

Intermediador - Módulo de demostración

Revisión:	003
Fecha:	17-11-2015
Período de retención:	Permanente durante su período de vigencia + 3 años después de su anulación

INDICE

Introducción....3
Pasos durante la ejecución....5
Flujo de negocio....9
Certificados....11
Fichero de propiedades....11
Configurar SP....11
Clases de SP....12
Guía de integración de SP....13

Introducción

El sistema de intermediación es un sistema que permite identificar ciudadanos para terceros a través de diferentes sistemas, entre los que destacan los servicios de identificación europeos, el sistema 24H de la Agencia Tributaria y el sistema de Clave Permanente de la Seguridad Social.

Qué es el Intermediador

Es un Portal Web que recibe redirecciones desde aplicaciones externas y permite que los usuarios redireccionados se identifiquen utilizando un certificado español, una credencial europea o una contraseña de la Agencia Tributaria o de la Seguridad Social. Tras una identificación exitosa, el sistema responde adjuntando un ticket SAML.

Qué es un ticket SAML

Un ticket SAML (Security Assertion Markup Language) emplea un estándar XML abierto para el intercambio de datos de autenticación, auspiciado por OASIS. En los mensajes que se intercambian entre diferentes sistemas, se incluyen datos que se aseguran mediante el uso de una firma electrónica.

Para qué sirve

Esta forma de identificar ciudadanos telemáticamente puede ser explotado por diferentes sistemas que requieran identificar al usuario, por ejemplo, en webs de la administración, universidades, redes sociales, etc., permitiendo que el mecanismo de identificación este centralizado, evitando que la presentación de credenciales personales se haga a través de intermediarios, y descargando a los portales web de este trabajo sin detrimento de la seguridad.

Cómo funciona

Un proveedor de servicios (SP) quiere identificar a un ciudadano. Firma un ticket SAML y envía al ciudadano al portal intermediador. Este portal recibe el ticket y lo valida. Si todo está bien (es auténtico, no está modificado, y confía en la procedencia de la firma) procede a mostrar al usuario varias opciones. Tras seleccionar una de ellas, se hace efectiva la identificación solicitando al ciudadano algún tipo de credencial (algo que sólo él tiene, o algo que sabe) y se responde al SP con otra firma electrónica, donde se indican los datos del ciudadano que se han recuperado.

El portal recibe de nuevo al ciudadano, que ahora tiene en su poder un ticket firmado por el intermediador donde aparecen sus datos. El portal procede a validar dicho ticket y si confía en el certificado del intermediador, confía por extensión en los datos de la respuesta SAML y finalmente, confía en la veracidad de los datos que el ciudadano presenta, identificándole y siguiendo su flujo normal de negocio.

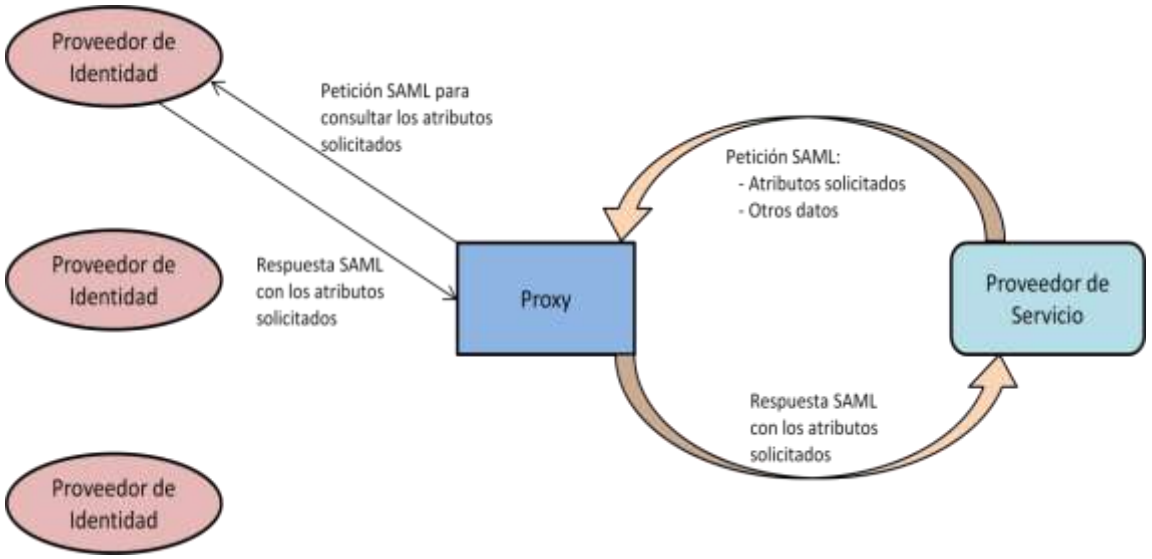
Paquete de integración

Para demostrar el funcionamiento de este módulo Intermediador se ha creado un módulo de pruebas que simula ser un SP (Service Provider ó Proveedor de Servicios). Este módulo permite lanzar peticiones al sistema aplicando unos cambios mínimos en su configuración, y se publica junto a su código fuente con el ánimo de que sirva de referencia para que cualquiera pueda establecerse como un proveedor de servicios. El módulo SP implementa la función requerida para cualquier Proveedor de Servicios:

- Crear una petición SAML.
- Redirigir al ciudadano.
- Esperar una respuesta.
- Validar la respuesta y extraer su contenido.

El contenido de la petición son una serie de atributos que siguen un esquema predefinido, así como otra metainformación (nivel de QAA o seguridad, lista de Proveedores de Identidad excluidos, etc.).

El intermediador, que actúa como un servidor proxy, se encarga de recibir esa petición, leerla y enviar al usuario hacia los Proveedores de Identidad correspondientes, con el fin último de rellenar la lista de atributos y crear una repuesta SAML en la que se almacenan los valores obtenidos por el Proveedor de Identidad, que es enviada de vuelta al Proveedor de Servicio que la solicitó.



Pasos durante la ejecución

Al acceder a SP desde el navegador se le muestra al usuario una ventana con dos botones: “Lanzar petición” y “Petición a medida”, así como un menú para seleccionar el destino de la petición (Imagen 1).

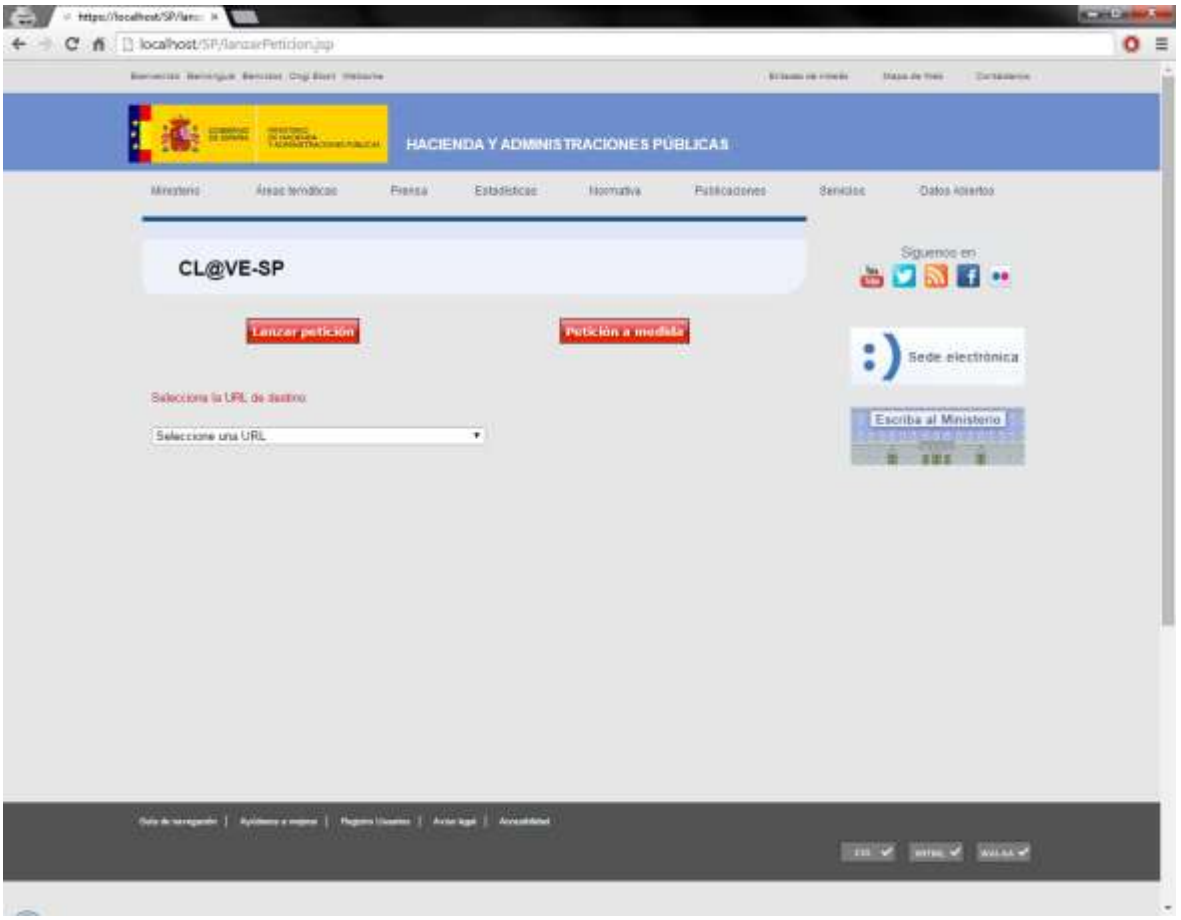


Imagen 1

En caso de que el usuario seleccione “Lanzar petición”, el programa hará automáticamente todo lo necesario para enviar una petición SAML por defecto al Intermediador seleccionado sin que el usuario tenga que intervenir en absoluto. Esta petición solicitará los atributos “eIdentifier”, “givenName” y “surname” (DNI, Nombre y Apellidos) y aceptará una respuesta SAML de cualquier IdP.

