Университет ИТМО

Мегафакультет компьютерных технологий и управления

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2

по дисциплине

“Бизнес-логика программных систем”

Вариант: 503

Группа: P33112

Выполнили: Атикеев Роман,

Кульбако Артемий Юрьевич

Преподаватель: Каюков Иван Алексеевич

Санкт-Петербург

2021 год

Задание

Доработать приложение из лабораторной работы #1, реализовав в нём управление транзакциями и разграничение доступа к операциям бизнес-логики в соответствии с заданной политикой доступа.

Управление транзакциями необходимо реализовать следующим образом:

1. Переработать согласованные с преподавателем прецеденты (или по согласованию с ним разработать новые), объединив взаимозависимые операции в рамках транзакций.
2. Управление транзакциями необходимо реализовать с помощью Spring JTA.
3. В реализованных (или модифицированных) прецедентах необходимо использовать программное управление транзакциями.
4. В качестве менеджера транзакций необходимо использовать Atomikos.

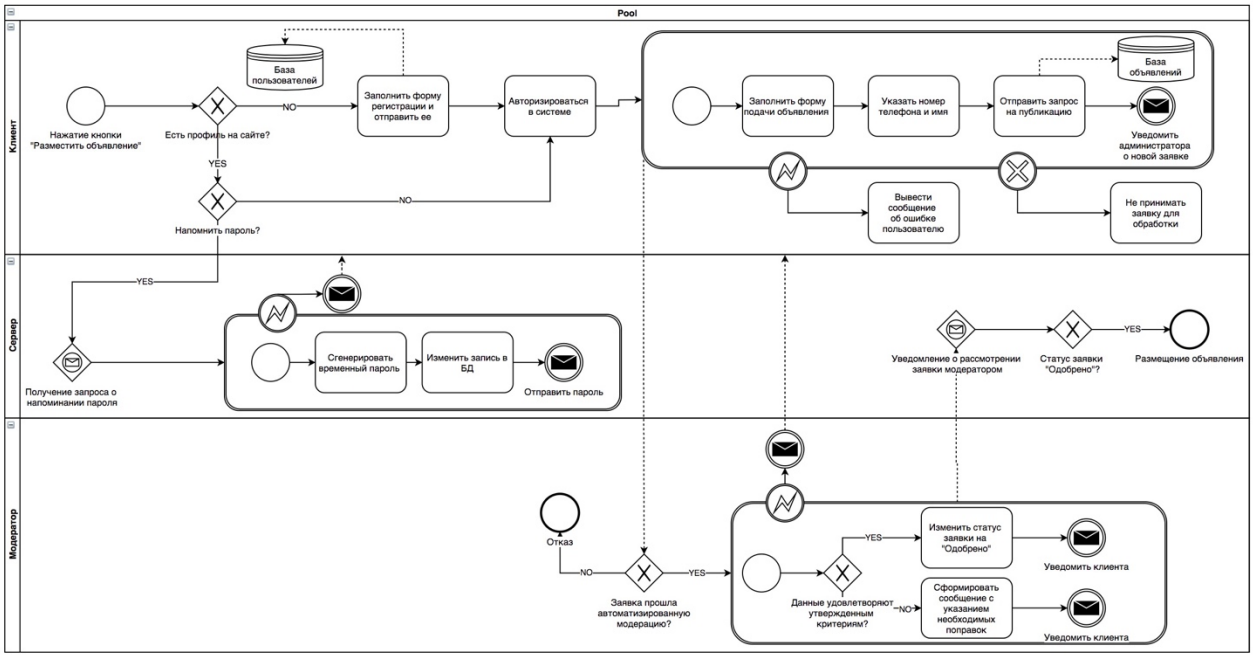
Разграничение доступа к операциям необходимо реализовать следующим образом:

1. Разработать, специфицировать и согласовать с преподавателем набор привилегий, в соответствии с которыми будет разграничиваться доступ к операциям.
2. Специфицировать и согласовать с преподавателем набор ролей, осуществляющих доступ к операциям бизнес-логики приложения.
3. Реализовать разработанную модель разграничений доступа к операциям бизнес-логики на базе Spring Security. Информацию об учётных записах пользователей необходимо сохранять в файле XML, для аутентификации использовать JWT.

