ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| кандидат технических наук |  |  |  | П.А. Степанов |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №7  Разработка приложения с асинхронной очередью сообщений |
| по дисциплине: Технология разработки серверных информационных систем |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 4134к |  |  |  | Костяков Н.А. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург

2023

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.

package io.swagger.kafka;  
  
import com.fasterxml.jackson.databind.ObjectMapper;  
import io.swagger.db.TimeRepository;  
import io.swagger.model.Time;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.context.annotation.Profile;  
import org.springframework.kafka.annotation.KafkaListener;  
import org.springframework.stereotype.Service;  
import io.swagger.service.TimeServiceImpl;  
import io.swagger.db.TimeRepository;  
  
@Service  
// @Profile("consumer")  
public class MyKafkaConsumer {  
 private final TimeRepository repository;  
 private final ObjectMapper objectMapper = new ObjectMapper();  
  
 @Autowired  
 public MyKafkaConsumer(TimeRepository repository) {  
 this.repository = repository;  
 System.*out*.println("MyKafkaConsumer");  
 }  
 @KafkaListener(topics = "myTopic")  
 public void listen(String message) {  
 System.*out*.println(message);  
 Time itemObj = convertMessageToItemObj(message);  
 repository.save(itemObj);  
 }  
 private Time convertMessageToItemObj(String message) {  
 try {  
 return objectMapper.readValue(message, Time.class);  
 } catch (Exception e) {  
 throw new RuntimeException("Ошибка при преобразовании сообщения", e);  
 }  
 }  
}

package io.swagger.kafka;  
import com.fasterxml.jackson.databind.ObjectMapper;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.kafka.core.KafkaTemplate;  
import org.springframework.stereotype.Service;  
  
@Service  
public class MyKafkaProducer {  
 private final KafkaTemplate<String, String> kafkaTemplate;  
 @Autowired  
 public MyKafkaProducer(KafkaTemplate<String, String> kafkaTemplate) {  
 this.kafkaTemplate = kafkaTemplate;  
 System.*out*.println("MyKafkaProducer");  
 }  
 public void sendMessage(String topic, String key, String message) {  
 System.*out*.println("SEND smth");  
 System.*out*.println(message);  
 kafkaTemplate.send(topic, key, message);  
 }  
}

(value = "/{name}/{type}", method = RequestMethod.*POST*)  
public ResponseEntity<Time> add(  
 @PathVariable("name") String name,  
 @PathVariable("type") Integer type,  
 Principal principal) throws Exception {  
 if (principal == null) {  
 throw new ForbiddenException();  
 }  
  
 return ResponseEntity.*ok*(tService.add(0, name, type));  
}