Guía sobre el desarrollo del micro san-coplst-plardpcaslate

Este documento sirve como a modo de documentación sobre el funcionamiento del nuevo micro san-coplst-plardpcaslate

# Explicación del funcionamiento

Este nuevo micro fue desarrollado para gestionar aquellos mensajes del micro plard-pcas que producen los llamados timeout sobre una cola a la hora de hacer el envio

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Estos mensajes ahora mismo serán enviados a un nuevo topico Kafka, ya implementado llamado **SANES.COPLST.AUTHORIZATION.DELAYED**

El micro tiene 3 tareas principales

1. Lectura de un mensaje procedente del tópico, dicho mensaje será insertado en una lista. El mensaje tendrá asignado un tiempo de expiración, el cual esta indicado para gestionar el tiempo de permanencia en dicha lista.
2. Para cada uno de los mensajes que están en la lista, se deberá de ejecutar el nuevo método de la librería **sgt-concomp-globalpaycon** llamado receiveMessageNoWait, el cual obtiene el mensaje que se quedó pendiente de enviar y que produjo el timeout.

Una vez se obtenga el mensaje, se deberá de ejecutar el API:

<https://stratos-late-response-ops.test.uat.pci.np.santander-es.eu-west-1.santanderglobalcards.corp/stratos-late-response/late_reponse_pcas>

y finalmente se eliminará dicho mensaje de la lista de mensajes que han producido el timeout.

En el caso en que el método **receiveMessageNoWait**, no devuelva nada, el mensaje permanecerá en la lista hasta que finalice su tiempo de expiración. (Comentaron los compañeros que trabajan en la librería, que existe la posibilidad de que en una primera iteración de la lista, algunos mensajes no se obtengan en la primera llamada al método **receiveMessageNoWait** y sí que se obtengan en iteraciones siguientes)

1. Esta última tarea se encargará de eliminar los mensajes que ya están expirados y que no hemos podido obtener en las sucesivas llamadas al método **receiveMessageNoWait**

# Notas sobre la implementación

Para la implementación de este micro, se me pidió que las 3 tareas se ejecutaran de manera concurrente, de ahí que se haya empleado la interfaz **PooledObjectFactory**, la cual nos ayuda a manejar la concurrencia de las 3 tareas principales. La clase **ConnectionResourceFactory** implementa dicha interfaz.

La configuración de dicha concurrencia se encuentra en las properties:

Texto, Sitio web

El contenido generado por IA puede ser incorrecto.

Respecto al numero de llamadas que se hace al método **receiveMessageNoWait** para cada uno de los mensajes, no existe un numero fijo, ya que se harán tantas llamadas como sea posible hasta que el tiempo de expiración asociado al mensaje finalice, entonces será eliminado de la lista.

Para configurar el tiempo de expiración, hay que modificar la propertie:

Texto

El contenido generado por IA puede ser incorrecto.

El tiempo aquí está en segundos.

Por último, la lista de los mensajes es de tipo CopyOnWritteArrayList, la cual permite iterar y modificar a la vez (es decir ir actualizando la lista eliminando los mensajes ya expirados, mientras seguimos iterando). Esto lo que hace es evitar el error que me apareció y que fue:

Texto

El contenido generado por IA puede ser incorrecto.