Spring Data Redis
как неудачное решение

О себе

Telegram: @Skrynfor

- В IT более 7 лет
- Из них 3 года в Axenix
- Использую Spring

О себе

Telegram: @Skrynfor

- В IT более 7 лет
- Из них 3 года в Axenix
- Использую Spring
- Катаю на сноуборде

О себе

Telegram: @Skrynfor

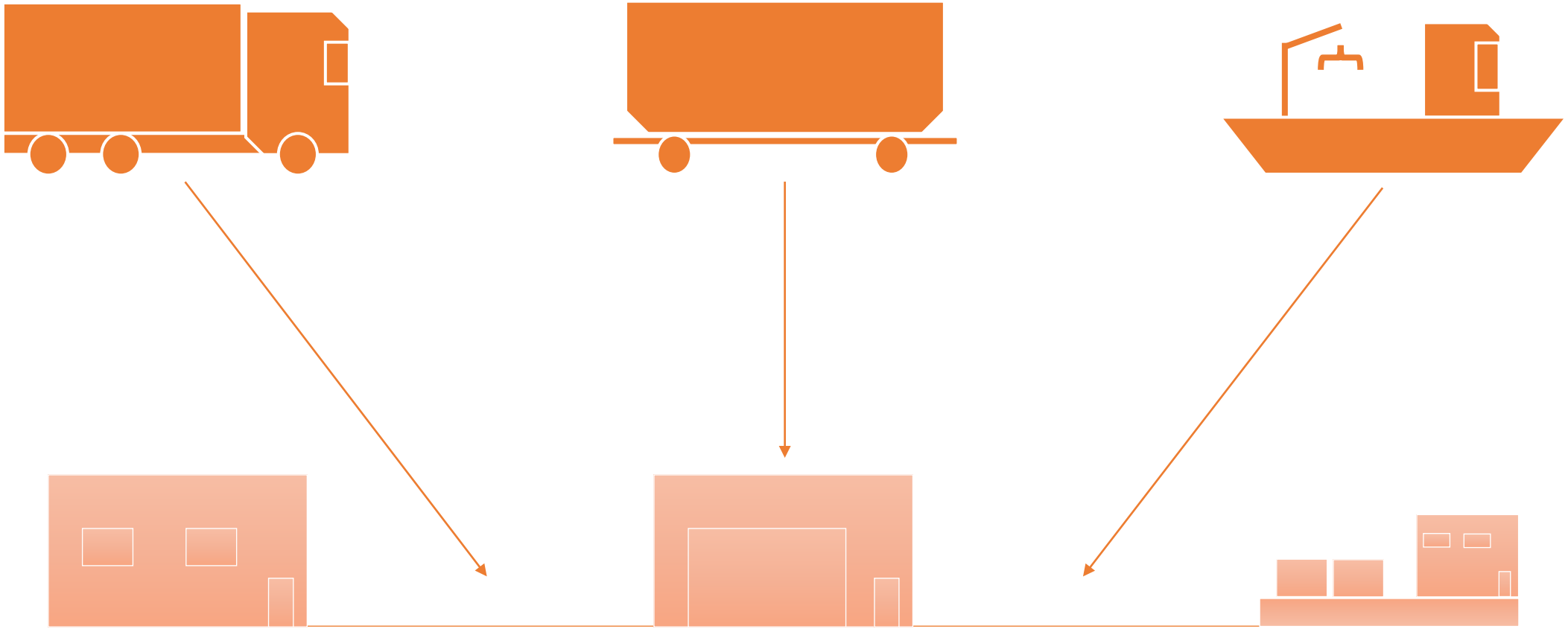
- В IT более 7 лет
- Из них 3 года в Axenix
- Использую Spring
- Катаю на сноуборде
- Люблю собак

О себе

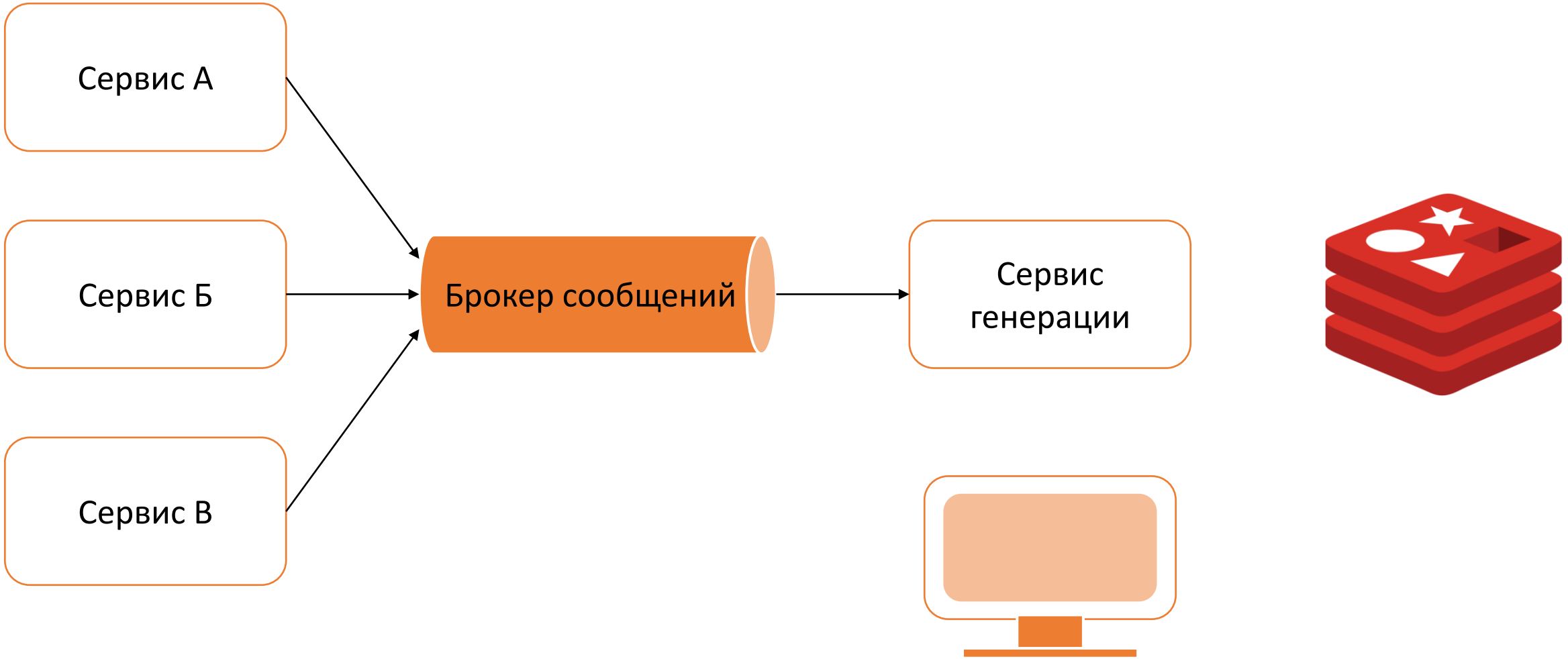
Telegram: @Skrynfor

- В IT более 7 лет
- Из них 3 года в Axenix
- Использую Spring
- Катаю на сноуборде
- Люблю собак
- И кошек тоже

