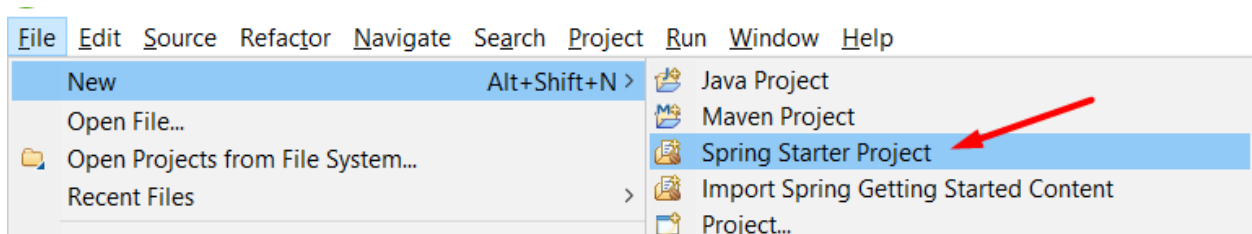


PROYECTO "REDIS LISTA FIFO"

Paso 1: Crear el Proyecto en STS

Abrir Spring Tool Suite (STS).

Ir a **File → New → Spring Starter Project**



Completar los datos del proyecto:

- **Name:** redis-waitlist-demo
- **Type:** Maven
- **Java Version:** 17
- **Group:** com.example
- **Artifact:** redis-waitlist-demo
- **Package:** com.example.rediswaitlistdemo
- **Packaging:** Jar

Service URL	<input type="text" value="https://start.spring.io"/>		
Name	<input type="text" value="redis-waitlist-demo-1"/>		
<input checked="" type="checkbox"/> Use default location			
Location	<input type="text" value="C:\Users\JR\Documents\workspace\redis-waitlist-demo-1"/>	<input type="button" value="Browse"/>	
Type:	<input type="text" value="Maven"/>	Packaging:	<input type="text" value="Jar"/>
Java Version:	<input type="text" value="17"/>	Language:	<input type="text" value="Java"/>
Group	<input type="text" value="com.example"/>		
Artifact	<input type="text" value="redis-waitlist-demo-1"/>		
Version	<input type="text" value="0.0.1-SNAPSHOT"/>		
Description	<input type="text" value="Demo project for Spring Boot"/>		
Package	<input type="text" value="com.example.rediswaitlistdemo"/>		

- **Dependencies:**
 - **Spring Web (Para la API REST).**
 - **Spring Data Redis (Access + Driver)** (Para conectarnos a Redis).
 - **Lombok** (Para reducir código boilerplate).

<input type="checkbox"/> H2 Database	<input checked="" type="checkbox"/> Lombok	<input type="checkbox"/> PostgreSQL Driver
<input type="checkbox"/> Spring Boot DevTools	<input type="checkbox"/> Spring Cache Abstraction	<input type="checkbox"/> Spring Data JPA
<input checked="" type="checkbox"/> Spring Data Redis (Access+Driver)	<input checked="" type="checkbox"/> Spring Web	<input type="checkbox"/> Spring Web Services

Hacer clic en **Finish** para generar el proyecto.

Paso 2: Configurar Redis en application.properties

Añadimos la conexión a nuestra base de datos Redis.



Configuración de Redis

spring.data.redis.host=redis-15022.c82.us-east-1-2.ec2.redns.redis-cloud.com

spring.data.redis.port=15022

spring.data.redis.username=default

spring.data.redis.password=x4M4OjOpwpwr5B331n4NH43FSIRgrwkm

Explicación:

- Se configuran los datos de conexión a Redis.

Paso 3: Crear la Clase Usuario

Creamos la clase Usuario



```
package com.example.rediswaitlistdemo.model;
```

```
import lombok.AllArgsConstructor;
```

```
import lombok.Getter;
```

```
import lombok.NoArgsConstructor;
```

```
import lombok.Setter;
```

```
import java.io.Serializable;
```

```
@Getter
```

```
@Setter
```

```
@NoArgsConstructor
```

```
@AllArgsConstructor
```

```
public class Usuario implements Serializable {
```

```
    private String id;
```

```
    private String nombre;
```

```
}
```

Explicación:

- Usuario representa a una persona en la lista de espera.
- Serializable permite que los datos sean guardados correctamente en Redis.
- Usamos Lombok (@Getter, @Setter, etc.) para simplificar el código.

