use(helmet());
app.use(express.json());
app.use(morgan('dev'));

app.use('/api', routes);

const swaggerDefinition = {
```

```

openapi: '3.0.0',
info: {
  title: 'Rent A Property API',
  version: '1.0.0',
  description: 'Auto-generated swagger docs',
},
servers: [
  {
    url: `http://localhost:${process.env.PORT || 3000}`,
  },
],
components: {
  securitySchemes: {
    bearerAuth: {
      type: 'http',
      scheme: 'bearer',
      bearerFormat: 'JWT',
    },
  },
},
};

const swaggerOptions = {
  swaggerDefinition,
  apis: [
    './src/routes/**/*.ts',
    './src/controllers/**/*.ts',
    './src/entities/**/*.ts',
  ],
};

const swaggerSpec = swaggerJsdoc(swaggerOptions);
app.use('/docs', swaggerUi.serve, swaggerUi.setup(swaggerSpec));

app.use(errorHandler);

export default app;

```

Добавляем комментарии в код, отражающие запрос и возможные респонсы. Пример из booking request:

```

/**
 * @swagger
 * /api/booking-requests:
 *   get:

```

```

*      summary: Get all booking requests
*      tags: [Booking Requests]
*      security:
*        - bearerAuth: []
*      responses:
*        200:
*          description: List of booking requests
*/
router.get('/', checkJwt, BookingRequestController.getAll);

```

Полученный результат по энд-пойнту <http://localhost:3000/docs/>

Booking Requests <small>Endpoints for managing booking requests</small>			^
POST	/api/booking-requests	Create a new booking request	🔒 ▼
GET	/api/booking-requests/{id}	Get booking request by ID	🔒 ▼
PUT	/api/booking-requests/{id}	Update a booking request	🔒 ▼
DELETE	/api/booking-requests/{id}	Delete a booking request	🔒 ▼
Chat <small>Chat management and messaging</small>			^
GET	/api/chat/{id}	Get a chat by ID	🔒 ▼
POST	/api/chat	Create a new chat	🔒 ▼
POST	/api/chat/{id}/message	Send a message to a chat	🔒 ▼

Complaints

Complaint management endpoints



POST /api/complaints Create a new complaint



GET /api/complaints/{id} Get complaint by ID



PUT /api/complaints/{id} Update a complaint



DELETE /api/complaints/{id} Delete a complaint



Favorites

Favorite management endpoints



GET /api/favorites/{id} Get a favorite by ID



PUT /api/favorites/{id} Update a favorite



DELETE /api/favorites/{id} Delete a favorite



POST /api/favorites Create a new favorite



Main

Main API routes for the application



GET /api/booking-requests Booking request-related operations



GET /api/complaints Complaint-related operations



GET /api/favorites Favorite-related operations



GET /api/chats Chat-related operations



GET /api/property-images Property image-related operations



Properties

Property management endpoints



GET /api/properties Get all properties



POST /api/properties Create a new property



GET /api/properties/{id} Get a property by ID



PUT /api/properties/{id} Update a property



DELETE /api/properties/{id} Delete a property



PropertyImages

Property image management endpoints



POST **/api/property-images/upload** Upload images for a property



GET **/api/property-images/{propertyId}** Get all images for a property



Rentals

Rental management endpoints



GET **/api/rentals** Get all rentals



POST **/api/rentals** Create a new rental



GET **/api/rentals/{id}** Get a rental by ID



PUT **/api/rentals/{id}** Update a rental



DELETE **/api/rentals/{id}** Delete a rental



Reviews

Review management endpoints



GET **/api/reviews** Get all reviews



POST **/api/reviews** Create a new review



GET **/api/reviews/{id}** Get a review by ID



PUT **/api/reviews/{id}** Update a review



DELETE **/api/reviews/{id}** Delete a review



Users

User management endpoints



GET **/api/users** Get all users



GET **/api/users/{id}** Get user by ID



PUT **/api/users/{id}** Update user by ID



DELETE **/api/users/{id}** Delete user by ID



POST **/api/users/{id}/change-password** Change user password



POST **/api/users/login** Login user



Пример задокументированного энд-пойнта:

Booking Requests

Endpoints for managing booking requests

POST /api/booking-requests Create a new booking request

Try it out

Parameters

No parameters

Request body required application/json

Example Value | Schema

```
{  "propertyId": "string",  "startDate": "2025-05-11",  "endDate": "2025-05-11"}  
```

Responses

Code	Description	Links
201	Booking request created	No links

Пример выполненного запроса регистрации и логина пользователя с помощью Swagger

POST /api/users/register Register a new user

Cancel Reset

Parameters

No parameters

Request body required application/json

