UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Universidad Internacional de La Rioja
Máster universitario en Seguridad Informática

Implementación de un sistema proveedor de identidad para ofrecer servicios de autenticación mediante Inicio de Sesión Único en una universidad de tamaño grande

Trabajo Fin de Máster

presentado por: Restrepo Guzmán, Santiago

Director/a: Botella, Juan Guillermo

Ciudad: Bogotá
Fecha: 24/09/2015

Resumen

En este trabajo de fin de master se plantea un sistema de autenticación centralizado que permita ofrecer servicios de Inicio de Sesión Único (SSO) en una organización con el objetivo de aportar a la seguridad de la misma en el sentido de que con estas tecnologías se delega la lógica de autenticación de todas las aplicaciones a un elemento central.

Se realiza un análisis del estado del arte para poder tomar decisiones que permitan escoger una herramienta que cumpla a cabalidad con los requerimientos de autenticación y centralización de las organizaciones, permitiendo llegar a una solución confiable de manera fácil, económica y segura.

Mediante la ejecución de un piloto experimental se evalúa una solución de alto grado de adopción en el mercado, así mismo otras herramientas que permiten ejecutar un entorno de pruebas muy cercano a la realidad de los entornos de producción de la actualidad.

De los resultados positivos que se obtienen en la validación de la herramienta se procede a la federación de identidad de Google Apps for Education para que delegue la autenticación de sus usuarios al Proveedor de Identidad (IdP). Así mismo se implementa un conector para que las aplicaciones de la organización puedan integrarse al nuevo servidor de autenticación centralizado y le deleguen a este último la lógica de acceso a los recursos.

Palabras Clave: Autenticación, SAML, SSO, Inicio de Sesión Único Proveedor de Identidad.

Abstract

In this master´s final project it´s proposed a centralized authentication system that allows to offer Single Sign On services within organization in order to contribute to the security project considering with this technologies all the application delegates to a broker.

An analysis of the State of the art is performed to take decisions that allow to choose one tool that fully meets the organization’s requirements, allowing a reliability solution for easy, economic and secure way.

Through the implementation of an experimental pilot evaluates a solution of high degree of adoption in the market, likewise other tools that allow you to run a test environment close to the reality of the present production environments.

With the positive results obtained in the validation of the tool, we proceed to the federation of identity for Google Apps for Education to delegate authentication of users to the identity provider (IdP). Also implements a connector so that the Organization's applications can be integrated to the new centralized authentication server and delegated the logic of access to resources to the Identity Provider.

Keywords: Authentication, SAML, SSO, Single Sign On, Identity Provider.

Índice

Índice de ilustraciones

Índice de tablas

CAPÍTULO 1. INTRODUCCIÓN

1.1. Antecedentes

El creciente mercado de aplicaciones web que ofrecen servicios, ya sea en la nube o dentro de una organización, conlleva que a diario los usuarios se deban de identificar para poder acceder a los recursos que les ofrecen en el trabajo, la escuela o en el ocio personal.

La gran mayoría de recursos que encontramos en línea requieren de una autenticación por parte del usuario, ya sea para brindar una experiencia personalizada por servicios agregados que se contratan con el proveedor del mismo o, en caso de ser un servicio que se ofrece de manera gratuita, simplemente requieren que nos identifiquemos para realizar una trazabilidad del comportamiento del usuario mientras consume el servicio ofrecido, con fines de mercadeo y publicidad. Es así que el usuario debe de identificarse en todo dispositivo y momento, desde que se levanta de la cama en la mañana y accede desde el móvil a la página de internet del periódico que le gusta, cuando llega a su trabajo y debe consultar el correo electrónico institucional para revisar los pendientes a tratar en el día, cuando en el entretiempo de la mañana accede a una red social para ojear lo que familiares y amigos comparten a través de ella, cuando al salir a tomar el almuerzo sube a su automóvil y usa una aplicación de ubicación para conseguir información sobre la ruta a casa, en fin, cada vez que interactúa con un dispositivo electrónico como móvil, tableta, computador personal, etc., y desea consumir un servicio se presenta alguna pantalla que requiere que se identifique ante un sistema para poder determinar si se ofrece el servicio o se rechaza la solicitud y no permite su uso.

El problema para el usuario aparece cuando los diferentes servicios a los cuales se conecta, manejan la seguridad en el acceso a sus sistemas de manera propia, independiente y en la mayoría de los casos diferente, es decir cada quien impone un sistema de identificación para sus usuarios, aunque la gran mayoría no se alejan del bien conocido método de las credenciales, es decir un usuario y una contraseña sobre la cual cada proveedor es libre de imponer las características deseadas de seguridad como lo son un número mínimo de caracteres, la combinación de letras mayúsculas minúsculas, entre otras, lo que hace que el usuario en teoría deba manejar un abanico de contraseñas diferentes para poderse identificar y acceder a sus servicios de forma segura.

Pero que un usuario maneje una contraseña segura e irrepetible para cada servicio es solo un ideal ya que lo más sencillo para él es manejar una sola contraseña “maestra”, que en muchas ocasiones escribe en un papel para recordarla fácilmente, y usa la misma contraseña

en todos los servicios que utiliza, enfatizando en la usabilidad pero dejando de lado la seguridad [1].

La proliferación de credenciales de usuario conlleva a una sobrecarga para la memoria del usuario quien tratara de crear contraseñas simples, dejando de lado las premisas de una contraseña segura, y cuando el sistema lo obligue a usar características de seguridad específicas sobre la contraseña, lo más seguro es que termine olvidándola y llamando al servicio de mesa de ayuda de la institución o del proveedor del servicio para que le reestablezca la contraseña con frecuencia causando congestión con los demás usuarios que tratan de solucionar problemas más relevantes en estos sitios [2].

Un estudio realizado en el año 2007 informó que el usuario promedio maneja alrededor de 25 cuentas en diferentes proveedores de servicios e introduce en promedio un total de 8 contraseñas al día [3], esto ha conllevado a la aparición de síndromes en los usuarios que solo tienen cabida en el mundo acelerado del crecimiento tecnológico y la interconexión global, y un ejemplo es que lo que hemos descrito en los párrafos anteriores ha sido apodado el síndrome de fatiga de las contraseñas [4].

Pero en una organización el dolor de cabeza por las múltiples autenticaciones a través de diferentes sistemas no es solo para los usuarios sino también para los administradores de los mismos [5]. Ellos deben administrar un sinnúmero de bases de datos con información de usuarios para con base en lo que se conoce de un individuo afinar los permisos de acceso a los recursos o más difícil aún puede resultar la eliminación de dichos permisos cuando los usuarios dejan la organización, ya que no se tiene conocimiento del total de accesos y/o permisos que tenía en las múltiples plataformas de la organización.

Con el objetivo de encontrar una alternativa a los inconvenientes que presentan las autenticaciones mediante credenciales en cada servicio han aparecido soluciones tecnológicas que pretenden ayudar con los procesos de identificación de usuarios haciendo uso de dispositivos físicos que el usuario cargue consigo o mediante características biométricas [6], los cuales incrementan la seguridad en la autenticación y facilitan la usabilidad para el usuario porque no es algo que deba crear y recordar si no que son tecnologías que se basan en algo que el usuario posee o es, pero desafortunadamente su adopción es baja debido a los costos de implementación de los elementos hardware involucrados en el proceso, pero también han aparecido otras tecnologías entre las que se encuentra el Inicio de Sesión Único (SSO, *Single Sign On*) que le ofrece al usuario la posibilidad de que con unas credenciales únicas pueda acceder a múltiples servicios sin la necesidad de tener que autenticarse de nuevo cuando cambie entre ellos. La ventaja del Inicio de Sesión Único (SSO) es que se puede combinar con las tecnologías mencionadas como dispositivos de

autenticación y biométricas para incrementar la seguridad de la autenticación cuando la información así lo requiera, es decir que un usuario puede acceder a los datos menos sensibles de una aplicación con credenciales de usuario y contraseña, pero si se necesita mayor seguridad para proteger la información crítica, puede hacer uso de certificados digitales, tokens[1] de seguridad, tarjetas inteligentes, biométricas o una combinación de estos [7].

1.2. Motivación

Como organización, la Universidad del Cauca no es ajena a los problemas descritos anteriormente, actualmente existen muchos sistemas con los que un usuario debe interactuar y cada vez que desea cambiar entre ellos se debe autenticar en cada aplicación web. Debido a esto los usuarios pretenden evitar el síndrome de la fatiga de contraseñas mediante el uso de contraseñas simples o la reutilización de una contraseña para varios servicios.

Un factor a tener en cuenta en cualquier organización es el componente humano de quien opera con sus sistemas, por ejemplo en la Universidad del Cauca se cuenta con alrededor de cuarenta mil (40.000) usuarios, de diferentes edades y funciones, es decir, hay estudiantes y egresados que en su gran mayoría son jóvenes que tienen un alto grado de adaptación de las TIC, pero también hay otro grupo demográfico compuesto por docentes y administrativos dentro del cual la mayoría son adultos y un alto porcentaje de ellos no cuentan con una alta adopción a las TIC y son precisamente ellos los que deben manejar la información de mayor sensibilidad para los procesos académicos y administrativos, pero desafortunadamente son quienes menos aplican las recomendaciones sobre la creación de contraseñas seguras y la privacidad de las mismas.

Adicional a esto, como sucede con muchas organizaciones hoy en día, se tienen convenios con servicios externos como Google Apps for Education[2], en donde se mantienen recursos adicionales en la nube, a los cuales se les debe garantizar el acceso a todos los usuarios, generando nuevos retos para los administradores de los sistemas como lo es que se mantengan múltiples almacenes de usuarios que requieren de sincronización diaria para que los cambios en los atributos de un usuario se vean reflejados en el servicio externo de la forma más rápida posible.

[1] Token: Es un dispositivo físico que acompañado de algo que el usuario conoce, como una contraseña, le permite acceder a un sistema informático o una red [37].
[2] Google Apps for Education: Es una suite gratuita de herramientas de productividad para la colaboración en el aula educativa [38].

La existencia de almacenes de datos independientes para las aplicaciones internas y externas presenta un inconveniente muy notorio y molesto para los usuarios y es que los cambios que realicen, como por ejemplo su contraseña, no se reflejen de inmediato en ambas plataformas por lo que se ven obligados a esperar tiempos muertos a que se realicen los procesos de sincronización, generalmente al medio día y en la noche, para poder ingresar a los recursos con los que cuentan en la nube causando tiempos muertos que son inaceptables para algunos usuarios.

La motivación del presente trabajo de fin de master radica en el interés de toda organización de poder mantener un control sobre la información personal de sus usuarios de forma centralizada y evitar que cada vez que se contrate un nuevo servicio externo se deban realizar réplicas de los almacenes de datos de usuarios para poder garantizar el acceso a los nuevos recursos, además de eliminar por completo la fatiga de las contraseñas en los usuarios, contribuyendo así a mejorar la cultura general respecto al tema de las contraseñas y que los usuarios comprendan que deben mantener unas credenciales seguras eliminando la apatía hacia este tema al saber que no deben de estarla ingresando cada vez que requieran cambiar de aplicación.

El centralizar la autenticación y evitar que cada aplicación realice una autenticación cuando la consulten permite mejorar la usabilidad de los sistemas por parte de los usuarios, a los administradores les permite mejorar el mantenimiento, la trazabilidad y la seguridad de los sistemas involucrados en los procesos de autenticación de los usuarios y a los desarrolladores les simplifica la lógica de autenticación ya que no la deben implementar en cada aplicación si no que esta tarea queda delegada al sistema de autenticación centralizado.

Es importante para las organizaciones de mediano y gran tamaño el contar con sistemas de autenticación centralizados que ofrezcan características de Inicio de Sesión Único para mejorar la administración de la infraestructura y la experiencia de los usuarios. De la identificación de estas necesidades de las organizaciones nace el objetivo de este trabajo de fin de master, que busca mediante la ejecución de un piloto experimental demostrar que se puede alcanzar el objetivo de un sistema de autenticación centralizado con Inicio de Sesión Único (SSO) haciendo uso de herramientas y estándares de libre distribución de una forma rápida, lo cual es de gran utilidad para todas las organizaciones de carácter público que no desean involucrarse en los grandes rubros que una solución de este tipo puede costar en el mercado.

Muchas organizaciones desisten de la idea de la implementación de un sistema centralizado de autenticación debido a que las soluciones que actualmente se ofrecen en la nube ofrecen servicios que se cobran de acuerdo al número de usuarios que se conecten y en ocasiones

para lograr el cubrimiento de todos los usuarios de una organización eleva el costo de la solución a presupuestos que son inalcanzables para organizaciones sobre todo del entorno educativo, cuyos rubros son limitados en áreas como las tecnologías de información.

Con el fin de encontrar una solución al problema planteado en los párrafos anteriores, se consolida una pregunta experimental: ¿Cómo implementar un sistema de autenticación centralizado para ofrecer servicios con experiencia de Inicio de Sesión Único (SSO) en la Universidad del Cauca, haciendo uso de herramientas de libre distribución que minimice al máximo los costos relacionados?

1.3. Planteamiento del trabajo

Para dar respuesta a la pregunta experimental planteada en el apartado anterior y abordar los problemas anteriormente mencionados de forma global, se propone la implementación de un sistema de autenticación centralizado al cual se conecten todas las aplicaciones que ofrecen servicios web a los usuarios de la Universidad del Cauca, desplegadas tanto al interior de la organización como las que se contratan con terceros en la nube o que se ejecuten en cualquier infraestructura ajena a la organización, además debe contar con Inicio de Sesión Único (SSO) para que mejore la experiencia del usuario al interactuar con el conjunto de servicios ofrecidos.

El sistema a implementar debe servir para que a largo plazo pueda convertirse en el proveedor de identidad para todas las aplicaciones web que ofrecen servicios a los usuarios de la Universidad del Cauca.

Debido a las características propias de las organizaciones públicas referentes a rubros asignados para las inversiones en TI se deben usar herramientas de libre distribución amparadas por licencias que permitan su modificación dentro de los parámetros que el proveedor de la misma estipule en la respectiva licencia, pero que básicamente permita adaptarla al esquema particular de cada organización sin tener que por ello pagar por dicha adaptación.

Por ser el primer paso hacia la evolución a un sistema centralizado completo que además de autenticación administre la autorización dentro de los aplicativos web se requiere que sea lo menos invasivo posible con la estructura actual, particularmente con el directorio activo de la organización que se encuentra desplegado en OpenLDAP[3] por lo que se requiere que la solución tenga compatibilidad con este directorio de libre distribución y no lo modifique.

[3] OpenLDAP: Es un esfuerzo colaborativo para desarrollar una suite LDAP robusta, de nivel comercial, completa y de libre distribución [39].

Teniendo en cuenta que en la organización en donde se va a desarrollar el piloto experimental la mayoría de servidores usan sistema operativo Debian[4], es necesario que la herramienta escogida tenga compatibilidad para ser desplegada en este sistema operativo, esto con el fin de que el nuevo servidor de identidad no se mantenga como un elemento independiente de la infraestructura actual si no que se integre con las protocolos de mantenimiento y actualización actuales.

Además la herramienta seleccionada debe permitir manejar la salida de los usuarios de los recursos, es decir, debe implementar mecanismos que faciliten la administración del cierre de sesión de los usuarios una vez que deseen abandonar la sesión en cualquiera de los sistemas a los cuales el Inicio de Sesión Único (SSO) les concedió el acceso.

Uno de los principales proveedores de servicios en la nube para la Universidad del Cauca es Google Apps for Education por lo que la solución debe ser compatible con el estándar SAML[5] en su versión 2.0 que es el que el proveedor implementa para la autenticación en entornos de Inicio de Sesión Único (SSO).

Finalmente en este trabajo de fin de master se busca implementar un sistema proveedor de identidad para ofrecer servicios de autenticación mediante Inicio de Sesión Único (SSO) en la Universidad del Cauca, este sistema debe ser fácil de implementar, escalable, compatible y seguro.

1.4. Estructura del trabajo

Este trabajo de fin de master está conformado por cinco capítulos que se describen brevemente a continuación:

Capítulo 1. Introducción: describe de una manera general al lector los antecedentes, motivación, planteamiento del problema y estructura del presente trabajo.

Capítulo 2. Estado del arte y marco teórico: presenta los trabajos relacionados con el área a tratar, las brechas existentes en los mismos y los aportes del presente trabajo de fin master, además aquí se desarrolló el marco teórico relevante al tema tratado.

Capítulo 3. Objetivos y metodología: en este capítulo se plantean tanto el objetivo general como lo objetivos específicos, también se presenta la metodología usada para la ejecución del piloto experimental.

[4] Debian: Sistema operativo de libre distribución basado en Linux, la versión más reciente hasta el momento es la 8.1 [40].
[5] SAML: Security Assertion Markup Language es un estándar XML que permite el intercambio de datos de autenticación y autorización entre dominios web seguros [36].

Capítulo 4. Implementación y validación de la solución: este capítulo se divide en cuatro partes donde la primera corresponde a la implementación de la solución del proveedor de identidad, la segunda parte corresponde al plan de pruebas para la validación de la solución donde se realizan configuraciones de proveedores de servicios de pruebas, la tercera parte es la implementación en producción para un proveedor de servicios interno y uno externo (Google Apps for Education) y la última parte presenta una discusión de los resultados.

Capítulo 5. Conclusiones, limitaciones y trabajos futuros: presenta las conclusiones generadas al finalizar el trabajo, además de las limitaciones que se presentaron durante el desarrollo de este trabajo de fin de master. También presenta los trabajos futuros que se derivan de las necesidades identificadas durante el desarrollo del presente trabajo de fin de master.

CAPÍTULO 2. ESTADO DEL ARTE Y MARCO TEÓRICO

En este capítulo se presentan los trabajos relacionados más relevantes, del análisis de los mismos se plasman las brechas existentes y los aportes de este trabajo de fin de master. La segunda parte del capítulo desarrolla el marco teórico de las tecnologías de Inicio de Sesión Único.

2.1. Trabajos relacionados

- **Implantación de un SSO (Single Sign On)** [8]**:** en este proyecto de fin de master el autor se enfoca en aplicaciones J2EE e implementa el servidor de identidad mediante la herramienta OpenAM[6] de ForgeRock. Se implementa en un entorno de alta disponibilidad mediante el uso de un balanceador. Los análisis se realizan sobre una aplicación de ejemplo proporcionada por OpenAM para realizar pruebas de autenticación y autorización.
- **Developing a Single Sign-On System - A Java-based authentication platform aimed at the web** [9]**:** En esta tesis de master el autor crea una plataforma de inicio de sesión único, denominada NaviBase, basada en el lenguaje JAVA. En la plataforma implementa el estándar SAML para proveer servicios de Inicio de Sesión Único (SSO) a aplicaciones y usuarios.
- **Strong Mobile Authentication in Single Sign-On Systems** [10]**:** En esta tesis de master el autor se enfoca en la creación de un protocolo seguro de autenticación para sistemas de Inicio de Sesión Único (SSO) usando el teléfono móvil como un token de seguridad. El autor implementa Shibboleth 2.2 en un servidor CentOS 5.4 para probar el prototipo que plantea.
- **Single Sign-on solution for MYSEA services** [11]**:** en esta tesis de master los autores generan un diseño de alto nivel y las especificaciones de un marco de Inicio de Sesión Único (SSO) para la Monterey Security Architecture (MYSEA). Aquí se hace un análisis del estado del arte de las actuales tecnologías que se usan comúnmente.
- **Web Single Sign-On System for WRL Company** [12]**:** en esta tesis de master el autor implementa un sistema de Inicio de Sesión Único (SSO) para las aplicaciones web de la compañía Wuhan Railway Logistics (WRL) basado en el protocolo KERBEROS[7].

[6] OpenAM: Es la evolución del proyecto OpenSSO que fue descontinuado por Sun, hoy Oracle. Ofrece una plataforma de servidor para administración de accesos y servicios de federación [41].
[7] KERBEROS es un protocolo de autenticación de red ofrecido bajo licencia de libre distribución mediante una implementación del Massachusetts Institute of Technology (MIT) [42].

- **Identity, Access Management and Single Sign-On Web-based Solutions** [13]**:** en este trabajo de master el autor realiza un estudio del estado del arte de tecnologías de autenticación fuertes y mediante un análisis comparativo escoge una arquitectura basada en la herramienta *RSA Access Manager* de la empresa EMC[2], para la cual diseña módulos para las aplicaciones web para que funcionen en un ambiente de Inicio de Sesión Único en Web (WSSO).
- **Single Sign-On SSO** [14]**:** Los autores de esta tesis de master realizan un análisis de soluciones de inicio de sesión único (SSO) para que la compañía AerotechTelub pueda tomar decisiones de implementación. Está enfocada a redes heterogéneas. Realizan una comparación de los protocolos KERBEROS y SESAME[8].
- **Implementation of Multi-tier Authentication Technique for Single-Sign On access of Cloud Services** [15]**:** en esta tesis de master el autor se enfoca en diseñar un modelo de autenticación segura con Inicio de Sesión Único (SSO) para servicios registrados en un ambiente de la nube. Hace uso de Google App Engine (GAE) para la simulación del diseño.
- **Analysis and Implementation of a SSO Solution for Several Web Portal** [16]**:** en este trabajo de fin de carrera el autor se enfoca en la implementación de un sistema de Inicio de Sesión Único (SSO) bajo el estándar de OpenID[9] para autenticación en portales web.

Actualmente en el mercado se encuentran múltiples herramientas que permiten la implementación de plataformas de autenticación, algunas cuentan con licencias de pago y otras son de libre distribución, entre ellas tenemos:

- **CAS - Central Authentication Service:** Es un protocolo de libre distribución desarrollado originalmente en la Universidad de Yale que se convirtió en un proyecto del *Java in Administration Special Interest Group* (JASIG) [17]. Una de las implementaciones de CAS más conocida es la de la empresa Apereo.
- **Gluu Server:** es una suite de administración de identidad y acceso que se compone de herramientas y estándares de libre distribución. Usa un servidor Shibboleth para integrar el estándar SAML y Asimba SAML Proxy para manejar las conexiones SAML entrantes [18].
- **JOSSO - Java Open Single Sign-On:** es una solución de libre distribución de Inicio de Sesión Único (SSO) basado en el estándar SAML listo para aplicaciones de internet

[8] Secure European System for Applications in a Multi-vendor Environmen (SESAME) es una tecnología que nace de un proyecto de la Comision Europea y ofrece Inicio de Sesion Unico (SSO) y control de acceso distribuido [43].
[9] OpenID es un estándar de autenticación basado en las especificaciones de OAuth 2.0, se basa en flujo de mensajes REST/JSON [44].

[19]. Adicionalmente implementa los estándares de WS-Trust, SPML, Java EE y OSGi. Se ofrece al público en dos ediciones: “Enterprise” y “Community”, la primera edición requiere de una suscripción anual paga y ofrece beneficios como soporte de la empresa Attricore de acuerdo a niveles de servicio contratados, la segunda edición no requiere de pagos por suscripción pero tiene limitaciones en los estándares que soporta ya que solo implementa SAML, solo se puede instalar en un (1) nodo y el soporte es comunitario, es decir, mediante foros [20].

- **Okta:** es un servicio en la nube que ofrece administración de identidad y movilidad para conectar personas desde cualquier dispositivo. Para la federación usa el estándar SAML y WS-Federation[10]. Se ofrece en cuatro ediciones, todas de pago, que son: *SSO, SSO Plus, Enterprise* y *Enterprise Plus* [21], donde la edición más económica de SSO cuesta dos dólares (USD 2) por usuario por mes.
- **OneLogin:** es un servicio web que ofrece administración de identidad e Inicio de Sesión Único (SSO) basado en el estándar SAML. Se ofrece en cuatro ediciones donde están *Starter, Enterprise* y *Unlimited* que requieren de un pago mensual por usuario conectado que varía entre los dos dólares (USD 2) y los ocho dólares (USD 8) por usuario al mes, además está la edición gratuita que se limita a cinco aplicaciones personales y tres aplicaciones ofrecidas por la compañía [22].
- **OpenAM:** es un producto de la empresa ForgeRock que ofrece una plataforma de administración de identidad que se puede adquirir de forma gratuita o por suscripción. Tiene soporte para múltiples estándares de autenticación y autorización entre los que están SAML, WS-Federation, OpenID y XACML. Su arquitectura se basa completamente en JAVA [23].
- **Shibboleth:** es un proyecto de Internet2 para ofrecer una solución de identidad federada para conectar usuarios dentro y entre organizaciones. Se basa fundamentalmente en el estándar SAML. Se desarrolla como software de libre distribución bajo licencia de software Apache [24].
- **SimpleSAMLphp:** es un proyecto liderado por UNINETT, implementa el estándar SAML íntegramente en PHP [25].
- **SSOCircle:** es un servicio de proveedor de identidad en la nube, implementa los estándares SAML y OpenID. Se ofrece en cinco ediciones, algunas por subscripciones mensuales y una sola edición gratuita con algunas limitaciones de capacidad y personalización [26].

[10] WS-Federation es un estándar que permite administración e intermediación de relaciones de confianza e intercambio de tokens de seguridad a través de servicios web [45].

- **WSO2 Identity Server:** es una plataforma de administración de identidad para aplicaciones web, ofrece seguridad sofisticada y reduce los tiempos de despliegue. Soporta SAML, OpenID y XACML. Es completamente de libre distribución y se distribuye bajo los términos de la licencia de Apache 2.0. Soporta varios almacenes de usuarios como LDAP externo, Microsoft Active Directory, Apache Cassandra o cualquier base JDBC [27]. Provee una interfaz de administración, interfaz para el usuario final para el manejo de perfiles, recuperación de cuentas y administración de aplicaciones autorizadas. Permite la personalización total de las páginas de consentimiento y las interfaces gráficas.

De lo anterior se puede observar que las aplicaciones que se ofrecen en la nube tienen costos muy cercanos y rondan los dos dólares (USD 2) por usuario al mes, en la Tabla 1 se presentan los costos en los que incurriría la organización al contratar un servicio de Inicio de Sesión Único (SSO) en la nube.

	Usuarios	Valor por mes (US)	Costo total por mes (US)	Costo por año (US)
Total de usuarios	43093	USD 2	USD 86.186	USD 1.034.232
Usuarios activos 2014 - 2015	24465	USD 2	USD 48.930	USD 587.160

Tabla 1. Costos de servicios de SSO en la nube

A pesar de que en la mayoría de soluciones en la nube ofrecen un descuento que supera el 50% del total de la inversión por ser un licenciamiento por volumen y por ser una institución del entorno educativo, los costos de implementación de este tipo de solución rondaría los quinientos mil dólares (USD 500.000) al año para poder ofrecer servicios a todos los usuarios registrados en la organización, no solo a los que han estado activos en el último periodo, lo cual es un rubro que supera el monto total del presupuesto otorgado por la administración de la Universidad de Cauca para el funcionamiento completo de la división de TI de la Universidad del Cauca para el año 2015, que es de alrededor de trescientos cincuenta mil dólares (USD 350.000), por lo que no se podría adquirir ninguna de estas soluciones si no que se debe optar por analizar el resto de opciones que son de libre distribución, se pueden instalar en la infraestructura propia de la organización y se pueden mantener con el personal de TI de la organización.

2.2. Brechas existentes

En [8] no logran llevar a cabo la federación de la autenticación para los servicios de Google Apps. Además la herramienta OpenAM exige modificaciones al esquema de OpenLDAP para poder integrarlo como almacén de usuarios.

En [9] realizan una implementación básica del estándar SAML para ofrecer servicios de Inicio de Sesión Único (SSO) dedicando todos los esfuerzos al desarrollo de la aplicación mientras que lo que se busca en este piloto experimental es la implementación de un sistema proveedor de identidad de forma rápida y segura haciendo uso de herramientas establecidas en el mercado.

En [10] el autor se enfoca en crear un segundo factor de autenticación y realiza una implementación básica de un proveedor de identidad solo para probar que el prototipo funciona.

En [11] los autores no realizan ningún tipo de implementación, el trabajo se limita a investigación del estado del arte y generación de un diseño para su posterior desarrollo y puesta en funcionamiento.

En [12] a pesar que el objetivo final del sistema de Inicio de Sesión Único (SSO) son las aplicaciones web de una organización, el autor usa un protocolo que no está optimizado para ofrecer Inicio de Sesión Único en Web (*Web Single Sign On*, WSSO) si no que hace uso del protocolo KERBEROS que generalmente se emplea para ofrecer este servicio en aplicaciones cliente/servidor.

En [13] el autor no usa una herramienta basada en certificados y desarrolla el mismo los conectores para cada aplicación que desea conectar, lo que lo hace poco práctico en entornos de alta escalabilidad.

En [14] los autores no realizan ninguna implementación, solo realizan análisis del estado del arte y se enfocan a los protocolos KERBEROS y SESAME que no están optimizados para ofrecer Inicio de Sesión Único (SSO).

En [15] el autor aplica el estudio a servicios ofrecidos por proveedores de servicio en la nube, adicional a esto se enfoca en la configuración de un segundo paso de autenticación posterior al de las credenciales de usuario.

En [16] el autor opta por usar el estándar OpenID para soportar el servicio de Inicio de Sesión Único (SSO).

La herramienta CAS de JASIG, que implementa el protocolo CAS, requiere de un elemento intermedio para poder interpretar mensajes de entidades SAML.

2.3. Aportes

El principal aporte de este trabajo de fin de master es la implementación de un sistema proveedor de identidad para ofrecer servicios de autenticación mediante Inicio de Sesión Único (SSO) para las aplicaciones web de una organización mediante el uso de herramientas de libre distribución que reduzcan al máximo el costo de la solución.

Esta implementación servirá para incrementar el estado del arte del área ya que se enfoca en plasmar una forma económica, fácil, eficiente, rápida y segura de poner en funcionamiento un servidor de identidad, que es lo que hoy en día requieren la mayoría de las organizaciones que no cuentan con esta tecnología y no cuentan con los recursos y el tiempo que demandaría el desarrollo de un sistema desde cero.

El sistema de Inicio de Sesión Único (SSO) implementado servirá para que las organizaciones manejen de manera centralizada a sus usuarios y puedan ofrecer autenticación a los servicios que se contraten con terceros en la nube.

El piloto experimental contribuye en el proceso básico completo que debe adoptar una organización que se involucre en un proyecto de implementación de Inicio de Sesión Único (SSO) pasando por todos los pasos necesarios para entrar en operación y esto incluye la elección de la herramienta que se ajuste a sus necesidades particulares, la implementación de dicha herramienta, las pruebas con proveedores de servicios ficticios y la conexión con proveedores de servicios reales locales y federados. El éxito de la solución se basa en el análisis de costos que se realice sobre la solución implementada que tiene que arrojar un presupuesto bajo, respecto a las soluciones en la nube que se analizaron en apartados anteriores, que se ajuste a la realidad de una organización de educación pública.

Los aportes del trabajo de fin de master frente a los trabajos relacionados que se presentan en al apartado 2.1 se consolidan en la Tabla 2.

Trabajo	Implementación	Herramienta de libre distribución	Estándar SAML	Federación aplicaciones	Desarrollo conector SAML
Implantación de un SSO (Single Sign On) [8]	Si	Si (solo actualizaciones principales)	Si	No	No

Single Sign-on solution for MYSEA services [11]	No	No	Si	No	No
Web Single Sign-On System for WRL Company [12]	Si	Si	No	No	No
Identity, Access Management and Single Sign-On Web-based Solutions [13]	No	Si	No	No	Si
Analysis and Implementation of a SSO Solution for Several Web Portal [16]	Si	Si	No	No	Si
Implementación de un sistema proveedor de identidad para ofrecer servicios de autenticación mediante SSO en la Universidad del Cauca	Si	Si	Si	Si	Si

Tabla 2. Resumen de aportes de los trabajos relacionados

2.4. Marco teórico

2.4.1. Estándares

2.4.1.1. Security Assertion Markup Language (SAML)

Es uno de los estándares más conocidos en la actualidad, desarrollado por el comité Security Services Technical Committee of the Organization for the Advancement of Structured Information Standards (OASIS) y especifica un esquema basado en XML para intercambiar información sobre autenticación y autorización de sujetos (Subjects) entre diferentes dominios

seguros, específicamente entre un proveedor de identidad quien es quien produce la información de autenticación y un proveedor de servicios quien es quien consume dicha información [9].

Su primera versión es SAML 1.0 que data del año 2001 y fue estandarizado por OASIS en noviembre de 2002. De allí sufrió modificaciones menores, sobre todo de reorganización, y se presentó un nuevo estándar en 2003 denominado SAML 1.1. Posteriormente se realizó una modificación considerable sobre el estándar y en el año 2005 se presenta la última versión del estándar conocida como SAML 2.0, que es la que se encuentra vigente en la actualidad. Esta última versión incluye modificaciones aportadas desde el marco de la Liberty Alliance Identity Federation (ID-FF) y debido a sus remarcadas mejoras no mantuvo la compatibilidad hacia atrás con las versiones previas del protocolo.

SAML especifica tres roles: Usuario (Principal), Proveedor de Identidad (IdP) y Proveedor de Servicios (SP) [28], en un escenario normal el usuario solicita un servicio del Proveedor de Servicios (SP), este a su vez solicita y obtiene un aserto de identidad del Proveedor de Identidad (IdP) y basado en este aserto el Proveedor de Servicios (SP) puede tomar una decisión de control de acceso, es decir, si decide permitir que el usuario consuma el servicio ofrecido o le deniega el mismo.

