|  |
| --- |
| Unidad de Modernización y Gobierno Electrónico – Minsegpres |
| PISEE - REST |
| Nuevas Integraciones |
|  |

**Marzo 2014**

**Elaborado por:**Daniel Troncoso R.

**Revisado por:**

Equipo PISEE

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Manual para nuevas integración de servicios vía REST sobre los WS integrados en PISEE.  **Información del Documento:**   |  |  |  |  | | --- | --- | --- | --- | | **Nombre del Informe:** | Manual Usuario | | | | **Responsable del informe:** | Daniel Troncoso Rojas | | | | **Versión del Documento:** | 1.0 | **Fecha de la versión del documento:** | 1.0 | |  | | | | | **Preparado por:** | Daniel Troncoso Rojas | **Fecha de Creación:** | 19-03-2014 | | **Revisado por:** | Equipo Pisee | **Fecha de Revisión:** |  |   **Control de Cambios:**   |  |  |  |  | | --- | --- | --- | --- | | **Versión:** | **Fecha:** | **Revisado por:** | **Descripción:** | | 1.0 | 19-03-2014 | Daniel Troncoso R. | Creación de documento | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |

[1 Objetivo 3](#_Toc382907536)

[2 Pre Condiciones 4](#_Toc382907537)

[2.1 Servicio Integrado en la PISEE 4](#_Toc382907538)

[2.2 XML de entrada vía SOAP-UI 4](#_Toc382907539)

[2.3 Generación TOKEN 4](#_Toc382907540)

[3 Integración Servicios REST 7](#_Toc382907541)

[3.1 Crear Archivos de Configuración 7](#_Toc382907542)

[3.1.1 Crear Template Entrada de WS 8](#_Toc382907543)

[3.1.2 Crear Template Salida de WS 13](#_Toc382907544)

[3.1.3 Crear Template KeyStore de WS 17](#_Toc382907545)

[3.2 Modificar Aplicación 18](#_Toc382907546)

[3.2.1 Modificar Codigo Fuente 18](#_Toc382907547)

[3.2.2 Modificar Archivo “applicationContext.xml” 22](#_Toc382907548)

[3.2.3 Empaquetar Aplicación 23](#_Toc382907549)

[4 Pos Condiciones 24](#_Toc382907550)

[4.1 Comprobar Servicio por URL 24](#_Toc382907551)

[4.2 Revisar Trazabilidad en Backoffice 26](#_Toc382907552)

# Objetivo

Tal como se tiene conocimiento, actualmente existen una serie de Web Services integrados en la Plataforma, todos estos bajo protocolo SOAP tal como lo dicen los decretos que regulan la utilización de la PISEE, sin embargo aún existen ciertos inconvenientes para su utilización por nuevos organismo que desean incorporarse a la Plataforma, es en este escenario que se crea la aplicación de PISEE – REST con la cual es posible exponer los mismo actualmente integrados vía REST.

Teniendo claro el enfoque de esta aplicación es necesario detallar cual es el procedimiento para exponer los servicios SOAP de la Plataforma vía REST lo cual se explica a continuación. Para esto es necesario contar con ciertas condiciones previas al comienzo del desarrollo para luego entrar de lleno en la integración del servicio finalizando con las actividades para comprobar que la exposición del servicio ha sido correcta.

Finalmente cabe señalar que toda la información relacionada al código fuente y como ambientar la aplicación se encuentra en el documento “[REST] - Documentación Técnica v.1.0.docx”

# Pre Condiciones

Antes de realizar la operación de integrar un SOAP y exponerlo vía REST es necesario comprobar y realizar una serie de condiciones las cuales se detallan a continuación:

## Servicio Integrado en la PISEE

El insumo principal para poder exponer un servicio vía REST en la PISEE es que dicho servicio ya esté integrado en la Plataforma, se debe recordar que esta nueva aplicación es un “envoltorio” de la Plataforma la cual procesa la respuesta SOAP otorgada por la PISEE y la transforma en una respuesta JSON.

Se entiende que cuando se menciona “servicio integrado en la PISEE” éste WS se encuentra en producción, con registro de traza, es decir, con consumo activo por lo tanto es posible exponerlo vía REST. Por lo tanto todo lo relacionado con: Certificados digitales, credenciales del organismo consumidor y códigos de trámites e instituciones se encuentra resuelto.

## XML de entrada vía SOAP-UI

Para que un servicio se encuentre integrado en la PISEE se deben haber realizado pruebas de dicha integración, estas pruebas son realizadas en la aplicación SOAP-UI con XML de entrada de los servicios, por lo tanto se entiende que si el servicio se encuentra habilitado ya se cuenta con dichos archivos de pruebas los cuales se transforman en un insumo para exponer un servicio vía REST.

## Generación TOKEN

En palabras simples un TOKEN es un componente léxico (cadena de caracteres) el cual tiene un significado coherente en cierto lenguaje de programación o en cierta aplicación, por lo tanto para poder agrupar la combinación “Proveedor – Servicio y Consumidor – Tramite” es que se origina la creación de un TOKEN.

Para crear un TOKEN se debe ingresar a la aplicación Backoffice con el perfil de Administrador y luego ir a la opción de “Administración > Token REST” en dicha página se debe hacer clic en el link “Agregar Nuevo Token REST” y en dicha opción rellenar los campos requeridos.