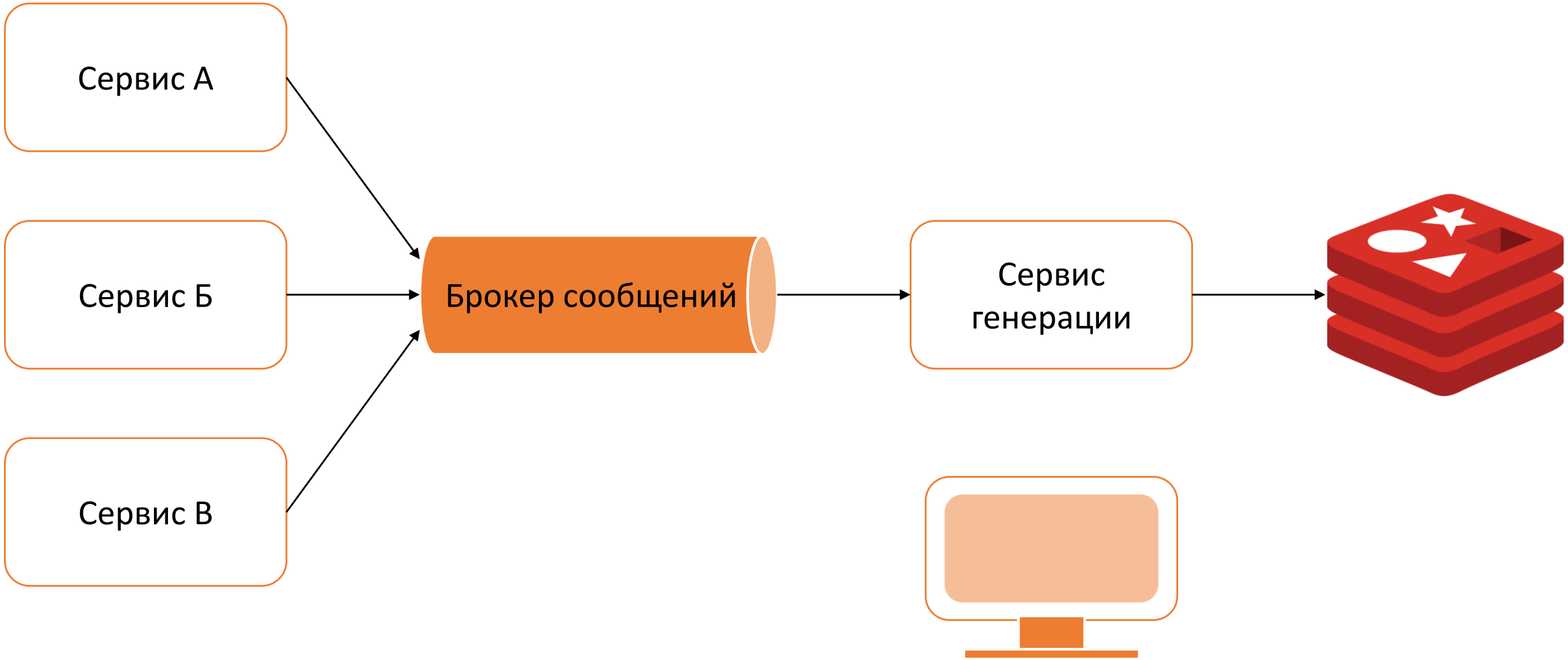
С чего все начиналось



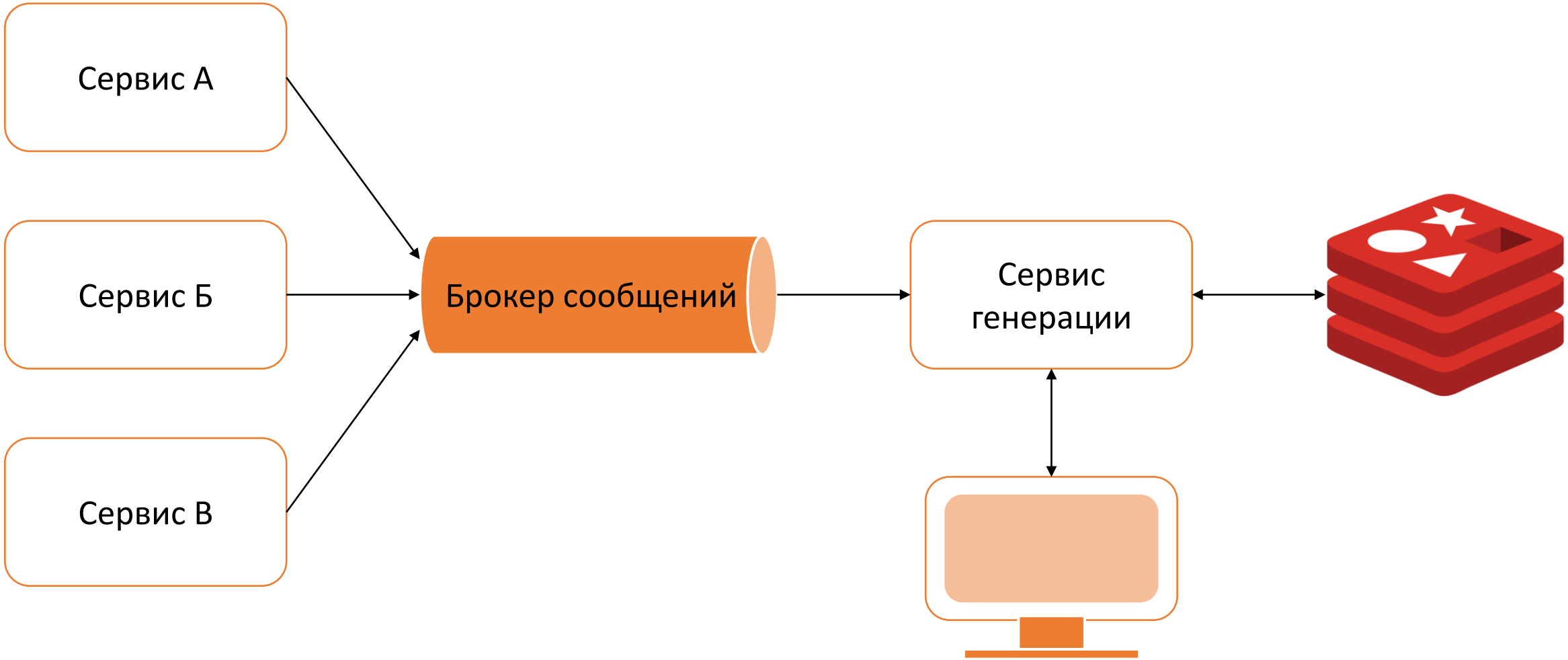
С чего все начиналось



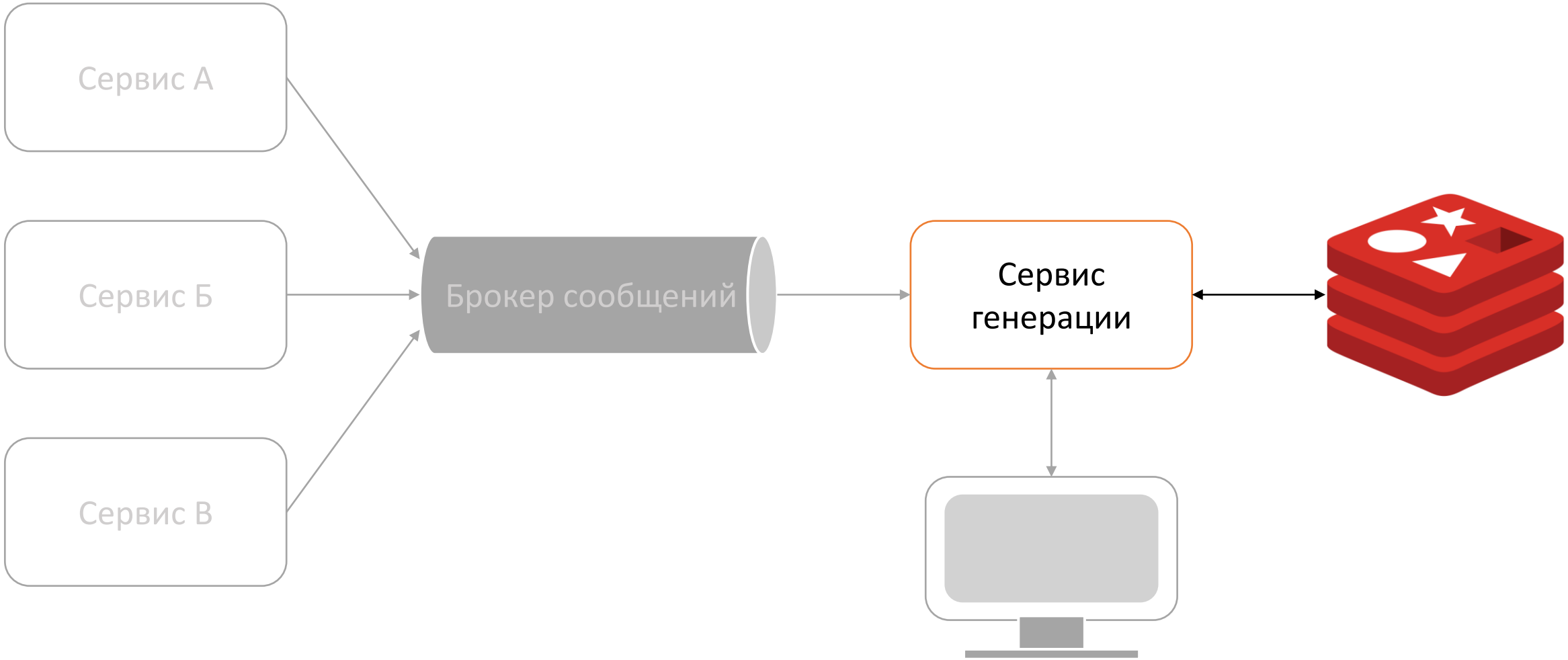
С чего все начиналось



С чего все начиналось



С чего все начиналось



Выбор решения

Jedis

Выбор решения

Jedis

Lettuce

Выбор решения

Jedis

Lettuce

Redisson

Выбор решения

Jedis

Lettuce

Redisson

RedisTemplate

Выбор решения

Jedis

Lettuce

Redisson

RedisTemplate

Spring Data

Выбор решения

Jedis

Lettuce

Redisson

RedisTemplate

Spring Data

Spring Data Redis

```
implementation ‘org.springframework.boot:spring-boot-starter-data-redis’
```

```
spring.data.redis.host=redis
spring.data.redis.port=6379
```

```
public interface ComponentRepository extends KeyValueRepository<Component, Long> {
}
```

Spring Data Redis

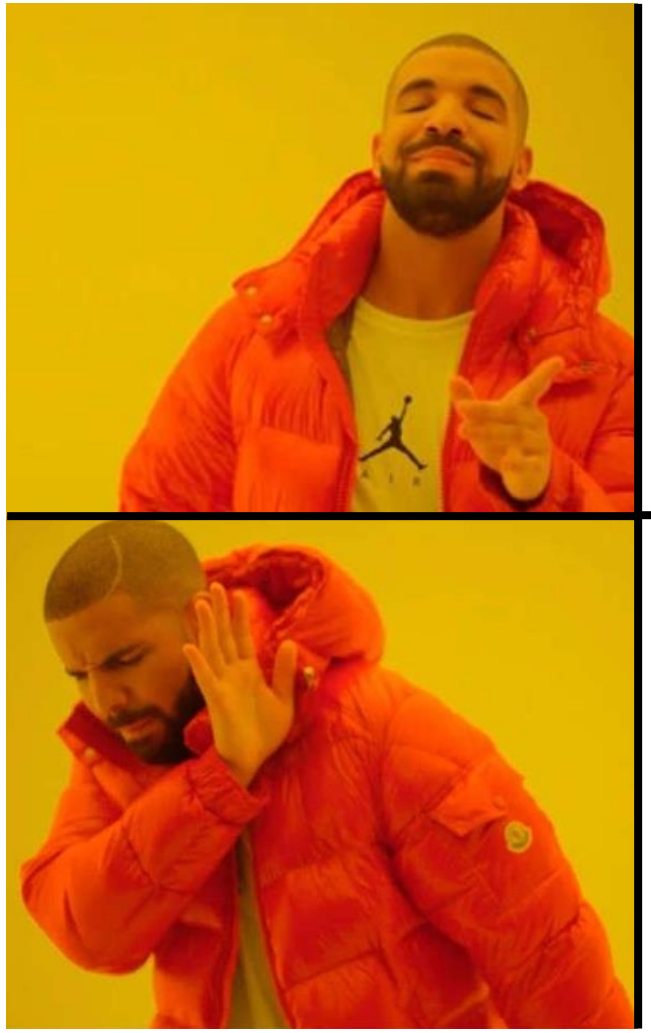
```
implementation ‘org.springframework.boot:spring-boot-starter-data-redis’
```

```
spring.data.redis.host=redis
spring.data.redis.port=6379
```

```
public interface ComponentRepository extends KeyValueRepository<Component, Long> {
}
```

```
public interface KeyValueRepository<T, ID> extends ListCrudRepository<T, ID>,
        ListPagingAndSortingRepository<T, ID> {
}
```

Проблемы



До знакомства с
Spring Data Redis

После знакомства с
Spring Data Redis

План доклада

Структура хранения

План доклада

Структура хранения

Индексы

План доклада

Структура хранения

Индексы

Пользовательские запросы

План доклада

Структура хранения

Индексы

Пользовательские запросы

Обходные решения

План доклада

Структура хранения

Индексы

Пользовательские запросы

Обходные решения

Кастомная реализация

План доклада

Структура хранения

Индексы

Пользовательские запросы

Обходные решения

Кастомная реализация

Итог проекта

Команды Redis

Команды Redis

```
SADD key element
```

Команды Redis

```
SADD key element
SREM key element
```

Команды Redis

```
SADD key element
SREM key element
SMEMBERS key
```

Команды Redis

```
SADD key element
SREM key element
SMEMBERS key
SINTER key1 key2
```

Команды Redis

```
SADD key element
SREM key element
SMEMBERS key
SINTER key1 key2
HMSET key
      hash-key1 hash-value1
      hash-key2 hash-value2
```

Команды Redis

```
SADD key element
SREM key element
SMEMBERS key
SINTER key1 key2
HMSET key hash-key1 hash-value1 hash-key2 hash-value2
HGETALL key
```

Команды Redis

```
SADD key element
SREM key element
SMEMBERS key
SINTER key1 key2
HMSET key hash-key1 hash-value1 hash-key2 hash-value2
HGETALL key
HGET key hash-key
```

Команды Redis

```
SADD key element
SREM key element
SMEMBERS key
SINTER key1 key2
HMSET key hash-key1 hash-value1 hash-key2 hash-value2
HGETALL key
HGET key hash-key
HDEL key hash-key
```

Команды Redis

```
SADD key element
SREM key element
SMEMBERS key
SINTER key1 key2
HMSET key hash-key1 hash-value1 hash-key2 hash-value2
HGETALL key
HGET key hash-key
HDEL key hash-key
TYPE key
```

Команды Redis

```
SADD key element
SREM key element
SMEMBERS key
SINTER key1 key2
HMSET key hash-key1 hash-value1 hash-key2 hash-value2
HGETALL key
HGET key hash-key
HDEL key hash-key
TYPE key
KEYS ke*
```

Команды Redis

```
SADD key element
SREM key element
SMEMBERS key
SINTER key1 key2
HMSET key hash-key1 hash-value1 hash-key2 hash-value2
HGETALL key
HGET key hash-key
HDEL key hash-key
TYPE key
KEYS ke*
DEL key
```

Структура хранения

Индексы

Пользовательские запросы

Обходные решения

Кастомная реализация

Итог проекта

Модель компонента

```
import org.springframework.data.annotation.Id;
import org.springframework.data.redis.core.RedisHash;

@RedisHash(“component”)
public class Component {

    @Id
    private Long id;

    private Integer number;

    private ComponentType type;

    private Zone zone;

    private ControlPoint scanPoint;

    private ZonedDateTime scannedAt;

}
```

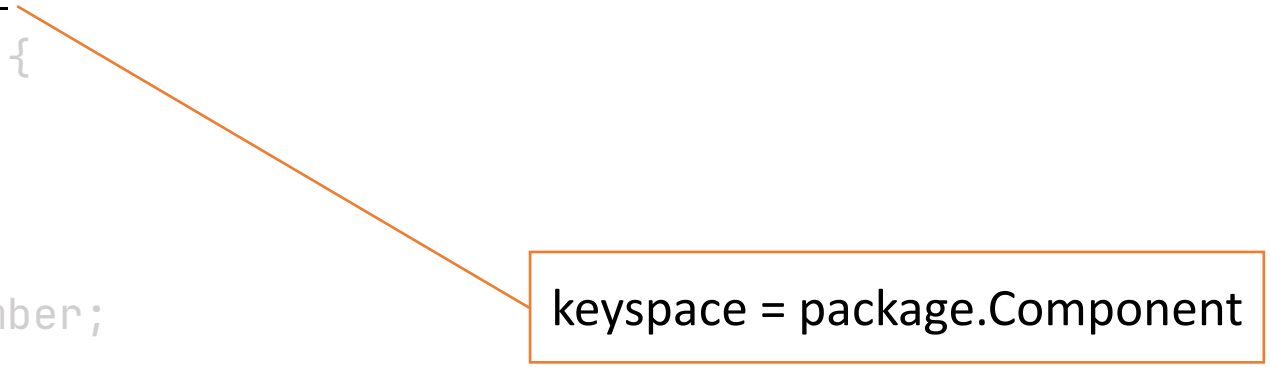
@RedisHash

```java
import org.springframework.data.annotation.Id;
import org.springframework.data.redis.core.RedisHash;

@RedisHash("component")
public class Component {

    @Id
    private Long id;

    private Integer number;

    private ComponentType type;

    private Zone zone;

    private ControlPoint scanPoint;

    private ZonedDateTime scannedAt;

}
```

keyspace = component

@RedisHash

```
import org.springframework.data.annotation.Id;
import org.springframework.data.redis.core.RedisHash;

@RedisHash("component")
public class Component {

    @Id
    private Long id;

    private Integer number;

    private ComponentType type;

    private Zone zone;

    private ControlPoint scanPoint;

    private ZonedDateTime scannedAt;

}
```



