Single Sign-On © ADR Infor SL

Indice

Single Sign-On

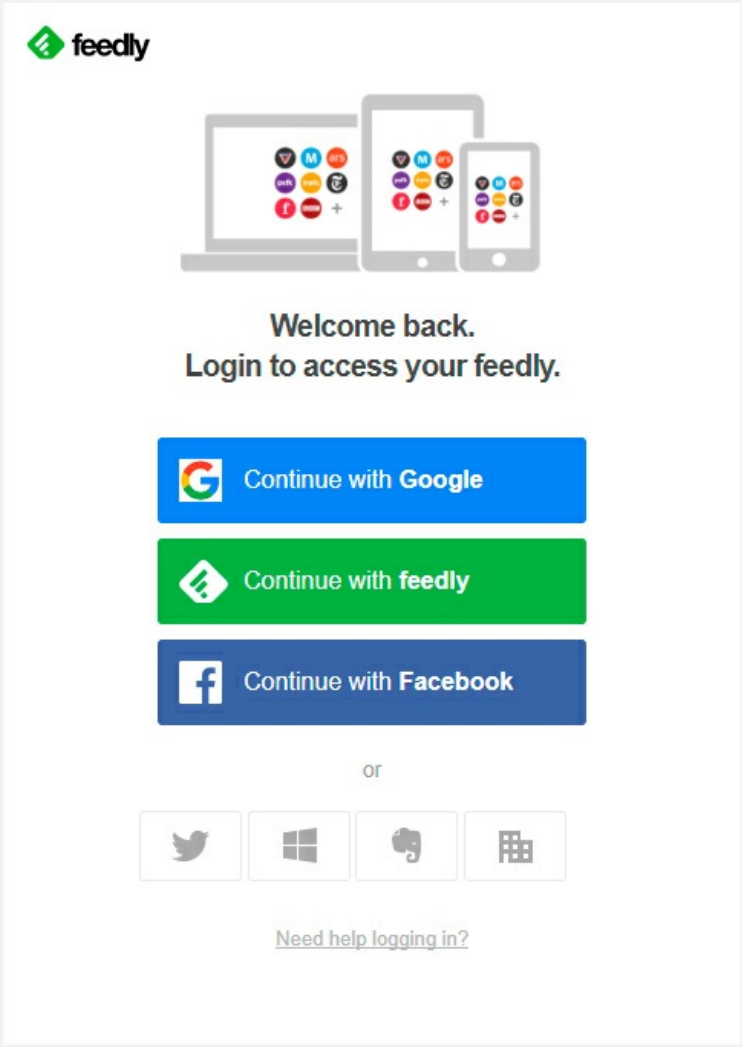
Al finalizar esta unidad el alumno será capaz de crear un sistema de autenticación entre dos aplicaciones web independientes para que el usuario pueda identificarse en ambas con un solo login.

Desarrollar aplicaciones web significa **trabajar en un entorno interconectado**. Nuestras aplicaciones no se encuentran aisladas en un espacio cerrado, como sucede con las clásicas aplicaciones de escritorio, sino que forman parte de un ecosistema enorme en el que miles de aplicaciones conviven y **pueden interactuar entre sí**.

A lo largo de este curso, descubrirás que la relación entre distintas aplicaciones web es muy habitual y forma parte de su filosofía. En esta unidad nos centraremos en un aspecto muy concreto y relevante de esta interconexión: cómo hacer que un usuario, que ha iniciado sesión con su usuario y contraseña, **mantenga su login entre diferentes sitios web**.

Ejemplo: Iniciar sesión con redes sociales

Habrás visto en multitud de ocasiones, y seguramente utilizado, sitios web que ofrecen registrarse o iniciar sesión utilizando redes sociales o servicios populares como Facebook, Google, Twitter o similares:



Al pulsar sobre el método de login seleccionado, el usuario puede iniciar sesión en la web actual, sin necesidad de identificarse, utilizando únicamente el inicio de sesión que ha realizado en otra web.