Правила выполнения работы:

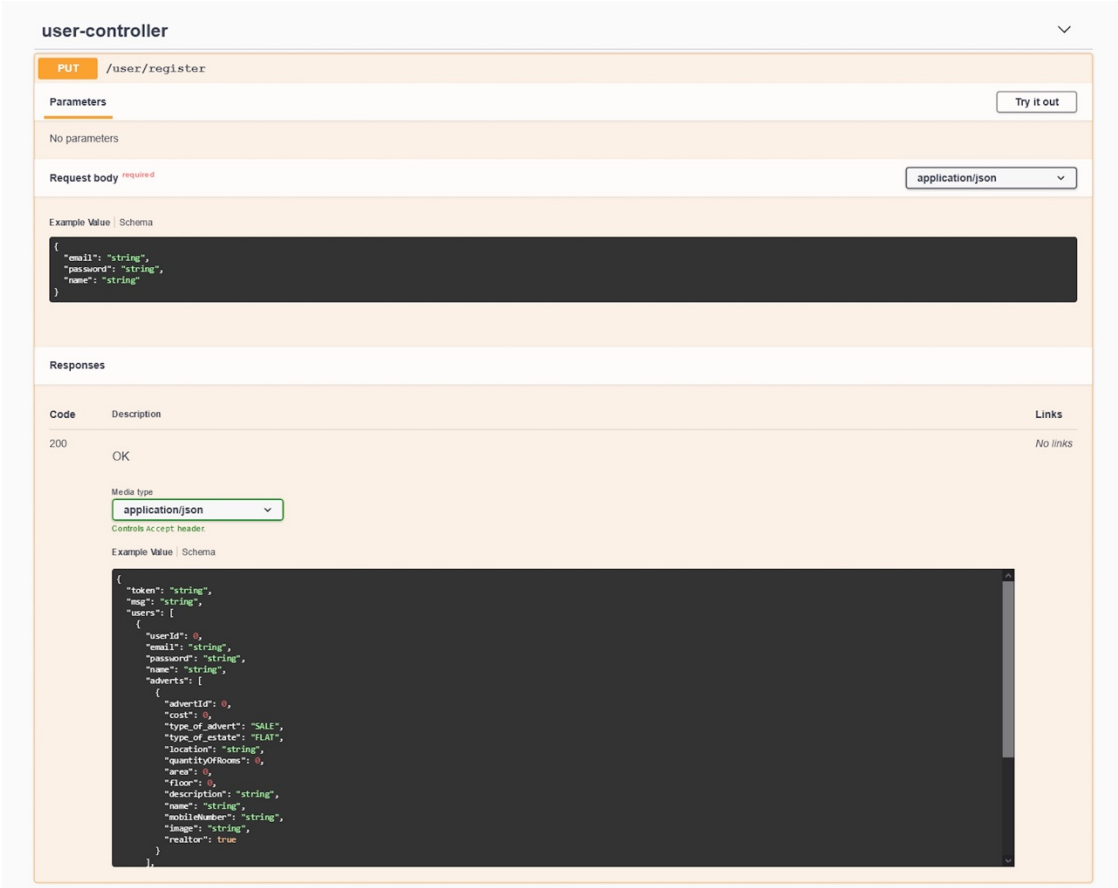
1. Все изменения, внесённые в реализуемый бизнес-процесс, должны быть учтены в описывающей его модели, REST API и наборе скриптов для тестирования публичных интерфейсов модуля.
2. Доработанное приложение необходимо развернуть на сервере helios.

