Gestión de identidad en la NUBE

Cesar Augusto Mesa Henao

Jefe Departamento

Edwin Nelson Montoya Múnera

Asesor

David Manuel Holguin Arias

UNIVERSIDAD EAFIT

DEPARTAMENTO DE INGENIERIA DE SISTEMAS

MEDELLIN

2015

Tabla de contenido

Tabla de ilustraciones

1. Definición

Con el crecimiento de usuarios en internet las empresas están ampliando su portafolio de productos y servicios creando estrechas relaciones con entidades.

La mayoría de los servicios que las organizaciones exponen en la red requieren como mínimo una identificación que valide que la persona que está intentando ingresar a algún recurso realmente sea un cliente. ¿Pero qué pasa si son varios recursos? El cliente debe identificarse una y otra vez ¿y qué pasa si son otros servicios expuestos por otra organización? ¿Cómo podemos valorar o cuantificar la eficiencia de un proveedor de contenidos o de servicios? ¿Los usuarios no quieren recordar más claves, es posible integrar el proveedor de identidades con redes sociales? ¿A nivel de desarrollo y arquitectura que tan difícil puede resultar esta implementación?

Con la integración cada vez mayor de diferentes sistemas de información, se busca un procedimiento que habilite a los usuarios el acceso a varios recursos de diferentes entidades con una sola instancia de identificación. Basado no solo en login y contraseña si no que verifique otras características propias de un cliente. En pocas palabras autenticar y autorizar el ingreso a diferentes servicios web con una sola identificación.

Con esto no solo logramos un ambiente más amigable para el cliente, sino además un seguimiento que permita ofrecer productos y servicios que satisfagan las necesidades de los clientes.

2. Objetivos

2.1 Objetivo General:

- Desarrollar un sistema de autenticación federada

2.2 Objetivos Específicos:

- Tomar requisitos del sistema
- Desarrollar web services para la consulta de clientes
- Establecer modelos de comunicación vía metadatos para la comunicación entre entidades
- Implementar modelos de datos que permita generar estadísticas del acceso a las plataformas por parte del usuario
- Desarrollar formulario de autenticación
- Implementar sistemas de reportes
- Ampliar el sistema de identificación de usuarios a las redes sociales
- Implementar un proceso para la integración del sistema con nuevos servicios

3. Alcance

Se desarrollará un sistema de autenticación federada basado en WEB como proceso transversal de proyectos actuales y futuros de software cara cliente

4. Introducción

En los últimos tiempos el incremento exponencial del acceso a la web ha motivado a la mayoría de las empresas expandir sus mercados, crear nuevas alianzas estratégicas y proveer a sus clientes nuevos productos y servicios.

La implementación tecnológica que presentamos busca precisamente integrar los clientes con servicio externos a la empresa, permitiendo unir fuerzas que apoyen el plan estratégico de las organizaciones y además sea una fuente importante para facilitar el cumplimiento de las metas financieras.

Esta integración se basa en el concepto de Autenticación y Autorización. Bajo el esquema de **SP** (Services provider) e **IDP** (identity provider), permitiendo un procedimiento de single sign on web al cliente que facilite una navegación más amigable.

Este procedimiento será la base para hacer seguimiento a una fuente de información para apoyar el mercadeo. La integración se implementara en multiplataforma, eso quiere decir que el cliente no solo podrá ingresar via web sino también desde android , IOS , Smart Tv y PC

Nuestro anteproyecto formula la pregunta ¿cómo se puede establecer un modelo de autenticación con diferentes proveedores y autorizando según la características del cliente, que permita mantener una sesión sin necesidad de ingresar de nuevo las credenciales de ingreso y generar una trazabilidad que ayuden a la toma de decisiones? Para ello establecemos un marco teorico que nos oriente para llegar a la solución del problema.

5. Justificación e Importancia

Con accesos a internet cada vez más rápidos, más servicios online disponibles, nuestro modelo permitirá:

5.1. Plataforma neutral

Se abstrae el marco de la seguridad lejos de puestas en práctica y de arquitecturas particulares de vendedores.

5.2 Acoplador flojo de directorios

No se requiere la información del usuario para ser mantenido y sincronizado entre principales.

5.3 Experiencia en línea mejorada para usuarios finales

Las aserciones de la autentificación permiten “single sign-on” consintiendo que los usuarios se autentifiquen en un proveedor de identidad y después tengan acceso a servicios/recursos en los proveedores de servicio sin autentificación adicional.

5.4 Costes administrativos reducidos para los proveedores de servicio

El uso del modelo federado entre dominios de identidad puede reducir el coste de mantenimiento de la información de la cuenta (por ejemplo el nombre de usuario y la contraseña). Esta carga se pone en el proveedor de identidad.

5.5 Transferencia del riesgo

Se puede ceder la responsabilidad de la gestión de las identidades al proveedor de identidad, que es a menudo más compatible con su modelo de negocios que el de un proveedor de servicios.

6. Metodología

Para el proyecto se usara una metodología iterativa, haciendo entregas parciales del proyecto.

Actividad
Análisis
Diseño y modelamiento de base de datos
Desarrollo web services
Desarrollo de vistas
Desarrollo de la autenticación federada
Desarrollo integración Facebook

7. Marcos referenciales

7.1 Marco contextual

La autenticación es el proceso de verificación de un afirmación que se hace relativa a una identidad, un atributo perteneciente a dicha identidad, (p. ej. “esta persona es ciudadana de los Estados Unidos de América"), o un grupo de atributos. Tradicionalmente, la gran demanda para soluciones de autenticación seguras ha provenido de empresas que buscaban satisfacer sus propias necesidades de seguridad al interior de sus organizaciones, así como de organizaciones gubernamentales en contextos donde se creía que los intereses de la seguridad nacional estaban en juego. En los últimos años, la demanda por (y la adopción de) soluciones de autenticación segura han estado marcadamente en aumento en todos los otros tipos de contexto que afectan directamente la privacidad de los individuos en una escala inimaginable hace dos décadas. Mucho de esto es provocado por la creciente popularidad de la Internet y de las redes de comunicaciones móviles, así como del rápido incremento en la cantidad de PCs y de dispositivos de información tales como teléfonos móviles con capacidades Web y las computadoras portátiles.

A medida que nuevas arquitecturas de autenticación están siendo desarrolladas y adoptadas para una siempre creciente cantidad de aplicaciones, la privacidad de los individuos está siendo erosionada a un ritmo impresionante, a menudo con poca o ninguna justificación. Los nuevos mecanismos de comunicación electrónica y de transacción automáticamente capturan y registran identidades en sistemas de computadoras centrales sin que los individuos si quiera se percaten de ello. A medida que más y más información personal es recolectada y registrada en los

sistemas centrales, las políticas y las tradicionales salvaguardas de seguridad para prevenir la filtración y el abuso rápidamente se están volviendo ineficaces.

Mucha de esta explosiva tensión entre la (percibida o real) necesidad de autenticación por un lado y las demandas de privacidad por el otro pueden ser atribuidas a una falsa creencia muy extendida: a saber, que la identificación es lo mismo que autenticación, y que la privacidad y la autenticación son objetivos opuestos. Esta errónea creencia es perpetuada por todo tipo de influyentes organizaciones normativas. La Organización Internacional para la Estandarización (ISO)1, por ejemplo, define autenticación como "la condición de garantía de la afirmación de identidad de una entidad," y el Grupo de Trabajo de Ingeniería de Internet2 (IETF) define la autenticación como "el proceso de verificación de una identidad afirmada por o para una entidad del sistema." Asimismo, a nivel político, la autenticación y la identidad a menudo son erróneamente equiparadas.

El hecho real es que la autenticación es una noción mucho más amplia que la identificación. En muchos contextos, la autenticación no requiere identificación. Es más, las organizaciones a menudo no están interesadas en la identidad per se de la persona con la cual están tratando, sino solo la confirmación de contactos previos de dicha persona, la afiliación de la persona a un grupo, la autenticidad de los datos personales de esa persona, los derechos o privilegios de esa persona, y así sucesivamente. Por ejemplo, para autenticar si un usuario está permitido de comprar alcohol, todo lo que necesita ser autenticado es que el usuario tiene al

menos 21 años de edad. En este ejemplo, la identificación de la persona solo servirá como un medio indirecto para lograr la autenticación que es de interés ("mayor de 21 años de edad").

En el "viejo" mundo, los individuos podían fácilmente lograr acceder a servicios sin revelar su identidad, ya sea mostrando sus privilegios o derechos o proporcionando a los proveedores de servicios identificadores "adecuados al contexto", tales como un número de empleado o un número del seguro social. Mientras que tales identificadores sirven para identificar a los usuarios, estos solo lo hacen dentro de esferas especificas de actividad; las organizaciones no pueden utilizarlos para cruzar perfiles de usuarios entre esferas de actividad.

Lamentablemente, la mayoría de las tecnologías de autenticación más difundidas hoy en día (tales como contraseñas, características biométricas, Kerberos, y PKI) básicamente causan ineludibles identificaciones a través de identificadores que son mundialmente únicos. Estas tecnologías de autenticación basadas en la identidad fueron inventadas hace muchas décadas, cuando las redes abiertas existían difícilmente, ni que decir de organizaciones buscando compartir información personal en forma segura por medio de esas redes. Consecuentemente, la única protección de privacidad que los diseñadores de técnicas tradicionales de autenticación tenían en mente fue la protección ante las interceptaciones y otros intrusos no autorizados. Sin embargo, las tecnologías tradicionales de autenticación no son apropiadas para tratar las crecientes necesidades de autenticación en estos días y épocas, ya que ellas permiten a las organizaciones rastrear y hacer cruces de perfiles de usuarios sobre la base de identificadores mundiales únicos (tales como claves cifradas) que son ineludiblemente asignadas a ellos.

Una tendencia igualmente preocupante es la centralización de los poderes de autenticación de diferentes organizaciones dentro de una sola organización confiable que actúa en representación de todas las organizaciones constituyentes. En su arquitectura original de Passport por ejemplo, Microsoft confió en la centralización de todos los datos recolectados de los visitantes de páginas web con el fin de proporcionar servicios de autenticación en nombre de un número rápidamente creciente de sitios web. Microsoft abandonó esta arquitectura luego de demandas de privacidad por parte de grupos de consumidores y funcionarios de la UE,así como de una falta de adopción por parte de los proveedores de servicios quienes fueron altamente reticentes a confiarles los datos de sus clientes.

La arquitectura de autenticación "federada" promovida por la Liberty Alliance (una alianza industrial de aproximadamente 160 industrias claves en una amplia gama de sectores, dirigidas por muchas compañías principales que fueron reticentes a delegar su autonomía a la iniciativa original del Passport de Microsoft) deja los datos personales en las organizaciones que los colectan, y permite la coexistencia de múltiples "círculos de confianza". Sin embargo, aún esta arquitectura no hace nada para mejorar la privacidad de los usuarios: el poder de autenticación (y por tanto el poder del control de acceso) permanece centralizado.

Específicamente, cuando un proveedor de servicio trata con un usuario, este consulta en tiempo real a la central "proveedora de identidad" en su círculo de confianza; el proveedor de identidad simplemente devuelve una confirmación de la autenticación como validación de la identidad afirmada del acceso solicitado, la cual es utilizada por el proveedor del servicio para su

propio proceso de autorización. Aunque los usuarios puedan ser "seudónimos" dirigidos a los proveedores de servicios (en Liberty Alliance el proveedor de identidad asigna diferentes nombres de usuario al mismo usuario, uno por proveedor de servicio), estos no están ciertamente vis-à-vis con la más poderosas de las partes en esta arquitectura: el proveedor de identidades. Dentro de cada círculo de confianza, el proveedor de identidad puede hacer un seguimiento, ubicar y enlazar en tiempo real todas las interacciones entre los usuarios y las organizaciones. El proveedor de identidad puede incluso suplantar usuarios y falsamente negarles acceso a cualquier sitio.

Mientras que tales enfoques de centralización de autenticación puedan satisfacer las necesidades de las grandes empresas que desean hacer manejo de identidades relacionadas a empleados y a proveedores dentro de sus sucursales internas, más allá de este restringido contexto el enfoque rápidamente se vuelve altamente problemático con relación a la privacidad. Este podría incluso estar en conflicto con la legislación de privacidad. Si es adoptado a una escala gubernamental, las implicaciones de estas arquitecturas invasivas de la privacidad serían ciertamente sin precedentes.

En un mundo electrónico, si al nivel técnico (por medio del análisis del flujo electrónico de datos) todos es inevitablemente identificable a través de identificadores únicos globales, la legislación sobre privacidad se vuelve virtualmente intrascendente; ¿Cómo podría uno forzar a las organizaciones a no colectar información identificable cuándo no pueden impedir que esta sea entregada a ellos? La única salida del aparente conflicto entre la autenticación y la privacidad es

recurrir a tecnologías de autenticación que técnicamente separen la noción de autenticación de aquella de identificación. Dos décadas de investigación en criptografía han demostrado que la autenticación segura y la privacidad no son concesiones, sino que se refuerzan mutuamente cuando se implementan apropiadamente. Utilizando las técnicas que están enraizadas en la criptografía moderna tales como la Digital Credentials,6 es completamente posible el realizar una autenticación segura sin necesariamente requerir la identificación. Por ejemplo, la autenticación basada en roles puede ser implementada de manera tal que el solicitante del acceso no pueda ser identificado.

De manera más general, las técnicas de autenticación que preservan la identidad permiten a cada parte involucrada en el procesamiento electrónico y en el reenvío de información sensible en términos de privacidad a retener de manera segura un control fino sobre la información, aun cuando la información es transmitida electrónicamente más allá de los cortafuegos de la corporación y a través de dominios organizacionales arbitrarios. En ningún punto de la cadena de información electrónica la transferencia de información de una parte a la siguiente permitirá a ninguna de las partes saber más de lo que el remitente expresamente permite.

Esfuerzos internacionales de investigación están actualmente en curso para crear sistemas de autenticación que preserven el anonimato, e incluyan el desarrollo de nuevas tecnologías que favorezcan la privacidad para ser utilizadas en tales programas. Estas tecnologías favorables a la privacidad permiten la separación de autenticación e identificación y están siendo desplegadas en respuesta a vulnerabilidades de seguridad. Tales tecnologías pueden insertarse en metasistemas

de identificación, tales como el CardSpace de Microsoft Mientras que la configuración por defecto de CardSpace no cumple actualmente con estándares establecidos para la preservación de la privacidad, este modelo debe ser estudiado en detalle cuando se consideren tecnologías de autenticación.

En junio de 2007, el Consejo de la OECD aprobó la Recomendación alentando a los Estados Miembros a establecer enfoques compatibles, tecnológicamente neutros para facilitar la autenticación a través de las fronteras. La Guía fija el contexto y la importancia de la autenticación electrónica para el comercio electrónico, el gobierno electrónico y muchas otras interacciones sociales. Este dispone varios principios fundamentales y operacionales que constituyen un denominador común para la interoperatividad a través de las jurisdicciones. Según la OEDCD, la autenticación electrónica es un componente esencial de cualquier estrategia para proteger los sistemas y redes de información, datos financieros, información personal y otros activos del acceso no autorizado o del robo de identidad. La autenticación electrónica es por tanto esencial para el establecimiento de una rendición de cuentas en línea.

Cada vez que se implemente una nueva medida de autenticación para un mecanismo de transacción existente o nuevo, es imperativo que los diseñadores y adoptantes analicen cuanta información personalmente identificable es realmente necesaria de ser revelada para propósitos de autenticación. Asumiendo que la revelación de información es necesaria y proporcional con relación a la naturaleza de la transacción, entonces se debe buscar implementar las necesidades de

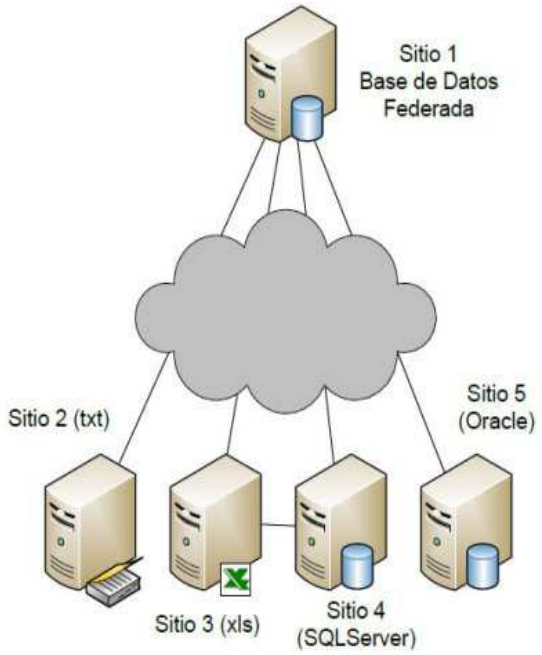
seguridad utilizando las tecnologías de autenticación que protejan la privacidad, en vez de recurrir a los enfoques basados en la ineludible identificación.