Es evidente que para que un usuario pueda identificarse en nuestra aplicación web utilizando otra aplicación web, **que no se encuentra en nuestro servidor y de la que no somos propietarios**, es necesario que exista **transferencia de información** entre ambas.

En todo procedimiento de autenticación entre varios sistemas siempre hay, al menos, dos actores:

Proveedor de identidad

Es el sistema **encargado de proveer los datos del login**. El usuario debe haberse identificado en este sistema para extender su identificación a otro sistema.

Proveedor de servicio

Es el sistema que **recibe los datos del proveedor de identidad** y da acceso a sus servicios con ellos.

Además toda transferencia de información entre estos dos actores debe atender a varias premisas:

Acceso limitado

El proveedor de identidad, evidentemente **no nos ofrecerá un acceso a su base de datos** para que la consultemos.

Del mismo modo, si nosotros somos el proveedor de identidad, no debemos ofrecer tampoco al proveedor de servicio un acceso a nuestra base de datos.

Toda la transmisión de información se realizará mediante **parámetros** a través de **conexiones HTTP**. De esta forma, sólo se transfiere la información que el proveedor de identidad decida y **se independiza el tipo de tecnología** (lenguaje, servidor, base de datos...) que utilice cada aplicación.

Seguridad

Para realizar una acción tan sensible, como es un login dentro de nuestra aplicación web, debemos tomar ciertas precauciones de seguridad.

Es necesario **identificar al proveedor de identidad** de manera inequívoca para que no se produzca ningún tipo de suplantación de identidad y a la vez, establecer un mecanismo para asegurarnos de que **los datos que nos han llegado, son los mismos que éste nos ha enviado**.

Identificación común

Para poder relacionar un usuario en ambas aplicaciones, **es necesario que ambos tengan un identificador común**.

Esta es la única forma de saber que un usuario de un sistema, es el mismo que el de otro sistema.

¿Qué es SSO?

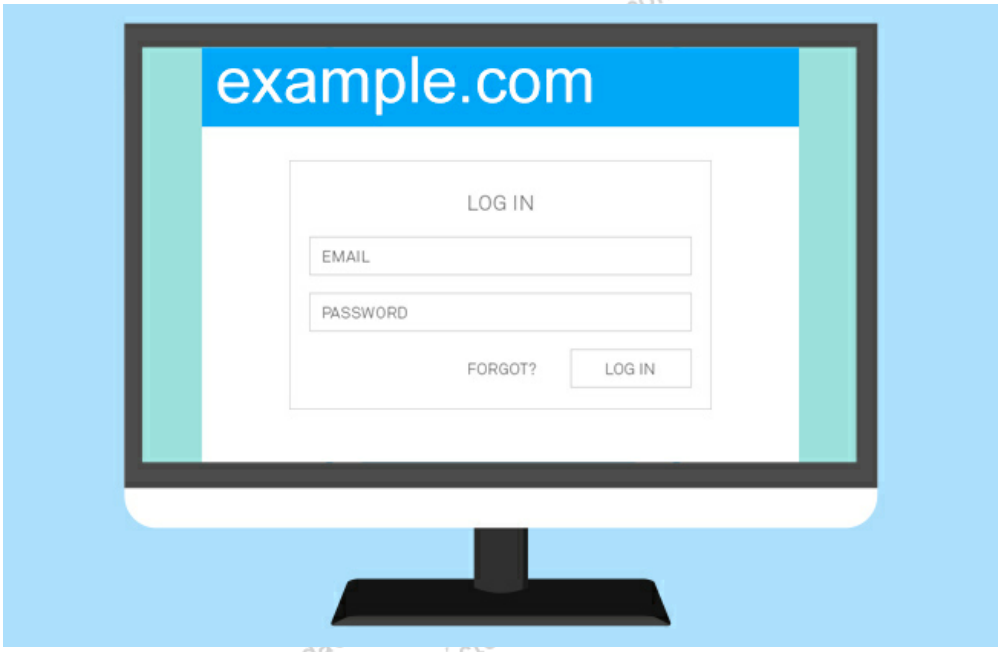
Single Sign-On, más comúnmente conocido como SSO, es un **procedimiento de autenticación**, muy habitual en aplicaciones web. Su funcionamiento es muy sencillo de implementar a la vez que eficaz.