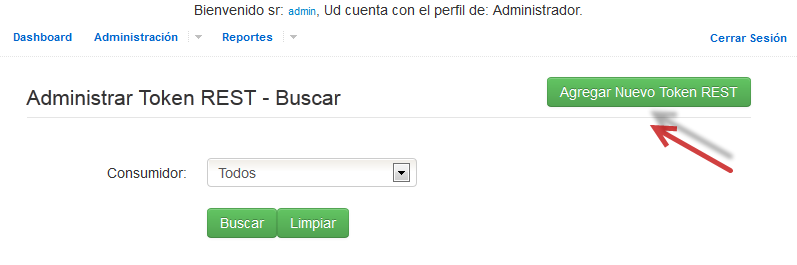


Figura 1 – Administración > Token REST

Luego de hacer clic en dicha opción se abre la siguiente página (señalar que la siguiente página corresponde a la version 0.0.16 de Backoffice por lo tanto pueden existir diferencias visuales).

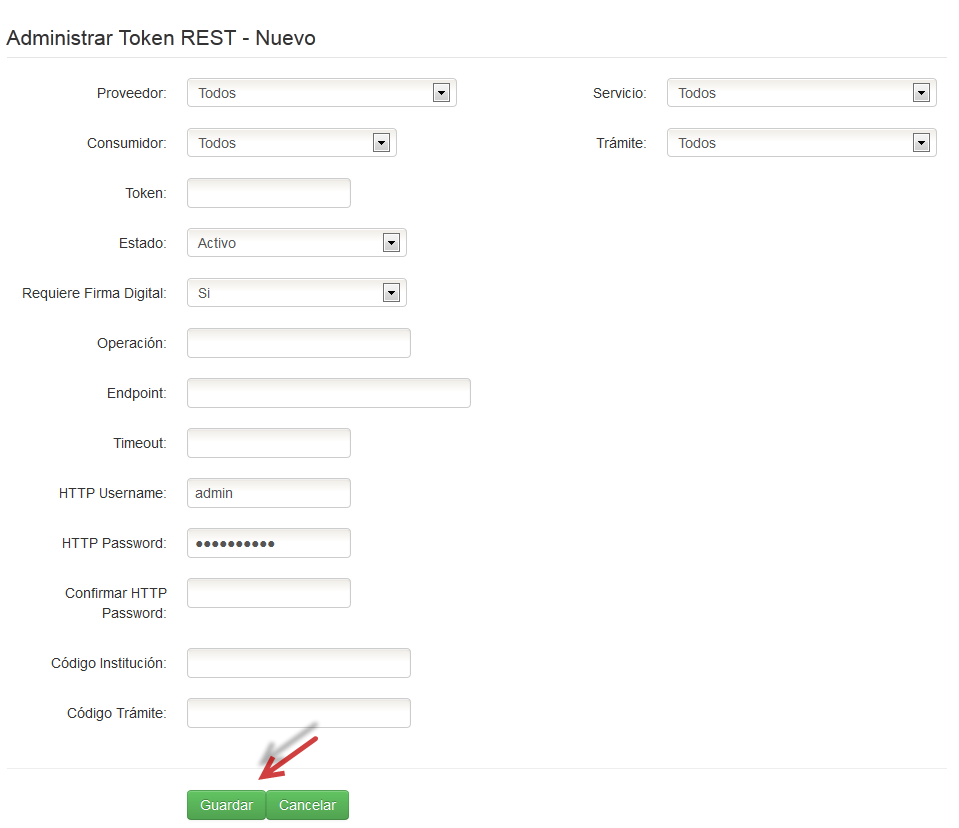


Figura 2 – Crear Nuevo TOKEN

Como se puede identificar en la figura anterior existen una serie de campos los cuales se deben llenar, estos campos se detallan a continuación.

* Proveedor: Organismo el cual expone el servicio
* Servicio: Corresponde al servicio a consumir
* Consumidor: Organismo que desea llamar al servicio
* Trámite: Motivo por el cual se desea consumir el servicio
* Token: Corresponde a una cadena de caracteres la cual asocia al Proveedor-Servicio-Consumidor-Tramite. Debe ser alfanumérico.
* Estado: Indica si se encuentra Activo o bloqueado el TOKEN
* Operación: En la operación del endpoint del servicio
* Endpoint: URI en donde se encuentra expuesto el servicio
* Timeout: Tiempo en milisegundos para el Timeout
* HTTP Username: Credencial del organismo consumidor
* HTTP Password: Credencial del organismo consumidor
* Requiere Firma Digital: Indica si dicho servicio requiere firma digital
* Código Institución: Utilizado para la generación del ID Sobre
* Código Trámite: Utilizado para la generación del ID Sobre

# Integración Servicios REST

El proceso de integración se encuentra dividido en dos secciones la primera que hace mención a la creación de archivos de configuración o plantillas y la segunda corresponde a los cambios que se deben realizar en el código fuente de la aplicación.

## Crear Archivos de Configuración

En el proceso de una nueva integración de un servicio de la Plataforma e formato REST, es necesario crear diferentes archivos de configuración ya sea para crear el archivo XML de entrada del servicio, para procesar la respuesta o para contener los datos necesarios para leer la respuesta cuando se encuentra firmada.

La necesidad de la creación de estos archivos es originada por el tipo de negocio de los servicios web en los cuales el elemento “cuerpo” de los servicios depende del proveedor y no es estándar por tanto no se puede automatizar la lectura de este elemento dado que no en todos los servicios se encuentra con un archivo descriptor que señale su estructura, este escenario aplica tanto para el archivo XML de entrada y salida.

El nombre de archivo para estos casos se divide en dos secciones en donde la primera indica el servicio para el cual se utiliza la plantilla y la segunda sección indica el tipo de archivo, de esta forma se pueden encontrar los siguientes casos:

* SRCEI\_CertificadoNacimientoGENCHI**\_input.xml**: Corresponde al archivo que se utiliza como plantilla para consumir los servicios SOAP de la Plataforma actual.
* SRCEI\_CertificadoNacimientoGENCHI**\_output.xml**: Corresponde a la configuración para procesar la respuesta del servicio.
* SRCEI\_CertificadoNacimientoGENCHI**\_keystore.xml**: corresponde a la configuración del KeyStore para los servicios que lo requieran.