7.2 Marco teórico

La identidad federada es una de las soluciones para abordar la gestión de identidad en los sistemas de información. El valor añadido adicional respecto a otras soluciones es la gestión de identidad interdependiente entre compañías.

Como cualquier solución de gestión de identidades, su objetivo es obtener una gestión de usuarios eficiente, la sincronización de los datos identificativos, gestión de acceso, servicios de agrupación, servicios de directorio, auditoría e informes.

Las organizaciones que deseaban desplegar identidades federadas se veían obligadas a negociar el protocolo a utilizar con cada socio de la federación. De este modo, tenían que soportar múltiples protocolos mediante técnicas de conversión que no siempre garantizaban la integridad de las características clave de estas soluciones.



Mediante soluciones de Identidad Federada los individuos pueden emplear la misma identificación personal (típicamente usuario y contraseña) para identificarse en redes de diferentes departamentos o incluso empresas. De este modo las empresas comparten información sin compartir tecnologías de directorio, seguridad y autenticación, como requieren otras soluciones. Para su funcionamiento es necesaria la utilización de estándares que definan mecanismos que permiten a las empresas compartir información entre dominios. El modelo es aplicable a un grupo de empresas o a una gran empresa con numerosas delegaciones y se basa en el "círculo de confianza" de estas, un concepto que identifica que un determinado usuario es conocido en una comunidad determinada y tiene acceso a servicios específicos.

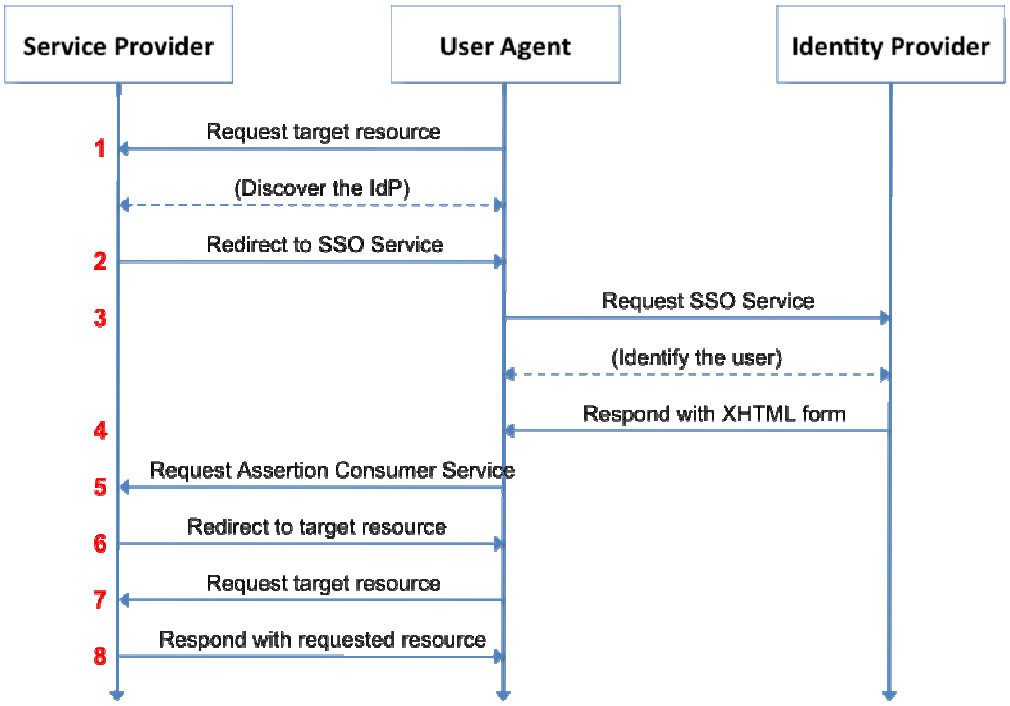
Nuestra autenticación federada describe dos roles de federación: el de proveedor del servicio, que es la entidad que da acceso al usuario a un recurso o aplicación; y el de proveedor

de identidad, responsable de la autenticación del usuario. Ambos, proveedor del servicio y proveedor de identidad, intercambian mensajes para permitir la firma única y un único log de acceso. Tal intercambio de mensajes puede ser iniciado por cualquiera de las dos entidades. Para la firma única, el proveedor de identidad se responsabiliza de crear y enviar al proveedor de servicio una "aserción" (assertion), que contiene la identidad del usuario. El proveedor de servicio, por su parte, se hace cargo de validar la aserción antes de permitirle acceder a la aplicación.

> Una aserción es un documento XML que contiene diversos elementos relativos a la identidad del usuario, como la forma en que el usuario ha sido autenticado y, opcionalmente, atributos sobre su identidad. El intercambio de tales mensajes puede producirse por medios diferentes, ya sea en forma HTTP vía el navegador o una interacción SOA.

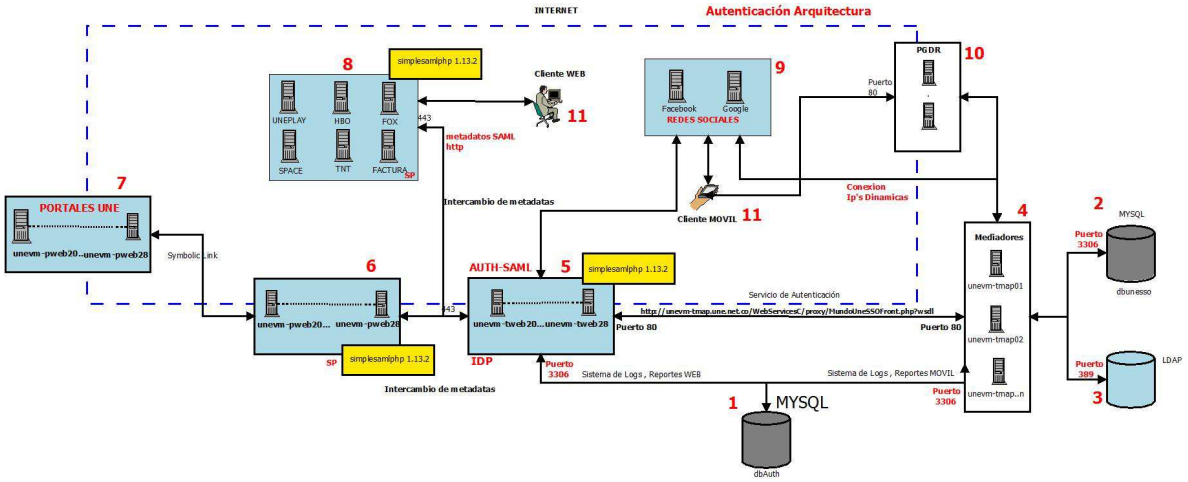
> La convergencia de las formas de uso que aporta nuestra autenticación federada tendrá un efecto fundamental sobre el interés de las empresas por el uso de la federación como medio de compartir información sobre identidades sin restricciones. Simplifica, además, la elección de la tecnología a adoptar y elimina la necesidad de soluciones multiprotocolo confusas, complejas y caras de mantener.

Este sería la secuencia para este proceso de autenticación



8. Arquitectura Física

El sistema de autenticación federada tiene diferentes elementos que están integrados por medio de protocolos de redes y sistemas distribuidos



Para este caso en particular tenemos diferentes elementos que para entendimiento del lector vamos a dar una breve descripción cada número representa el elemento en la *ilustración 1*

Base de datos Mysql: Esta base de datos se encarga de almacenar el sistema de logs y reportes para hacer el respectivo seguimiento a nuestro sistema de autenticación federado.

Base de datos Mysql: Almacena los registros de los usuarios

LDAP acceso al directorio donde está almacenado las contraseñas encriptados de los clientes.

Middleware (mediación): capa de abstracción de software distribuida, se encarga de hacer las peticiones al LDAP y Mysql

IDP: Se encarga de proveer identidades a proveedores de servicio además, es el cargado de exponer el login page, tiene toda la lógica del negocio, a través de peticiones SOAP se integrar con los sistemas de bajo nivel.

SP: Proveedor interno de servicios, se encarga por medio de enlaces simbólicos integrar todos los portales que están dentro de la red privada, para integrarlos con el IDP

Portales internos: Portales internos que requieren autenticación

Portales Externos: Portales externos que requieren autenticación

Redes Sociales: Integración con redes sociales

Proxy inverso: es un Servidor (un programa o sistema informático), que sirve de intermediario en las peticiones de recursos que realiza un cliente externo a otro servidor interno

Usuario: Dispara las peticiones en el sistema para ingresar a zonas web protegidas por contraseña

9. Base de datos

Está diseñada en un modelo relacional, las dos capas de diseño conceptual y lógico se parecen mucho. Generalmente se implementan mediante **diagramas de Entidad/Relación** (modelo conceptual) y **tablas y relaciones** entre éstas (modelo lógico). El gestor de datos que estamos implementando para este proceso es Mysql.

El modelo relacional de bases de datos se rige por algunas normas sencillas:

- Todos los datos se representan en forma de **tablas** (también llamadas “relaciones”). Incluso los resultados de consultar otras tablas. La tabla es además la unidad de almacenamiento principal

- Las tablas están compuestas por filas (o registros) y columnas (o campos) que almacenan cada uno de los registros (la información sobre una entidad concreta, considerados una unidad).

- Las filas y las columnas, en principio, carecen de orden a la hora de ser almacenadas. Aunque en la implementación del diseño físico de cada SGBD esto no suele ser así. Por ejemplo, en SQL Server si añadimos una clave de tipo "Clustered" a una tabla haremos que los datos se ordenen físicamente por el campo correspondiente.

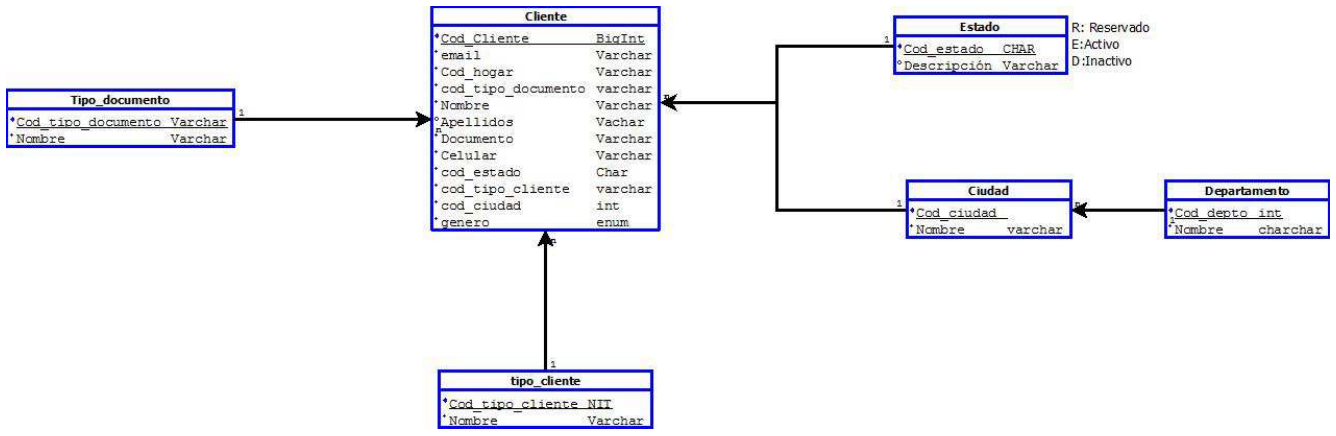
- El orden de las columnas lo determina cada consulta

- Cada tabla debe poseer una clave primaria, esto es, un identificador único de cada registro compuesto por una o más columnas.

- Para establecer una relación entre dos tablas es necesario incluir, en forma de columna, en una de ellas la clave primaria de la otra. A esta columna se le llama clave externa. Ambos conceptos de clave son extremadamente importantes en el diseño de bases de datos.

Basándose en estos principios se diseñan las diferentes bases de datos relacionales, definiendo un diseño conceptual y un diseño lógico, que luego se implementa en el diseño físico usando para ello el gestor de bases de datos de nuestra elección (por ejemplo SQL Server).

La base de datos refleja el proceso de registro, donde tenemos un modelo simple E/R , con este modelo almacenaremos el registro de los clientes para ser usados luego en el proceso de autenticación.



Todas las consultas a la base de datos serán ejecutada mediante protocolo SOAP (web services) esto con el fin de que otros agentes externos puedan hacer uso de los registros almacenados en esta base de datos.

9.1 Sentencia SQL

```
DROP DATABASE IF EXISTS `dbunesso`;

CREATE DATABASE IF NOT EXISTS `dbunesso`;

USE `dbunesso`;
```

Tabla cliente

```
DROP TABLE IF EXISTS `cliente`;

CREATE TABLE IF NOT EXISTS `cliente` (

  `cod_cliente` bigint(20) NOT NULL AUTO_INCREMENT,

  `email` varchar(70) NOT NULL,

  `codigo_hogar` varchar(20) DEFAULT NULL,

  `documento` varchar(20) NOT NULL,

  `cod_tipo_documento` varchar(3) NOT NULL,

  `nombres` varchar(45) NOT NULL,

  `apellidos` varchar(45) NOT NULL,

  `celular` varchar(15) DEFAULT NULL,

  `cod_estado` char(1) NOT NULL,
```

```
  `cod_ciudad` int(11) NOT NULL,
  `genero` enum('M','F') NOT NULL,


  PRIMARY KEY (`cod_cliente`),
  UNIQUE KEY `email_UNIQUE` (`email`),
  KEY `fk_cliente_estado1` (`cod_estado`),
  KEY `fk_cliente_tipo_documento1` (`cod_tipo_documento`),
  KEY `fk_cliente_ciudad1` (`cod_ciudad`),
  KEY `index_codigo_hogar` (`codigo_hogar`),
  KEY `index_email` (`email`),
  KEY `index_documento` (`documento`),
  KEY `fk_cliente_tipo_cliente1` (`cod_tipo_cliente`),
  CONSTRAINT `fk_cliente_ciudad1` FOREIGN KEY (`cod_ciudad`) REFERENCES `ciudad`
(`cod_ciudad`) ON UPDATE CASCADE,
  CONSTRAINT `fk_cliente_estado1` FOREIGN KEY (`cod_estado`) REFERENCES `estado`
(`cod_estado`) ON UPDATE CASCADE,
  CONSTRAINT `fk_cliente_tipo_cliente1` FOREIGN KEY (`cod_tipo_cliente`) REFERENCES
`tipo_cliente` (`cod_tipo_cliente`) ON DELETE NO ACTION ON UPDATE NO ACTION,
```

```
  CONSTRAINT `fk_cliente_tipo_documento1` FOREIGN KEY (`cod_tipo_documento`)
REFERENCES `tipo_documento` (`cod_tipo_documento`) ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=227353 DEFAULT CHARSET=latin1;
```

Tabla ciudad

```
DROP TABLE IF EXISTS `ciudad`;
CREATE TABLE IF NOT EXISTS `ciudad` (
  `cod_ciudad` int(11) NOT NULL AUTO_INCREMENT,
  `nombre` varchar(45) DEFAULT NULL,
  `cod_depto` int(11) NOT NULL,
  PRIMARY KEY (`cod_ciudad`),
  KEY `fk_ciudad_depto1` (`cod_depto`),
  CONSTRAINT `fk_ciudad_depto1` FOREIGN KEY (`cod_depto`) REFERENCES `depto`
(`cod_depto`) ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=99774 DEFAULT CHARSET=latin1;
```

Tabla Departamento

```
DROP TABLE IF EXISTS `depto`;
```

```
CREATE TABLE IF NOT EXISTS `depto` (
  `cod_depto` int(11) NOT NULL AUTO_INCREMENT,
  `nombre` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`cod_depto`)
) ENGINE=InnoDB AUTO_INCREMENT=1000 DEFAULT CHARSET=latin1;
```

Tabla estado

```
DROP TABLE IF EXISTS `estado`;
CREATE TABLE IF NOT EXISTS `estado` (
  `cod_estado` char(1) NOT NULL,
  `descripcion` varchar(45) NOT NULL,
  PRIMARY KEY (`cod_estado`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Tabla tipo cliente

```
DROP TABLE IF EXISTS `tipo_cliente`;

CREATE TABLE IF NOT EXISTS `tipo_cliente` (

  `cod_tipo_cliente` int(11) NOT NULL AUTO_INCREMENT,

  `nombre` varchar(45) NOT NULL,

  PRIMARY KEY (`cod_tipo_cliente`)

) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=latin1;
```

Tabla tipo document

```
DROP TABLE IF EXISTS `tipo_documento`;

CREATE TABLE IF NOT EXISTS `tipo_documento` (

  `cod_tipo_documento` varchar(3) NOT NULL,

  `nombre` varchar(45) NOT NULL,

  PRIMARY KEY (`cod_tipo_documento`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Este código permite tener una base de datos sencilla para la creación de usuarios, en la cual el IDP desarrollara consultas para que sirvan de información a los diferentes proveedores de servicio que actúen en el sistema.