El otro botón le da al usuario la capacidad de personalizar la petición antes de enviarla para que sean solicitados únicamente los atributos elegidos por el usuario (sin embargo sólo se podrán hacer cambios sobre los tres atributos anteriormente mencionados, y los atributos “inheritedFamilyName”, “isdnie”, “registerType” y “ocspResponse”, ya que el resto de atributos están deshabilitados). Asimismo, en esta página también se pueden seleccionar los IdPs que se

excluyen para la autenticación del usuario (también existe la posibilidad de forzar la utilización de un IdP en concreto), si se aceptan certificados de persona jurídica, el nivel de QAA requerido, y se da la posibilidad de forzar la autenticación del usuario, ignorando una posible sesión previa que tenga el usuario en la pasarela (Imagen 2).

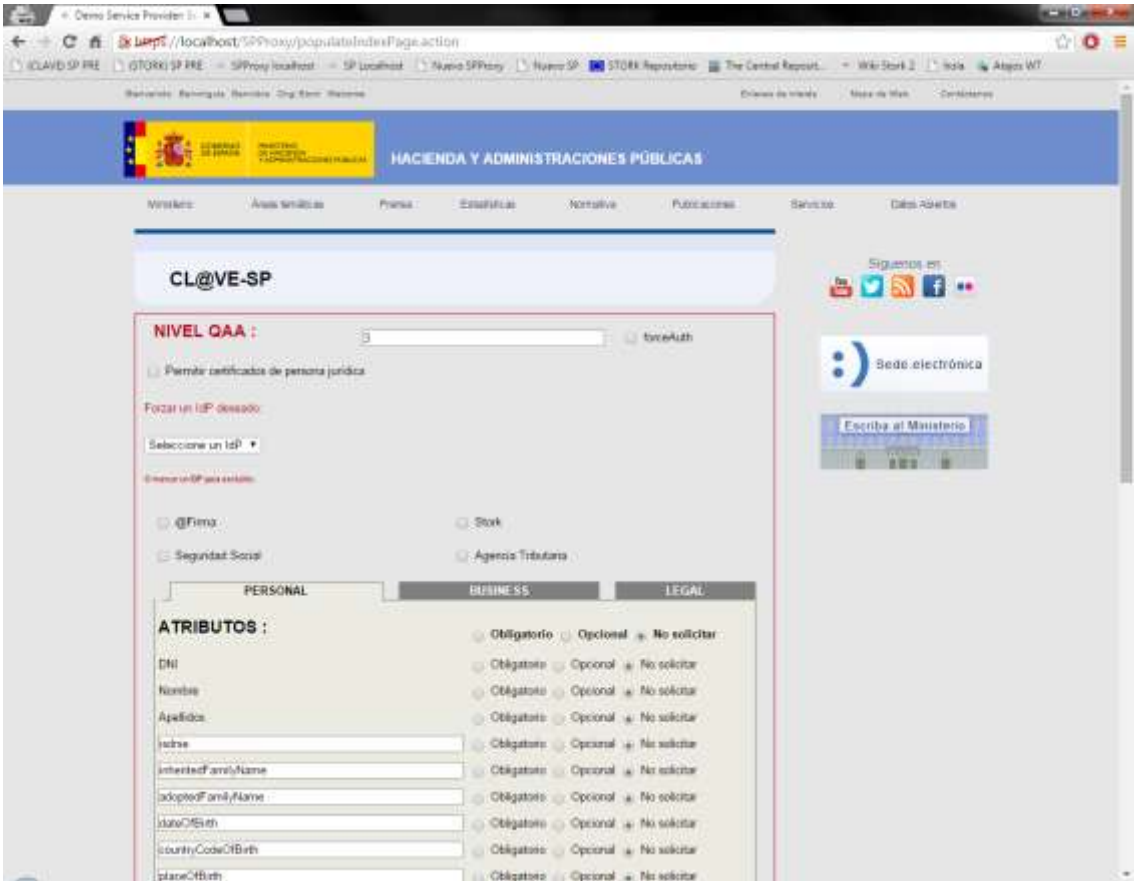


Imagen 2

Una vez que el usuario selecciona los atributos y la configuración deseada, habrá que presionar el botón “Submit”, que llevará al usuario a otra página (Imagen 3) en la que se le muestra la petición SAML que será enviada al Intermediador.