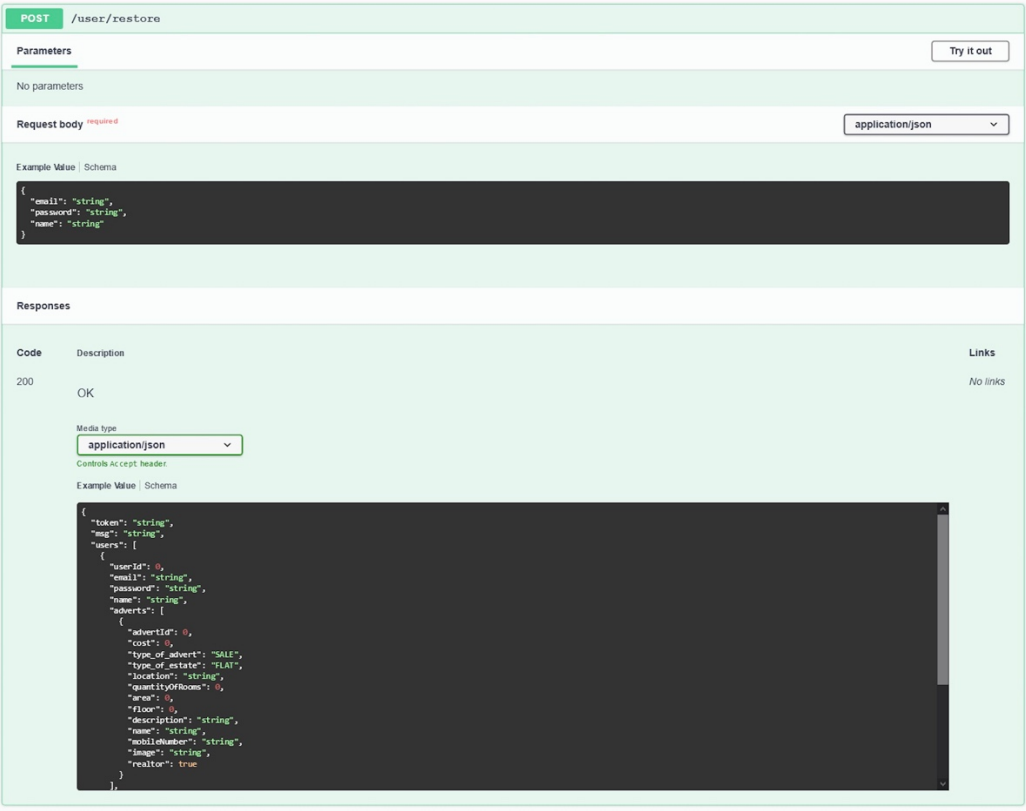
Модель потока управления

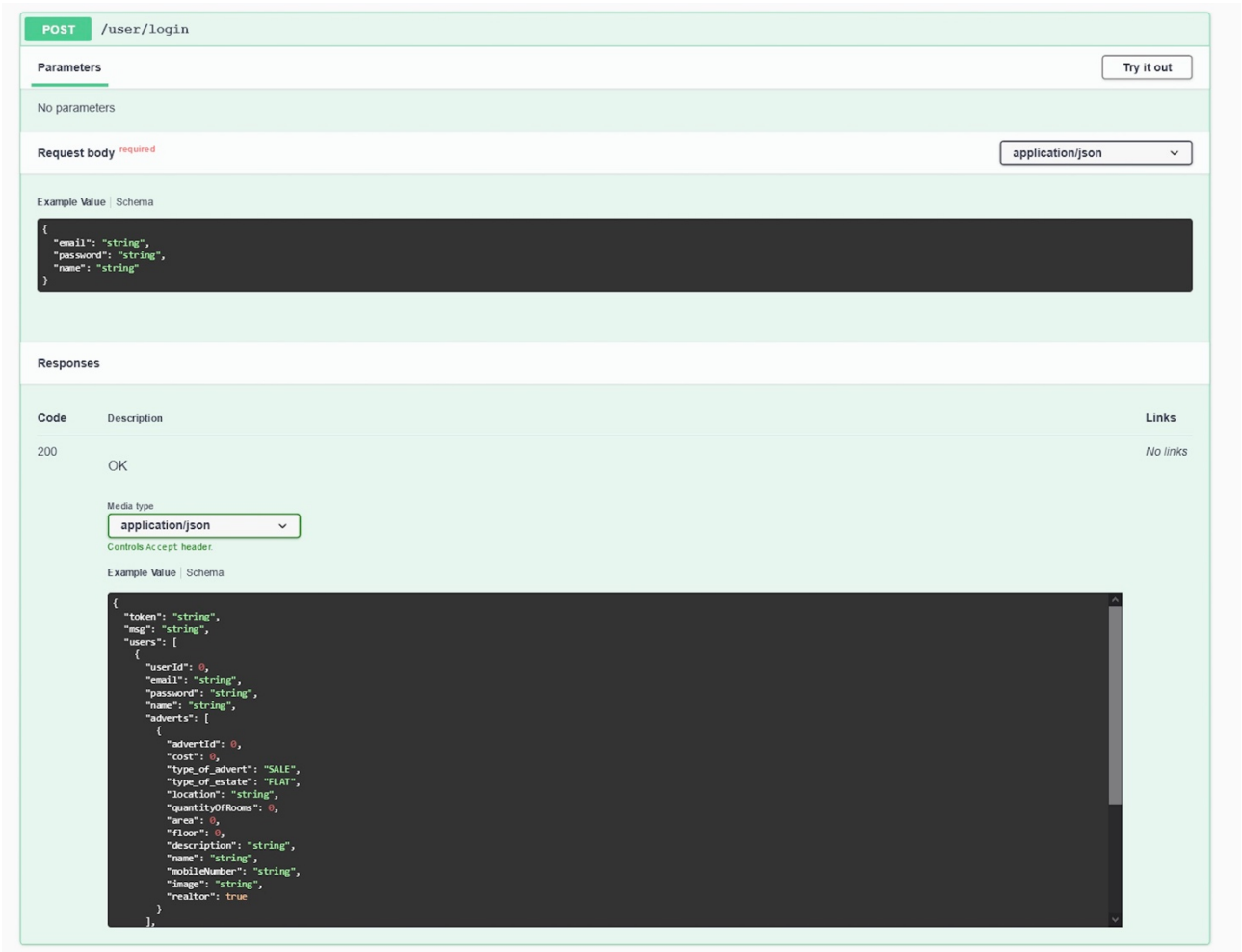


UML-диаграмма

REST API


user-controller
PUT
/user/register
Parameters
Try it out
No parameters
Request body
required
application/json
Example Value | Schema
{
"email": "string",
"password": "string",
"name": "string"
}
Responses
Code
Description
Links
200
OK
No links
Media type
application/json
Controls Accept header.
Example Value | Schema
{
"token": "string",
"msg": "string",
"users": [
{
"userId": 0,
"email": "string",
"password": "string",
"name": "string",
"adverts": [
{
"advertId": 0,
"cost": 0,
"type_of_advert": "SALE",
"type_of_estate": "FLAT",
"location": "string",
"quantityOfRooms": 0,
"area": 0,
"floor": 0,
"description": "string",
"name": "string",
"mobileNumber": "string",
"image": "string",
"realtor": true
}
],


POST
/user/restore
Parameters
Try it out
No parameters
Request body
required
application/json
Example Value | Schema
{
"email": "string",
"password": "string",
"name": "string"
}
Responses
Code
Description
Links
200
OK
No links
Media type
application/json
Controls Accept header.
Example Value | Schema
{
"token": "string",
"msg": "string",
"users": [
{
"userId": 0,
"email": "string",
"password": "string",
"name": "string",
"adverts": [
{
"advertId": 0,
"cost": 0,
"type_of_advert": "SALE",
"type_of_estate": "FLAT",
"location": "string",
"quantityOfRooms": 0,
"area": 0,
"floor": 0,
"description": "string",
"name": "string",
"mobileNumber": "string",
"image": "string",
"realtor": true
}
],