10. Registro

El registro es diseñado por medio del modelo MVC en lenguaje PHP (framework YII) usando servicios web que integran la conexión con la base de datos. Para entender un poco el registro vamos a analizar sus diferentes componentes:

10.1 MVC

El portal de web para La gestión de cuentas está desarrollado por medio del modelo **MVC** (modelo, vista, controlador) en líneas generales, MVC es una propuesta de diseño de software utilizada para implementar sistemas donde se requiere el uso de interfaces de usuario. Surge de la necesidad de crear software más robusto con un ciclo de vida más adecuado, donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de conceptos.

Su fundamento es la separación del código en tres capas diferentes, acotadas por su responsabilidad, en lo que se llaman Modelos, Vistas y Controladores, o lo que es lo

mismo, Model, Views & Controllers, si lo prefieres en inglés. En este artículo estudiaremos con detalle estos conceptos, así como las ventajas de ponerlos en marcha cuando desarrollamos. MVC es un "invento" que ya tiene varias décadas y fue presentado incluso antes de la aparición de la Web. No obstante, en los últimos años ha ganado mucha fuerza y seguidores gracias a la aparición de numerosos frameworks de desarrollo web que utilizan el patrón MVC como modelo para la arquitectura de las aplicaciones web.

10.1.1 Porque MVC

La rama de la ingeniería del software se preocupa por crear procesos que aseguren calidad en los programas que se realizan y esa calidad atiende a diversos parámetros que son deseables para todo desarrollo, como la estructuración de los programas o reutilización del código, lo que debe influir positivamente en la facilidad de desarrollo y el mantenimiento.

Los ingenieros del software se dedican a estudiar de qué manera se pueden mejorar los procesos de creación de software y una de las soluciones a las que han llegado es la arquitectura basada en capas que separan el código en función de sus responsabilidades o conceptos. Por tanto, cuando estudiamos MVC lo primero que tenemos que saber es que está ahí para ayudarnos a crear aplicaciones con mayor calidad.

10.1.2 Ventajas de usar MVC

Aunque no tenga nada que ver, comencemos con algo tan sencillo como son el HTML y las CSS. Al principio, en el HTML se mezclaba tanto el contenido como la presentación. Es decir, en el propio HTML tenemos etiquetas como "font" que sirven para definir las características de una fuente, o atributos como "bgcolor" que definen el color de un fondo. El resultado es que tanto el contenido como la presentación estaban juntos y si algún día pretendíamos cambiar la forma con la que se mostraba una página, estábamos obligados a cambiar cada uno de los archivos HTML que componen una web, tocando todas y cada una de las etiquetas que hay en el documento. Con el tiempo se observó que eso no era práctico y se creó el lenguaje CSS, en el que se separó la responsabilidad de aplicar el formato de una web.

Al escribir programas en lenguajes como PHP, cualquiera de nosotros comienza mezclando tanto el código PHP como el código HTML (e incluso el Javascript) en el mismo archivo. Esto produce lo que se denomina el "Código Espagueti". Si algún día pretendemos cambiar el modo en cómo queremos que se muestre el contenido, estamos obligados a repasar todas y cada una de las páginas que tiene nuestro proyecto. Sería mucho más útil que el HTML estuviera separado del PHP.

Si queremos que en un equipo intervengan perfiles distintos de profesionales y trabajen de manera autónoma, como diseñadores o programadores, ambos tienen que tocar los mismos archivos y el diseñador se tiene necesariamente que relacionar con mucho código en un lenguaje de programación que puede no serle familiar, siendo que a éste quizás solo le interesan los bloques donde hay HTML. De nuevo, sería mucho más fácil la separación del código.

Durante la manipulación de datos en una aplicación es posible que estemos accediendo a los mismos datos en lugares distintos. Por ejemplo, podemos acceder a los datos de un artículo desde la página donde se muestra éste, la página donde se listan los artículos de un manual o la página de backend donde se administran los artículos de un sitio web. Si un día cambiamos los datos de los artículos (alteramos la tabla para añadir nuevos campos o cambiar los existentes porque las necesidades de nuestros artículos varían), estamos obligados a cambiar, página a página, todos los lugares donde se consumían datos de los artículos. Además, si tenemos el código de acceso a datos disperso por decenas de lugares, es posible que estemos repitiendo las mismas sentencias de acceso a esos datos y por tanto no estamos reutilizando código.

Quizás te hayas visto en alguna de esas situaciones en el pasado. Son solo son simples ejemplos, habiendo decenas de casos similares en los que resultaría útil aplicar una arquitectura como el MVC, con la que nos obliguemos a separar nuestro código atendiendo a sus responsabilidades.

Ahora que ya podemos tener una idea de las ventajas que nos puede aportar el MVC, analicemos las diversas partes o conceptos en los que debemos separar el código de nuestras aplicaciones.

10.1.3 Modelos

Es la capa donde se trabaja con los datos, por tanto contendrá mecanismos para acceder a la información y también para actualizar su estado. Los datos los tendremos habitualmente en una

base de datos, por lo que en los modelos tendremos todas las funciones que accederán a las tablas y harán los correspondientes selects, updates, inserts, etc.

No obstante, cabe mencionar que cuando se trabaja con MCV lo habitual también es utilizar otras librerías como PDO o algún ORM como Doctrine, que nos permiten trabajar con abstracción de bases de datos y persistencia en objetos. Por ello, en vez de usar directamente sentencias SQL, que suelen depender del motor de base de datos con el que se esté trabajando, se utiliza un dialecto de acceso a datos basado en clases y objetos.

10.1.4 Vistas

Las vistas, como su nombre nos hace entender, contienen el código de nuestra aplicación que va a producir la visualización de las interfaces de usuario, o sea, el código que nos permitirá renderizar los estados de nuestra aplicación en HTML. En las vistas nada más tenemos los códigos HTML y PHP que nos permite mostrar la salida.

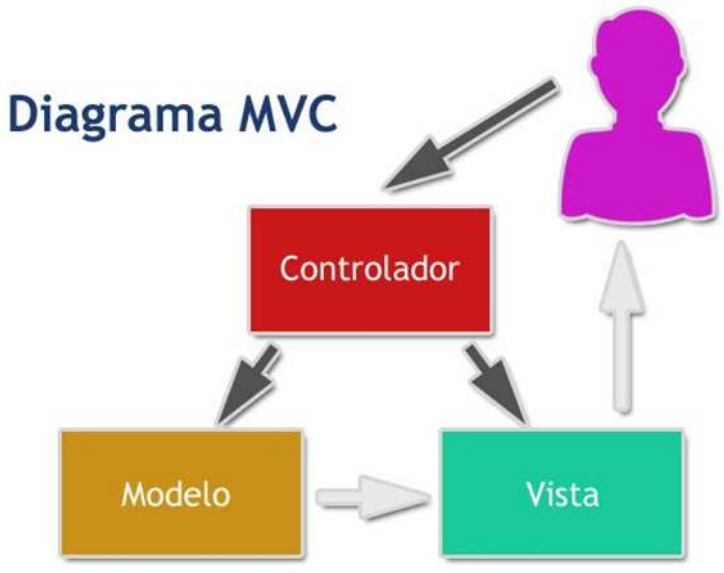
En la vista generalmente trabajamos con los datos, sin embargo, no se realiza un acceso directo a éstos. Las vistas requerirán los datos a los modelos y en ellas se generarán la salida, tal como nuestra aplicación requiera.

10.1.5 Controladores

Contiene el código necesario para responder a las acciones que se solicitan en la aplicación, como visualizar un elemento, realizar una compra, una búsqueda de información, etc.

En realidad es una capa que sirve de enlace entre las vistas y los modelos, respondiendo a los mecanismos que puedan requerirse para implementar las necesidades de nuestra aplicación. Sin embargo, su responsabilidad no es manipular directamente datos, ni mostrar ningún tipo de salida, sino servir de enlace entre los modelos y las vistas para implementar las diversas necesidades del desarrollo.

10.1.6 Arquitectura MVC



En esta imagen hemos representado con flechas los modos de colaboración entre los distintos elementos que formarían una aplicación MVC, junto con el usuario. Como se puede ver, los controladores, con su lógica de negocio, hacen de puente entre los modelos y las vistas. Pero además en algunos casos los modelos pueden enviar datos a las vistas. Veamos paso a paso cómo sería el flujo de trabajo característico en un esquema MVC.

1. El usuario realiza una solicitud a nuestro sitio web. Generalmente estará desencadenada por acceder a una página de nuestro sitio. Esa solicitud le llega al controlador.

2. El controlador comunica tanto con modelos como con vistas. A los modelos les solicita datos o les manda realizar actualizaciones de los datos. A las vistas les solicita la salida correspondiente, una vez se hayan realizado las operaciones pertinentes según la lógica del negocio.

3. Para producir la salida, en ocasiones las vistas pueden solicitar más información a los modelos. En ocasiones, el controlador será el responsable de solicitar todos los datos a los modelos y de enviarlos a las vistas, haciendo de puente entre unos y otros. Sería corriente tanto una cosa como la otra, todo depende de nuestra implementación; por eso esa flecha la hemos coloreado de otro color.

4. Las vistas envían al usuario la salida. Aunque en ocasiones esa salida puede ir de vuelta al controlador y sería éste el que hace el envío al cliente, por eso he puesto la flecha en otro color.

Hay un concepto que se usa mucho cuando se explica el MVC que es la "lógica de negocio". Es un conjunto de reglas que se siguen en el software para reaccionar ante distintas situaciones. En una aplicación el usuario se comunica con el sistema por medio de una interfaz, pero cuando acciona esa interfaz para realizar acciones con el programa, se ejecutan una serie de procesos que se conocen como la lógica del negocio. Este es un concepto de desarrollo de software en general.

La lógica del negocio, aparte de marcar un comportamiento cuando ocurren cosas dentro de un software, también tiene normas sobre lo que se puede hacer y lo que no se puede hacer. Eso también se conoce como reglas del negocio. Bien, pues en el MVC la lógica del negocio queda del lado de los modelos. Ellos son los que deben saber cómo operar en diversas situaciones y las cosas que pueden permitir que ocurran en el proceso de ejecución de una aplicación.

Por ejemplo, pensemos en un sistema que implementa usuarios. Los usuarios pueden realizar comentarios. Pues si en un modelo nos piden eliminar un usuario nosotros debemos borrar todos los comentarios que ha realizado ese usuario también. Eso es una responsabilidad del modelo y forma parte de lo que se llama la lógica del negocio.

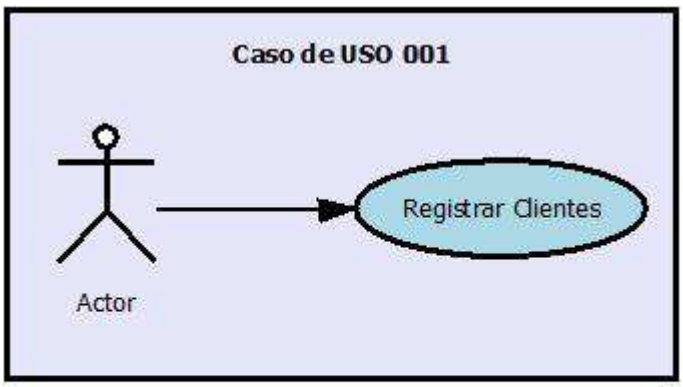
Incluso, en nuestra aplicación podría haber usuarios especiales, por ejemplo "administradores" y que no está permitido borrar, hasta que no les quitemos el rol de administrador. Eso también lo deberían controlar los modelos, realizando las comprobaciones necesarias antes de borrar efectivamente el usuario.

Sin embargo existe otro concepto que se usa en la terminología del MVC que es la "lógica de aplicación", que es algo que pertenece a los controladores. Por ejemplo, cuando me piden ver el resumen de datos de un usuario. Esa acción le llega al controlador, que tendrá que acceder al modelo del usuario para pedir sus datos. Luego llamará a la vista apropiada para poder mostrar esos datos del usuario. Si en el resumen del usuario queremos mostrar los artículos que ha publicado dentro de la aplicación, quizás el controlador tendrá que llamar al modelo de artículos, pedirle todos los publicados por ese usuario y con ese listado de artículos invocar a la vista correspondiente para mostrarlos. Todo ese conjunto de acciones que se realizan invocando métodos de los modelos y mandando datos a las vistas forman parte de la lógica de la aplicación.

El módulo de registro nos permite tener una interacción directa con los usuarios, por medio de un formulario el cliente ingresas los datos y estos serán enviados a través de un servicio web a la base de datos.

El registro está diseñado por medio del proceso Modelo, vista y controlador, donde el modelo se encarga de enviar los datos a la BD

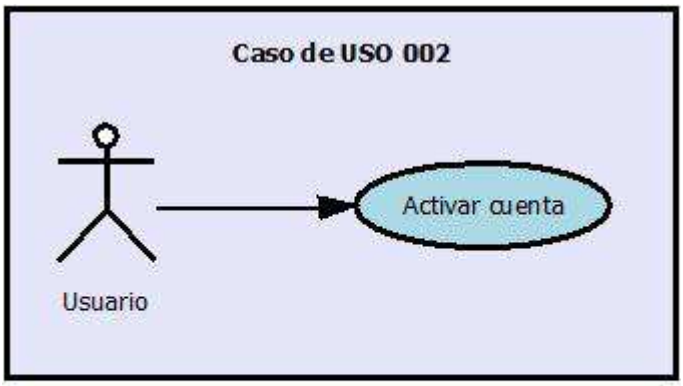
10.2 Casos de uso proceso de Registro



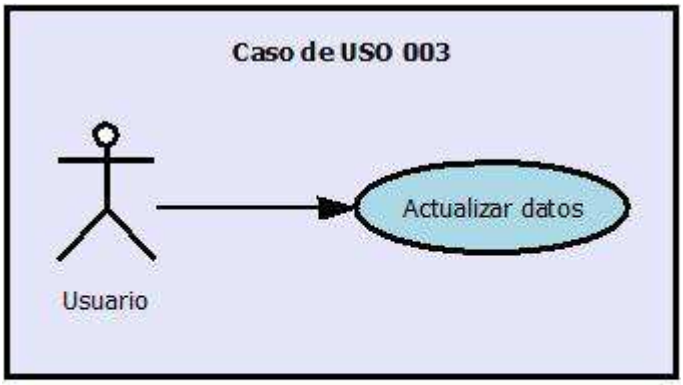
ID	Nombre	Descripción
001	Registrar cliente	1. El cliente ingresa a la página web 2. El cliente hace clic en registrar 3. El cliente ingresa su identificador de hogar 4. El sistema valida si el id del cliente está en el CRM 5. Si el id es válido , el cliente ingresa los datos de los campos requeridos 6. El sistema se encarga de enviarle un correo de

validación al cliente

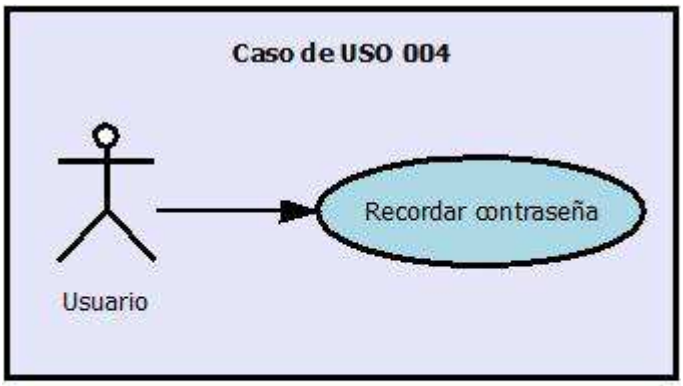
7. El cliente ingresa al correo para validar su cuenta
8. El registro completo
9. FIN