```
{  "email": "maria@mail.ru",  "password": "Maria12345!",  "firstName": "Maria",  "lastName": "Bakhareva",  "birthDate": "2004-07-21",  "phone": "89897951199",  "role": "tenant"}  
```

Execute Clear

Curl

```
curl -X 'POST' \
  'http://localhost:3000/api/users/register' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "email": "maria@mail.ru",
    "password": "Maria12345!",
    "firstName": "Maria",
    "lastName": "Bakhareva",
    "birthDate": "2004-07-21",
    "phone": "89897951199",
    "role": "tenant"
  }'
```

Request URL

http://localhost:3000/api/users/register

Server response

Code

Details

201

Response body

```
{
  "firstName": "Maria",
  "lastName": "Bakhareva",
  "birthDate": "2004-07-21",
  "phone": "89897951199",
  "email": "maria@mail.ru",
  "role": "tenant",
  "id": 1,
  "createdAt": "2025-05-11T14:42:24.520Z"
}
```



Download

Response headers

```
connection: keep-alive
content-length: 177
content-security-policy: default-src 'self';base-uri 'self';font-src 'self' https: data:;form-action 'self';frame-ancestors 'self';img-src 'self' data:;object-src 'none';script-src 'self';script-src-attr 'none';style-src 'self' https: 'unsafe-inline';upgrade-insecure-requests
content-type: application/json; charset=utf-8
cross-origin-opener-policy: same-origin
cross-origin-resource-policy: same-origin
date: Sun 11 May 2025 14:42:24 GMT
```

POST /api/users/login Login user

Parameters

Cancel

Reset


No parameters

Request body required

application/json

