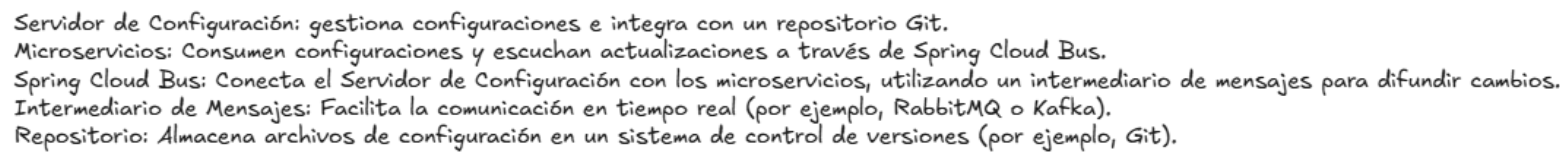
Diagrama

El contenido generado por IA puede ser incorrecto.



**INTRODUCCIÓN**

El objetivo de este proyecto es conseguir que los diferentes microservicios que tengamos puedan actualizar ciertos atributos en tiempo de ejecución, sin la necesidad de reiniciarlos. Esto se conseguirá mediante otros cuatro elementos.

Primero necesitaremos un repositorio que contendrá todas las configuraciones de los distintos micros; estos se pueden organizar en carpetas y en distintas ramas. De esta manera, podemos tener configuraciones distintas para cada parte del desarrollo.

Después, tendremos que crear un microservicio que actuará de Config-Server (CS). Los servicios aplicativos, al iniciarse, se comunicarán con este CS para obtener su respectiva configuración. Cuando esto ocurra, el CS se comunicará con el repositorio que hemos creado anteriormente para obtener el archivo YAML correspondiente y devolverlo al microservicio.

Ahora, cuando creemos los microservicios que simularán a los servicios aplicativos, deberemos añadir distintas dependencias para que actúen como clientes del Config-Server. De esta manera, al iniciarse, irán a buscar su respectiva configuración.

Finalmente, para conseguir la actualización en tiempo de ejecución, usaremos Spring Cloud Bus junto a un Message Broker de nuestra elección (RabbitMQ o Kafka). Para esta POC usaremos RabbitMQ, y este nos permitirá actualizar los diferentes atributos de los microservicios sin tener que reiniciarlos, añadiendo diferentes dependencias tanto en el config-server como en los microservicios.

Ahora entraremos más en detalle.

**IMPLEMENTACIÓN CONFIG-SERVER**

**REPOSITORIO:**

Para este ejemplo, he usado GitLab y he creado un repositorio para las configuraciones con la siguiente organización:

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

P Podemos observar que contamos con distintas carpetas, cada una de las cuales representa un perfil que los microservicios podrán escoger para obtener una configuración u otra.

Estos perfiles pueden servir, por ejemplo, para representar las diferentes partes del desarrollo. Además, hay una carpeta llamada “COMMON”, que contiene una configuración general para todos los microservicios en caso de que no tengan ningún perfil especificado.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Esta es la carpeta COMMON, que solo contiene un application.yml, el cual, como ya he comentado, es una configuración por defecto para los microservicios.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Ahora volvamos atrás, al inicio del repositorio y entraremos en la carpeta “dev”:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Aquí podemos observar que tenemos un archivo de configuración para cada microservicio que lo requiera.

Es importante destacar que el nombre del archivo YAML debe seguir el siguiente formato: {nombre\_de\_la\_aplicacion}-{profile}.yml, de esta manera el config-server podrá distinguir entre todas las configuraciones.

Veamos el user-service-dev.yml como ejemplo:

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Colocamos los atributos que consideremos para esta configuración y el puerto que queramos.

Para las demás carpetas o perfiles que hemos visto al inicio, siguen todos el mismo formato: {nombre\_de\_la\_aplicacion}-{profile}.yml.

Por último, también tenemos la opción de usar las Labels que ofrecen los repositorios. Podemos crear una label directamente o crear distintas ramas, que es lo que he hecho en este ejemplo. A partir del main, creamos dos ramas nuevas: public-config y private-config.

