|  |  |
| --- | --- |
| ГУАП  КАФЕДРА № 43  ОТЧЕТ  ЗАЩИЩЕН С ОЦЕНКОЙ  ПРЕПОДАВАТЕЛЬ |  |
|  |  |
| ассистент | Кочин Д.А. |

подпись, дата

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №7

Разработка приложения с асинхронной очередью сообщений

по дисциплине: ТЕХНОЛОГИИ РАЗРАБОТКИ СЕРВЕРНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4936 Петровнина Д.В.

подпись, дата фамилия, инициалы

Санкт-Петербург 2021

**Лабораторная работа №7**

Разработка приложения с асинхронной очередью сообщений

**Цель работы:**   
Реализовать простую систему распределенной репликации (“писатели-читатели”). Научиться использовать брокер сообщений Apache Kafka совместно со Spring.  
  
**Задание на лабораторную работу:**

1. Скачайте и разверните Apache Kafka.
2. Модифицируйте свое приложение со встраиваемой базой данных так, чтобы его можно было запустить в нескольких экземплярах на разных портах.
3. Реализуйте в рамках своего приложения Producer и Consumer такие, что

А) Producer при каждой операции записи оповещает соответствующий топик

Б) Consumer при получении информации из топика записывает обновление в локальную (встроенную в приложение) базу

1. Продемонстрируйте, что информация, записанная одним приложением, доступна второму приложению

16. Система безопасности предприятия (помещения, люди, права на вход)  
**Ход работы:**

1. Скачаем и развернем Apache Kafka совместно с Zookeeper:

Запустим Zookeeper с помощью команды

$ bin/zookeeper-server-start.sh config/zookeeper.properties

Так же запустим и Kafka, создадим топик и проверим создание

$ bin/kafka-server-start.sh config/server-properties

$ bin/kafka-topics.sh –create –topic test-events –bootstrap-server localhost:9092

$ bin/kafka-topics.sh –describe –topic test-events –bootstrap-server localhost:9092

Создадим новое событие, как “писатель”

$ bin/kafka-console-producer.sh –topic test-events -bootstrap-server localhost:9092

>> This is my first event!

И прочитаем, как “читатель”

$ bin/kafka-console-consumer.sh –topic test-events –from-beginning –bootstrap-server localhost:9092

<< This is my first event!

1. Напишем код для приложения писателя (producer) с использованием Kafka Spring:

(Полный код программы приведен в Приложении 1)

@Service  
public class ProducerService {  
  
 @Autowired  
 private KafkaTemplate<String, EstateEntity> kafkaTemplate;  
  
 public void produce(EstateEntity estate) {  
 kafkaTemplate.send("estatess", estate);  
 }  
}

1. Также, с использованием такого же модуля установим для приложения читателя (consumer) “прослушивание” топика, в котором будут происходить события:

@Service  
public class ConsumerService {  
  
 private final EstateService estateService;  
  
 public ConsumerService(EstateService estateService) {  
 this.estateService = estateService;  
 }  
  
 @KafkaListener(topics = "estatess", groupId = "estates\_group\_id${server.port}")  
 public void consumeEstate(EstateEntity estate){  
 System.*out*.println("Consuming the message: \n" + estate.toString());  
 estateService.addEstate(estate);  
 }  
}

1. Создадим у читателя локальную базу данных (H2) с помощью Hibernate для операций над приходящими сообщениями (чтение, добавление, удаление):

spring.h2.console.enabled=true  
spring.datasource.driverClassName=org.h2.Driver  
spring.datasource.username=sa  
spring.datasource.password=password  
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect  
spring.jpa.hibernate.ddl-auto=update

1. И создадим “копию” читателя на другом порте 8081 и 8082 с помощью двойного запуска приложения (имитация двух читателей). Для каждого нового читателя на новом порте необходимо также создать свою локальную базу данных и соответственно свой group\_id, для чтения сообщений писателя:

#Первое приложение

server.port=8081

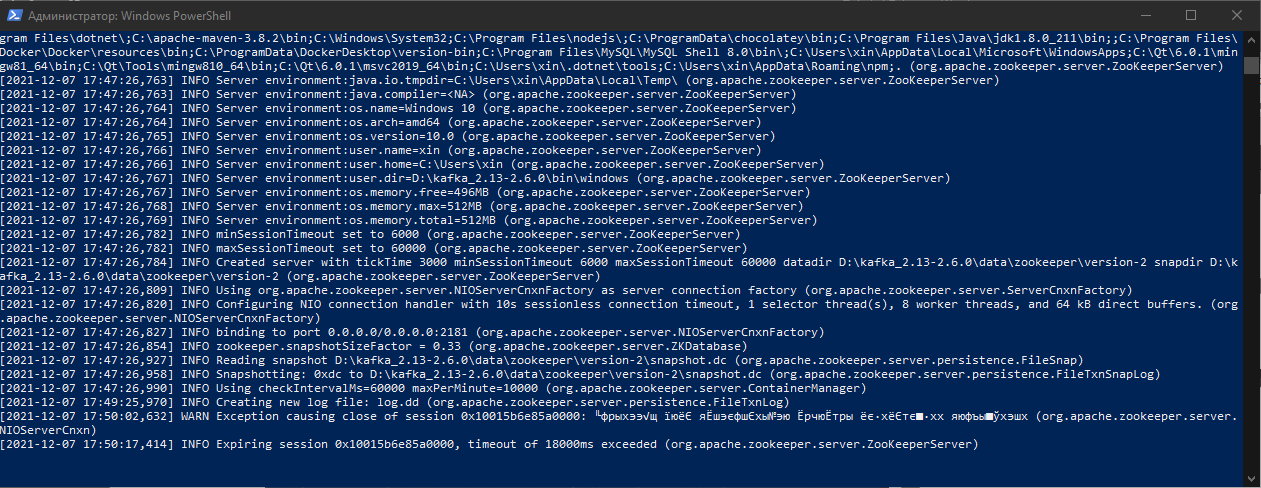
#Второе приложение

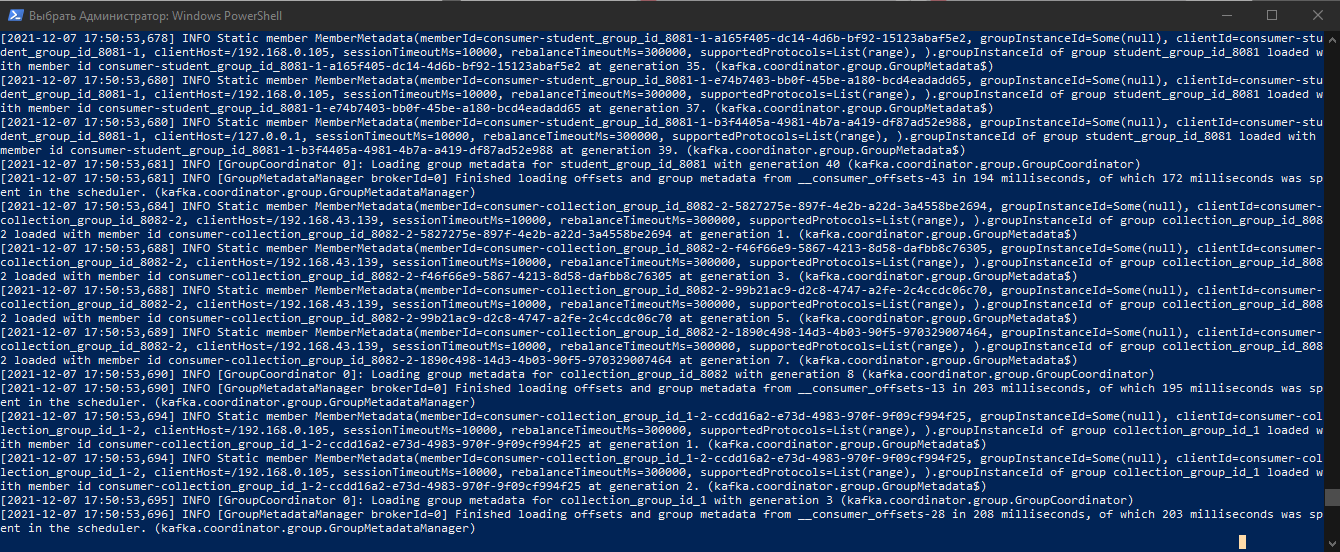
server.port=8082

spring.datasource.url=jdbc:h2:./estatess${server.port}

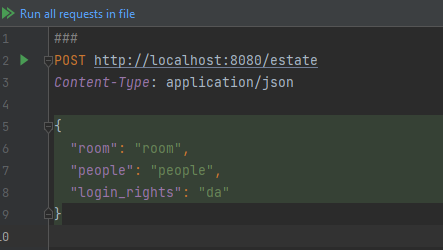
1. Проверим работы программы:

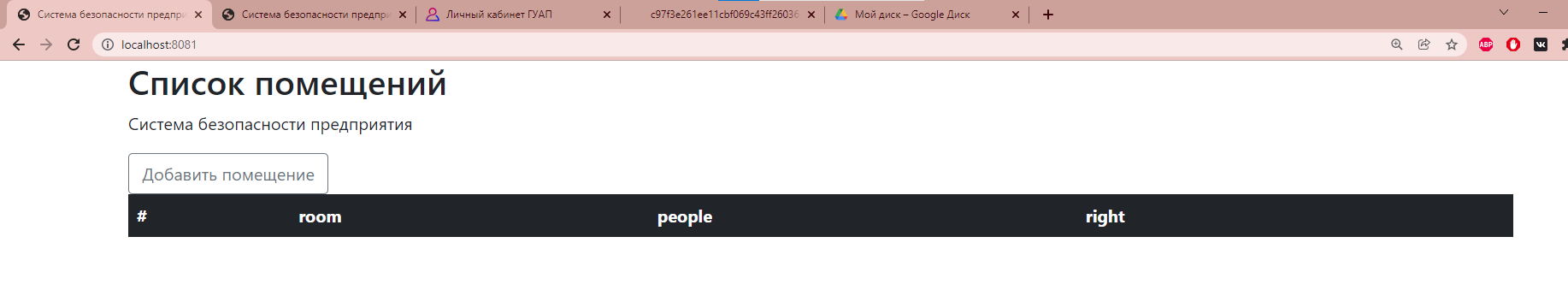
Запустим Kafka и Zookeeper:

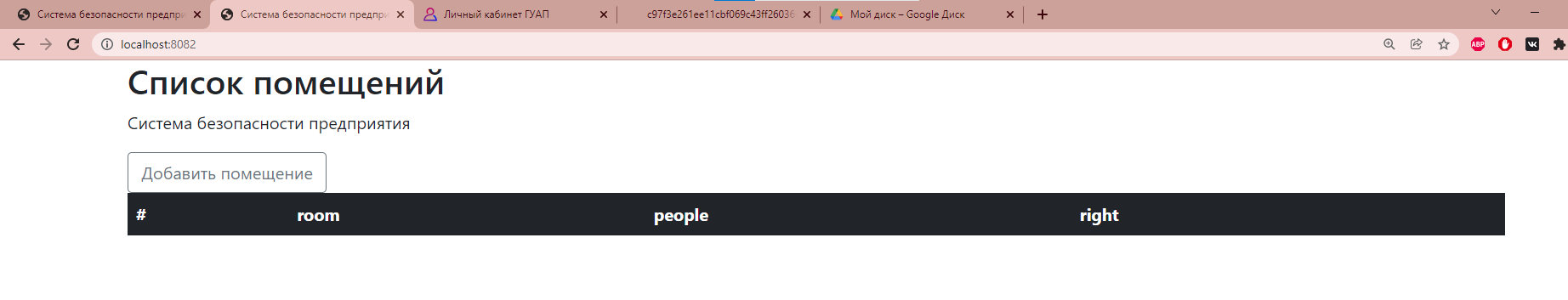




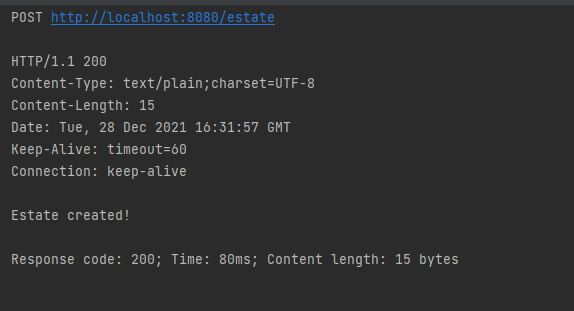
Создадим событие в писателе (с помощью POST запроса):