La forma más sencilla de uso es la existencia de un **enlace** en el cual el usuario hace clic e inmediatamente **aparece logeado en el proveedor de servicio**

El **proveedor de identidad es el encargado de generar este enlace**, que apuntará a una **URL específica del proveedor de servicio** y le enviará una serie de parámetros para que éste pueda identificar al usuario (Por ejemplo el ID de usuario).

Vamos a verlo más claramente con un ejemplo:

Un usuario **se logea** en la web www.example.com con su email (manolo@gmail.com) y su contraseña.



Comienza a trabajar con normalidad en www.example.com.

En un momento determinado la web **le ofrece un enlace** para acceder a un servicio diferente.

La URL de este enlace será: http://www.otroservicio.com/sso.php?email=manolo@gmail.com.

El usuario **hace clic en el enlace** que le lleva a:

http://www.otroservicio.com/sso.php?email=manolo@gmail.com

El script sso.php de www.otroservicio.com **comprueba la dirección de email** que le ha llegado. Si está en su base de datos de usuarios, **crea las variables de sesión correspondientes para logear al usuario en su sistema**.

Si el email no está en su base de datos,**crea un usuario asociado al mismo** y lo logea igualmente.

Finalmente redirige al usuario a la portada dehttp://www.otroservicio.com/miperfil.php

El usuario aparece en http://www.otroservicio.com/miperfil.php ya logeado y comienza a trabajar con normalidad en la página.

Evidentemente a este proceso hay que añadirle las **verificaciones de seguridad** ya que cualquiera podría generar el enlace del SSO y acceder al proveedor de servicios.

Explicaremos este paso más adelante.

Proveedor de identidad: Crear el enlace

La misión del proveedor de identidad es **generar el enlace** para que el usuario pueda logearse en el proveedor de servicio **con un solo clic**.

Este enlace consta de dos partes:

Endpoint

Un **endpoint**, es un **punto de acceso** que nos provee un sistema para comunicarnos y realizar acciones en él.

En el caso de la implementación de un SSO, el endpoint será la **URL del script que ha desarrollado el proveedor de servicio** para recibir la identificación.

Se denomina endpoint porque es el **punto final de una infraestructura** a la que no podemos acceder, es decir, es el **único punto de acceso visible** de la misma.

Parámetros

Utilizaremos parámetros para enviar la **información necesaria para identificar al usuario**, al endpoint.

Lo más habitual es enviar estos parámetros mediante GET para **poder construir una URL con ellos**, no obstante, también puede implementarse una llamada mediante POST dependiendo del caso.

La información a enviar debe ser una **identificación del usuario**. Un ejemplo de identificación válida puede ser un email o un ID.

En cualquier caso, ambos actores **deben compartir esta misma identificación**, y relacionarla con el usuario en cada uno de sus sistemas, por lo que deben ponerse de acuerdo previamente sobre la información enviada.

Aunque, a priori no es estrictamente necesario, pueden enviarse parámetros adicionales para añadir información extra.

Vamos a ver un ejemplo de enlace generado por el proveedor de identidad:

http://www.proveedordeservicio.com/sso.php/?email=usuario@gmail.com

- El **endpoint** que nos ha suministrado el proveedor de servicio es: **http://www.proveedordeservicio.com/sso.php**
- A este endpoint le hemos enviado un **parámetro** mediante GET indicando el **email del usuario que actualmente está registrado** en nuestro sistema.
- Este email será lo que **identifique al usuario en ambos sistemas**.

