

Implantación de sistema SSO

Juan Manuel Álvarez Rodríguez

Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones (MISTIC)

Sistemas de autenticación y autorización

Antoni González Ciria

Víctor García Font

06/2020



Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada [3.0 España de Creative Commons](https://creativecommons.org/licenses/by-nc-nd/3.0/es/)

FICHA DEL TRABAJO FINAL

| | |
|--|---|
| Título del trabajo: | <i>Implantación de sistema SSO</i> |
| Nombre del autor: | <i>Juan Manuel Álvarez Rodríguez</i> |
| Nombre del consultor/a: | <i>Antoni González Ciria</i> |
| Nombre del PRA: | <i>Víctor García Font</i> |
| Fecha de entrega (mm/aaaa): | 06/2020 |
| Titulación: | <i>Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones</i> |
| Área del Trabajo Final: | Sistemas de autenticación y autorización |
| Idioma del trabajo: | <i>Castellano</i> |
| Palabras clave | SSO <i>Autenticación</i> <i>Autorización</i> |
| Resumen del Trabajo (máximo 250 palabras): | |
| <p>Con la paulatina digitalización de los servicios de las administraciones públicas y el aumento de plataformas digitales privadas, se los usuario cuentan con una cantidad ingente de credenciales que deben gestionar.</p> <p>Además, el uso de estas credenciales básicas está perdiendo relevancia en favor de métodos de autenticación más seguros como los certificados digitales. Los servicios proporcionados de manera telemática no pueden ofrecer el mismo nivel de acceso a usuarios autenticados con diferentes medidas de seguridad. Los procesos más críticos deben tener unos controles más estrictos de la identidad de los usuarios.</p> <p>Ambos problemas se pueden solucionar implementando un sistema de Single Sign On. Esto permite a un usuario autenticado acceder a todas las aplicaciones que hagan uso de este sistema con unas credenciales únicas.</p> <p>En este trabajo se analiza, diseña e implementa un sistema SSO que proporciona esta solución. Para ello se utilizará una metodología de desarrollo de sistemas informáticos conocida como desarrollo en cascada.</p> <p>Finalmente, se llega a la conclusión de que no todas las plataformas elegidas, aunque proveen la funcionalidad deseada, tienen el mismo nivel de madurez y un trabajo más exhaustivo de análisis de estas debe ser llevada a cabo durante la fase de estudio de plataformas. Así mismo, la implantación del sistema no exime al equipo de puesta en producción de ejecutar otras tareas que hagan el sistema mucho más seguro y que quedan fuera del objetivo de este trabajo.</p> | |

Abstract (in English, 250 words or less):

With the gradual digitization of public administration services and the increase in private digital platforms, users have a huge number of credentials to manage.

Furthermore, the use of these basic credentials is losing relevance in favour of more secure authentication methods such as digital certificates. The services provided electronically cannot offer the same level of access to authenticated users with different security measures. The most critical processes must have stricter controls on the identity of the users.

Both problems can be solved by implementing a Single Sign On system. This allows an authenticated user to access all the applications that make use of this system with just a set of credentials.

This project analyses, designs and implements a Single Sign On authentication system that provides this solution. To do this, a methodology for developing computer systems known as cascade development will be used.

Finally, it is concluded that not all the chosen platforms, although the desired functionality was verified, have the same level of maturity and a more exhaustive work of analysis of these platforms must be carried out during the proof of concept phase. Likewise, the implementation of the system does not exempt the commissioning production team from other tasks that would make the system much more secure and that are outside of the scope of this project.

Índice

| | |
|---|----|
| 1. Introducción..... | 1 |
| 1.1 Contexto y justificación..... | 1 |
| 1.2 Objetivos | 2 |
| 1.3 Metodología..... | 2 |
| 1.4 Estado del arte | 3 |
| 1.5 Planificación de tareas | 5 |
| 1.6 Análisis de riesgos | 7 |
| 1.7 Planificación presupuestaria..... | 9 |
| 2. Requisitos..... | 9 |
| 2.1 Funcionales | 9 |
| 2.2 No funcionales..... | 10 |
| 2.3 Casos de prueba | 11 |
| 3. Análisis de OpenAM..... | 13 |
| 3.1 Arquitectura | 13 |
| 3.2 Autenticación..... | 15 |
| 3.3 Sesiones y SSO | 16 |
| 3.4 Configuración | 17 |
| 3.5 OpenDS..... | 18 |
| 3.6 Gestión de usuarios..... | 20 |
| 4. Análisis de CAS..... | 20 |
| 4.1 Arquitectura | 21 |
| 4.2 Sesiones y SSO | 23 |
| 4.3 Autorización..... | 23 |
| 5. Análisis de HAProxy | 24 |
| 5.1 Arquitectura | 25 |
| 5.2 Procesado de peticiones | 25 |
| 5.3 Funcionalidades | 26 |
| 6. Análisis de OpenIG | 26 |
| 6.1 Arquitectura | 27 |
| 6.2 Procesado de peticiones | 28 |
| 7. Plataformas seleccionadas..... | 28 |
| 8. Diseño | 29 |
| 8.1 Diagramas de secuencia | 30 |
| 8.2 Diagrama de componentes | 32 |
| 8.3 Diagrama de despliegue..... | 33 |
| 8.4 Diagrama de red..... | 34 |
| 9. Instalación de plataformas y tecnologías base..... | 35 |
| 9.1 Instalación de OpenDS..... | 36 |
| 9.2 Instalación de OpenAM | 40 |
| 9.3 Instalación de aplicaciones funcionales..... | 41 |
| 9.4 Instalación de HAProxy | 42 |
| 9.5 Instalación de OpenIG..... | 43 |
| 10. Configuración de la arquitectura..... | 44 |
| 10.1 OpenDS Replicación | 44 |
| 10.2 Configuración de OpenAM | 46 |
| 10.3 Configuración de HAProxy | 47 |
| 10.4 Configuraciones adicionales..... | 48 |
| 11. Configuración funcional | 49 |