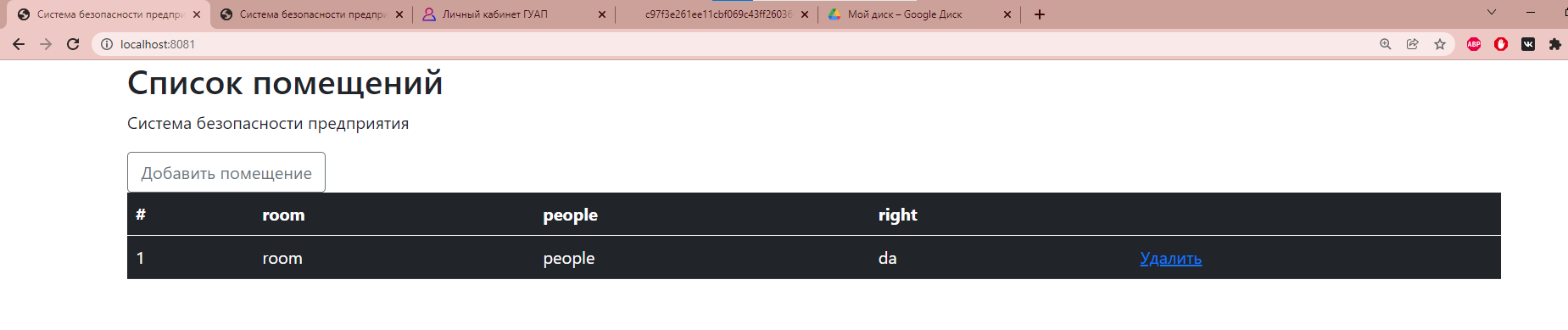


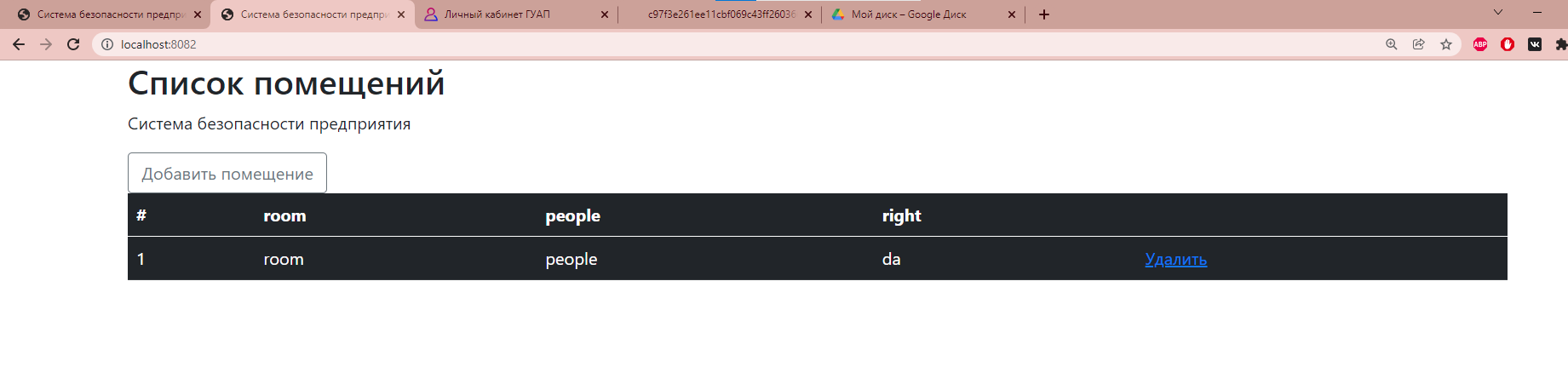
Читатели получили сообщение из прослушиваемого топика:  
До:  




После:







**Вывод:** в ходе лабораторной работы были получены основные навыки работы асинхронный очередью сообщений и соответственно брокером сообщений Apache Kafka, изучены основы взаимодействия “читатели-писатели”, закреплены основы использования Hibernate и Spring Boot.

**Приложение 1.**