Todos los archivos de configuración se deben copiar a la carpeta: “%JBOSS\_INSTANCE%/ deploy/base/piseeConf/rest\_services” entendiéndose como “%JBOSS\_INSTANCE%” la ruta donde se encuentra instalado y configurado la instancia JBoss.

### Crear Template Entrada de WS

Tal como se mencionó anteriormente uno de los insumos con que se debe contar antes de la integración de un nuevo web servicio vía REST es el archivo XML de entrada vía SOAP-UI, dado que con esto se genera el Template para la entrada del servicio.

Para ejemplificar de mejor forma el proceso de integración se tomará como base el servicio del “Disponibilidad Licencia Enseñanza Media” del proveedor “Mineduc” desde el XML de entrada vía SOAP-UI hasta la generación del Template. El archivo se muestra a continuación:

|  |
| --- |
| <soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:aem="http://valida.aem.gob.cl" xmlns:xd="http://www.w3.org/2000/09/xmldsig#" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">  <soapenv:Header/>  <soapenv:Body>  <aem:sobre>  <aem:encabezado>  <aem:idSobre>000000000000000000000009999</aem:idSobre>  <aem:fechaHora>2012-02-27T14:19:29</aem:fechaHora>  <aem:proveedor>  <aem:nombre>MINEDUC</aem:nombre>  <aem:servicios>  <aem:servicio>DISPONIBILIDAD LICENCIA ENSENANZA MEDIA</aem:servicio>  <aem:respuestaServicio>  <aem:estado>SI</aem:estado>  <aem:glosa>RESPUESTA EXITOSA</aem:glosa>  </aem:respuestaServicio>  </aem:servicios>  </aem:proveedor>  <aem:consumidor>  <aem:nombre>MIDEPLAN</aem:nombre>  <aem:tramite>PLAN DE SUPERVISION DE ESCOLARIDAD EN LA FPS</aem:tramite>  <aem:certificado>  <xd:X509Data>  <xd:X509IssuerSerial>  <xd:X509IssuerName>IN</xd:X509IssuerName>  <xd:X509SerialNumber>0</xd:X509SerialNumber>  </xd:X509IssuerSerial>  </xd:X509Data>  </aem:certificado>  </aem:consumidor>  <aem:fechaHoraReq>2012-02-27T14:19:29</aem:fechaHoraReq>  <aem:emisor>MIDEPLAN</aem:emisor>  <aem:metadataOperacional>  <aem:estadoSobre>00</aem:estadoSobre>  <aem:glosaSobre>TRANSACCION EXITOSA</aem:glosaSobre>  </aem:metadataOperacional>  </aem:encabezado>  <aem:cuerpo>  <aem:documento>  <aem:EntradaDisponibilidadLicencias>  <aem:Run>  <aem:numero>12909801</aem:numero>  <aem:dv>5</aem:dv>  </aem:Run>  </aem:EntradaDisponibilidadLicencias>  </aem:documento>  </aem:cuerpo>  </aem:sobre>  </soapenv:Body>  </soapenv:Envelope> |

Tabla 1 – XML Entrada SOAP-UI

Con base en este archivo, se pueden mencionar varias características las cuales se presentan a continuación en diferentes colores para una mejor explicación:

|  |
| --- |
| <soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:aem="http://valida.aem.gob.cl" xmlns:xd="http://www.w3.org/2000/09/xmldsig#" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">  <soapenv:Header/>  <soapenv:Body>  <aem:sobre>  <aem:encabezado>  <aem:idSobre>000000000000000000000009999</aem:idSobre>  <aem:fechaHora>2012-02-27T14:19:29</aem:fechaHora>  <aem:proveedor>  <aem:nombre>MINEDUC</aem:nombre>  <aem:servicios>  <aem:servicio>DISPONIBILIDAD LICENCIA ENSENANZA MEDIA</aem:servicio>  <aem:respuestaServicio>  <aem:estado>SI</aem:estado>  <aem:glosa>RESPUESTA EXITOSA</aem:glosa>  </aem:respuestaServicio>  </aem:servicios>  </aem:proveedor>  <aem:consumidor>  <aem:nombre>MIDEPLAN</aem:nombre>  <aem:tramite>PLAN DE SUPERVISION DE ESCOLARIDAD EN LA FPS</aem:tramite>  <aem:certificado>  <xd:X509Data>  <xd:X509IssuerSerial>  <xd:X509IssuerName>IN</xd:X509IssuerName>  <xd:X509SerialNumber>0</xd:X509SerialNumber>  </xd:X509IssuerSerial>  </xd:X509Data>  </aem:certificado>  </aem:consumidor>  <aem:fechaHoraReq>2012-02-27T14:19:29</aem:fechaHoraReq>  <aem:emisor>MIDEPLAN</aem:emisor>  <aem:metadataOperacional>  <aem:estadoSobre>00</aem:estadoSobre>  <aem:glosaSobre>TRANSACCION EXITOSA</aem:glosaSobre>  </aem:metadataOperacional>  </aem:encabezado>  <aem:cuerpo>  <aem:documento>  <aem:EntradaDisponibilidadLicencias>  <aem:Run>  <aem:numero>12909801</aem:numero>  <aem:dv>5</aem:dv>  </aem:Run>  </aem:EntradaDisponibilidadLicencias>  </aem:documento>  </aem:cuerpo>  </aem:sobre>  </soapenv:Body>  </soapenv:Envelope> |

Tabla 2 – XML Entrada SOAP-UI

El detalle de las características es el siguiente:

Color Azul: Son todos los elementos “fijos” que se encuentran definidos en el esquema del Sobre, para todos los servicios integrados en la Plataforma son los mismos elementos.