```
{
  "email": "maria@mail.ru",
  "password": "Maria12345!"
}
```


Server response

Code	Details
200	<p>Response body</p> <pre>{ "finalUser": { "id": 1, "name": "Maria", "lastName": "Bakhareva", "email": "maria@mail.ru", "role": "tenant" }, "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiJEsInJvbGUiOiJ0ZW5hbnQiLCJpYXQiOiJlE3NDY5Nz01NjksImV4cCI6MTc0Njk3ODE2OX0.8vACho6gkvUxA0Yw-q6x97Rp4C2SyDGiRdoKjQ6Fq6g" }</pre> <div> Download</div>
	<p>Response headers</p> <pre>connection: keep-alive content-length: 275 content-security-policy: default-src 'self';base-uri 'self';font-src 'self' https: data;;form-action 'self';frame-ancestors 'self';img-src 'self' data;object-src 'none';script-src 'self';script-src-attr 'none';style-src 'self' https: 'unsafe-inline';upgrade-insecure-requests content-type: application/json; charset=utf-8 cross-origin-opener-policy: same-origin cross-origin-resource-policy: same-origin date: Sun,11 May 2025 14:42:49 GMT etag: W/"113-I0LR5BKg+ye7qSU0h/PUHRNphE4" keep-alive: timeout=5 origin-agent-cluster: 71 referrer-policy: no-referrer strict-transport-security: max-age=31536000; includeSubDomains x-content-type-options: nosniff x-dns-prefetch-control: off x-download-options: noopen x-frame-options: SAMEORIGIN x-permitted-cross-domain-policies: none x-xss-protection: 0</pre>

Документирование Postman

Зададим в [app.ts](#) отдельный энд-пойнт для генерации swagger.json.

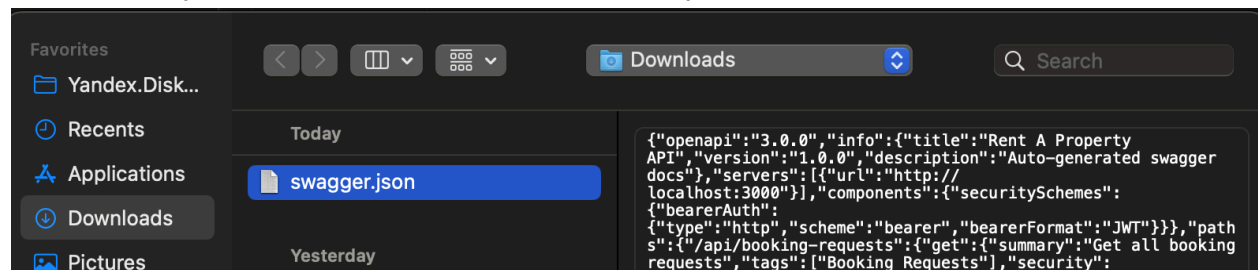
```
app.get('/swagger.json', (req, res) => {
  res.setHeader('Content-Type', 'application/json');
  res.send(swaggerSpec);
});
```

Получаем по энд-пойнту json и скачиваем его

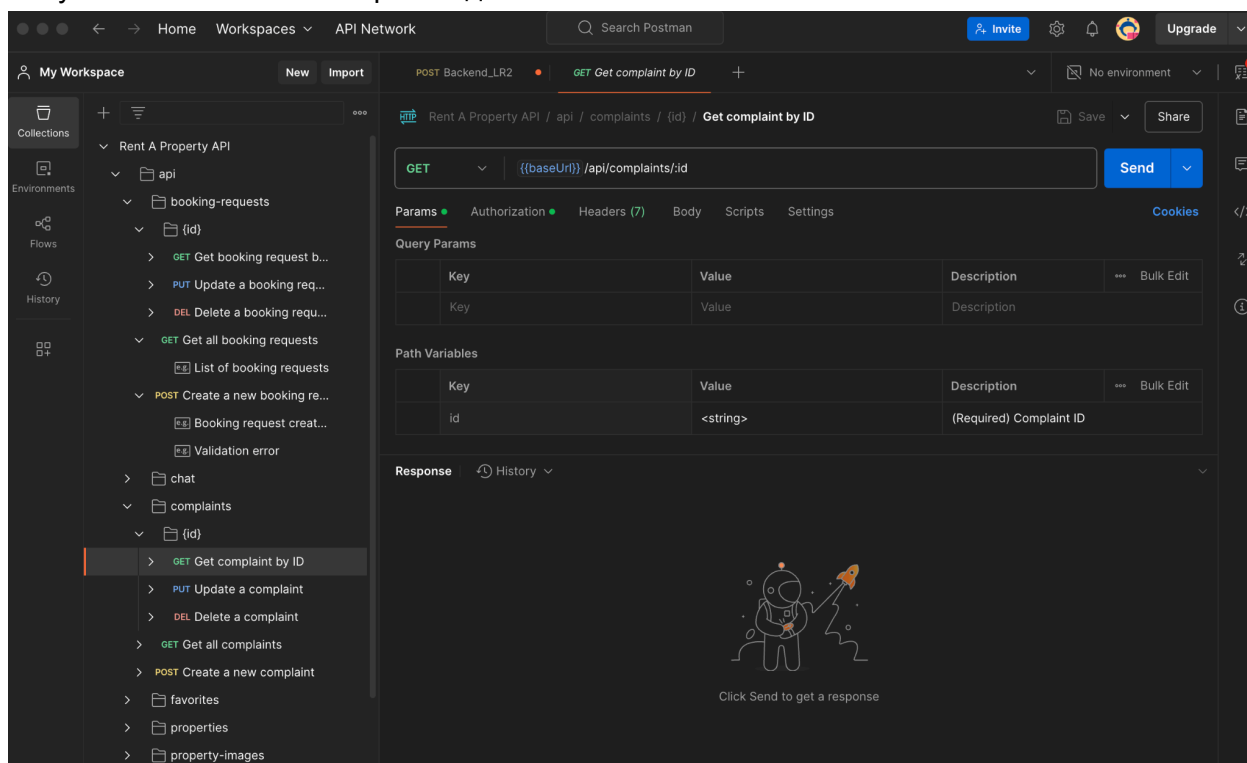
```
localhost:3000/swagger.json x +
localhost:3000/swagger.json
Автоформатировать ☐