Esto nos permitirá tener muchas más configuraciones; podemos indicarle a un microservicio que tenga la configuración privada o la pública. Dentro de estas ramas, el formato de perfiles será el mismo. Si no escogemos ninguna label, la main será la que se abra por defecto.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Hay que recordar que si el repositorio es privado, para poder acceder fácilmente a través de Spring Boot, podemos crear un token personal que nos permita acceder a repositorios, solo de lectura. De esta manera podremos usarlo de contraseña en nuestro Config-Server.

**CONFIG-SERVER**

Para el config-server tendremos la siguiente estructura:

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

En el pom.xml añadiremos las siguientes dependencias:

Texto

El contenido generado por IA puede ser incorrecto.

Y en el Application añadiremos la siguiente anotación:

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Por ultimo en el application.yml añadiremos la siguiente configuración:

Texto

El contenido generado por IA puede ser incorrecto.

Los search-path se encarga de saber los distintos perfiles que existen. Se carga de arriba hacia abajo. Es decir, si ponemos primero el common este será el que haya por defecto y viceversa.

Esta configuración se conecta al repositorio, usa el token como contraseña ya que es privado ya demás añade seguridad básica de spring. Los micros deberán usar el nombre y contraseña para acceder a las configuraciones.

**MICRO-SERVICIOS**

Para esta POC, tendremos dos microservicios de ejemplo:

* User-service
* Product-service

Estos contienen lo básico para poder crear Usuarios y Productos a través del API Rest y que se guarden en una base de datos local como puede ser H2. También contienen un application.yml donde deberemos añadir diferentes configuraciones que explicare mas adelante.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Primero deberemos añadir las siguientes dependencias:

Texto

El contenido generado por IA puede ser incorrecto.

Y en el application.yml añadiremos lo siguiente:

Texto

El contenido generado por IA puede ser incorrecto.

Es importante que el nombre coincida con los archivos de configuración. Vemos que podemos establecer los perfiles de los que queremos la configuración. Al igual que en el config.server, estos se cargan de arriba abajo. Por lo tanto, primero cargara el test, luego el dev y por ultimo se quedara con la configuración de prod. Se van sobrescribiendo.

Añadimos la URL al config-server y ponemos la contraseña y nombre (root:root). Finalmente podemos especificarle que label queremos usar. (En nuestro caso, la de la main, private o public).

**IMPLEMENTACIÓN SPRING CLOUD BUS & RABBIT-MQ**

Ahora solo queda la parte de actualizar los micros con la base que ya tenemos creada.

**CLIENT ACTUATOR:**

Para esto añadiremos la siguiente dependencia en los MICROSERVICIOS:

Texto

El contenido generado por IA puede ser incorrecto.

Ahora en el archivo de “Custom Properties” que tenemos en nuestros microservicios, podemos añadir todos los atributos que queramos tener. Además añadiremos la anotación de @RefreshScope.

Texto

El contenido generado por IA puede ser incorrecto.

Con el @ConfigurationProperties y el prefijo, le estamos indicando donde estarán situados estos atributos en los archivos de configuración. Es decir, nosotros tenemos “custom” por lo tanto cuando se inicie el micro y obtenga la configuración, buscara el prefijo “custom” y asignara los valores respectivamente en esta clase.

Texto

El contenido generado por IA puede ser incorrecto.

Cabe destacar que en el archivo yaml, los nombre tiene que estar separados con guiones, pero en la clase de atributos del micro tiene que estar en camelCase. Para que Spring Cloud los identifique por si mismo.

Ahora el actuator que hemos añadido con las dependencias nos proporciona diferentes endpoints que podemos utilizar para actualizar el microservicio. Es posible que por defecto estos estén bloqueados, es por esto que podemos añadir la siguiente configuración en el application.yml del microservicio:

Texto

El contenido generado por IA puede ser incorrecto.