| | |
|---|----|
| 11.1 OpenAM | 49 |
| 11.2 OpenIG..... | 52 |
| 12. Gestión de certificados | 54 |
| 13. Desarrollo aplicaciones de prueba | 57 |
| 14. Pruebas | 58 |
| 15. Conclusiones..... | 60 |
| 16. Posibles extensiones..... | 61 |
| 17. Glosario | 62 |
| 18. Bibliografía | 63 |
| 20. Anexo I: Instalación de OpenAM | 64 |
| 20.1 Primer nodo..... | 64 |
| 20.2 Segundo nodo y siguientes | 68 |
| 21. Anexo II: Filtros de OpenIG | 69 |

Lista de figuras

| | |
|---|----|
| Figura 1 – Listado de tareas..... | 5 |
| Figura 2 – Planificación de tareas | 6 |
| Figura 3 - Ejemplo de arquitectura OpenAM Multi-CPD | 15 |
| Figura 4 - Entrada directorio activo | 19 |
| Figura 5 - Arquitectura CAS | 22 |
| Figura 6 - Diagrama de secuencia RF-01 | 30 |
| Figura 7 - Diagrama de secuencia RF-02 | 31 |
| Figura 8 - Diagrama de componentes | 32 |
| Figura 9 - Diagrama de despliegue | 33 |
| Figura 10 - Diagrama de red | 34 |
| Figura 11 - Configuración inicial OpenAM | 46 |
| Figura 12 - Monitorización estado HAProxy | 48 |
| Figura 13 - Árbol de autenticación intranet..... | 49 |
| Figura 14 - Políticas de autorización | 50 |
| Figura 15 - Política de autorización básica..... | 51 |
| Figura 16 - Política de autorización certificado..... | 51 |
| Figura 17 - Ejemplo de usuario en el almacén de identidades | 52 |
| Figura 18 - Rutas de OpenIG | 53 |
| Figura 19 - Cadena de filtros ruta básica | 53 |
| Figura 20 - Gestor de eventos de auditoría OpenIG | 54 |
| Figura 21 - Creación clave de firma CA | 55 |
| Figura 22 - Creación certificado CA | 55 |
| Figura 23 – Creación clave de firma servidor OpenIG | 55 |
| Figura 24 - Creación CSR servidor OpenIG | 56 |
| Figura 25 - Creación certificado OpenIG | 56 |
| Figura 26 - Conversión certificado crt a pfx..... | 56 |
| Figura 27 - Conversión e importación a jks | 56 |
| Figura 28 - Controlador de Spring Framework | 57 |
| Figura 29 - Html creada con Thymeleaf | 57 |
| Figura 30 - Captura aplicación basic | 58 |
| Figura 31 - Creación de contraseña usuario amadmin..... | 64 |
| Figura 32 - Configuración básica de servidor | 64 |
| Figura 33 - Configuración del OpenDS de configuración | 65 |
| Figura 34 - Configuración del almacén de usuarios | 65 |
| Figura 35 - Configuración del Site | 66 |
| Figura 36 - Resumen de configuración | 66 |
| Figura 37 - Ejecución de configuración | 67 |
| Figura 38 - Configuración de los almacenes de tokens..... | 67 |
| Figura 39 - Alta disponibilidad de almacén de configuración | 68 |
| Figura 40 - Alta disponibilidad almacén de identidades | 68 |
| Figura 41 - Configuración de OpenAM en una instalación ya existente | 69 |

1. Introducción

1.1 Contexto y justificación

Actualmente, con la paulatina digitalización de los servicios de las administraciones públicas y el aumento de plataformas digitales privadas, se ha llegado a un punto en la que los usuarios cuentan con una cantidad ingente de credenciales de usuario que deben gestionar. Habitualmente, estas credenciales tienen el formato de nombre de usuario y contraseña. Las contraseñas, además, tienen en cada una de las plataformas unos requisitos específicos de complejidad.

Como consecuencia de estos dos factores, los usuarios suelen reutilizar un número muy reducido de contraseñas para que su gestión no se haga complicada y sean fáciles de recordar. Este hecho supone un riesgo de seguridad por las posibles filtraciones de usuarios de servicios públicos y privados.

Según (Wang, Jan, Hu, Bossart, & Wang, 2018) el 52% de los usuarios reutiliza sus contraseñas o utiliza una modificación muy básica de su contraseña en diferentes servicios online. Sus observaciones indican que los usuarios utilizan esas mismas contraseñas años después de que se hayan producido filtraciones. Además, evaluaciones posteriores indican que la mayoría de las contraseñas pueden ser adivinadas en menos de diez intentos.

En el entorno empresarial existe el mismo problema en los servicios proporcionados a sus empleados aunque a una menor escala. Además del problema de gestión descrito anteriormente, el uso de ese tipo de credenciales está perdiendo cada vez más fuerza en favor de otros métodos de autenticación más seguros como los certificados digitales y los procesos de autenticación en varios pasos (Multifactor Authentication).

Los servicios proporcionados de manera telemática, tanto por empresas públicas como privadas, no pueden ofrecer el mismo nivel de acceso a usuarios autenticados con diferentes medidas de seguridad. Los procesos más críticos deben tener unos controles más estrictos de la identidad de los usuarios.