Generando el enlace para el SSO

Proveedor de servicio: Implementar el endpoint

La misión del proveedor de servicio es **generar el endpoint** al cual debe llamar el enlace generado.

El endpoint será un **script de PHP** que realice las operaciones necesarias para la comunicación:

- Obtenemos el **identificador de usuario** que nos ha sido suministrado a través de un parámetro. Éste identificador, **siempre debe estar relacionado en nuestra base de datos con el usuario**.
- Buscamos el **usuario que corresponde con el identificador** en nuestra base de datos.
- Si no existiese, **creamos un nuevo usuario** en nuestra base de datos y lo **asociamos con el identificador**. Depende de cómo sea la arquitectura de nuestro sistema, puede que para dar de alta un usuario, necesitemos información adicional. En tal caso, el proveedor de identidad, debe suministrarnos también esta información con parámetros adicionales.
- **Logeamos al usuario** en nuestro sistema. Para ello debemos seguir **el mismo método** que utilizamos habitualmente para el logeo de nuestros usuarios. Por ejemplo, si el logeo consiste en rellenar una variable de sesión con el nombre del usuario, rellenaremos esta misma variable de sesión.
- Redirigimos a la página de bienvenida correspondiente.

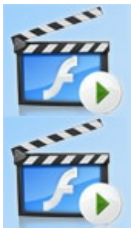
Importante: Al crear un usuario desde el SSO ¿Qué contraseña le asigno?

Cuando creamos un nuevo usuario a través del SSO, el proveedor de identidad **nunca debe proporcionarnos la contraseña** como dato para crearlo.

Lo más habitual en estos casos es **generar una contraseña aleatoria**.

Nuestro usuario utilizará como método de identificación, el SSO. Aunque si nuestro sistema tiene un procedimiento para cambiar de contraseña, puede utilizarlo para establecer una nueva contraseña y logearse de forma directa.

En los siguientes vídeos vamos a ver un ejemplo de implementación del endpoint:



Implementando el endpoint

Creando usuarios desde el endpoint

Securizando el SSO

Aunque nuestro SSO en este momento ya sería plenamente funcional, es evidente que tiene un grave problema de seguridad ya que **cualquiera podría generar el enlace** necesario para logearse en nuestro sistema.

Es **totalmente necesario que añadamos ciertas restricciones de seguridad** para limitar su uso y evitar accesos no autorizados. Vamos a ver cómo hacerlo:

Firma de datos

Para verificar que los datos que llegan al endpoint, **han sido suministrados exclusivamente por el proveedor de identidad**, debemos realizar un **firmado de datos**.

Los procesos de firma de datos son frecuentes entre **dos sistemas que deben comunicarse entre sí a través de internet**.

Un ejemplo muy habitual son los **pagos electrónicos** en los que una tienda online, se comunica con una pasarela de pago de una entidad bancaria. En estos casos siempre se utiliza un firmado de datos para evitar suplantaciones de identidad.

Para realizar cualquier firmado de datos, necesitaremos dos elementos:

Clave privada

Una clave privada es una **cadena de texto que contiene una serie de caracteres**

El proveedor de identidad y el proveedor de servicio **deberán conocer esta clave privada, pero nunca deben darla a conocer** bajo ninguna circunstancia.

La clave privada **nunca se enviará en una comunicación** y **nunca debe salir del servidor** donde esté almacenada para que no pueda ser interceptada.

Algoritmo de encriptación

Para generar la firma, utilizaremos un **algoritmo de encriptación de un solo sentido**.

Esta clase de algoritmos generan un **hash**, es decir, una **cadena de texto de longitud fija** que se obtiene tras realizar una serie de procesos con la cadena de texto original.