ID	Nombre	Descripción
002	Activar cuenta	1. El sistema se encarga de enviarle un correo de validación al cliente 2. El cliente ingresa al correo para validar su cuenta 3. El registro completo 4. FIN

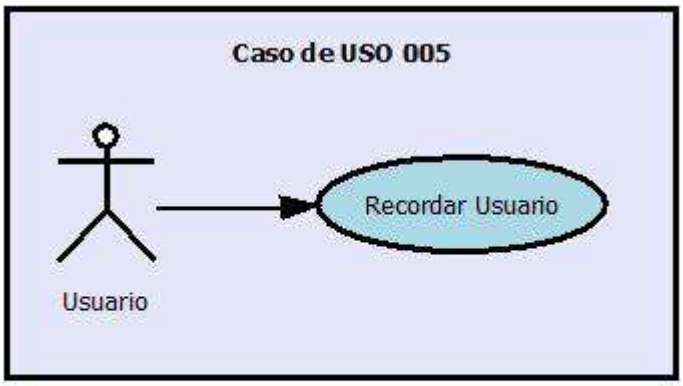


ID	Nombre	Descripción
003	Actualizar datos	1. El usuario ingresa al portal 2. El usuario se autentica 3. El usuario ingresa al modulo actualizar datos 4. El usuario cambia los datos que el sistema le permite 5. El usuario hace clic en actualizar 6. Fin



ID	Nombre	Descripción
004	Recordar contraseña	1. El usuario ingresa al portal 2. El usuario ingresa al modulo de autenticación 3. El usuario solicita recordar contraseña 4. El usuario ingresa su nombre de usuario 5. El sistema envía un correo de solicitud de cambio de contraseña 6. El usuario ingresa a su cuenta de correo hace clic en el enlace 7. El sistema muestra una vista para ingresar la nueva contraseña 8. El usuario ingresa su nueva contraseña 9. El usuario hace clic en actualizar

10. FIN



ID	Nombre	Descripción
005	Recordar usuario	1. El usuario ingresa al portal 2. El usuario ingresa al modulo de autenticación 3. El usuario solicita recordar usuario 4. El usuario ingresa su cedula 5. El sistema busca la cedula y asocia el correo registrado a esa cedula 6. El sistema envía un correo a la cuenta de correo

encontrada

7. FIN

ID	Nombre	Descripción
009	Validar Codigo hogar	1. El cliente ingresa a la pagina web 2. El cliente hace clic en registrar 3. El cliente ingresa su identificador de hogar 4. El sistema valida si el id del cliente está en el CRM 5. FIN

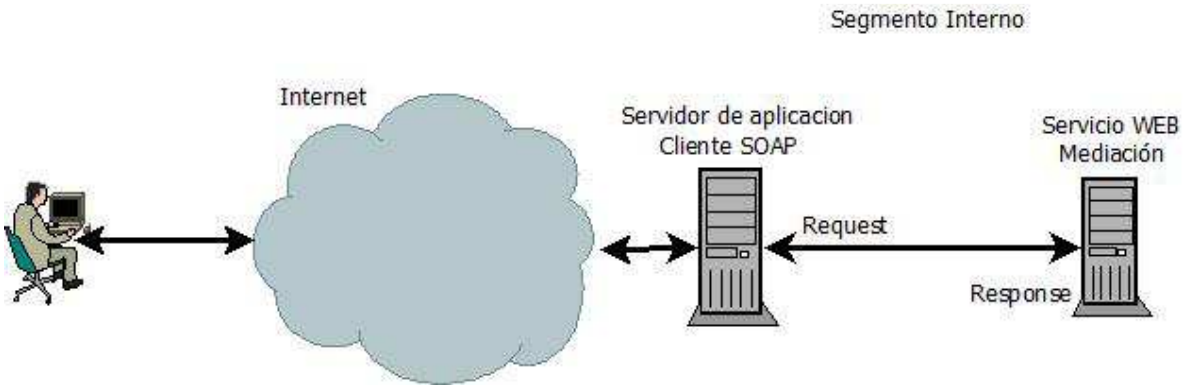
10.3 Servicio WEB

Es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet.
La interoperabilidad se consigue mediante la adopción de estándares abiertos.

Para nuestro caso desarrollamos dos servicios web

10.3.1 Servicio Web Front

Este servicio permite tener una interoperabilidad con sistemas de información internos, sus métodos solamente interactúan desde clientes de una red interna (no son expuestos en internet).



10.3.1.1 Los métodos creados para este Web services son:

Método	Descripción
AgregarCliente	Es consumido en caso de uso número 001, y permite agregar un cliente en la base de datos.
ConfirmarCuenta	Es consume en el caso de uso 002, permite validar la identidad de la cuenta de correo.
BuscarCliente	Se consume en el caso de uso 005 , permite consultar el correo del cliente en la base de datos
ResertPassword	Se consume en el caso de uso 004 , Permite al usuario cambiar su password en caso de olvido
ActualizarCliente	Se consume en el caso de uso 003 Permite al usuario cambiar algunos datos de su cuenta
CodigoHExiste	Se consume en el caso de uso 009 , permite validar si el ID del usuario existe en los CRMs' con el fin de verificar si es cliente
CodigoHRegistrado	Se consume en el caso de uso 009 , Valida si el Id no ha sido registrado con anterioridad
GetDatosCliente	Se consume en el caso de uso 003, Permite obtener los datos del cliente ya sea para temas de autenticación o actualización de datos, también es utilizado para obtener los datos del cliente cuando se

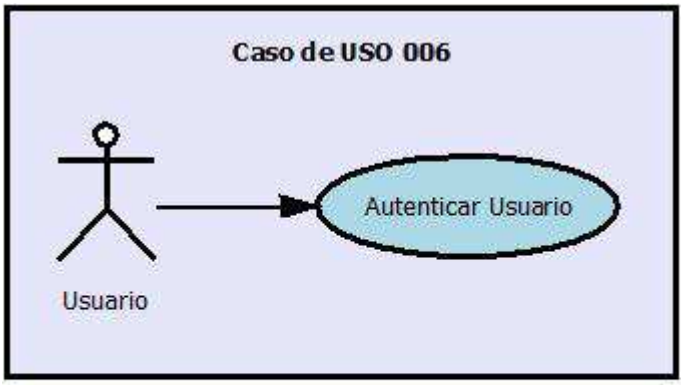
autentique con una red social

11. Autenticación

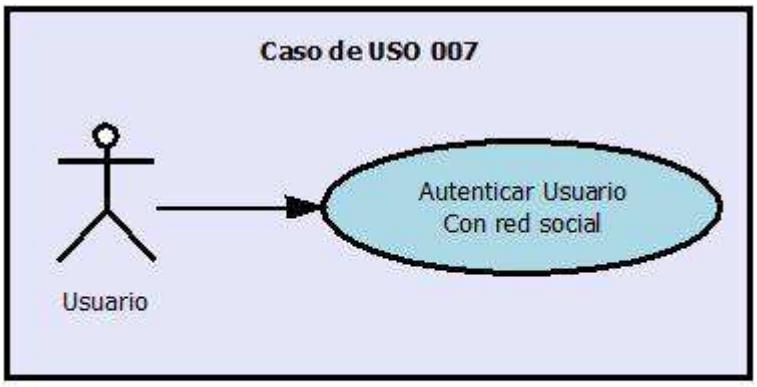
La Autenticación es el acto o proceso para el establecimiento o confirmación de algo (o alguien) como real. La autenticación de un objeto puede significar (pensar) la confirmación de su procedencia, mientras que la autenticación de una persona a menudo consiste en verificar su identidad.

Regístrate con nosotros

11.1 casos de uso



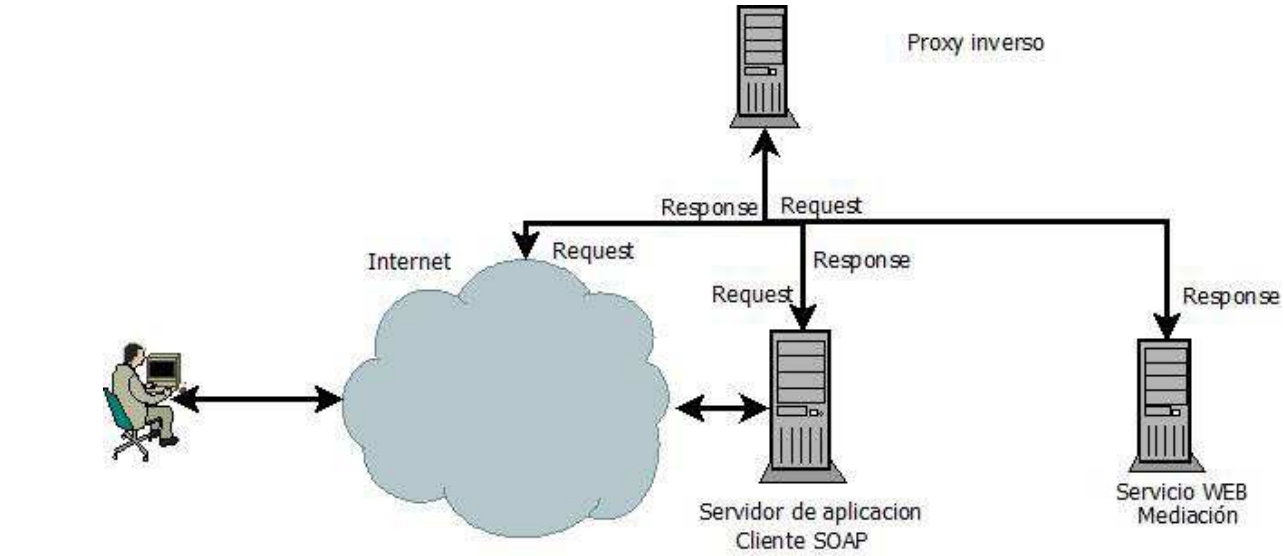
ID	Nombre	Descripción
005	Recordar usuario	1. El usuario ingresa al portal 2. El usuario ingresa al modulo de autenticación 3. El usuario ingresa su cuenta 4. El Usuario ingresa su password 5. El sistema valida en LDAP 6. El sistema buscar la información del cliente en la base de datos 7. El cliente queda autenticado 8. FIN



ID	Nombre	Descripción
007	Autenticar Usuario Red Social	1. El usuario ingresa al portal 2. El usuario ingresa al modulo de autenticación 3. El usuario elige autenticar con red social 4. El portal redirige al usuario a la red social 5. El usuario se autentica con la red social 6. La red social redirige al usuario al portal 7. El cliente queda autenticado 8. FIN

11.2 Servicio Web app:

Este servicio permite tener conectividad para clientes externos



11.2.1 Los métodos creados para este Web services son:

Método	Descripción
Validarcliente	Permite autenticar al usuario contra el LDAP y contra la base de datos de clientes

11.3 Estándares de Autenticación

Con la proliferación de servicios en Internet, especialmente los relacionados con redes sociales, el problema de la multitud de sistemas de autenticación de usuarios y de acceso a los recursos y datos personales se hace cada vez más importante. En este trabajo se describen y discuten los principales estándares y protocolos de autenticación y autorización en Internet utilizados en la actualidad, no tanto desde un punto de vista técnico de funcionamiento de los protocolos (su especificación y documentación relacionada está disponible en la red), sino desde la perspectiva de los usuarios finales de esos servicios.

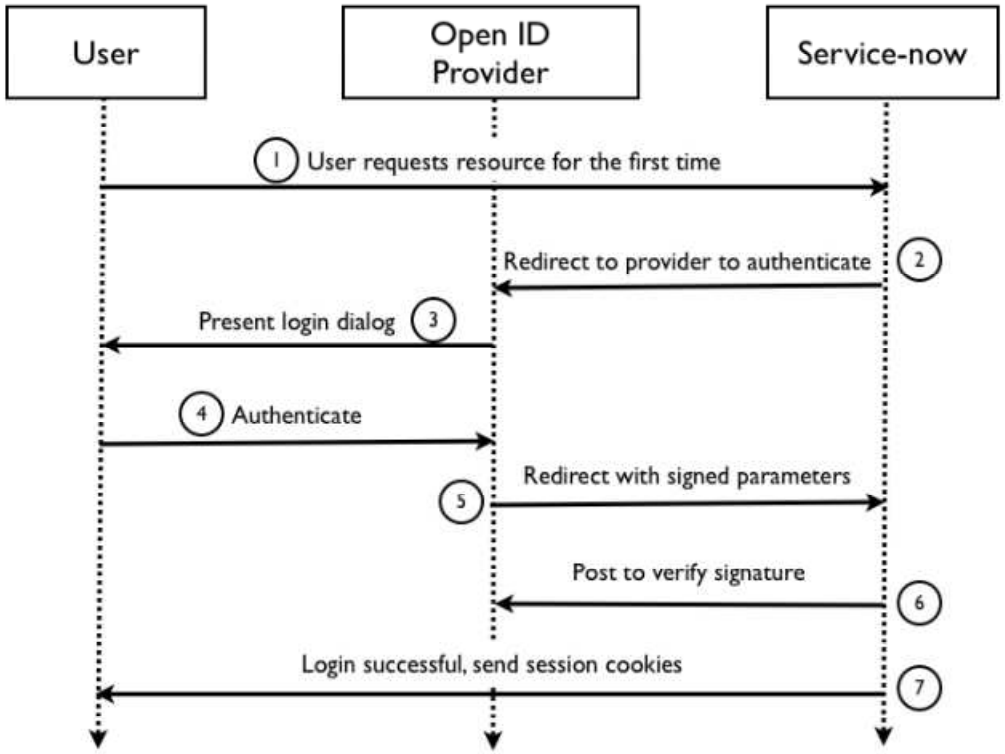
Para solucionar el problema de la autenticación múltiple encontramos una serie de protocolos que nos ayudan a tener una visión más clara para encontrar la solución.

11.3.1 OpenID

Protocolo de autenticación federada, y consiste básicamente en que el usuario selecciona un servidor externo (el "proveedor" de OpenID) que va a ser el que va a validar su identidad en un sistema determinado (el "consumidor" de OpenID).

El protocolo funciona así:

a. El usuario quiere acceder a su cuenta en un servidor.
b. Si ese servidor soporta el protocolo OpenID (es "consumidor" de OpenID), solicita al usuario su OpenID (la URL externa del "proveedor" de OpenID).
c. El usuario introduce o selecciona su OpenID
d. El servidor redirige al usuario al proveedor de OpenID.
e. El usuario se autentica contra el proveedor de OpenID.
f. El proveedor de OpenID redirige al usuario de vuelta al servidor, validando su identidad.



Utilizando siempre un mismo proveedor de OpenID, el usuario sólo necesita recordar su identificador OpenID y una única contraseña para autenticarse, con una misma identidad, en todos los servicios que acepten OpenID como sistema de autenticación.

11.3.1.1 Problemas de OpenID

Se han realizado múltiples críticas al protocolo OpenID , que se resumen básicamente en los siguientes aspectos:

● Complejidad: no se implementa fácilmente. OpenID ha recibido críticas por parte de los desarrolladores, que lo consideran excesivamente complejo de implementar.

● Seguridad: es un protocolo muy vulnerable al phishing. El phishing [12] consiste en la suplantación maliciosa de una página de autenticación (login) con el objetivo de conseguir el identificador y la contraseña de un usuario. Habitualmente los sitios maliciosos tienen que engañar al usuario para que lleguen a la página suplantada (usualmente mediante enlaces en correos electrónicos); en un servidor que utilice OpenID ese acceso es más sencillo de conseguir: basta con ofrecer al usuario la autenticación mediante OpenID para acceder a una aplicación determinada (que puede ser falsa), y después redirigir al usuario a una página que suplanta la página del proveedor de OpenID para conseguir su contraseña.

● Privacidad: los proveedores de OpenID tendrán mucha información del usuario. Si los usuarios utilizan un proveedor de OpenID para autenticarse en múltiples servicios, ese proveedor obtendrá potencialmente muchísima información de la actividad de los usuarios en la red, lo que lleva a posibles problemas de privacidad.

● Confianza: ¿quién es realmente el usuario? El protocolo OpenID ofrece un mismo sistema de autenticación para diferentes servidores, pero no ofrece ninguna garantía de la identidad real del usuario. Nada impide, por ejemplo, que procesos de spam creen y validen identidades automáticamente mediante OpenID para tener acceso a determinadas aplicaciones.