@Id

```
import org.springframework.data.annotation.Id;
import org.springframework.data.redis.core.RedisHash;

@RedisHash(“component”)
public class Component {

    @Id
    private Long id;

    private Integer number;

    private ComponentType type;

    private Zone zone;

    private ControlPoint scanPoint;

    private ZonedDateTime scannedAt;

}
```

@Id

```java
import org.springframework.data.annotation.Id;
import org.springframework.data.redis.core.RedisHash;

@RedisHash(“component”)
public class Component {

    @Id
    private Long identifier;

    private Integer number;

    private ComponentType type;

    private Zone zone;

    private ControlPoint scanPoint;

    private ZonedDateTime scannedAt;

}
```

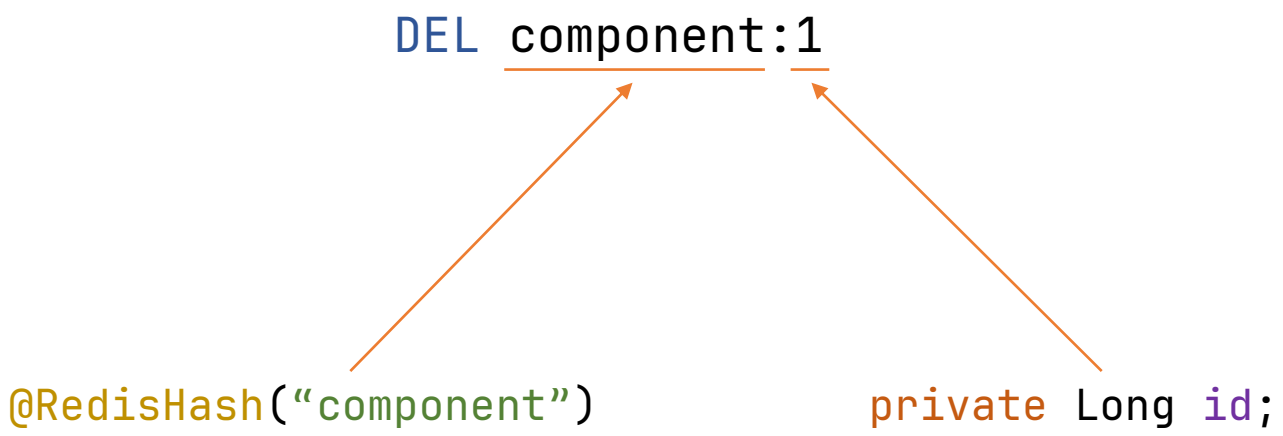
componentRepository.save

```
{
    “id”: 1,
    “number”: 100,
    “type”: “T_001”,
    “zone”: “Z_001”,
    “scanPoint”: “P_01”,
    “scannedAt”: “2024-10-15T12:59:59.999Z”
}
```

componentRepository.save

```
DEL component:1
```

componentRepository.save



componentRepository.save

```
DEL component:1
HMSET component:1
      id            1
      number        100
      type          T_001
      zone          Z_001
      scanPoint     P_01
      scannedAt     2024-10-15T12:59:59.999Z
      _class        package.Component
```

componentRepository.save

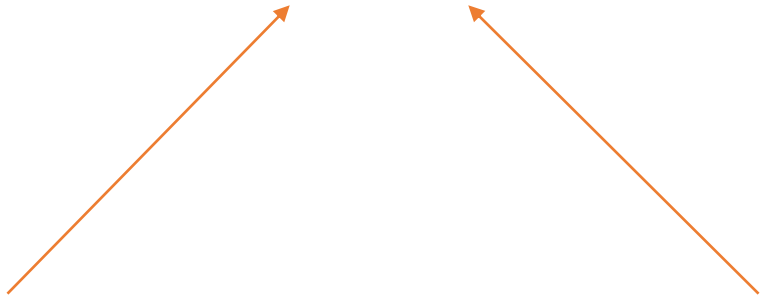
```
DEL component:1
HMSET component:1
      id            1
      number        100
      type          T_001
      zone          Z_001
      scanPoint     P_01
      scannedAt     2024-10-15T12:59:59.999Z
      _class        package.Component
```

componentRepository.save

```
DEL component:1
HMSET component:1
      id           1
      number       100
      type         T_001
      zone         Z_001
      scanPoint    P_01
      scannedAt    2024-10-15T12:59:59.999Z
      _class       package.Component
SADD component 1
```

componentRepository.save

```
DEL component:1
HMSET component:1
      id            1
      number        100
      type          T_001
      zone          Z_001
      scanPoint     P_01
      scannedAt     2024-10-15T12:59:59.999Z
      _class        package.Component
SADD component 1
```



```
@RedisHash("component")          private Long id;
```

Структура хранения

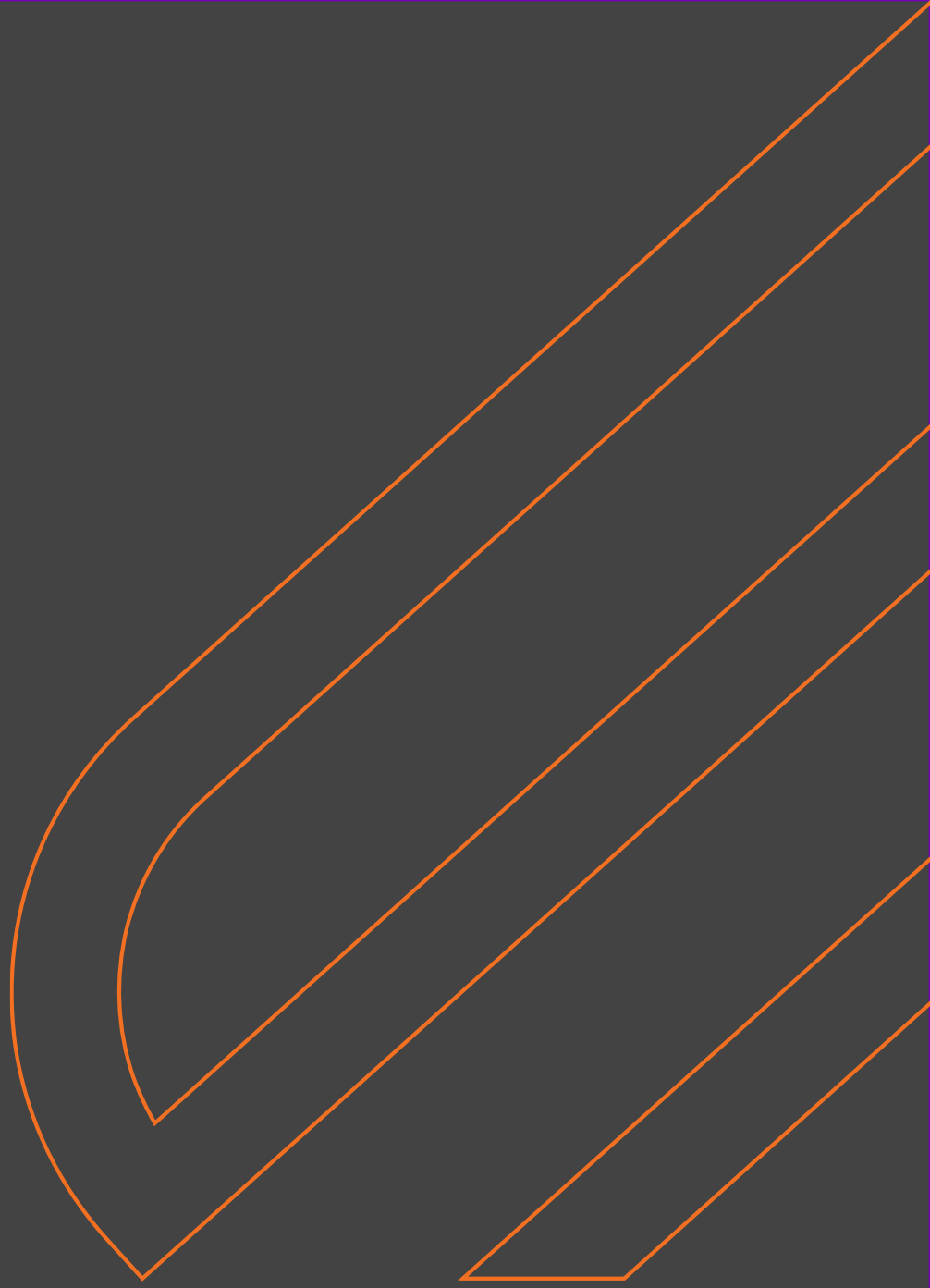
Индексы

Пользовательские запросы

Обходные решения

Кастомная реализация

Итог проекта



componentRepository.findByTypeAndZone(T_001, Z_001)

```
@RedisHash(“component”)
public class Component {

    private Long id;

    private Integer number;

    private ComponentType type;

    private Zone zone;

    private ControlPoint scanPoint;

    private ZonedDateTime scannedAt;

}
```

- ○ 10
- ○ 0
- ○ Ошибка

componentRepository.findByTypeAndZone(T_001, Z_001)

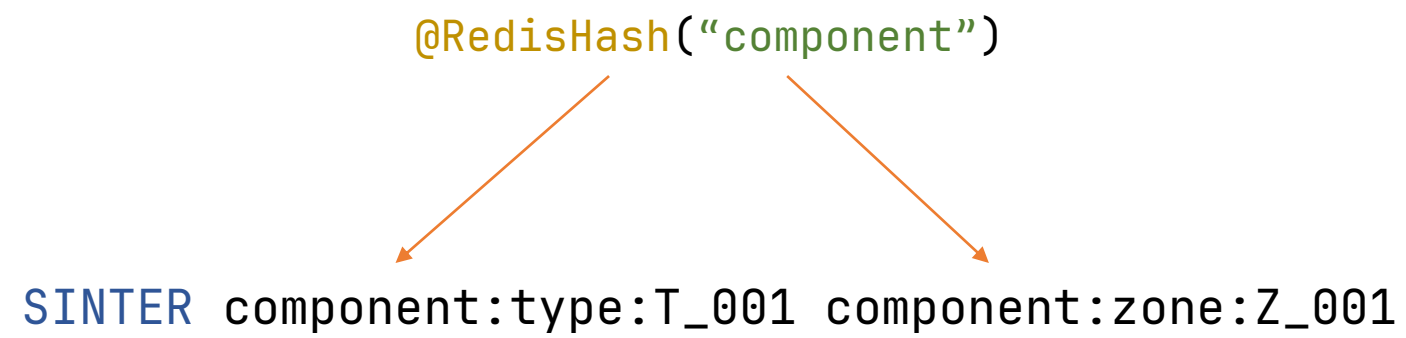
```
@RedisHash(“component”)
public class Component {

    private Long id;

    private Integer number;

    private ComponentType type;

    private Zone zone;

    private ControlPoint scanPoint;

    private ZonedDateTime scannedAt;

}
```

- ○ 10
- ● 0
- ○ Ошибка

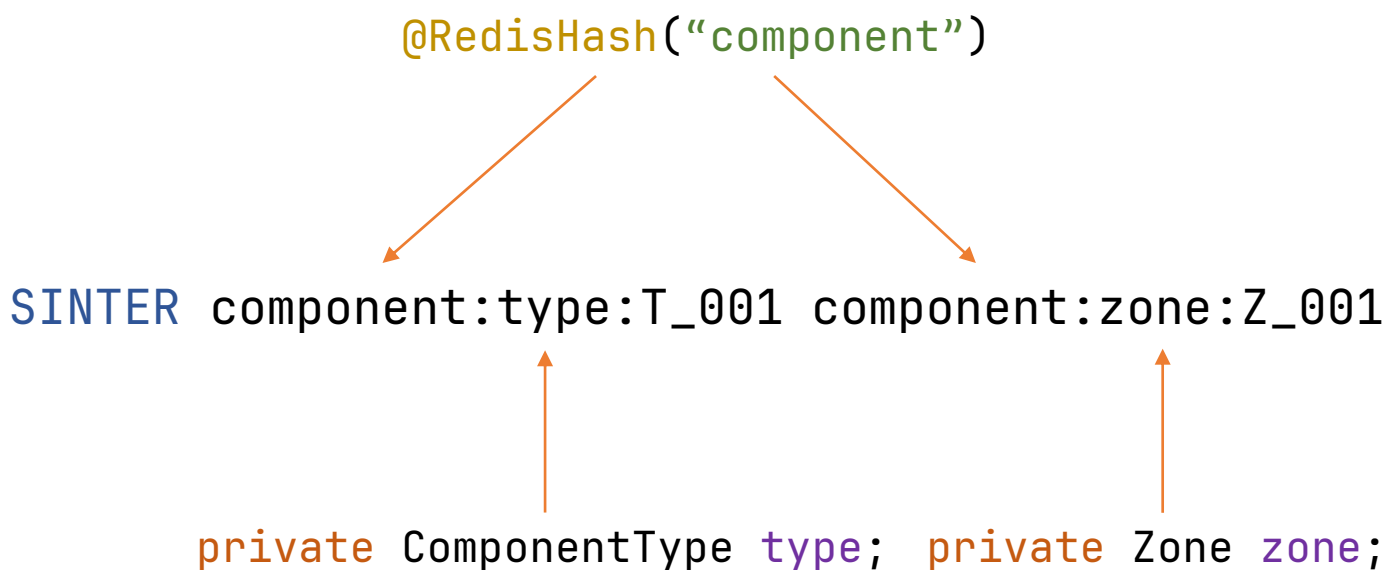
componentRepository.findByTypeAndZone(T_001, Z_001)

```
SINTER component:type:T_001 component:zone:Z_001
```