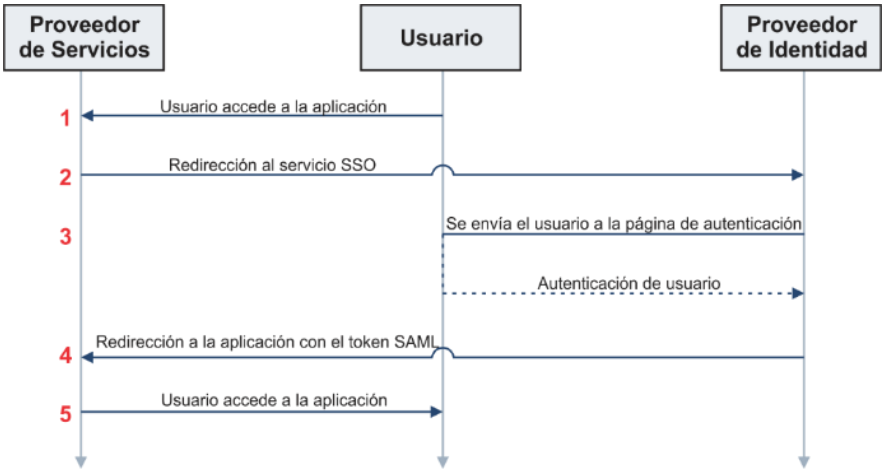

Ilustración 1. Flujo de autenticación en SAML

La seguridad en SAML se abarca en dos formas, la primera es la seguridad a nivel de transporte del protocolo para lo cual especifica el uso de protocolos de cifrado de información como lo son TLS 1.0 o SSL 3.0 y además se tiene en cuenta la seguridad a nivel del mensaje en si mediante la implementación de los estándares XML Signature y XML Encryption.

Los componentes principales del estándar son cuatro: núcleo (core), protocolos, perfiles y bindings. A continuación se presenta una breve descripción de cada componente.

Núcleo (Core): es el corazón del protocolo y es donde residen las definiciones de mensajes que pueden ser intercambiados por las entidades. Se compone de:

- **Asertos (Assertions):** Es la esencia de SAML, en ellos se describe el aserto de una entidad a otra respecto a la identidad de un sujeto. Transportan afirmaciones y declaraciones del sujeto obtenidas de una parte confiable.

  Están definidos tres tipos de asertos que una entidad SAML puede crear [29]:

  - **Autenticación:** Trata de acerca de la identidad de un sujeto.
  - **Atributos:** especifica información detallada acerca de un sujeto.
  - **Decisión de autorización:** especifica que elementos o recursos tiene permitidos un sujeto.

  Algunos de los elementos que componen un aserto SAML son:

  - **Version:** identifica la versión del estándar.
  - **ID:** es un identificador aleatorio y único para la solicitud.
  - **Issuer:** este elemento es obligatorio y específica a la entidad SAML que certifica la afirmación.
  - **Signature:** firma que asegura la integridad y autenticidad de las afirmaciones de seguridad
  - **Subject:** especifica el sujeto o usuario sobre el cual están hechas todas las afirmaciones del aserto.
  - **NameID:** este elemento es opcional, es una representación en cadena de texto del sujeto.
  - **SubjectConfirmation:** este elemento es opcional, provee los medios para confirmar al sujeto. Requiere de un atributo denominado “Method” que define el método usado para hacer la confirmación, existen tres métodos: “bearer”, “sender-vouches” y “holder-of-key”
  - **SubjectConfirmationData:** especifica información adicional para la confirmación del sujeto, como por ejemplo información de la llave usada en el método “holder-of-key”. Tiene cuatro atributos que son opcionales: “NotBefore”, “NotOnOrAfter”, “InResponseTo” y “Recipient”
  - **Conditions:** establece las condiciones bajo las cuales el aserto puede ser considerado valido.
  - **AudienceRestriction:** define si el aserto va dirigido a una determinada audiencia.
  - **Statements:** las declaraciones brindan información de un contexto particular, puede ser de tipo “AuthnStatement” que específica que la entidad del aserto

autentico de manera correcta al sujeto por determinados medios y tiempo específico, debe contener un elemento “AuthnContext” y un atributo “AuthnInstant”. La declaración también puede ser de tipo “AuthzDecisionStatement” que especifica algo a lo cual el sujeto está autorizado o de tipo “AttributeStatement” que especifica atributos del sujeto.

A continuación se presenta un ejemplo de aserto de autenticación:

```
<Assertion xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion" ID="dhndpho…"
IssueInstant="2015-08-31T01:47:41.503Z" Version="2.0">
 <Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">unicauca</Issuer>
 <Subject>
  <NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">sanrestrepo</NameID>
   <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
    <saml2:SubjectConfirmationData InResponseTo="_2CAAA…" NotOnOrAfter="2015-08-31T01:52:41.50
3Z" Recipient="https://unicauca.my.salesforce.com"/>
   </SubjectConfirmation>
 </Subject>
 <Conditions NotBefore="2015-08-31T01:47:41.503Z" NotOnOrAfter="2015-08-31T01:52:41.503Z">
  <AudienceRestriction>
   <Audience>https://saml.salesforce.com</Audience>
  </AudienceRestriction>
 </Conditions>
 <AuthnStatement AuthnInstant="2015-08-31T01:47:41.504Z" SessionIndex="aa1…">
  <AuthnContext>
<AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:Password</AuthnContextClassRef>
  </AuthnContext>
 </AuthnStatement>
    <AttributeStatement/>
</Assertion>
```

- **Protocolos:** Especifican la forma como se comunican las entidades, hay varios pero lo más usados son:
  - **Solicitudes (Request):** las más comunes son las “AuthnRequest” que son solicitudes mediante las cuales los Proveedores de Servicios (SP) le piden a los Proveedores de Identidad (IdP) que traten asertos de sobre un sujeto especifico.
  - **Respuestas (Responses):** las más comunes son las “Response” donde un Proveedor de Servicios (IdP) puede incluir el aserto solicitado o un mensaje de error.
- **Bindings:** especifican como se envían los asertos SAML mapeando los mensajes de solicitud y respuesta en mensajes estándar o protocolos de comunicaciones, como SOAP o HTML. Entre otras funciones, especifican la codificación y las respuestas ante fallos. En SAML 2.0 se definan seis tipos de bindings, los más comunes son HTTP-POST, HTTP-REDIRECT y SOAP [30].
  - **HTTP-REDIRECT:** especifica la forma para transportar un mensaje del protocolo SAML como un parámetro de una URL de un mensaje de redirección HTTP. En este método una entidad envía un mensaje a otra devolviéndole un

HTTP-REDIRECT al usuario, dirigiéndolo a una URL de la otra entidad la cual incluye como parámetros el mensaje de solicitud y el estado (Relay State), de la siguiente forma:

```
https://sp.unicauca.edu.co/Response?SAMLResponse=nVKp...g==
```

Para convertir el mensaje del protocolo SAML a un parámetro de una URL se usa el algoritmo de deflación (DEFLATE)[11] que primero comprime el mensaje, los datos comprimidos se codifican en Base64 y luego se codifican según el RFC 3986 para agregarse a la URL como una cadena de consulta.

La URL se envía como una respuesta HTTP que contiene un código de estado de redirección "HTTP 302" lo que hace que el usuario lleve los parámetros adjuntos en la URL a la entidad SAML receptora.

Este método se ve limitado por que los navegadores web establecen un máximo para la longitud de una URL.

- **HTTP-POST:** especifica la forma para transportar un mensaje del protocolo SAML como el contenido de un formulario HTTP. Funciona de igual forma que el HTTP-REDIRECT con la salvedad de que se usa un formulario mediante JavaScript en lugar del mensaje HTTP-REDIRECT, permitiéndole de esta manera que se transmitan mensajes de mayor tamaño. El mensaje de protocolo SAML viaja oculto en un campo del formulario codificado en Base64, el campo lleva el nombre de "SAMLRequest" o "SAMLResponse" según sea el caso. El formulario se transmite mediante el método HTTP POST.

- **Perfiles:** un perfil especifica cómo se usan los asertos y protocolos para alcanzar un fin determinado [31]. Existen un total de 13 perfiles definidos en el estándar SAML cada uno concebido para un caso de uso específico. Por ejemplo el perfil Inicio de Sesión Único en Web (WSSO) permite que un usuario solicite acceso a un servicio web protegido.

[11] DEFLATE es un método basado en el algoritmo LZ77 y codificación Huffman que permite compresión sin pérdida de información. Se define en el RFC 1951 [46].

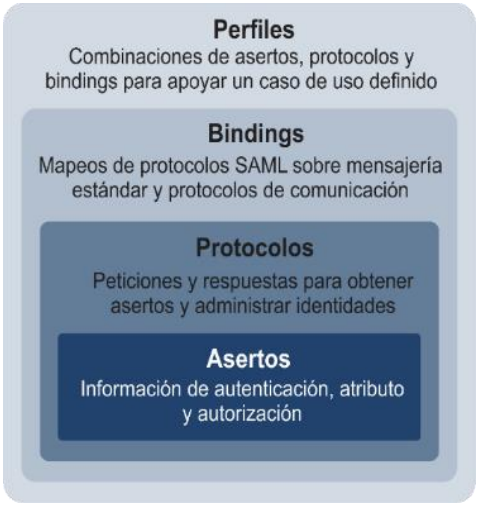

Ilustración 2. Componentes del estándar SAML

2.4.1.2. OpenID Connect

Es la evolución de OpenID 2.0 que es un proyecto de código libre para administración de autenticación y autorización. OpenID es descentralizado y por tanto cualquiera puede configurar su propio servidor OpenID, además no requiere de ninguna configuración previa o de confianza entre el Proveedor de Servicios (SP) y el Proveedor de Identidad (IdP), que en este estándar pasan a llamarse Parte Confiante “Relying Party” (RP) y el Proveedor OpenID (OP), respectivamente.

OpenID Connect asume gran parte de las tareas de su predecesor y se define como una capa simple de identidad sobre el protocolo OAuth 2.0 que permite a los clientes verificar la identidad de un usuario final basada en la autenticación realizado por un servidor de autorización [32].

Es fácil de integrar debido a que se basa en un flujo de mensajes REST/JSON y el diseño de API´s es amigable, mejorando los problemas de OpenID 2.0.

2.4.2. Modelos de implementación

En la actualidad existen diferentes modelos que se usan en las arquitecturas de Inicio de Sesión Único (SSO), la escogencia de cada uno para una implementación real depende de diferentes factores como el tamaño de la organización, los equipos físicos con los que cuenta la organización y la inversión presupuestada para la adopción del sistema dentro de la misma, a continuación se presenta un breve descripción de los modelos típicos que se pueden encontrar en la actualidad:

- **Centralizado (Brokered):** En este modelo existe un servidor central que se encarga de la autenticación, autorización y administración de usuarios. Se simplifica el uso por

parte de los usuarios debido a que se cuenta con una interfaz única de autenticación para todos los servicios.

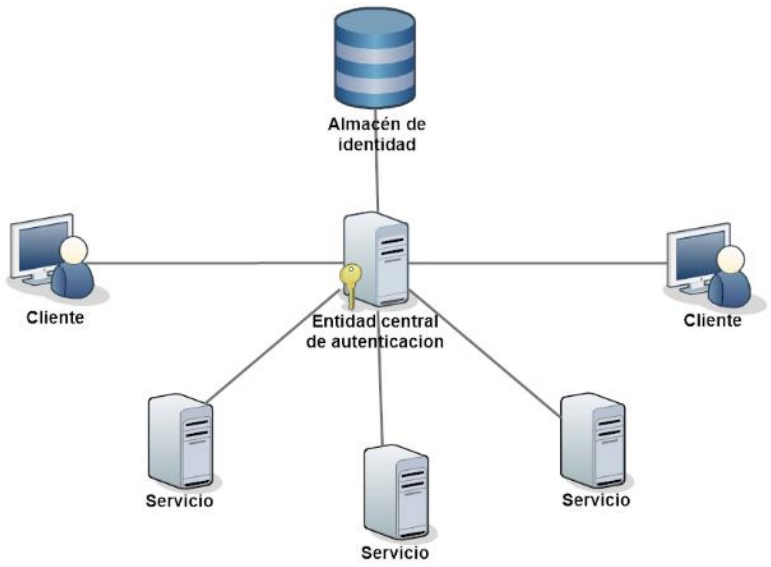

Ilustración 3. Modelo centralizado

- **Agentes (Agent):** En este modelo existe un aplicativo en el lado usuario que se usa para identificar todas las solicitudes de dicho equipo, esto se realiza sin la intervención del usuario ya que el agente en si puede contener listas de contraseñas o llaves de cifrado que el servidor conoce y basado en ellas realiza la autenticación.
- **Pasarela (Gateway):** En este modelo se hace uso de un elemento intermedio entre el usuario y los recursos protegidos, ubicando una pasarela de autenticación que puede ser un dispositivo cortafuegos que se encarga de autenticar a los usuarios y permitirles el acceso a todos los recursos que están detrás del mismo. En esta aproximación el control de los accesos de los usuarios autenticados se puede realizar mediante la implementación de reglas basadas en las direcciones de los equipos dentro de la red. Las concesiones de acceso se pueden modificar haciendo uso de estas reglas evitando la modificación de las aplicaciones.

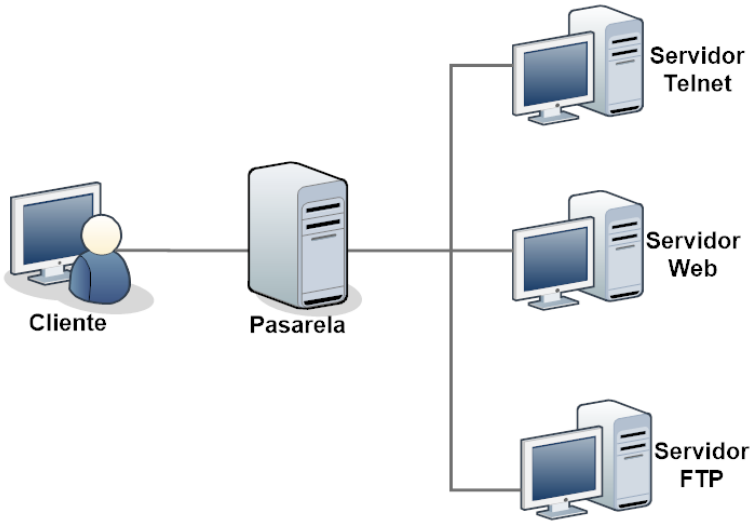

Ilustración 4. Modelo pasarela

Adicional a estos modelos básicos se encuentran modelos híbridos que no son más que modelos que combinan las características y ventajas de varios de estos.

- **Token:** Este modelo se puede englobar en los de tipo centralizado ya que se autentica con un servidor pero le agrega un elemento conocido como token que es generalmente un dispositivo físico que genera contraseñas de un solo uso variables con el tiempo permitiéndole al usuario contar un segundo factor de autenticación. Estos dispositivos deben ser sincronizados en su primer uso con el servidor de autenticación para su correcto funcionamiento. La contraseña única se solicita en el inicio de sesión de un recurso protegido y se almacena durante el tiempo determinado para dicha sesión habilitando al usuario para navegar a través de los recursos protegidos sin solicitarle de nuevo el ingreso de una nueva contraseña generada por el dispositivo.
- **Centralizado/Agente (Agent/Broker):** Es una combinación de dos de los modelos explicados previamente y se utiliza para usar las facilidades del modelo de agentes que evita la modificación de las aplicaciones con el modelo centralizado de permisos que le puede otorgar acceso al agente a ciertos recursos y este agente se los transmite al cliente de manera transparente.

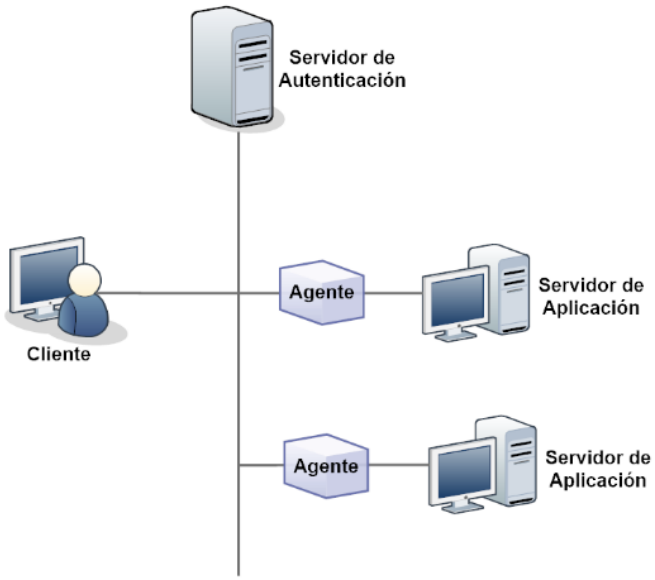

Ilustración 5. Modelo Centralizado/Agente

A continuación se presenta un resumen a modo de comparación entre los diferentes modelos enfocado a cuatro aspectos diferenciales entre ellos [5], el primero es la facilidad de implementación que se relaciona con la complejidad del sistema y la medida de que tan fácil es la integración del sistema actual con la nueva implementación de SSO, el segundo es la administración en cuanto a que tan fácil es manejar las funcionalidades, el tercero es relacionado con la seguridad y que tan vulnerable es a los ataques y por último la usabilidad desde el punto de vista del usuario.

	Facilidad de implementación	Administración	Seguridad	Usabilidad
Centralizado	Se debe modificar la aplicación	Facilidad al contar con un servidor central	Depende de la implementación, métodos criptográficos elevados	Un único punto de fallo, si no está disponible nadie puede acceder
Agente	Variedad de agente que pueden comunicarse con aplicaciones antiguas	Introduce otro elemento (agente) sobre el cual también se debe tener planificación de permisos	Los agentes pueden ser reemplazados por software malicioso	Los usuarios se pueden resistir o no comprender bien el concepto de agente
Pasarela	Es fácil su instalación y configuración	Se dificulta en entornos con múltiples pasarelas que	Se recomienda proteger la pasarela con un cortafuegos	Existe un punto único de fallo, como en el

		requieran de sincronización de usuarios	dedicado a detectar y prevenir ataques	modelo centralizado
Token		Se introduce otro elemento al sistema	Incrementa la seguridad	
Centralizado /Agente	El agente elimina la necesidad de modificar la aplicación	Adicional al modelo centralizado, se simplifica en cuanto a la distribución de agentes	El agente se autentica automáticamente sin la intervención del usuario	Existe un punto único de fallo por ser un tipo de modelo centralizado

Tabla 3. Comparación de los modelos de SSO

CAPÍTULO 3. OBJETIVOS Y METODOLOGÍA

3.1. Objetivo general

El objetivo general de este trabajo de fin de master es implementar un sistema Proveedor de Identidad (IdP) para ofrecer servicios de autenticación mediante Inicio de Sesión Único (SSO) en la Universidad del Cauca, haciendo uso de herramientas y estándares de libre distribución que minimicen al máximo el costo total de la solución.

3.2. Objetivos específicos

- Analizar el estado del arte para escoger la solución óptima para cubrir las necesidades del piloto experimental.
- Implementar el sistema de Inicio de Sesión Único (SSO) mediante la instalación del servidor WSO2 Identity Server.
- Verificar el correcto funcionamiento del sistema de Inicio de Sesión Único (SSO) mediante la utilización de aplicaciones de prueba.
- Configurar en producción Google Apps for Education como un proveedor de servicios y validar su correcto funcionamiento.
- Modificar una aplicación de producción para que funcione con el sistema Inicio de Sesión Único (SSO) y validar su correcto funcionamiento.

3.3. Metodología

Con el fin de desarrollar el piloto experimental y el trabajo de fin de master en general, se estableció el uso del modelo lineal secuencial, también conocido como modelo en cascada, el cual define una serie de fases o etapas dentro de las cuales se enmarcan las diversas actividades de este trabajo de fin de master. Las fases son:

- Capacitación.
- Análisis y diseño.
- Implementación.
- Análisis y validación de resultados
- Entrega.

3.3.1. Actividades y cronograma

A continuación se detallan las actividades a desarrollar en cada fase del desarrollo de este trabajo de fin de master.

Fase	Nombre	Actividades	Tiempo
1	Capacitación	1. Documentación y capacitación en sistemas proveedores de identidad 2. Documentación y capacitación acerca de Single Sign On y sus características. 3. Documentación y capacitación en el protocolo SAML 2.	3 semanas
2	Análisis y diseño	4. Análisis de especificaciones y requerimientos. 5. Diseño de la solución. 6. Diseño del plan de pruebas.	3 semanas
3	Implementación	7. Implementación del proveedor de identidad. 8. Ejecución de pruebas de funcionamiento del proveedor de identidad. 9. Configuración de aplicaciones de producción al proveedor de identidad	5 semanas
4	Redacción	10. Redacción de la memoria.	3 semanas
5	Entrega	11. Entrega de productos (Documento final). 12. Preparación y realización de la sustentación.	1 semana

Tabla 4. Detalle de actividades de cada fase

En la Ilustración 6 se presenta el diagrama de Gantt de las fases que se cumplirán durante el desarrollo de este piloto experimental.

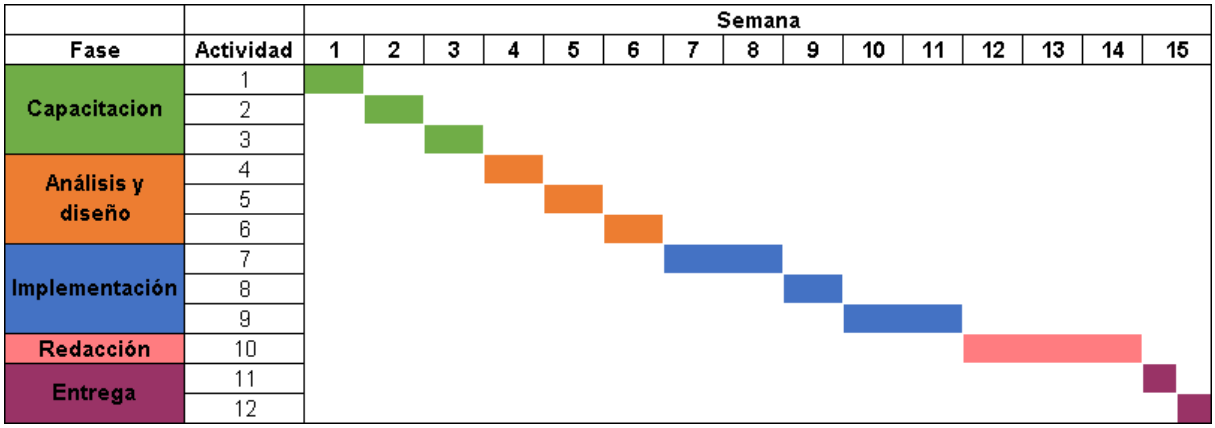

Ilustración 6. Diagrama de Gantt para las fases del piloto experimental

CAPÍTULO 4. IMPLEMENTACIÓN Y VALIDACIÓN DE LA SOLUCIÓN

4.1. Implementación de la solución

El experimento se desarrollara en el centro de datos de la Universidad del Cauca, esta organización educativa cuenta con alrededor de 30.000 usuarios en el almacén de usuarios.

A los usuarios se les ofrecen diversidad de servicios mediante aplicaciones web desarrollados al interior de la Universidad por parte del área de desarrollo de la División de Tecnologías de Información y Telecomunicaciones y también hay un contrato para los servicios de correo y herramientas de colaboración con Google Apps for Education.

Cada aplicación implementa una lógica interna que mediante un formulario recibe el usuario institucional y la contraseña y lo valida contra el almacén de datos institucional para tomar decisiones de autenticación.

Es de rescatar que en épocas de matrículas académicas, que es cuando toda la comunidad universitaria hace uso de los servicios con mayor intensidad, las peticiones hacia los servidores LDAP se incrementan considerablemente.

En la Ilustración 7 podemos observar la arquitectura actual de la organización, se observan múltiples aplicaciones con lógicas de autenticación independientes, además se observa que existen dos almacenes de datos independientes que requieren de sincronización continua, lo cual adiciona un reto adicional. Este paso adicional requiere de recursos adicionales como lo es un servidor dedicado que soporte las herramientas de sincronización que provee Google Apps for Education como lo es Google Apps Directory Syncing (GADS).

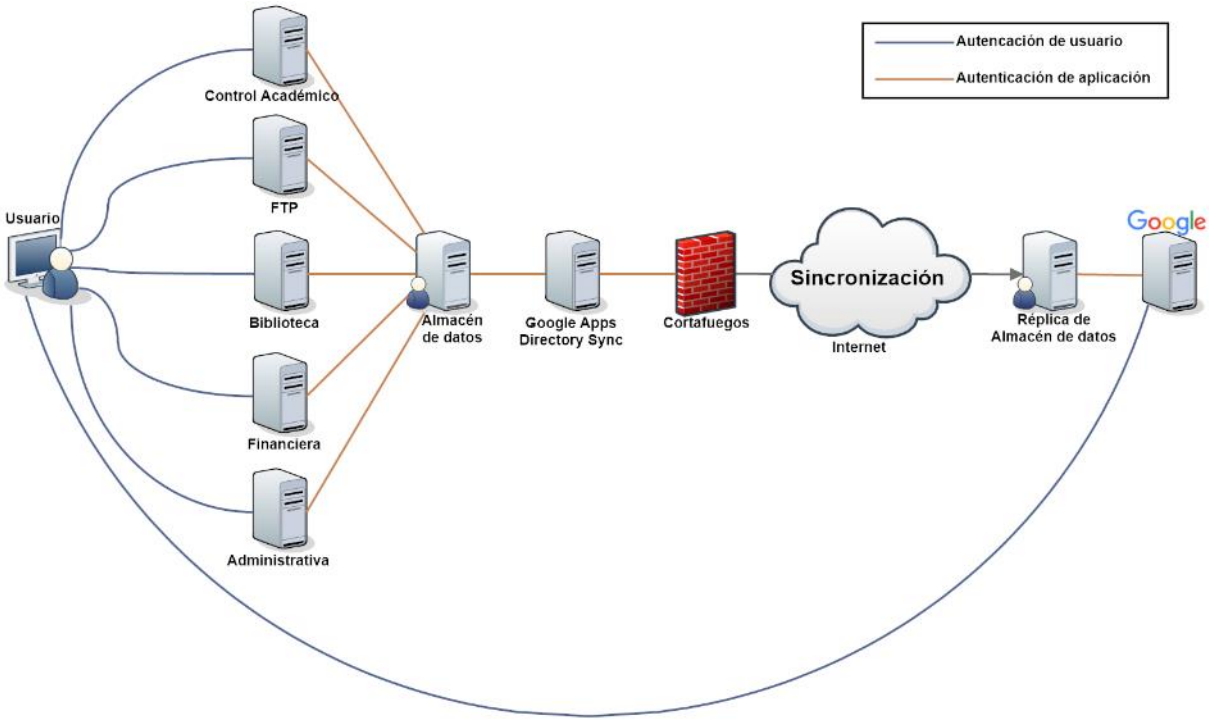

Ilustración 7. Arquitectura actual

La arquitectura no es la adecuada para una organización en crecimiento que necesita cada día escalar sus aplicaciones y los usuarios también van a sentir la fatiga de la autenticación cuando el número de aplicaciones y servicios se incrementen.

El piloto experimental plantea la arquitectura de la Ilustración 8, la cual centraliza la autenticación a un elemento denominado Proveedor de Identidad (IdP) el cual será el único elemento que interactuará con el almacén de usuarios logrando así un nivel de seguridad mayor ya que no estará en contacto directo con las aplicaciones. Así mismo se elimina la lógica de autenticación propia dentro de cada aplicación y esta actividad será realizada por agentes que se comunicaran con el Proveedor de Identidad (IdP) solicitándole realice la lógica de autenticación para determinar si un usuario es válido o no.

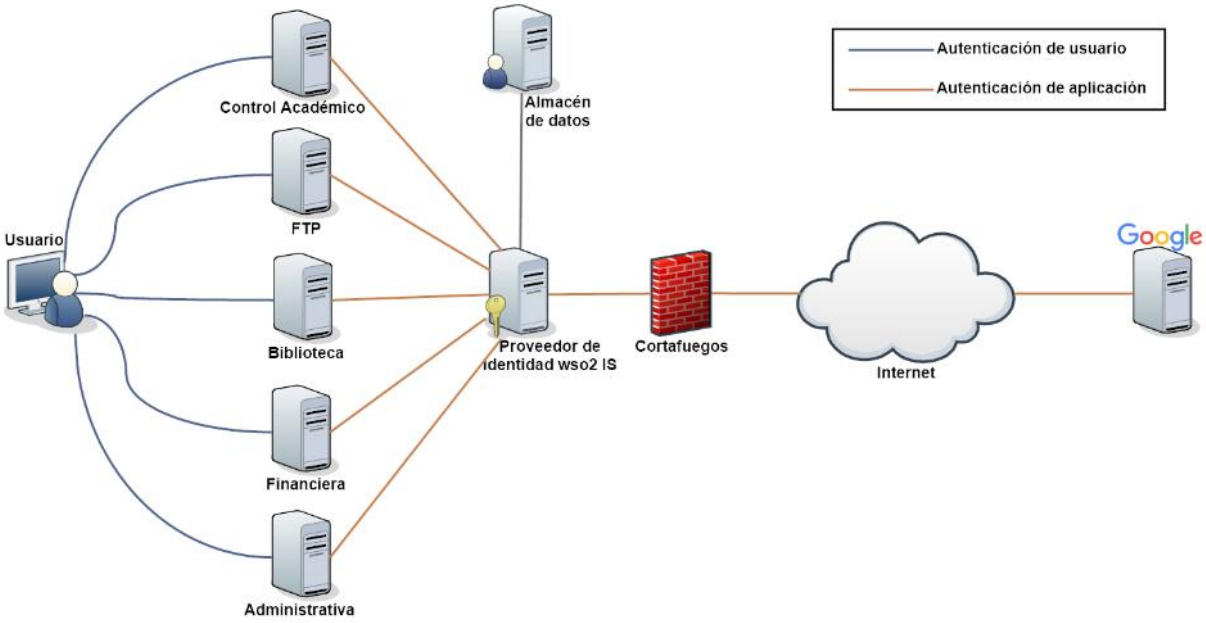

Ilustración 8. Arquitectura entorno de Inicio de Sesión Único (SSO)

4.1.1. Tecnologías utilizadas

La elección de las tecnologías a utilizar en el piloto experimental es de vital importancia para cumplir con el objetivo del trabajo de fin de master. Todos los elementos involucrados en el ejercicio deben ser de libre distribución y su uso no debe estar limitado a subscripciones con costos adicionales. El código debe ser libre para poder adaptarse a las necesidades de personalización de cada organización. Se escogen las tecnologías basándose en la premisa de que lo que se quiere siempre por parte de las organizaciones es que el proyecto se concluya en el menor tiempo posible por lo que se deben escoger herramientas afianzadas en el mercado, seguras, fáciles de instalar, configurar y administrar.

El primer elemento a describir es el sistema operativo sobre el cual se va a soportar el servidor del proveedor de identidad, que debe ser basado en Linux. De la gran cantidad de distribuciones para servidores se escoge Debian 8.1 debido a que es un sistema operativo seguro y estable, además, en el centro de datos de la organización la mayoría de equipos Linux se encuentran usando esta distribución lo que facilitaría el mantenimiento del mismo.

Se debe analizar de todas las herramientas presentadas en el capítulo 2 se debe analizar cuál es la que mejor se adapta a las necesidades de la organización para ello se tienen en cuenta los siguientes aspectos:

- **Compatibilidad:** se refiere a la variedad de estándares que implemente la herramienta, a pesar de que este trabajo de fin de master se enfoca solo en el estándar SAML, la idea es que a mediano plazo se implemente también mecanismos de autorización en el Proveedor de Identidad (IdP) por lo que es importante que la

solución incluye estándares que permitan soportar esta funcionalidad en el mismo servidor para optimizar los recursos de la organización y facilitar la administración del sistema completo. La compatibilidad se evaluará en tres categorías a las cuales se les asignará un puntaje para obtener una ponderación final. La puntuación mínima es un punto (1), e indica que solo admite un estándar de autenticación, dos puntos (2) indica que es compatible con múltiples estándares de autenticación y tres puntos (3) significa que la herramienta soporta estándares de autenticación y de autorización.

- **Sistema operativo:** como se explicó este aspecto es de vital importancia para el presente trabajo de fin de master, ya que uno de los objetivos de la herramienta es que sea una alternativa económica frente a las actuales soluciones que se ofrecen en el mercado, razón por la que se busca que todas las herramientas software inmersas en su implementación sean soportados por los sistemas operativos de libre distribución basados en Linux. Teniendo en cuenta las razones expuestas, si la herramienta es soportada en todos los sistemas operativos (Windows, OS X y Linux) se le otorgará una calificación de tres puntos (3), si la herramienta es soportada en sistemas operativos Linux se le dará dos (2) puntos y si solo es soportada para sistemas operativos Microsoft Windows y Mac OS X no se le otorgará ningún punto (0).
- **Licenciamiento:** la evaluación de la licencia de distribución está directamente relacionada con el costo de adquisición de la herramienta. Si ésta es de libre distribución y sin restricciones de tiempo obtiene una calificación de dos puntos (2), en caso de ser una herramienta comercial pero que cuente con una versión con limitaciones de libre distribución se le otorga una calificación de un punto (1) y si la herramienta solo posee una licencia de software propietario recibe una calificación de cero puntos (0).
- **Administración:** la forma en la que los administradores van a interactuar con el servidor es determinante al momento de escoger una herramienta que permita invertir el menor tiempo en su configuración es lo ideal para ofrecer tiempos de respuesta mínimos. Si la administración de la herramienta se realiza a través de archivos de configuración y también ofrece una interfaz gráfica de administración se le otorgará dos (2) puntos y si la configuración se realiza solo por un medio ya sea archivos de configuración o interfaz gráfica se le otorgará un (1) punto.
- **Visualización:** la personalización de todas las interfaces de usuario que use la herramienta permite mantener la imagen institucional de la organización y fomentan el sentido de pertenencia de los usuarios. Si la herramienta desea personalizar por completo la interfaz de administración se le otorgará una calificación de tres (3) puntos, si solo permite agregar el logo y el nombre de la organización se le otorgará un (1) punto y si no ofrece opciones de personalización no se le dará ningún punto.