POST
/user/login
Parameters
Try it out
No parameters
Request body required
application/json
Example Value | Schema
{
"email": "string",
"password": "string",
"name": "string"
}
Responses
Code
Description
Links
200
OK
No links
Media type
application/json
Controls Accept header.
Example Value | Schema
{
"token": "string",
"msg": "string",
"users": [
{
"userId": 0,
"email": "string",
"password": "string",
"name": "string",
"adverts": [
{
"advertId": 0,
"cost": 0,
"type_of_advert": "SALE",
"type_of_estate": "FLAT",
"location": "string",
"quantityOfRooms": 0,
"area": 0,
"floor": 0,
"description": "string",
"name": "string",
"mobileNumber": "string",
"image": "string",
"realtor": true
}
],

PATCH /user/modify

Parameters | Try it out

No parameters

Request body required | application/json

Example Value | Schema

```
{
  "email": "string",
  "password": "string",
  "name": "string"
}
```

Responses

Code	Description	Links
200	OK Media type: application/json Controls Accept header. Example Value | Schema	*No links*

```
{
  "token": "string",
  "msg": "string",
  "users": [
    {
      "userId": 0,
      "email": "string",
      "password": "string",
      "name": "string",
      "adverts": [
        {
          "advertId": 0,
          "cost": 0,
          "type_of_advert": "SALE",
          "type_of_estate": "FLAT",
          "location": "string",
          "quantityOfRooms": 0,
          "area": 0,
          "floor": 0,
          "description": "string",
          "name": "string",
          "mobileNumber": "string",
          "image": "string",
          "realtor": true
        }
      ],
```

GET /user/{userId}

Parameters | Try it out

Name	Description
userId * required integer($int64) *(path)*	userId

Responses

Code	Description	Links
200	OK Media type: application/json Controls Accept header. Example Value | Schema	*No links*

```
{
  "token": "string",
  "msg": "string",
  "users": [
    {
      "userId": 0,
      "email": "string",
      "password": "string",
      "name": "string",
      "adverts": [
        {
          "advertId": 0,
          "cost": 0,
          "type_of_advert": "SALE",
          "type_of_estate": "FLAT",
          "location": "string",
          "quantityOfRooms": 0,
          "area": 0,
          "floor": 0,
          "description": "string",
          "name": "string",
          "mobileNumber": "string",
          "image": "string",
          "realtor": true
        }
      ],
```

GET /user/all

Parameters | Try it out

No parameters

Responses

Code	Description	Links
200	OK Media type: application/json Controls Accept header. Example Value | Schema	*No links*


```
{
  "token": "string",
  "msg": "string",
  "users": [
    {
      "userId": 0,
      "email": "string",
      "password": "string",
      "name": "string",
      "adverts": [
        {
          "advertId": 0,
          "cost": 0,
          "type_of_advert": "SALE",
          "type_of_estate": "FLAT",
          "location": "string",
          "quantityOfRooms": 0,
          "area": 0,
          "floor": 0,
          "description": "string",
          "name": "string",
          "mobileNumber": "string",
          "image": "string",
          "realtor": true
        }
      ],
```