Ambos problemas tienen solución implementando un sistema de Single Sign On, en adelante SSO. Esto permite a un usuario autenticado mediante algún tipo de credencial acceder a todas las aplicaciones que hagan uso de este sistema. Este sistema de credenciales únicas no impone un proceso de autenticación en concreto sino que deja en manos del equipo que lo implanta la decisión de los tipos de autenticación necesarios en cada caso.

En los siguientes apartados se discuten las distintas posibilidades que existen de implementación, sus ventajas e inconvenientes y como funcionarían en caso de necesidades de integración con aplicaciones de terceros.

Este proyecto pretende analizar, diseñar e implementar un sistema de autenticación que proporcione solución a las dos casuísticas expuestas. Por un lado, el sistema debe permitir a un usuario autenticarse en múltiples aplicaciones con un solo conjunto de credenciales. Por otro lado, las aplicaciones deben tener en cuenta las credenciales presentadas para permitir acceso a distintos procesos en función del nivel de seguridad requerido.

Además de las necesidades funcionales definidas, este trabajo tiene en cuenta otras dos necesidades de implantación u operacionales.

En primer lugar, el SSO no debe estar expuesto a internet desde la red interna de la compañía en la que se implanta. Debe contar con un mecanismo situado en la Demilitarized Zone, en adelante DMZ, que añade una capa más de seguridad a los sistemas de intranet, evitando el acceso directo a aplicaciones críticas desde el exterior.

En segundo lugar, el SSO debe contar con la característica de alta disponibilidad, en adelante HA de sus siglas en inglés (High Availability). Esta característica de una plataforma software asegura que, aunque existan problemas técnicos en alguna parte del sistema, cuenta con mecanismos de redundancia que le permitan seguir manteniendo un nivel de servicio aceptable.

1.2 Objetivos

El objetivo general de este trabajo es analizar, diseñar e implementar un sistema de autenticación Single Sign On que dé respuesta a los problemas planteados en la introducción de este documento.

Los objetivos específicos de este trabajo son:

- Analizar la plataforma SSO para cubrir las necesidades de autenticación.
- Analizar el balanceador de carga para proporcionar HA a la plataforma SSO.
- Analizar el reverse proxy para aumentar la seguridad de todo el sistema.
- Implantar en un entorno de pruebas todas las soluciones seleccionadas para el sistema.
- Configurar los procesos de autenticación para dar soporte a credenciales de tipo certificado.
- Configurar los procesos de autenticación para dar soporte a credenciales de tipo usuario y contraseña.
- Analizar, diseñar e implementar una aplicación piloto.
- Integrar la aplicación piloto en el sistema.
- Hacer las pruebas funcionales necesarias para dar validez a la solución implementada.

1.3 Metodología

En los proyectos de desarrollo de software existen dos metodologías predominantes: en cascada y Agile.

Por un lado, la metodología en cascada define una serie de fases en las que se divide el proyecto de desarrollo. Por norma general, los equipos solamente pasan a la siguiente fase cuando la anterior está finalizada. Esta metodología es muy sencilla de implementar ya que cada fase cuenta con unos entregables específicos lo que facilita la gestión de dependencias entre distintas tareas del proyecto.

La metodología en cascada está enfocada a equipos pequeños donde los requisitos del proyecto están muy bien definidos. Tiene la desventaja de que es difícil revertir decisiones tomadas en fases anteriores. Teniendo en cuenta que las pruebas se realizan al final de la fase de implementación, los errores de análisis o diseño pueden ser muy costosos. Esta situación hace que el riesgo asociado al uso de esta metodología se incremente según avanzan las fases.

Por otro lado, la metodología Agile promueve el desarrollo, pruebas y despliegue de pequeñas iteraciones sobre un proyecto. Esta metodología está enfocada a involucrar a todas las partes interesadas a la hora de definir qué se implementa y cómo se ejecutan los trabajos.

La metodología Agile implica, especialmente, a los clientes o usuarios finales del software lo que reduce drásticamente los riesgos encontrados en proyectos poco definidos. Debido a su naturaleza es fácil detectar problemas y revertir cambios en subsiguientes iteraciones. Una limitación de esta metodología es la necesidad de contar con un experto (Scrum master) en las reuniones con el objetivo de tomar decisiones importantes y guiar al resto de los integrantes del equipo en la implementación de la propia metodología.

En este trabajo se toma la decisión de utilizar la metodología en cascada por adaptarse más a un proyecto bien definido y que cuenta con un equipo de trabajo pequeño. Además, cuenta con la ventaja de que las distintas fases se pueden asociar con las entregas parciales que han de entregarse durante el desarrollo de este trabajo.

Según esta metodología se ejecutarán las siguientes fases:

- **Toma de requisitos:** es la fase inicial del proyecto en la que se define el alcance y funcionalidades que ha de tener el proyecto final.
- **Análisis y diseño:** en esta fase, partiendo de los requisitos definidos, se analiza y diseña el sistema en cuanto a componentes y procesos.
- **Implementación:** esta etapa comprende la implementación de los procesos que dan respuesta a los requisitos.
- **Pruebas:** una vez finalizada la fase de implementación, se validan las funcionalidades desarrolladas y se verifica el grado de completación de los requisitos.

1.4 Estado del arte

Antes de empezar con las siguientes fases del trabajo, se hace una investigación de las opciones existentes para los distintos componentes del sistema con la intención de seleccionar las tecnologías y plataformas que más se adecúen al trabajo.