- **Soporte:** el soporte es un aspecto relevante ya que se buscan herramientas que sean adoptadas en un alto nivel por la comunidad para que cuenten con grupos o foros de ayuda en donde se pueda encontrar soporte a problemas de instalación, configuración y administración de la plataforma, debido a que se busca que la solución funcione con un mínimo de costos asociados para la organización no se desea adquirir subscripciones por soporte con la institución. De esta manera la calificación se pondera de la siguiente manera, si la herramienta cuenta con soporte únicamente por suscripción se le otorgara una calificación de cero puntos (0) y si cuenta con una comunidad que la soporta de manera gratuita se le dará una calificación de dos puntos (2).
- **Actualizaciones:** las liberación de las actualizaciones por parte del desarrollador de la herramienta es muy importante para mantener la seguridad de toda la solución ya que en comúnmente en los parches de actualización se solucionan bug´s de seguridad que pueden comprometer la información almacenada en los equipos donde está instalada la herramienta y en este caso se trata de información de identidad de alrededor de 30.000 usuarios, lo cual es de carácter sensible y debe ser tratada con el mayor cuidado. Si la herramienta ofrece todas las actualizaciones bajo la misma licencia de libre distribución del producto principal se le otorgara una calificación de tres (3) puntos, si la herramienta solo ofrece de forma gratuita el producto y no las actualizaciones se le otorga una calificación de un (1) punto y si la aplicación no se actualiza constantemente sino solo en cambios de versiones grandes no se le dará ningún punto.
- **Alta disponibilidad:** debido a que el sistema debe estar siempre disponible por que la información que maneja es indispensable para poder ofrecer servicios mediante el resto de aplicativos de la organización, el sistema debe permitir configuraciones de alta disponibilidad (clúster) para configurar en situaciones de alta demanda. Si la herramienta permite configurar alta disponibilidad de manera nativa se le otorga una calificación de tres (3) puntos, si se puede realizar esta configuración pero con herramientas adicionales de terceros se le calificará con un (1) punto y si no puede trabajar en un entorno de alta disponibilidad no se le dará ningún punto.

A continuación se presenta la Tabla 5 que es la evaluación de las herramientas de Inicio de Sesión Único (SSO), en este evaluación no se ha tenido en cuenta la herramienta CAS ya que no implementa de forma nativa el protocolo SAML, además las herramientas que se basan en la nube tampoco se tienen en cuenta para la evaluación por que el requerimiento es que la infraestructura sea implementada de manera local de las instalaciones de la organización.

Categoría / Herramienta	Gluu	Josso	Openam	Shibboleth	Simplesamlphp	Wso2is
Compatibilidad	3	3	3	2	1	3
Sist. operativo	2	3	3	3	3	3
Licenciamiento	1	1	2	2	2	2
Administración	1	1	2	1	1	2
Visualización	3	1	3	3	3	3
Soporte	2	2	2	2	2	2
Actualizaciones	3	3	1	3	3	3
Clustering	0	0	3	3	1	3
CALIFICACIÓN	15	14	19	19	16	21

Tabla 5. Evaluación de las herramientas

Como se observa claramente en la Tabla 5 la herramienta que obtuvo la mayor calificación es la herramienta WSO2 Identity Server y a partir de los requerimientos establecidos por el fabricante [33] para la instalación de esta herramienta en particular se procede a analizar que más tecnologías se deben implementar.

Para poder ejecutar la herramienta es necesario utilizar una máquina virtual de JAVA, para ello se usa Oracle Java SE Development Kit (JDK) en la versión 1.7, ya que la última versión no es compatible como tampoco recomiendan OpenJDK.

En la Tabla 6 se presentan los sistemas operativos donde han sido probadas todas las características de la herramienta, aunque el sistema operativo Debian no ha sido completamente probado el fabricante especifica que lo único que se necesita es que el entorno JAVA sea compatible y por ende debe correr en la mayoría de sistemas operativos.

Producto	Versión	Windows	Red Hat	Ubuntu	CentOS	Fedora	MAC OS X	Debian
WSO2 Identity Server	5.0.0	Windows 2008 server	Red Hat Enterprise Linux Server release 5.5 (Tikanga)	Ubuntu 13.10 Ubuntu 14.04 Ubuntu 12.04	No ha sido probado	No ha sido probado	No ha sido probado	No ha sido probado

Tabla 6. Compatibilidad de la herramienta WSO2 IS

El almacén de datos que la herramienta WSO2 IS utilizará es el directorio activo que ya está en funcionamiento en la organización y se encuentra soportado por OpenLDAP, el Proveedor de Identidad (IdP) no aprovisionara al LDAP de nuevos usuarios, solo lo usara como almacén de consulta de usuarios por lo que las conexiones que se realicen hacia el mismo deben ser de solo lectura.

Como los servicios que se van a probar y a los cuales se les va a implantar el Inicio de Sesión Único (SSO) se prestan mediante aplicaciones web se usara el perfil de Inicio de Sesión Único (SSO) que viene predeterminado en el servidor WSO2 IS. Todas estas aplicaciones se acceden mediante un navegador y la herramienta debe ofrecer el mayor grado de compatibilidad para facilitar el acceso de los usuarios. Para la etapa de pruebas se debe usar un navegador que permita el análisis cookies y del flujo de mensajes entre el Proveedor de Identidad (IdP) y el Proveedor de Servicios (SP), para ello se va a utilizar el navegador Firefox de Mozilla en la versión 40.0.3, se escoge este navegador en específico ya que cuenta con dos extensiones particulares que servirán para usar en el proceso de análisis de las pruebas y son los siguientes:

- **SSO Tracer:** es un depurador para mensajes del estándar SAML mediante el filtrado de solicitudes HTTP [34].
- **Firebug:** integra herramientas de desarrollo al navegador Firefox, que inspecciona el código en tiempo real [35].

Para la creación de un conector SAML basado en JAVA para utilizar en las diferentes aplicaciones web de la organización se utiliza la librería OpenSAML que es un producto de libre distribución de Internet2 y soporta las versiones 1.0, 1.1 y 2.0 de SAML [36].

4.1.2. Instalación de WSO2 Identity Server

El primer paso para ejecutar el la herramienta es descargar el JDK[12] de JAVA en su versión 1.7, disponible en el siguiente enlace: http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html, de donde se obtiene la versión del paquete para Linux con arquitectura de 64 bits.

Se descomprime el archivo y se cambian los permisos del directorio y los archivos para que sean propiedad del usuario que se creó para tal fin:

```
root@sso:~# cd /usr/src/
root@sso:~# tar -xzvf jdk-7u79-linux-x64.tar.gz
root@sso:~# chown -R sso.sso-ssi jdk1.7.0_79
```

Se agrega la siguiente información al archivo */etc/profiles* para que se agregue la información de la ruta de los archivos ejecutables de java a las variables del sistema en cada reinicio:

```
JAVA_HOME=/usr/src/jdk1.7.0_79
PATH=$PATH:/usr/src/jdk1.7.0_79/bin
export JAVA_HOME
export PATH
```

12

Se debe verificar que el sistema reconozca la ruta del archivo ejecutable de java mediante el siguiente comando:

```
root@sso:~# java -version
java version "1.7.0_79"
Java(TM) SE Runtime Environment (build 1.7.0_79-b15)
Java HotSpot(TM) 64-Bit Server VM (build 24.79-b02, mixed mode)
```

Se procede a la descarga del paquete que contiene el aplicativo de WSO2 Identity Server disponible en el siguiente enlace: http://wso2.com/products/identity-server, mediante un registro previo de información personal, se habilita la descarga de los binarios de la versión 5.0.0 del aplicativo y del respectivo paquete de actualizaciones o “service pack”.

Se deben ubicar los dos directorios descargados en el mismo nivel para proceder con la instalación del paquete de actualización,

```
/usr/src/
    |--wso2is-5.0.0
    |--WSO2-IS-5.0.0-SP01
```

Se configuran los permisos de ambos directorios para que el propietario sea el usuario “sso”, así:

```
root@sso:/usr/src# chown -R sso.sso-ssi wso2is-5.0.0
root@sso:/usr/src# chown -R sso.sso-ssi WSO2-IS-5.0.0-SP01
```

Al ingresar al directorio del paquete de actualización se ejecuta el siguiente comando para actualizar el paquete base del aplicativo WSO2 IS.

```
root@sso:/usr/src/WSO2-IS-5.0.0-SP01# su -c "sh install_sp.sh" sso
[Warning] This Service Pack is only for a fresh Identity Server installation!
This will override all the existing configuration files and will delete the
internal H2 database.
Do you want to continue ? [y/n]:y
-------------------------------------------------------------------------
-                       WSO2 Identity Server SP1                        -
-------------------------------------------------------------------------
.........................................................................
Copying the wso2carbon-version.txt file to <CARBON_SERVER>/bin.
.........................................................................
Copying the patch1016 to <CARBON_SERVER>/repository/components/patches/
.........................................................................
Copying oauth2.war ,authenticationendpoint.war and wso2.war files inside
resources/webapps of the patch, to
<CARBON_SERVER>/repository/deployment/server/webapps/ folder.
.........................................................................
Copying the sql script files inside resources/dbscripts/identity folder of the
patch to <CARBON_SERVER>/dbscripts/identity folder.
.........................................................................
Copying the jar files inside resources/lib folder of the patch to
<CARBON_SERVER>/lib/ folder.
.........................................................................
Copying 'samlsso_federate.html' file inside 'resources' folder of the patch to
<CARBON_SERVER>/repository/resources/security/ folder.
.........................................................................
Copying configuration files inside 'resources/conf' folder of the patch to
<CARBON_SERVER>/repository/conf/ folder.
.........................................................................
Copying updated H2 database to repository/databases
.........................................................................
```

Al finalizar la copia de las actualizaciones se procede a ingresar al directorio “bin” del aplicativo para iniciar por primera vez el servicio de WSO2 Identity Server, se ejecutan los siguientes comandos para iniciar el servidor, aquí se incluye el comando *“-Dsetup”* que se usa para inicializar la base de datos interna del aplicativo:

```
root@sso:~# chmod +x /usr/src/wso2is-5.0.0/bin/wso2server.sh
root@sso:/usr/src/wso2is-5.0.0/bin# su -c "sh wso2server.sh -Dsetup" sso
```

Al finalizar el primer inicio se contará con el servidor de WSO2 Identity Server 5.0.0 corriendo con la configuración por defecto y con el paquete de actualizaciones 01.

Ahora se crea el archivo de inicio para que el servidor de WSO2 IS se ejecute como un servicio del sistema, el archivo será */etc/systemd/system/sso.service* y debe incluir los siguientes comandos:

```
[Unit]
Description=Servidor de Identidad - WSO2 Identity Server
After=syslog.target

[Service]
TimeoutStartSec=0
Type=simple
User=sso
Group=sso-ssi
Environment=JAVA_HOME=/usr/src/jdk1.7.0_79
ExecStart=/usr/src/wso2is-5.0.0/bin/wso2server.sh
ExecStop=/usr/src/wso2is-5.0.0/bin/wso2server.sh stop
Restart=on-abort

[Install]
WantedBy=multi-user.target
```

Se debe activar el servicio para que se inicie automáticamente en los niveles especificados mediante el siguiente comando:

```
root@sso:/etc/systemd/system# systemctl enable sso.service
```

De ahora en adelante los comandos para iniciar, reiniciar y detener el servicio de WSO2 IS son los siguientes:

```
root@sso:~# systemctl start sso.service

root@sso:~# systemctl stop sso.service

root@sso:~# systemctl status sso.service
● sso.service - Servidor de Identidad - WSO2 Identity Server
   Loaded: loaded (/etc/systemd/system/sso.service; enabled)
   Active: active (running) since mar 2015-08-25 18:39:18 COT; 1min 47s
ago
 Main PID: 518 (wso2server.sh)
   CGroup: /system.slice/sso.service
           ├─518 /bin/sh /usr/src/wso2is-5.0.0/bin/wso2server.sh
           └─812 /usr/src/jdk1.7.0_79/bin/java -Xbootclasspath/a: -
Xms256m -Xmx1024m -XX:MaxPermSize=256m -XX:+HeapDumpOnOutOfMemoryError
-XX:HeapDumpPath=/usr/src/wso2is-5.0.0/repository/logs/heap-dump.hprof
-Dcom.sun.management.jmxremote -classpath /usr/src/jdk1.7....
```

4.1.3. Configuración de WSO2 Identity Server

4.1.3.1. Configuración del nombre del servidor

Para asegurar la identidad del servidor se debe configurar el nombre de servidor o hostname en los archivos de configuración propios de WSO2 Identity Server y deben coincidir con el nombre que aparece en el certificado público otorgado por la autoridad certificadora de confianza, es decir con el “Common Name (CN)” para el cual se adquirió el certificado.

El nombre de dominio para el servidor es sso.unicauca.edu.co y el nombre del servidor es sso, por lo que se actualiza el archivo */usr/src/wso2is-5.0.0/repository/conf/carbon.xml* en las etiquetas “HostName” y “MgtHostName”:

```
<!--
   Host name or IP address of the machine hosting this server
   e.g. www.wso2.org, 192.168.1.10
   This is will become part of the End Point Reference of the
   services deployed on this server instance.
-->
<HostName>sso.unicauca.edu.co</HostName>

<!--
Host name to be used for the Carbon management console
-->
<MgtHostName>sso.unicauca.edu.co</MgtHostName>
```

4.1.3.2. Creación del almacén de llaves

Uno de los primeros pasos en la configuración es cambiar el almacén de llaves o “kesytore” que usa el aplicativo para las actividades relacionadas con la seguridad y la criptografía de llave publica, como lo es la autenticación de las comunicaciones sobre protocolo TLS (Transport Layer Security), la firma de mensajes, entre otras.

Existen dos almacenes de llaves por defecto identificados como “client-truststore.jks”, que contiene los certificados de confianza y “wso2carbon.jks”, que contiene un par de llaves creadas para un servidor con hostname localhost. Ambos almacenes se protegen con contraseñas conocidas ya que se distribuyen como ejemplo en todos los productos de la empresa WSO2.

El servidor se utilizara bajo el dominio de sso.unicauca.edu.co, para cual se debe tener una llave privada de servidor y un certificado firmado por una autoridad certificadora de confianza que valide la identidad del servidor, en este caso se usa un certificado de tipo wildcard a partir del cual se creara el almacén de llaves que contiene el par de llaves necesarias para la operación del aplicativo.

Los almacenes de llaves se deben almacenar en la ruta */usr/src/wso2is-5.0.0/repository/resources/security/*. Para crear el almacén de llaves se deben exportar en formato PKCS12 mediante el siguiente comando, incluyendo contraseñas generadas aleatoriamente para mantener la seguridad de la llave privada, además se debe tener en cuenta que debido a limitaciones de tomcat la contraseña del almacén de llaves debe ser la misma de la llave privada.

```
root@sso:/usr/src/wso2is-5.0.0/repository/resources/security/# openssl
pkcs12 -export -in wildcard_unicauca.cer -inkey wildcard_unicauca.key -
name "wildcard_unicauca" -out wildcard_unicauca.pfx
```

Del paso anterior obtenemos un solo archivo “wildcard_unicauca.pfx” en el formato requerido que incluye tanto la llave pública como la llave privada, a partir del cual se genera el almacén de llaves, así:

```
root@sso:/usr/src/wso2is-5.0.0/repository/resources/security/# keytool -
importkeystore -srckeystore wildcard_unicauca.pfx -srcstoretype pkcs12 -
destkeystore wildcard_unicauca.jks -deststoretype JKS
```

A continuación se adiciona la llave pública al almacén de llaves de confianza “client-truststore.jks”, cambiando primero la contraseña de dicho almacén por una generada aleatoriamente:

```
root@sso:/usr/src/wso2is-5.0.0/repository/resources/security/# keytool -
storepasswd -keystore client-truststore.jks

root@sso:/usr/src/wso2is-5.0.0/repository/resources/security/# keytool -
export -alias wildcard_unicauca -keystore wildcard_unicauca.jks -file
wildcard_unicauca.pem

root@sso:/usr/src/wso2is-5.0.0/repository/resources/security/# keytool -
import -alias wildcard_unicauca -file wildcard_unicauca.pem -keystore
client-truststore.jks
```

Cuando se ha creado el nuevo almacén de llaves y actualizado el existente se deben actualizar ciertos archivos de configuración que se encuentran en el directorio */usr/src/wso2is-5.0.0/repository/conf*, estos archivos son los siguientes:

```
axis2/axis2.xml
carbon.xml
identity.xml
log4j.properties
security/application-authentication.xml
security/secret-conf.properties
```

4.1.3.3. Configuraciones de seguridad adicionales

La configuración de seguridad de tomcat se realiza en el archivo */usr/src/wso2is-5.0.0/repository/conf/tomcat/catalina-server.xml*, en este archivo de configuración se controla el tipo de protocolos de seguridad que se manejaran a través del puerto seguro 9443, ya que por los ya conocidos ataques al protocolo SSL v3, como el ataque “POODLE”, es imperativo desactivar los protocolos vulnerables para permitir solo comunicaciones a través de las diferentes versiones del protocolo TLS, esta configuración se controla agregando la etiqueta “sslEnabledProtocols”:

```
<Connector  protocol="org.apache.coyote.http11.Http11NioProtocol"
                port="9443"
                bindOnInit="false"
                sslProtocol="TLS"
                sslEnabledProtocols="TLSv1,TLSv1.1,TLSv1.2"
                ...
>
```

Para verificar que la configuración de los protocolos de seguridad se ha aplicado de manera correcta, se usa una herramienta que permite obtener la información requerida en la línea de comandos y está disponible en el enlace http://www.bolet.org/TestSSLServer/TestSSLServer.jar, se descarga y se utiliza pasándole como argumento la dirección y el puerto al cual se va a analizar:

```
root@sso:~# wget http://www.bolet.org/TestSSLServer/TestSSLServer.jar
root@sso:~# java -jar TestSSLServer.jar localhost 9443
Supported versions: TLSv1.0 TLSv1.1 TLSv1.2
Deflate compression: no
Supported cipher suites (ORDER IS NOT SIGNIFICANT):
  TLSv1.0
     RSA_WITH_RC4_128_MD5
     RSA_WITH_RC4_128_SHA
     RSA_WITH_3DES_EDE_CBC_SHA
     DHE_RSA_WITH_3DES_EDE_CBC_SHA
     RSA_WITH_AES_128_CBC_SHA
     DHE_RSA_WITH_AES_128_CBC_SHA
     TLS_ECDHE_RSA_WITH_RC4_128_SHA
     TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
     TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
  (TLSv1.1: idem)
  TLSv1.2
     RSA_WITH_RC4_128_MD5
     RSA_WITH_RC4_128_SHA
     RSA_WITH_3DES_EDE_CBC_SHA
     DHE_RSA_WITH_3DES_EDE_CBC_SHA
     RSA_WITH_AES_128_CBC_SHA
     DHE_RSA_WITH_AES_128_CBC_SHA
     RSA_WITH_AES_128_CBC_SHA256
     DHE_RSA_WITH_AES_128_CBC_SHA256
     TLS_ECDHE_RSA_WITH_RC4_128_SHA
     TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
     TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
     TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
-----------------------
Server certificate(s):
  4dceb6a275871c15716d73f06827e86df425900d: CN=*.unicauca.edu.co,
O=Universidad del Cauca, OU=División de las TIC, L=Popayan, ST=Cauca,
C=CO
-----------------------
Minimal encryption strength:     strong encryption (96-bit or more)
Achievable encryption strength:  strong encryption (96-bit or more)
```

Se puede verificar la correcta configuración del nombre del servidor y el certificado visitando la página de administración del servidor WSO2 Identity Server en el enlace https://sso.unicauca.edu.co:9443/carbon y verificar que la información de certificado que se presenta es la que se configuro en el almacén de llaves, como se muestra en la Ilustración 9.

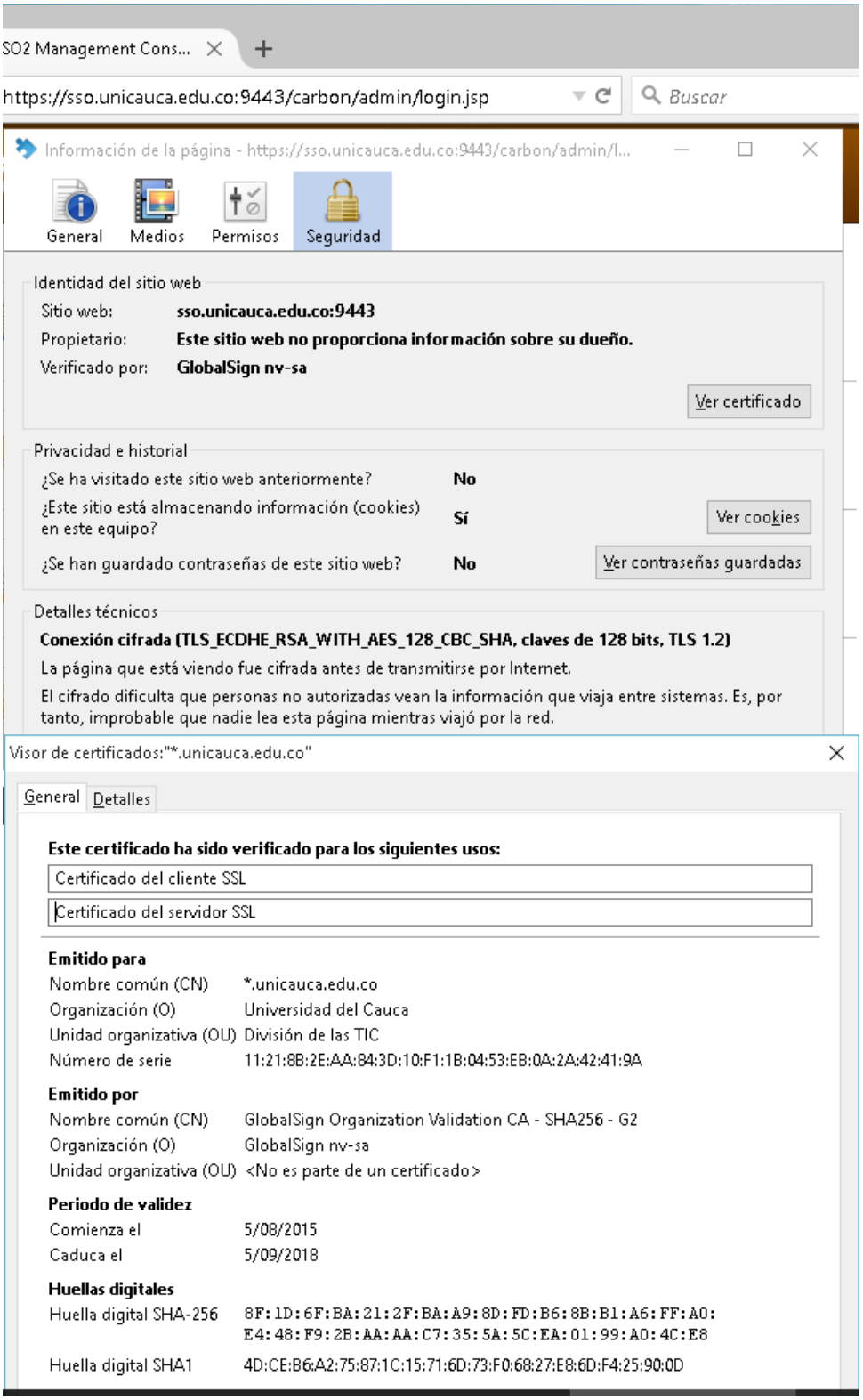

Ilustración 9. Certificado digital del servidor

Además se puede iniciar sesión credenciales para verificar el correcto funcionamiento del servidor y visitar la pestaña “Home” en donde se encuentra información relacionada a la instalación del servidor y su entorno, como se observa en la Ilustración 10, donde entre otra información se observa el nombre del equipo, el tiempo de servicio, sistema operativo, máquina virtual de java y la base de datos interna.

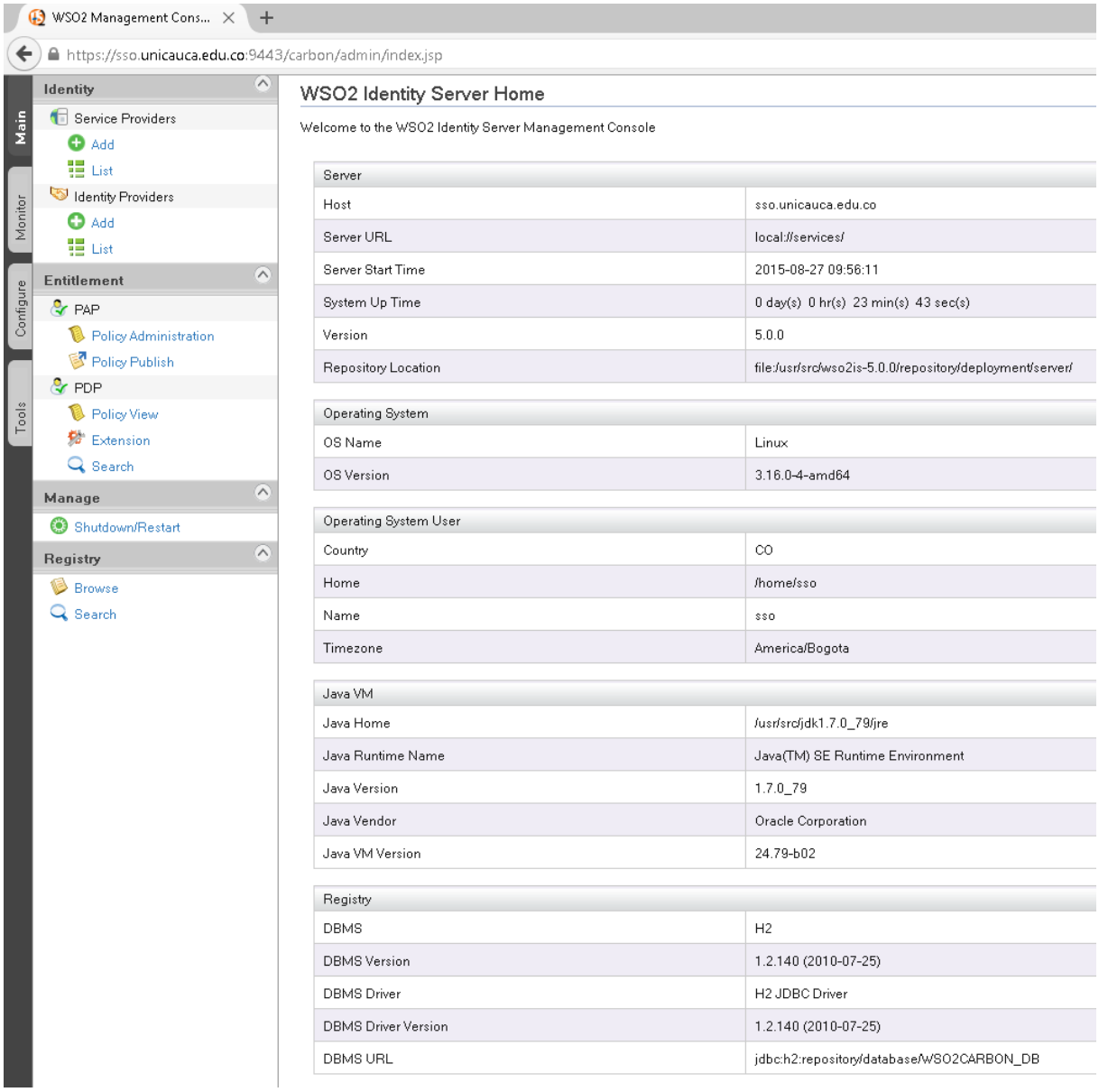

Ilustración 10. Información del servidor

4.1.3.4. Configuración del almacén de usuario

Los almacenes de usuario se configuran en el archivo */usr/src/wso2is-5.0.0/repository/conf/user-mgt.xml* en el cual se encuentran conectores para diferentes tipos de almacenes predefinidos, se usa el conector para LDAP en modo solo lectura ya que en este piloto experimental no se realizara ningún aprovisionamiento al directorio de la organización.

Para este conector se debe contar con un usuario que pueda leer todos los atributos en los objetos del directorio LDAP y que no tenga permisos para realizar modificaciones en el mismo.

Mediante el atributo “UserNameAttribute” se controla mediante qué información se realiza la búsqueda en el directorio para cada autenticación de usuario y el atributo “ReadGroups” se fija en el valor “false” ya que en este caso se trabajó solo con atributos individuales y no se permite autenticaciones por pertenencia o membresía a grupos.

```
<UserStoreManager
class="org.wso2.carbon.user.core.ldap.ReadOnlyLDAPUserStoreManager">
    <Property
name="TenantManager">org.wso2.carbon.user.core.tenant.CommonHybridLDAPT
enantManager</Property>
    <Property name="ReadOnly">true</Property>
    <Property name="Disabled">false</Property>
    <Property name="MaxUserNameListLength">100</Property>
    <Property name="ConnectionURL">ldap://10.20.6.135:389</Property>
    <Property
name="ConnectionName">cn=ReaderAD,dc=unicauca,dc=edu,dc=co</Property>
    <Property name="ConnectionPassword">XXXXXXXXX</Property>
    <Property name="passwordHashMethod">MD5</Property>
    <Property name="UserSearchBase">dc=unicauca,dc=edu,dc=co</Property>
    <Property name="UserNameListFilter">(objectClass=person)</Property>
    <Property
name="UserNameSearchFilter">(&amp;(objectClass=person)(uid=?))</Propert
y>
    <Property name="UserNameAttribute">uid</Property>
    <Property name="ReadGroups">false</Property>
    <Property name="GroupSearchBase">ou=system</Property>
    <Property
name="GroupNameListFilter">(objectClass=groupOfNames)</Property>
    <Property
name="GroupNameSearchFilter">(&amp;(objectClass=groupOfNames)(cn=?))</P
roperty>
    <Property name="GroupNameAttribute">cn</Property>
    <Property name="SharedGroupNameAttribute">cn</Property>
    <Property
name="SharedGroupSearchBase">ou=SharedGroups,dc=wso2,dc=org</Property>
    <Property
name="SharedGroupNameListFilter">(objectClass=groupOfNames)</Property>
    <Property
name="SharedTenantNameListFilter">(objectClass=organizationalUnit)</Pro
perty>
    <Property name="SharedTenantNameAttribute">ou</Property>
    <Property
name="SharedTenantObjectClass">organizationalUnit</Property>
    <Property name="MembershipAttribute">member</Property>
    <Property name="UserRolesCacheEnabled">false</Property>
    <Property name="ReplaceEscapeCharactersAtUserLogin">true</Property>
    <Property name="MaxRoleNameListLength">100</Property>
    <Property name="MaxUserNameListLength">100</Property>
    <Property name="SCIMEnabled">false</Property>
</UserStoreManager>
```

Se debe configurar el “REALM” que es el elemento que le indica al servidor quien será el usuario administrador del sistema, y en casos de configuración de lectura y escritura se especifica si lo debe crear. En este caso como el directorio se configura en modo solo lectura le indicamos que no debe crear usuarios y que el usuario de administración es “sanrestrepo”, la contraseña no se debe ingresar en este archivo ya que esta será leída del directorio.

```
<Realm>
        <Configuration>
        <AddAdmin>false</AddAdmin>
                <AdminRole>admin</AdminRole>
                <AdminUser>
                     <UserName>sanrestrepo</UserName>
                     <Password>XXXXXX</Password>
                </AdminUser>
            <EveryOneRoleName>everyone</EveryOneRoleName>
            <Property name="dataSource">jdbc/WSO2CarbonDB</Property>
        </Configuration>
```

4.1.3.5. Configuración de registro de eventos

El registro de eventos es una función muy importante en un servidor en producción, esta actividad es vital para la detección de errores, ataques y detectar patrones de uso de la herramienta. Se configura en el archivo */usr/src/wso2is-5.0.0/repository/conf/log4j.properties* el cual implementa un mecanismo basado en Apache Commons Logging.

Los registros de eventos se pueden configurar a través de la interfaz de administración y los cambios realizados por este medio persisten en la base de datos del servidor.

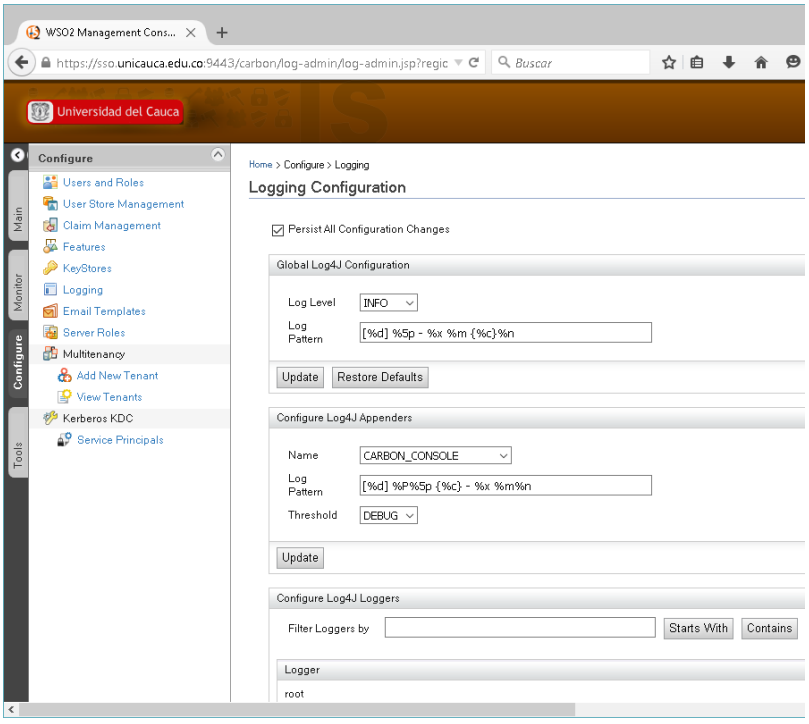

Ilustración 11. Administración del registro de eventos

En este caso se edita el archivo de configuración para especificar el nivel de “DEBUG” al componente de identidad para poder ver los asertos SAML en registro de eventos, para ello se agrega la siguiente línea al archivo en cuestión:

```
log4j.logger.org.wso2.carbon.identity=DEBUG
```

Mediante la interfaz de administración existe una herramienta de monitoreo del registro muy útil para la visualización de los elementos como se observa en la Ilustración 12.