DELETE /user/delete

Parameters | Try it out

No parameters

Request body required | application/json

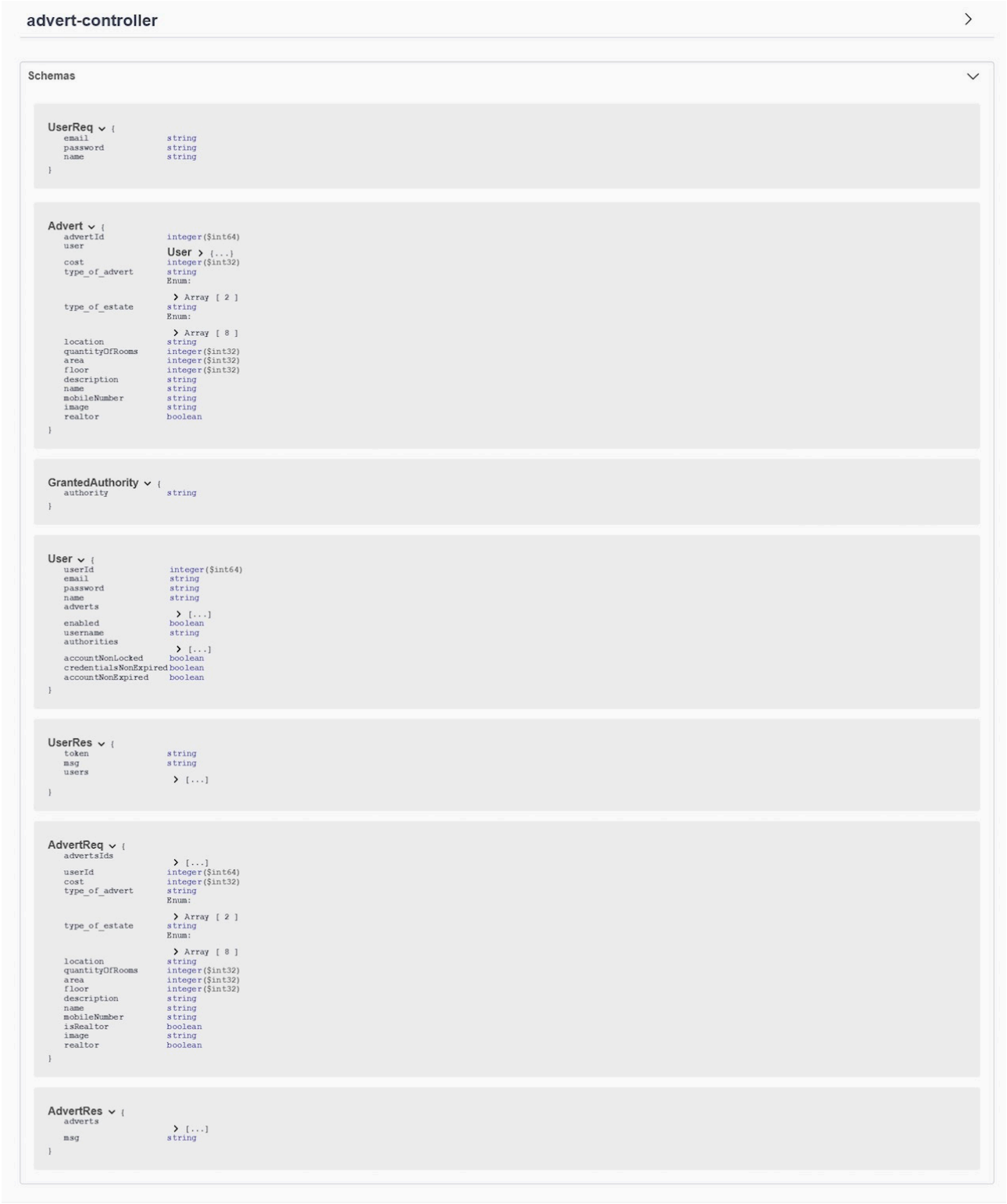
Example Value | Schema


```
{
  "email": "string",
  "password": "string",
  "name": "string"
}
```


Responses

Code	Description	Links
200	OK Media type: application/json Controls Accept header. Example Value | Schema	*No links*


```
{
  "token": "string",
  "msg": "string",
  "users": [
    {
      "userId": 0,
      "email": "string",
      "password": "string",
      "name": "string",
      "adverts": [
        {
          "advertId": 0,
          "cost": 0,
          "type_of_advert": "SALE",
          "type_of_estate": "FLAT",
          "location": "string",
          "quantityOfRooms": 0,
          "area": 0,
          "floor": 0,
          "description": "string",
          "name": "string",
          "mobileNumber": "string",
          "image": "string",
          "realtor": true
        }
      ],
```



Исходный код и развертывание

Исходный код программы можно найти в репозитории на github:

https://github.com/testpassword/Business-logic-of-software-systems

Приложение развернуто на сервере helios.

Вывод

В данной лабораторной работе мы подробнее ознакомились с механизмом работы транзакций и применили их для нашего конкретного случая. Помимо этого в процессе выполнения данной лабораторной работы были разработаны REST API, а также была выполнена непосредственно реализация самого бизнес-процесса с помощью языка kotlin.