Color Rojo: Son todos los elementos “variables”, es decir, los que dependen del servicio que expone el proveedor, estos elementos varían según el servicio.

Color Verde: Son los datos que componen el encabezado del sobre, con estos se comprueba la autorización para el consumo del servicio en particular.

Una vez analizado el contenido del archivo se realiza la diferenciación de los parámetros de la cabecera y del cuerpo en donde estos último son los “variables” para cada servicio y los primeros son la información relacionada a la autorización, por tanto aplicando la tecnología de “Expression Language” se configura este Template quedando de la siguiente forma:

|  |
| --- |
| <soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:aem="http://valida.aem.gob.cl" xmlns:xd="http://www.w3.org/2000/09/xmldsig#" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">  <soapenv:Header/>  <soapenv:Body>  <aem:sobre>  <aem:encabezado>  <aem:idSobre>ELPISEE.inputParameter.**headerParameters**['id\_sobre']</aem:idSobre>  <aem:fechaHora>ELPISEE.inputParameter.**headerParameters**['fecha\_hora']</aem:fechaHora>  <aem:proveedor>  <aem:nombre>ELPISEE.inputParameter.**headerParameters**['nombre\_proveedor']</aem:nombre>  <aem:servicios>  <aem:servicio>ELPISEE.inputParameter.**headerParameters**['nombre\_servicio']</aem:servicio>  <aem:respuestaServicio>  <aem:estado>SI</aem:estado>  <aem:glosa>RESPUESTA EXITOSA</aem:glosa>  </aem:respuestaServicio>  </aem:servicios>  </aem:proveedor>  <aem:consumidor>  <aem:nombre>ELPISEE.inputParameter.**headerParameters**['nombre\_consumidor']</aem:nombre>  <aem:tramite>ELPISEE.inputParameter.**headerParameters**['nombre\_tramite']</aem:tramite>  <aem:certificado>  <xd:X509Data>  <xd:X509IssuerSerial>  <xd:X509IssuerName>IN</xd:X509IssuerName>  <xd:X509SerialNumber>0</xd:X509SerialNumber>  </xd:X509IssuerSerial>  </xd:X509Data>  </aem:certificado>  </aem:consumidor> <aem:fechaHoraReq>ELPISEE.inputParameter.**headerParameters**['fecha\_hora']</aem:fechaHoraReq>  <aem:emisor>ELPISEE.inputParameter.**headerParameters**['nombre\_consumidor']</aem:emisor>  <aem:metadataOperacional>  <aem:estadoSobre>00</aem:estadoSobre>  <aem:glosaSobre>REST</aem:glosaSobre>  </aem:metadataOperacional>  </aem:encabezado>  <aem:cuerpo>  <aem:documento>  <aem:EntradaDisponibilidadLicencias>  <aem:Run>  <aem:numero>ELPISEE.inputParameter.**bodyParameters**['rut']</aem:numero>  <aem:dv>ELPISEE.inputParameter.**bodyParameters**['dv']</aem:dv>  </aem:Run>  </aem:EntradaDisponibilidadLicencias>  </aem:documento>  </aem:cuerpo>  </aem:sobre>  </soapenv:Body>  </soapenv:Envelope> |

Tabla 3 – XML Entrada SOAP-UI

El detalle de las características es el siguiente:

Color Naranjo: Son todos los valores que se completan en los elementos “fijos”, estos valores son completados por la solución PISEE – REST, algunos son calculados como es el caso del valor “id\_sobre” y otros son extraídos de los datos asociados al TOKEN. Finalmente esta estructura es igual para todos los servicios es por esto que los valores se encuentran en la colección **headerParameters** a diferencia de los campos del elemento “documento” que se encuentran en la colección **bodyParameters.**

Color Canela: Son los valores insertados en los elementos “variables”, es decir, los que dependen del servicio. A diferencia de los parámetros del encabezado los valores que se encuentran en la colección **bodyParameters** deben tener el mismo nombre en la clase JAVA que recibe las peticiones, este detalle se deja en el próximo apartado.

En resumen, las características más importantes del Template de entrada son las siguientes:

* Los datos del elemento “encabezado” son iguales para todos los servicios y están bajo la colección **headerParameters.**
* Los datos del elemento “documento” varían según la lógica del servicio y por tanto deben tener los mismos nombres que en la clase JAVA y se encuentran en la colección **bodyParameters.**
* El nombre del archivo debe terminar con “**\_input.xml**” para diferenciar que es un Template de entrada.

### Crear Template Salida de WS

Bajo el ya mencionado pre requisito de contar con un archivo XML de SOAP-UI, se espera que también se cuente con la salida del servicio, siguiendo con el mismo ejemplo se deja a continuación la salida exitosa del servicio de “Disponibilidad Licencia Enseñanza Media”.

|  |
| --- |
| <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">  <soapenv:Body>  <sobre xmlns="http://valida.aem.gob.cl">  <encabezado>  <idSobre>000000000000000000000009999</idSobre>  <fechaHora>2014-03-17T11:46:21.417Z</fechaHora>  <proveedor>  <nombre>MINEDUC</nombre>  <servicios>  <servicio>DISPONIBILIDAD LICENCIA ENSENANZA MEDIA</servicio>  <respuestaServicio>  <estado>SI</estado>  <glosa>RESPUESTA EXITOSA</glosa>  </respuestaServicio>  </servicios>  </proveedor>  <consumidor>  <nombre>MIDEPLAN</nombre>  <tramite>PLAN DE SUPERVISION DE ESCOLARIDAD EN LA FPS</tramite>  <certificado>  <ns1:X509Data xmlns:ns1="http://www.w3.org/2000/09/xmldsig#">  <ns1:X509IssuerSerial>  <ns1:X509IssuerName>IN</ns1:X509IssuerName>  <ns1:X509SerialNumber>0</ns1:X509SerialNumber>  </ns1:X509IssuerSerial>  </ns1:X509Data>  </certificado>  </consumidor>  <fechaHoraReq>2014-03-17T11:46:21.417Z</fechaHoraReq>  <emisor>MINEDUC</emisor>  <metadataOperacional>  <estadoSobre>00</estadoSobre>  <glosaSobre>TRANSACCION EXITOSA</glosaSobre>  </metadataOperacional>  </encabezado>  <cuerpo>  <documento>  <aem:LicenciaEnsenanzaMedia ns2:schemaLocation="http://valida.aem.gob.cl/documentales/MINEDUC/DisponibilidadLicenciaEnsenanzaMedia-v1-0.xsd" xmlns:aem="http://valida.aem.gob.cl" xmlns:ns2="xsi">  <aem:RegistraLicencia>true</aem:RegistraLicencia>  </aem:LicenciaEnsenanzaMedia>  </documento>  </cuerpo>  </sobre>  </soapenv:Body>  </soapenv:Envelope> |