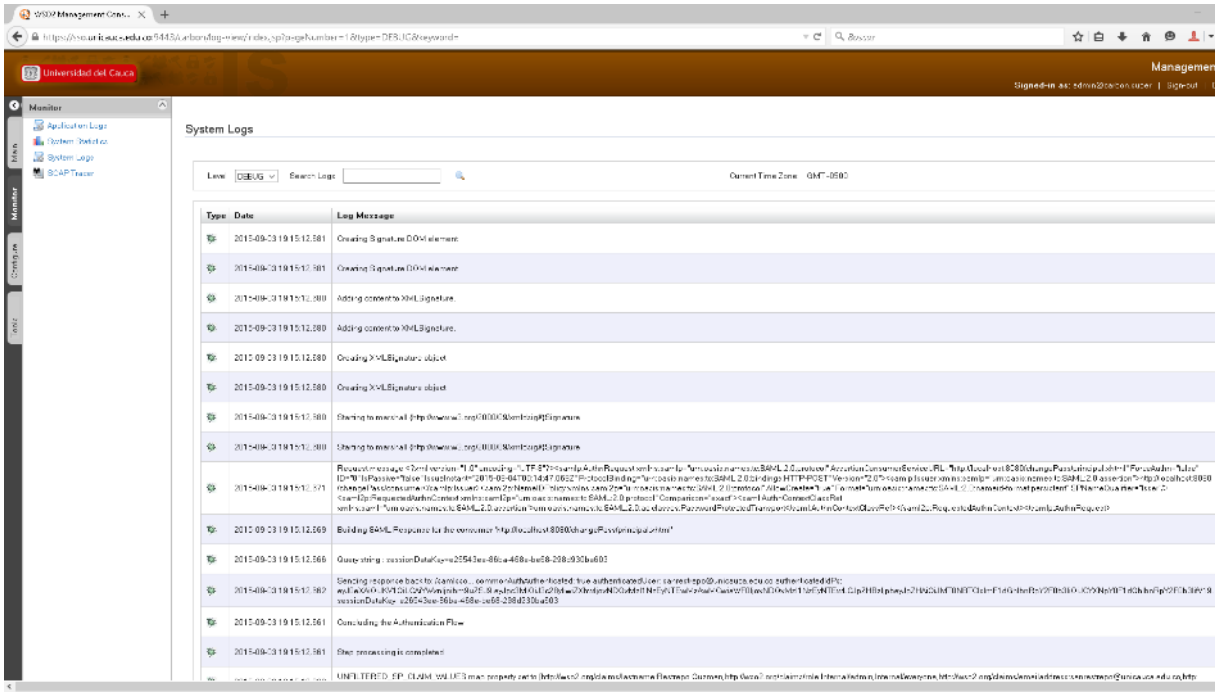

Ilustración 12. Interfaz de monitoreo del registro de eventos

4.1.4. Configuración del proveedor de identidad

El proveedor de identidad es la entidad encargada de administrar la información de identificación que se usa en los tokens SAML2. Dentro de la página de administración del servidor hay un menú para la configuración de los proveedores de identidad, ver la ilustración, en este caso se usa el proveedor de identidad local o “Resident Identity Provider” ya que no necesitamos de federación o características avanzadas y en la arquitectura propuesta solo existe un proveedor de identidad el cual se encarga de la validación de las peticiones de autenticación.

Se edita el archivo */usr/src/wso2is-5.0.0/repository/conf/identity.xml* en la etiqueta que configura el servicio de SSO, denominada “SSOService”, se editan los valores de los atributos “EntityID” y “IdentityProviderURL” para que correspondan con los del servidor:

```
<SSOService>
                <EntityId>sso.unicauca.edu.co</EntityId>
<IdentityProviderURL>https://sso.unicauca.edu.co:9443/samlsso</IdentityP
roviderURL>
                ...
</SSOService>
```

Desde la interfaz de administración en el menú de listar los proveedores de identidad se encuentra la configuración del proveedor de identidad local, al cual le asignamos un identificador local de Realm y desplegamos los atributos de la configuración de autenticación de entrada en donde se modifica la configuración para el perfil WSSO de SAML2 asignando un identificador de identidad o "EntityID" que será el valor con que se identifique al proveedor de identidad frente a los proveedores de servicios SP.

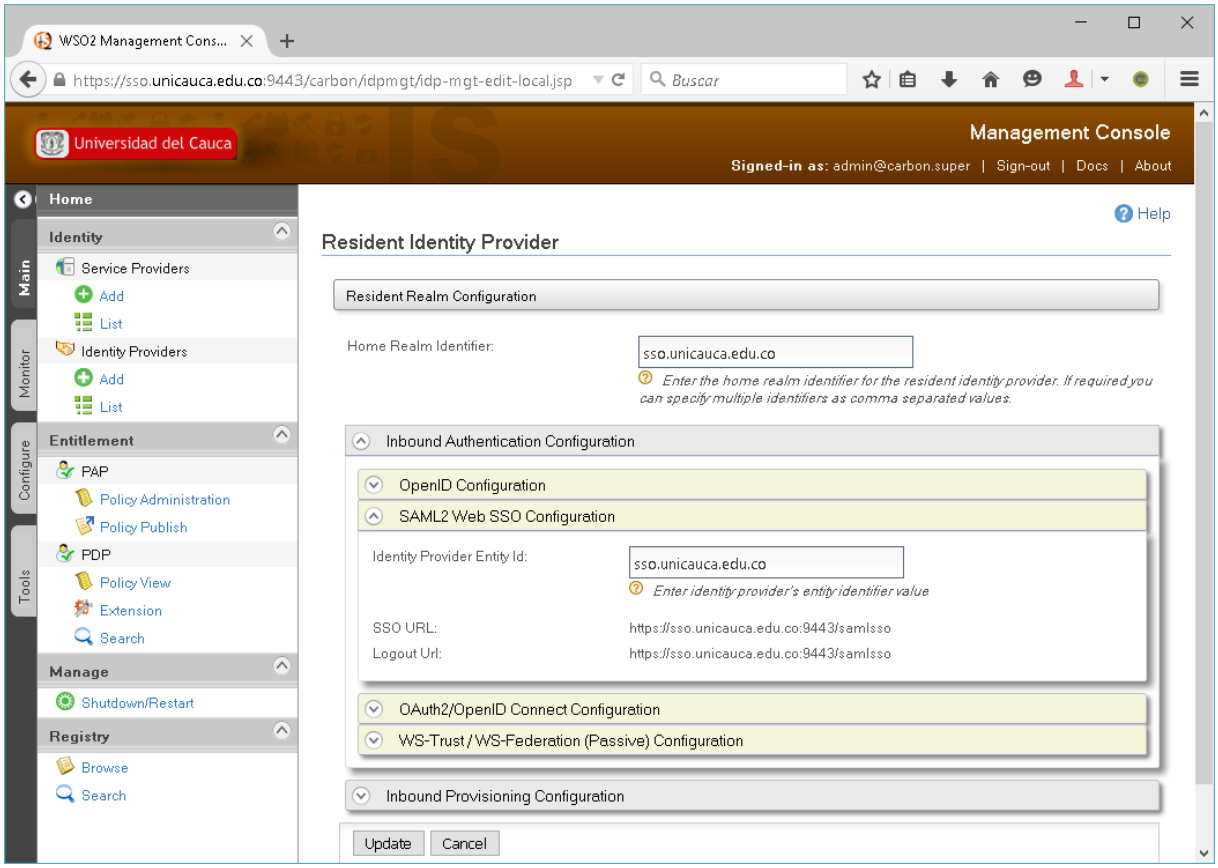

Ilustración 13. Configuración del proveedor de identidad

4.2. Plan de pruebas para la validación de la solución

Con la configuración realizada en el punto anterior ya se cuenta con un Proveedor de Identidad listo para recibir asertos y realizar afirmaciones de autenticación de los usuarios. Antes de poder llevar este servidor a producción se debe verificar que es capaz de autenticar usuarios.

La etapa de pruebas consiste en configurar aplicaciones de prueba como Proveedores de Servicios (SP) y conectarlos con el Proveedor de Identidad y corroborar que efectivamente se puede consumir el servicio después de haber ingresado las credenciales correctas de un usuario.

Los Proveedores de Servicios de prueba que se van a probar con el servidor son:

- **simpleSAMLphp:** esta herramienta que ya se nombró en el capítulo 2 también puede trabajar en el rol del Proveedor de Servicios (SP).
- **Travelocity.com:** es una aplicación de prueba escrita en JAVA que usa la librería OpenSAML, es distribuida por WSO2 para realizar las pruebas de los protocolos que incluye el servidor WSO2 IS.
- **Salesforce:** es una solución en la nube para ofrecer servicios de CRM pero incluye herramientas para funcionar como Proveedor de Identidad (IdP) y como Proveedor de Servicios (SP). Es una aplicación de pago pero ofrece un servicio de prueba de quince (15) días con el cual se realizarán las pruebas.

Para ello se destina un servidor Ubuntu 14.04.3 LTS con dominio http://ssopruebas.unicauca.edu.co y para Salesforce se crea una cuenta de prueba gratuita con el dominio https://unicauca.my.salesforce.com.

4.2.1. Instalación y configuración de Proveedores de Servicios (SP)

4.2.1.1. SimpleSAMLphp

Se debe agregar un proveedor de identidad al servidor simpleSAMLphp, esto se realiza mediante el archivo */var/simplesamlphp/simplesamlphp/config/authsources.php* en el cual se agrega la sección del Proveedor de Servicios para WSO2 IS, que incluye tres elementos: el primero es el valor del “entityID” que debe ser único para el Proveedor del Identidad (IdP), el valor de “idp” que es el cual va a buscar en la lista de proveedores configurados en ese servidor y por último el valor de “discoURL” que especifica la dirección de un servicio de descubrimiento, si existe.

```
'wso2-sp' => array(
    'saml:SP',
    'entityID' => 'simplesamlphp',
    'idp' => 'sso.unicauca.edu.co',
    'discoURL' => NULL,
    ),
```

Luego se debe crear el Proveedor de Identidad que se indicó en el archivo de configuración anterior, esto se realiza en el archivo */var/simplesamlphp/simplesamlphp/metadata/saml20-idp-remote.php* en el cual se agregan los datos del Proveedor de Identidad (IdP) de WSO2 IS incluyendo la huella digital del certificado del servidor WSO2 IS con el cual va firmada la respuesta del aserto SAML.

```
$metadata['sso.unicauca.edu.co'] = array(
  'name' => array(
  'en' =>  'WSO2 IS',
  'no' =>  'WSO2 IS',
  ),
  'description'   =>  'Proveedor de Identidad  WSO2 IS SAML2 IdP',
  'SingleSignOnService'  => 'https://sso.unicauca.edu.co:9443/samlsso',
  'SingleLogoutService'  => 'https://sso.unicauca.edu.co:9443/samlsso',
  'certFingerprint'      => '4dceb6a275871c15716d73f06827e86df425900d'
  );
```

Con esta configuración el Proveedor de Servicios con simpleSAMLphp ya está listo para realizar autenticaciones mediante el Proveedor de Identidad (IdP) WSO2 IS, por lo que el siguiente paso es crear el Proveedor de Servicios (SP) en el servidor WSO2 IS para que lo identifique y pueda responderle correctamente los asertos que este le envié, proceso que se explica más adelante.

4.2.1.2. Travelocity.com

Esta es la aplicación de prueba que provee WSO2 para probar el Inicio de Sesión Único (SSO) con SAML mediante un Proveedor de Servicios que utiliza OpenSAML para JAVA. Las instrucciones de instalación de la misma pueden ser consultadas en el anexo A.3.

La configuración de la aplicación se realiza mediante el archivo */usr/src/tomcat7/webapps/travelocity.com/WEB-INF/classes/travelocity.properties*, en el cual se configura el “Issuer”, la URL del ACS y la URL del Proveedor de Identidad (IdP).

```
SAML.IssuerID=travelocity.com

SAML.ConsumerUrl=http://ssopruebas.unicauca.edu.co:8080/travelocity.com/home.jsp

SAML.IdPUrl=https://sso.unicauca.edu.co:9443/samlsso
```

4.2.1.3. Salesforce

Se inicia sesión en el servicio web de Salesforce y se accede al menú de configuración, en la sección “Administrar” del menú izquierdo dentro del submenú “Controles de seguridad” se encuentra la “Configuración de inicio de sesión único”, como se observa en la . En esta interfaz se accede a la “Inicio de sesión único federado mediante SAML” y se activa la casilla de verificación del estándar SAML.

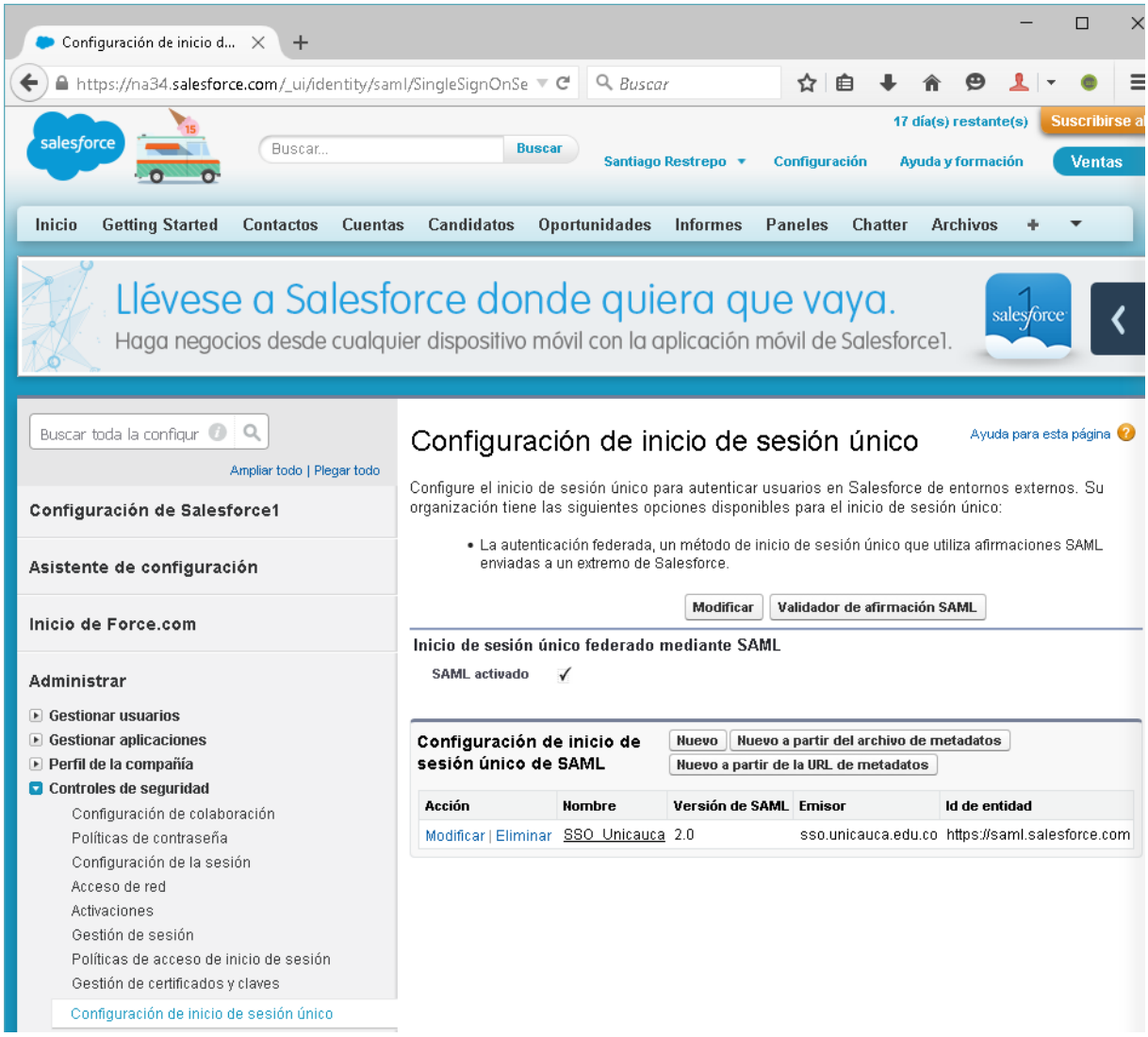

Ilustración 14. Configuración de inicio de sesión único de Salesforce

Se agrega un nuevo conector de inicio de sesión único con SAML mediante el formulario que se muestra en la Ilustración 15. Los campos que aparecen con una línea roja son campos obligatorios. En el campo “Emisor” se especifica la identidad del Proveedor de Identidad (IdP). El campo “Id de entidad” será el identificador único de este Proveedor de Servicios (SP). El formulario permite cargar el certificado público del Proveedor de Identidad (IdP) que sirve para verificar la firma de los asertos que llegan al Proveedor de Servicios (SP).

Al conector se le debe especificar como obtener el usuario que va a autenticar en Salesforce, en este caso se le indica que el sujeto del que trata el aserto es el usuario de Salesforce, también se puede obtener el usuario de los atributos que viajan dentro del aserto. Se indica el formato de binding que usa el Proveedor de Identidad para enviar el aserto, en este caso los envía como un parámetro oculto de un formulario en una petición HTTP-POST. Finalmente se especifica la URL del Proveedor de Identidad al cual Salesforce enviara las solicitudes de autenticación.

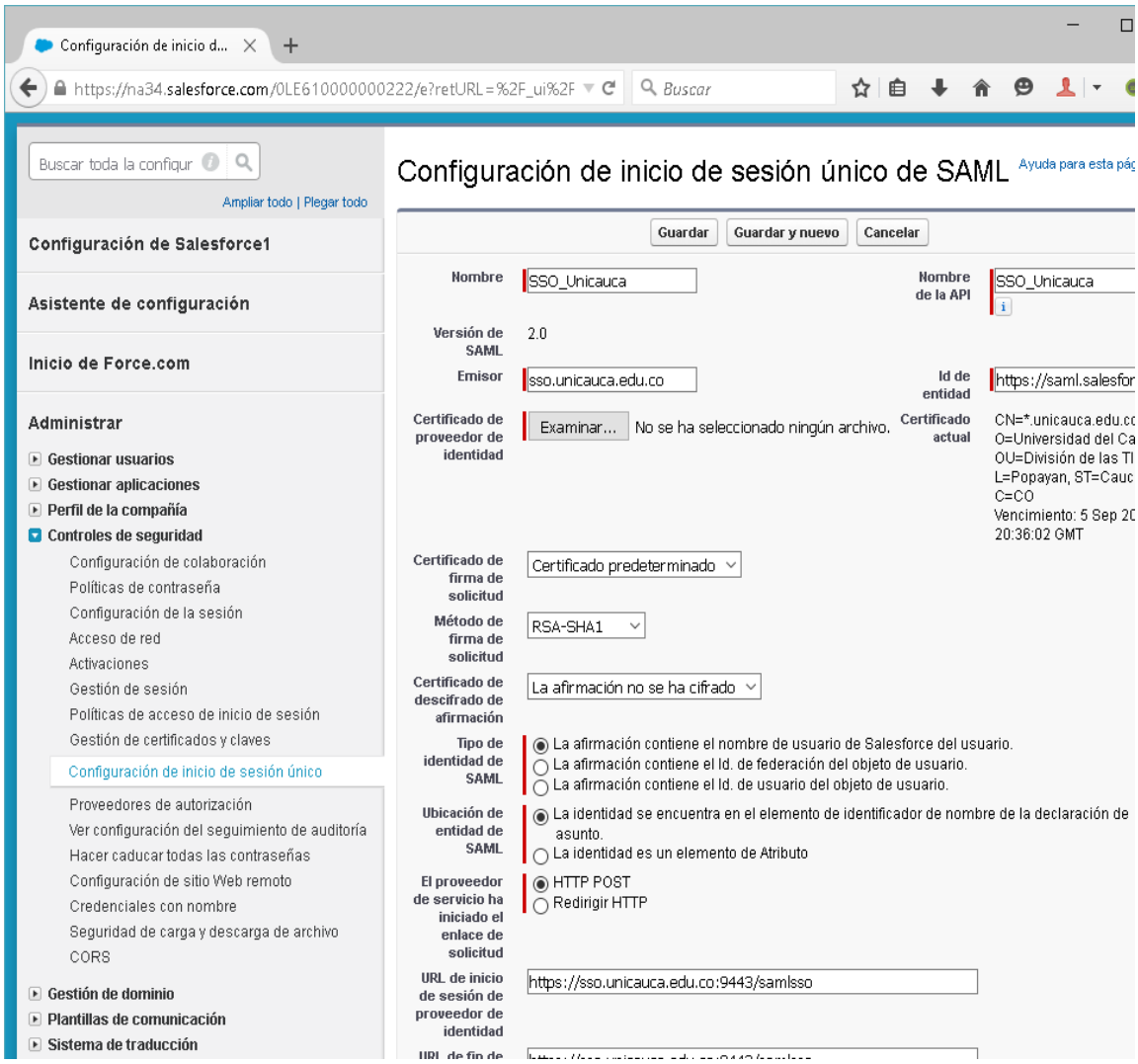

Ilustración 15. Configuración del Proveedor de Servicio (SP) Salesforce

4.2.1.4. Agregar un Proveedor de Servicios a WSO2 IS

Cuando se han configurado las herramientas para trabajar con el servidor WSO2 IS como el Proveedor de Identidad (IdP) se debe agregar en el este último para que sepa cómo debe responderle a cada aplicación.

Un Proveedor de Servicios se agrega mediante la interfaz de administración, aunque también se puede hacer por medio de archivos de configuración, a continuación se explica el proceso para el proveedor simpleSAMLphp pero no es diferente para los demás proveedores teniendo en cuenta los valores únicos para cada uno.

En la Ilustración 16 se presenta la interfaz de registro de un Proveedor de Servicios (SP) en donde se asigna un nombre y descripción que identifique unívocamente a cada proveedor.

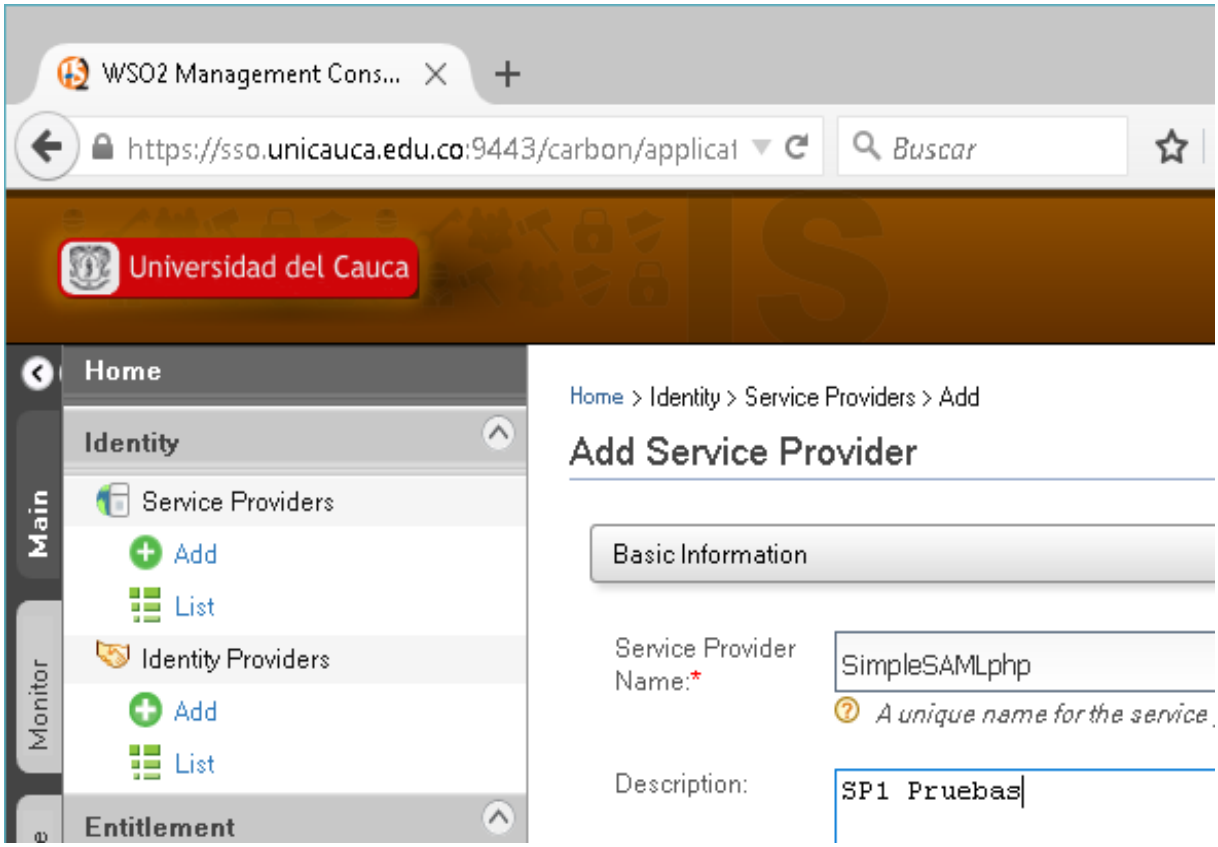

Ilustración 16. Registro de un Proveedor de Servicios

Posteriormente se procede a configurar los parámetros de autenticación de entrada para el protocolo SAML, la Ilustración 17 muestra el formulario de configuración completo del estándar.

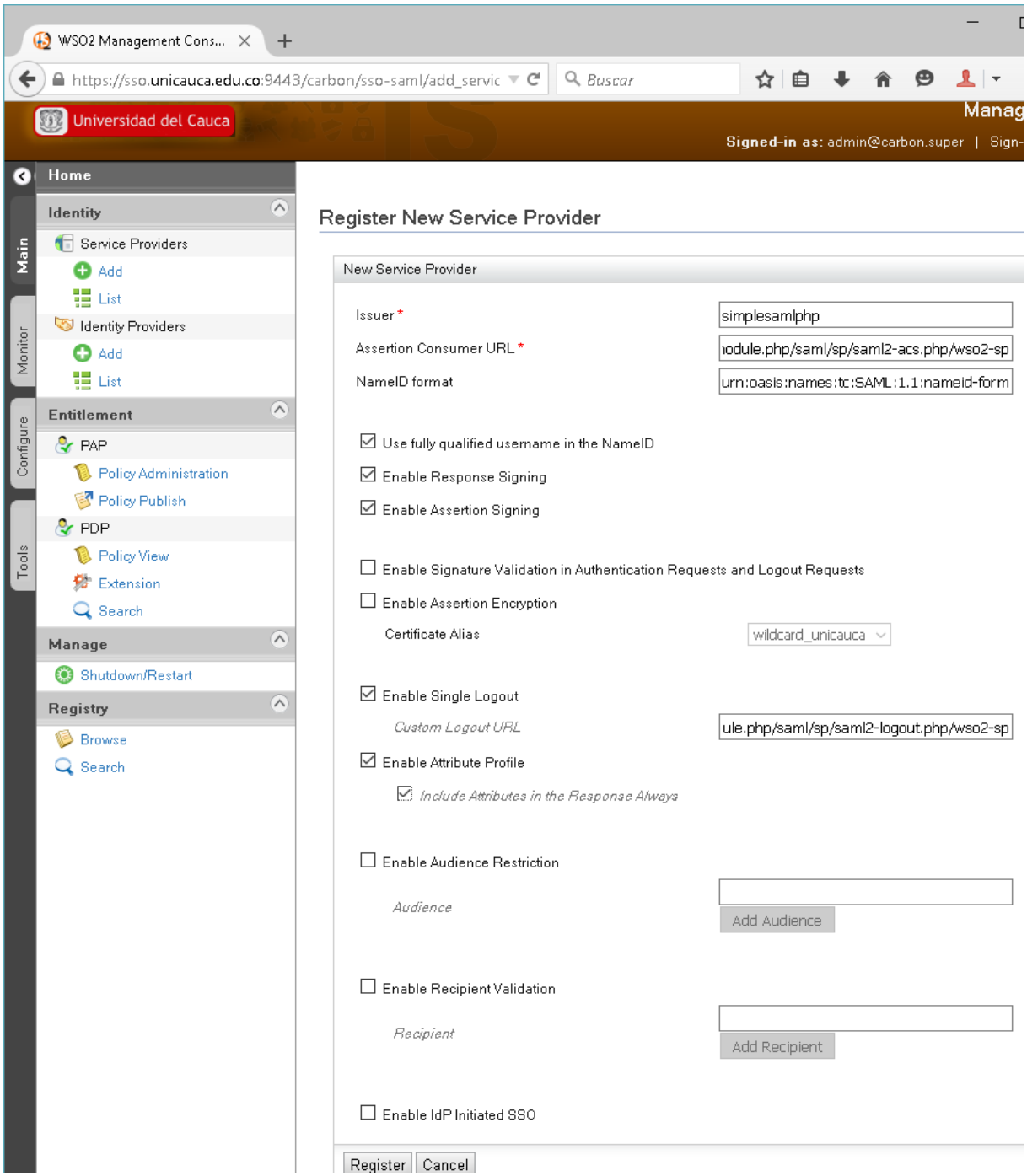

Ilustración 17. Configuración de autenticación de entrada SAML

Los elementos requeridos para la configuración de SAML se explican a continuación:

- **Issuer:** este elemento es obligatorio. Contiene el identificador único del Proveedor de Servicios (SP). Dentro del aserto SAML se identifica con la etiqueta <saml:Issuer>. Debe ser el mismo valor que viene en el aserto de solicitud de autenticación enviado por el proveedor.
- **Assertion Consumer URL:** es el último elemento obligatorio. Especifica la URL de consumo del servicio, es decir, a donde será redirigido el usuario después de una

autenticación exitosa. En el Proveedor de Servicios es la URL del Assertion Consumer Service (ACS).

- **NameID Format:** especifica el formato en el que se encuentra la información de un sujeto, generalmente y para las aplicaciones que se tratan en este piloto experimental se usa el tipo "urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress". Otros formatos pueden ser:
  - urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos
  - urn:oasis:names:tc:SAML:2.0:nameid-format:entity
  - urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified
  - urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
  - urn:oasis:names:tc:SAML:2.0:nameid-format:transient
  - urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
- **Use fully qualified username in the NameID:** especifica si se desea que a la información de usuario se le agregue el dominio.
- **Enable Response Signing:** especifica si se desea firmar las respuestas SAML después del proceso de autenticación.
- **Enable Assertion Signing:** especifica si se desea firmar los asertos SAML que se reenvían después de la autenticación. Generalmente las entidades SAML esperan que los asertos vengan firmados por el Proveedor de Identidad (IdP)
- **Enable Signature Validation in Authentication Requests and Logout Requests:** especifica si se desea que el Proveedor de Identidad valide la firma del aserto de solicitud de autenticación y del aserto de solicitud de cierre de sesión que provienen del Proveedor de Servicios (SP).
- **Assertion Encryption:** posibilita el cifrado del aserto SAML.
- **Certificate Alias:** permite escoger el certificado del Proveedor de Servicios (SP), sirve para validar la firma de las solicitudes SAML que llegan al Proveedor de Identidad (IdP) y para generar el cifrado.
- **Enable Single Logout:** permite activar el perfil de Single Logout, significa que todas las sesiones se terminan una vez el usuario cierra sesión en un servidor, para ello el Proveedor de Identidad (IdP) envía un aserto de solicitud de cierre de sesión a todos los Proveedores de Servicios (SP).
- **Custom Logout URL:** permite redirigir al usuario a una URL de cierre de sesión del Proveedor de Servicios (SP), si cuenta con una, en caso de que no el Proveedor de Identidad (IdP) usa la URL del ACS.

- **Enable Attribute Profile:** permite habilitar el envío de atributos de usuario obtenidos mediante las peticiones o “Claims” en el aserto SAML dentro la declaración de atributos.
- **Include Attributes in the Response Always:** especifica si se desean enviar siempre los atributos en el aserto SAML.
- **Enable Audience Restriction:** especifica si se desea restringir la audiencia que para la cual está dirigido el aserto SAML.
- **Enable Recipient Validation:** especifica si se requiere que se envié el ACS registrado en el Proveedor de Identidad (IdP) en cada aserto dirigido al Proveedor de Servicios (SP).
- **Enable IdP Initiated SSO:** permite habilitar el escenario de Inicio de Sesión Único (SSO) iniciado por el Proveedor de Identidad (IdP).