Solución SSO

Actualmente existen varias plataformas que proveen servicios de SSO para aplicaciones web. Entre ellas destacan:

- **OpenAM:** es una plataforma de la compañía ForgeRock que provee servicios de SSO para aplicaciones web. Es compatible con distintos formatos de autenticación y provee autorización basada en XACML. Implementa varios estándares como OAuth2, OIDC y SAML.
- **CAS Enterprise SSO:** el proyecto Apereo Central Authentication Service (CAS) es una plataforma de SSO para aplicaciones web que cuenta con soluciones de autenticación y autorización. CAS es, además, un protocolo de autenticación cuya implementación principal es la propia solución de SSO de Apereo. La plataforma soporta otros protocolos como OAuth, OIDC y SAML. El código del servidor CAS es abierto.
- **Shibboleth identity provider:** la plataforma identity provider de shibboleth está desarrollada por el Shibboleth Consortium. Al igual que las otras dos plataformas, provee servicios de SSO para aplicaciones web. Soporta los mismos protocolos, incluyendo CAS. Como en el caso de CAS su código es abierto.

Es necesario mencionar otras plataformas que no aplicarían en este proyecto como Azure Active Directory que dan servicios de SSO a aplicaciones desplegadas en la nube.

Solución HA

Existen múltiples soluciones open source para proveer al sistema de un balanceador de carga que permita una arquitectura HA. Las soluciones expuestas a continuación se basan en software. Se excluyen las soluciones hardware y cloud.

- **HAProxy:** es una de las opciones más usadas para proporcionar HA, proxy y capacidades de balanceo de carga. Soporta: IPv6, sockets de Unix, compresión, comprobación de estado y sesiones “sticky” entre otras características.
- **Træfik:** es un reverse proxy y balanceador de carga orientado al despliegue sencillo de microservicios. Soporta Websockets, HTTP/2, y renovación automática de certificados entre otras funcionalidades. Cuenta con una interfaz web para gestionar y monitorizar los recursos. Træfik ha sido creado por Containous.
- **Gobetween:** es un balanceador de carga pequeño pero que provee un alto rendimiento en nivel L4 de TCP, TLS & UDP. Gobetween es de código abierto.

Solución DMZ

Algunas de las soluciones vistas en el apartado de balanceador de carga también cuentan con funciones de proxy reverso. En este apartado se destaca la existencia de una solución específica que se integra dentro del paquete de soluciones de ForgeRock, empresa de OpenAM.

- **OpenIG:** es una solución basada en estándares que se puede utilizar como punto de forzado de políticas para aplicaciones web, APIs Rest y dispositivos. Entre sus características se encuentran: SSO y sign-off, forzado de autorización, limitación de peticiones, monitorización, auditorías, etc.

1.5 Planificación de tareas

| | Nombre | Duración | Inicio | Terminado |
|----|--|----------|----------------|-----------------|
| 1 | <input type="checkbox"/> Plan de trabajo | 9 days? | 19/02/20 8:00 | 3/03/20 8:00 |
| 2 | Contexto | 1 day? | 19/02/20 8:00 | 19/02/20 17:00 |
| 3 | Objetivos | 0,5 days | 20/02/20 8:00 | 20/02/20 13:00 |
| 4 | Metodología | 0,5 days | 20/02/20 13:00 | 20/02/20 17:00 |
| 5 | Estado del arte | 1 day | 21/02/20 8:00 | 21/02/20 17:00 |
| 6 | <input type="checkbox"/> Planificación | 1,5 days | 24/02/20 8:00 | 25/02/20 13:... |
| 7 | Definición de tareas | 1 day | 24/02/20 8:00 | 24/02/20 17:00 |
| 8 | Elaboración Gantt | 0,5 days | 25/02/20 8:00 | 25/02/20 13:00 |
| 9 | Planificación presupuestaria | 0,5 days | 25/02/20 13:00 | 25/02/20 17:00 |
| 10 | Entrega PEC 1 | 0 days | 3/03/20 8:00 | 3/03/20 8:00 |
| 11 | <input type="checkbox"/> Análisis | 23 days? | 27/02/20 8:00 | 31/03/20 8:00 |
| 12 | Definición de requisitos | 0,5 days | 27/02/20 8:00 | 27/02/20 13:00 |
| 13 | Definición de pruebas | 0,5 days | 27/02/20 13:00 | 27/02/20 17:00 |
| 14 | Análisis de plataforma SSO | 2 days | 28/02/20 8:00 | 2/03/20 17:00 |
| 15 | Análisis de balanceador de carga | 2 days | 3/03/20 8:00 | 4/03/20 17:00 |
| 16 | Análisis de gateway | 2 days | 5/03/20 8:00 | 6/03/20 17:00 |
| 17 | Diseño de infraestructura despliegue | 2 days | 9/03/20 8:00 | 10/03/20 17:00 |
| 18 | Diagrama de componentes | 1 day? | 11/03/20 8:00 | 11/03/20 17:00 |
| 19 | Diagrama de despliegue | 1 day? | 12/03/20 8:00 | 12/03/20 17:00 |
| 20 | Redacción PEC2 | 6 days | 13/03/20 8:00 | 20/03/20 17:00 |
| 21 | Entrega PEC2 | 0 days | 31/03/20 7:00 | 31/03/20 8:00 |
| 22 | <input type="checkbox"/> Implementación | 26 days? | 23/03/20 8:00 | 28/04/20 8:00 |
| 23 | Configuración máquina virtual | 2 days | 23/03/20 8:00 | 24/03/20 17:00 |
| 24 | Configuración infraestructura | 2 days | 25/03/20 8:00 | 26/03/20 17:00 |
| 25 | Instalación plataforma SSO HA | 3 days | 27/03/20 8:00 | 31/03/20 17:00 |
| 26 | Configuración plataforma SSO | 1 day? | 1/04/20 8:00 | 1/04/20 17:00 |
| 27 | Instalación gateway | 3 days | 2/04/20 8:00 | 6/04/20 17:00 |
| 28 | Configuración gateway | 1 day? | 7/04/20 8:00 | 7/04/20 17:00 |
| 29 | Configuración canales SSL | 5 days | 8/04/20 8:00 | 14/04/20 17:00 |
| 30 | Implementación aplicación piloto | 1 day? | 15/04/20 8:00 | 15/04/20 17:00 |
| 31 | Ejecución de casos de prueba | 1 day? | 16/04/20 8:00 | 16/04/20 17:00 |
| 32 | Redacción PEC3 | 6 days | 17/04/20 8:00 | 24/04/20 17:00 |
| 33 | Entrega PEC3 | 0 days | 28/04/20 7:00 | 28/04/20 8:00 |
| 34 | <input type="checkbox"/> Memoria | 26 days | 27/04/20 8:00 | 2/06/20 8:00 |
| 35 | Redacción de memoria final | 15 days | 27/04/20 8:00 | 15/05/20 17:00 |
| 36 | Preparación entregables | 2 days | 18/05/20 8:00 | 19/05/20 17:00 |
| 37 | Entrega memoria | 0 days | 2/06/20 8:00 | 2/06/20 8:00 |
| 38 | <input type="checkbox"/> Vídeo presentación | 14 days | 20/05/20 8:00 | 9/06/20 8:00 |
| 39 | Guión | 2 days | 20/05/20 8:00 | 21/05/20 17:00 |
| 40 | Producción | 2 days | 22/05/20 8:00 | 25/05/20 17:00 |
| 41 | Entrega vídeo presentación | 0 days | 9/06/20 7:00 | 9/06/20 8:00 |