Tabla 4 – XML Salida SOAP-UI

Como se puede apreciar la estructura es la misma que el XML de entrada, es decir, existen los elementos de encabezado y cuerpo en donde en este último se encuentra los datos que retornan del servicio.

|  |
| --- |
| <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">  <soapenv:Body>  <sobre xmlns="http://valida.aem.gob.cl">  <encabezado>  <idSobre>000000000000000000000009999</idSobre>  <fechaHora>2014-03-17T11:46:21.417Z</fechaHora>  <proveedor>  <nombre>MINEDUC</nombre>  <servicios>  <servicio>DISPONIBILIDAD LICENCIA ENSENANZA MEDIA</servicio>  <respuestaServicio>  <estado>SI</estado>  <glosa>RESPUESTA EXITOSA</glosa>  </respuestaServicio>  </servicios>  </proveedor>  <consumidor>  <nombre>MIDEPLAN</nombre>  <tramite>PLAN DE SUPERVISION DE ESCOLARIDAD EN LA FPS</tramite>  <certificado>  <ns1:X509Data xmlns:ns1="http://www.w3.org/2000/09/xmldsig#">  <ns1:X509IssuerSerial>  <ns1:X509IssuerName>IN</ns1:X509IssuerName>  <ns1:X509SerialNumber>0</ns1:X509SerialNumber>  </ns1:X509IssuerSerial>  </ns1:X509Data>  </certificado>  </consumidor>  <fechaHoraReq>2014-03-17T11:46:21.417Z</fechaHoraReq>  <emisor>MINEDUC</emisor>  <metadataOperacional>  <estadoSobre>00</estadoSobre>  <glosaSobre>TRANSACCION EXITOSA</glosaSobre>  </metadataOperacional>  </encabezado>  <cuerpo>  <documento>  <aem:LicenciaEnsenanzaMedia ns2:schemaLocation="http://valida.aem.gob.cl/documentales/MINEDUC/DisponibilidadLicenciaEnsenanzaMedia-v1-0.xsd" xmlns:aem="http://valida.aem.gob.cl" xmlns:ns2="xsi">  <aem:RegistraLicencia>true</aem:RegistraLicencia>  </aem:LicenciaEnsenanzaMedia>  </documento>  </cuerpo>  </sobre>  </soapenv:Body>  </soapenv:Envelope> |

Tabla 5 – XML Salida SOAP-UI

El detalle de las características es el siguiente:

Color Azul: Son todos los elementos “fijos” que se encuentran definidos en el esquema del Sobre, para todos los servicios integrados en la Plataforma son los mismos elementos.

Color Verde: Son los datos que componen el encabezado del sobre, con estos se comprueba la autorización para el consumo del servicio en particular.

Color Rojo: Son todos los elementos “variables”, es decir, los que dependen del servicio que expone el proveedor, estos elementos varían según el servicio.

Como se puede apreciar, en todos los servicios integrados de la Plataforma, la sección de “encabezado” es la misma por tanto no es necesario realizar su configuración a diferencia de los elementos que se encuentran en el elemento “cuerpo” los cuales si se debe configurar. En este caso es solo un dato que retorna el servicio el cual viene en el elemento de nombre “RegistraLicencia” que a su vez está inmerso en el elemento “LicenciaEnsenanzaMedia” para poder diferenciar en donde se encuentra almacenado el dato se utiliza XPATH, de esta forma el archivo de configuración para procesar la salida queda de la siguiente forma:

|  |
| --- |
| <campos>  <campo>  <nombre>TieneLicencia</nombre>  <ruta>Body/sobre/cuerpo/documento/LicenciaEnsenanzaMedia/RegistraLicencia</ruta>  </campo>  </campos> |

Tabla 6 – Template Salida

La estructura de este archivo es la siguiente:

Por cada dato que se desea extraer de la respuesta se debe crear un nodo “campo” en donde el elemento “nombre”, tal como lo indica, corresponde al nombre del campo en el JSON que retorna el servicio y el campo “ruta” indica la ruta XPATH de la respuesta del servicio

En resumen, las características más importantes del Template de Salida son las siguientes:

* Por cada dato que retorna el servicio se debe crear un elemento “campo”.
* El XPATH no debe contener el prefijo y debe comenzar desde el elemento “Body”
* El nombre del archivo debe terminar con “**\_output.xml**” para diferenciar que es un Template de salida.