En esta página (Imagen 3) se puede ver el ticket SAML en formato Base64 y en claro, para que se pueda revisar la firma electrónica que se va a enviar cuando el ciudadano pulse sobre el botón *Submit*.

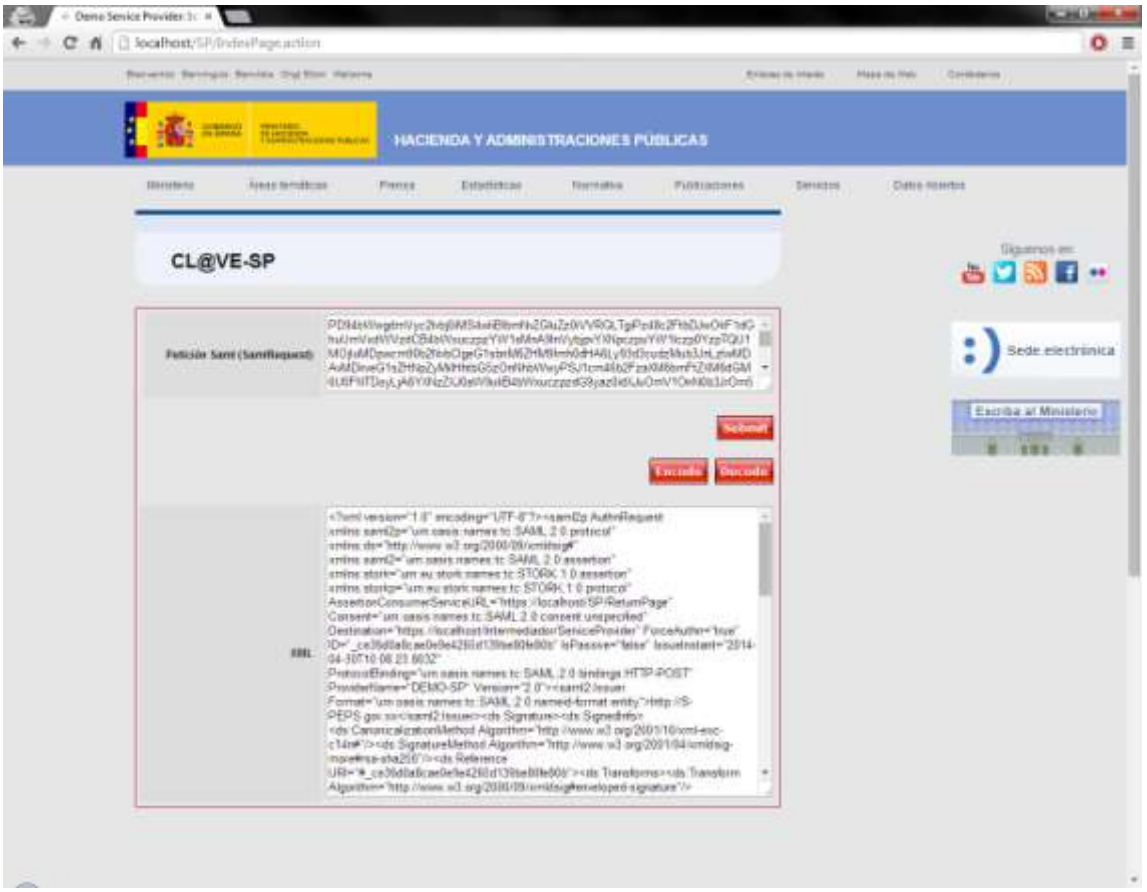


Imagen 3

Tras realizar el proceso de identificación, y tras recibir la respuesta, se muestra una página en la que aparecen los valores recabados por parte del IdP consultado por el Intermediador. En esta página también hay un botón de “Logout” que mandaría al Intermediador una petición de desconexión, que eliminaría la sesión del usuario tanto en el Intermediador como en los IdPs en los que se haya autenticado (Imagen 4).