La particularidad de estos algoritmos es que **no es posible desencriptar sus resultados**, lo que significa, que **desde el hash no se puede obtener la cadena de texto original**

Algunos de los algoritmos de encriptación de un solo sentido más utilizados son:

- MD5
- SHA1
- SHA256

El proceso de firmado de datos es el siguiente:

- Juntamos en **una sola cadena de texto**, todos la información que deseamos firmar. En este caso, serían los parámetros que vamos a enviar al endpoint.
- Añadimos a dicha cadena de texto la **clave privada**.
- **Encriptamos la cadena de texto** con un algoritmo de encriptación en un solo sentido. El hash devuelto será **la firma**.
- Enviamos la firma como parámetro adicional al endpoint.

Para verificar la firma desde el endpoint seguiremos los siguientes pasos:

- Juntamos en **una sola cadena de texto**, los parámetros que nos ha enviado el proveedor de identidad. Debemos hacerlo en el mismo orden que lo ha hecho él, para obtener la misma cadena de texto.
- Añadimos a dicha cadena de texto la **clave privada**.
- **Encriptamos la cadena de texto** con un algoritmo de encriptación en un solo sentido. El hash devuelto será **la firma**.
- Si la firma generada, coincide con la firma enviada, podemos realizar el proceso.

Fíjate que en ningún momento la clave privada ha salido transmitida, ni ha salido de los servidores.

Ejemplo: Firmar los datos de un enlace SSO

Tenemos el siguiente enlace y queremos firmar sus datos:

http://www.ps.com/sso.php/?email=usuario@gmail.com

Para lo cual tenemos la siguiente clave privada:

PN98dS9dsnj789ds

Generamos la cadena de texto de la siguiente forma:

usuario@gmail.comPN98dS9dsnj789ds

La encriptaremos con el algoritmo SHA1 obteniendo el siguiente hash:

9527ffd2e93300bfe4a1299ebc7f6ded96d7db41

Añadiremos la firma al enlace que quedará de este modo:

http://www.ps.com/sso.php/?
email=usuario@gmail.com&firma=9527ffd2e93300bfe4a1299ebc7f6ded96d7db41

El proveedor de servicio realizará este mismo proceso con la información suministrada y comprobará ambas firmas.

Firmando los datos enviados al SSO

Funcionamiento de algoritmos criptográficos

Aunque hemos dicho que el hash generado por un algoritmo de encriptación de un solo sentido, no puede ser desencriptado, **esto no es del todo cierto**.

Estos algoritmos de encriptación son creados por matemáticos y basan su funcionamiento en generar un mecanismo en el que **en muy pocos pasos, se llegue de la cadena inicial al hash**, pero se necesiten **muchísimos pasos para llegar del hash a la cadena inicial**.

Esto significa que **la encriptación será muy rápida y la desencriptación extremadamente lenta**. Para hacernos una idea, la encriptación, con un ordenador medio, se realiza en **milisegundos**, pero la desencriptación, aun utilizando un superordenador, costaría **décadas** o **siglos** de procesamiento.

Por este motivo, y si cambiamos nuestras claves privadas con una frecuencia razonable, podemos decir que, **en la práctica**, no se puede desencriptar un hash.

No obstante, a medida que avanza el desarrollo de ordenadores mucho más potentes, los algoritmos se ven cada vez más amenazados **pudiendo llegar a ser inseguros al reducir el tiempo de desencriptación** debido a una mayor velocidad de proceso. Recientemente, todos los sistemas bancarios cambiaron su algoritmo de firma de SHA1 a SHA256 por este motivo.

El desarrollo de ordenadores cuánticos podría suponer un problema para la criptografía y el firmado de datos.

Hacer que el enlace caduque

Otra técnica de seguridad adicional, sería hacer que el enlace generado **funcionase solo por un determinado periodo de tiempo**. Esta es una medida muy sencilla de implementar que nos previene ante un posible **robo del enlace**.

Para establecer la caducidad del enlace realizaremos los siguientes pasos:

- Enviaremos un **parámetro adicional con el timestamp** actual.
- **Firmaremos el timestamp** junto con el resto de parámetros.
- Desde el endpoint, verificamos que **el timestamp no está caducado**, es decir, que no ha pasado más de un determinado plazo desde que se generó.