Para el ejemplo del Proveedor de Servicios (SP) simpleSAMLphp se tienen los siguientes valores:

```
Issuer: simplesamlphp
Assertion Consumer URL:
http://ssopruebas.unicauca.edu.co/simplesaml/module.php/saml/sp/saml2-acs.php/wso2-sp
NameID format: urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress
Logout URL:
http://ssopruebas.unicauca.edu.co/simplesaml/module.php/saml/sp/saml2-logout.php/wso2-sp
```

Cuando se ha configurado la autenticación de entrada se requiere que se especifiquen las peticiones o “Claims” que se van a obtener del almacén de datos para el sujeto que se está autenticando, en la Ilustración 18 se puede observar cómo se configuran estas peticiones.

Estas peticiones se definen en un “Dialecto” que es un conjunto de peticiones preestablecido y el usuario puede crear los dialectos que requiera, en este piloto experimental basta con trabajar con el dialecto por defecto de WSO2 IS.

En el formulario se adicionan las peticiones que se deseen que se retornen al Proveedor de Servicios (SP) y que petición identificara al sujeto que se está autenticando.

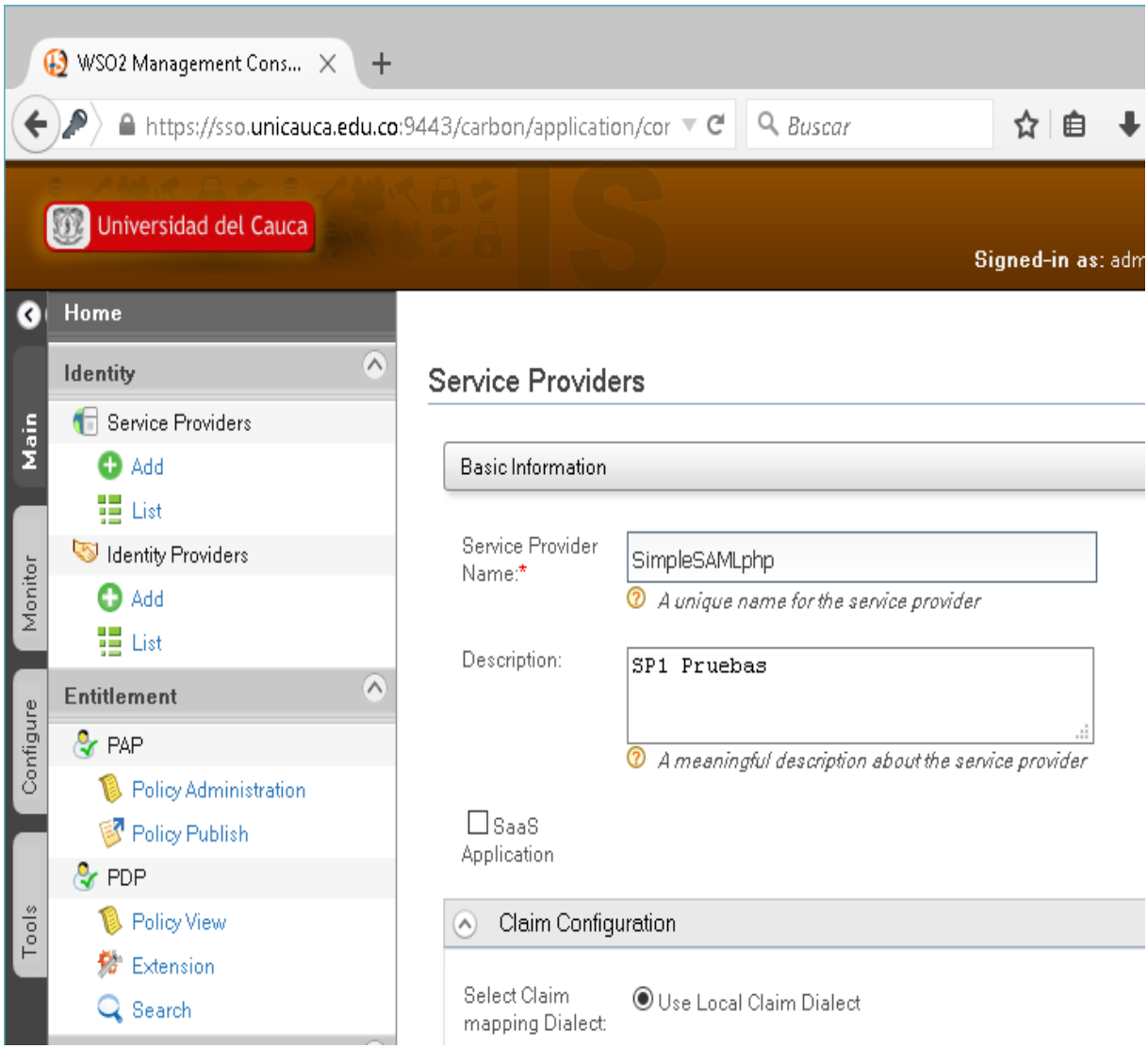

Ilustración 18. Configuración de Peticiones (Claims) para simpleSAMLphp

Con esta configuración se tiene correctamente configurado el Proveedor de Servicios (SP) de una forma básica y puede empezar a intercambiar asertos SAML con el Proveedor de Identidad (IdP).

Se debe seguir el mismo procedimiento para agregar los Proveedores de Servicios (SP) de Travelocity.com y Salesforce al servidor WSO2 IS.

La Ilustración 19 presenta la configuración de peticiones del Proveedor de Servicios (SP) Travelocity.com.

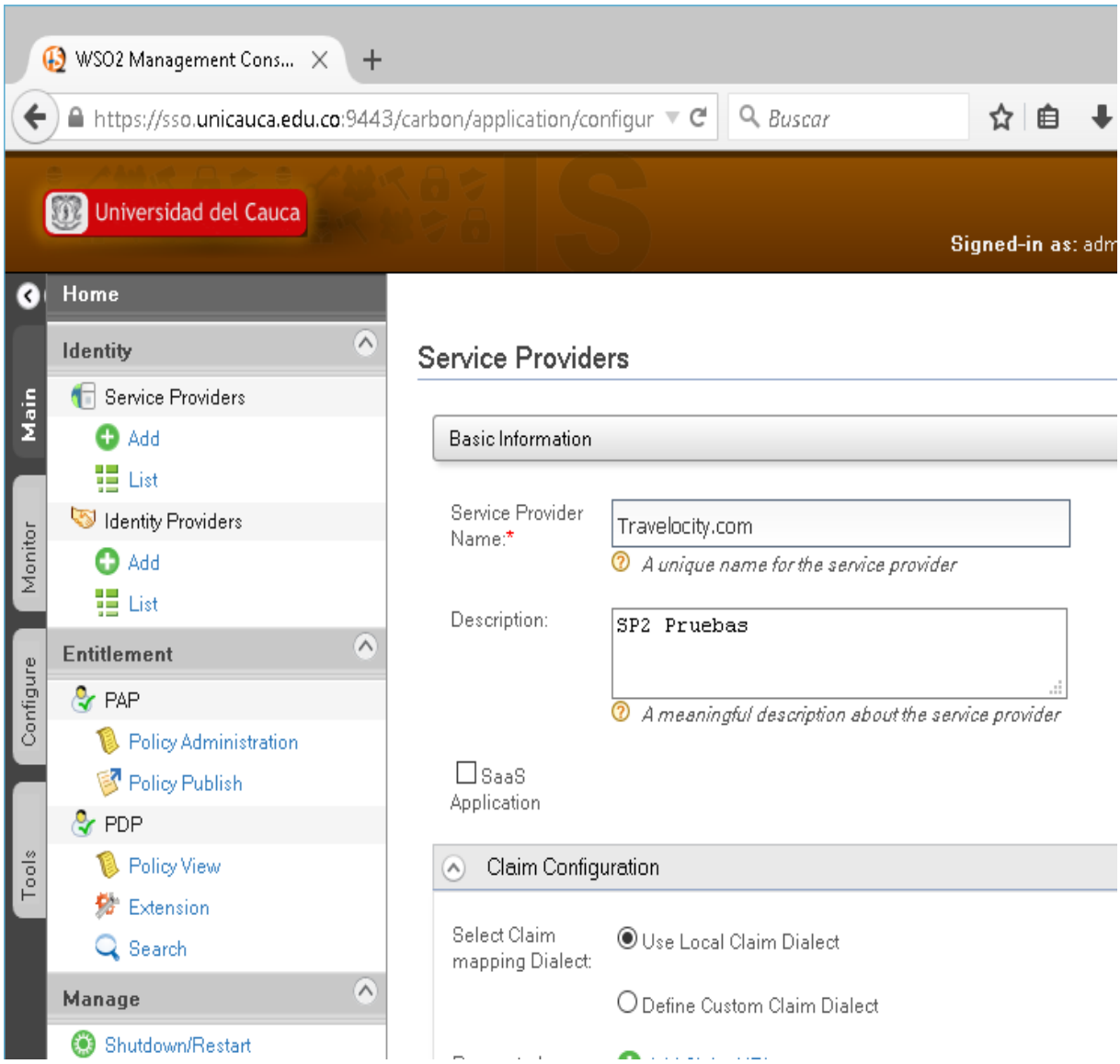

Ilustración 19. Configuración de Peticiones (Claims) para Travelocity.com

Así mismo en la Ilustración 20 se puede observar la configuración de autenticación de entrada SAML para Travelocity.com.

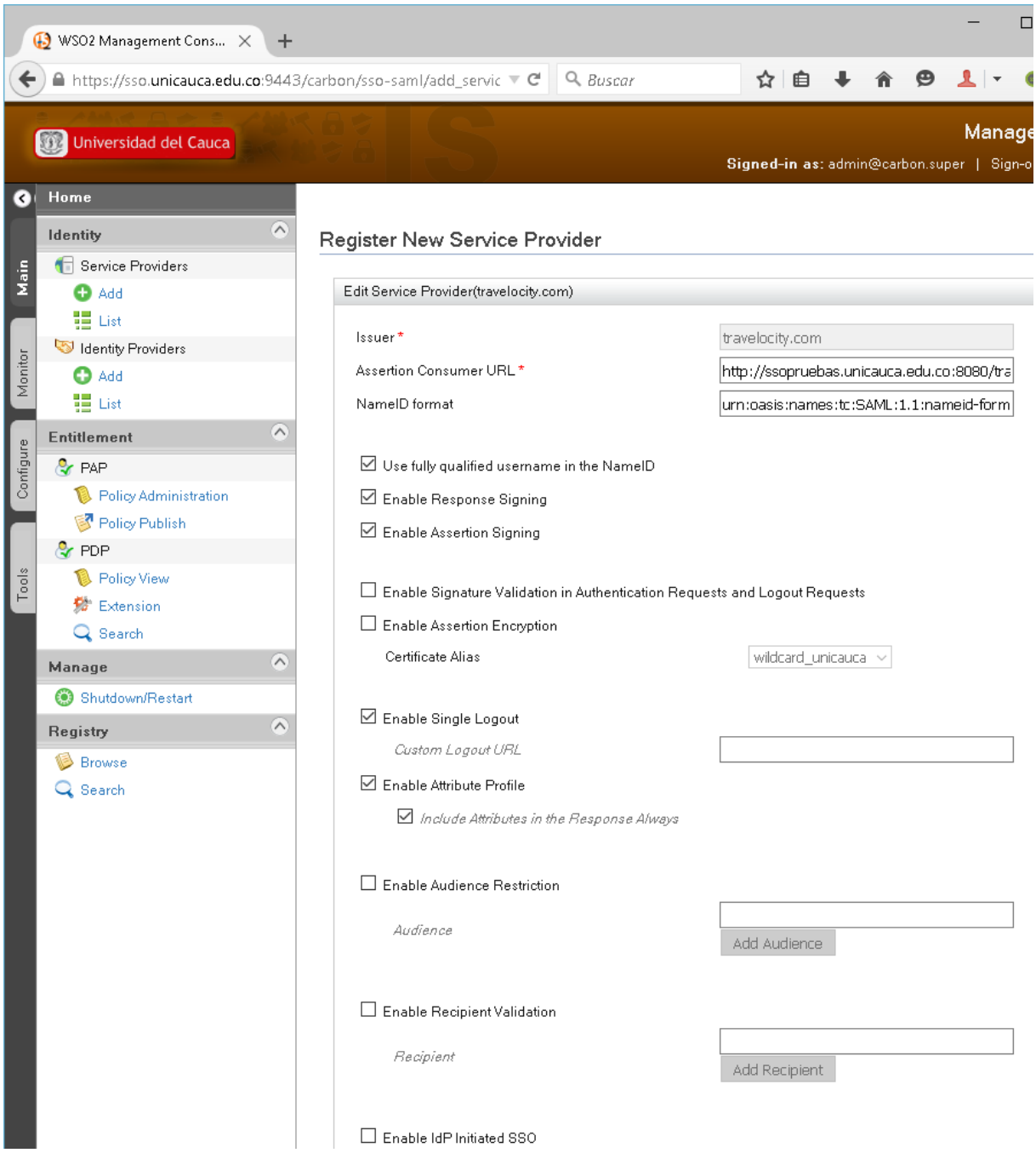

Ilustración 20. Configuración de autenticación SAML para Travelocity.com

Finalmente en la Ilustración 21 y en la Ilustración 22 se observa la configuración del Proveedor de Servicio (SP) Salesforce.

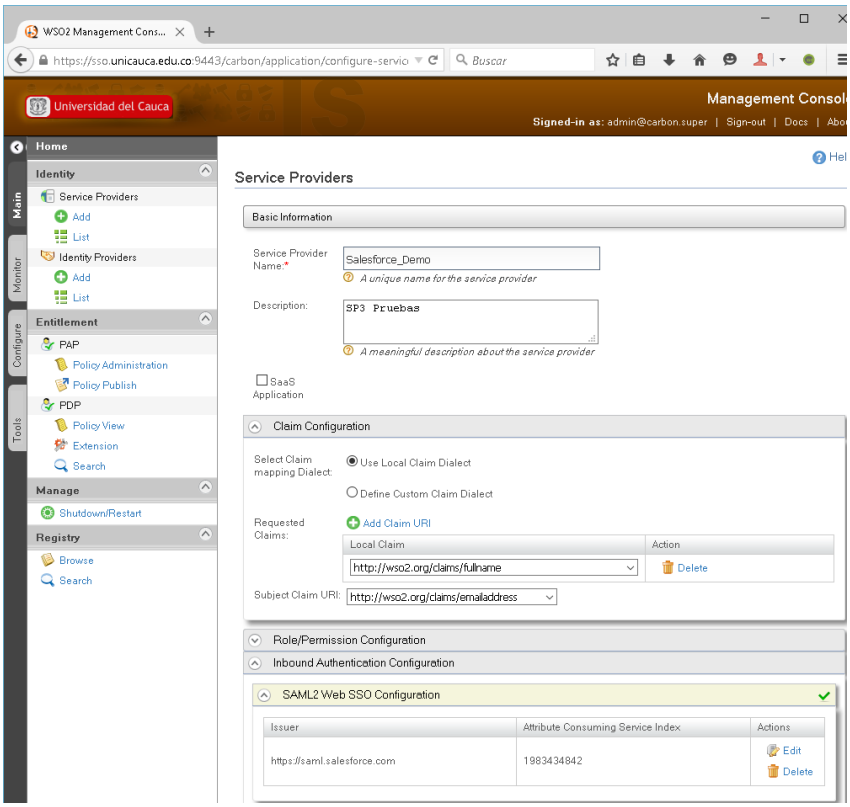

Ilustración 21. Configuración de Peticiones (Claims) para Salesforce

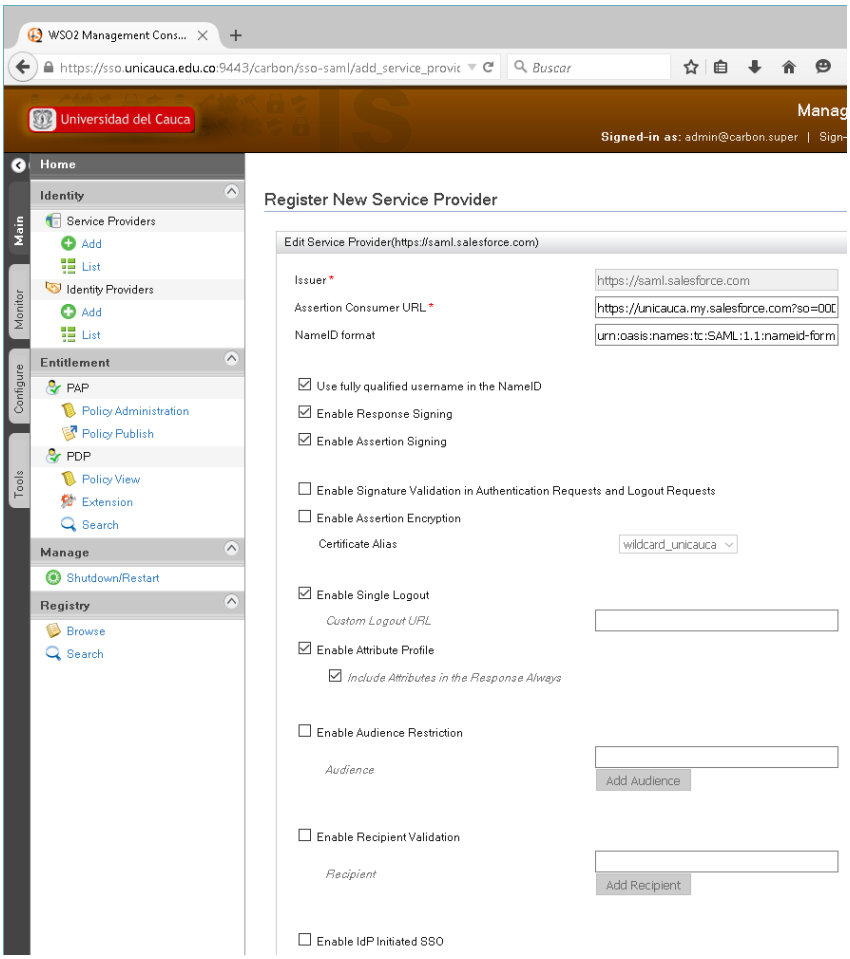

Ilustración 22. Configuración de autenticación SAML para Salesforce

4.2.2. Pruebas del entorno de Inicio de Sesión Único (SSO)

Se considera la implementación de un entorno de pruebas que facilite la validación de la herramienta WSO2 Identity Server, en este entorno se facilita la configuración del sistema ya que los Proveedores de Servicio (SP) involucrados en este punto son instalados y configurados por el autor por lo que se tiene la facilidad de manipular parámetros como el "Issuer", "ACS" y el "Subject" con el que van a operar dichos aplicativos.

A continuación se presenta la prueba del entorno de Inicio de Sesión Único (SSO) de pruebas, para mantener un orden durante la prueba el Proveedor de Servicios 1 (SP1) será simpleSAMLphp, el Proveedor de Servicios 2 (SP2) será Travelocity.com y finalmente el Proveedor de Servicios 3 será Salesforce.

Básicamente se deben realizar dos pruebas para verificar el correcto funcionamiento del sistema de Inicio de Sesión Único (SSO) implementado en este piloto experimental: una prueba de inicio de sesión y una prueba de cierre de sesión.

En la primera parte de la prueba se necesita verificar que cuando el usuario accede al Proveedor de Servicios 1 (SP1) y no tenga una sesión iniciada, el Proveedor de Identidad (IdP) lo redirige a la página de autenticación, en esta página se debe verificar que se autentique con credenciales correctas y una vez sea autenticado satisfactoriamente sea redirigido al Proveedor de Servicios 1 (SP1) y se le permita el acceso. Ya con una sesión iniciada se debe acceder al Proveedor de Servicios 2 (SP2) y se debe permitir el ingreso sin solicitar de nuevo las credenciales ya que el usuario cuenta con una sesión iniciada previamente en el Proveedor de Identidad (IdP), lo mismo para al Proveedor de Servicios 3 (SP3).

La Ilustración 23 presenta el diagrama de flujo de la primera parte de la prueba que se divide en tres fases, dependiendo del Proveedor de Servicios (SP) que se trate en cada momento de la prueba. La línea de color naranja representa el flujo requerido para que la prueba se considere exitosa.

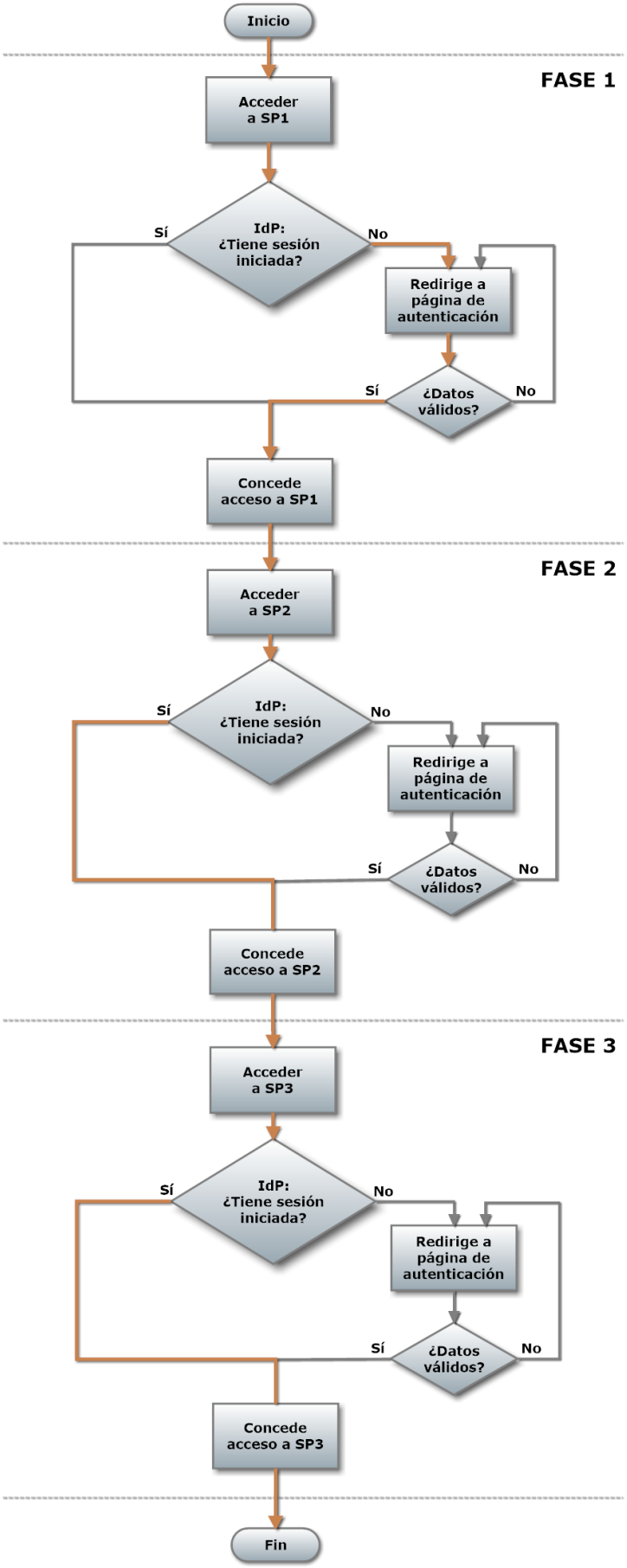

Ilustración 23. Diagrama de flujo de prueba de inicio de sesión

La segunda parte de la prueba consiste en acceder al Proveedor de Servicios 1 (SP1) y cerrar la sesión, cuando el Proveedor de Identidad (IdP) redirija a la página principal la sesión habrá finalizado y se procede a acceder al Proveedor de Servicios 2 (SP2) y deberá mostrar la página de autenticación del Proveedor de Identidad (IdP), ya que si el cierre de sesión se ejecutó de manera correcta, todas las sesiones de los Proveedores de Servicio (SP) deben de haber sido invalidadas, lo mismo debe suceder cuanto se intente acceder de nuevo al Proveedor de Servicios 3 (SP3).

La Ilustración 24 presenta el diagrama de flujo de la segunda parte de la prueba que también está dividida en tres fases, dependiendo del Proveedor de Servicios (SP) que se trate en cada momento de la prueba. La línea de color naranja representa el flujo que se desea siga la prueba para que se considere exitosa.

Es importante recalcar que la totalidad de la prueba se debe realizar en la misma sesión del navegador, es decir, no se puede cerrar el navegador ya que esto borra por completo las cookies de sesión y cuando se vuelva a iniciar el navegador será como si no hubiera iniciado sesión.

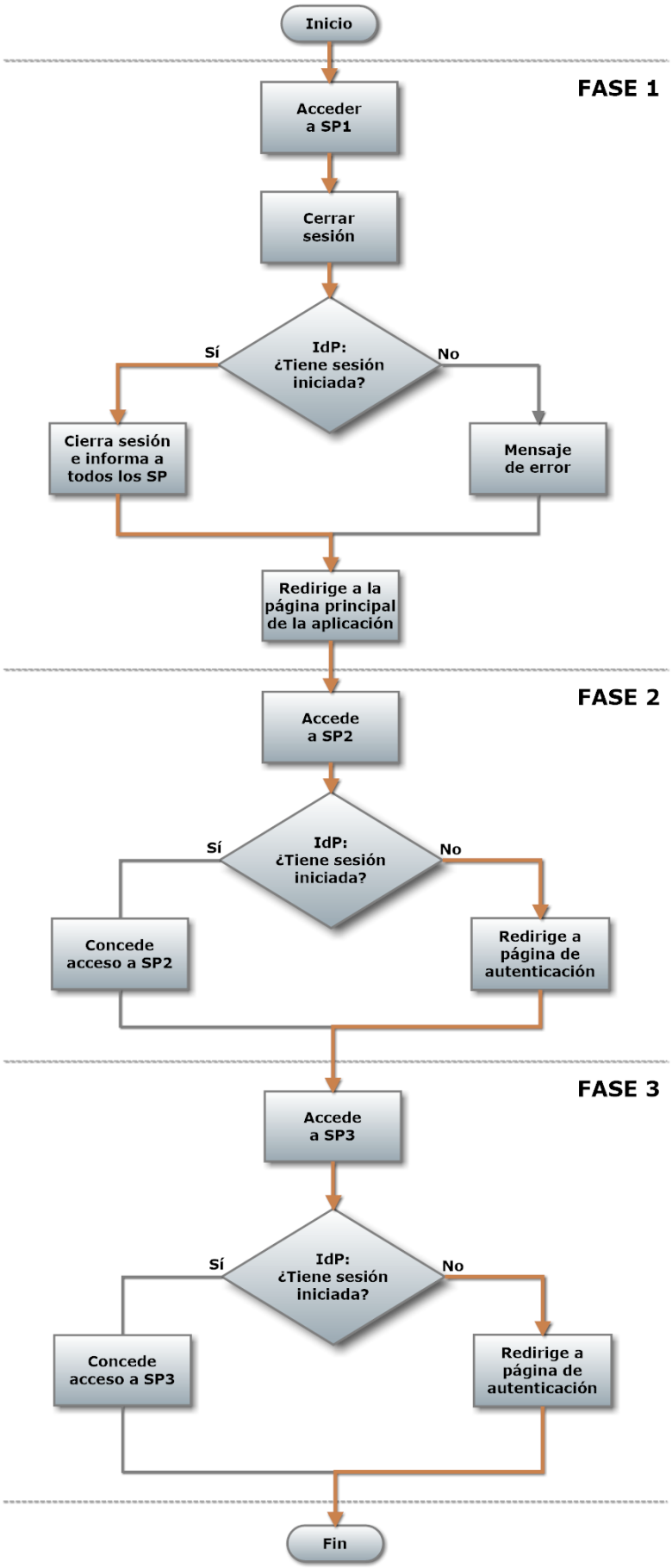

Ilustración 24. Diagrama de flujo de prueba de cierre de sesión

Parte 1:

Fase 1:

La prueba inicia cuando el usuario accede al Proveedor de Servicios 1 (SP1) que en este caso es simpleSAMLphp mediante la URL http://ssopruebas.unicauca.edu.co/simplesaml, aquí se dirige a la sección de pruebas de autenticación y accede al Proveedor de Servicios (SP) que se configuro previamente denominado “wso2-sp”. En la Ilustración 25 se observa la interfaz y mediante la extensión del navegador se puede ver que hasta este punto la única cookie de sesión es de PHP.

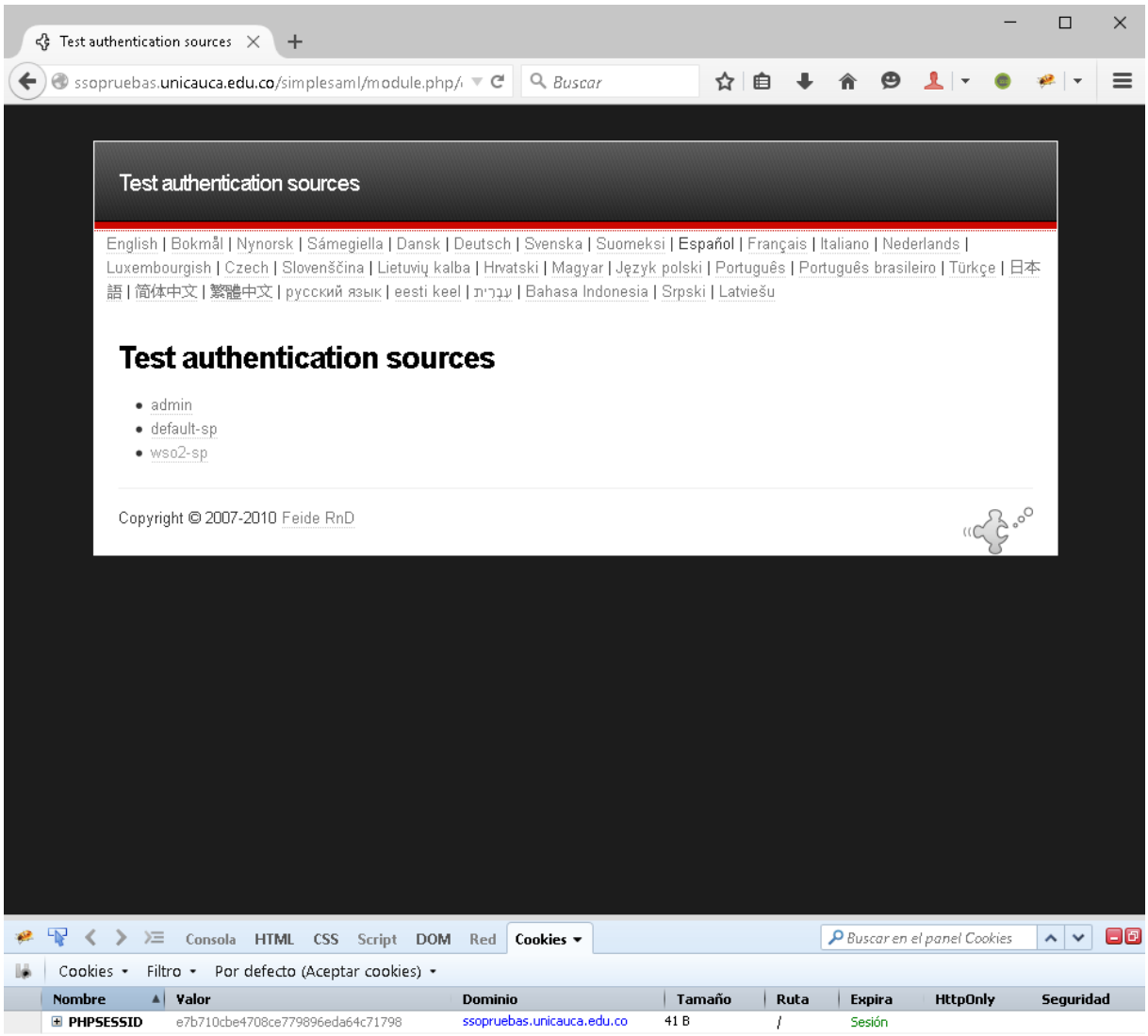

Ilustración 25. Interfaz de pruebas de autenticación de simpleSAMLphp

Al momento de acceder al Proveedor de Servicios (SP) “wso2-sp” este redirige la petición al Proveedor de Identidad (IdP) mediante un aserto SAML de solicitud de autenticación, este recibe el aserto y detecta que no existe una sesión activa de dicho Proveedor de Servicios (SP) por lo que presenta la interfaz de autenticación, como se observa en la Ilustración 25.