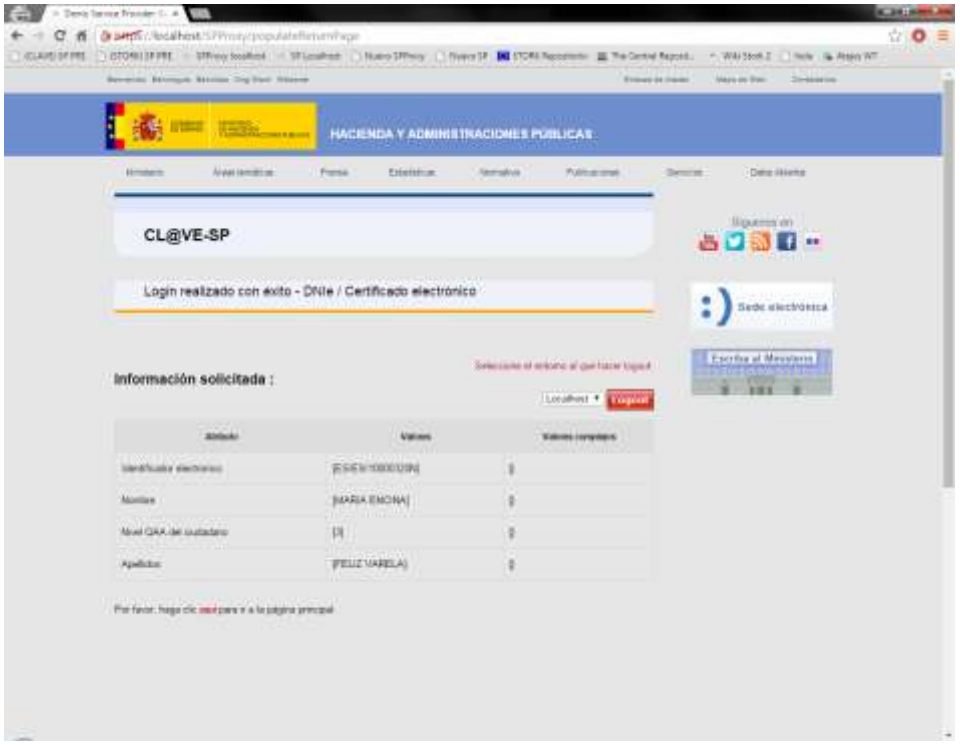


Imagen 4

En caso de que se pida como atributo el “inheritedFamilyName” (primer apellido), el SP al procesar la respuesta del Intermediador separará automáticamente el primer del segundo apellido y los presentará en dos atributos como “Primer apellido” y “Segundo apellido” (Imagen 5).