- Usabilidad: puede ser incómodo y/o complejo para el usuario. El proceso de elegir el proveedor de OpenID y autenticarse en un servidor diferente al de la aplicación puede resultar confuso y/o complejo para los usuarios.

- Adopción: los proveedores de servicios tienen pocos motivos para aceptarlo como autenticación. El número de proveedores de servicios que aceptan OpenID como mecanismo de autenticación (es decir, que son consumidores) es relativamente bajo en comparación con el de proveedores de OpenID. El motivo es que a las organizaciones y empresas les resulta beneficioso tener las cuentas (los datos) de los usuarios en su servicio; por tanto, siempre estarán más interesados en que los usuarios utilicen las cuentas en su servicio para autenticarse en otros, que en utilizar la autenticación de proveedores externos para su propio servicio (con lo que dejarían de ser los gestores de los datos de los usuarios).

11.3.2 Oauth

El protocolo abierto OAuth, a diferencia de OpenID, es un protocolo de autorización; más exactamente, de delegación de acceso; es decir, permite definir cómo un tercero va a acceder a los recursos propios. Empezó a definirse en 2006 ante las carencias del protocolo OpenID, y en 2007 se publicó la primera versión oficial.

El propósito de este protocolo es, pues, que un usuario que tiene determinados recursos en un servidor (el “proveedor” de OAuth) pueda dar acceso a un tercero (el “consumidor”, usualmente

un sitio web) a parte o todos esos recursos, sin necesidad de que ese tercero conozca su usuario y contraseña, ya que con esos datos tendría el control total de la cuenta.

a. El usuario dispone de una serie de recursos propios en un servidor (el “proveedor”).

b. Un servidor externo (el “consumidor”) desea acceder a un subconjunto de esos recursos.

c. El consumidor redirige al usuario hacia el proveedor.

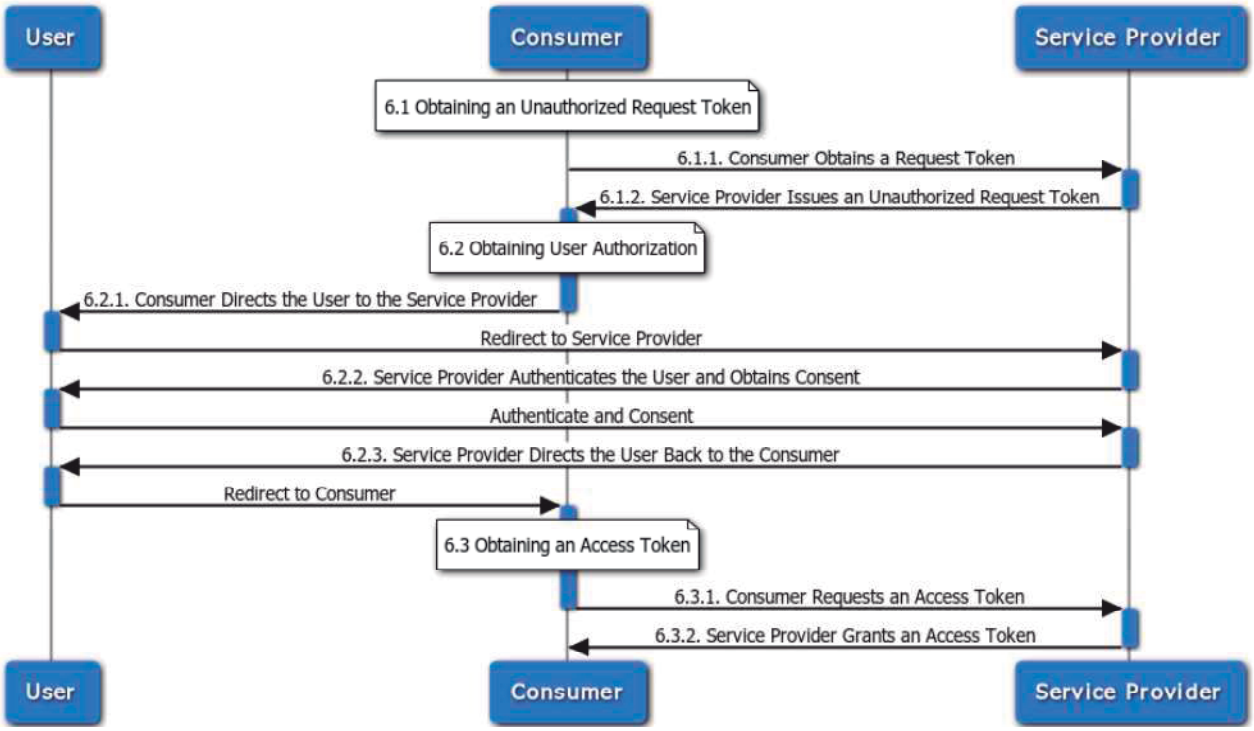
d. El usuario se autentica en el proveedor (si no lo estaba previamente)

e. El proveedor pregunta al usuario si autoriza al consumidor a que utilice esos determinados recursos.

f. El usuario autoriza al consumidor a utilizar esos recursos.

g. El servidor externo (consumidor) consigue acceso a esos recursos.

Ilustración 9 Oauth



11.3.2.1 Problemas de Oauth

Estas son algunas de las críticas que se han realizado al protocolo:

- **Complejidad**: El protocolo ha sido definido como "una solución poco elegante a un problema complejo"; eso hace que existan diversas implementaciones y librerías que realizan implementaciones poco robustas y poco documentadas
- **Orientado a navegadores**. El protocolo está especialmente orientado a su uso en navegadores de Internet, de modo que su uso en otro tipo de aplicaciones (de escritorio,

móviles, embebidas, etc.) resulta problemático, por ejemplo, por el uso de redirecciones entre URLs.

- **Seguridad** En parte como consecuencia de los puntos anteriores, se han detectado algunos problemas de seguridad en el protocolo.

11.3.3 Oauth 2.0

Para solucionar parte de esos problemas, durante algún tiempo se trabajó en definir una versión de OAuth más sencilla, llamada OAuth WRAP [17], que fue abandonada en favor de la nueva versión 2.0 de OAuth.

Pretende ser una versión revisada y simplificada de OAuth, y a pesar de estar todavía en versión borrador, ya ha sido oficialmente adoptada por grandes compañías como Facebook, Twitter, Yahoo!, Google o Microsoft. Aventaja a la versión anterior en una mayor simplicidad de implementación, y en una arquitectura más robusta y que da soporte a mayor número de plataformas.

11.3.3.1 ¿Es posible autenticación de usuarios con OAuth?

Existe cierta confusión sobre si OAuth es o no un protocolo de autenticación de usuario. Estrictamente hablando no lo es, ya que no define ningún mecanismo explícito destinado a autenticar la identidad del usuario.

Sin embargo, uno de los pasos de este protocolo de autorización es la autenticación del usuario final en el servidor proveedor de OAuth para poder dar autorización a otros a acceder a tus recursos, antes tienes que autenticar quién eres tú realmente.

Por tanto, cuando se habla del “mecanismo de autenticación de OAuth”, en realidad se están refiriendo al mecanismo de autenticación propio del sitio web proveedor de OAuth, que puede ser cualquiera: autenticación http básica, OpenID, etc.

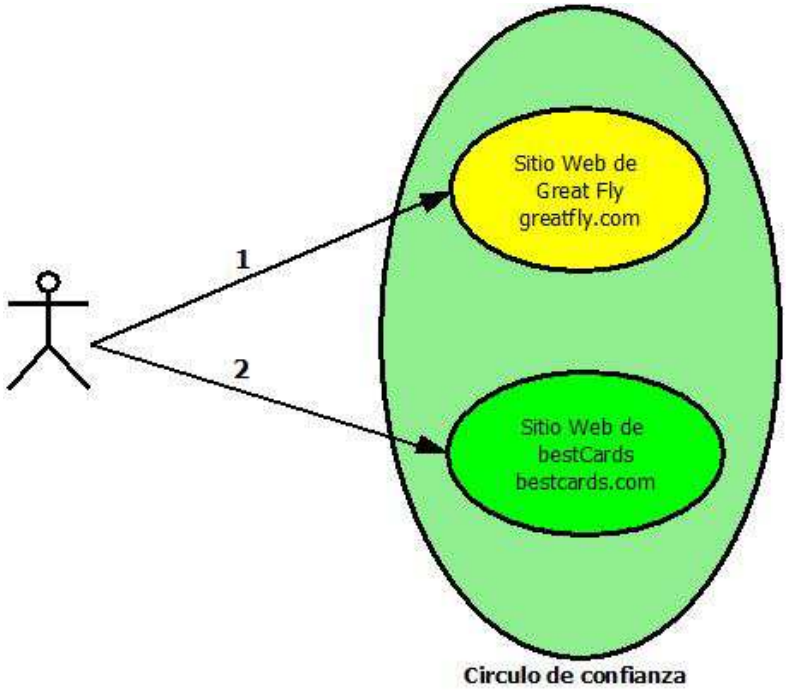
11.4 SAML

|Este es el esquema que vamos a usar para solucionar nuestro inconveniente de autenticación, SAML fue desarrollado tomando como base el problema del Single Sign-On entre aplicaciones web. La solución que especifica SAML define la existencia de dos componentes lógicos: un generador de información de seguridad y un consumidor, también son llamados IDP (identity provider) y SP (service provider) respectivamente.

Para comprender un poco más la problemática vamos a describir a continuación los dos casos de uso principales que fueron la base para el desarrollo de SAML:

Single Sign-On

Web y Federación de Identidad.



En este caso el usuario ingresa al sitio web de la empresa great Fly y se loguea. Luego en forma transparente o no, el usuario es dirigido hacia el sitio web de BESTCARS. En este caso se asume que ya existe una federación de identidades para el usuario (es decir existe una asociación entre la identidades locales de forma que los sistemas puedan referirse al usuario de forma unívoca). El IDP (greatfly.com) envía junto con la solicitud, información adicional (por ejemplo información sobre la identidad del usuario) de forma que el SP (bestcars.com) pueda autorizar al usuario a acceder al recurso solicitado sin que se requiera volver a autenticarlo.

Una variante de la situación descripta anteriormente se produce cuando el usuario no se encuentra autenticado en el IDP y comienza la interacción desde el SP. En este caso el usuario ingresa a bestcars.com y como no está autenticado se lo redirige al sitio greatfly.com junto con una solicitud de autenticación de SAML. En el IDP el usuario es

Autenticado y se lo redirige nuevamente a bestcars.com junto con la información de Autenticación correspondiente.

11.4.1 Federación de identidad

Veamos ahora utilizando como base el ejemplo anterior como se logra la federación de identidad.

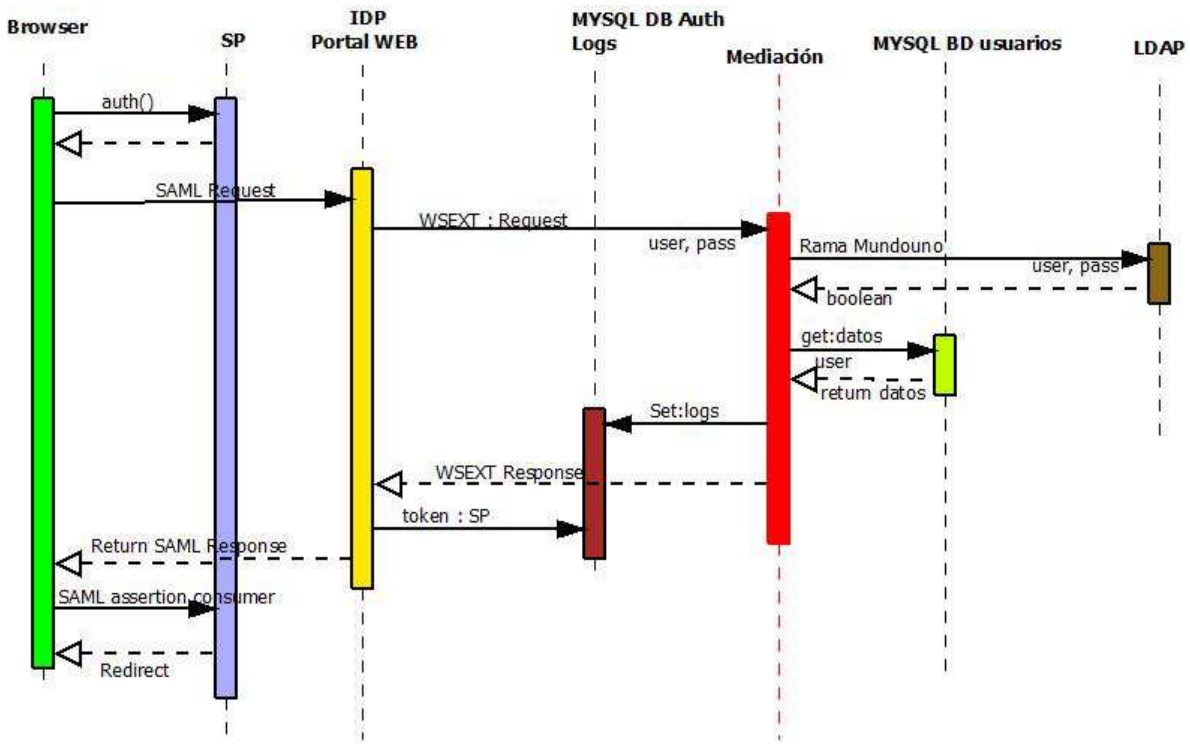
1. El usuario estando en el IDP tiene la posibilidad de federar su identidad local con la del SP.
2. El usuario es redirigido al IDP.
3. El usuario se autentica con su cuenta local y el IDP genera un identificador (seudónimo como se lo denomina en el estándar) que será utilizado únicamente por el ID y el SP para referirse un forma unívoca al usuario.
4. El usuario es redirigido al IDP junto con el seudónimo
5. El SP obtiene el seudónimo y busca el usuario correspondiente. Dado que es une nueva federación, este todavía no existe.
6. El usuario se autentica en el SP y se asocia el usuario local con el seudónimo.

Una vez completada la interacción se tiene una forma segura de relacionar los usuarios entre los sistemas.

EL IDP tiene la capacidad de hacer consultas a diferentes sistemas de información por medio de:

- Sentencias SQL
- Consultas LDAP
- Peticiones SOAP (este es nuestro método de consulta)
- Interface Radios

11.4.2 Diagrama de secuencia autenticación

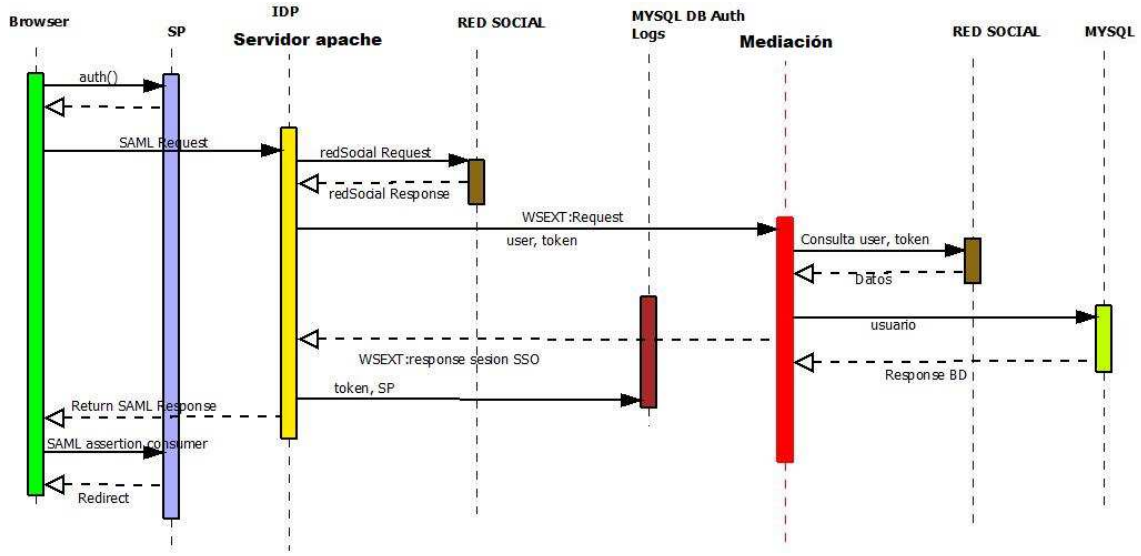


Para nuestro caso específico el diagrama de secuencia expuesto en la ilustración 11 nos ofrece una mirada más detallada del proceso SAML

1. El cliente (browser) hace una petición de autenticación en el service provider.
2. SP le responde al browser para que este genere un SAML request al proveedor de identidad.
3. En el IDP (proveedor de identidad) está el modelo de negocio y el login page, el (IDP) es el responsable de entregar un SAML response al SP, esta consulta se hace por medio de sentencias SOAP (para nuestro caso), El IDP hace una petición SOAP (request) a la medicación.
4. El IDP genera un request SOAP consumiendo el método **Validarcliente**
5. El mediador valida los datos en LDAP
6. Ldap responde y la mediación completa la información del cliente que se encuantra en la base de datos
7. El mediador guarda los logs del proceso
8. El mediador genera el response al IDP
9. El IDP genera un assertion y se lo envía al SP

11.4.3 Diagrama de secuencia para la autenticación con una Red Social

Nuestro sistema de autenticación permitirá a los usuarios autenticarse con las redes sociales de facebook y google, esto permite facilidad y amigabilidad al usuario con el manejo de contraseñas



1. El cliente (browser) hace una petición de autenticación en el service provider.
2. SP le responde al browser para que este genere un SAML request al proveedor de identidad.
3. En el IDP (proveedor de identidad) está el modelo de negocio y el login page, el (IDP) es el responsable de entregar un SAML response al SP, esta consulta se hace por medio de sentencias SOAP (para nuestro caso), El IDP hace una petición SOAP (request) a la medicación.
4. El IDP hace un request a la red social
5. La red social genera un response con los datos básicos del cliente al IDP

6. El IDP genera un request SOAP consumiendo el método **GetDatosCliente**
7. La mediación completa la información del cliente que se encuentra en la base de datos
8. El mediador guarda los logs del proceso
9. El mediador genera el response al IDP
10. El IDP genera un assertion y se lo envía al SP

11.4.4 Login page

Este sería nuestro diseño html del login page, es un diseño adaptativo a todas las resoluciones en la cuales se ejecute, este diseño debe estar expuesto dentro del IDP.

