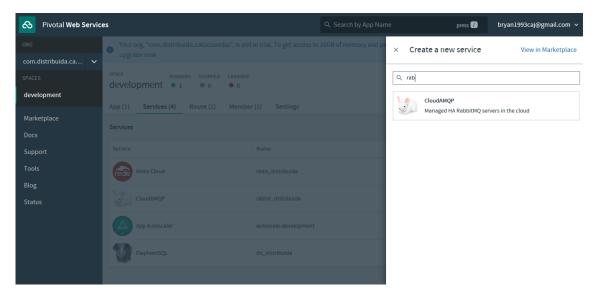
Manual

RabbitMQ en Pivotal

Agregamos el servicio en pivotal



Creamos:

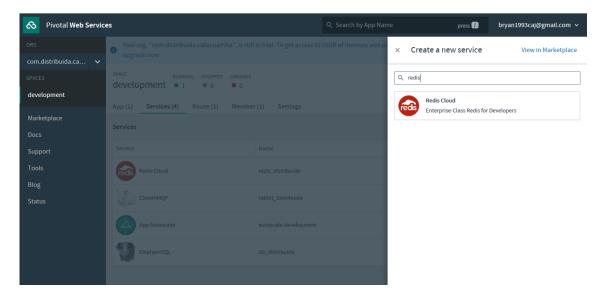
- productor s1 RabbitTemplate
- consumidor s2 Listener
- productor web RabbitTemplate

Agregamos las configuraciones respectivamente de RabbitMQ en el application.properties

```
spring.rabbitmq.username=gvnuppap
spring.rabbitmq.password=E3FvSwV2t5zFG52vdkPbrLE_xcpP4gUT
spring.rabbitmq.host=lion.rmq.cloudamqp.com
spring.rabbitmq.virtual-host=gvnuppap
```

REDIS en Pivotal

Agregamos el servicio en pivotal



Agregamos las configuraciones respectivamente de Redis en el application.properties

```
#redis
spring.redis.host=redis-10830.c52.us-east-1-4.ec2.cloud.redislabs.com
spring.redis.port=10830
spring.redis.password=Av7edTjTVkXnjdhYULFimqQNVhQ3IVbS
```

El manejo lo hacemos en s2 espcificamente

Se implementó la clase CacheSinger, en donde tenemos la lista de los Singer:

```
import java.util.List;
@Component
public class CacheSinger {

@Autowired
RestTemplate rt;

@Cacheable("listaCantantes")
public List<Singer> listarSingers() {
    System.out.println("Guardando datos en cache");
    List<Singer> singers;
    singers = rt.getForObject("http://localhost:9090/api/serverOljdbc/singer/singers" , List.class);
    System.out.println(singers);
    return singers;
}
```

En la clase SingerListener tenemos el método para refrescar el cache, cuando recibe la notificación

```
11
 12 @Component
 13 public class SingerListener {
 14
 15
 160
        @Autowired
         RestTemplate rt;
 18
 19⊜
        @CachePut(cacheNames="listaCantantes")
 20
         public List<Singer> receiveMessage(Singer singer) {
              System.out.println(">>>> actualizando lista de cantantes");
System.out.println("Refrescando datos en cache");
 21
 22
 23
              List<Singer> singers;
<u>24</u>
              singers = rt.getForObject("http://localhost:9090/api/server01jdbc/singer/singers" , List.class);
 25
              System.out.println(singers);
 26
              return singers;
         }
 27
 28
 29
    }
```

La clase principal, clase boot debe tener la anotación: @EnableCatching

```
^ 0 4⊕ import java.util.List;

## src/main/java
## uce.distribuida.catucuamba
## uce.distribuida.catucuamba
## uce.distribuida.catucuamba.cache
### uce.distribuida.catucuamba.cache
                                                                                       @SpringBootApplication(scanBasePackages = {"uce.distribuida.catucuamba"})
@EnableCaching
            > 🔑 CacheSinger.java
       41 public class Server02JdbcApplication {

■ uce.distribuida.catucuamba.controladore

□ AlbumControladori,ava

uce.distribuida.catucuamba.entidades

□ Albumjava

□ Singerjava

uce.distribuida.catucuamba.mensajeria
                                                                                           public static final String topicExchangeName = "distribuida-exchange";
                                                                                           public static final String queueName = "distribuida";
              SingerListener.java
       ✓ ∰ uce.distribuida.catucuamba.servicios
    > If AlbumServicio.java
> If AlbumServicio.java
> If AlbumServicioImpl.java
> templates
                                                                                           }
                                                                                            @Bean
public TopicExchange exchange() {
   return new TopicExchange(topicExchangeName);
          application.pro
                                                                                            @Bean
public Binding binding(Queue queue, TopicExchange exchange) {
    return BindingBuilder.bind(queue).to(exchange).with("foo.bar.#");
<
5 5 ■ 3 5 / 1 5 4 7
                                                                                             @Bean
public SimpleMessageListenerContainer container1(ConnectionFactory connectionFactory, MessageListenerAdapto
Type tags, projects, or working set names to match (incl. * and ? wildcard
```

Además de las declaraciones para poder implementar el servicio cache

```
@Bean
JedisConnectionFactory jedisConnectionFactory() {
    return new JedisConnectionFactory();
}

@Bean
public RedisTemplate<String, Object> redisTemplate() {
    RedisTemplate<String, Object> template = new RedisTemplate<>>();
    template.setConnectionFactory(jedisConnectionFactory());
    return template;
}
```

No olvidemos las dependencias necesarias

```
// https://mvnrepository.com/artifact/org.springframework.data/spring-data-redis
compile group: 'org.springframework.data', name: 'spring-data-redis', version: '2.0.9.RELEASE'
    // https://mvnrepository.com/artifact/redis.clients/jedis
compile group: 'redis.clients', name: 'jedis', version: '2.9.0'
```