## 

Análisis Librería PGE

# Objetivo

Evaluar la factibilidad de extraer el código "*core*" involucrado en el consumo de servicios a través de la PDI y convertirlo en librería. De esta forma, puede ser utilizado tanto por el Conector o por cualquier otro sistema que quiera utilizar los servicios de la PDI sin utilizar el Conector.

# Introducción

A grandes rasgos, el código que realiza la invocación a un servicio de la PGE se puede dividir en los siguientes pasos:

* Determinar los roles de seguridad a utilizar, en base al mensaje recibido
* Obtener los datos de los certificados de seguridad, asociados a la configuración del Conector
* Obtención del token SAML
* Invocación al Servicio, que involucra
  + Insertar token SAML en el mensaje recibido del Cliente
  + Agregar headers de WS-\* (por ejemplo WS-Addressing)
  + Invocar al endpoint

En base a los pasos mencionados, resulta necesario analizar cómo puede abstraerse la lógica, de manera que pueda ser utilizada como librería.

# 

# Librería PGE-Client

Vale destacar que en una versión previa del Conector, se encontraba disponible la librería *pge-client.jar*, la cual contaba con lógica necesaria para obtener un token SAML. Dicha librería dejó de utilizarse en la versión actual, y la forma de obtener el token SAML fue actualizada, pasando de usar *PicketLink* a *OpenSAML* cómo implementacion del standard.

# Propuesta de creación de Librería

Se propone analizar las clases que participan del procesamiento del mensaje, extraer aquellas que sea de interés que se sean independientes del Conector, y armar una librería a partir de ellas.

Esta nueva librería definirá un punto de entrada, el cual servirá de método principal, solicitando los siguientes datos:

* Mensaje SOAP a enviar a la PDI
* Información necesaria para la obtención del token SAML (rol, keystores, truststores, contraseñas, etc)
* Url lógica del servicio a invocar

Analizando el código actual del Conector, resulta de interés considerar las clases que se mencionan a continuación:

* *WsInvokeService* - Obtención del token SAML e inyección del mismo en el mensaje *SOAP* a enviar. También se encarga de agregar los headers de *WS-Addressing*
* *ConectorRestTemplate* - Invocación del servicio, habiendo utilizado *WsInvokeService* previamente

# Desafíos encontrados

Sin haber realizado un análisis detallado del requerimiento, ya se pueden observar algunos desafíos:

* El uso de *WsInvokeService* y *ConectorRestTemplate* son utilizados de forma independiente. O sea, en el *pipeline* expuesto por *spring-files-config.xml*, primero se ejecuta *WsInvokeService*, y más adelante *ConectorRestTemplate*. En caso de extraer dichos servicios en una librería, el *pipeline* también debería ser actualizado
* Las clases recién mencionadas hacen uso de librerías de *Spring* (por ejemplo *RestTemplate*, *MessageBuilder*, *Message*, etc). Por lo tanto, es necesario analizar si conviene mantener su uso, y hacer que la librería funcione sin un servidor de aplicaciones, o de lo contrario quitar las depencia a *Spring* y actualizar el código necesario para ello.

Pasos concretos a realizar:

- Revisar pge-client.jar y los fuentes proporcionados por Andrés, y confirmar que dichos fuentes solamente obtienen el token SAML, y no realizan la invocación al servicio

- En caso de confirmar lo anterior, analizar la forma de migrar el código actual del Conector a una librería Java, teniendo en cuenta:

- Intentar de respetar interfaz original

- Intentar que sea standalone

- También analizar la posibilidad de que la librería soporte SAML 2.0 (o sea soporta 1.1 y 2.0)

A futuro, se puede retomar la idea de también migrar la lógica de invocación de servicios.