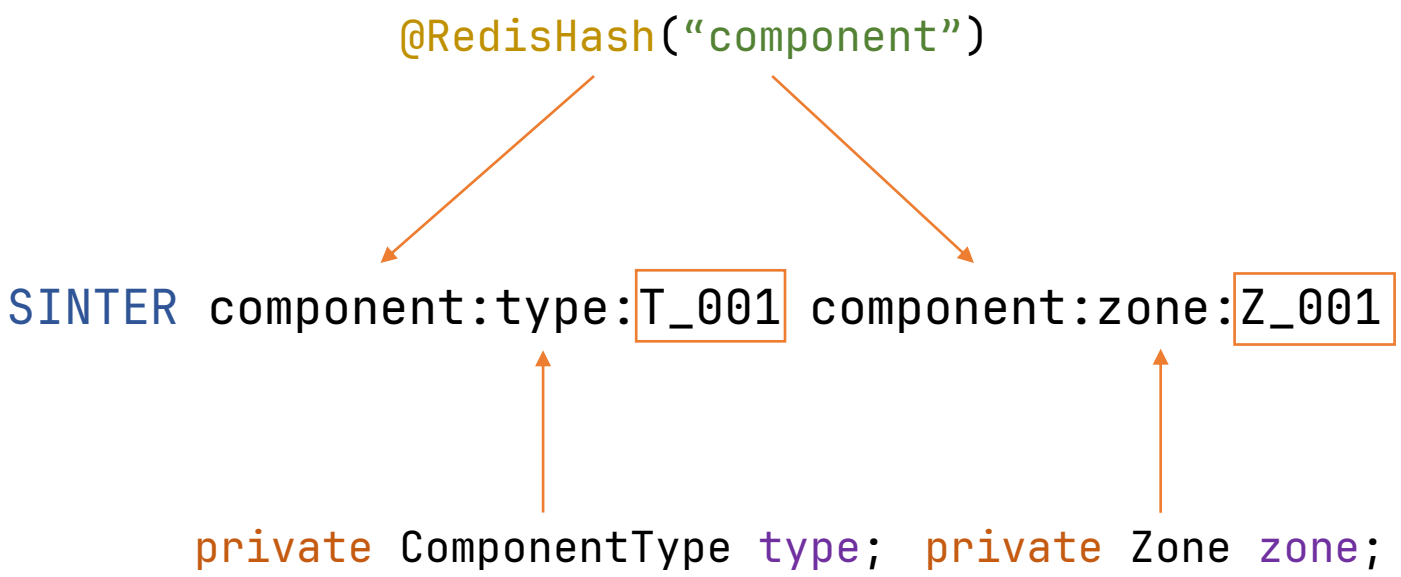
componentRepository.findByTypeAndZone(T_001, Z_001)



componentRepository.findByTypeAndZone(T_001, Z_001)



componentRepository.findByTypeAndZone(T_001, Z_001)



@Indexed

```
import org.springframework.data.redis.core.index.Indexed;

@RedisHash(“component”)
public class Component {

    private Long id;

    private Integer number;

    @Indexed
    private ComponentType type;

    @Indexed
    private Zone zone;

    private ControlPoint scanPoint;

    private ZonedDateTime scannedAt;

}
```

componentRepository.save + @Indexed

```
DEL component:1
```

componentRepository.save + @Indexed

```
DEL component:1
HMSET component:1 id 1 number 100 type T_001 zone Z_001 scanPoint
P_01 scannedAt 2024-10-15T12:59:59.999Z _class package.Component
```

componentRepository.save + @Indexed

```
DEL component:1
HMSET component:1 id 1 number 100 type T_001 zone Z_001 scanPoint
P_01 scannedAt 2024-10-15T12:59:59.999Z _class package.Component
SADD component 1
```

componentRepository.save + @Indexed

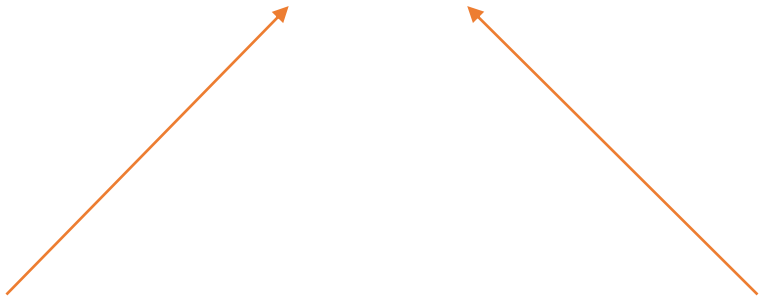
```
DEL component:1
HMSET component:1 id 1 number 100 type T_001 zone Z_001 scanPoint
P_01 scannedAt 2024-10-15T12:59:59.999Z _class package.Component
SADD component 1
SADD component:zone:Z_001 1
```

componentRepository.save + @Indexed

```
DEL component:1
HMSET component:1 id 1 number 100 type T_001 zone Z_001 scanPoint
P_01 scannedAt 2024-10-15T12:59:59.999Z _class package.Component
SADD component 1
SADD component:zone:Z_001 1
SADD component:1:idx component:zone:Z_001
```

componentRepository.save + @Indexed

```
DEL component:1
HMSET component:1 id 1 number 100 type T_001 zone Z_001 scanPoint
P_01 scannedAt 2024-10-15T12:59:59.999Z _class package.Component
SADD component 1
SADD component:zone:Z_001 1
SADD component:1:idx component:zone:Z_001
```



```
@RedisHash("component")
```

```
private Long id;
```

componentRepository.save + @Indexed

```
DEL component:1
HMSET component:1 id 1 number 100 type T_001 zone Z_001 scanPoint
P_01 scannedAt 2024-10-15T12:59:59.999Z _class package.Component
SADD component 1
SADD component:zone:Z_001 1
SADD component:1:idx component:zone:Z_001
SADD component:type:T_001 1
SADD component:1:idx component:type:T_001
```

Синхронизация индексов

```
@RedisHash(“component”)
public class Component {

    private Long id;


    private Integer number;

    @Indexed
    private ComponentType type;

    @Indexed
    private Zone zone;

    private ControlPoint scanPoint;

    private ZonedDateTime scannedAt;

}
```

Синхронизация индексов

```
@RedisHash(“component”)
public class Component {

    private Long id;

    @Indexed
    private Integer number;

    @Indexed
    private ComponentType type;

    @Indexed
    private Zone zone;

    private ControlPoint scanPoint;

    private ZonedDateTime scannedAt;

}
```

Синхронизация индексов

```
@Indexed
private Long number;

SADD component:number:100 1
SADD component:1:idx component:number:100
```

```
@Indexed
private ComponentType type;

SREM component:1:idx component:type:T_001
SREM component:type:T_001 1
```

Синхронизация индексов

```
@Indexed
private Long number;

SADD component:number:100 1
SADD component:1:idx component:number:100
```

Ненадежный

```
@Indexed
private ComponentType type;

SREM component:1:idx component:type:T_001
SREM component:type:T_001 1
```

Синхронизация индексов

```
@Indexed
private Long number;

SADD component:number:100 1
SADD component:1:idx component:number:100
```

Ненадежный

```
@Indexed
private ComponentType type;

SREM component:1:idx component:type:T_001
SREM component:type:T_001 1
```

Непостоянный

Синхронизация индексов

```
@RedisHash(“component”)
public class Component {

    private Long id;

    @Indexed private Integer number;

    @Indexed private ComponentType type;

    @Indexed private Zone zone;

    @Indexed private ControlPoint scanPoint;

    @Indexed private ZonedDateTime scannedAt;

}
```

Синхронизация индексов

```
@RedisHash(“component”)
public class Component {

    private Long id;

    @Indexed private Integer number;

    @Indexed private ComponentType type;

    @Indexed private Zone zone;

    @Indexed private ControlPoint scanPoint;

    @Indexed private ZonedDateTime scannedAt;

}
```

Низко-производительный

Синхронизация индексов

```
@RedisHash(“component”)
public class Component {

    private Long id;

    @Indexed private Integer number;

    @Indexed private ComponentType type;

    @Indexed private Zone zone;

    @Indexed private ControlPoint scanPoint;

    @Indexed private ZonedDateTime scannedAt;

}
```

Низко-производительный

Ресурсоемкий

Синхронизация индексов

```
@RedisHash(“component”)
public class Component {

    private Long id;

    @Indexed private Integer number;

    @Indexed private ComponentType type;

    @Indexed private Zone zone;

    @Indexed private ControlPoint scanPoint;

    @Indexed private ZonedDateTime scannedAt;

    @Indexed private Double weight;
}
```

Низко-производительный

Ресурсоемкий

Неэффективный

Синхронизация индексов

```
@EventListener(ApplicationStartedEvent.class)
public void syncIndex(){
    List<Component> all = componentRepository.findAll();
    componentRepository.saveAll(all);
}
```

componentRepository.save + @Indexed

```
DEL component:1
```

componentRepository.save + @Indexed

```
DEL component:1
HMSET component:1 id 1 number 100 type T_001 zone Z_001 scanPoint
P_01 scannedAt 2024-10-15T12:59:59.999Z _class package.Component
```

componentRepository.save + @Indexed

```
DEL component:1
HMSET component:1 id 1 number 100 type T_001 zone Z_001 scanPoint
P_01 scannedAt 2024-10-15T12:59:59.999Z _class package.Component
SADD component 1
```

componentRepository.save + @Indexed

```
DEL component:1
HMSET component:1 id 1 number 100 type T_001 zone Z_001 scanPoint
P_01 scannedAt 2024-10-15T12:59:59.999Z _class package.Component
SMEMBERS component:1:idx
```

componentRepository.save + @Indexed

```
DEL component:1
HMSET component:1 id 1 number 100 type T_001 zone Z_001 scanPoint
P_01 scannedAt 2024-10-15T12:59:59.999Z _class package.Component
SMEMBERS component:1:idx
TYPE component:type:T_001
```

componentRepository.save + @Indexed

```
DEL component:1
HMSET component:1 id 1 number 100 type T_001 zone Z_001 scanPoint
P_01 scannedAt 2024-10-15T12:59:59.999Z _class package.Component
SMEMBERS component:1:idx
TYPE component:type:T_001
SREM component:type:T_001 1
```

componentRepository.save + @Indexed

```
DEL component:1
HMSET component:1 id 1 number 100 type T_001 zone Z_001 scanPoint
P_01 scannedAt 2024-10-15T12:59:59.999Z _class package.Component
SMEMBERS component:1:idx
TYPE component:type:T_001
SREM component:type:T_001 1
TYPE component:zone:Z_001
SREM component:zone:Z_001 1
```

componentRepository.save + @Indexed

```
DEL component:1
HMSET component:1 id 1 number 100 type T_001 zone Z_001 scanPoint
P_01 scannedAt 2024-10-15T12:59:59.999Z _class package.Component
SMEMBERS component:1:idx
TYPE component:type:T_001
SREM component:type:T_001 1
TYPE component:zone:Z_001
SREM component:zone:Z_001 1
DEL component:1:idx
```

componentRepository.save + @Indexed

```
DEL component:1
HMSET component:1 id 1 number 100 type T_001 zone Z_001 scanPoint
P_01 scannedAt 2024-10-15T12:59:59.999Z _class package.Component
SMEMBERS component:1:idx
TYPE component:type:T_001
SREM component:type:T_001 1
TYPE component:zone:Z_001
SREM component:zone:Z_001 1
DEL component:1:idx
SADD component:zone:Z_001 1
SADD component:1:idx component:zone:Z_001
SADD component:type:T_001 1
SADD component:1:idx component:type:T_001
```

componentRepository.save + @Indexed

```
DEL component:1
HMSET component:1 id 1 number 100 type T_001 zone Z_001 scanPoint
P_01 scannedAt 2024-10-15T12:59:59.999Z _class package.Component
SMEMBERS component:1:idx
TYPE component:type:T_001
SREM component:type:T_001 1
TYPE component:zone:Z_001
SREM component:zone:Z_001 1
DEL component:1:idx
SADD component 1
SADD component:zone:Z_001 1
SADD component:1:idx component:zone:Z_001
SADD component:type:T_001 1
SADD component:1:idx component:type:T_001
```