De esta manera exponemos todos los endpoints. Ahora mismo si inicamos todo, y llamamos a este endpoint de este microservicio :

http://localhost:{puerto}/actuator/refresh

Volverá a buscar su configuracion y actualizara sus atributos si han cambiado. Sin la neceseidad de reiniciar el micro.

**SPRING CLOUD BUS & RABBIT\_MQ:**

Priemero vamos a configurar rabbitMQ. Una vez lo tengamos instalado si nos da problemas, podemos parar el servicio para volver a iniciarlo: (Se supone que con la instalación viene una terminal de rabbir que puedes ejecutar, pero en mi caso no funcionaba, asi que use la terminal de windows.)

Texto

El contenido generado por IA puede ser incorrecto.

Antes o despues de iniciar el servicio, no lo recuerdo bien. Debemos añadir el siguiente plug in de rabbit para poder acceder a la consola:

**C:\Program Files\RabbitMQ Server\rabbitmq\_server-4.1.0\sbin> *rabbitmq-plugins.bat enable rabbitmq\_management***

Ahora podemos acceder a la consola de rabbitMQ para comprobar que funciona:   
<http://localhost:15672>

Accederemos a la consola de rabbit para poder crear los exchanges y las colas necesarias para nuestros micros:

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Una cola para cada microservicio que tengamos.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Ademas crearemos el siguiente Exchange y añadiremos los binding a las colas que hemos creado anteriormente, no tenemos que poner argumentos y de routing key debemos poner un **#**.

Ahora una vez tenemos rabbit funcionando. En los microservicios deberemos añadir la siguiente dependencia:

Texto

El contenido generado por IA puede ser incorrecto.

Ademas de la dependecia tuve que añadir una exclusion de commons-logging, ya que me generaba un conflicto con spring-jcl. (No estoy seguro de si es generad por otras dependencias, pero excluirlo aquí lo solucionaba).

Ahora añadiremos la siguiente configuracion en su application.yml:

Texto

El contenido generado por IA puede ser incorrecto.

En destination ponemos el nombre del exchange y en el group la cola que usaran. El nombre debe coincidir.

Para poder identificarlos mejor, les añadi perfiles dentro de rabbitMQ. Es por eso que el username es diferente a guest.

Ademas quise que tuvieran un nombre de conexión especifico, asi que podemos añadir lo siguiente en una clase de configuracion:

Texto

El contenido generado por IA puede ser incorrecto.

Ahora si iniciamos todo y llamamos al endpoint del actuator del microservicio:

/actuator/busrefresh

Se actualizaran todos los microservicios que esten suscritos. Sin tener que ir uno por uno.

Para finalizar, nos falta añadir un par de cosas en el config-server. Primero, añadiremos las siguientes dependencias:

Texto

El contenido generado por IA puede ser incorrecto.

Y ahora deberemos añadir la siguiente configuracion en su application.yml:

Texto

El contenido generado por IA puede ser incorrecto.

Ahora, con la dependencia de monitor que hemos añadido, nos proporciona un endpoint que podemos llamar desde un webHook de nuestro repositorio. Crearemos un WEBHOOK que se active cunado hagamos un push en cualquiera de los archivos de configuracion. Cuando se active, llamara al siguiente endpoint:

http://localhost:8888/monitor

Esto hara que el config-server busque que archivos se han actualizado y si afectan a los microservicios activos. Si es asi, actualizara solo los microservicios que tengan algo distinto de manera automatica, los demas no.

De esta manera solo tenemos que hacer un cambio en el repo para que se actualice todo solo.

Por ultimo, el monitor me daba problema de autorizacion cunado lo llamaba, por eso tuve que añadir la siguiente configuracion:

Texto

El contenido generado por IA puede ser incorrecto.

De esta manera lo ignoraba para el endpoint de monitor.

**EXTRA**

Los webhook no permiten llamar a localhost, por lo tanto deberiamos tener una herramienta de tunneling para exponer nuestro puerto local. O bien desplegarlo, solo si lo estamos probando en un entorno local como es mi caso.