### Crear Template KeyStore de WS

Para los servicios que cuentan con la utilización de firma, se debe configurar un archivo para poder firmar y descifrar el mensaje, es por esto que a continuación se explica cómo se encuentra compuesto el archivo de configuración, en relación a los elementos y el significado de cada uno de estos:

|  |
| --- |
| <keystore>  <keystore\_type> </keystore\_type>  <keystore\_file> </keystore\_file>  <keystore\_pass> </keystore\_pass>  <private\_key\_alias> </private\_key\_alias>  <private\_key\_pass> </private\_key\_pass>  </keystore> |

Tabla 7 – Template Keystore

El detalle de cada uno de los campos es el siguiente:

* El campo “keystore\_type”: Corresponde al tipo de certificado, ejemplo “PKCS12”.
* El campo “keystore\_file”: Corresponde a la ruta en donde se encuentra almacenado el certificado.
* El campo “keystore\_pass”: Corresponde a la contraseña del certificado.
* El campo “private\_key\_alias”: Corresponde al alias que tiene el certificado en su información.
* El campo “private\_key\_pass”: Corresponde a la contraseña del alias que tiene el certificado.

En resumen, las características más importantes del Template de Keystore son las siguientes:

* Por cada dato servicio que utilice certificado digital se debe configurar un archivo.
* Se debe conocer toda la información del certificado, incluyendo su password.
* El nombre del archivo debe terminar con “**\_keystore.xml**” para diferenciar que es un Template de salida.

## Modificar Aplicación

Luego de crear los diferentes archivos de configuración comienza el proceso de realizar las modificaciones necesarias, a nivel de código fuente, para que la solución pueda exponer un servicio vía REST.

### Modificar Codigo Fuente

Para poder exponer un servicio vía REST es necesario modificar los dos proyectos de la solución, es decir, se debe trabar en los proyectos “rest-business” y “rest-web”.

En el proyecto “rest-business” el cambio es menor dado que solo se debe modificar la clase “cl.gob.minsegpres.pisee.rest.business. ProveedoresServicios”, en donde se debe agregar una nueva variable que indica cual es el servicio a exponer. Si bien este nombre a primera impresión no se ve una mayor importancia es todo lo contrario dado que depende de este valor es el nombre con el cual se van a buscar los archivos de configuración, con base al mismo servicio de disponibilidad se ejemplifica el impacto de este cambio.

En la clase se genera la siguiente variable:

|  |
| --- |
| public static final String MINEDUC\_\_LICENCIA\_ENSENANZA\_MEDIA = "MINEDUC\_DispoLicenciaEnsenanzaMedia"; |

Tabla 8 – Classe ProveedoresServicios

Por tanto los nombres de los archivos de configuración deben ser:

|  |
| --- |
| MINEDUC\_DispoLicenciaEnsenanzaMedia\_input.xml  MINEDUC\_DispoLicenciaEnsenanzaMedia\_output.xml |

Tabla 9 – Nombres de Archivos

Una vez que se realiza esta modificación se procede a modificar el proyecto “rest-web”. La razón de esta acción se origina por el hecho de crear o modificar una clase JAVA la cual se encargue de procesar la petición de una URI en particular.

Para poder ejemplificar el caso antes mencionado, se continúa con el caso de integrar el servicio “Disponibilidad Licencia Enseñanza Media” del proveedor Mineduc, asumiendo que esta es la primera generación de un servicio se crea la clase “cl.gob.minsegpres.pisee.rest.services.RestMineduc”, cabe señalar que todas las clases que se encarguen de procesar las peticiones a los servicios deben estar en el package “cl.gob.minsegpres.pisee.rest.services” y el nombre de la clase debe comenzar con la palabra “Rest” seguido del nombre del organismo proveedor. De esta forma la claseRestMineduc queda de la siguiente forma:

|  |
| --- |
| package cl.gob.minsegpres.pisee.rest.services;  import java.util.Date;  import javax.ws.rs.GET;  import javax.ws.rs.Path;  import javax.ws.rs.Produces;  import javax.ws.rs.QueryParam;  import javax.ws.rs.core.MediaType;  import org.apache.commons.lang.StringUtils;  import org.apache.commons.logging.Log;  import org.apache.commons.logging.LogFactory;  import org.springframework.context.annotation.Scope;  import org.springframework.stereotype.Component;  import cl.gob.minsegpres.pisee.rest.business.ProveedoresServicios;  import cl.gob.minsegpres.pisee.rest.business.RestBusiness;  import cl.gob.minsegpres.pisee.rest.entities.InputParameter;  import cl.gob.minsegpres.pisee.rest.entities.response.PiseeRespuesta;  import cl.gob.minsegpres.pisee.rest.util.AppConstants;  import cl.gob.minsegpres.pisee.rest.util.JSONUtil;  import cl.gob.minsegpres.pisee.rest.util.ParametersName;  @Path("/mineduc")  @Component  @Scope("singleton")  public class RestMineduc {    private final static Log LOGGER = LogFactory.getLog(RestMineduc.class);  private final static String \_METHOD\_GETLICENCIA = "getLicencia";  @GET  @Path("licenciaMedia")  @Produces(MediaType.APPLICATION\_JSON + ";charset=utf-8")  public String getLicencia(@QueryParam("rut") String rut,  @QueryParam("dv") String dv,  @QueryParam("pisee\_token") String piseeToken) {  InputParameter inputParameter;  JSONUtil jsonUtil;  PiseeRespuesta respuesta;  RestBusiness restService;  inputParameter = new InputParameter();  jsonUtil = new JSONUtil();  restService = new RestBusiness();  inputParameter.addBodyParameter(“rut”, rut);  inputParameter.addBodyParameter(“dv” dv);  if (isValidParameters(\_METHOD\_GETLICENCIA, inputParameter)){  respuesta = restService.callService(ProveedoresServicios.MINEDUC\_\_LICENCIA\_ENSENANZA\_MEDIA, inputParameter);  }else{  respuesta = createValidParameterError(\_METHOD\_GETLICENCIA, inputParameter);  }  return jsonUtil.toJSON(respuesta);  }    private boolean isValidParameters(String serviceName, InputParameter inputParameter){  if (\_METHOD\_GETLICENCIA.equals(serviceName)){  if (!StringUtils.isEmpty((String)inputParameter.getBodyParameter(ParametersName.RUT))  && !StringUtils.isEmpty((String)inputParameter.getBodyParameter(ParametersName.DV))  && !StringUtils.isEmpty((String)inputParameter.getBodyParameter(ParametersName.PISEE\_TOKEN))){  return true;  }  }  return false;  }    private PiseeRespuesta createValidParameterError(String serviceName, InputParameter inputParameter){  PiseeRespuesta respuesta = new PiseeRespuesta();  StringBuffer mensaje = new StringBuffer();  if (\_METHOD\_GETLICENCIA.equals(serviceName)){  if (StringUtils.isEmpty((String)inputParameter.getBodyParameter(ParametersName.RUT))){  mensaje.append(ParametersName.RUT);  mensaje.append(AppConstants.SPACE);  }  if (StringUtils.isEmpty((String)inputParameter.getBodyParameter(ParametersName.DV))){  mensaje.append(ParametersName.DV);  mensaje.append(AppConstants.SPACE);  }  if (StringUtils.isEmpty((String)inputParameter.getBodyParameter(ParametersName.PISEE\_TOKEN))){  mensaje.append(ParametersName.PISEE\_TOKEN);  mensaje.append(AppConstants.SPACE);  }  respuesta.getEncabezado().setEmisorSobre(AppConstants.\_EMISOR\_PISEE);  respuesta.getEncabezado().setEstadoSobre(AppConstants.\_CODE\_NOK\_PARAMETER);  respuesta.getEncabezado().setGlosaSobre(AppConstants.\_MSG\_INVALID\_NULL\_PARAMETER + mensaje.toString());  }  return respuesta;  }    } |