Ilustración 26. Interfaz de autenticación del Proveedor de Identidad (IdP)

El primer mensaje SAML que se intercambia entre el Proveedor de Identidad (IdP) y el Proveedor de Servicios 1 (SP1) es la solicitud de autenticación y tiene la siguiente estructura:

```
<samlp:AuthnRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
                    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
                    ID="_55720a76609b4bbcb8b2461d0d285f7c995c3f2e83"
                    Version="2.0"
                    IssueInstant="2015-09-06T21:41:41Z"
                    Destination="https://sso.unicauca.edu.co:9443/samlsso"
AssertionConsumerServiceURL="http://ssopruebas.unicauca.edu.co/simplesaml/module
.php/saml/sp/saml2-acs.php/wso2-sp"
             ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
>
       <saml:Issuer>simplesamlphp</saml:Issuer>
       <samlp:NameIDPolicy Format="urn:oasis:names:tc:SAML:2.0:nameid-
format:transient"
       AllowCreate="true"/>
</samlp:AuthnRequest>
```

Cuando el usuario es autenticado correctamente por el Proveedor de Identidad (IdP), este le responde con el segundo mensaje SAML al Proveedor de Servicios (SP) y es un aserto de respuesta con la siguiente estructura:

```
<saml2p:Response xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
Destination="http://ssopruebas.unicauca.edu.co/simplesaml/module.php/saml/sp/saml2-
acs.php/wso2-sp" ID="eblfpknipibfiihcikacocanofpapgkmidbglbjg"
InResponseTo="_55720a76609b4bbcb8b2461d0d285f7c995c3f2e83" IssueInstant="2015-09-
06T21:42:39.617Z" Version="2.0">
  <saml2:Issuer xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
Format="urn:oasis:names:tc:SAML:2.0:nameid-
format:entity">sso.unicauca.edu.co</saml2:Issuer>
  <saml2p:Status>
    <saml2p:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
  </saml2p:Status>
  <saml2:Assertion xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
xmlns:xs="http://www.w3.org/2001/XMLSchema" ID="mmpknecnaenmhjhcnhnlfccemjgpakljlmocgojd"
IssueInstant="2015-09-06T21:42:39.617Z" Version="2.0">
    <saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-
format:entity">sso.unicauca.edu.co</saml2:Issuer>
    <saml2:Subject>
      <saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:emailAddress">sanrestrepo@unicauca.edu.co</saml2:NameID>
      <saml2:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
        <saml2:SubjectConfirmationData
InResponseTo="_55720a76609b4bbcb8b2461d0d285f7c995c3f2e83" NotOnOrAfter="2015-09-
06T21:47:39.617Z"
Recipient="http://ssopruebas.unicauca.edu.co/simplesaml/module.php/saml/sp/saml2-
acs.php/wso2-sp"/>
      </saml2:SubjectConfirmation>
    </saml2:Subject>
    <saml2:Conditions NotBefore="2015-09-06T21:42:39.617Z" NotOnOrAfter="2015-09-
06T21:47:39.617Z">
      <saml2:AudienceRestriction>
        <saml2:Audience>simplesamlphp</saml2:Audience>
      </saml2:AudienceRestriction>
    </saml2:Conditions>
    <saml2:AuthnStatement AuthnInstant="2015-09-06T21:42:39.618Z" SessionIndex="5c7d09f3-
0499-4314-891d-f2bce121357e">
      <saml2:AuthnContext>
<saml2:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:Password</saml2:AuthnC
ontextClassRef>
      </saml2:AuthnContext>
    </saml2:AuthnStatement>
    <saml2:AttributeStatement>
      <saml2:Attribute Name="http://wso2.org/claims/emailaddress"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
        <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">sanrestrepo@unicauca.edu.co</saml2:AttributeValue>
      </saml2:Attribute>
      <saml2:Attribute Name="http://wso2.org/claims/givenname"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
        <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">Santiago</saml2:AttributeValue>
      </saml2:Attribute>
    </saml2:AttributeStatement>
  </saml2:Assertion>
</saml2p:Response>
```

Cuando el Proveedor de Servicios (SP) recibe la respuesta con la afirmación de identidad sobre el sujeto redirige el navegador del usuario a la URL del “Assertion Consumer Service (ACS)”, como se observa en la Ilustración 27. En este punto en la cookie de sesión se crea un token SAML que identifica la sesión única,

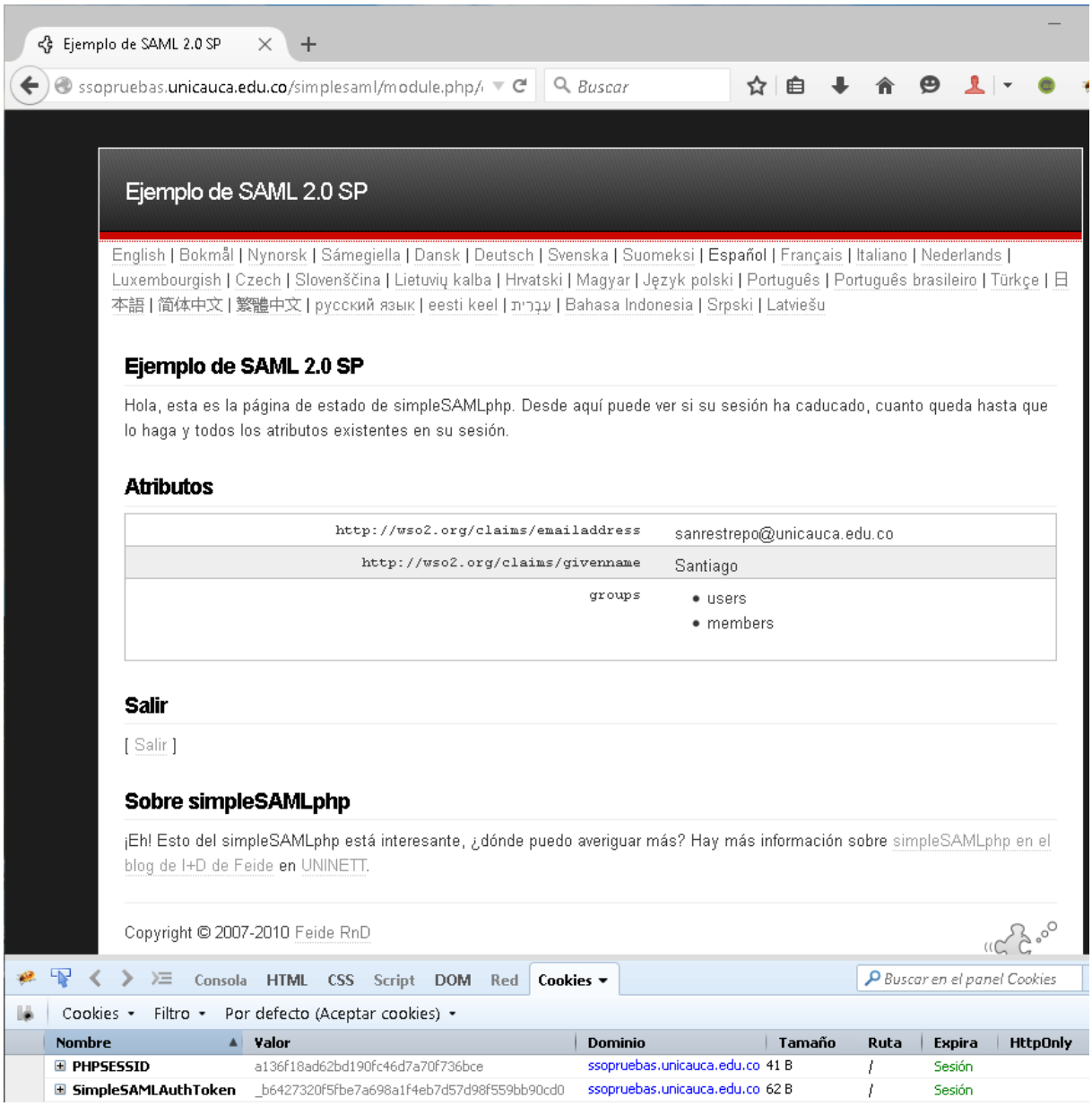

Ilustración 27. Interfaz de ACS para simpleSAMLphp

Fase 2:

Se abre una nueva pestaña en el navegador y se accede al Proveedor de Servicios 2 (SP2) mediante la URL http://ssopruebas.unicauca.edu.co:8080/travelocity.com, el aplicativo al envía un aserto de autenticación al Proveedor de Identidad (IdP), este último al verificar que ya existe una sesión activa emite de inmediato un aserto SAML de respuesta con los datos del sujeto que es el dueño de la sesión activa y el navegador se redirige a la URL de ACS y se le concede el acceso al usuario sin requerirle que ingrese nuevamente sus credenciales, como se observa en la Ilustración 28. En este punto se envían dos asertos SAML con estructura similar a los que se enviaron en la Fase 1 por lo que no se presentan de nuevo estos elementos.

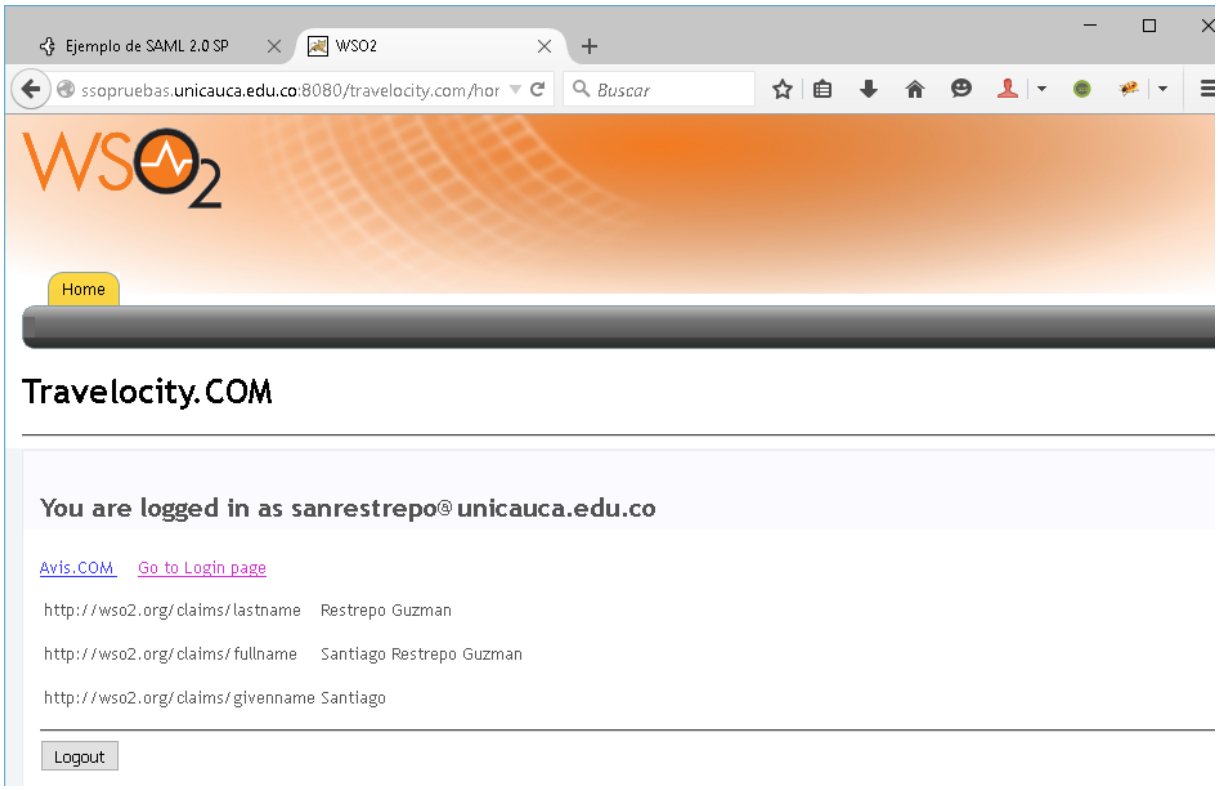

Ilustración 28. Interfaz de ACS para Travelocity.com

Fase 3:

Nuevamente en una pestañase accede al Proveedor de Servicios 3 (SP3) mediante la URL https://unicauca.my.salesforce.com en donde se repite el proceso de la fase 2, donde automáticamente se concede el acceso al usuario a la URL de ACS, como se observa en la Ilustración 29.

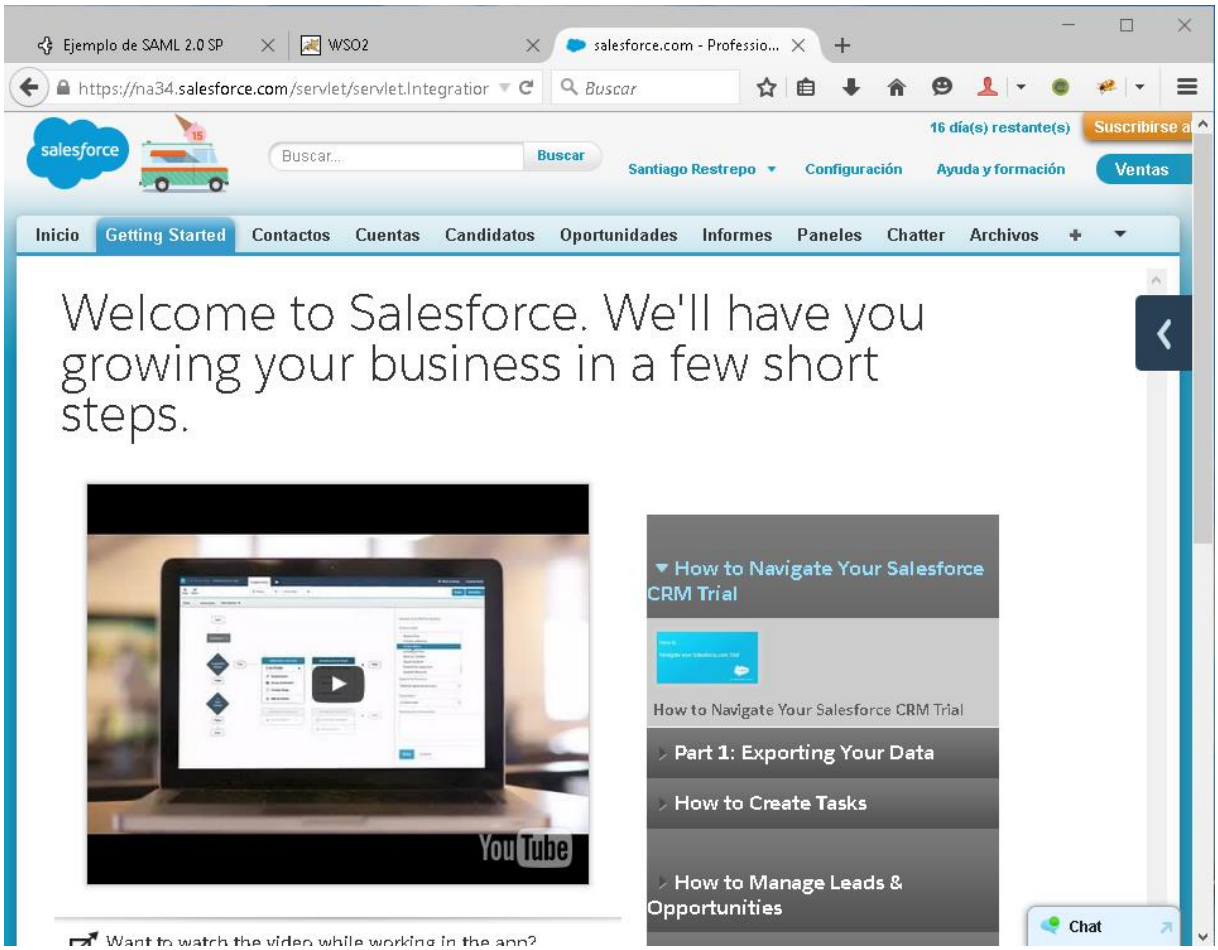

Ilustración 29. Interfaz de ACS para Salesforce

Parte 2:

Fase 1:

Se debe iniciar el cierre de sesión desde el Proveedor de Servicios 1 (SP1), este envía un aserto SAML de cierre de sesión con la siguiente estructura:

```
<samlp:LogoutRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
                     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
                     ID="_00105aabe133eb7380da847847329715fd012c25c0"
                     Version="2.0"
                     IssueInstant="2015-09-06T21:47:00Z"
                     Destination="https://sso.unicauca.edu.co:9443/samlsso"
                     >
        <saml:Issuer>simplesamlphp</saml:Issuer>
        <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:emailAddress">sanrestrepo@unicauca.edu.co</saml:NameID>
        <samlp:SessionIndex>5c7d09f3-0499-4314-891d-
f2bce121357e</samlp:SessionIndex>
</samlp:LogoutRequest>
```

El Proveedor de Identidad (IdP) lo recibe, valida que el identificador de sesión corresponda a una sesión activa y en caso de afirmativo procede a eliminar la sesión de todos los

Proveedores de Servicios (SP) asociados a ese índice, después de este proceso emite un aserto de respuesta de cierre de sesión con la siguiente estructura:

```
<saml2p:LogoutResponse xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
ID="hecjganmnnigahjmlilphndghhlmkgmdeldengnp"
InResponseTo="_00105aabe133eb7380da847847329715fd012c25c0" IssueInstant="2015-
09-06T21:47:01.987Z" Version="2.0">
  <saml2:Issuer xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
Format="urn:oasis:names:tc:SAML:2.0:nameid-
format:entity">sso.unicauca.edu.co</saml2:Issuer>
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  …
  </ds:Signature>
  <saml2p:Status>
    <saml2p:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
  </saml2p:Status>
</saml2p:LogoutResponse>
```

El Proveedor de Servicios (SP) recibe el aserto de respuesta que le indica que la sesión fue cerrada correctamente y redirige al navegador a la página de cierre de sesión exitoso como se observa en la Ilustración 30.

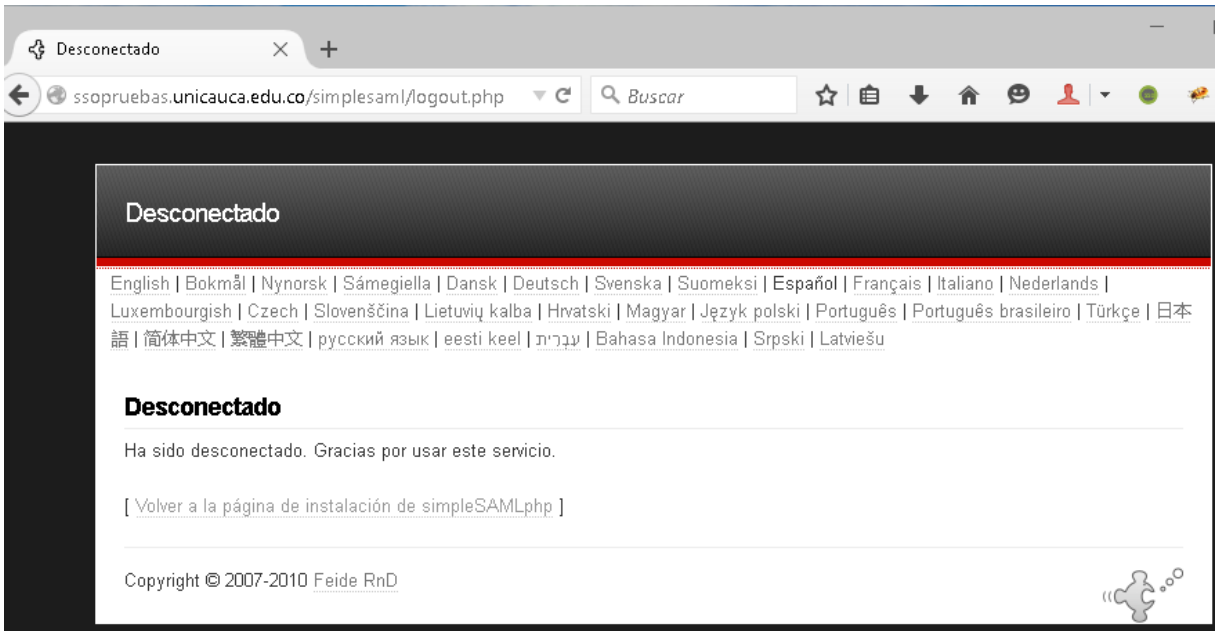

Ilustración 30. Interfaz de cierres de sesión de simpleSAMLphp

Fase 2:

Ahora se debe intentar acceder en una nueva ventana del navegador al Proveedor de Servicios 2 (SP2) y al no existir ya una sesión iniciada para el navegador el aplicativo debe presentar la interfaz principal de Travelocity.com para que el usuario inicie un nuevo proceso de autenticación, como se observa en la .

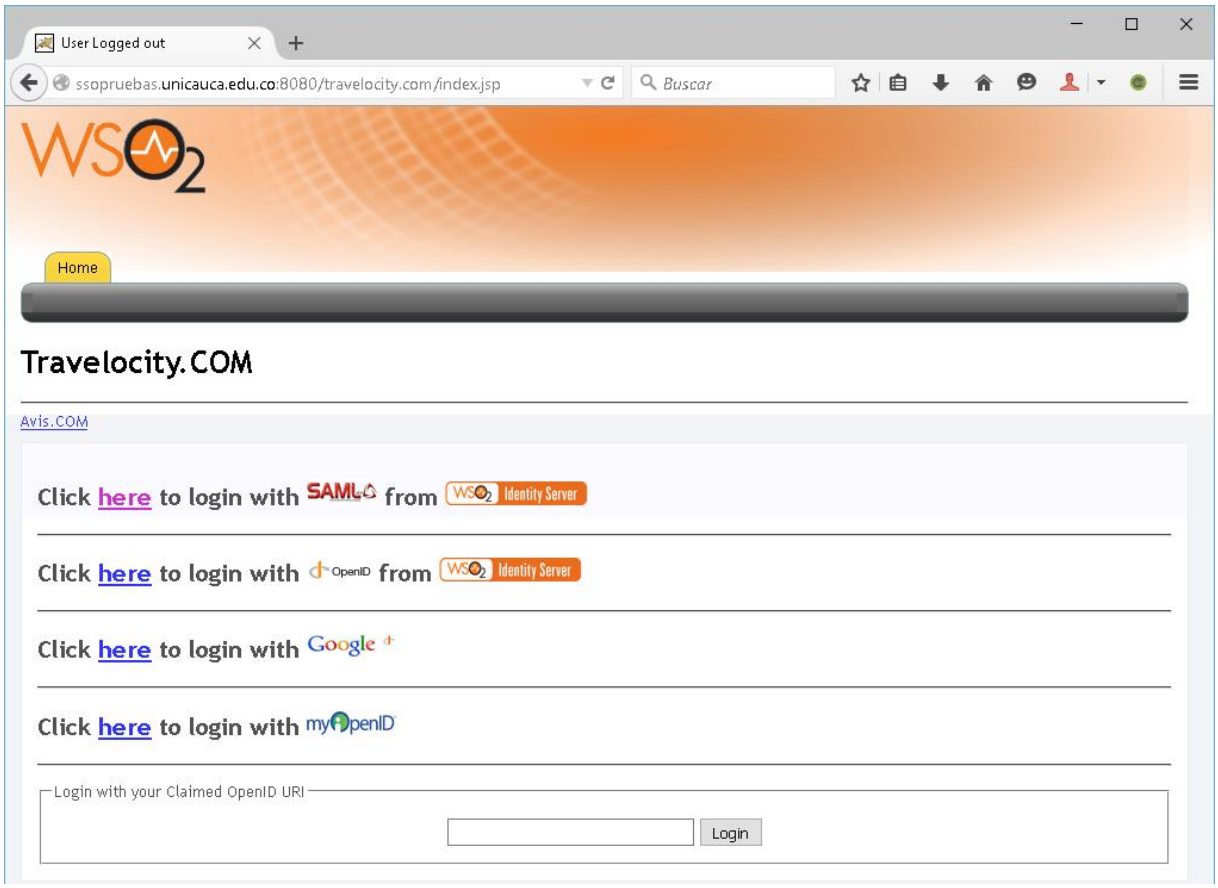

Ilustración 31. Interfaz de selección de estándar de Travelocity.com

Fase 3:

Finalmente en una nueva ventana del navegador se accede al Proveedor de Servicios 3 (SP3) y este debe redirigir al Proveedor de Identidad (IdP) mediante un aserto de solicitud de autenticación, este último al validar que no hay sesiones activas para el navegador redirige el navegador a la interfaz de autenticación, confirmando que la sesión se cerró correctamente para todos los Proveedores de Servicios (SP).

4.3. Implementación en producción

En el entorno en producción los Proveedores de Servicios (SP) son aplicaciones web que ya están implementadas y sobre las cuales no se tiene ningún control, por lo que se debe indagar de manera previa con cada proveedor los datos de requeridos para que el Proveedor de Identidad (IdP) pueda agregarlo como un servicio de confianza y prestarle el servicio de autenticación. Dentro de los elementos que se deben conocer de cada Proveedor de Servicios (Sp) están el “Issuer”, “ACS” y “Subject” que utiliza para realizar la autenticación del usuario.

4.3.1. Configuración de Google Apps for Education

Google Apps for Education presta servicios en la nube a la organización y ofrece el servicio de Inicio de Sesión Único (SSO) personalizado por el usuario como alternativa al servicio de Inicio de Sesión Único (SSO) que maneja Google para sus aplicaciones.

Esta configuración se realiza a través de la consola de administración de Google dentro de las opciones de seguridad avanzada, como se observa en la Ilustración 32. Es un formulario básico que no permite personalizar la forma en que se realiza el Inicio de Sesión Único (SSO).

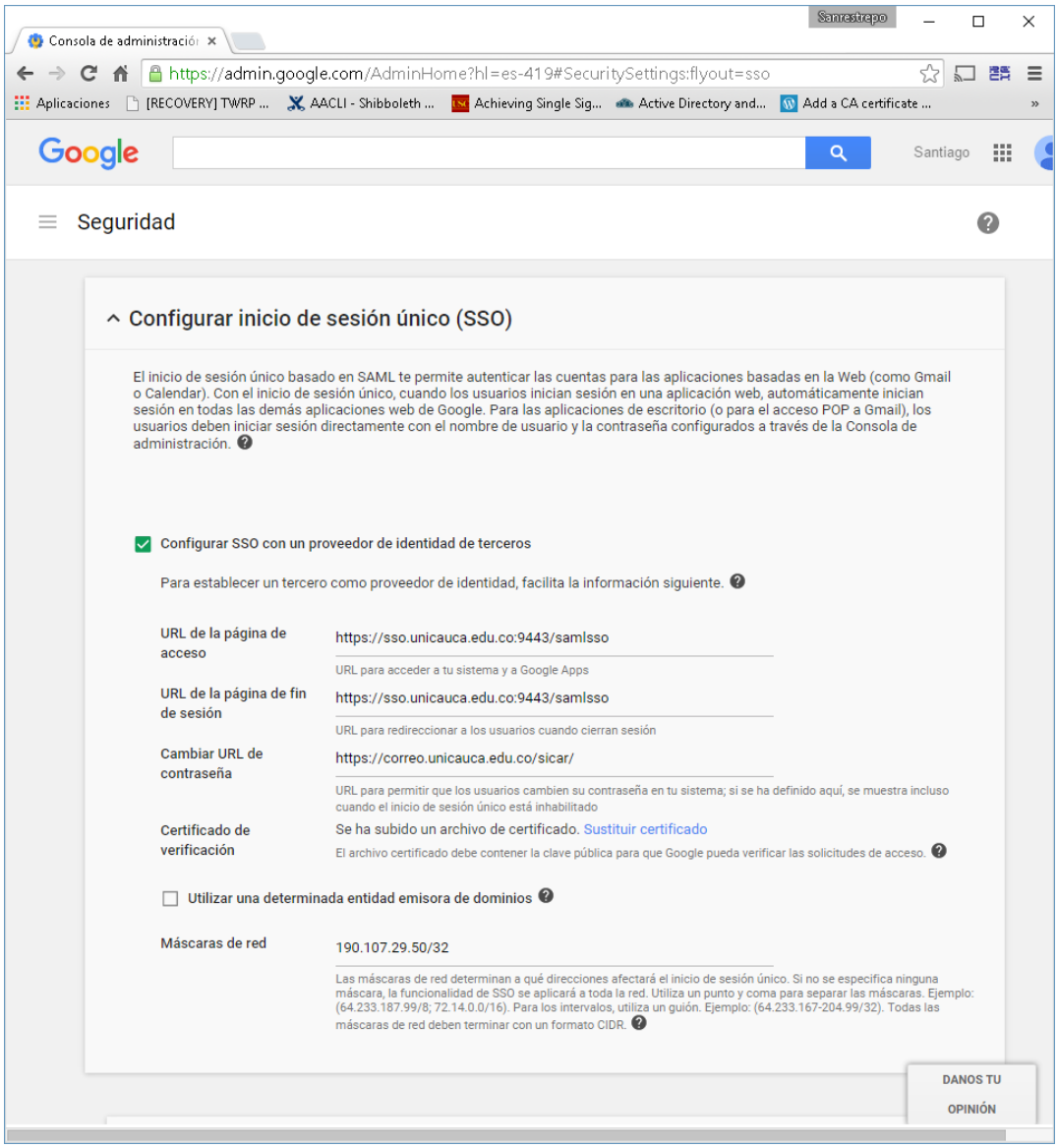

Ilustración 32. Configuración de Inicio de Sesión Único (SSO) para Google

El servicio de Inicio de Sesión Único (SSO) se activa mediante una casilla de verificación. A continuación se explican los campos de este formulario [37].

- **URL de la página de acceso:** es la URL del Proveedor de Identidad (IdP)

- **URL de la página de fin de sesión:** es la URL donde se redirige al usuario después de cerrar sesión.
- **Cambiar URL de contraseña:** es la URL donde se realiza el cambio de contraseña, en este caso redirige a una aplicación denominada “SICAR” que es la que en el siguiente apartado se configurará para integrarle un conector SAML basado en JAVA para que funcione con Inicio de Sesión Único (SSO).
- **Certificado de verificación:** es el certificado público del Proveedor de Identidad (IdP)
- **Utilizar una determinada entidad emisora de dominios:** mediante esta casilla se control el identificador del Proveedor de Servicios (SP) de Google, si esta activada el nombre de la entidad será google.com/a/unicauca.edu.co, si no será simplemente por google.com.
- **Máscaras de red:** mediante este parámetro se puede controlar a que rangos de direcciones se les va aplicar el servicio de Inicio de Sesión Único (SSO). Se debe especificar en formato CIDR[13].

Después de guardar la configuración se procede a crear el Proveedor de Servicios (SP) como se explicó en apartados anteriores, con la información respectiva de Google así:

```
Issuer: google.com
ACS: https://www.google.com/a/unicauca.edu.co/acs
NameID-format: urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress
```

La Ilustración 33 presenta la configuración de autenticación SAML para el Proveedor de Servicios (IdP) Google en el servidor WSO2 IS.

[13] Classless Inter-Domain Routing es un formato de nombrar direcciones IP y su máscara de red, es un estándar para interpretar direcciones que permite que la máscara de red no sea el límite de los octetos [50].

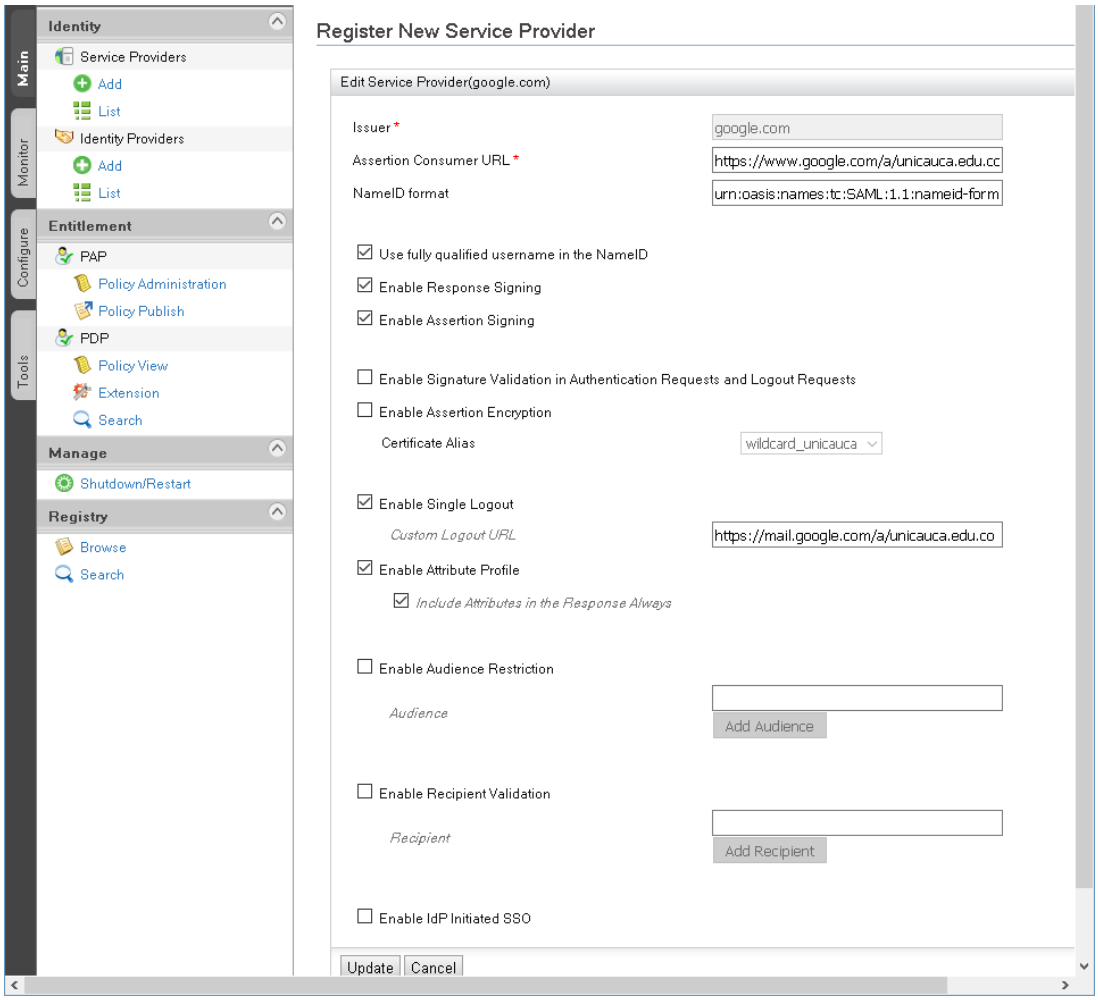

Ilustración 33. Configuración de autenticación SAML para Google

Además se debe configurar las peticiones (Claims) para que el Proveedor de Servicio (SP) pueda autenticar al usuario con el correo electrónico del sujeto del aserto, para ello la petición que realiza para que identifique al usuario es “emailAddress”, como se observa en la Ilustración 34.