Синхронизация индексов

Надежный

```
@EventListener(ApplicationStartedEvent.class)
public void syncIndex(){
    List<Component> all = componentRepository.findAll();
    componentRepository.saveAll(all);
}
```

Синхронизация индексов

```
@EventListener(ApplicationStartedEvent.class)
public void syncIndex(){
    List<Component> all = componentRepository.findAll();
    componentRepository.saveAll(all);
}
```

Надежный

Постоянный

Синхронизация индексов

```
@EventListener(ApplicationStartedEvent.class)
public void syncIndex(){
    List<Component> all = componentRepository.findAll();
    componentRepository.saveAll(all);
}
```

Надежный

Постоянный

Избирательный

Синхронизация индексов

```
@EventListener(ApplicationStartedEvent.class)
public void syncIndex(){
    List<Component> all = componentRepository.findAll();
    componentRepository.saveAll(all);
}
```

Надежный

Постоянный

Избирательный

OutOfMemory

Структура хранения

Индексы

Пользовательские запросы

Обходные решения

Кастомная реализация

Итог проекта

@Query

```
@Query(“SINTER component:type:?1 component:zone:?2”)
List<Integer> findIdsBySegment(ComponentType type, Zone zone);
```

Запросы на основе имени метода

```
List<Component> findByTypeAndZone(Component type, Zone zone);
```

Запросы на основе имени метода

```
public class KeyValuePartTreeQuery implements RepositoryQuery {

    protected Object doExecute(Object[] parameters, KeyValueQuery<?> query) {

        if (queryMethod.isPageQuery() || queryMethod.isSliceQuery()){
            Pageable page = (Pageable) parameters[...];
            Iterable<?> result = keyValueOperations.find(...);
            long count = queryMethod.isSliceQuery() ? 0 : keyValueOperations.count(...);
            return new PageImpl(toList(result), page, count);
        } else if (queryMethod.isCollectionQuery()) {
            return keyValueOperations.find(...);
        } else if (partTree.get().isExistsProjection()) {
            return keyValueOperations.exists(...);
        } else if (queryMethod.isQueryForEntity()) {
            Iterable<?> result = keyValueOperations.find(...);
            return result.iterator().hasNext() ? result.iterator().next() : null;
        }

        throw new UnsupportedOperationException("Query method not supported");
    }

}
```

до v3.2.4

Запросы на основе имени метода

```
public class KeyValuePartTreeQuery implements RepositoryQuery {

    protected Object doExecute(Object[] parameters, KeyValueQuery<?> query) {

        if (queryMethod.isPageQuery() || queryMethod.isSliceQuery()){
            Pageable page = (Pageable) parameters[...];
            Iterable<?> result = keyValueOperations.find(...);
            long count = queryMethod.isSliceQuery() ? 0 : keyValueOperations.count(...);
            return new PageImpl(toList(result), page, count);
        } else if (queryMethod.isCollectionQuery()) {
            return keyValueOperations.find(...);
        } else if (partTree.get().isExistsProjection()) {
            return keyValueOperations.exists(...);
        } else if (queryMethod.isQueryForEntity()) {
            Iterable<?> result = keyValueOperations.find(...);
            return result.iterator().hasNext() ? result.iterator().next() : null;
        }

        throw new UnsupportedOperationException("Query method not supported");
    }

}
```

до v3.2.4

Запросы на основе имени метода

```
public class KeyValuePartTreeQuery implements RepositoryQuery {

    protected Object doExecute(Object[] parameters, KeyValueQuery<?> query) {

        if (queryMethod.isPageQuery() || queryMethod.isSliceQuery()){
            Pageable page = (Pageable) parameters[...];
            Iterable<?> result = keyValueOperations.find(...);
            long count = queryMethod.isSliceQuery() ? 0 : keyValueOperations.count(...);
            return new PageImpl(toList(result), page, count);
        } else if (queryMethod.isCollectionQuery()) {
            return keyValueOperations.find(...);
        } else if (partTree.get().isExistsProjection()) {
            return keyValueOperations.exists(...);
        } else {
            Iterable<?> result = keyValueOperations.find(...);
            return result.iterator().hasNext() ? result.iterator().next() : null;
        }


    }

}
```

с v3.2.4

Запросы на основе имени метода

Table 1. Supported keywords inside method names

Keyword	Sample	Redis snippet
`And`	`findByLastnameAndFirstname`	`SINTER …:firstname:rand …:lastname:al'thor`
`Or`	`findByLastnameOrFirstname`	`SUNION …:firstname:rand …:lastname:al'thor`
`Is, Equals`	`findByFirstname`, `findByFirstnameIs`, `findByFirstnameEquals`	`SINTER …:firstname:rand`
`IsTrue`	`FindByAliveIsTrue`	`SINTER …:alive:1`
`IsFalse`	`findByAliveIsFalse`	`SINTER …:alive:0`
`Top,First`	`findFirst10ByFirstname`,`findTop5ByFirstname`	

Запросы с пагинацией и сортировкой

```
componentRepository.findByTypeAndZone(
        ComponentType.T_001,
        Zone.Z_001,
        PageRequest.of(0, 2, Sort.by(Sort.Direction.DESC, “number”))
);
```

Запросы с пагинацией и сортировкой

1.

```
id: 1
number: 100
```

```
id: 2
number: 200
```

```
id: 3
number: 300
```

Запросы с пагинацией и сортировкой

1. `id: 1` `number: 100` | `id: 2` `number: 200` | `id: 3` `number: 300`
2. `id: 3` `number: 300` | `id: 2` `number: 200` | `id: 1` `number: 100`

Запросы с пагинацией и сортировкой

1.
```
id: 1
number: 100
```
```
id: 2
number: 200
```
```
id: 3
number: 300
```

2.
```
id: 3
number: 300
```
```
id: 2
number: 200
```
```
id: 1
number: 100
```

3.
```
id: 3
number: 300
```
```
id: 2
number: 200
```

componentRepository.findByTypeAndZone(T_001, Z_001) + Pageable

```
SINTER component:type:T_001 component:zone:Z_001
```

componentRepository.findByTypeAndZone(T_001, Z_001) + Pageable

```
SINTER component:type:T_001 component:zone:Z_001
HGETALL component:1
HGETALL component:2
```

```
id: 1
number: 100
```

```
id: 2
number: 200
```

```
id: 3
```

componentRepository.findByTypeAndZone(T_001, Z_001) + Pageable

```
SINTER component:type:T_001 component:zone:Z_001
HGETALL component:1
HGETALL component:2
SINTER component:type:T_001 component:zone:Z_001
```

```
id: 1
number: 100
```

```
id: 2
number: 200
```

totalElements = 3

componentRepository.findByTypeAndZone(T_001, Z_001) + Pageable

```
SINTER component:type:T_001 component:zone:Z_001
HGETALL component:1
HGETALL component:2
SINTER component:type:T_001 component:zone:Z_001

result.sort(Comparator.comparing(Component::getNumber);
```

```
id: 2
number: 200
```

```
id: 1
number: 100
```

totalElements = 3

Структура хранения

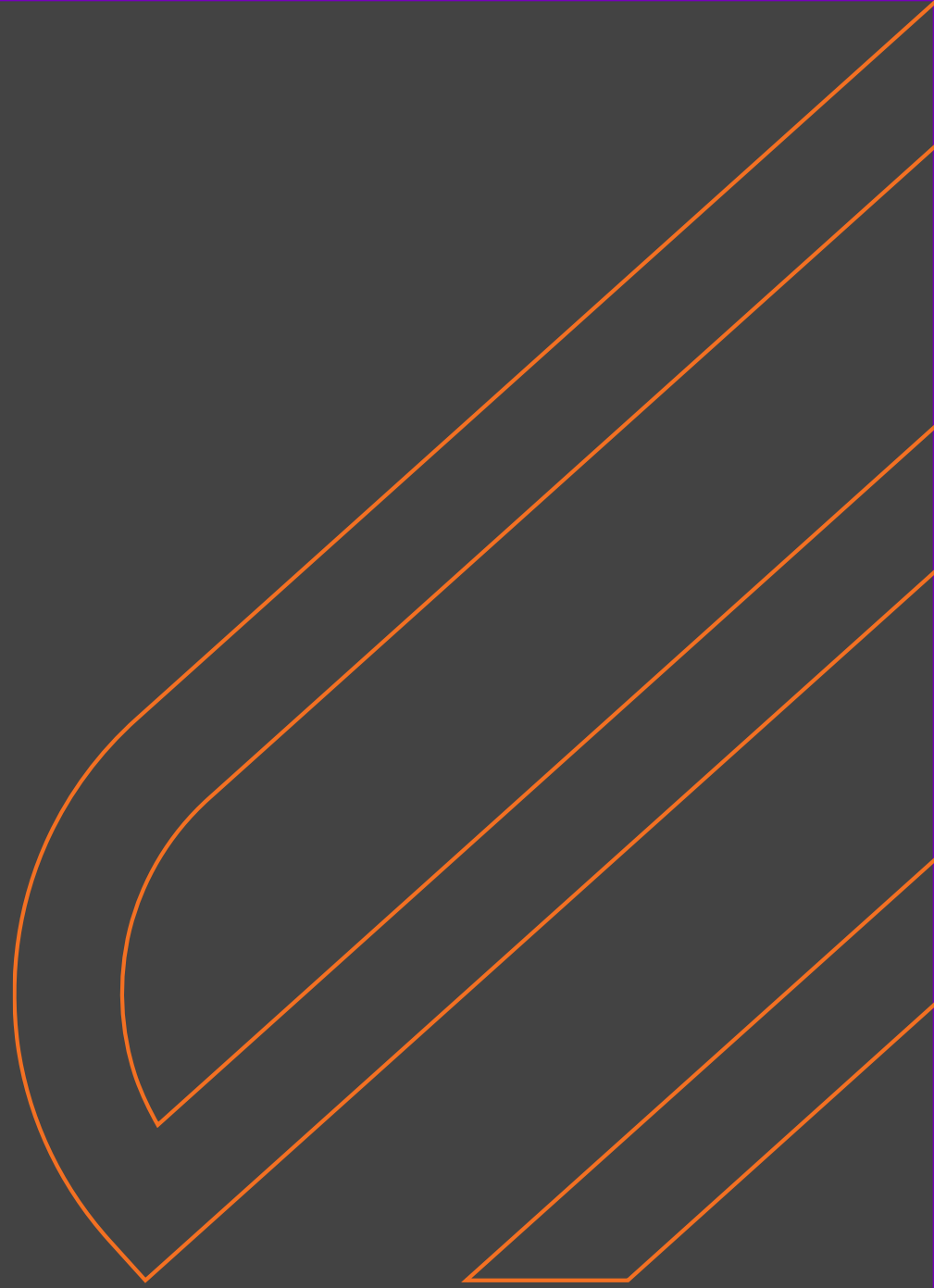
Индексы

Пользовательские запросы

Обходные решения

Кастомная реализация

Итог проекта



Обходные решения

Конвенции

Обходные решения

Конвенции

Сервисный метод

Синхронизация индексов

```
@EventListener(ApplicationStartedEvent.class)
public void syncIndex(){
    List<Component> all = componentRepository.findAll();
    componentRepository.saveAll(all);
}
```

componentService.deleteByType

```
public void deleteByType(ComponentType type) {
    List<Component> components = componentRepository.findByType(type);
    componentRepository.deleteAll(components);
}
```

componentService.deleteByType

findByType

```
SINTER component:type:T_001
```

componentService.deleteByType

findByType

```
SINTER component:type:T_001
HGETALL component:1
```

componentService.deleteByType

findByType

```
SINTER component:type:T_001
HGETALL component:1
```

deleteAll

```
HGETALL component:1
```

componentService.deleteByType

findByType

```
SINTER component:type:T_001
HGETALL component:1
```

deleteAll

```
HGETALL component:1
DEL component:1
```

componentService.deleteByType

findByType

```
SINTER component:type:T_001
HGETALL component:1
```

deleteAll

```
HGETALL component:1
DEL component:1
SREM component 1
```

componentService.deleteByType

findByType

```
SINTER component:type:T_001
HGETALL component:1
```

deleteAll

```
HGETALL component:1
DEL component:1
SREM component 1
SMEMBERS component:1:idx
```

componentService.deleteByType

findByType

```
SINTER component:type:T_001
HGETALL component:1
```

deleteAll

```
HGETALL component:1
DEL component:1
SREM component 1
SMEMBERS component:1:idx
TYPE component:zone:Z_001
SREM component:zone:Z_001 1
TYPE component:type:T_001
SREM component:type:T_001 1
```

componentService.deleteByType

findByType

```
SINTER component:type:T_001
HGETALL component:1
```

deleteAll

```
HGETALL component:1
DEL component:1
SREM component 1
SMEMBERS component:1:idx
TYPE component:zone:Z_001
SREM component:zone:Z_001 1
TYPE component:type:T_001
SREM component:type:T_001 1
DEL component:1:idx
```

