ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| доцент, к.т.н. |  |  |  | П.А. Степанов |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

­­

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| РАЗРАБОТКА AJAX ПРИЛОЖЕНИЯ С АСИНХРОННОЙ ОЧЕРЕДЬЮ СООБЩЕНИЙ |
| по дисциплине: ТЕХНОЛОГИИ РАЗРАБОТКИ СЕРВЕРНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4032 |  |  |  | М.В. Мухортов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

**Задание**

1. Подготовьте Ваше приложение к разворачиванию в облачном сервисе или компоненте Docker.
2. Реализуйте файл docker-compose.xml, который будет содержать все необходимые для работы Вашего приложения ресурсы

Вариант7: Ведение списка литературы согласно последнему актуальному ГОСТу

**Описание разрабатываемого продукта**

В ходе выполнения работы необходимо разработать серверное приложение с формой логина для выполнения операций записи и удаления литературы в списке литературы, а также с использованием базы данных. Необходимо развернуть два экземпляра приложения в Docker, а также реализовать синхронизацию обеих баз данных.

**Текст основных фрагментов кода**

**literature.java**

@Entity

@Table(name = "Literature")

@Data

public class Literature {

public Literature() {

}

public Literature(Integer nomer, String name) {

this.number = nomer;

this.name = name;

}

@Id

@Column(name = "literature\_id")

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Integer id;

@Column(name = "literature\_number", unique = true)

private Integer number;

@Column(name = "literature\_name")

private String name;

}

**KafkaConsumerListeners.java**

@Service

public class KafkaConsumerListeners {

@Autowired

LiteratureRepository literatureRepository;

@Transactional

@KafkaListener(topics = "${kafka.topic}", groupId = "${kafka.groupid}")

public void debeziumListener1(ConsumerRecord<org.apache.avro.generic.GenericData.Record, org.apache.avro.generic.GenericData.Record> record) {

if (record.value() == null) {

return;

}

String op = ((org.apache.avro.util.Utf8) record.value().get("op")).toString();

if (op.equals("d")) {

Integer idToDelete = (Integer) record.key().get("literature\_id");

try {

literatureRepository.deleteById(idToDelete);

} catch (org.springframework.dao.EmptyResultDataAccessException e) { }

return;

}

org.apache.avro.generic.GenericData.Record value = (org.apache.avro.generic.GenericData.Record) record.value().get("after");

Integer id = (Integer) value.get("literature\_id");

String name = ((org.apache.avro.util.Utf8) value.get("literature\_name")).toString();

Integer number = (Integer) value.get("literature\_number");

Literature literature = literatureRepository.findById(id).orElse(null);

if (literature == null) {

literatureRepository.save(new Literature(id,number,name));

} else {

literature.setName(name);

literature.setNumber(number);

literatureRepository.save(literature);

}

}

}

**docker-compose.yaml**

instance1:

build:

context: lab

links:

- kafka

- postgres1

depends\_on:

- kafka

- postgres1

ports:

- 9443:8443

environment:

- SPRING\_DATASOURCE\_URL=jdbc:postgresql://postgres1:5432/inst1

- SPRING\_DATASOURCE\_DRIVERCLASSNAME=org.hibernate.dialect.PostgreSQLDialect

- SPRING\_DATASOURCE\_USERNAME=postgres

- SPRING\_DATASOURCE\_PASSWORD=inst1pass

instance2:

build:

context: lab

links:

- kafka

- postgres2

depends\_on:

- kafka

- postgres2

ports:

- 10443:8443

environment:

- SPRING\_DATASOURCE\_URL=jdbc:postgresql://postgres2:5432/inst2

- SPRING\_DATASOURCE\_DRIVERCLASSNAME=org.hibernate.dialect.PostgreSQLDialect

- SPRING\_DATASOURCE\_USERNAME=postgres

- SPRING\_DATASOURCE\_PASSWORD=inst2pass