12. Simplesamlphp

Simplesamlphp es un framework open source desarrollado en php que permite integrar el esquema federado de autenticación SAML, desarrollado por UNINETT. Sin embargo también soporta algunos otros protocolos de identidad como Shibboleth 1.3, A-Select, CAS, OpenID, WS-Federation o OAuth por lo cual una de sus mayores ventajas es la capacidad para incorporar deferentes esquemas.

El framework soporta memcached esto quiere decir que puede trabajar con sistemas distribuidos de propósito general para caché basado en memoria, lo que lo hace más rápido en tiempo de ejecución.

Para empezar la implementación de nuestro sistema de autenticación en el nube , debemos instalar simplesamlphp en nuestro servidor pero antes de eso debemos cumplir con estos requisitos no funcionales:

- Some webserver capable of executing PHP scripts.
- PHP versión >= 5.3.0.
- Support for the following PHP extensions:
    - Siempre requerido date, dom, hash, libxml, openssl, pcre, SPL, zlib, mcrypt
    - Cuando exista autenticación contra LDAP: ldap

- Cuando exista autenticacion contra servicor radius: radius
- Almacenamiento de sessiones: memcache
- Base de datos:mysql

12.1 Instalación simplesamlphp

La instalación de SimpleSAMLphp es realmente sencilla, solamente es necesario descargar el código fuente en un directorio del sistema, por ejemplo en el directorio /var/www/sp/

```
yum install subversion
mkdir /var/www/sp
svn co http://simplesamlphp.googlecode.com/svn/branches/simplesamlphp-1.10 simplesamlphp
/var/www/sp/simplesamlphp
```

12.1.1 Estructura general

Cert: Directorio donde alojar el certificado y la clave que serán usados para firmar/cifrar las aserciones

config: Directorio con los ficheros de configuración de simpleSAMLphp

config-templates: Directorio con plantillas de los ficheros de configuración

metadata: Directorio donde alojar ficheros con los metadatos de los IdPs remotos en los que se confiará.

metadata-templates: Directorio con plantillas de los ficheros de metadatos

Modules: Directorio con los diferentes módulos de simpleSAMLphp

lib: Directorio con las librerias de simpleSAMLphp

www: Directorio con la lógica web de simpleSAMLphp

templates: Directorio con las plantillas web de simpleSAMLphp

dictionaries: Directorio con las traducciones de simpleSAMLphp

docs: Directorio con documentación

log: Directorio donde alojar los logs de simpleSAMLphp (Requiere configurar el config.php para que se guarden aquí.

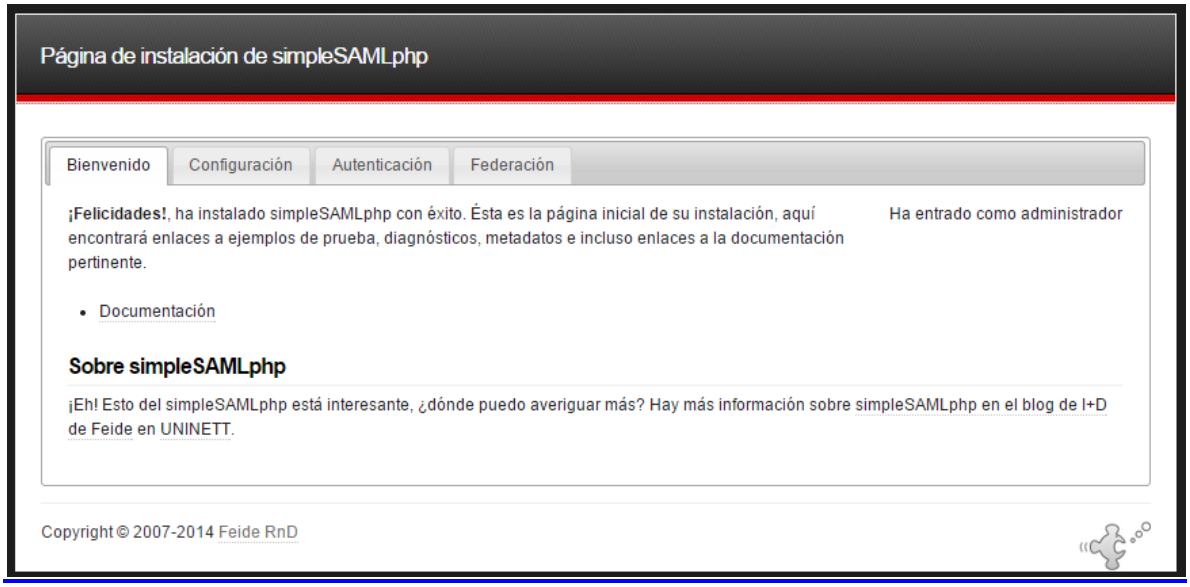
Attributemap: Directorio con ficheros con lógica para realizar el mapeo de atributos

Schemas: Directorio con esquemas xsd.

Bin: Directorio con herramientas de scripts

Hay que asegurarse que el servidor Apache tenga acceso de escritura a los directorios logs y metadata

12.1.2 Configuración general



Los ficheros de configuración de simpleSAMLphp se localizan dentro del directorio config donde se haya instalado simpleSAMLphp. En el directorio config-templates se encuentran plantillas de los principales ficheros de configuración que se pueden necesitar en simpleSAMLphp. Es bastante útil partir de una de estas plantillas y modificarla a conveniencia en lugar de crear el fichero de configuración correspondiente desde cero, lo cual suele ser mucho más propenso a errores.

El primer fichero a modificar es el fichero config.php. Por tanto, lo mejor es copiar su plantilla correspondiente

```
php -l /var/www/sp/simplesamlphp/config/config.php
# No syntax errors detected in /var/www/sp/simplesamlphp/config/config.php
```

A continuación se detallan las partes más habituales del fichero config.php que hay que modificar:

Contraseña de administración de simpleSAMLphp

```
'auth.adminpassword' => 'admin',
```

Es importante cambiar esta contraseña porque además de ser un problema de seguridad si se deja tal cual, simpleSAMLphp lo detectará y dará un error.

Sal aleatoria y secreta:

```
'secretsalt' => 'defaultsecretsalt'
```

Para generar este valor se puede utilizar el siguiente comando:

```
tr -c -d '0123456789abcdefghijklmnopqrstuvwxyz' </dev/urandom
    | dd bs=32 count=1 2>/dev/null;echo
1zpz5ztl1w5hnqhx8969thxab68us5og
```

Nombre y dirección de correo del administrador de este simpleSAMLphp:

```
'technicalcontact_name'  => 'Administrator',
```

```
'technicalcontact_email' => 'na@example.org',
```

Zona horaria:

```
'timezone' => NULL,
```

Un valor de ejemplo para este parámetro es ‘Europe/Madrid’

Idioma predeterminado:

```
'language.default' => 'es',
```

Lo más habitual es poner el valor ‘es’ para que la interfaz de simpleSAMLphp salga en español.

Hay muchas más opciones de configuración pero estás son las más básicas y necesarias de modificar. Los comentarios en el fichero config.php explican bastante bien para qué sirve cada uno de los parámetros de configuración.

Si la instalación de simplesamlphp fue exitosa, podemos ingresar a un pequeño portal de gestión, donde tendremos la posibilidad de:

- Ver la configuración del servidor y que cumpla con las necesidades del framework.
- Ver los módulos de autenticación previamente desarrollados y configurados
- Ver la metadatas de configuración.

12.2 Metadatos

Tanto el SP como el idP necesitan conocer acerca del otro para poder comunicarse entre ellos. La manera que el SP conoce acerca del iDP y viceversa es a través del intercambio de metadata, el cual consiste en la información básica de cada entidad.

La gestión de los metadatos constituye una tarea esencial para el funcionamiento de las federaciones de identidades basadas en el protocolo SAML. Es imprescindible proporcionar metadatos confiables, disponibles y precisos a los participantes de la federación: la descripción de las entidades, sus responsables y sobre todo las url donde ofrecen las funcionalidades y las claves públicas.

Dentro de los metadatos se incluyen 2 atributos opcionales que hay que tener muy en cuenta a la hora de realizar la gestión de los metadatos:

- **validUntil**: Indica en tiempo absoluto cuando caducan los metadatos
- **cacheDuration**: Indica el máximo periodo de tiempo en el que las entidades deberían de confiar en esos metadatos y albergarlos en su "cache" (en simpleSAMLphp la cache está basada en ficheros de sistema que se alojan dentro de la carpeta metadata).

También es importante ver como se llegó al consenso para que los entityIDs de las entidades coincidieran con las urls donde se alojan sus metadatos, lo cual facilita el trabajo de la gestión de los metadatos.

12. 2.1 Modelo de gestión de metadatas

12.2.1.1 Modelo de agregación

La arquitectura de agregación permite actualizar los metadatos en el lado de la entidad que los va a consumir y debido al simple modelo de confianza, los metadatos serán considerados válidos siempre que estos sean descargados de una fuente fiable (URL pre-acordada y los metadatos convenientemente firmados con una clave pública del publicador)

12.2.1.2 Modelo distribuido

En la arquitectura distribuida cada entidad es responsable de la gestión y publicación de sus propios metadatos. Con este modelo los metadatos de un determinado punto pueden ser obtenidos bajo demanda sin la necesidad de la descarga periódica de una colección de gran tamaño de metadatos. (Se realiza el descubrimiento dinámico de metadatos en función de las

necesidades). Es un modelo que puede acarrear problemas ya que se relaja el concepto de "confianza".

12.2.1.3 Modelo centralizado

Es una arquitectura que viene muy bien en federaciones Hub & Spoke, en el nodo bridge o en un nuevo nodo, se despliega un sistema que será responsable de almacenar, monitorizar y verificar los metadatos de las entidades de una federación.

Los metadatos de la federación son agregados periódicamente, almacenados y publicados como un archivo XML convenientemente firmado.

El sistema deberá de disponer de mecanismos para poder delegar las responsabilidades de gestión de un determinado metadato al administrador de la entidad a la que describe.

12.2.2 Configurar los metadatos en simplesamlphp

En el fichero config/config se especifica con el parámetro 'metadata.sources' donde simpleSAMLphp va a buscar los ficheros que contienen las fuentes de los diferentes metadatos.

Para el caso que nuestro SAML trabaje como un idp se deben configurar la metadatas del sp que requiere autenticación, para ello el sp debe exponer una serie de urls que se deben agregar en:

Auth/metadatas/saml20-sp-remote.php

Para el caso que nuestro SAML trabaje como un sp se deben configurar la metadatas del idp que requiere autenticación, para ello el idp debe exponer una serie de urls que se deben agregar en:

portalExterno/metadatas/saml20-idp-remote.php

ejemplo de metadatas de un **IDP**

```
$metadata['http://example.com.co/www/saml2/idp/metadata.php'] = array (
  'metadata-set' => 'saml20-idp-remote',
  'entityid' => 'http:// example.com.co/www/saml2/idp/metadata.php',
  'SingleSignOnService' =>
  array (
    0 =>
    array (
      'Binding' => 'urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect',
      'Location' => 'http://example.com.co/www/saml2/idp/SSOService.php',
    ),
  ),
  'SingleLogoutService' =>
array (
    0 =>
    array (
```

'Binding' => 'urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect',

'Location' => 'http://example.com.co/www/saml2/idp/SingleLogoutService.php',

),

),

'certFingerprint' => '85:12:A5:F5:D7:8D:82:A1:DA:61:29:90:EC:EB:B1:A6:B4:D5:A1:D4',

'NameIDFormat' => 'urn:oasis:names:tc:SAML:2.0:nameid-format:transient',

Ejemplo de metadatas de un **SP**

$metadata['http://example.com.co/saml/www/module.php/saml/sp/metadata.php/mundoune-sp'] = array (

'AssertionConsumerService' => 'http://example.com.co /saml/www/module.php/saml/sp/saml2-acs.php/mundoune-sp',

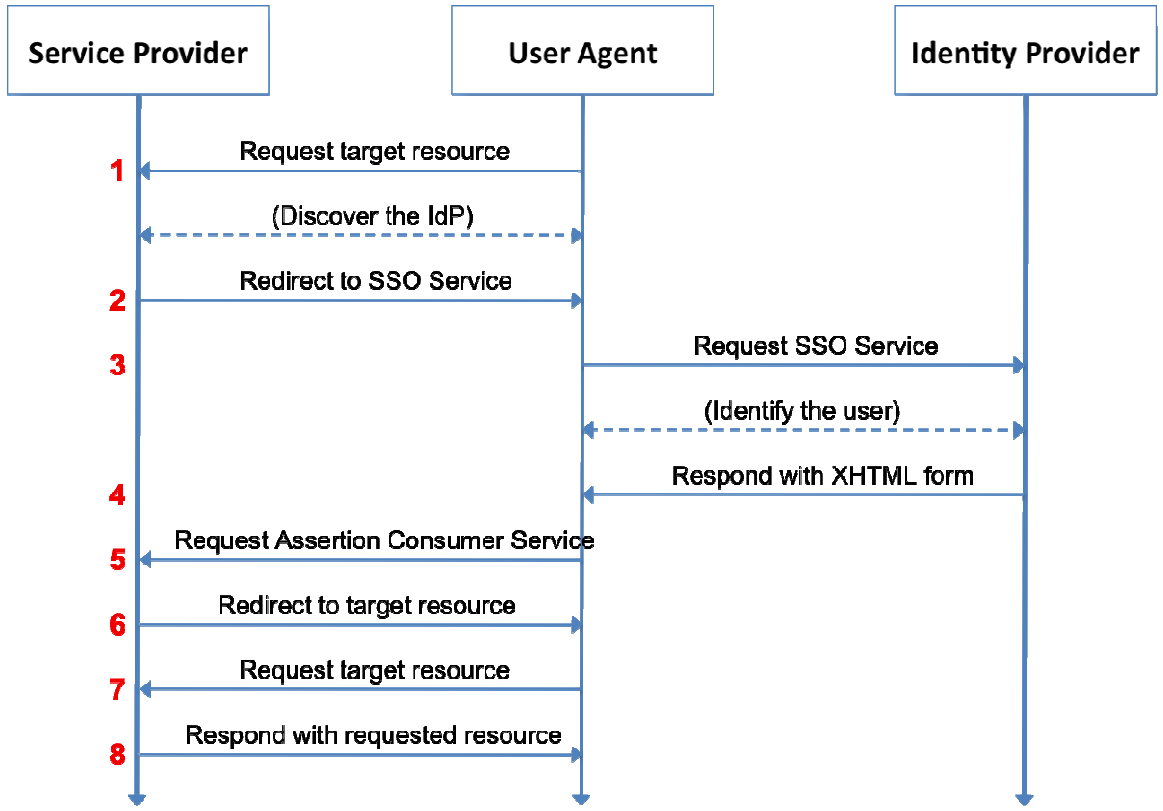
'SingleLogoutService' => 'http:// example.com.co.com.co/saml/www/module.php/saml/sp/saml2-logout.php/mundoune-sp',

);

Recordemos que SAML tiene dos importantes actores dentro del proceso de autenticación

IDP : Identity provider , encargado de proveer identidades

SP :service provider, encargado de ofrecer servicios



El SP tiene una zona protegida para que solo algunos usuarios puedan ver, por lo tanto le solicita al IDP la identidad del usuario para permitirle ingresar a la zona restringida

Simplesamlphp tiene la capacidad de trabajar ya sea como un SP o un IDP.