Figura 1 – Listado de tareas

| Nombre | Duración | Inicio | Terminado |
|---|------------------------|----------------|----------------|
| 1 El Plan de trabajo | 9 días ^{se} | 19/02/20 8:00 | 31/03/20 8:00 |
| 2 Contexto | 1 día [?] | 19/02/20 8:00 | 19/02/20 13:00 |
| 3 Objetivos | 0,5 días | 20/02/20 8:00 | 20/02/20 13:00 |
| 4 Metodología | 0,5 días | 20/02/20 13:00 | 20/02/20 17:00 |
| 5 Estado del arte | 1 día [?] | 21/02/20 8:00 | 21/02/20 17:00 |
| 6 El Planificación | 1,5 días ^{se} | 24/02/20 8:00 | 25/02/20 13:00 |
| 7 Definición de tareas | 1 día [?] | 24/02/20 8:00 | 24/02/20 17:00 |
| 8 Elaboración Gantt | 0,5 días | 25/02/20 8:00 | 25/02/20 13:00 |
| 9 Planificación presupuestaria | 0,5 días | 25/02/20 13:00 | 25/02/20 17:00 |
| 10 Entrega PEC1 | 0 días ^{se} | 3/03/20 8:00 | 3/03/20 8:00 |
| 11 El Análisis | 23 días ^{se} | 27/02/20 8:00 | 31/03/20 8:00 |
| 12 Definición de requisitos | 0,5 días | 27/02/20 8:00 | 27/02/20 13:00 |
| 13 Definición de pruebas | 0,5 días | 27/02/20 13:00 | 27/02/20 17:00 |
| 14 Análisis de plataforma SSO | 2 días | 28/02/20 8:00 | 2/03/20 17:00 |
| 15 Análisis de balanceador de carga | 2 días | 3/03/20 8:00 | 4/03/20 17:00 |
| 16 Análisis de gateway | 2 días | 5/03/20 8:00 | 6/03/20 17:00 |
| 17 Diseño de infraestructura despliegue | 2 días | 9/03/20 8:00 | 10/03/20 17:00 |
| 18 Diagrama de componentes | 1 día [?] | 11/03/20 8:00 | 11/03/20 17:00 |
| 19 Diagrama de despliegue | 1 día [?] | 12/03/20 8:00 | 12/03/20 17:00 |
| 20 Redacción PEC2 | 6 días | 13/03/20 8:00 | 20/03/20 17:00 |
| 21 Entrega PEC2 | 0 días ^{se} | 31/03/20 7:00 | 31/03/20 8:00 |
| 22 El Implementación | 26 días ^{se} | 23/03/20 8:00 | 28/04/20 8:00 |
| 23 Configuración máquina virtual | 2 días | 23/03/20 8:00 | 24/03/20 17:00 |
| 24 Configuración infraestructura | 2 días | 25/03/20 8:00 | 26/03/20 17:00 |
| 25 Instalación plataforma SSO HA | 3 días | 27/03/20 8:00 | 31/03/20 17:00 |
| 26 Configuración plataforma SSO | 1 día [?] | 1/04/20 8:00 | 1/04/20 17:00 |
| 27 Instalación gateway | 3 días | 2/04/20 8:00 | 6/04/20 17:00 |
| 28 Configuración gateway | 1 día [?] | 7/04/20 8:00 | 7/04/20 17:00 |
| 29 Configuración canales SSL | 5 días | 8/04/20 8:00 | 14/04/20 17:00 |
| 30 Implementación aplicación piloto | 1 día [?] | 15/04/20 8:00 | 15/04/20 17:00 |
| 31 Ejecución de casos de prueba | 1 día [?] | 16/04/20 8:00 | 16/04/20 17:00 |
| 32 Redacción PEC3 | 6 días | 17/04/20 8:00 | 24/04/20 17:00 |
| 33 Entrega PEC3 | 0 días ^{se} | 28/04/20 7:00 | 28/04/20 8:00 |
| 34 El Memoria | 26 días ^{se} | 27/04/20 8:00 | 2/06/20 8:00 |
| 35 Redacción de memoria final | 15 días | 27/04/20 8:00 | 15/05/20 17:00 |
| 36 Preparación entregables | 2 días | 18/05/20 8:00 | 19/05/20 17:00 |
| 37 Entrega memoria | 0 días | 2/06/20 8:00 | 2/06/20 8:00 |
| 38 El Vídeo presentación | 14 días ^{se} | 20/05/20 8:00 | 9/06/20 8:00 |
| 39 Guión | 2 días | 20/05/20 8:00 | 21/05/20 17:00 |
| 40 Producción | 2 días | 22/05/20 8:00 | 25/05/20 17:00 |
| 41 Entrega vídeo presentación | 0 días ^{se} | 9/06/20 7:00 | 9/06/20 8:00 |