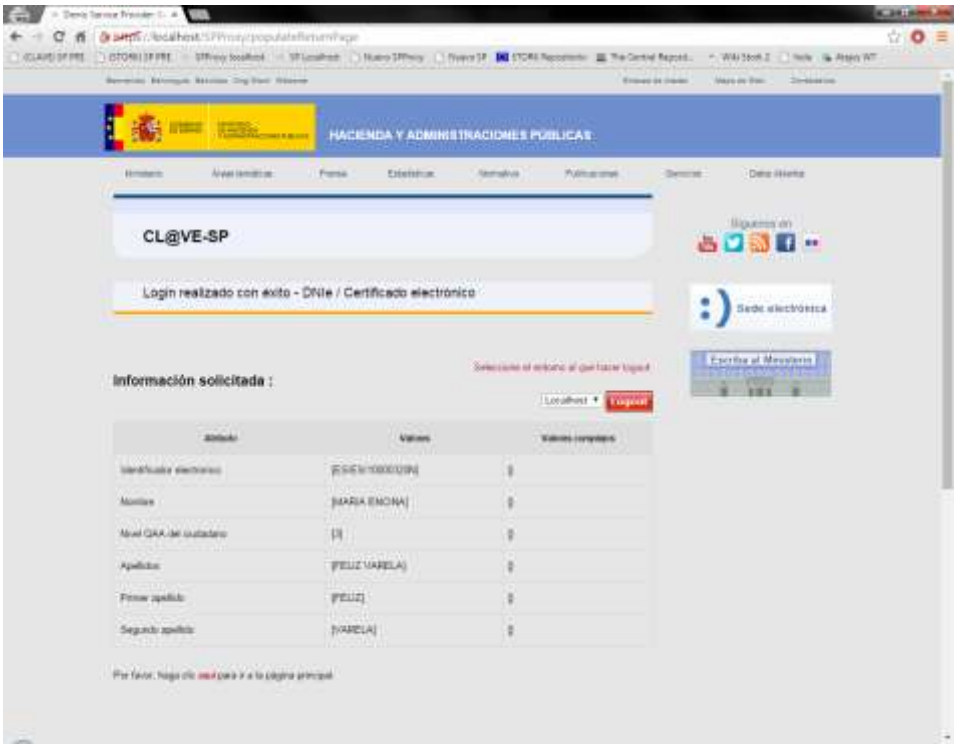


Imagen 5

Flujo de negocio

Dado que está implementado usando Apache Struts 2, SP sigue un patrón Modelo-Vista-Controlador (MVC). Para seguir el flujo es necesario consultar el fichero "struts.xml" de SP, ya que éste contiene la información sobre las conexiones entre el Modelo y la Vista, haciendo por lo tanto funciones de Controlador.

Cuando se accede a SP en el navegador, el usuario es redirigido automáticamente a "lanzarPeticion.jsp". En esta pantalla se muestran dos botones: "Lanzar petición" y "Petición a medida". El primero genera una petición SAML para el Intermediador con atributos e IdP's aceptados por defecto mientras que el segundo permite seleccionar los atributos a solicitar, así como los IdP's aceptados. El flujo también es diferente dependiendo de si se selecciona una opción o la otra.

Si se presiona "Lanzar petición" el usuario es dirigido a "defaultRequestIndexPage", que al consultar el fichero "SP/struts.xml" nos indica que se ejecutará el método "defaultRequest" de la clase "IndexAction.java". Dicho método genera una petición SAML que solicita los atributos "eIdentifier", "givenName" y "surname", y que acepta todos los IdP's como válidos. Al acabar la ejecución del método, éste retorna el valor "default", que según *struts.xml* hará que se cargue "defaultRedirect.jsp", que a su vez redirigirá la petición creada al Intermediador.

En caso de que se pulse "Petición a medida", el usuario es redirigido automáticamente a "populateIndexPage". Para saber dónde lleva esto, se puede consultar el fichero "SP/struts.xml", que nos indica que se ejecutará el método "populate" contenido en "IndexAction.java". Este método, después de cargar los atributos y realizar las acciones correspondientes, retorna el valor "populate", a lo cual según "struts.xml", le corresponde redirigir a "selectAttributes.jsp", que es la página que se le muestra al usuario.

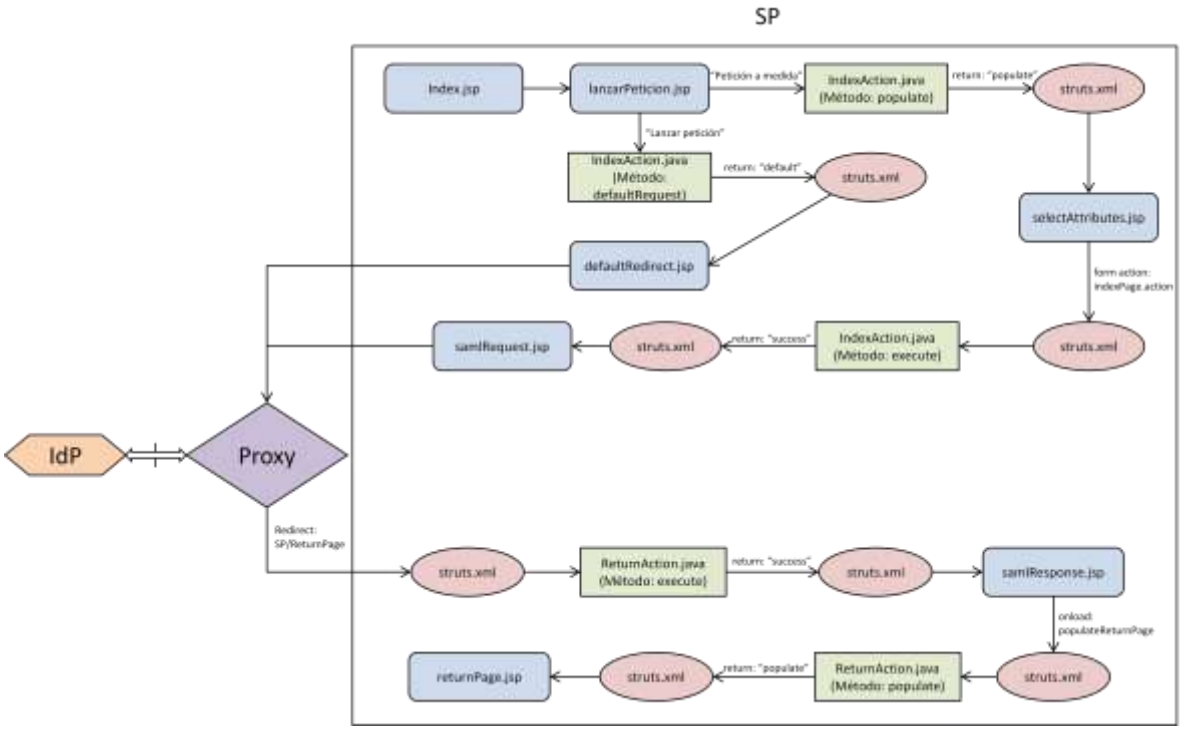
Una vez que el usuario selecciona los atributos que quiere solicitar, éste pulsa el botón "Submit". Según el código html de "selectAttributes.jsp", dicho botón está asociado a un formulario cuyo campo *action* vale "indexPage.action". Consultando el fichero "struts.xml" se aprecia que ese *action* corresponde con la clase "IndexAction.java", aunque en este caso no se especifica ningún método y por lo tanto se ejecuta el método por defecto, que es el método "execute". Dicho método, después de generar la petición SAML, retorna el valor "success", y según "struts.xml", lleva a "samlRequest.jsp". En este fichero "jsp" se muestra la petición SAML que se va a enviar al Intermediador. El envío se hará efectivo cuando el usuario pulse el botón
"Submit".