Paso 4: Crear WaitlistService para manejar la Cola de Espera

Creamos la lógica de negocio en el service WaitlistService



```
package com.example.rediswaitlistdemo.service;

import com.example.rediswaitlistdemo.model.Usuario;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class WaitlistService {

    private final RedisTemplate<String, String> redisTemplate;

    public WaitlistService(RedisTemplate<String, String> redisTemplate) {
        this.redisTemplate = redisTemplate;
    }

    public void agregarUsuarioALista(String eventold, Usuario usuario) {
        String key = "evento:" + eventold + ":espera";
        redisTemplate.opsForList().leftPush(key, usuario.getId() + ":" + usuario.getNombre());
    }

    public String obtenerSiguienteUsuario(String eventold) {
        String key = "evento:" + eventold + ":espera";
        return redisTemplate.opsForList().rightPop(key);
    }

    public List<String> obtenerListaEspera(String eventold) {
        String key = "evento:" + eventold + ":espera";
        return redisTemplate.opsForList().range(key, 0, -1);
    }
}
```

Explicación:

leftPush(key, value)

- Agrega un usuario **al inicio (izquierda)** de la lista (simulando una cola FIFO).
- **key** es la clave bajo la cual se almacena la lista en Redis.
- **value** es el valor que se va a insertar en la lista. En este caso, el valor es una cadena formada por la concatenación del ID y el nombre del usuario (usuario.getId() + ":" + usuario.getNombre()).

rightPop(key)

- Es una operación que elimina y devuelve el **último elemento (derecha)** de una lista en Redis.
- **key** es la clave bajo la cual se almacena la lista en Redis.

range(key, start, end)

- **range** es una operación que devuelve una **porción de la lista** almacenada en Redis, especificada por los índices start y end.
- **key** es la clave bajo la cual se almacena la lista en Redis.
- **start** es el índice inicial (basado en 0) desde el cual se comenzará a recuperar elementos.
- **end** es el índice final hasta el cual se recuperarán elementos.

Parámetros start y end:

- **start = 0:**
 - Indica que la recuperación de elementos comenzará desde el primer elemento de la lista (índice 0).
- **end = -1:**
 - En Redis, un valor de -1 para end significa "hasta el último elemento de la lista".
 - Por lo tanto, range(key, 0, -1) devuelve **todos los elementos de la lista**.

Paso 5: Crear el WaitlistController

Creamos el controller WaitlistController



```
package com.example.rediswaitlistdemo.controller;

import com.example.rediswaitlistdemo.model.Usuario;
import com.example.rediswaitlistdemo.service.WaitlistService;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/eventos")
public class WaitlistController {

    private final WaitlistService waitlistService;

    public WaitlistController(WaitlistService waitlistService) {
        this.waitlistService = waitlistService;
    }

    @PostMapping("/{eventold}/unirse")
    public String agregarUsuario(@PathVariable String eventold, @RequestBody Usuario usuario) {
        waitlistService.agregarUsuarioALista(eventold, usuario);
        return usuario.getNombre() + " se ha unido a la lista de espera del evento " + eventold;
    }

    @GetMapping("/{eventold}/siguiente")
    public String obtenerSiguienteUsuario(@PathVariable String eventold) {
        String usuario = waitlistService.obtenerSiguienteUsuario(eventold);
        return (usuario != null) ? "Siguiente en la lista: " + usuario : "No hay usuarios en la lista de espera.";
    }

    @GetMapping("/{eventold}/espera")
    public List<String> obtenerListaEspera(@PathVariable String eventold) {
        return waitlistService.obtenerListaEspera(eventold);
    }
}
```


Explicación:

- **POST /eventos/{eventold}/unirse:** Agrega un usuario a la lista en Redis.
- **GET /eventos/{eventold}/siguiente:** Saca el siguiente usuario de la lista FIFO.
- **GET /eventos/{eventold}/espera:** Devuelve la lista de espera actual.

Paso 6: Ejecutar y Probar

- **POST** http://localhost:8080/eventos/matricula-idat/unirse

Respuesta esperada



Juan Silva se ha unido a la lista de espera del evento matricula-idat

POST

http://localhost:8080/eventos/matricula-idat/unirse

Send

ParamsAuthorizationHeaders (8)BodyScriptsTestsSettings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Beautify

1

{

2

"id": "U003",

3

"nombre": "Juan Silva"

4

}

5

BodyCookiesHeaders (5)Test Results

200 OK • 116 ms • 233 B • Save Response

Raw


Preview

Visualize

1 Juan Silva se ha unido a la lista de espera del evento matricula-idat

- GET http://localhost:8080/eventos/matricula-idat/espera

Respuesta esperada



[
 "U003:Juan Silva",
 "U0012:Juan Pérez",
 "U001:Juan Pérez"
]

GET

http://localhost:8080/eventos/matricula-idat/espera

Send

ParamsAuthorizationHeaders (6)BodyScriptsTestsSettings

Query Params

	Key	Value	Description	Bulk Edit
	Key	Value	Description	

BodyCookiesHeaders (5)Test Results

200 OK • 111 ms • 222 B • Save Response

{ } JSON

Preview

Visualize

1

[

2

"U003:Juan Silva",

3

"U0012:Juan Pérez",

4

"U001:Juan Pérez"

5

]

- GET http://localhost:8080/eventos/matricula-idat/siguiente

Respuesta esperada



Siguiente en la lista: U003:Juan Silva

GET

http://localhost:8080/eventos/matricula-idat/siguiente

Send

ParamsAuthorizationHeaders (6)BodyScriptsTestsSettingsCookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

BodyCookiesHeaders (5)Test Results

200 OK • 106 ms • 202 B • Save Response

RawPreviewVisualize

1 Siguiente en la lista: U003:Juan Silva