Vamos a configurar el sistema para que trabaje como un IDP (luego miraremos su funcionamiento como SP)

12.3 Configuración IDP

Para configurar simpleSAMLphp como IdP es necesario habilitar el parámetro ‘enable.saml20-idp’ en el fichero de configuración config/config.php:

```
'enable.saml20-idp' => true,
```

12.3.1 Configuración de fuentes de autenticación

La configuración de una fuente de autenticación en SimpleSAMLphp se realiza en el fichero **config/authsources.php**. Para ello es necesario seguir dos pasos

1. Habilitar los módulos de autenticación que necesitemos, como ejemplo, habilitaremos el módulo ‘exampleauth’:

   ```
   # touch modules/exampleauth/enable
   ```

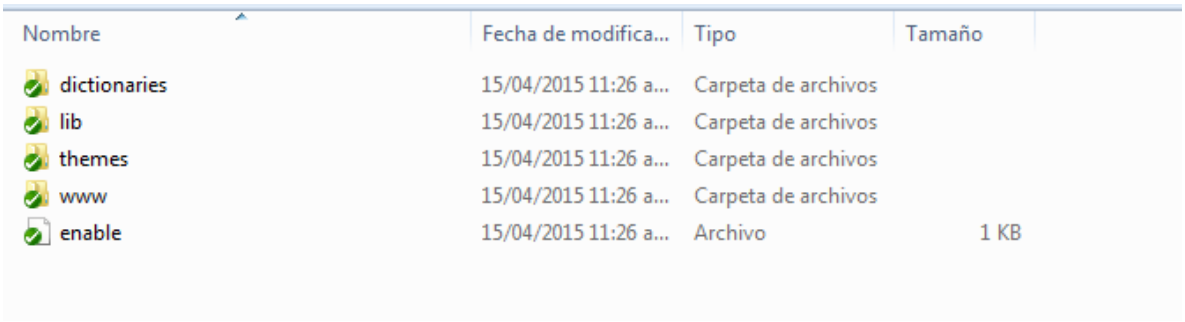
2. Se deberá editar el fichero config/authsources.php, añadiendo fuentes de autenticación. En este ejemplo vemos configuradas fuentes sencillas (admin, exampleauth):

```
<?php
    $config = array(
        'admin' => array(

                'core:AdminPassword',

```

```
9.          ),
10.         'example-userpass' => array(
11.             'exampleauth:UserPass',
12.                 'usuario1:usuario1.' => array(
13.                     'uid' => array('usuario1'),
14.                     'eduPersonAffiliation' => array('member', 'student'),
15.                 ),
16.                 'usuario2:usuario2.' => array(
17.                     'uid' => array('usuario2'),
18.                     'eduPersonAffiliation' => array('member', 'employee'),
19.                 ),
20.         ),
21.     );
22. ?>
```

Las fuentes de autenticación son la forma cómo vamos a autenticar al usuario, podemos usar un módulo por defecto o crear uno nuevo , para nuestro caso creamos un nuevo módulo llamado **auth**

En la vamos a crear un nuevo fichero llamado auth dentro de la carpeta modules con la siguiente estructura.



Donde tenemos una archivo vacio llamado enable, este permite al framework identificar que este modulo está disponible para uso de la autenticación,

Dictionaries: Esta carpeta nos permite controlar los mensajes de la autenticación

Themes: vistas del login page

www: repositorio de imágenes , estilos y scripts

Lib: tiene toda la lógica de la autenticación en php

Dentro de la carpeta lib tenemos el archivo llamado UneAuth.php que tiene toda la lógica para validar los clientes

Vamos a ver la función validar cliente

```
private function ValidarCliente($username, $password) {

        $configSAML = SimpleSAML_Configuration::getInstance();
        $WS_AUTH = new AuthSSO();
        $WS_FRONT = new FrontSSO();

        $res_status = $WS_FRONT->ValidarEstadoCliente($username);
        if($res_status['estado'] == 'E'){
```

```
if($WS_AUTH->getEstado() == true) {
        $res_auth = $WS_AUTH->ValidarCliente($username,$password);
        if($res_auth['res']==true){
        $datosCliente = $res_auth['datosCliente'];
        $r = $WS_FRONT->setIpCliente($datosCliente->token);
            switch($datosCliente->tipoCliente){

             case 1: // HOGARES tipo de cliente
                  return $this->GetAssertionHogares($datosCliente);
             break;

             case 2: // PYMES tipo de cliente
                  return $this ->GetAssertionPymes($datosCliente);
             break;

              default:
                  //GENERAMOS LOG DE LA TRANSACCION
                  $this->email = $redsocial;
                  $this->acceso = 0;
                  $this->sendLog();

                        // CERRAMOS SESION
        $r = $WS_FRONT->cerrarSesion($datosCliente->token);
```

```
                    throw new SimpleSAML_Error_Error('NOACCESS');
                    break;
                        }
                    }else{
                        //GENERAMOS LOG DE LA TRANSACCION
                        $this->email = $redsocial;
                        $this->acceso = 0;
                        $this->sendLog();
                switch($res_auth['tipo']){
                    case 'WRONGUSERCAPTCHA':
                        throw new
SimpleSAML_Error_Error('WRONGUSERCAPTCHA');
                    break;
                    case 'MULTISESSIONEXCED':
                        throw new
SimpleSAML_Error_Error('MULTISESSIONEXCED');                    break;
                    default:
                        throw new SimpleSAML_Error_Error('WRONGUSERPASS');
                    break;
                        }
                    }
```

```
                }else{
                        //GENERAMOS LOG DE LA TRANSACCION

                        $this->email = $username;

                        $this->acceso = 0;

                        $this->sendLog();

                        throw new SimpleSAML_Error_Error('NOHABIL');

                }

                }else{
                //GENERAMOS LOG DE LA TRANSACCION
                $this->email = $username;

                $this->acceso = 0;

                $this->sendLog();

                throw new SimpleSAML_Error_Error('NOHABIL');

        }

}
```

Esta función básicamente nos permite instanciar algunos objetivos para ser consultados vía web services

12.4 Configuración SP

Debemos crear una nueva instancia de SAML solamente es necesario descargar el código fuente en un directorio del sistema, por ejemplo en el directorio /var/www/sp/

```
yum install subversion
mkdir /var/www/sp
svn co http://simplesamlphp.googlecode.com/svn/branches/simplesamlphp-1.10 simplesamlphp
/var/www/sp/simplesamlphp
```

12.4.1 Estructura general

Cert: Directorio donde alojar el certificado y la clave que serán usados para firmar/cifrar las aserciones

config: Directorio con los ficheros de configuración de simpleSAMLphp

config-templates: Directorio con plantillas de los ficheros de configuración

metadata: Directorio donde alojar ficheros con los metadatos de los IdPs remotos en los que se confiará.

metadata-templates: Directorio con plantillas de los ficheros de metadatos

Modules: Directorio con los diferentes módulos de simpleSAMLphp

lib: Directorio con las librerias de simpleSAMLphp

www: Directorio con la lógica web de simpleSAMLphp

templates: Directorio con las plantillas web de simpleSAMLphp

dictionaries: Directorio con las traducciones de simpleSAMLphp

docs: Directorio con documentación

log: Directorio donde alojar los logs de simpleSAMLphp (Requiere configurar el config.php para que se guarden aquí.

Attributemap: Directorio con ficheros con lógica para realizar el mapeo de atributos

Schemas: Directorio con esquemas xsd.

Bin: Directorio con herramientas de scripts

Hay que asegurarse que el servidor Apache tenga acceso de escritura a los directorios logs y metadata

Esta instancia del SAML es la encargada de generar la petición para autenitcar a un usuario, dentro del framework no es necesario colocar alguna lógica de consulta, solo se debe configurar algunos aspectos relevantes para que tengamos nuestro proveedor de servicio.

Diferencia del IdP en el SP los metadatos propios se configuran en el fichero `config/authsources.php`

```
<?php

   $config = array(
      'saml' => array(
         'saml:SP',
         'host' => 'sp.example.com',
         'entityID' =>
'https://sp.example.com/simplesaml/module.php/saml/sp/metadata.php/saml',
         'idp' => 'https://idp.example.com/simplesaml/saml2/idp/metadata.php',
         'certificate' => 'sp.example.com.crt',
         'privatekey' => 'sp.example.com.key',

         // All communications are encrypted and signed
```

```
        'redirect.sign' => TRUE,
        'redirect.validate' => TRUE,
        'assertion.encryption' => TRUE,



        'OgnizationName' => array (
            'en' => 'Example organization',
            'es' => 'Organizacion de ejemplo',
        ),
        'OrganizationDisplayName' =>  array (
            'en' => 'Example organization',
            'es' => 'Organizacion de ejemplo',
        ),
        'OrganizationURL' => array (
            'en' => 'http://sp.example.com',
            'es' => 'http://sp.example.com',
        ),


    ),
  );
?>
```

A continuación se detallan los principales parámetros que hay que configurar:

host

Nombre del host de esta máquina que se utiliza en el VirtualHost de Apache correspondiente

entityId

Identificador de identidad. Debe corresponderse con la URL que obtiene los metadatos de este SP

idp

El IdP que debe utilizar este SP para autenticarse. Debe aparecer el entityId de un nodo IdP (o WAYF).

certificate y **privatekey**

Son las dos partes (pública y privada) del certificado usado en el SP. Estos ficheros deben encontrarse en el directorio cert del directorio raíz de simpleSAMLphp a no ser que se haya cambiado esa ubicación en el fichero config.php

redirect.sign

TRUE Si la petición de autenticación, petición de log out y respuesta de log out enviadas desde este SP a cualquier IdP deben de estar firmadas. Por defecto False.

redirect.validate

TRUE si las peticiones de autenticación, petición de logout o la respuesta de logout recibidas en este SP deben de ser validadas. Por defecto False

assertion.encryption

Este parámetro debe estar a TRUE si obligamos a que las aserciones recibidas en este SP provenientes de cualquier IdP estén cifradas. Por defecto False.

11.4.2 Configuración de los metadatos de los IdPs

Los metadatos de los IdPs en los que el SP va a confiar deben de definirse dentro de la carpeta metadata en el fichero saml20-idp-remote.php. A continuación se presenta un ejemplo

```
<?php

$metadata['https://idp.example.com/simplesaml/saml2/idp/metadata.php'] = array (
    'metadata-set' => 'saml20-idp-remote',
    'entityid' => 'https://idp.example.com/simplesaml/saml2/idp/metadata.php',
    'SingleSignOnService' => 'https://idp.example.com/simplesaml/saml2/idp/SSOService.php',
    'SingleLogoutService' =>
'https://idp.example.com/simplesaml/saml2/idp/SingleLogoutService.php',
    'certData' =>
'MIICQTCCAaoCCQCOMxLGB244ZzANBgkqhkiG9w0BAQUFADBlMQswCQYDVQQGEw
JFUzESMBAGA1UECBMJQW5kYWx1Y2lhMRAwDgYDVQQHEwdTZXZpbGxhMRYwFA
YDVQQKEw1ZYWNvIFN

pc3RlbWFzMRgwFgYDVQQDEw9pZHAuZXhhbXBsZS5jb20wHhcNMTEwNDE4MDg1ND
```

Q2WhcNMTYwNDE2MDg1NDQ2WjBlMQswCQYDVQQGEwJFUzESMBAGA1UECBMJQW5kYWx1Y2lhMRAwDg

YDVQQHEwdTZXZpbGxhMRYwFAYDVQQKEw1ZYWNvIFNpc3RlbWFzMRgwFgYDVQQDEw9pZHAuZXhhbXBsZS5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAK9gW0M87vOu4zhYb

zoSgahBggvVPhmKROi0ajlAfVQ1YPwjpDstkK19Dhi99diw+ipJ6q44AEanpc8x/h0f1r2FJfBlF1HYKA76vMe36/A05G6+lg7Si4tHIlEfEG1RsuxK5FE9cLOdPTa9haX8HhtRngcN

XbFtvbyngJTvZt91AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAYdYYFxKibC32nm4E9RZCVk74Xcx6LahnbDvCQCMUL+zNeWetfq6hSlvX2AVl9yY5D5+fnL1hfaukmcJNTDGU+R+80bI

s9zXNPgyweSujcP54IVBH0qBSFJCmPdZc+0uasi4KClwOMjVIq3WpVvycA/sHl7kH1jYTtNea8rr1vSw=',

'NameIDFormat' => 'urn:oasis:names:tc:SAML:2.0:nameid-format:transient',

'OrganizationName' =>

array (

'en' => 'Example organization',

'es' => 'Organizacion de ejemplo',

),

'OrganizationDisplayName' =>

array (

```
        'en' => 'Example organization',
        'es' => 'Organizacion de ejemplo',
    ),
    'OrganizationURL' =>
      array (
        'en' => 'http://idp.example.com',
        'es' => 'http://idp.example.com',
    ),
);


?>
```

12.4.2 Script de llamado al IDP

Para llamar al IDP se requiere un pequeño script el cual instancia un objeto de la clase SAML con el recurso de autenticación que se desarrollo previamente

```
require_once('./lib/_autoload.php');
$as = new SimpleSAML_Auth_Simple('AUTENTICACION');
$as->requireAuth();
/* $as->requireAuth(array(
    'ReturnTo' => 'http://unemas.une.com.co/index.php?sso=1',
```

Atributos

Identificador de usuario uid	cmesahenao@gmail.com
Nombre para mostrar displayName	Cesar Augusto
displayLastName	Mesa Henao
subscriber_id	*503331857
Cedula	8161036
country_code	CO
Cod_hog	*503331857
celular	3004860898
token_sso	f5fda06f702254752bc69cdda7ebb78f
Telefono	

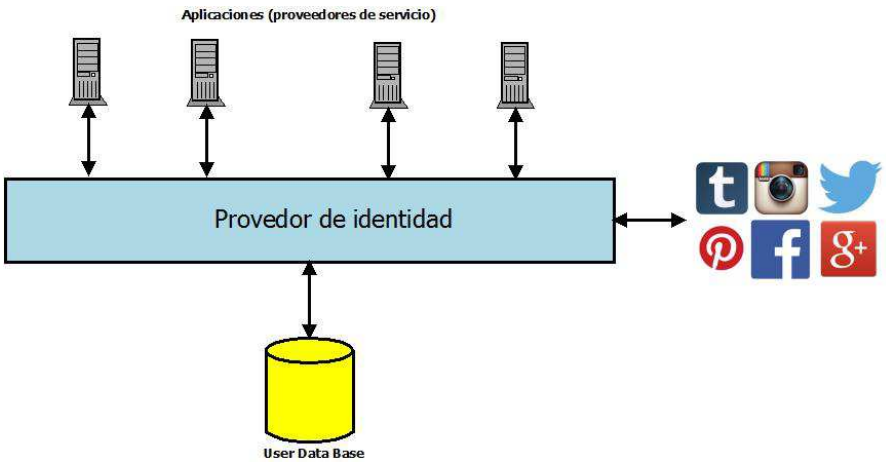
El SP puede hacer uso de estos datos para ofrecerle al usuario diferentes productos y/o servicios