De esta forma, el enlace solo será válido durante un determinado plazo.

Para que esta técnica funcione, ambos servidores **deben tener su fecha y hora establecida correctamente** para poder sincronizar sus comunicaciones.

Establecer caducidad para el enlace

Otras opciones de implementación

Hemos visto el caso de uso más habitual de un SSO, pero esta misma técnica **puede tener ligeras variaciones de implementación** dependiendo del entorno y nuestras necesidades.

Vamos a ver un par de ejemplos de otras implementaciones típicas:

Cuando el proveedor de identidad y el proveedor de servicio son la misma aplicación

Aunque pueda parecer extraño que el proveedor de identidad y el proveedor de servicio sean la misma aplicación, hay situaciones en las que **es muy interesante desarrollar esta opción**.

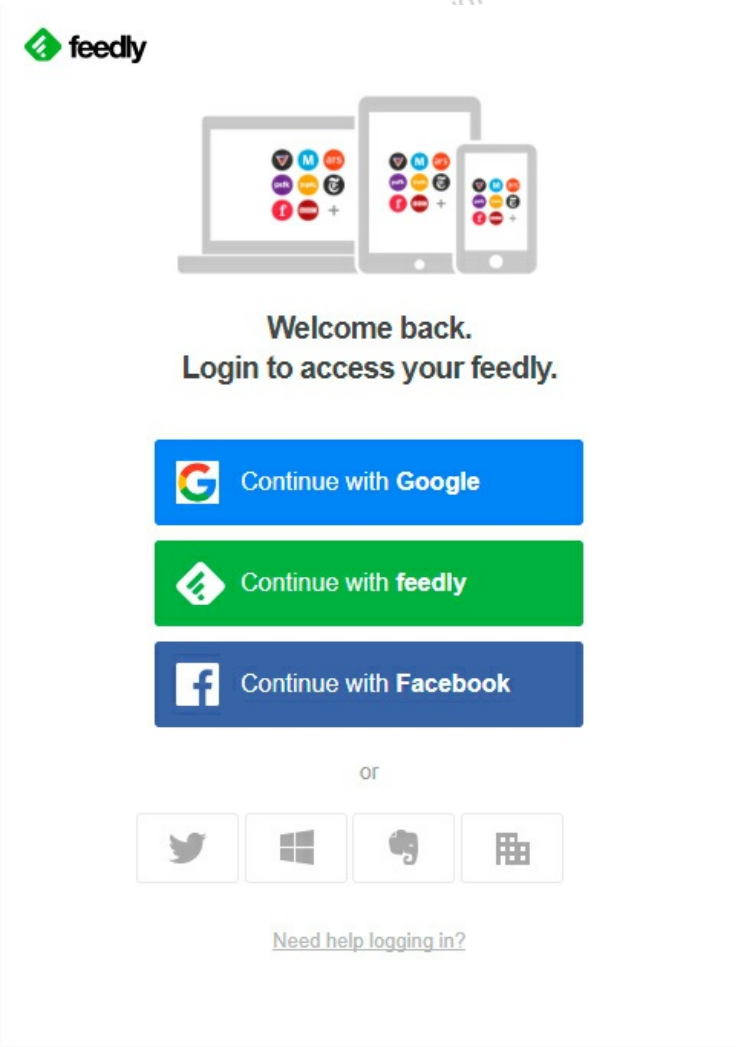
Suele ser utilizado sobre todo para **generar un enlace que pueda ser enviado por email**. De esta forma, el usuario aparecerá logeado inmediatamente en la página al pulsar el enlace.

Si enviamos emails en las que instamos al usuario a realizar acciones, pulsando un enlace del mismo, para las cuales debe estar registrado, **resultará imprescindible tener un SSO para nosotros mismos**.

Para implementar esta opción, solo debemos **generar el endpoint y el enlace** como si de un SSO normal se tratara.