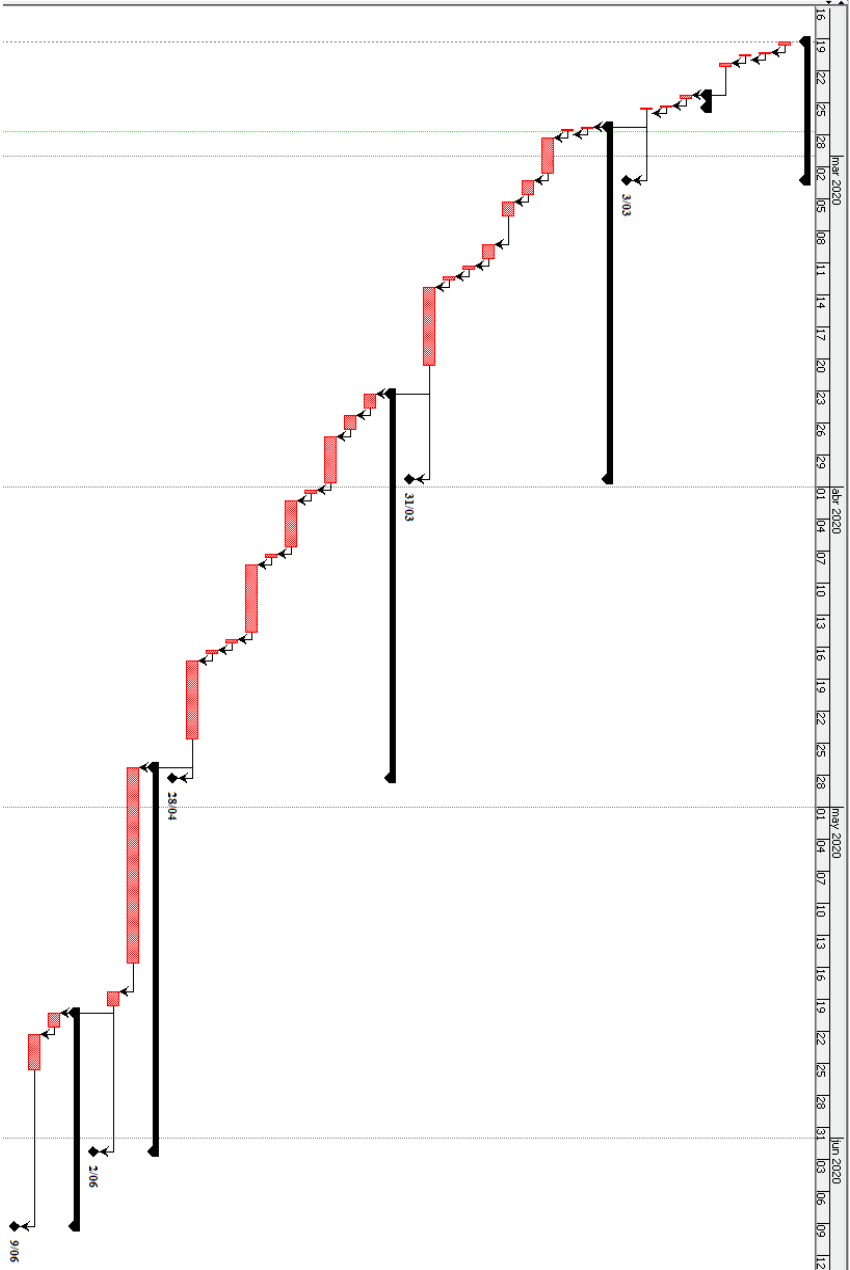


Figura 2 – Planificación de tareas

1.6 Análisis de riesgos

La guía de PMBOK define riesgo como: “An uncertain event or condition, that if it occurs, has a positive or negative effect on a project's objective”. La clave de esta descripción es que no se sabe el evento y ni si quiera si este afectará negativamente al proyecto. (Weaver, 2008)

Para analizar los riesgos de un proyecto debemos definir los conceptos de impacto y probabilidad.

- El primero se refiere a la magnitud de desviación sobre el plazo, el esfuerzo o el coste planificado para el proyecto.
- El segundo se refiere a la probabilidad de que ese riesgo se produzca a lo largo de la ejecución del proyecto.