Проблема метода save

Проблема метода save

key: component:1
value: HASH

Проблема метода save

key: component:1
value: HASH

key: component:type:T_001
value: [1]

Проблема метода save

findByType

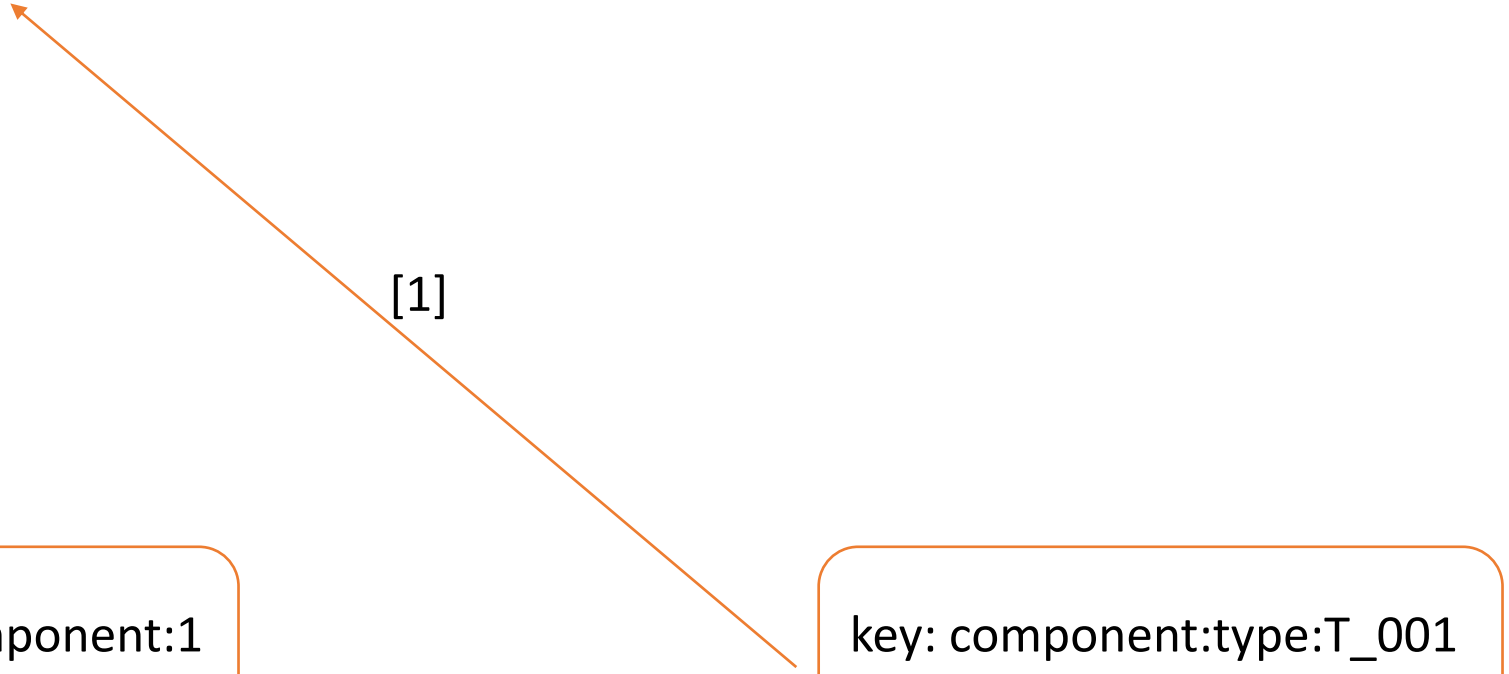
key: component:1
value: HASH

key: component:type:T_001
value: [1]

Проблема метода save

findByType

```
SINTER component:type:T_001
```



key: component:1
value: HASH

key: component:type:T_001
value: [1]

Проблема метода save

findBy**Type**

```
SINTER component:type:T_001
```

key: component:1
value: HASH

save

key: component:type:T_001
value: [1]

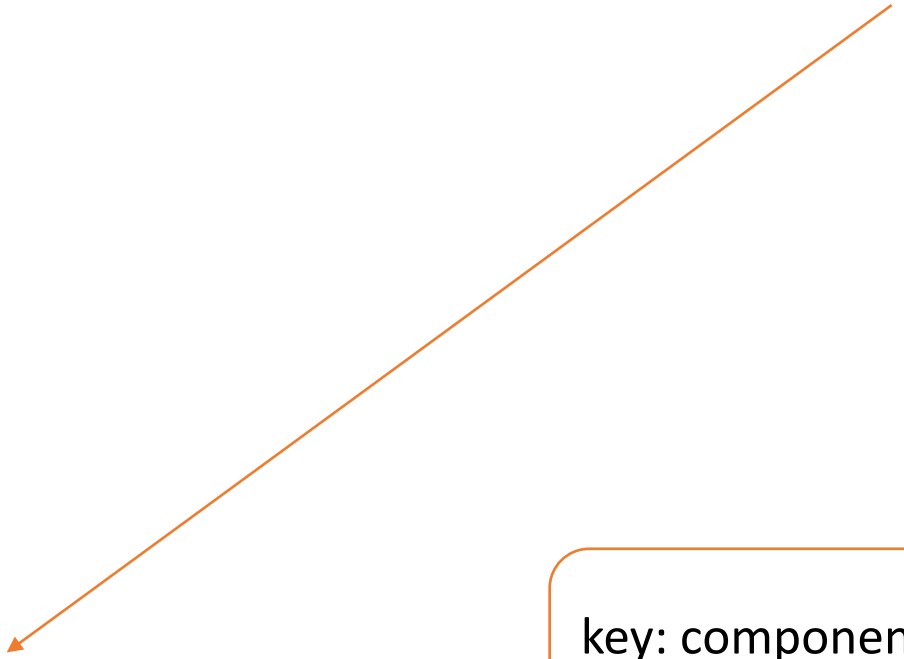
Проблема метода save

findByType

```
SINTER component:type:T_001
```

save

```
DEL component:1
```



key: component:1
value: NULL

key: component:type:T_001
value: [1]

Проблема метода save

findByType

```
SINTER component:type:T_001
HGETALL component:1
```

key: component:1
value: NULL

save

```
DEL component:1
```

key: component:type:T_001
value: [1]

Проблема метода save

```
redisOps.execute((RedisCallback<Object>) connection -> {
    byte[] key = toBytes(rdo.getId());
    byte[] objectKey = createKey(rdo.getKeyspace(), rdo.getId());




    boolean isNew = connection.del(objectKey) == 0;
    connection.hMSet(objectKey, rdo.getBucket().rawMap());



    if (isNew) {
        connection.sAdd(toBytes(rdo.getKeyspace()), key);
    }

    …

}
```

Проблема метода save

```
redisOps.execute((RedisCallback<Object>) connection -> {
    byte[] key = toBytes(rdo.getId());
    byte[] objectKey = createKey(rdo.getKeyspace(), rdo.getId());



    connection.multi();

    boolean isNew = connection.del(objectKey) == 0;
    connection.hMSet(objectKey, rdo.getBucket().rawMap());

    connection.exec();

    if (isNew) {
        connection.sAdd(toBytes(rdo.getKeyspace()), key);
    }

    …

}
```

Проблема метода save

```
redisOps.execute((RedisCallback<Object>) connection -> {
    byte[] key = toBytes(rdo.getId());
    byte[] objectKey = createKey(rdo.getKeyspace(), rdo.getId());



    connection.multi();

    boolean isNew = connection.del(objectKey) == 0;
    connection.hMSet(objectKey, rdo.getBucket().rawMap());

    connection.exec();

    if (isNew) {
        connection.sAdd(toBytes(rdo.getKeyspace()), key);
    }

    …

}
```

NullPointerException

Проблема метода save

```
redisOps.execute((RedisCallback<Object>) connection -> {
    byte[] key = toBytes(rdo.getId());
    byte[] objectKey = createKey(rdo.getKeyspace(), rdo.getId());



    connection.multi();

    boolean isNew = connection.del(objectKey) == 0;
    connection.hMSet(objectKey, rdo.getBucket().rawMap());

    connection.exec();

    if (isNew) {
        connection.sAdd(toBytes(rdo.getKeyspace()), key);
    }

    …

}
```

Проблема метода save

```
redisOps.execute((RedisCallback<Object>) connection -> {
    byte[] key = toBytes(rdo.getId());
    byte[] objectKey = createKey(rdo.getKeyspace(), rdo.getId());

    boolean isNew = connection.exists(objectKey);

    connection.multi();

    connection.del(objectKey) == 0;
    connection.hMSet(objectKey, rdo.getBucket().rawMap());

    connection.exec();

    if (isNew) {
        connection.sAdd(toBytes(rdo.getKeyspace()), key);
    }

    …

}
```

Проблема метода save

```
redisOps.execute((RedisCallback<Object>) connection -> {
    byte[] key = toBytes(rdo.getId());
    byte[] objectKey = createKey(rdo.getKeyspace(), rdo.getId());

    boolean isNew = connection.exists(objectKey);

    connection.multi();

    connection.del(objectKey) == 0;
    connection.hMSet(objectKey, rdo.getBucket().rawMap());

    connection.exec();

    if (isNew) {
        connection.sAdd(toBytes(rdo.getKeyspace()), key);
    }

    …

}
```

Проблема метода save

2 Answers

Sorted by: Highest score (default)

2

We have recently seen similar behavior. In our scenario, we can have multiple threads that read and write to the same repository. Our null return occurs when one thread is doing a save to an object while another is doing a findById for that same object. The findById will occasionally fail. It appears that the save implementation does a delete followed by an add; if the findById gets in during the delete, the null result is returned.

We've had good luck so far in our test programs that can reproduce the null return using a Java Semaphore to gate all access (read, write, delete) to the repository. When the repository access methods are all gated by the same semaphore, we have not seen a null return. Our next step is to try adding the synchronized keyword to the methods in the class that access the repository (as an alternative to using the Semaphore).

Share Edit Follow

answered Apr 21, 2021 at 15:36

D David Wilson

21 • 1

Thanks for the hint! Seems reasonable. I'm going to have a closer loot at this assumption – user1053031