Generar un botón desde el proveedor de servicio para logearse con un proveedor de identidad.

En muchas ocasiones habrás visto, que una página ofrece la opción de hacer un login en ella **mediante otro servicio**:



En este caso **es el proveedor de servicio el que ofrece el login**, en vez del proveedor de identidad, lo cual nos resultará útil para ofrecer varias opciones de ingreso.

Una vez desarrollado el SSO habitual, implementar esta opción es muy sencillo:

- El **proveedor de identidad** desarrollará otro **endpoint**.
- Desde el proveedor de servicio, **generamos un enlace que llame a este endpoint**, sin necesidad de parámetros específicos.
- En este nuevo endpoint, el proveedor de identidad, **generará el enlace hacia el endpoint del proveedor de servicio de la forma habitual**.
- El endpoint del proveedor de identidad **redirigirá al enlace**, siguiente el proceso habitual del SSO.

Vamos a ver un ejemplo:

Botón de login desde el proveedor de servicios

Utilizar redes sociales y otros servicios populares como proveedor de identidad

Las redes sociales y otros servicios relevantes de Internet, cuentan con **millones de usuarios** en sus sistemas. Esto hace que sean un **proveedor de identidad ideal** ya que, muchos de nuestros posibles usuarios, **tendrán una cuenta en dichos servicios**.

Muchos de estos servicios **nos ofrecen la posibilidad de ejercer como proveedores de identidad** para nuestros SSO. Algunos de los más utilizados son: **Facebook, Google, Twitter**...

Para utilizar uno de estos servicios como proveedor de identidad, debemos **implementar la operativa que éstos nos indican**. Habitualmente existe una **página dedicada a desarrolladores** en estos servicios donde nos indican instrucciones detalladas sobre la integración de nuestra aplicación con sus servicios.

Enlaces: Páginas de desarrolladores de servicios populares

Página para desarrolladores de Facebook

Página para desarrolladores de Google

Página para desarrolladores de Twitter

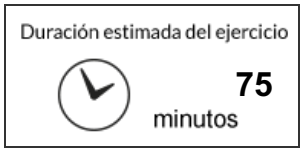
Hemos aprendido

En esta unidad hemos aprendido:

- Un Single Sign-On (SSO) es un **procedimiento de autenticación** en el que, con una única identificación, el usuario puede acceder a servicios en diferentes aplicaciones web.
- En el proceso de autenticación intervienen dos actores:
  - **Proveedor de identidad**: Es donde el usuario se encuentra logeado y se encarga de suministrar la información del login.
  - **Proveedor de servicio:** Es la aplicación que el usuario utilizará, usando la identificación del proveedor de identidades.
- El procedimiento de autenticación será el siguiente:
  - El proveedor de identidad **envía los parámetros necesarios para identificar al usuario al endpoint** del proveedor de servicio.
  - El proveedor de servicio, **recogerá los parámetros del proveedor de identidad en su endpoint y realizará el login del usuario**.
- Para securizar que el emisor de la información sea el proveedor de identidad utilizaremos obligatoriamente un proceso de **firma de datos**.

Ejercicios

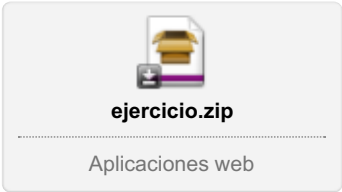
Implementa un SSO



Para realizar este ejercicio deberás implementar un SSO entre dos aplicaciones web. Es necesario que realices tanto la parte correspondiente a al proveedor de identidad, como la correspondiente al proveedor de servicio.

Lo necesario para comenzar

1. Descarga el archivo **ejercicio.zip** y descomprímelo en el directorio público de tu servidor.
2. Crea dos bases de datos MySQL llamadas **proveedoridentidad** y **proveedorservicio**.
3. Ejecuta los archivos **proveedoridentidad.sql** y **proveedorservicio.sql** en sus respectivas bases de datos.
4. Accede a tu servidor a través del navegador. Verás que hay dos carpetas. Cada una es una aplicación web totalmente independiente en la que hay implementado un login de usuarios. Una actuará como proveedor de identidad y otra como proveedor de servicio.