Con estas dos magnitudes se calcula la exposición del proyecto al riesgo según la siguiente tabla

| Impacto\Probabilidad | Alta | Media | Baja |
|-----------------------------|-------------|--------------|-------------|
| <i>Alto</i> | Alto | Alto | Medio |
| <i>Medio</i> | Alto | Medio | Bajo |
| <i>Bajo</i> | Medio | Bajo | Bajo |

Una vez definidos los riesgos y su exposición deben definirse las estrategias de gestión para intentar reducir el nivel de exposición. Existen varios tipos de estrategias de reducción de la exposición: prevención, minimización, contención, transferencia y aceptación. (Lavanya & Malarvizhi, 2008)

A continuación se enumeran los riesgos que se identifican en este proyecto:

Riesgo 1 - Alcance del proyecto

| | |
|---------------------|---|
| <i>Descripción</i> | El alcance del proyecto es más amplio de lo esperado. |
| <i>Impacto</i> | Alto |
| <i>Probabilidad</i> | Media |
| <i>Exposición</i> | Alta |
| <i>Estrategia</i> | Mitigación |
| | No dedicar más tiempo del planificado en las tareas y definir con el director los pasos a seguir. |

Riesgo 2 – Plataformas escogidas

| | |
|---------------------|--|
| <i>Descripción</i> | Las plataformas escogidas no son compatibles entre sí para la finalización del proyecto. |
| <i>Impacto</i> | Alto |
| <i>Probabilidad</i> | Baja |
| <i>Exposición</i> | Media |
| <i>Estrategia</i> | Minimizar |
| | Se seleccionan plataformas que tienen un bajo acoplamiento y que se basan en estándares. Se deben poder sustituir por otras con funcionalidades similares. |

Riesgo 3 – Implementación

| | |
|---------------------|--|
| <i>Descripción</i> | No se consigue la implementación de alguno de los objetivos del proyecto. |
| <i>Impacto</i> | Alto |
| <i>Probabilidad</i> | Media |
| <i>Exposición</i> | Alta |
| <i>Estrategia</i> | Contener |
| | Se explica en la memoria, de manera detallada, la teoría detrás de la implementación y se explica el motivo de la no consecución del objetivo. |

Riesgo 4 – Mala planificación

| | |
|---------------------|---|
| <i>Descripción</i> | Se subestiman las tareas del proyecto. |
| <i>Impacto</i> | Alto |
| <i>Probabilidad</i> | Baja |
| <i>Exposición</i> | Media |
| <i>Estrategia</i> | Prevenir |
| | Se hace un seguimiento de la planificación de manera semanal para identificar desviaciones y corregir lo antes posible. |

Riesgo 5 – Pérdida del trabajo realizado

| | |
|---------------------|--|
| <i>Descripción</i> | Se pierde el trabajo realizado hasta el momento por motivos técnicos. |
| <i>Impacto</i> | Alto |
| <i>Probabilidad</i> | Baja |
| <i>Exposición</i> | Media |
| <i>Estrategia</i> | Prevenir |
| | Se hacen copias de seguridad del trabajo hecho a la finalización de cada jornada en un almacenamiento alternativo al del trabajo habitual. |

Riesgo 6 – Indisponibilidad del equipo de trabajo

| | |
|---------------------|--|
| <i>Descripción</i> | El equipo de trabajo no está disponible debido a circunstancias externas al proyecto. |
| <i>Impacto</i> | Medio |
| <i>Probabilidad</i> | Baja |
| <i>Exposición</i> | Baja |
| <i>Estrategia</i> | Contener |
| | Se acepta que el riesgo se materializará y se intenta reducir su impacto teniendo en cuenta la indisponibilidad en la planificación. |

1.7 Planificación presupuestaria

Para calcular el presupuesto necesario para la ejecución de este proyecto se tiene en cuenta los costes de personal y materiales.

- Equipo de desarrollo:

$$1 \text{ persona} * 30 \frac{\text{€}}{\text{h}} * 450 \text{ h} = 13500 \text{ €}$$

- Conexión de internet:

$$38 \frac{\text{€}}{\text{mes}} * 4 \text{ meses} = 152 \text{ €}$$

- Amortización ordenador personal teniendo en cuenta una vida útil de 7 años (84 meses):

$$1200 \text{ €} * \left(\frac{4 \text{ meses}}{72 \text{ meses}} \right) \approx 67 \text{ €}$$

- Electricidad teniendo en cuenta que el proyecto no es el único consumidor:

$$15 \frac{\text{€}}{\text{mes}} * 4 \text{ meses} = 90 \text{ €}$$

Se ve claramente que el mayor porcentaje de coste se lo lleva el equipo de trabajo pero aun así no deben despreciarse el resto de costes asociados a un proyecto. El presupuesto final del proyecto será de unos **13809 €**.

2. Requisitos

En este apartado se analizan los requisitos del sistema. Se identifican tanto requisitos funcionales como no funcionales. Estos últimos pueden ser de seguridad, operativos, etc. Así mismo, se describen los casos de prueba que deberán ejecutarse al final de la fase de implementación para la validación final del sistema.

Para definir cada uno de los requisitos se usa la tabla siguiente:

| [ID] | [Nombre] |
|-------------|-----------------------|
| Descripción | |
| Prioridad | [Alta Media Baja] |

2.1 Funcionales

Se consideran requisitos funcionales aquellos que dan sentido a la existencia del propio sistema. Sin estos requisitos el proyecto no tendría un objetivo claro.

| | |
|--------------|---|
| RF-01 | Mostrar datos de acceso de usuario (nivel básico) |
| Descripción | Un usuario autenticado con medidas de seguridad básicas (usuario y contraseña) será capaz de acceder a una aplicación en la que se mostrarán los datos de acceso de ese mismo |

| | |
|------------------|----------|
| | usuario. |
| <i>Prioridad</i> | Alta |

RF-02 Mostrar datos de acceso de usuario (nivel avanzado)

| | |
|--------------------|---|
| <i>Descripción</i> | Un usuario autenticado con medidas de seguridad avanzadas (certificado) será capaz de acceder a una aplicación en la que se mostrarán los datos de acceso de ese mismo usuario. |
| <i>Prioridad</i> | Alta |

2.2 No funcionales

Se consideran requisitos no funcionales aquellos que dan soporte a los requisitos funcionales del sistema.