{"openapi":"3.0.0","info":{"title":"Rent A Property API","version":"1.0.0","description":"Auto-generated swagger docs"},"servers":[{"url":"http://localhost:3000"}],"components":{"securitySchemes":{"bearerAuth":{"type":"http","scheme":"bearer","bearerFormat":"JWT"}}},"paths":{"/api/booking-requests":{"get":{"summary":"Booking request-related operations","tags":["Main"],"security":[{"bearerAuth":[]}],"responses":{"200":{"description":"Successfully returned booking request routes"},"description":"This endpoint handles booking request actions"},"post":{"summary":"Create a new booking request","tags":["Booking Requests"],"security":[{"bearerAuth":[]}],"requestBody":{"required":true,"content":{"application/json":{"schema":{"type":"object","properties":{"propertyId":{"type":"string"},"startDate":{"type":"string","format":"date"},"endDate":{"type":"string","format":"date"}}}}},"responses":{"201":{"description":"Booking request created"}}},"/api/booking-requests/{id}":{"get":{"summary":"Get booking request by ID","tags":["Booking Requests"],"security":[{"bearerAuth":[]}],"parameters":{"name":"id","in":"path","description":"Booking request ID","required":true,"schema":{"type":"string"},"responses":{"200":{"description":"Booking request found"},"404":{"description":"Booking request not found"},"put":{"summary":"Update a booking request","tags":["Booking Requests"],"security":[{"bearerAuth":[]}],"parameters":{"name":"id","in":"path","description":"Booking request ID","required":true,"schema":{"type":"string"},"requestBody":{"required":true,"content":{"application/json":{"schema":{"type":"object","properties":{"startDate":{"type":"string","format":"date"},"endDate":{"type":"string","format":"date"}}}}},"responses":{"200":{"description":"Booking request updated"},"403":{"description":"Forbidden"},"404":{"description":"Not found"},"delete":{"summary":"Delete a booking request","tags":["Booking Requests"],"security":[{"bearerAuth":[]}],"parameters":{"name":"id","in":"path","description":"Booking request ID","required":true,"schema":{"type":"string"},"responses":{"204":{"description":"Booking request deleted"},"403":{"description":"Forbidden"},"404":{"description":"Not found"}}},"/api/chat/{id}":{"get":{"summary":"Get a chat by ID","tags":["Chat"],"security":[{"bearerAuth":[]}],"parameters":{"in":"path","name":"id","description":"Chat ID","required":true,"schema":{"type":"string"},"responses":{"200":{"description":"Chat retrieved successfully"},"404":{"description":"Chat not found"},"post":{"summary":"Create a new chat","tags":["Chat"],"security":[{"bearerAuth":[]}],"requestBody":{"required":true,"content":{"application/json":{"schema":{"type":"object","required":{"propertyId","recipientId"},"properties":{"propertyId":{"type":"string"},"recipientId":{"type":"string"}}}}},"responses":{"201":{"description":"Chat created"},"400":{"description":"Validation error"}}},"/api/chat/{id}/message":{"post":{"summary":"Send a message to a chat","tags":["Chat"],"security":[{"bearerAuth":[]}],"parameters":{"in":"path","name":"id","description":"Chat