Apr 22, 2021 at 2:22

Add a comment

Структура хранения

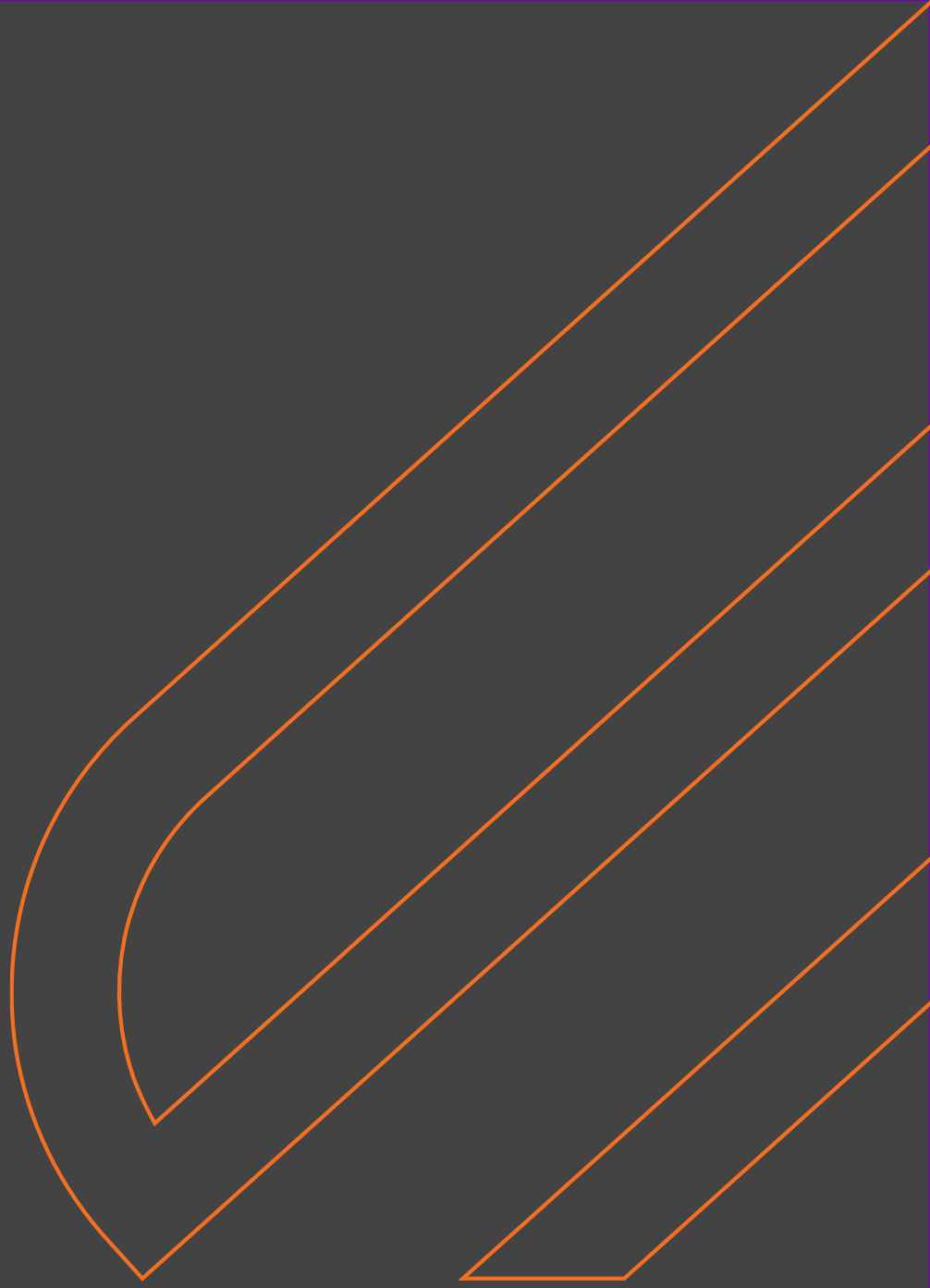
Индексы

Пользовательские запросы

Обходные решения

Кастомная реализация

Итог проекта



Абстракции Spring Data Redis

QueryExecutorMethodInterceptor

ImplementationMethodExecutionInterceptor

AopUtils

ExposeInvocationInterceptor

ReflectiveMethodInvocation

ExposeInvocationInterceptor

RepositoryMethodInvoker

DefaultMethodInvokingMethodInterceptor

RepositoryComposition

Абстракции Spring Data Redis

QueryExecutorMethodInterceptor

ImplementationMethodExecutionInterceptor

AopUtils

ExposeInvocationInterceptor

ReflectiveMethodInvocation

ExposeInvocationInterceptor

RepositoryMethodInvoker

DefaultMethodInvokingMethodInterceptor

RepositoryComposition

RedisPartTreeQuery

SimpleKeyValueRepository

KeyValuePartTreeQuery

Абстракции Spring Data Redis

QueryExecutorMethodInterceptor

ImplementationMethodExecutionInterceptor

AopUtils

ExposeInvocationInterceptor

ReflectiveMethodInvocation

ExposeInvocationInterceptor

RepositoryMethodInvoker

DefaultMethodInvokingMethodInterceptor

RepositoryComposition

RedisPartTreeQuery

SimpleKeyValueRepository

KeyValuePartTreeQuery

RedisTemplate

LettuceConnection

Абстракции Spring Data Redis

QueryExecutorMethodInterceptor

ImplementationMethodExecutionInterceptor

AopUtils

ExposeInvocationInterceptor

ReflectiveMethodInvocation

ExposeInvocationInterceptor

RepositoryMethodInvoker

DefaultMethodInvokingMethodInterceptor

RepositoryComposition

RedisPartTreeQuery

SimpleKeyValueRepository

KeyValuePartTreeQuery

RedisTemplate

LettuceConnection

RedisKeyValueTemplate

RedisKeyValueAdapter

IndexWriter

Абстракции Spring Data Redis

RedisKeyValueAdapter

- put
- update
- deleteAllOf
- delete
- getAllOf
- get
- count
- contains
- execute

Абстракции Spring Data Redis

RedisKeyValueAdapter

- put
- update
- deleteAllOf
- delete
- getAllOf
- get
- count
- contains
- execute

RedisKeyValueTemplate

- insert
- update
- delete
- findAll
- find
- findById
- findInRange
- count
- exists
- execute

Абстракции Spring Data Redis

RedisKeyValueAdapter

- put
- update
- deleteAllOf
- delete
- getAllOf
- get
- count
- contains
- execute

RedisKeyValueTemplate

- insert
- update
- delete
- findAll
- find
- findById
- findInRange
- count
- exists
- execute

IndexWriter

- createIndex
- updateIndex
- deleteAndUpdateIndexes
- removeKeyFromIndexes
- removeAllIndexes

Абстракции Spring Data Redis

RedisKeyValueAdapter

- put
- update
- deleteAllOf
- delete
- getAllOf
- get
- count
- contains
- execute

RedisKeyValueTemplate

- insert
- update
- delete
- findAll
- find
- findById
- findInRange
- count
- exists
- execute

IndexWriter

- createIndex
- updateIndex
- deleteAndUpdateIndexes
- removeKeyFromIndexes
- removeAllIndexes

Абстракции Spring Data Redis

RedisKeyValueAdapter

- put
- update
- deleteAllOf
- delete
- getAllOf
- get
- count
- contains
- execute

RedisKeyValueTemplate

- insert
- update
- delete
- findAll
- find
- findById
- findInRange
- count
- exists
- execute

IndexWriter

- createIndex
- updateIndex
- deleteAndUpdateIndexes
- removeKeyFromIndexes
- removeAllIndexes

```
if (obj instanceof PartialUpdate pu){
    return adapter.update(pu);
}
```

Абстракции Spring Data Redis

RedisKeyValueAdapter

- put
- update
- deleteAllOf
- delete
- getAllOf
- get
- count
- contains
- execute

RedisKeyValueTemplate

- insert
- update
- delete
- findAll
- find
- findById
- findInRange
- count
- exists
- execute

IndexWriter

- createIndex
- updateIndex
- deleteAndUpdateIndexes
- removeKeyFromIndexes
- removeAllIndexes

```
if (obj instanceof PartialUpdate pu){
    return adapter.update(pu);
}
```

```
INSERT INTO component VALUES(...)
UPDATE component SET ...
```

Абстракции Spring Data Redis

RedisKeyValueAdapter

- put
- update
- deleteAllOf
- delete
- getAllOf
- get
- count
- contains
- execute

RedisKeyValueTemplate

- insert
- update
- delete
- findAll
- find
- findById
- findInRange
- count
- exists
- execute

IndexWriter

- createIndex
- updateIndex
- deleteAndUpdateIndexes
- removeKeyFromIndexes
- removeAllIndexes

```
if (obj instanceof PartialUpdate pu){
    return adapter.update(pu);
}
```

```
HMSET component:1
      id              1
      number          100
      type            T_001
      zone            Z_001
      scanPoint       P_01
      ...
```

Абстракции Spring Data Redis

RedisKeyValueAdapter

- put
- update
- deleteAllOf
- delete
- getAllOf
- get
- count
- contains
- execute

RedisKeyValueTemplate

- insert
- update
- delete
- findAll
- find
- findById
- findInRange
- count
- exists
- execute

= save

IndexWriter

- createIndex
- updateIndex
- deleteAndUpdateIndexes
- removeKeyFromIndexes
- removeAllIndexes

```
if (obj instanceof PartialUpdate pu){
    return adapter.update(pu);
}
```

```
HMSET component:1
      id              1
      number          100
      type            T_001
      zone            Z_001
      scanPoint       P_01
      ...
```

PartialUpdate

PartialUpdate

```
new PartialUpdate<>(id, Component.class)
```

PartialUpdate

```
new PartialUpdate<>(id, Component.class)
        .set("type", ComponentType.T_004)
```

PartialUpdate

```
new PartialUpdate<>(id, Component.class)
        .set(“type”, ComponentType.T_004)
        .del(“scanPoint”);
```

PartialUpdate

```
new PartialUpdate<>(id, Component.class)
        .set(“type”, ComponentType.T_004)
        .del(“scanPoint”);

new PartialUpdate<>(component.getId(), component)
```

PartialUpdate

```
new PartialUpdate<>(id, Component.class)
        .set(“type”, ComponentType.T_004)
        .del(“scanPoint”);

new PartialUpdate<>(component.getId(), component)
        .del(“scanPoint”);
```

redisKeyValueAdapter.update(partialUpdate)

```
Component component = Component.builder()
        .id(1L)
        .type(T_002) //@Indexed
        .build;

redisKeyValueTemplate.update(
        new PartialUpdate<>(component.getId(), component)
              .del("scanPoint")
);
```

redisKeyValueAdapter.update(partialUpdate)

```
HGET component:1 scanPoint
```

redisKeyValueAdapter.update(partialUpdate)

```
HGET component:1 scanPoint
EXISTS component:scanPoint:P_01
```

redisKeyValueAdapter.update(partialUpdate)

```
HGET component:1 scanPoint
EXISTS component:scanPoint:P_01
```

Очищаются индексы полей без @Indexed

redisKeyValueAdapter.update(partialUpdate)

```
HGET component:1 scanPoint
EXISTS component:scanPoint:P_01
HDEL component:1 scanPoint
```

Очищаются индексы полей без @Indexed

redisKeyValueAdapter.update(partialUpdate)

del

```
HGET component:1 scanPoint
EXISTS component:scanPoint:P_01
HDEL component:1 scanPoint
```

Очищаются индексы полей без @Indexed

Первыми выполняются методы del

redisKeyValueAdapter.update(partialUpdate)

del
```
HGET component:1 scanPoint
EXISTS component:scanPoint:P_01
HDEL component:1 scanPoint
```

set
```
HMSET component:1 id 1 type T_002
```

Очищаются индексы полей без @Indexed

Первыми выполняются методы del

redisKeyValueAdapter.update(partialUpdate)

del

```
HGET component:1 scanPoint
EXISTS component:scanPoint:P_01
HDEL component:1 scanPoint
```

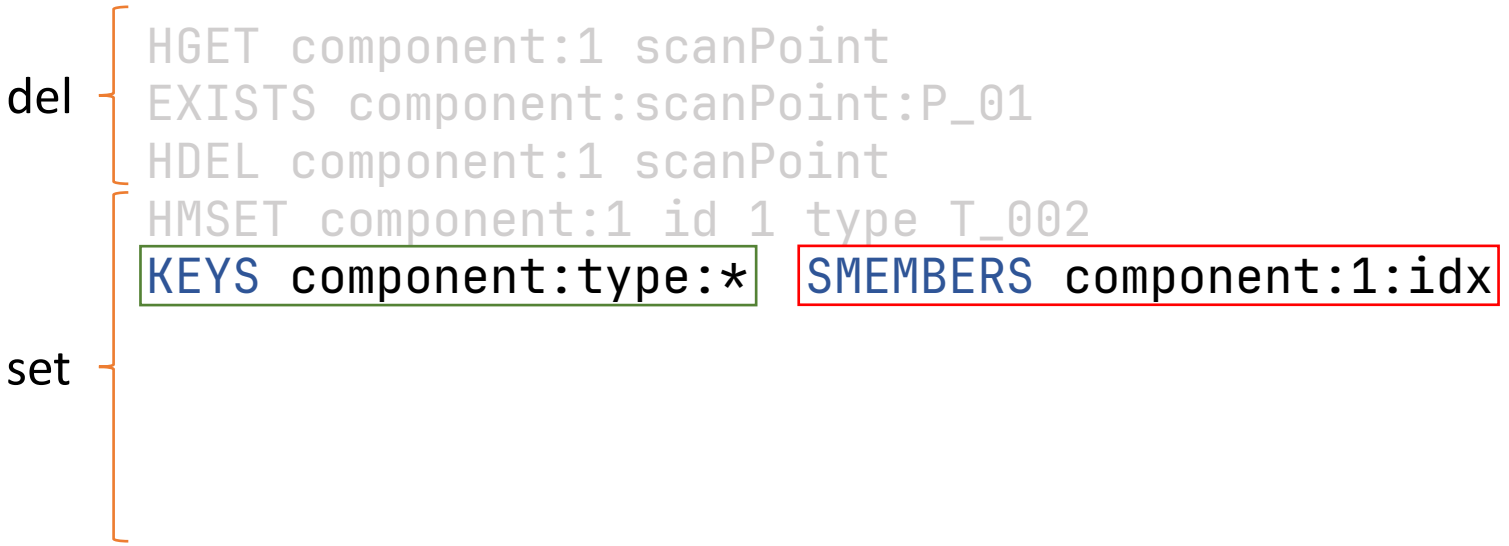
set

```
HMSET component:1 id 1 type T_002
KEYS component:type:*
```