A continuación puedes ver los usuarios creados en el proveedor de identidad para que puedas realizar pruebas con ellos:

Usuario	Contraseña
jose@gmail.com	1234
pedro@gmail.com	1234
juan@gmail.com	1234

Pasos a seguir

1. Crea un nuevo archivo PHP en el proveedor de servicio que servirá como **endpoint** del SSO.
2. En el área privada del proveedor de identidad **crea un enlace que apunte al endpoint** y envíale el **email como parámetro** de identificación.
3. Implementa el endpoint de tal forma que **busque en la base de datos el email** recibido y **cree un nuevo usuario** si no lo encuentra.

4. **Logea al usuario** rellenando el email y la contraseña en las **variables de sesión** correspondientes.
5. Comprueba que el SSO funciona con el usuario jose@gmail.com y con el usuario pedro@gmail.com.
6. Securiza tu SSO con un **proceso de firma**, utilizando una **clave privada** y el algoritmo **SHA1**.
7. Establece un **tiempo de caducidad** al enlace, añadiendo el **timestamp** a los parámetros del enlace y comprobando en el endpoint que **no hace más de 15 minutos que se generó**
8. Dota de funcionalidad al **botón ENTRAR CON PI** del proveedor de servicio. Este botón se encargará de hacer un login en el mismo servicio, utilizando el proveedor de identidad. Para ello debes seguir estos pasos:
    1. Crea un **archivo PHP en el proveedor de identidad** que actuará como endpoint del mismo.
    2. Haz que el botón ENTRAR CON PI **apunte a este endpoint** del proveedor de identidad.
    3. Desde el endpoint del proveedor de identidad **genera el mismo enlace** que llama al endpoint del proveedor de servicio desde el área privada.
    4. Haz una redirección hacia dicho enlace.

Recursos

Enlaces de Interés

https://developers.facebook.com/
https://developers.facebook.com/

Página para desarrolladores de Facebook

https://developers.google.com/
https://developers.google.com/

Página para desarrolladores de Google

https://dev.twitter.com/
https://dev.twitter.com/

Página para desarrolladores de Twitter

Preguntas Frecuentes

1. **¿Cómo realizo una encriptación con SHA256 en PHP?**

   La sintaxis para realizar la encriptación en SHA256 sería:

   hash('sha256','texto a encriptar')

2. **¿Todos los algoritmos de encriptación son igual de buenos?**

   No, hay algunos más eficientes que otros. Los algoritmos de encriptación evolucionan a medida que evoluciona la potencia de desencriptación de los ordenadores.

   Actualmente se está comenzando a utilizar **SHA256** como estándar, pero encontrarás muchos sistemas que utilizan SHA1 o MD5. Estos dos últimos, recientemente se han empezado a considerar potencialmente inseguros, pero depende de las características del proyecto, pueden utilizarse sin problema.

Glosario.

- **Endpoint**: Es una URL que nos provee un sistema para comunicarnos y realizar acciones en él. Se denomina endpoint porque es el punto final de una infraestructura a la que no podemos acceder, es decir, es el único punto de acceso visible de la misma.
- **Hash**: Resultado de un algoritmo de encriptación. Es una cadena de texto de longitud fija que se obtiene tras realizar una serie de procesos con la cadena de texto original.
- **Proveedor de identidad**: En un proceso de identificación entre dos sistemas, es el encargado de suministrar la identidad del usuario.

- **Proveedor de servicio**: En un proceso de identificación entre dos sistemas, es el encargado de recoger los datos suministrados por el proveedor de identidad y logear al usuario en su sistema para proporcionarle un servicio.

- **SSO**: Procedimiento de autenticación que habilita al usuario para acceder a varios sistemas con una sola instancia de identificación.