

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО
ПРИБОРОСТРОЕНИЯ»

ИНСТИТУТ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И ПРОГРАММИРОВАНИЯ

КАФЕДРА компьютерных технологий и программной инженерии

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

ассистент

должность, уч. степень, звание

подпись, дата

Кочин Д.А.

инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №7

«Разработка приложения с асинхронной
очередью сообщений»

по курсу: Технологии разработки серверных информационных систем

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №

4932

18.12.21

Н.С. Иванов

подпись, дата

инициалы, фамилия

Санкт-Петербург 2021

Цель работы

Целью работы является реализация простой системы распределенной репликации (“писатели-читатели”).

Задание:

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
 - a. Producer при каждой операции записи оповещает соответствующий топик
 - b. Consumer при получении информации из топика записывает обновление в локальную (встроенную в приложение) базу
4. Продемонстрируйте, что информация, записанная одним приложением, доступна второму приложению.

Вариант: 8. Учет трат в бюджете семьи.

Описание разрабатываемого продукта:

KafkaConsumer

```
@Service
public class KafkaConsumer {
    @Autowired
    CostRepository costRepository;

    @KafkaListener(topics = "ADD-EVENT", groupId = "group0")
    public void consume(ConsumerRecord<String,String> message) throws IOException {
        ObjectMapper mapper = new ObjectMapper();
        CostEntity authorModel = mapper.readValue(message.value(),
        CostEntity.class);
        costRepository.save(authorModel);
    }
}
```

KafkaProducer

```
@Service
public class KafkaProducer {
    private static final String TOPIC = "ADD-EVENT";

    @Autowired
    private KafkaTemplate<String, String> kafkaTemplate;

    public void sendMessage(CostEntity cost) throws JsonProcessingException {
        ObjectMapper mapper = new ObjectMapper();
    }
```

```

        String message = mapper.writeValueAsString(cost);
        this.kafkaTemplate.send(TOPIC, message);
    }
}

```

KafkaController

```

@RestController
public class KafkaController {
    @Autowired
    KafkaProducer producer;

    @Autowired
    KafkaConsumer consumer;

    @PostMapping("/kafka-add")
    public String sendMessage(@RequestBody CostEntity cost) throws JsonProcessingException {
        this.producer.sendMessage(cost);
        return "Successfully";
    }
}

```

application.yml

```

kafka:
  consumer:
    bootstrap-servers: localhost:9092
    group-id: 1
    auto-offset-reset: earliest
    key-deserializer: org.apache.kafka.common.serialization.StringDeserializer
    value-deserializer: org.apache.kafka.common.serialization.StringDeserializer
  producer:
    bootstrap-servers: localhost:9092
    key-serializer: org.apache.kafka.common.serialization.StringSerializer
    value-serializer: org.apache.kafka.common.serialization.StringSerializer

sql:
  init:
    platform: MySQLDialect

```

Request

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:8080/kafka-add
- Environment:** No Environment
- Body:**

```
{
  "name": "test",
  "price": 100,
  "category": "test",
  "customer": "test"
}
```
- Response:** 200 OK, 89 ms, 358 B
- Status:** Successfully

Выводы

- Кафка хорошо выполняет возложенную на него роль балансировщика нагрузки.