Tabla 10 – Clase JAVA

El detalle de las características es el siguiente:

Color Rojo: Corresponde a datos de clase tanto para su configuración via Spring como para el mapeo hacia que URI va a escuchar. De esta forma al mencionar la anotación “Path” con valor “/Mineduc” quiere decir que esta clase hará cargo de todas las URI que estén en el Path “https://www.pisee.cl/rest-web/mineduc” (en el caso de producción).

Color Verde: Corresponde a la información que entregará el método REST, cada uno de las anotaciones se explican a continuación:

* Get: Corresponde a la forma en la cual este servicio REST podrá ser consultado
* Path: Es el nombre del servicio expuesto vía REST, en ambiente productivo seria: https://www.pisee.cl/rest-web/mineduc/licencia.
* Produces: Es el tipo de respuesta el cual en este caso sería en formato JSON con encoding UTF-8.

Color Azul: Son los parámetros del servicio REST y al estar como QueryParam quedan de la misma forma que una URL conocida, de esta forma la URL final de este servicio seria: https://www.pisee.cl/rest-web/mineduc/licenciaMedia?rut=11111111&dv=1&pisee\_token=123ABC

Color Naranjo: Corresponde al nombre del servicio declarado en el punto anterior que hace mención a los cambios en el proyecto de “rest-business”.

### Modificar Archivo “applicationContext.xml”

Tal como se menciona en la documentación técnica, la solución utiliza, dentro de las tecnologías, Spring y Jersey para exponer los servicios REST. De esta forma el archivo “applicationContext.xml” toma vital importancia dado que es en este archivo donde se configuran las clases que se encargan de procesar las peticiones de cada uno de los servicios.

El contenido del archivo es solamente configuraciones en XML de cada una de las clases, en caso de que se agrega una nueva clase para exponer servicios de un proveedor es necesario activar esta clase desde este archivo, siguiendo con el mismo ejemplo la configuración del archivo seria la siguiente.

|  |
| --- |
| <bean  id="restMineduc"  class="cl.gob.minsegpres.pisee.rest.services.RestMineduc"  autowire="byName"  /> |

Tabla 11 – Clase JAVA

Donde finalmente el archivo quedaría de la siguiente forma:

|  |
| --- |
| <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:context="http://www.springframework.org/schema/context"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context.xsd" >  <!-- Configuracion base -->  <context:component-scan base-package="cl.gob.minsegpres.pisee.rest.services" />  <!-- Configuracion de los organismos proveedores -->  <bean id="restMineduc"  class="cl.gob.minsegpres.pisee.rest.services.RestMineduc"  autowire="byName" />    </beans> |

Tabla 12 – Clase JAVA

### Empaquetar Aplicación

Las indicaciones para generar e instalar la aplicación son los mismos que los descritos para el proyecto de Backoffice con la salvedad de que ahora los proyectos a instalar son “rest-business” y “rest-web”. Por tanto una vez que se agregue un servicio se deben modificar los dos proyectos, subir la version de cada uno de estos. La configuración del proyecto cumple con el mismo procedimiento que el proyecto Backoffice, el detalle de este procedimiento se describe en el documento: [http://www.app.pisee.cl/svn/PISEE/backoffice/trunk/documentacion/tecnica/[backoffice] - documentación técnica v.1.5.docx](http://www.app.pisee.cl/svn/PISEE/backoffice/trunk/documentacion/tecnica/%5bbackoffice%5d%20-%20documentación%20técnica%20v.1.5.docx) , “Capitulo 10 – Indicaciones para Deploy”

Finalmente todos los archivos de configuración se deben copiar a la carpeta:

“%JBOSS\_SERVER%/ deploy/base/piseeConf/rest\_services”

Se debe entender “%JBOSS\_SERVER%” como la ruta donde se encuentra instalado y configurado la instancia JBoss.

