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 AJAX приложения с использованием Hibernate |
| по дисциплине: Технология разработки серверных информационных систем |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 4134к |  |  |  | Самарин Д. В. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург

2023

**Целью работы является реализация простой системы распределенной репликации**

**(“писатели-читатели”).**

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.

Или альтернативное задание – настройка синхронизации средствами CDC –Change Data

Capture (для владельцев мощных ноутбуков)

1 Скачайте и разверните Apache Kafka

2 Модифицируйте свое приложение так, чтобы оно работало с базой данных Postgres

либо другой, поддерживаемой Kafka Connect. Сделайте так, чтобы на Вашем

компьютере можно было запустить две базы данных и два разных экземпляра

приложения (на разных портах)

3 Используя, например, Debezium, настройте канал из обеих баз данных в Kafka.

Реализуйте консьюмеры так, чтобы изменения в каждой из баз отображались в обеих

базах. Избегайте зацикливания обновлений.

4 Продемонстрируйте, что информация, записанная одним приложением, доступна

второму приложению и наоборот.

Вариант: безопасность предприятия.

package io.swagger.model;  
  
import io.swagger.models.auth.In;  
import lombok.Getter;  
  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.persistence.Table;  
  
@Entity  
@Table(name = "Times")  
public class Time {  
 public Time(Integer id, String name, Integer type){  
 this.id = id;  
 this.name = name;  
 this.type = type;  
 }  
 @Id  
 @GeneratedValue (strategy = GenerationType.*IDENTITY*)  
 @Column(name = "id")  
 public Integer id;  
  
 public Time() {  
  
 }  
  
 public void setId(Integer id) {  
 this.id = id;  
 }  
 public Integer GetId(){  
 return id;  
 }  
  
  
 @Getter  
 @Column(name = "name")  
 private String name;  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 @Getter  
 @Column(name = "type")  
 private Integer type;  
  
 public void setType(Integer type) {  
 this.type = type;  
 }  
}

package io.swagger.db;  
  
import io.swagger.model.Time;  
import java.util.List;  
import org.springframework.data.repository.CrudRepository;  
public interface TimeRepository  
 extends CrudRepository<Time, Integer> {  
 List<Time> findByType(Integer type);  
}

package io.swagger.service;  
  
import io.swagger.model.Time;  
  
public interface TimeService {  
 Iterable<Time> findByType(Integer type);  
  
 Iterable<Time> listAll();  
  
 void delete(Integer id);  
  
 Time add(Integer id, String name, Integer Type);  
  
 Time getById(Integer id);  
  
  
}

package io.swagger.service;  
  
import io.swagger.db.TimeRepository;  
import org.apache.log4j.Logger;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import io.swagger.model.Time;  
  
@Service  
public class TimeServiceImpl implements TimeService {  
 private final static Logger *log* = Logger.*getLogger*(TimeServiceImpl.class);  
  
 @Autowired  
 private TimeRepository tRep;  
  
 @Override  
 public Iterable<Time> findByType(Integer type){  
 return tRep.findByType(type);  
 }  
  
 @Override  
 public Iterable<Time> listAll() {  
 return tRep.findAll();  
 }  
  
 @Override  
 public void delete(Integer id) {  
 try {  
 tRep.deleteById(id);  
 } catch (org.springframework.dao.EmptyResultDataAccessException e) {  
 }  
 }  
  
 @Override  
 public Time add(Integer id, String name, Integer type) {  
 return tRep.save(new Time(id, name, type));  
 }  
  
 @Override  
 public Time getById(Integer id) {  
 return tRep.findById(id).orElse(null);  
 }  
  
}

package io.swagger.service;  
  
import io.swagger.model.Time;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestMethod;  
import org.springframework.web.bind.annotation.RestController;  
import java.security.Principal;  
import io.swagger.api.ForbiddenException;  
import io.swagger.service.TimeService;  
import lombok.extern.slf4j.Slf4j;  
  
@Slf4j  
@RestController  
@RequestMapping("public/rest/time")  
public class timeRestService {  
  
 @Autowired  
 private TimeService tService;  
  
 @RequestMapping(value = "", method = RequestMethod.*GET*)  
 public ResponseEntity <Iterable<Time>> browse(Principal principal) {  
 return ResponseEntity.*ok*(tService.listAll());  
 }  
  
 @RequestMapping(value = "/{id}", method = RequestMethod.*DELETE*)  
 public void delete(@PathVariable("id") Integer id, Principal principal) {  
 if (principal == null) {  
 throw new ForbiddenException();  
 }  
  
 tService.delete(id);  
 }  
  
 @RequestMapping(value = "/{id}", method = RequestMethod.*GET*)  
 public ResponseEntity<Time> getOne(@PathVariable("id") Integer id, Principal principal) {  
 if (principal == null) {  
 throw new ForbiddenException();  
 }  
  
 Time TimeById = tService.getById(id);  
 if (TimeById == null) {  
 return ResponseEntity.*notFound*().build();  
 } else {  
 return ResponseEntity.*ok*(TimeById);  
 }  
 }  
  
 @RequestMapping(value = "/{name}/{type}", method = RequestMethod.*POST*)  
 public ResponseEntity<Time> add(  
 @PathVariable("name") String name,  
 @PathVariable("type") Integer type,  
 Principal principal) {  
 if (principal == null) {  
 throw new ForbiddenException();  
 }  
  
 return ResponseEntity.*ok*(tService.add(null, name, type));  
 }  
  
 @RequestMapping(value = "/search/{type}", method = RequestMethod.*GET*)  
 public ResponseEntity<Iterable<Time>> search(@PathVariable("type") Integer type, Principal principal) {  
 if (principal == null) {  
 throw new ForbiddenException();  
 }  
  
 Iterable<Time> Times = tService.findByType(type);  
 return ResponseEntity.*ok*(Times);  
 }  
}