Очищаются индексы полей без @Indexed

Первыми выполняются методы del

redisKeyValueAdapter.update(partialUpdate)



Очищаются индексы полей без @Indexed

Первыми выполняются методы del

Не используется список вторичных индексов

redisKeyValueAdapter.update(partialUpdate)

del
```
HGET component:1 scanPoint
EXISTS component:scanPoint:P_01
HDEL component:1 scanPoint
```

set
```
HMSET component:1 id 1 type T_002
KEYS component:type:*
SREM component:type:T_001 1
```

Очищаются индексы полей без @Indexed

Первыми выполняются методы del

Не используется список вторичных индексов

redisKeyValueAdapter.update(partialUpdate)

del
```
HGET component:1 scanPoint
EXISTS component:scanPoint:P_01
HDEL component:1 scanPoint
```

set
```
HMSET component:1 id 1 type T_002
KEYS component:type:*
SREM component:type:T_001 1
SADD component:type:T_002 1
SADD component:1:idx component:type:T_002
```

Очищаются индексы полей без @Indexed

Первыми выполняются методы del

Не используется список вторичных индексов

redisKeyValueAdapter.update(partialUpdate)

del
```
HGET component:1 scanPoint
EXISTS component:scanPoint:P_01
HDEL component:1 scanPoint
```

set
```
HMSET component:1 id 1 type T_002
KEYS component:type:*
-SREM component:type:T_001 1
+SADD component:type:T_002 1
+SADD component:1:idx component:type:T_002
-SREM component:1:idx component:type:T_001
```

Очищаются индексы полей без @Indexed

Первыми выполняются методы del

Не используется список вторичных индексов

Не очищается список вторичных индексов

Особенности PartialUpdate

Особенности PartialUpdate

RedisKeyValueTemplate
insert/update

```
public <T> T insert(Object id, T obj) {

    if (obj instanceof PartialUpdate pu){
        return adapter.update(pu);
    }

    …

}

public <T> T update(Object id, T obj) {

    if (obj instanceof PartialUpdate pu){
        return adapter.update(pu);
    }

    …

}
```

Особенности PartialUpdate

RedisKeyValueTemplate insert/update

При del передавать null

```
Component component = Component.builder()
        .id(1L)
        .type(T_002)
        .scanPoint(P_01)
        .build;

redisKeyValueAdapter.update(
        new PartialUpdate<>(component.getId(), component)
                .del("scanPoint")
);
```

Особенности PartialUpdate

RedisKeyValueTemplate
insert/update

При del передавать null

Не наполняется
первичный индекс

```
HGET component:1 scanPoint
EXISTS component:scanPoint:P_01
HDEL component:1 scanPoint
HMSET component:1 id 1 type T_002
KEYS component:type:*
SREM component:type:T_001 1
SADD component:type:T_002 1
SADD component:1:idx component:type:T_002
SADD component 1
```

Особенности PartialUpdate

RedisKeyValueTemplate insert/update

При del передавать null

Не наполняется первичный индекс

```
@Id
@Indexed
private Long identifier;

componentRepository.findByIdentifier(id);
```

Плюс PartialUpdate

Не удаляется объект из базы данных

```
DEL component:1
HMSET component:1 id 1 type T_002
```

Минусы PartialUpdate

Не удаляется объект из базы данных

```
DEL component:1
HMSET component:1 id 1 type T_002
```

Минусы PartialUpdate

Не удаляется объект из базы данных

Не используется список вторичных индексов

```
HMSET component:1 id 1 type T_002
KEYS component:type:*  SMEMBERS component:1:idx
SREM component:type:T_001 1
SADD component:type:T_002 1
SADD component:1:idx component:type:T_002
```

Минусы PartialUpdate

Не удаляется объект из базы данных

Не используется список вторичных индексов

```
HMSET component:1 id 1 type T_002
KEYS component:type:*
SREM component:type:T_001 1
SADD component:type:T_002 1
SADD component:1:idx component:type:T_002
```

Минусы PartialUpdate

Не удаляется объект из базы данных

Не используется список вторичных индексов

```
HMSET component:1 id 1 type T_002
KEYS component:type:*
SREM component:type:T_001 1
SREM component:type:T_002 1
SREM component:type:T_003 1
…
SADD component:type:T_002 1
SADD component:1:idx component:type:T_002
```

Минусы PartialUpdate

Не удаляется объект из базы данных

Не используется список вторичных индексов

Не очищается список вторичных индексов

```
  HMSET component:1 id 1 type T_002
  KEYS component:type:*
- SREM component:type:T_001 1
+ SADD component:type:T_002 1
+ SADD component:1:idx component:type:T_002
- SREM component:1:idx component:type:T_001
```

Минусы PartialUpdate

Не удаляется объект из базы данных

Не используется список вторичных индексов

Не очищается список вторичных индексов

```
@EventListener(ApplicationStartedEvent.class)
public void syncIndex(){
    List<Component> all = componentRepository.findAll();
    componentRepository.saveAll(all);
}
```

Минусы PartialUpdate

Не удаляется объект из базы данных

Не используется список вторичных индексов

Не очищается список вторичных индексов

```
@Scheduled(cron = “@weekly”)
@EventListener(ApplicationStartedEvent.class)
public void syncIndex(){
    List<Component> all = componentRepository.findAll();
    componentRepository.saveAll(all);
}
```

Структура хранения

Индексы

Пользовательские запросы

Обходные решения

Кастомная реализация

Итог проекта

Ключевые процессы

Ключевые процессы

Сохранение данных

CustomComponentRepositoryImpl

```
@RequiredArgsConstructor
public class CustomComponentRepositoryImpl implements CustomComponentRepository {

    private final RedisKeyValueTemplate redisKeyValueTemplate;

    @Override
    public Component save(Component component) {
        PartialUpdate<Component> pu = new PartialUpdate<>(component.getId(), component);

        if (component.getScanPoint() == null) pu.del("scanPoint");
        if (component.getScannedAt() == null) pu.del("scannedAt");
        if (component.getWeight() == null) pu.del("weight");

        pu = redisKeyValueTemplate.update(pu);
        return pu.getValue();
    }

}
```

CustomComponentRepositoryImpl

```
@RequiredArgsConstructor
public class CustomComponentRepositoryImpl implements CustomComponentRepository {

    private final RedisKeyValueTemplate redisKeyValueTemplate;

    @Override
    public Component save(Component component) {
        PartialUpdate<Component> pu = new PartialUpdate<>(component.getId(), component);

        if (component.getScanPoint() == null) pu.del("scanPoint");
        if (component.getScannedAt() == null) pu.del("scannedAt");
        if (component.getWeight() == null) pu.del("weight");

        pu = redisKeyValueTemplate.update(pu);
        return pu.getValue();
    }

}
```

CustomComponentRepositoryImpl

```
@RequiredArgsConstructor
public class CustomComponentRepositoryImpl implements CustomComponentRepository {

    private final RedisKeyValueTemplate redisKeyValueTemplate;

    @Override
    public Component save(Component component) {
        PartialUpdate<Component> pu = new PartialUpdate<>(component.getId(), component);

        if (component.getScanPoint() == null) pu.del("scanPoint");
        if (component.getScannedAt() == null) pu.del("scannedAt");
        if (component.getWeight() == null) pu.del("weight");

        pu = redisKeyValueTemplate.update(pu);
        return pu.getValue();
    }

}
```

CustomComponentRepositoryImpl

```
@RequiredArgsConstructor
public class CustomComponentRepositoryImpl implements CustomComponentRepository {

    private final RedisKeyValueTemplate redisKeyValueTemplate;

    @Override
    public Component save(Component component) {
        PartialUpdate<Component> pu = new PartialUpdate<>(component.getId(), component);

        if (component.getScanPoint() == null) pu.del("scanPoint");
        if (component.getScannedAt() == null) pu.del("scannedAt");
        if (component.getWeight() == null) pu.del("weight");

        pu = redisKeyValueTemplate.update(pu);
        return pu.getValue();
    }

}
```

CustomComponentRepositoryImpl

```
@RequiredArgsConstructor
public class CustomComponentRepositoryImpl implements CustomComponentRepository {

    private final RedisKeyValueTemplate redisKeyValueTemplate;

    @Override
    public Component save(Component component) {
        PartialUpdate<Component> pu = new PartialUpdate<>(component.getId(), component);

        if (component.getScanPoint() == null) pu.del("scanPoint");
        if (component.getScannedAt() == null) pu.del("scannedAt");
        if (component.getWeight() == null) pu.del("weight");

        pu = redisKeyValueTemplate.update(pu);
        return pu.getValue();
    }

}
```

Ключевые процессы

Сохранение данных

Получение данных

ComponentRepository

```
public class ComponentRepository extends KeyValueRepository<Component, Long>,
                                         CustomComponentRepository {

    List<Component> findByType(ComponentType type);

    List<Component> findByTypeAndZone(ComponentType type, Zone zone);

    Optional<Component> findByTypeAndNumber(ComponentType type, Integer number);

}
```

ComponentRepository

```
public class ComponentRepository extends KeyValueRepository<Component, Long>,
                                         CustomComponentRepository {

    List<Component> findByType(ComponentType type);

    List<Component> findByTypeAndZone(ComponentType type, Zone zone);

    Optional<Component> findByTypeAndNumber(ComponentType type, Integer number);

}
```

Ключевые процессы

Сохранение данных

Получение данных

Удаление данных

ComponentServiceImpl

```
@Service
@RequiredArgsConstructor
public class ComponentServiceImpl {

    private final ComponentRepository repository;

    @Override
    public void deleteByNumberAndType(Integer number, ComponentType type) {
        componentRepository.findByNumberAndSegmentType(number, type)
                .ifPresent(componentRepository::delete);
    }

}
```

Ключевые процессы

Сохранение данных

Получение данных

Удаление данных

Синхронизация модели/индексов

Synchronizer

```
@Service
@RequiredArgsConstructor
public class Synchronizer {

    private final ComponentRepository componentRepository;

    @EventListener(ApplicationStartedEvent.class)
    @SchedulerLock(name = “IndexSynchronizer_startupSync”)
    public void startupSync() {
        syncIndex();
    }

    @Scheduled(cron = “@weekly”)
    @SchedulerLock(name = “IndexSynchronizer_weekendSync”)
    public void weekendSync() {
        syncIndex();
    }

…
```

Synchronizer

```
@Service
@RequiredArgsConstructor
public class Synchronizer {

    private final ComponentRepository componentRepository;

    @EventListener(ApplicationStartedEvent.class)
    @SchedulerLock(name = “IndexSynchronizer_startupSync”)
    public void startupSync() {
        syncIndex();
    }

    @Scheduled(cron = “@weekly”)
    @SchedulerLock(name = “IndexSynchronizer_weekendSync”)
    public void weekendSync() {
        syncIndex();
    }

…
```

Synchronizer

```
@Service
@RequiredArgsConstructor
public class Synchronizer {

    private final ComponentRepository componentRepository;

    @EventListener(ApplicationStartedEvent.class)
    @SchedulerLock(name = “IndexSynchronizer_startupSync”)
    public void startupSync() {
        syncIndex();
    }

    @Scheduled(cron = “@weekly”)
    @SchedulerLock(name = “IndexSynchronizer_weekendSync”)
    public void weekendSync() {
        syncIndex();
    }

…
```

Synchronizer

```
…

    private void syncIndex() {
        Arrays.stream(ComponentType.values())
                .map(componentRepository::findBySegmentType)
                .flatMap(List::stream)
                .forEach(component -> {
                    componentRepository.delete(component);
                    componentRepository.save(component);
                }
    }

}
```

Synchronizer

```
…

    private void syncIndex() {
        Arrays.stream(ComponentType.values())
            .map(componentRepository::findBySegmentType)
            .flatMap(List::stream)
            .forEach(component -> {
                componentRepository.delete(component);
                componentRepository.save(component);
            }
    }

}
```

Synchronizer

```
…

    private void syncIndex() {
        Arrays.stream(ComponentType.values())
                .map(componentRepository::findBySegmentType)
                .flatMap(List::stream)
                .forEach(component -> {
                    componentRepository.delete(component);
                    componentRepository.save(component);
                }
    }

}
```

Spring Data Redis неудачное решение?

Spring Data Redis неудачное решение?

- Синхронизация индексов
- deleteBy не поддерживается
- Сортировка выполняется после пагинации
- Команды выполняются не атомарно
- RedisKeyValueAdapter/Template, IndexWriter

axenix

Спасибо за внимание!