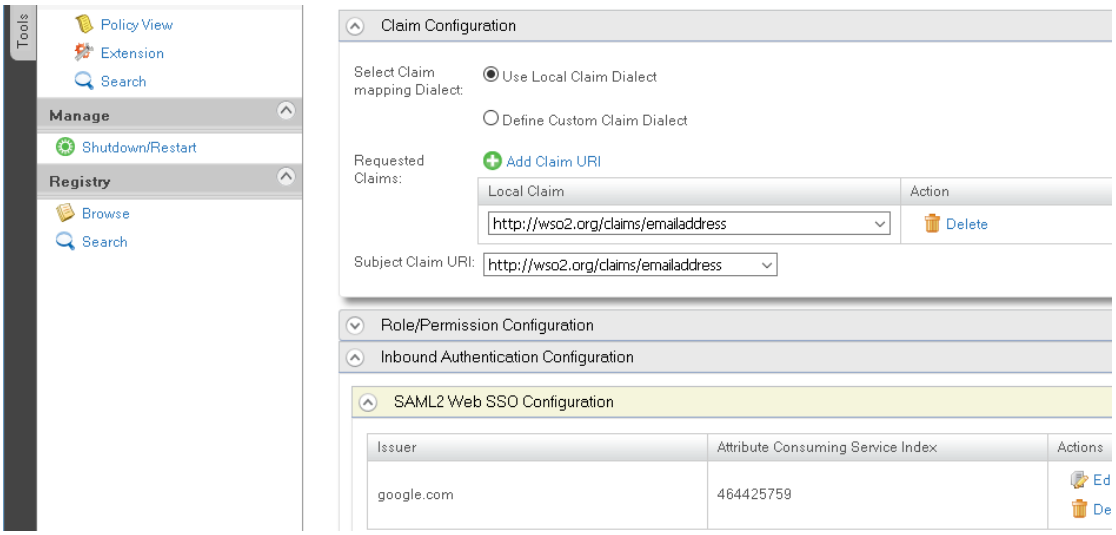

Ilustración 34. Configuración de Peticiones (Claims) para Google

Se procede a ingresar al dominio de la organización mediante la URL https://mail.google.com/a/unicauca.edu.co donde el Proveedor de Servicio (SP) envía el aserto SAML de solicitud de autenticación con los datos configurados y el Proveedor de Identidad toma la acción necesaria para autenticar o conceder el acceso dependiendo si cuenta con una sesión previa iniciada o no.

Un ejemple de aserto de solicitud de autenticación del Proveedor de Servicios (SP) Google se presenta a continuación:

```
<samlp:AuthnRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
                    ID="bbgdflmpkmgefiiflgbaapenbblidkkobepebhmm"
                    Version="2.0"
                    IssueInstant="2015-09-07T03:16:33Z"
                    ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
                    ProviderName="google.com"
                    IsPassive="false"

AssertionConsumerServiceURL="https://www.google.com/a/unicauca.edu.co/acs"
                    >
        <saml:Issuer
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">google.com</saml:Issuer>
        <samlp:NameIDPolicy AllowCreate="true"
                            Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
                            />
</samlp:AuthnRequest>
```

4.3.2. Configuración aplicación propia

Para validar el sistema de Inicio de Sesión Único (SSO) en un entorno real o de producción se hace necesario implementar un conector en JAVA que maneje el protocolo SAML y permita que una aplicación que tiene una lógica propia de autenticación delegue esta lógica al Proveedor de Identidad implementado con WSO2 IS.

WSO2 ofrece una breve introducción en la documentación de referencia del producto WSO2 Identity Server el cual servirá de guía en este piloto experimental para obtener el conector SAML [38].

En el ejemplo se codifica un servlet en JAVA para crear y leer asertos SAML, básicamente cuando el usuario se desea autenticar accede a un método del servlet el cual crea un aserto SAML de solicitud de autenticación y se lo envía al Proveedor de Identidad (IdP), cuando este último autentica de forma correcta responde con un aserto SAML de respuesta el cual llega al servlet y es capturado por otro método que ese encarga de obtener el aserto SAML de la respuesta y se pasa a otro método para su tratamiento.

Un esquema del servlet se muestra en el siguiente código:

```
public class Resource extends HttpServlet {

    private static SamlConsumer consumer = new SamlConsumer();

    public void doGet(HttpServletRequest request, HttpServletResponse response) {
                requestMessage = consumer.buildRequestMessage();
                response.sendRedirect(requestMessage);
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response) {
                responseMessage = request.getParameter("SAMLResponse").toString();
                result = consumer.processResponseMessage(responseMessage);
    }
}
```

Crear un aserto SAML:

Para crear un aserto SAML de solicitud de autenticación se crea primero el elemento "<Issuer>":

```
String issuerUrl = "http://localhost:8080/saml2.demo/consumer";
IssuerBuilder issuerBuilder = new IssuerBuilder();
Issuer issuer =
issuerBuilder.buildObject("urn:oasis:names:tc:SAML:2.0:assertion", "Issuer",
"samlp");
issuer.setValue(issuerUrl);
```

Luego se crea el aserto agregando los siguientes elementos:

```
DateTime issueInstant = new DateTime();
AuthnRequestBuilder authnRequestBuilder = new AuthnRequestBuilder();
AuthnRequest authnRequest = authnRequestBuilder.buildObject("urn:oasis:names:tc:SAML:2.0:protocol", "AuthnRequest", "samlp");
    authnRequest.setForceAuthn(new Boolean(false));
    authnRequest.setIsPassive(new Boolean(false));
    authnRequest.setIssueInstant(issueInstant);
    authnRequest.setProtocolBinding("urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST");
    authnRequest.setAssertionConsumerServiceURL(issuerUrl);
    authnRequest.setIssuer(issuer);
    authnRequest.setID(aRandomId);
    authnRequest.setVersion(SAMLVersion.VERSION_20);
```

Para poder enviar el mensaje se debe comprimir con DEFLATE, luego codificarlo en Base64 y finalmente codificarlo para agregarlo en una URL:

```
Marshaller marshaller = Configuration.getMarshallerFactory().getMarshaller(authnRequest);
Element authDOM = marshaller.marshall(authnRequest);

StringWriter rspWrt = new StringWriter();
XMLHelper.writeNode(authDOM, rspWrt);
String requestMessage = rspWrt.toString();

/* Comprimir el mensaje */
Deflater deflater = new Deflater(Deflater.DEFLATED, true);
ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
DeflaterOutputStream deflaterOutputStream = new DeflaterOutputStream(byteArrayOutputStream, deflater);
deflaterOutputStream.write(requestMessage.getBytes());
deflaterOutputStream.close();

/* Codificar el mensaje compreso */
String encodedRequestMessage = Base64.encodeBytes(byteArrayOutputStream.toByteArray(), Base64.DONT_BREAK_LINES);
String encodedAuthnRequest = URLEncoder.encode(encodedRequestMessage,"UTF-8").trim();
```

Se crea la URL de redirección que contiene el aserto SAML de solicitud de autenticación:

```
redirectionUrl = identitypProviderUrl+ "?SAMLRequest=" + encodedRequestMessage;
```

Mediante la URL anterior se redirige el navegador al Proveedor de Identidad (IdP):

```
response.sendRedirect(redirectionUrl);
```

Leer un aserto SAML:

Cuando el Proveedor de Identidad (IdP) responde con un “SAMLResponse” el mensaje debe ser obtenido de la respuesta con el siguiente código:

```
responseMessage = request.getParameter("SAMLResponse").toString();
```

Lo que se captura en la variable “responseMessage” esta codificado por lo que debe ser tratado para obtener el aserto SAML original:

```
DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
documentBuilderFactory.setNamespaceAware(true);
DocumentBuilder docBuilder = documentBuilderFactory.newDocumentBuilder();
Document document = docBuilder.parse(new ByteArrayInputStream(authnReqStr.trim().getBytes()));
Element element = document.getDocumentElement();
UnmarshallerFactory unmarshallerFactory = Configuration.getUnmarshallerFactory();
Unmarshaller unmarshaller = unmarshallerFactory.getUnmarshaller(element);
Response response = (Response) unmarshaller.unmarshall(element);
```

Cuando el mensaje es decodificado y descomprimido se pueden obtener los elementos del aserto para almacenarlos, por ejemplo para obtener el nombre de usuario del sujeto que trata el aserto SAML:

```
String subject = response.getAssertions().get(0).getSubject()
.getNameID().getValue();
```

Teniendo en cuenta que el objetivo de este piloto experimental no es profundizar en el desarrollo de código para el conector, se usa y adapta el ejemplo de WSO2 que se distribuye bajo licencia Apache. El código adaptado del conector se puede consultar el anexo A.4.

La aplicación escogida para la implementación del conector SAML en JAVA basado en OpenSAML es la que ofrece el servicio de cambio de contraseña a los usuarios de la organización.

El servicio de esta aplicación se consume directamente mediante la URL https://correo.unicauca.edu.co/sicar/ o desde Google Apps for Education se redirige automáticamente cuando el usuario accede a la opción de cambio de contraseña en las opciones de cuenta. En cualquier caso el usuario siempre es redirigido a la página de

autenticación de la aplicación que observa en la maneja una lógica interna para consultar las credenciales directamente contra el almacén de usuarios LDAP de la organización.

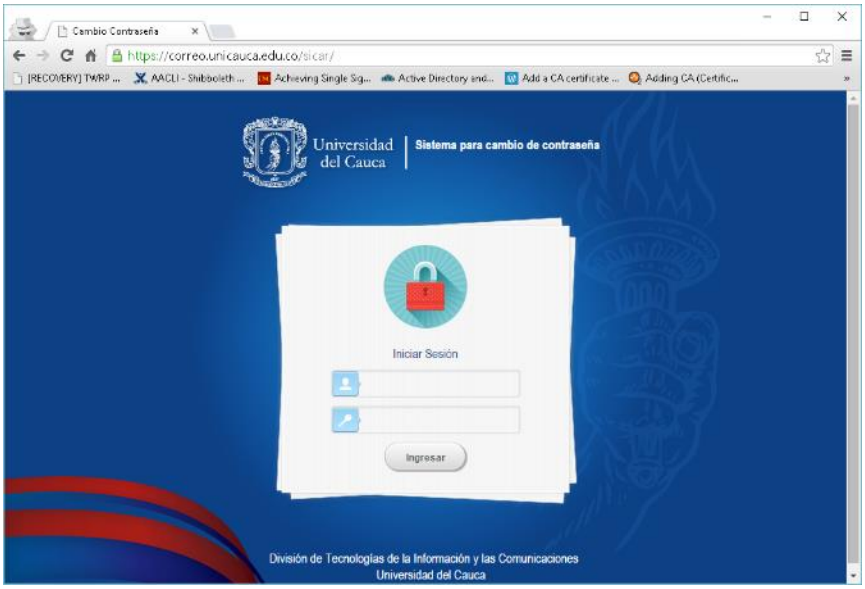

Ilustración 35. Interfaz de autenticación de SICAR

Al reemplazar la lógica propia de autenticación de la organización por el conector SAML, cuando el usuario accede por cualquier medio a la URL de la aplicación ya no se presenta la interfaz de la si no que el aplicativo envía un aserto SAML de solicitud de autenticación al Proveedor de Identidad (IdP) para que le permita acceder a la página principal de la aplicación SICAR si el usuario ya ha iniciado una sesión con Google o si no ha iniciado sesión en ningún otro aplicativo para que se le redirija a la página de autenticación del Proveedor de Identidad (IdP).

El aserto SAML de solicitud de autenticación que envía la aplicación SICAR al Proveedor de Identidad tiene la siguiente estructura:

```
<samlp:AuthnRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"

AssertionConsumerServiceURL="https://correo.unicauca.edu.co/sicar/consumer"
                    ForceAuthn="false"
                    ID="0"
                    IsPassive="false"
                    IssueInstant="2015-09-07T22:20:52.738Z"
                    ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
                    Version="2.0"
                    >
    <samlp:Issuer
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:assertion">sicar.unicauca.edu.co</samlp:Issuer>
    <saml2p:NameIDPolicy xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
                         AllowCreate="true"
                         Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent"
                         SPNameQualifier="Isser"
                         />
    <saml2p:RequestedAuthnContext xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
                                  Comparison="exact"
                                  >
            <saml:AuthnContextClassRef
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">urn:oasis:names:tc:SAML:2.0:ac:classes:
PasswordProtectedTransport</saml:AuthnContextClassRef>
    </saml2p:RequestedAuthnContext>
</samlp:AuthnRequest>
```

4.4. Métrica de calidad

Es necesario definir una medida que permita evaluar los resultados o el éxito del presente piloto experimental, lo que se conoce como métrica de calidad.

Esta métrica permite establecer un umbral mínimo de conformidad con el cual comparar los resultados obtenidos en el presente trabajo de fin de master.

Existen diferentes tipos de medidas, los cuales se resumen en la Tabla 7.

Tipología	Descripción	Ejemplo
Cuantitativa	Indicadores cuantitativos, con un objetivo numérico y cuantificable claro.	Número o porcentaje.
Cualitativa	Se refiere a una evaluación externa cualitativa	Citaciones en publicaciones.
Informe	Este tipo de medida indica que los indicadores de éxito son por una parte cuantitativos, y por otra parte cualitativos; para tener una mejor evaluación, es necesario realizar un análisis detallado que combina los dos tipos de datos.	Capacidad de influir políticas y directrices.

Entrevista	Para todos los indicadores relacionados con la interacción del usuario y su satisfacción es necesario realizar un análisis del comportamiento real del usuario para evaluar el nivel de éxito del proyecto o de sus componentes. Por lo cual, este tipo de indicadores será usado donde la interacción del usuario corresponde al criterio de calidad.	Satisfacción del usuario.
Documentación	El criterio para medir la calidad se basa en la evaluación de la documentación generada por el proyecto.	Manual de usuario y documentación del código software.

Tabla 7. Tipos de medida [39]

La métrica de calidad escogida para evaluar los resultados del piloto experimental es de tipo cuantitativo y se define como el coste en dinero de la solución. La medición de la calidad para la métrica escogida se valora mediante un criterio determinado por un "Mínimo Valor Valido (MVV)" que permite establecer un grado de desviación respecto al umbral mínimo de conformidad, este Mínimo Valor Valido (MVV) se define como la cantidad en dinero que la organización está dispuesta a invertir en la solución de Inicio de Sesión Único (SSO) y cuantitativamente es el 10% del presupuesto total de la División TIC de la Universidad del Cauca que para el año 2015 ascendió a trescientos cincuenta mil dólares (USD 350.000), lo que da un Valor Mínimo Valido de treinta y cinco mil dólares (USD 35.000), esto significa que la solución no debe superar el Valor Mínimo Valido (MVV) para poder decir que el piloto experimental tuvo éxito ya que el costo total de la solución se mantuvo dentro del umbral mínimo de conformidad. En la Tabla 8 se resumen los criterios que se usaran para medir la calidad en la métrica escogida, donde la variable "Y" es igual a treinta y cinco mil dólares (USD 35.000).

Criterio	Fórmula	Desviación	
Costo de la solución < Mínimo Valor Valido (MVV)	X < Y	Verde	
Costo de la solución = Mínimo Valor Valido (MVV)	X = Y	Amarillo	
Costo de la solución > Mínimo Valor Valido (MVV)	X > Y	Rojo	

Tabla 8. Criterios para la medición de la calidad

De la desviación obtenida se pueden obtener las conclusiones y recomendaciones respecto a la calidad de los resultados obtenidos mediante el piloto experimental, estas se resumen en la Tabla 9.

Desviación	Conclusión	Recomendación
	Los resultados están alineados con los objetivos.	El piloto experimental concluyó con un nivel alto de calidad.
	Los resultados siguen la línea establecida por el proyecto pero los niveles de calidad no se han alcanzado.	El piloto experimental concluyó pero se necesita de un análisis de utilidad y eficiencia de los resultados obtenidos para determinar si mediante cambios se pueda confirmar la validez de la solución dentro de la métrica de calidad establecida.
	Los resultados están claramente por debajo del nivel de calidad esperado.	El piloto experimental concluyó pero no se cumplió con la métrica de calidad establecida por lo que se requiere un análisis profundo de los objetivos para plantear un nuevo rumbo para un nuevo proyecto que se trace.

Tabla 9. Conclusiones y recomendaciones sobre la calidad

4.4.1. Costo de la solución

En este apartado se calcula el costo total de la solución para poder determinar si está dentro la desviación adecuada dentro de los criterios de calidad establecidos en el punto anterior.

Para ello se tienen en cuenta los siguientes elementos:

- **Tiempo:** El tiempo durante el cual se planeó el piloto experimental que es de 16 semanas.
- **Recursos humanos:** El recurso humano que se considera es el del autor del trabajo de fin de master quien es un ingeniero que trabaja en la organización cuyo valor por de hora laborada en la organización es de seis punto seis dólares (USD 6.6),

Semanas	Horas	Total Horas	Valor Hora (USD)	Total Recurso Humano (USD)
16	48	768	6,6	5075,84

Tabla 10. Costo del recurso humano

- **Recursos técnicos**
  - **Hardware:** La solución se implementó mediante el uso de máquinas virtuales en un servidor Dell Power Edge M620 que tiene un valor de nueve mil dólares (USD 9000), para el piloto experimental se usaron tres máquinas virtuales, una para las pruebas de instalación del Proveedor de Identidad (IdP), otra para la

instalación en producción de este y la última para la instalación de los Proveedores de Servicios (SP) de prueba locales. Como uno de los trabajos futuros especificados en este trabajo de fin de master considera la implementación de un esquema de alta disponibilidad, se calculan los costos de recursos hardware para que incluyan en un mediano plazo este tipo de configuraciones por lo que se estima que se requerirán de dos (2) servidores con esta configuración, asumiendo un costo total de recurso hardware de dieciocho mil dólares (USD 18.000).

- **Software:** Una de las bondades principales del piloto experimental presentado en este trabajo de fin de master es que tanto para la implementación del Proveedor de Identidad (IdP) como de los conectores para las aplicaciones se usaron herramientas y estándares de libre distribución. Los sistemas operativos que se usaron en cada paso de la implementación eras basados en Linux y las herramientas de ofimática también eran de software libre. Estos planteamientos hacen que el costo total del recurso software sea igual a cero (0).
- **Mantenimiento:** En los costos de mantenimiento se tienen en cuenta los costos de colocación de los dos servidores en el centro de datos de la Universidad del Cauca, estos costos serian principalmente de la energía eléctrica y el de aire acondicionado de precisión durante todo el año, ya que las soluciones en la nube ofrecían servicios mensuales y se calculó el costo de las mismas por el mismo periodo de tiempo, además el costo de mantenimientos programados de hardware y software. Según [40] el costo de colocación por cada servidor asciende a los dos mil dólares (USD 2.000) más los costos de mantenimiento preventivo que equivaldrían a mil dólares (USD 1.000) adicionales, lo que conllevaría a un costo total de mantenimiento de seis mil dólares (USD 6.000) por los recursos hardware durante un año.

Finalmente, en la Tabla 11 se resume el costo total de la solución, discriminado por cada recurso utilizado, aquí se observa que el costo total de la solución no excede el Mínimo Valor Valido (MVV) planteado como criterio para medir la calidad del presente piloto experimental por lo que se puede asegurar que los objetivos del proyecto se alcanzaron dentro de unos parámetros de alta calidad.

Recurso	Valor total (USD)
Humano	5075,84
Técnico	
Hardware	1.8000
Software	0
Mantenimiento	6.000
COSTO TOTAL	**29.075,84**

Tabla 11.Costo total de la solución

4.5. Discusión de resultados

En la etapa de pruebas del apartado anterior donde se trató de simular un entorno producción al configurar Proveedores de Servicios (SP) internos como en la nube se puede decir que los resultados obtenidos fueron los esperados ya que el sistema de Inicio de Sesión Único (SSO) logró autenticar al usuario cuando no tenía ninguna sesión iniciada y de allí en adelante cuando este iba accediendo a los recursos de otras aplicaciones estas consultaban al Proveedor de Identidad (IdP) acerca del estado de la sesión y este respondía afirmativamente permitiéndole acceder a los recursos sin tener que ingresar de nuevo las credenciales. Lo propio sucedió en el entorno de cierre de sesión, ya que cuando el usuario cerro sesión en una Proveedor de Servicios (SP), el Proveedor de Identidad (IdP) invalido de forma inmediata la sesión activa del usuario por lo que cuando el usuario intento de nuevo acceder a otros Proveedores de Servicios (SP) no lo pudo hacer con la misma sesión ya que se le presentaba de nuevo el proceso de autenticación, confirmando así que el proceso de cierre de sesión se propagaba correctamente a través de todos los Proveedores de Servicios (SP).

Al analizar los comportamientos que se obtuvieron durante la prueba, tanto del Proveedor de Identidad (IdP) como de los Proveedores de Servicios (SP), se pudo concluir que el sistema centralizado de autenticación estaba ofreciendo de manera correcta el servicio de Inicio de Sesión Único (SSO) por lo que sirvió para poder continuar con el siguiente paso del piloto experimental que consistía en configurar una aplicación en producción que en este caso fue federación de identidad con Google Apps for Education y con una aplicación interna propia cuyo código se debió adaptar para que delegara la lógica de autenticación al Proveedor de Identidad. Con estas dos aplicaciones se realizaron las mismas pruebas de inicio y cierre de sesión y los resultados fueron los mismos que se obtuvieron para el entorno de pruebas lo que permitió validar la solución como un sistema de producción y operativo listo para ofrecer servicios de autenticación centralizada mediante Inicio de Sesión Único (SSO).

El parámetro de métrica de calidad establecido para el piloto experimental permite concluir que el piloto experimental cumplió con el objetivo general del trabajo de fin de master que

pretendía realizar una implementación de un sistema de Inicio de Sesión Único (SSO) para una organización, que requería como primer paso dentro de un proceso de migración a este tipo de soluciones, que se centralizara la autenticación para sus aplicaciones locales y en la nube y que se le ofreciera una forma rápida y económica conectar las aplicaciones web actuales al nuevo sistema implementado.

Los costos totales distan enormemente de una aplicación en la nube como Okta u OneLogin ya que este tipo de soluciones además de ofrecer el servicio de Inicio de Sesión Único (SSO) ofrecen toda una plataforma de administración de identidad que en este caso no eran relevantes para la organización que lo que buscaba era solo centralizar la autenticación y ofrecer servicios web con experiencia de Inicio de Sesión Único (SSO) y al contratar una solución como las planteadas en el apartado 2.1 se estaría invirtiendo dinero en una solución que al final quedaría subutilizada por la organización.

A lo largo del trabajo de fin de master se desarrollan diferentes etapas o fases con el fin de lograr el objetivo general mediante el cumplimiento de los objetivos específicos, a continuación se presenta los resultados obtenidos:

- Se obtuvo el estado del arte actual relacionado con los trabajos académicos que pueden servir de referencia para dar un enfoque holístico al aporte del trabajo de grado, además del estado actual de las tecnologías y herramientas del mercado que se utilizan en las organizaciones para las actividades de autenticación. De este estado del arte se logró identificar una herramienta de libre distribución que soportada a cabalidad las necesidades de la organización.
- Se implementó un servidor Proveedor de Identidad para la Universidad del Cauca mediante el uso de la herramienta WSO2 Identity Server logrando así centralizar la autenticación para las aplicaciones web internas y externas de la organización y ofreciendo servicios de Inicio de Sesión Único (SSO) de una forma fácil, económica, eficiente y segura.
- Se analizaron e implementaron varias herramientas que cumplen la función de Proveedores de Servicios (SP) lo cual permitió contar con un entorno de pruebas valido que simula un entorno real de una organización al contar con aplicaciones web desarrolladas en diferentes lenguajes y que operan dentro de la infraestructura de la organización y en la nube.
- Se logró la configuración de federación de identidad para Google Apps for Education para que funcione como Proveedor de Servicios dentro del entorno de Inicio de Sesión Único (SSO) permitiendo unificar los almacenes de datos que usan todas las aplicaciones tanto internas como externas.

- Se logró adaptar el código de un conector SAML basado en JAVA para su implementación en una aplicación web en producción de la Universidad del Cauca denominada SICAR, este conector será la base para la migración de las lógicas de autenticación del resto de aplicaciones que se deseen incluir en el sistema centralizado de autenticación.

Cada resultado anterior es fruto del esfuerzo de cada etapa que se permiten dar respuesta a la pregunta que dio inicio al planteamiento que aborda este trabajo de fin de master la cual era la siguiente: ¿Cómo implementar un sistema de autenticación centralizado para ofrecer servicios con experiencia de Inicio de Sesión Único (SSO) en la Universidad del Cauca, haciendo uso de herramientas de libre distribución que minimice al máximo los costos relacionados?.

Al responder a la pregunta del planteamiento del problema se da cumplimiento al objetivo general del piloto experimental el cual era implementar un sistema Proveedor de Identidad (IdP) para ofrecer servicios de autenticación mediante Inicio de Sesión Único (SSO) en la Universidad del Cauca, haciendo uso de herramientas y estándares de libre distribución que minimicen al máximo el costo total de la solución.

CAPÍTULO 5. CONCLUSIONES, LIMITACIONES Y TRABAJO FUTURO

5.1. Conclusiones

- El piloto experimental desarrollado en el marco del trabajo de fin de master se puede usar como referencia dentro de las organizaciones que desean implementar un servicio de autenticación centralizado con servicio de Inicio de Sesión Único de una forma rápida, económica y fácil, lo que les permite optimizar el tiempo y los recursos en proyectos de este tipo.
- El uso de herramientas de libre distribución hace que el sistema final sea de bajo costo sin dejar de ser un sistema óptimo para entornos en producción manteniendo la seguridad y compatibilidad con otros sistemas al estar basado en estándares ampliamente adoptados en el mercado.
- La métrica de calidad usada en el trabajo de fin de master permite evaluar los resultados del piloto experimental ubicándolos dentro de un umbral mínimo de conformidad que representa que el costo total de la solución presenta aquí ofrece bondades a quienes pretendan realizar una implementación de un sistema de Inicio de Sesión Único (SSO) de una manera rápida y económica.
- El plan de pruebas determinado para la validación de la solución permitió verificar el correcto funcionamiento de todos los elementos que hacen parte de la solución implementada durante el piloto experimental.
- Los sistemas de Inicio de Sesión Único (SSO) contribuyen de manera positiva al proyecto de seguridad en una organización ya que al centralizar la autenticación en el Proveedor de Identidad (IdP) se agrega un paso adicional para la protección del almacén de usuarios, evitando que aplicaciones débiles directamente conectadas a dicho almacén se puedan convertir en un plataforma para lanzar ataques contra este sensible elemento de la organización.
- Los sistemas de Inicio de Sesión Único (SSO) permiten eliminar el síndrome de fatiga de las contraseñas y al no existir este problema se puede incentivar campañas de seguridad de contraseñas seguras dentro de la organización.
- La federación de autenticación que se logra con los sistemas de Inicio de Sesión Único (SSO) es necesaria para que las organizaciones puedan establecer relaciones de confianza entre sí y puedan ofrecer servicios a los usuarios, ya que en la actualidad las organizaciones no pueden trabajar aisladas del mundo exterior debido a que los servicios en la nube son cada vez más comunes.

- El uso de herramientas que tienen ya tiempo en el mercado, y si se quiere de libre distribución facilitan la tarea de implementación de un sistema de Inicio de Sesión Único (SSO) y son la alternativa más factible cuando una organización desea iniciar un proyecto de este tipo, ya que desarrollar un sistema propio desde cero, conlleva gran consumo de tiempo y recursos y difícilmente logra igualar la seguridad que un sistema que ya ha sido probado y depurado por toda la comunidad.
- WSO2 Identity Server se consolida en el mercado como una opción muy fuerte frente a sus competidores al ser completamente de libre distribución e implementar los estándares más usados en la actualidad. Además su facilidad de personalización le otorgan un valor agregado que en ocasiones solo se iguala en aplicaciones comerciales.

5.2. Limitaciones

- Debido a que la implementación del servicio de SSO demando una alta cantidad de tiempo, sobre todo en el tema relacionado a la creación de un conector debido a la falta de experiencia en el desarrollo de aplicaciones en lenguaje JAVA no se logró investigar lo relacionado con la parte de autorización en el servidor.
- Debido a que el autor del presente trabajo de fin de master dejó de laborar en la Universidad del Cauca, lugar donde se ejecutó el piloto experimental, no se logró realizar pruebas de desempeño de la solución, debido a que al final del trabajo de fin de master ya no contaba con acceso a los recursos proporcionados por la organización ni al sistema implementado.

5.3. Trabajo futuro

- Implementar el estándar XAML para ofrecer servicios de autorización en el Proveedor de Identidad (IdP) y diseñar e implementar una aplicación que permite la validación del entorno.
- Realizar pruebas de desempeño al sistema Inicio de Sesión Único (SSO) para determinar el uso de una configuración de alta disponibilidad que debe incluir múltiples instancias a nivel de almacén de usuarios, base de datos de registros y servidores WSO2 Identity Server.
- Realizar un análisis comparativo de las herramientas de libre distribución analizadas en el estado del arte para obtener un estudio en profundidad que permita caracterizar a cada herramienta y permita la toma de decisiones en las organizaciones basadas en sus necesidades particulares.

- Implementar en el Proveedor de Identidad la posibilidad de autenticación mediante certificados digitales, incluyendo la configuración de una PKI que permita atender a los usuarios de la organización.
- Configuración de los servicios de aprovisionamiento que ofrece la herramienta WSO2 Identity Server para que funcione como un servidor de administración de identidad.
- Realizar un análisis de vulnerabilidades a todos los elementos involucrados en el sistema Inicio de Sesión Único (SSO) para detectar y corregir fallos que puedan comprometer la seguridad de la solución.

REFERENCIAS

[1] C. Mitchell y A. Pashalidis, «A taxonomy of Single Sign-On Systems,» de *A taxonomy of Single Sign-On Systems*, Wollongong, 2003.

[2] P. Yacano, «Once is enough: Single Sign On,» de *18th Australasian Conference on Information Systems*, Toowoomba, 2007.

[3] D. Florencio y C. Herley, «A Large-Scale Study of Web Password Habits,» de *Proceedings of the 16th international conference on World Wide Web*, Banff, 2007.

[4] Muy Interesante, «¿Qué es el síndrome de fatiga de las contraseñas?,» [En línea]. Disponible en: http://www.muyinteresante.es/curiosidades/preguntas-respuestas/que-es-el-sindrome-de-fatiga-de-las-contrasenas-821394791796. [Último acceso: 12 Agosto 2015].

[5] J. Hursti, «Single Sign-On,» de *Proceedings of Helsinki University of Technology Seminar on Network Security 1997*, Helsinki, 1997.

[6] M. D. Guel, «A Framework for Choosing Your Next Generation Authentication/Authorization System,» *Information Security Technical Report,* vol. 7, pp. 63-78, 2002.

[7] G. Kumrawat y A. Amit, «A Survey: Single Sign On (SSO),» *International Journal of Technology Research and Management,* vol. 2, nº 3, 2015.

[8] J. I. Martín, «Implantación de un SSO (Single Sign On),» Universitat Oberta de Catalunya, 2014.

[9] H. Jernevad, «Developing a Single Sign-On System - A Java-based authentication platform aimed at the web.,» Chalmers University of Technology, Gothenburg, 2009.

[10] A. Palas Andrade, «Strong Mobile Authentication in Single Sign-On Systems,» Aalto University, Espoo, 2011.

[11] S. Bui, C. E. Irvine y T. D. Nguyen, «Services, Single Sign-on Solution for MYSEA,» Monterey, 2005.

[12] S. Xiong, «Web Single Sign-On System for WRL Company,» Royal Institute of Technology, Estocolmo, 2005.

[13] S. Rhermini, «Identity, Access Management and Single Sign-On Web-based Solutions,» KTH Information and Communication Technology, Estocolmo, 2012.

[14] F. Martinsson y M. Akerlund, «Single Sig-On SSO,» 2001.

[15] M. P. Pravinbhai, «Implementation of Multi-tier Authentication Technique for Single-Sign On access of Cloud Services,» Odisha, 2014.

[16] X. Magrinya Bonafont, «Analysis and Implementation of a SSO Solution for Several Web Portal,» 2012.

[17] Apereo, «About CAS | Apereo,» [En línea]. Disponible en: https://www.apereo.org/projects/cas/about-cas. [Último acceso: 08 Agosto 2015].

[18] Gluu, Inc., «Gluu Server Overview | The Gluu Server for SSO, WAM, & 2FA | Gluu,» [En línea]. Disponible en: http://www.gluu.org/gluu-server/overview/. [Último acceso: 08 Agosto 2015].

[19] S. Gonzalez Oyuela, «JOSSO - Java Open Single Sign-On Project Home - Atricore,» Atlassian Confluence, 31 Mayo 2014. [En línea]. Disponible en: http://www.josso.org/confluence/display/JOSSO1/JOSSO+-+Java+Open+Single+Sign-On+Project+Home. [Último acceso: 08 Agosto 2015].

[20] Atricore, «JOSSO,» [En línea]. Disponible en: http://www.atricore.com/products/josso.html. [Último acceso: 08 Agosto 2015].

[21] Okta, Inc., «Product: Okta,» [En línea]. Disponible en: https://www.okta.com/product/identity-management/. [Último acceso: 08 Agosto 2015].

[22] OneLogin, Inc., «Why Choose OneLogin? See Reasons Why Customers Choose Us.,» [En línea]. Disponible en: https://www.onelogin.com/why-onelogin. [Último acceso: 08 Agosto 2015].

[23] ForgeRock, «Access Management - OpenAM - Home - ForgeRock.com,» [En línea]. Disponible en: https://www.forgerock.com/app/uploads/2015/06/3.pdf. [Último acceso: 08 Agosto 2015].

[24] Shibboleth, «Shibboleth Consortium - What's Shibboleth,» [En línea]. Disponible en: https://shibboleth.net/about/. [Último acceso: 08 Agosto 2015].