ID","required":true,"schema":{"type":"string"},"requestBody":{"required":true,"content":{"application/json":{"schema":{"type":"object","required":{"content"},"properties":{"content":{"type":"string"}}}}},"responses":{"201":{"description":"Message sent"},"400":{"description":"Validation error"},"404":{"description":"Chat not found"}}},"/api/complaints":{"get":{"summary":"Complaint-related operations","tags":["Main"],"security":[{"bearerAuth":[]}],"responses":{"200":{"description":"Successfully returned complaint routes"},"description":"This endpoint handles complaint actions (file a complaint, resolve)","post":{"summary":"Create a new complaint","tags":["Complaints"],"security":[{"bearerAuth":[]}],"requestBody":{"required":true,"content":{"application/json":{"schema":{"type":"object","required":{"subject","message"},"properties":{"subject":{"type":"string"},"message":{"type":"string"}}}}},"responses":{"201":{"description":"Complaint created"},"400":{"description":"Validation error"},"403":{"description":"Forbidden"},"404":{"description":"Complaint not found"},"get":{"summary":"Get complaint by ID","tags":["Complaints"],"security":[{"bearerAuth":[]}],"parameters":{"in":"path","name":"id","required":true,"description":"Complaint ID","schema":{"type":"string"},"responses":{"200":{"description":"Complaint found"},"404":{"description":"Complaint not found"},"put":{"summary":"Update a complaint","tags":["Complaints"],"security":[{"bearerAuth":[]}],"parameters":{"in":"path","name":"id","required":true,"description":"Complaint ID","schema":{"type":"string"},"requestBody":{"required":true,"content":{"application/json":{"schema":{"type":"object","properties":{"subject":{"type":"string"},"message":{"type":"string"}}}}},"responses":{"200":{"description":"Complaint updated"},"403":{"description":"Forbidden"},"404":{"description":"Complaint not found"},"delete":{"summary":"Delete a complaint","tags":["Complaints"],"security":[{"bearerAuth":[]}],"parameters":{"in":"path","name":"id","required":true,"description":"Complaint ID","schema":{"type":"string"},"responses":{"204":{"description":"Complaint deleted"},"403":{"description":"Forbidden"},"404":{"description":"Complaint not found"}}
```

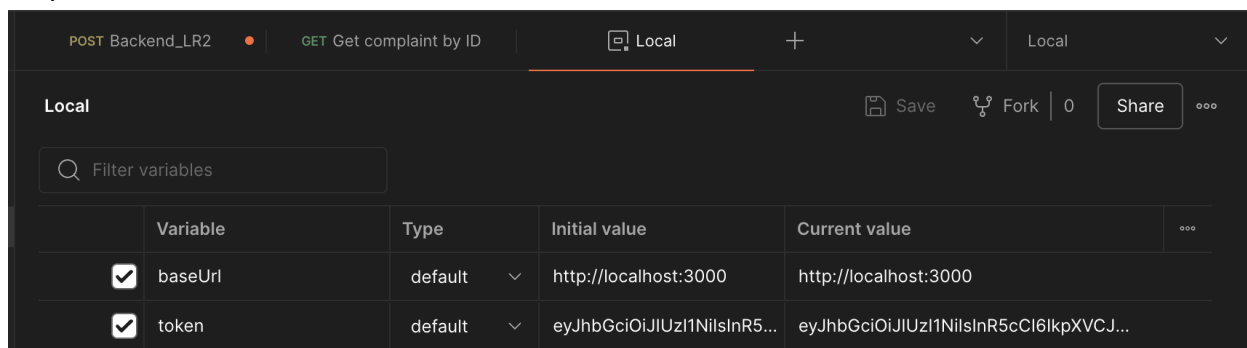
Создаем новую коллекцию в Postman и импортируем в нее этот файл



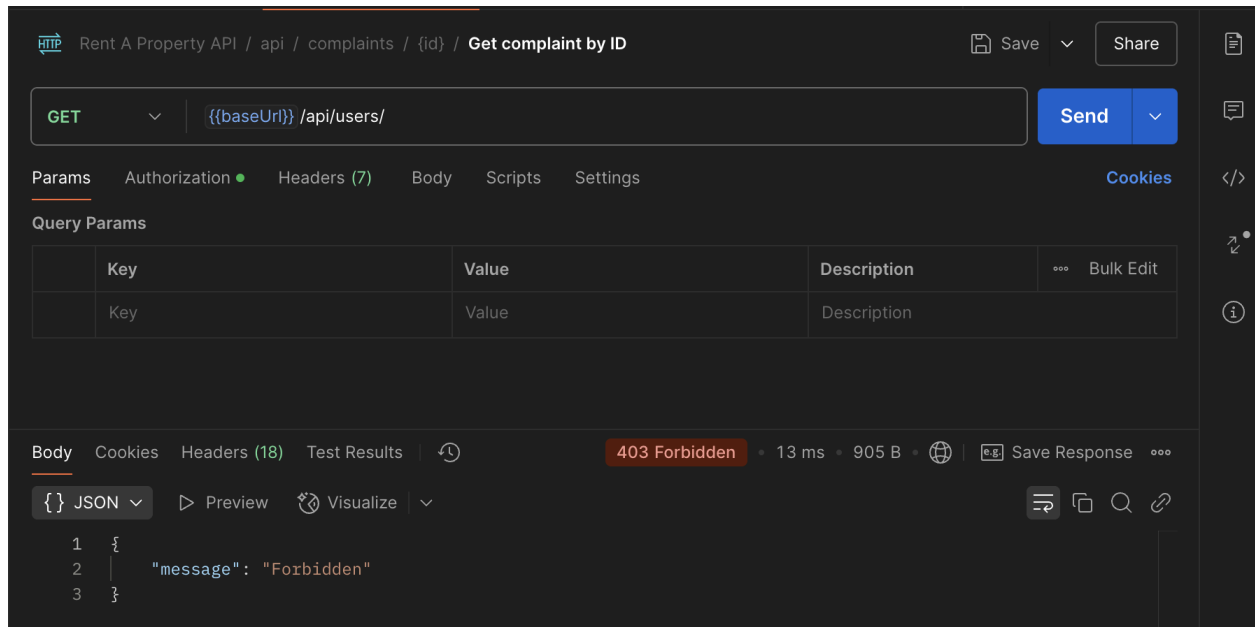
Получаем Collection с набором энд-пойнтов



Настроим окружение. Добавим дефолтный URL, а также пропишем токен конкретного юзера (созданного с помощью Swagger выше), чтобы проверить работоспособность запросов.



Проверим работоспособность. Запрос прошел с корректным респонсом в виде запрета видеть всех юзеров системы, так как пользователь не обладает полным набором прав (у него роль - tenant)



3. Вывод

В рамках выполнения данной работы была реализована документация REST API средствами Swagger и Postman. Работа включала как оформление эндпоинтов в виде комментариев OpenAPI в коде, так и проверку корректности работы API через импорт документации в Postman.