# Pos Condiciones

Luego de integrar un servicio tipo SOAP en la plataforma y exponerlo vía REST es necesario comprobar el correcto funcionamiento, es por esto que se indican dos actividades con las cuales es posible validar dicha integración:

## Comprobar Servicio por URL

Existen diversas formas para comprobar que el servicio se encuentre correctamente expuesto vía REST en la Plataforma, una forma es hacer una prueba con la aplicación SOAP-UI, pero también se puede realizar dicha verificación de una forma más sencilla, la cual sería visitar la URL desde cualquier browser.

Para consumir el servicio desde un browser se debe tener conocimiento de: La URL formada por la clase Java la cual procesa las peticiones, el nombre de los parámetros del servicio y por ultimo contar con un Token valido para el consumo. De esta forma un ejemplo de URL para consumir el servicio de Licencia Enseñanza Media provisto Mineduc sería la siguiente:

|  |
| --- |
| http://www.pisee.cl/rest-web/mineduc/licenciaMedia?rut=11111111&dv=9&pisee\_token=123ABC |

Tabla 13 – URI de Servicio REST

De esta forma la URL se compone de:

* **1 - Servidor**: http://www.pisee.cl/
* **2 - Contexto Aplicación**: rest-web/
* **3 - Organismo Proveedor**: mineduc/
* **4 - Nombre de Servicio**: licenciaMedia/
* **5 - Parámetros**: rut, dv, pisee\_token

La construcción de la URL depende tanto del ambiente donde se encuentra deployado la aplicación como de la clase que procesa la petición del servicio. El ejemplo anterior corresponde a la aplicación deployado en producción por lo tanto el Servidor y Contexto de la Aplicación no cambian pero tanto el Proveedor como el nombre de los servicio cambia según el WS SOAP expuesto vía REST.

Para conocer los valores del Organismo Proveedor, Nombre de Servicio y Parámetros es necesario comprobar los valores en la clase JAVA, en este caso la clase que se utiliza se encuentra en “cl.gob.minsegpres.pisee.rest.services. RestMineduc” luego se debe revisar la anotación “Path” de la clase y de los métodos, en el primer caso corresponde al nombre del Organismo expuesto en la URL y en el caso del nombre del servicio es la anotación que se encuentra sobre el método, para una mejor visualización se muestra el código:

|  |
| --- |
| @**Path**("/mineduc") -------> **3 - Organismo Proveedor**  public class RestMineduc |

Tabla 14 – Path de la Clase JAVA

|  |
| --- |
| @**Path**("licenciaMedia") -------> 4 **– Nombre de Servicio**  public String getLicencia(@QueryParam("rut") String rut,  @QueryParam("dv") String dv,  @QueryParam("pisee\_token") String piseeToken)  -------> 5 **– Parámetros** |

Tabla 15 – Path del método JAVA

Una vez conocido la composición de la URL y los valores de los parámetros se procede a consumir el servicio, la respuesta viene en formato JSON, un ejemplo de respuesta del mismo servicio anteriormente descrito es el siguiente:

|  |
| --- |
| {  "encabezado": {  "idSobre": "666666444420140303000000100",  "emisorSobre": "MINEDUC",  "estadoSobre": "00",  "glosaSobre": "TRANSACCION EXITOSA",  "nombreProveedor": "MINEDUC",  "nombreServicio": "DISPONIBILIDAD LICENCIA ENSENANZA MEDIA",  "nombreConsumidor": "SOLICITUD DE INFORMACION LICENCIA MEDIA",  "fechaHora": "2014-03-03T15:12:14.328Z",  "fechaHoraReq": "2014-03-03T15:12:14.328Z"  },  "metadata":[  {  "TieneLicencia": "true"  }  ]  } |

Tabla 16 – Ejemplo respuesta servicio

## Revisar Trazabilidad en Backoffice

La segunda actividad para comprobar la correcta integración de un servicio de la Plataforma vía REST es comprobar que ésta se registre de forma correcta en la base de datos, para esto es necesario ingresar a la aplicación Backoffice e ir a la opción Reportes, específicamente la opción “Transacciones por Organismo” una vez realizo una búsqueda ya sea de tipo Online o Diaria se desplegara los resultados en una tabla en la cual al pie se totalizaran los valores donde es posible distinguir el registro de las transacciones. Para una mejor visualización de la opción se presenta una captura de pantalla de la aplicación

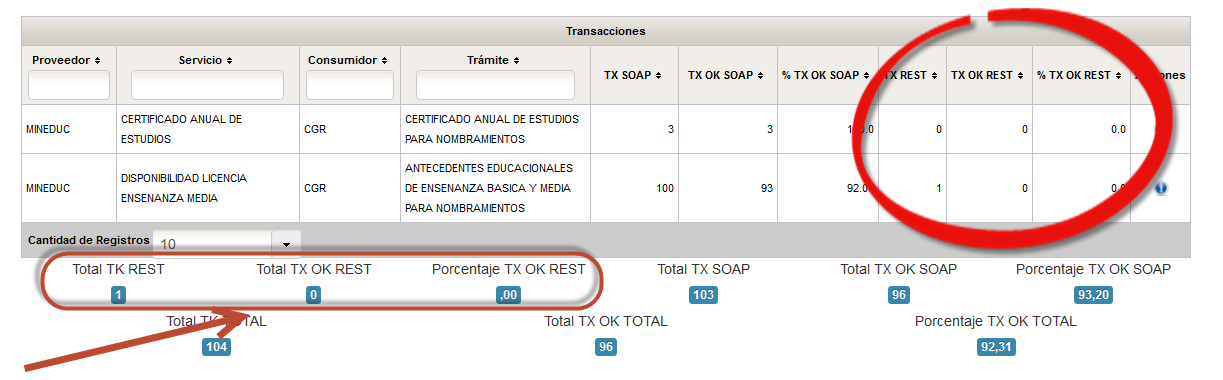


Figura 3 – Transacciones por Organismo