[25] Uninnet, «SimpleSAMLphp,» [En línea]. Disponible en: https://simplesamlphp.org/. [Último acceso: 08 Agosto 2015].

[26] SSOCircle, «Public IDP,» [En línea]. Disponible en: http://www.ssocircle.com/en/portfolio/publicidp/. [Último acceso: 28 Agosto 2015].

[27] WSO2, «WSO2 Identity Server: The First Enterprise Identity Bus | WSO2 Inc,» [En línea]. Disponible en: http://wso2.com/products/identity-server/. [Último acceso: 08 Agosto 2015].

[28] J. R. Bermejo Higuera, «Seguridad en aplicaciones online y bases de datos: Seguridad en servicios web,» 2015.

[29] S. Cantor, J. Kemp, R. Philpott y E. Maler, Assertions and protocols for the OASIS security assertion markup language (SAML) v2.0, Oasis standard, OASIS Open, 2005.

[30] S. Cantor, J. Kemp, R. Philpott, F. Hirsch y E. Maler, «Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0,» OASIS Open, 2005.

[31] S. Cantor, J. Kemp, R. Philpott, F. Hirsch y E. Maler, «Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0,» OASIS Open, 2005.

[32] N. Sakimura, «OpenID Connect | OpenID,» OpenID, [En línea]. Disponible en: http://openid.net/connect/. [Último acceso: 08 Agosto 2015].

[33] WSO2, «WSO2 Identity Server Documentation - Identity Server 5.0.0 - WSO2 Documentation,» Atlassian, [En línea]. Disponible en: https://docs.wso2.com/display/IS500/WSO2+Identity+Server+Documentation. [Último acceso: 08 Agosto 2015].

[34] S. Roering, «SSO Tracer :: Add-ons for Firefox,» Mozilla, [En línea]. Disponible en: https://addons.mozilla.org/En-us/firefox/addon/sso-tracer/. [Último acceso: 08 Agosto 2015].

[35] J. Odvarko y J. Hewitt, «Firebug,» Mozilla, [En línea]. Disponible en: http://getfirebug.com/. [Último acceso: 08 Agosto 2015].

[36] Confluence, «Home - OpenSAML 2 - Confluence,» Shibboleth, [En línea]. Disponible en: https://wiki.shibboleth.net/confluence/display/OpenSAML/Home/. [Último acceso: 08 Agosto 2015].

[37] Google, «Configurar el inicio de sesión único (SSO) para las cuentas de Google Apps - Ayuda de Administrador de Google Apps,» [En línea]. Disponible en: https://support.google.com/a/answer/60224?hl=es. [Último acceso: 08 Agosto 2015].

[38] S. Attanayake, «SAML2 Web Browser based SSO with WSO2 Identity Server | WSO2 Inc,» WSO2, 13 Julio 2010. [En línea]. Disponible en: http://wso2.com/library/articles/2010/07/saml2-web-browser-based-sso-wso2-identity-server/. [Último acceso: 08 Agosto 2015].

[39] J. J. Rainer, «Gestión de la calidad, riesgos y evaluación: Herramientas de gestión y evaluación».

[40] Dell Inc., «Server consolidation and TCO: Dell PowerEdge M620 vs. Dell PoerEdge M710HD,» 2012.

[41] Google, «SAML Single Sign-On (SSO) Service for Google Apps,» 6 Enero 2015. [En línea]. Disponible en: https://developers.google.com/google-apps/sso/saml_reference_implementation. [Último acceso: 7 Agosto 2015].

[42] TechTarget, «What is token? - Definition from WhatIs.com,» TechTarget, Septiembre 2005. [En línea]. Disponible en: http://whatis.techtarget.com/definition/token. [Último acceso: 08 Agosto 2015].

[43] Google, «Google for Education: Save time and stay connected,» Google Inc., [En línea]. Disponible en: https://www.google.com/edu/products/productivity-tools/ . [Último acceso: 08 Agosto 2015].

[44] The OpenLDAP Project, «OpenLDAP, Project,» The OpenLDAP Foundation, [En línea]. Disponible en: http://www.openldap.org/project/. [Último acceso: 08 Agosto 2015].

[45] Debian Project, «Debian -- Acerca de Debian,» Software in the Public Interest, Inc., 19 Julio 2015. [En línea]. Disponible en: https://www.debian.org/intro/about#what. [Último acceso: 08 Agosto 2015].

[46] Wikipedia, the free encyclopedia, «OpenAM - Wikipedia, the free encyclopedia,» Wikimedia Foundation, Inc., 24 Marzo 2015. [En línea]. Disponible en: https://en.wikipedia.org/wiki/OpenAM. [Último acceso: 08 Agosto 2015].

[47] MIT, «Kerberos: The Network Authentication Protocol,» 18 07 2015. [En línea]. Disponible en: http://web.mit.edu/kerberos/#what_is. [Último acceso: 08 Agosto 2015].

[48] wwwcosic, «What is SESAME,» [En línea]. Disponible en: https://www.cosic.esat.kuleuven.be/sesame/html/sesame_what.html. [Último acceso: 08 Agosto 2015].

[49] OpenID, «OpenID Connect FAQ and Q&As,» 2015. [En línea]. Disponible en: http://openid.net/connect/faq/. [Último acceso: 08 Agosto 2015].

[50] empowerID, «WS-Trust and WS-Federation,» [En línea]. Disponible en: http://www.empowerid.com/learningcenter/standards/ws-trust-fed. [Último acceso: 08 Agosto 2015].

[51] D. Estryk y G. Ríos, «Métodos De Compresión En HTTP,» Noviembre 2002. [En línea]. Disponible en: http://www.fiuba6662.com.ar/6648/presentaciones/Moebius/index.htm#MDeflate. [Último acceso: 08 Agosto 2015].

[52] B. Halabi, S. Halabi y D. McPherson, Internet Routing Architectures, Cisco, 2000.

ANEXOS

A.1. Preparación del servidor

El servidor que se utilizó para el desarrollo del piloto experimental cuenta con las siguientes características:

Hardware:

- Servidor: Dell PowerEdge M620 - Máquina Virtual desplegada en VMware ESXI 5.5.0 Build 1331820.
- Memoria RAM: 6 Gb
- Procesadores: 2 núcleos de 4 sockets virtuales de un procesador Intel Xeon E5-2620v2 de 2.10 GHz
- Adaptador de red: E1000 a 1Gbps

Software:

- Sistema operativo: Debian 8.1
- Java: 1.7.0.79
- WSO2 Identity Server 5.0.0 SP 1

La información del sistema operativo se aprecia a continuación:

```
root@sso:~# uname -a
Linux sso 3.16.0-4-amd64 #1 SMP Debian 3.16.7-ckt11-1+deb8u3 (2015-08-04)
x86_64 GNU/Linux
```

Debido a la importancia del servicio que va a albergar este servidor, se hace necesaria una configuración previa para mejorar la seguridad que por defecto ofrece una instalación limpia del sistema operativo conocida como hardening.

Para ello se personaliza la configuración del servidor ssh mediante la configuración del demonio OpenSSH[14], se edita el archivo */etc/ssh/sshd_config* para agregar las siguientes etiquetas:

- Deshabilitar el protocolo SSH-1 porque está obsoleto:

```
Protocol 2
```

- Deshabilitar comandos RSH obsoletos:

```
IgnoreRhosts yes
```

14

- Comentar la etiqueta de escucha por defecto (puerto 22) y agregar el puerto 2322 para la conexión de los administradores:

```
Port 2322
```

- Cambiar la siguiente etiqueta para evitar las conexiones mediante el usuario root:

```
PermitRootLogin no
```

- Permitir solo la escucha por IPv4:

```
AddressFamily inet
```

Se debe además crear un usuario denominado “sso” y un grupo “sso-ssi” para que este sea el encargado de ejecutar los procesos relacionados al proveedor de identidad, de la siguiente manera:

```
root@sso:~# addgroup sso-ssi
Añadiendo el grupo `sso-ssi' (GID 1001) ...
Hecho.
root@sso:~# useradd -m -s /bin/sh sso
root@sso:~# adduser sso sso-ssi
Añadiendo al usuario `sso' al grupo `sso-ssi' ...
Añadiendo al usuario sso al grupo sso-ssi
Hecho.
```

A.2. Instalación de simpleSAMLphp

Se instala el servidor web apache desde el repositorio:

```
root@sso:~# apt-get install apache2
```

Instalar PHP y las extensiones relacionadas:

```
root@sso:~# apt-get install apache2
root@sso:~# apt-get install php5
root@sso:~# apt-get install php5-cli
root@sso:~# apt-get install php5-common
root@sso:~# apt-get install php5-curl
root@sso:~# apt-get install php-pear
root@sso:~# apt-get install php5-mcrypt
root@sso:~# apt-get install php5-json
```

Se procede a instalar la herramienta simpleSAMLphp mediante la siguiente secuencia de comandos:

```
root@sso:~# cd /var/simplesamlphp/
root@sso:~# mkdir /var/simplesamlphp/
root@sso:~# cd /var/simplesamlphp/
root@sso:~# wget
http://simplesamlphp.googlecode.com/files/simplesamlphp-1.11.0.tar.gz
root@sso:~# tar xvf simplesamlphp-1.11.0.tar.gz
root@sso:~# mv simplesamlphp-1.11.0 simplesamlphp
root@sso:~# cd simplesamlphp
root@sso:~# cp -r metadata-templates/*.php metadata/
root@sso:~# cp -r config-templates/*.php config
root@sso:~# cd /var/www/html
root@sso:~# ln -s /var/simplesamlphp/simplesamlphp/www simplesaml
root@sso:~# apachectl start
```

Se verifica que el servidor y la herramienta se ejecuten correctamente accediendo a la dirección http://ssopruebas.unicauca.edu.co/simplesaml como se observa en la Ilustración 36.

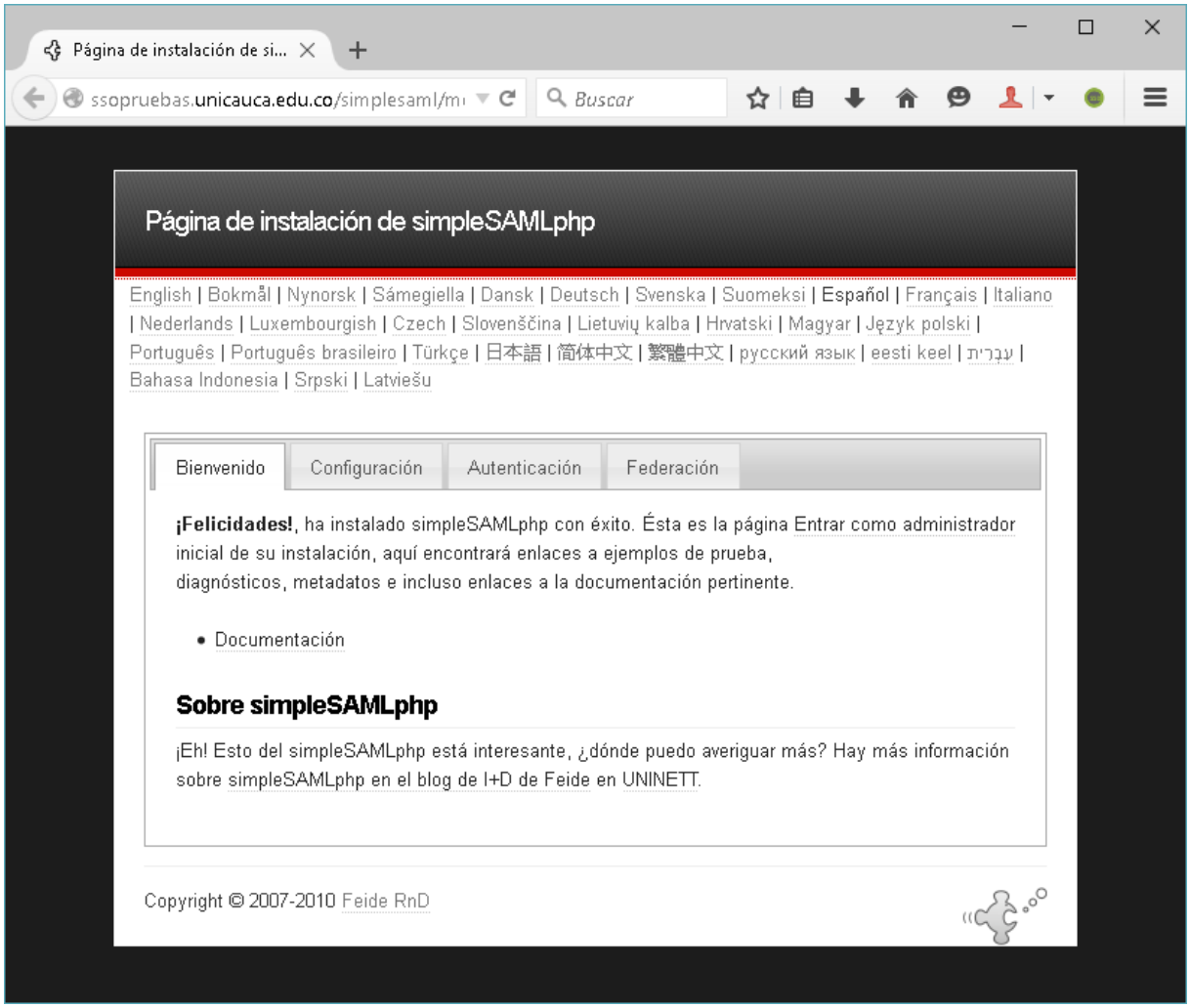

Ilustración 36. Interfaz de simpleSAMLphp

A.3. Instalación de Travelocity.com

Se obtiene la aplicación de ejemplo del repositorio mediante subversión, esto crea una carpeta denominada “sso”.

```
root@sso:~# svn co
http://svn.wso2.org/repos/wso2/carbon/platform/branches/turing/products/
is/5.0.0/modules/samples/sso/
```

Dentro de la carpeta “sso” se debe editar el archivo *pom.xml* y asignarle el siguiente contenido:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
    <groupId>org.wso2.identity</groupId>
    <version>5.0.0</version>
    <modelVersion>4.0.0</modelVersion>
    <artifactId>wso2is-identity-samples-sso</artifactId>
    <packaging>pom</packaging>
    <name>Identity Server : SSO Samples</name>
    <modules>
        <module>SSOAgentSample</module>
    </modules>
</project>
```

Se compila la aplicación de ejemplo mediante el siguiente comando:

```
root@sso:~# mvn clean install
```

Esto genera el archivo *travelocity.com.war* en la ruta *./sso/SSOAgentSample/target*, copiar este archivo en la carpeta de despliegues del servidor de aplicaciones, la ruta en el servidor de pruebas es */usr/src/tomcat7/webapps/*.

Par verificar que se desplego de manera correcta accedemos a la dirección de la aplicación de ejemplo en http://ssopruebas.unicauca.edu.co:8080/travelocity.com/index.jsp, como se observa en la Ilustración 37.

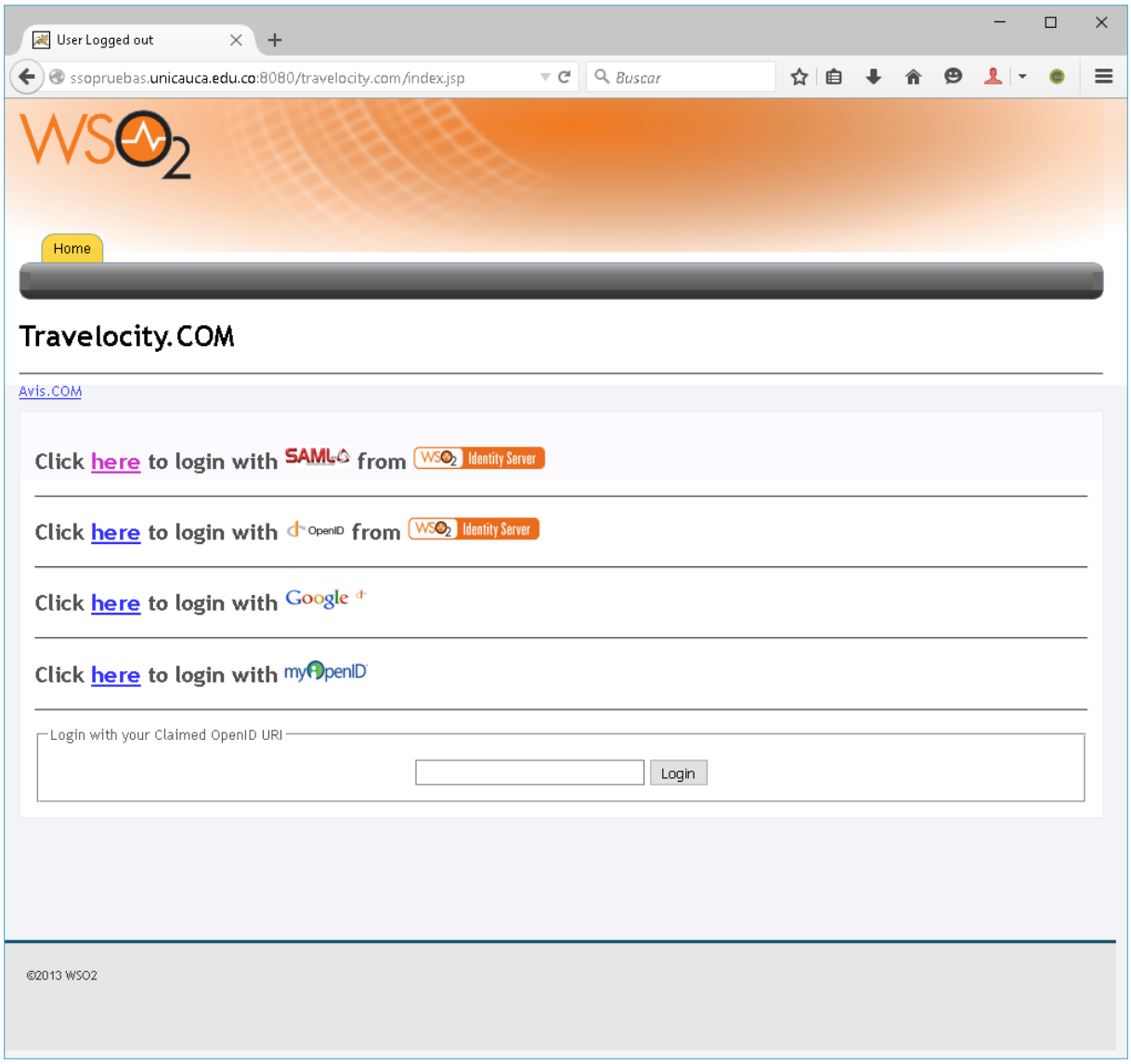

Ilustración 37. Interfaz de Travelocity.com

A.4. Código JAVA

SAML2ConsumerServlet.java

```
/*

* Copyright (c) 2005-2010, WSO2 Inc. (http://www.wso2.org) All Rights Reserved.

 *
 * WSO2 Inc. licenses this file to you under the Apache License,
 * Version 2.0 (the "License"); you may not use this file except
 * in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 * software distributed under the License is distributed on an
 * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
```

```
 * KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations
 * under the License.
 */
package co.edu.unicauca.cambiopass.sso;

import java.io.IOException;
import java.util.Map;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.opensaml.xml.ConfigurationException;

/**
 * Clase de implementacion del servlet SAML2ConsumerServlet
 */
public class SAML2ConsumerServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;
    private SamlConsumerManager consumer;

    /**
     * Servlet init
     */
    public void init(ServletConfig config) throws ServletException {
        try {
            consumer = new SamlConsumerManager(config);
        } catch (ConfigurationException e) {
            throw new ServletException("Errow while configuring
SAMLConsumerManager", e);
        }
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
     *      response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException,

IOException {
        doPost(request, response);
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
     *      response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException,

IOException {
        String responseMessage = request.getParameter("SAMLResponse");

        if (responseMessage != null) { /* Respuesta del proveedor de identidad */
```

```
            Map<String, String> result =
consumer.processResponseMessage(responseMessage);

            String subject = result.get("Subject");

            if (subject!=null) {
                /*
                 * No se retornaron atributos. Se redirige a principal
                 */
                response.sendRedirect("principal.xhtml?subject=" +
subject.replace("@unicauca.edu.co", ""));
//                response.sendRedirect("principal.xhtml");
            } else {
                // Fallo. Autenticacion de nuevo
                response.sendRedirect("index.xhtml");
            }

        } else { /* Creacion del aserto de solicitud de autenticacion o cierre de
sesion */

            try {
                String requestMessage = consumer.buildRequestMessage(request);

                response.sendRedirect(requestMessage);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

SamlConsumerManager.java

```
package co.edu.unicauca.cambiopass.sso;


import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.StringWriter;
import java.net.URLEncoder;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.zip.Deflater;
import java.util.zip.DeflaterOutputStream;
import javax.servlet.ServletConfig;
import javax.servlet.http.HttpServletRequest;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.apache.axiom.util.UIDGenerator;
import org.joda.time.DateTime;
import org.opensaml.Configuration;
import org.opensaml.DefaultBootstrap;
import org.opensaml.common.SAMLVersion;
```

```
import org.opensaml.saml2.core.Assertion;
import org.opensaml.saml2.core.Attribute;
import org.opensaml.saml2.core.AttributeStatement;
import org.opensaml.saml2.core.AuthnContextClassRef;
import org.opensaml.saml2.core.AuthnContextComparisonTypeEnumeration;
import org.opensaml.saml2.core.AuthnRequest;
import org.opensaml.saml2.core.Issuer;
import org.opensaml.saml2.core.LogoutRequest;
import org.opensaml.saml2.core.NameID;
import org.opensaml.saml2.core.NameIDPolicy;
import org.opensaml.saml2.core.RequestAbstractType;
import org.opensaml.saml2.core.RequestedAuthnContext;
import org.opensaml.saml2.core.Response;
import org.opensaml.saml2.core.SessionIndex;
import org.opensaml.saml2.core.impl.AuthnContextClassRefBuilder;
import org.opensaml.saml2.core.impl.AuthnRequestBuilder;
import org.opensaml.saml2.core.impl.IssuerBuilder;
import org.opensaml.saml2.core.impl.LogoutRequestBuilder;
import org.opensaml.saml2.core.impl.NameIDBuilder;
import org.opensaml.saml2.core.impl.NameIDPolicyBuilder;
import org.opensaml.saml2.core.impl.RequestedAuthnContextBuilder;
import org.opensaml.saml2.core.impl.SessionIndexBuilder;
import org.opensaml.xml.ConfigurationException;
import org.opensaml.xml.XMLObject;
import org.opensaml.xml.io.Marshaller;
import org.opensaml.xml.io.MarshallingException;
import org.opensaml.xml.io.Unmarshaller;
import org.opensaml.xml.io.UnmarshallerFactory;
import org.opensaml.xml.io.UnmarshallingException;
import org.opensaml.xml.util.Base64;
import org.opensaml.xml.util.XMLHelper;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.xml.sax.SAXException;

public class SamlConsumerManager {

    private String consumerUrl = null;
    private String issuer = "sicar.unicauca.edu.co";
    private String AuthReqRandomId = Integer.toHexString(new
Double(Math.random()).intValue());
    private String relayState = null;
    private ServletConfig servletConfig;

    public SamlConsumerManager(ServletConfig config) throws ConfigurationException
{
        this.servletConfig = config;
        /* Inicializacion de la libreria, loading default configurations */
        DefaultBootstrap.bootstrap();
    }

    /**
     * Devuelve la URL de redireccion con la codificacion SAML2
     * Request message
     *
     * @param request
     *
     * @return redirectionUrl<dependency>
     *         <groupId>org.opensaml</groupId>
     *         <artifactId>opensaml</artifactId>
     *         <version>2.2.6</version>
```

```
     *         </dependency>
     */
    public String buildRequestMessage(HttpServletRequest request) {

        String consumerURL = request.getRequestURL().toString();

        String contextPath = "https://correo.unicauca.edu.co/sicar";

        consumerUrl = contextPath + "/consumer";

        String redirectionUrl = Util.getConfiguration(servletConfig, "IdpUrl");

        RequestAbstractType requestMessage = null;

        // Construccion del aserto de solicitud de autenticacion o cierre de sesion
        if (request.getParameter("logout") == null) {
            requestMessage = buildAuthnRequestObject(consumerURL);

        } else { // Solicitud de cierre de sesion
            requestMessage = buildLogoutRequest((String)
request.getSession().getAttribute("user"));
        }

        String encodedRequestMessage = null;
        try {
            encodedRequestMessage = encodeRequestMessage(requestMessage);
        } catch (MarshallingException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

        /* Se agrega la solicitud de autenticacion SAML2 a la URL del IdP */
        return redirectionUrl + "?SAMLRequest=" + encodedRequestMessage +
"&RelayState=" +
                relayState;
    }

    private LogoutRequest buildLogoutRequest(String user) {

        System.out.println(">>> Building a logout request!");

        LogoutRequest logoutReq = new LogoutRequestBuilder().buildObject();

        logoutReq.setID(Util.createID());

        DateTime issueInstant = new DateTime();
        logoutReq.setIssueInstant(issueInstant);
        logoutReq.setNotOnOrAfter(new DateTime(issueInstant.getMillis() + 5 * 60 *
1000));

        IssuerBuilder issuerBuilder = new IssuerBuilder();
        Issuer issuer = issuerBuilder.buildObject();
        issuer.setValue(this.issuer);
        logoutReq.setIssuer(issuer);

        NameID nameId = new NameIDBuilder().buildObject();
        nameId.setFormat("urn:oasis:names:tc:SAML:2.0:nameid-format:entity");
        nameId.setValue(user);
        logoutReq.setNameID(nameId);
```

```
        SessionIndex sessionIndex = new SessionIndexBuilder().buildObject();
        sessionIndex.setSessionIndex(UIDGenerator.generateUID());
        logoutReq.getSessionIndexes().add(sessionIndex);

        logoutReq.setReason("Single Logout");

        return logoutReq;
    }

    private AuthnRequest buildAuthnRequestObject(String consumerURL) {

        /* Construccion del objeto ISSUER */
        IssuerBuilder issuerBuilder = new IssuerBuilder();
        Issuer issuer =

issuerBuilder.buildObject("urn:oasis:names:tc:SAML:2.0:assertion",
                                                "Issuer", "samlp");
//      issuer.setValue(consumerUrl); /* IssuerUL and ConsumerURl are same here */
        issuer.setValue(this.issuer); /* IssuerUL and ConsumerURl are same here */

        /* NameIDPolicy */
        NameIDPolicyBuilder nameIdPolicyBuilder = new NameIDPolicyBuilder();
        NameIDPolicy nameIdPolicy = nameIdPolicyBuilder.buildObject();
        nameIdPolicy.setFormat("urn:oasis:names:tc:SAML:2.0:nameid-
format:persistent");
        nameIdPolicy.setSPNameQualifier("Isser");
        nameIdPolicy.setAllowCreate(new Boolean(true));

        /* AuthnContextClass */
        AuthnContextClassRefBuilder authnContextClassRefBuilder = new
AuthnContextClassRefBuilder();
        AuthnContextClassRef authnContextClassRef =

authnContextClassRefBuilder.buildObject("urn:oasis:names:tc:SAML:2.0:assertion",

"AuthnContextClassRef",

"saml");

authnContextClassRef.setAuthnContextClassRef("urn:oasis:names:tc:SAML:2.0:ac:classe
s:PasswordProtectedTransport");

        /* AuthnContex */
        RequestedAuthnContextBuilder requestedAuthnContextBuilder =
                                                                    new
RequestedAuthnContextBuilder();
        RequestedAuthnContext requestedAuthnContext =
requestedAuthnContextBuilder.buildObject();

requestedAuthnContext.setComparison(AuthnContextComparisonTypeEnumeration.EXACT);
        requestedAuthnContext.getAuthnContextClassRefs().add(authnContextClassRef);

        DateTime issueInstant = new DateTime();

        /* Construccion de AuthRequestObject */
        AuthnRequestBuilder authRequestBuilder = new AuthnRequestBuilder();
        AuthnRequest authRequest =

authRequestBuilder.buildObject("urn:oasis:names:tc:SAML:2.0:protocol",
                                                              "AuthnRequest",
"samlp");
```

```
        authRequest.setForceAuthn(new Boolean(false));
        authRequest.setIsPassive(new Boolean(false));
        authRequest.setIssueInstant(issueInstant);
        authRequest.setProtocolBinding("urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
POST");
        authRequest.setAssertionConsumerServiceURL(consumerUrl);
        authRequest.setIssuer(issuer);
        authRequest.setNameIDPolicy(nameIdPolicy);
        authRequest.setRequestedAuthnContext(requestedAuthnContext);
        authRequest.setID(AuthReqRandomId);
        authRequest.setVersion(SAMLVersion.VERSION_20);

        return authRequest;
    }

    private String encodeRequestMessage(RequestAbstractType requestMessage)
                                                                        throws
MarshallingException,

IOException {

        Marshaller marshaller =
Configuration.getMarshallerFactory().getMarshaller(requestMessage);
        Element authDOM = marshaller.marshall(requestMessage);

        Deflater deflater = new Deflater(Deflater.DEFLATED, true);
        ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
        DeflaterOutputStream deflaterOutputStream =
                                                    new
DeflaterOutputStream(byteArrayOutputStream,

deflater);
        /* Comprimiendo el mensaje */
        StringWriter rspWrt = new StringWriter();
        XMLHelper.writeNode(authDOM, rspWrt);
        deflaterOutputStream.write(rspWrt.toString().getBytes());
        deflaterOutputStream.close();

        /* Codificando el mensaje comprimido */
        String encodedRequestMessage =

Base64.encodeBytes(byteArrayOutputStream.toByteArray(),
                                                        Base64.DONT_BREAK_LINES);
        return URLEncoder.encode(encodedRequestMessage, "UTF-8").trim();
    }

    public Map<String, String> processResponseMessage(String responseMessage) {

        XMLObject responseXmlObj = null;

        try {
            responseXmlObj = unmarshall(responseMessage);

        } catch (ConfigurationException e) {
            e.printStackTrace();
        } catch (ParserConfigurationException e) {
            e.printStackTrace();
        } catch (SAXException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
```

```
        } catch (UnmarshallingException e) {
            e.printStackTrace();
        }

        return getResult(responseXmlObj);
    }

    private XMLObject unmarshall(String responseMessage) throws
ConfigurationException,

ParserConfigurationException, SAXException,
                                                        IOException,
UnmarshallingException {

        DocumentBuilderFactory documentBuilderFactory =
DocumentBuilderFactory.newInstance();
        documentBuilderFactory.setNamespaceAware(true);
        DocumentBuilder docBuilder = documentBuilderFactory.newDocumentBuilder();

        byte[] base64DecodedResponse = Base64.decode(responseMessage);

        System.out.println("Response: " + new String(base64DecodedResponse));

        ByteArrayInputStream is = new ByteArrayInputStream(base64DecodedResponse);

        Document document = docBuilder.parse(is);
        Element element = document.getDocumentElement();
        UnmarshallerFactory unmarshallerFactory =
Configuration.getUnmarshallerFactory();
        Unmarshaller unmarshaller = unmarshallerFactory.getUnmarshaller(element);
        return unmarshaller.unmarshall(element);

    }

    /*
     * Manejo de la respuesta y retorno de resultados
     */
    private Map<String, String> getResult(XMLObject responseXmlObj) {

        if (responseXmlObj.getDOM().getNodeName().equals("saml2p:LogoutResponse"))
{
            return null;
        }

        Response response = (Response) responseXmlObj;

        Assertion assertion = response.getAssertions().get(0);
        Map<String, String> results = new HashMap<String, String>();

        /*
         * If the request has failed, the IDP shouldn't send an assertion.
         * SSO profile spec 4.1.4.2 <Response> Usage
         */
        if (assertion != null) {

            String subject = assertion.getSubject().getNameID().getValue();
            results.put("Subject", subject); // get the subject

            List<AttributeStatement> attributeStatementList =
assertion.getAttributeStatements();
```

```
            if (attributeStatementList != null) {
                // Se reciben los atributos del usuario
                Iterator<AttributeStatement> attribStatIter =
attributeStatementList.iterator();
                while (attribStatIter.hasNext()) {
                    AttributeStatement statment = attribStatIter.next();
                    List<Attribute> attributesList = statment.getAttributes();
                    Iterator<Attribute> attributesIter = attributesList.iterator();
                    while (attributesIter.hasNext()) {
                        Attribute attrib = attributesIter.next();
                        Element value =
attrib.getAttributeValues().get(0).getDOM();
                        String attribValue = value.getTextContent();
                        results.put(attrib.getName(), attribValue);
                    }
                }
            }
        }
        return results;
    }
}
```

Util.java

```
package co.edu.unicauca.cambiopass.sso;

import java.util.Random;

import javax.servlet.ServletConfig;

public class Util {

    /**
     * Genera un ID unico para incluir en las solicitudes de autenticacion
     *
     * @return ID unico generado
     */
    public static String createID() {

        byte[] bytes = new byte[20]; // 160 bit

        new Random().nextBytes(bytes);

        char[] charMapping = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j',
'k', 'l', 'm', 'n', 'o', 'p'};

        char[] chars = new char[40];

        for (int i = 0; i < bytes.length; i++) {
            int left = (bytes[i] >> 4) & 0x0f;
            int right = bytes[i] & 0x0f;
            chars[i * 2] = charMapping[left];
            chars[i * 2 + 1] = charMapping[right];
        }

        return String.valueOf(chars);
    }

    /**
     * Lee las opciones de configuracion del archivo web.xml
```

```
     * @param servletConfig
     * @param configuration
     * @return
     */
    public static String getConfiguration(ServletConfig servletConfig, String
configuration){
        return servletConfig.getInitParameter(configuration);

    }

}
```