12.5 Assertions

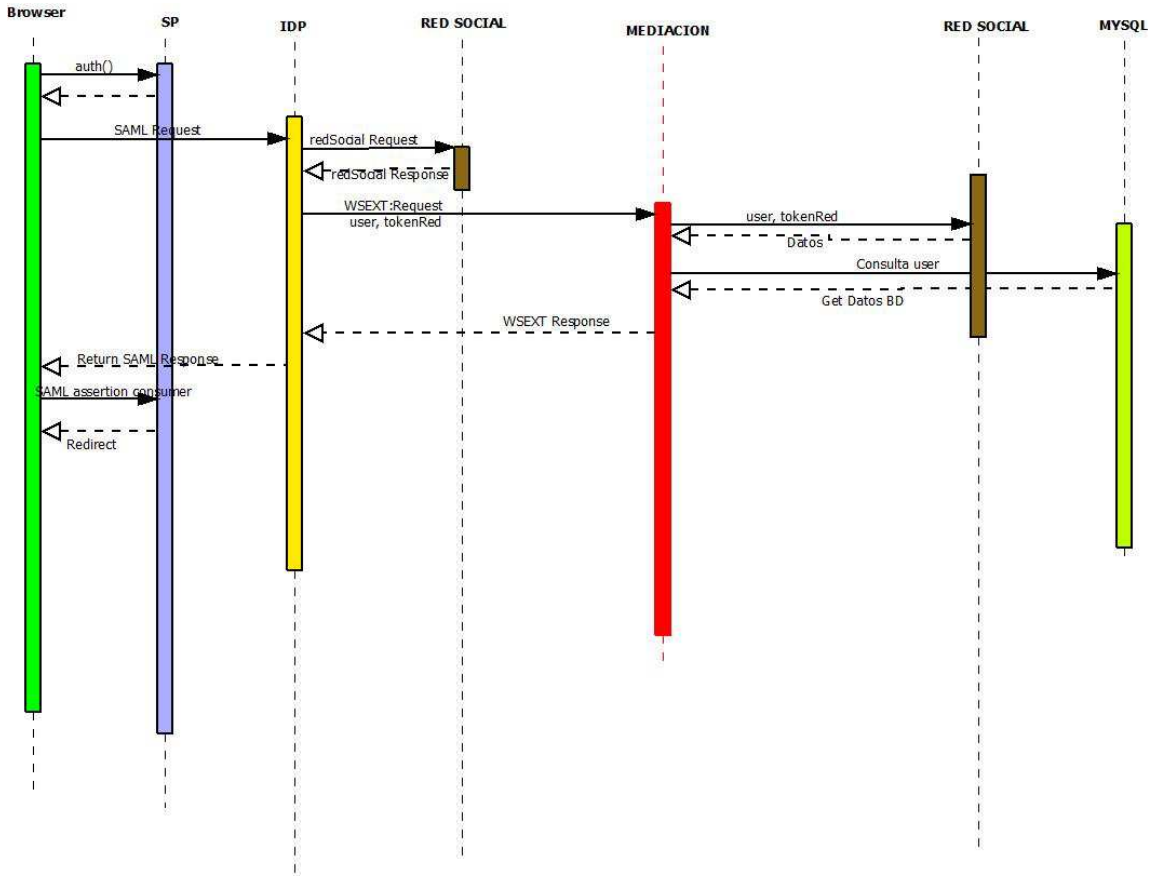
Una assertion es una respuesta SAML que envía el IDP al SP para que este tome la decisión correspondiente respecto al usuario que se autentica, esta respuesta es configurada y desarrollada dentro del IDP el cual dependiendo de su consulta va almacenando dentro un array los parámetros más relevantes del usuario autenticado. Desde la cónsula SAML podemos ver un ejemplo de assertion

13. Integración con redes sociales

El esquema de autenticación tiene la posibilidad dentro del loginpage incluir las redes sociales más representativas, dentro de nuestro alcance vamos a integrar facebook y google, ambas redes sociales utiliza un esquema de autenticación basada en **Oauth** la cual puede ser integrado dentro del framework simplesamlphp, para ello ambas redes exponen una api incluyendo el **key** y el **secret** propios de la autenticación Oauth



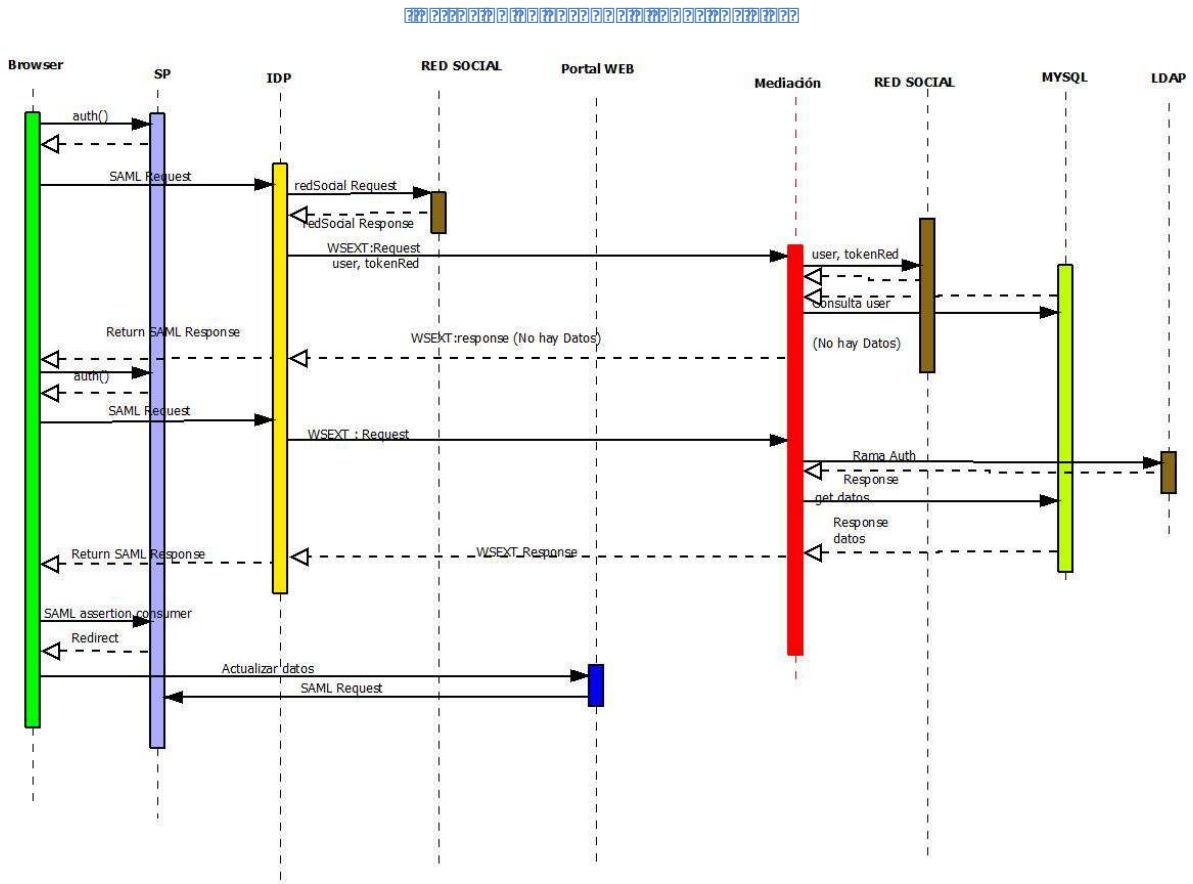
12.1 Proceso de autenticación con redes sociales



1. El cliente ingresa a un portal WEB y hace una petición de autenticación , el browser llama al **SP**
2. El **SP** hace el Request para que el browser llame al **IDP**
3. El cliente elije la red social y el **IDP** hace el request
4. La red social responde con los datos del cliente
5. Tomamos de los datos el correo asociado a la redsocial y lo consultamos en la base de datos
6. Si encontramos el usuario en la base de datos creamos la sesión
7. El **IDP** responde al browser con el assertion (datos del cliente en array)

8. El browser envía el assertion al **SP**
9. El SP redirige al cliente a sector protegido

12.2 Proceso de autenticación con redes sociales cliente NO identificado



1. El cliente ingresa a un portal WEB y hace una petición de autenticación , el browser llama al **SP**
2. El **SP** hace el Request para que el browser llame al **IDP**
3. El cliente elije la red social y el **IDP** hace el request
4. La red social responde con los datos del cliente

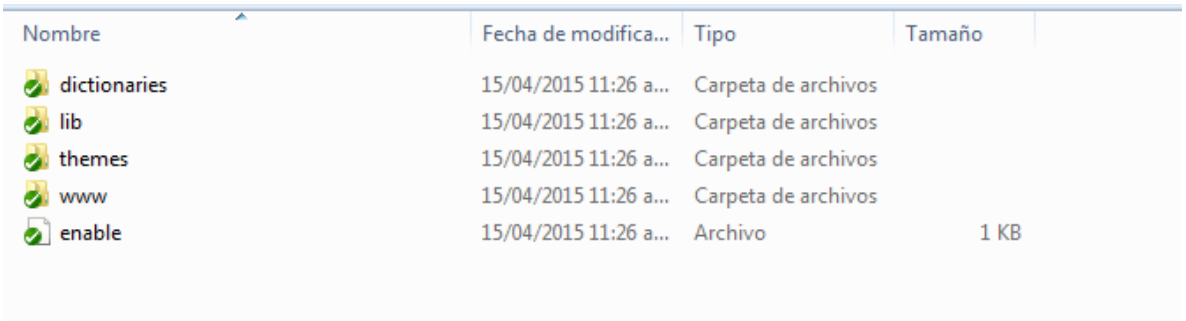
5. Tomamos de los datos el correo asociado a la redsocial y lo consultamos en la base de datos
6. Si encontramos el usuario no está en la base de datos , el servicio responde false
7. El **IDP** responde al browser enviando al usuario autenticarse por la vía normal
8. El usuario se autentica en el IDP (primera secuencia del sistema de autenticación)
9. El usuario registra su red social
10. El sistema queda habilitado para que el usuario se autentique por la red social

12.3 Configuración y desarrollo de los módulos de autenticación

Para autenticar contra una red social, debemos configurar dentro de las Apis de Facebook y Google nuestro dominio (el cliente que solicita la autenticación), ellos generan datos que deben ser configurados dentro del authsource de simplesamlphp

```
// AUTENTICACION VIA FACEBOOK
    'facebook' => array(
        'authfacebook:Facebook',
        'api_key' => '153269985798347569914 71',
        'secret' => 'f4e333aaojeiobafe5a6e41645e2af27df672',
        'req_perms' => 'email,user_birthday',
    ),

// Autenticación GOOGLE
  'google' => array(
        'authgoogle:Google',
        'key' =>'139919257606-mj6e151mfekc8t3t5nshuvqr2at5roos0oik.apps.googleusercontent.
        'secret' => 'w23SDsgA9LW3_FuñemMkOeK0zCU',
),
```

Se debe crear un modulo de autenticación por red social, con la misma estructura de un modulo de autenticación normal



Donde tenemos una archivo vacio llamado enable, este permite al framework identificar que este modulo está disponible para uso de la autenticación,

Dictionaries: Esta carpeta nos permite controlar los mensajes de la autenticación

Themes: vistas del login page

www: repositorio de imágenes , estilos y scripts

Lib: tiene toda la lógica de la autenticación en php

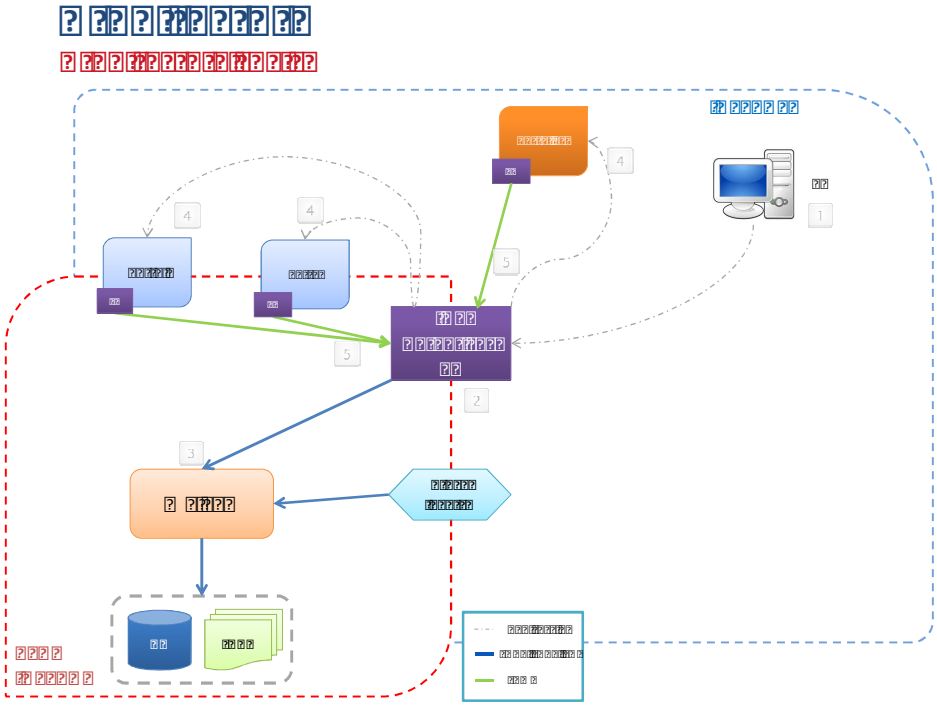
Este es un pequeño fragmento de la autenticacion contra facebook

```
public function authenticate(&$state) {
assert('is_array($state)');
/*Necesitamos el authid para recibir este recurso de autenticacion luego */
$state[self::AUTHID] = $this->authId;
$stateID = SimpleSAML_Auth_State::saveState($state, self::STAGE_INIT);
```

?

```
$facebook = new sspmod_authfacebook_Facebook(array('appId' => $this->api_key, 'secret' =>
$this->secret), $state);

$facebook->destroySession();

$linkback = SimpleSAML_Module::getModuleURL('authfacebook/linkback.php',
array('AuthState' => $stateID));

$url = $facebook->getLoginUrl(array('redirect_uri' => $linkback, 'scope' => $this->req_perms));

SimpleSAML_Auth_State::saveState($state, self::STAGE_INIT);

SimpleSAML_Utilities::redirectTrustedURL($url);
```

13. Arquitectura lógica de la solución



14. Conclusiones

- Tener un sistema federado de autenticación permite tener más control operaciones de la plataforma.
- Un sistema federado de autenticación basa costos de desarrollo, diseño, implementación y operatividad.
- Podemos afirmar que SAML resulta ser una especificación satisfactoria para lograr Single Sing-on web entre diversas aplicaciones.
- Mitigación del riesgo, con este sistema centralizado mitigamos riesgo de errores y/o vulnerabilidades en el sistema
- Implementando SAML podemos extender círculos de confianza implementando nuevos servicios a los clientes
- Con un sistema single sign on , los usuarios tendrán mas amigabilidad para viajar entre portales.
- La creciente demanda de usuarios con redes sociales nos genera la necesidad de involucrarlas en nuestros procesos , con el fin de que el cliente pueda disminuir su número de usuarios y passwords para facilitarles el acceso a los servicios.

15. Referencias

[SAMLAuthnCxt] J. Kemp et al. Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS SSTC, Marzo 2005. http://docs.oasisopen.org/security/saml/v2.0/saml-authncontext-2.0-os.pdf.

[SAMLBind] S. Cantor et al. Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS SSTC, Marzo 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf.

[SAMLCore] S. Cantor et al. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS SSTC, Marzo 2005. https://simplesamlphp.org/

Wikipedia. Single sign-on. http://en.wikipedia.org/wiki/Single_sign-on

Bizagi autenticación federada http://help.bizagi.com/bpmsuite/es/index.html?sso_authentication.htm

Setting up a simpleSAMLphp SAML 2.0 IdP to use with Google Apps for Education https://rnd.feide.no/doc/simplesamlphp-googleapps_1.2.pdf

SP manual on how to integrate SimpleSAMLphp in your PHP application http://taat.edu.ee/main/wp-content/uploads/SP-php-eng.pdf

Web Single Sign-On Implementation Using the SimpleSAMLphp Application

http://www.jmeds.eu/index.php/jmeds/article/viewFile/Web-Single-Sign-On-Implementation-Using-the-SimpleSAMLphp-Application/pdf

Single Sign-On

Single Sign-On – A Contrarian View

http://www.databaseanswers.org/ibm_sso.htm

DIGIPASS Authentication for SimpleSAMLphp using IDENTIKEY Federation Server

https://www.vasco.com/large_download/Tracking/IFS-SimpleSAMLphp-Integration.pdf