Después de que la petición sea procesada por el Intermediador y se redirija al usuario al proveedor de identidades correspondiente, el modulo de intermediación recibe los datos y envía a SP una respuesta SAML con los valores de los atributos solicitados. La dirección de retorno a SP está especificada en el fichero de propiedades “sp.properties” y va metida dentro de la petición SAML que se hace al modulo de intermediación, de forma que éste pueda mandar la respuesta SAML a la dirección requerida.

En el caso de este modulo de demostración, la dirección de retorno es “SP/ReturnPage”. En “struts.xml” esa dirección está asociada a la clase “ReturnAction.java”, y como no hay ningún método especificado se ejecuta el método por defecto (execute). El retorno de dicho método hace que la próxima página a mostrar sea “samlResponse.jsp”, que en su etiqueta “body” tiene un “onload” para redirigir automáticamente a “populateReturnPage”. Esto lleva una vez más a la clase “ReturnAction.java”, aunque esta vez al método “populate”, que después de retornar “populate” redirecciona a “returnPage.jsp”.

En esta última página se muestran los valores devueltos por el IdP correspondiente en respuesta a los atributos solicitados.

En caso de que haya algún error durante la ejecución, el retorno de los métodos sería “error”, lo que llevaría la página de error “errorPage.jsp”.



Certificados

El módulo SP funciona con un intercambio de firmas, y por lo tanto el servidor tiene que confiar en el certificado del SP y éste a su vez tiene que confiar en el certificado del servidor.

En el fichero “SignModule_SP.xml” se configura la ruta del almacén de certificados, el cual tiene que contener el certificado privado que usará SP para realizar firmas, así como los certificados públicos en los que se confía.

Fichero de propiedades

El fichero de propiedades “sp.properties” contiene todos los atributos que se le mostrarán al usuario para que elija los que quiere solicitar. Éstos están divididos en tres categorías: atributos personales, de negocio y legales. Además de estos atributos, el fichero de propiedades también contiene información de configuración como es el nombre del proveedor, la url del módulo, la urls del Intermediador al que se podrán enviar peticiones, la url a la que se retornará la respuesta del Intermediador (después de realizar la petición SAML), el nivel de QAA, las urls a las que dirigir las peticiones de logout y dónde recibir las confirmaciones de logout.

Configurar SP

Dentro del fichero .war proporcionado, en la ruta \WEB-INF\classes se encontrarán todos los ficheros de configuraciones que permitirán ajustar el SP al sistema en el que vaya a implantarse. Los cambios a realizar en los ficheros serán los siguientes:

log4j.xml y logback-test.xml:

En la etiqueta *<param name=”File”>* deberá modificarse el campo *value* e introducir la ruta en la que quieran guardarse los *logs*.

SignModule_SP.xml:

Deberá modificarse la etiqueta *<entry key=”keystorePath”>* y usar la ruta en la que se vaya a guardar el almacén de certificados (donde estará el certificado privado de SP para realizar firmas y también los certificados públicos en los que se confía).

El resto de etiquetas deberán modificarse acorde a las propiedades del almacén de certificados previamente indicado.

sp.properties:

Los campos *sp.url* y *sp.return* deberán modificarse acorde con la URL del sitio que va a alojar el SP (sin borrar más allá de "/SP…".
También habrá que modificar el campo *sp.logout.response.url* para indicar dónde tiene que enviar el Intermediador la respuesta confirmando la petición de logout.

spanishSp.properties:

Este fichero deberá modificarse si desea traducirse el SP.

Clases de SP

El módulo SP cuenta con las siguientes clases:

- *ApplicationSpecificServiceException.java*: Es una clase que extiende a RuntimeException. Es usada en otras clases como excepción que se lanza en caso de que haya un error específico de la aplicación en tiempo de ejecución.
- *Constants.java*: Clase que contiene constantes que serán usadas por otras clases
- *IndexAction.java*: Esta clase contiene varios atributos de configuración que se cargan desde el fichero apuntado por "SP_PROPERTIES" de la clase "Constants.java". IndexAction.java cuenta con dos métodos principales:
  - Populate: Rellena la información que se le muestra al usuario, que luego elegirá los atributos solicitados para generar una petición SAML.
  - Execute: Genera una petición SAML a partir de los datos rellenados por el usuario para mandársela al Intermediador, el cual a su vez se la enviará a un Proveedor de Identidad (IdP) para que rellene los datos solicitados.
  - defaultRequest: Genera una petición SAML por defecto que solicita los atributos "eIdentifier", "givenName" y "surname" (DNI, Nombre y Apellidos). Acepta cualquier una respuesta SAML de cualquier IdP.
- *LogoutAction.java:* Esta clase es la encargada de generar las peticiones de logout y procesar las respuestas a estas peticiones. Sus métodos principales son:
  - Execute: Genera la petición de logout.
  - ValidateResponse: Método encargado de validar la respuesta que llega desde el Intermediador, después de haberle solicitado el SP un logout.
- *ReturnAction.java*: Clase usada para procesar la respuesta recibida, una vez que los atributos solicitados por el usuario han sido rellenos por el IdP correspondiente. Consta de dos métodos principales:
  - Execute: Traduce la respuesta SAML recibida a formato XML para poder mostrarla en un JSP posteriormente.
  - Populate: Valida la respuesta SAML recibida y muestra el valor de los atributos solicitados.

- *SPUtil.java*: Clase que cuenta con tres métodos abstractos que aportan utilidad a otras clases:
    - LoadConfigs: Método usado para cargar un fichero de propiedades cuya ruta es pasada como parámetro y que devuelve un objeto de tipo “Properties”.
    - CreatePersonalAttributeList: Método que convierte una PersonalAttrubuteList a un String.
    - CreatePersonalAttributeList: Método que convierte un String a una PersonalAttrubuteList. El String tiene que estar en el mismo formato que el String devuelto por el otro método createPersonalAttributeList.

Guía de integración de SP

SP puede integrarse fácilmente con cualquier servidor, pero tomaremos como ejemplo un servidor Apache Tomcat v7.0.53.

Lo primero será configurar el puerto SSL. En nuestro caso, modificando el fichero *server.xml* que se encuentra en la carpeta *\conf*.

El puerto a configurar es el 443. Los cambios más importantes son:

- Habilitar tráfico SSL sin autenticación de cliente.

- Indicar los ficheros de certificados correspondientes para el handshake SSL (hay que indicar el certificado con el que se identifica el servidor).

- El protocolo que maneja el tráfico entrante es *HTTP/1.1* y el esquema es HTTPS.

Para un servidor Tomcat, la configuración de los puertos quedaría más o menos así:

<Connector SSLEnabled="true" clientAuth="false" connectionTimeout="20000" keystoreFile=[…] keystorePass=[…] keystoreType="JKS" port="443" protocol="HTTP/1.1" scheme="https" secure="true" />

Después es necesario configurar el paquete “SP.war” accediendo a su contenido como si fuera un fichero comprimido (es posible renombrar su extensión a .zip y abrirlo con un compresor que implemente este estándar) para editar el fichero:

- \WEB-INF\classes\sp.properties .- Para indicar el nombre del SP, e indicar la dirección del intermediador (country.url) y la dirección de retorno para este proveedor (sp.return).
- \WEB-INF\classes\SignModule_SP.xml .- Para indicar la ruta de acceso al fichero JKS que contiene los certificados necesarios para el intercambio SAML.
- \WEB-INF\classes\log4j.xml .- Para generar trazas que ilustren el flujo del programa, y se escriban los posibles errores en un fichero de log.

Por último, se ha desplegar el fichero .war en la carpeta "*deployment"s* correspondiente (en el caso de Tomcat, en *webapps*) y arrancar el servidor.

Se podrá comprobar el correcto funcionamiento del sistema accediendo desde un navegador a la dirección https://localhost/SP.