De seguridad

RS-01 Autenticación mediante certificado

| | |
|--------------------|--|
| <i>Descripción</i> | El sistema debe contar con un proceso de autenticación que permita el uso de certificados emitidos por una autoridad certificadora reconocida. Los usuarios autenticados mediante este tipo de credenciales tendrán acceso a todas las aplicaciones del sistema. |
| <i>Prioridad</i> | Alta |

RS-02 Autenticación mediante credenciales básicas

| | |
|--------------------|--|
| <i>Descripción</i> | El sistema debe contar con un proceso de autenticación que permita el uso de credenciales básicas. Los usuarios autenticados mediante este tipo de credenciales tendrán acceso restringido a partes del sistema. |
| <i>Prioridad</i> | Alta |

RS-03 Escalado de privilegios

| | |
|--------------------|--|
| <i>Descripción</i> | Un usuario autenticado mediante credenciales básicas tendrá la posibilidad de escalar sus privilegios presentando un certificado. Al autenticarse mediante certificado de manera correcta, el usuario tendrá acceso a partes del sistema que estaban restringidas previamente. |
| <i>Prioridad</i> | Media |

RS-04 Redes separadas

| | |
|--------------------|--|
| <i>Descripción</i> | Las partes más críticas del sistema (SSO y aplicaciones) deben estar desplegadas en una red interna (intranet). Los servicios expuestos en esta red solamente serán accesibles desde la DMZ. |
| <i>Prioridad</i> | Alta |

Operacionales

RO-01 Alta disponibilidad

| | |
|--------------------|---|
| <i>Descripción</i> | El sistema debe contar con mecanismos de redundancia para |
|--------------------|---|

| | |
|------------------|--|
| | asegurar que se mantienen un nivel de servicio adecuado en circunstancias desfavorables. |
| <i>Prioridad</i> | Media |

| | |
|--------------------|---|
| RO-02 | Auditoría |
| <i>Descripción</i> | El tanto el reverse proxy como la plataforma SSO deben contar con mecanismos de auditoría que permitan tener trazabilidad sobre las peticiones efectuadas al sistema. |
| <i>Prioridad</i> | Baja |

2.3 Casos de prueba

A continuación se describen los casos de prueba que darán validez al sistema. Estos casos de prueba están enfocados a medir el grado de cumplimiento de los requisitos definidos en el apartado anterior. Para la definición de los casos de prueba se usa la siguiente tabla:

| [ID] | [Nombre] |
|-----------------------------|-----------------|
| <i>Descripción</i> | |
| <i>Precondiciones</i> | |
| <i>Resultado esperado</i> | |
| <i>Requisitos cubiertos</i> | |

Funcionales

| | |
|-----------------------------|--|
| CP-01 | Acceso de nivel básico correcto |
| <i>Descripción</i> | Un usuario accede a la web protegida con autenticación básica. |
| <i>Precondiciones</i> | Usuario autenticado con cualquier tipo de credenciales |
| <i>Resultado esperado</i> | Se muestran los datos de acceso del usuario. |
| <i>Requisitos cubiertos</i> | RF-01 RF-02 |

| | |
|-----------------------------|--|
| CP-02 | Acceso de nivel avanzado correcto |
| <i>Descripción</i> | Un usuario accede a la web protegida con autenticación avanzada. |
| <i>Precondiciones</i> | Usuario autenticado con credenciales de tipo certificado. |
| <i>Resultado esperado</i> | Se muestran los datos de acceso del usuario. |
| <i>Requisitos cubiertos</i> | RF-02 |

No funcionales

| | |
|--------------------|---|
| CP-03 | Acceso mediante credenciales de tipo certificado |
| <i>Descripción</i> | Un usuario intenta acceder a las webs internas |

| | |
|-----------------------------|---|
| | autenticándose mediante certificado digital. |
| <i>Precondiciones</i> | |
| <i>Resultado esperado</i> | Se finaliza de manera correcta el proceso de autenticación. |
| <i>Requisitos cubiertos</i> | RS-01 |

CP-04 Acceso mediante credenciales de usuario

| | |
|-----------------------------|---|
| <i>Descripción</i> | Un usuario intenta acceder a las webs internas autenticándose mediante credenciales de usuario. |
| <i>Precondiciones</i> | |
| <i>Resultado esperado</i> | Se finaliza de manera correcta el proceso de autenticación. |
| <i>Requisitos cubiertos</i> | RS-02 |

CP-05 Acceso incorrecto

| | |
|-----------------------------|---|
| <i>Descripción</i> | Un usuario intenta acceder a las webs internas autenticándose con credenciales inválidas. |
| <i>Precondiciones</i> | |
| <i>Resultado esperado</i> | Se finaliza de manera errónea el proceso de autenticación y se muestra el correspondiente mensaje de error. |
| <i>Requisitos cubiertos</i> | RS-01 RS-02 |

CP-06 Escalado de privilegios

| | |
|-----------------------------|--|
| <i>Descripción</i> | Un usuario intenta acceder a la web que requiere autenticación avanzada. |
| <i>Precondiciones</i> | Usuario autenticado mediante usuario y contraseña. |
| <i>Resultado esperado</i> | Se redirige al usuario a la autenticación mediante certificado. |
| <i>Requisitos cubiertos</i> | RS-03 |

CP-07 Acceso a directo a intranet

| | |
|-----------------------------|--|
| <i>Descripción</i> | Un usuario intenta acceder a las webs internas sin pasar por el Gateway. |
| <i>Precondiciones</i> | |
| <i>Resultado esperado</i> | No se puede acceder a las webs internas. |
| <i>Requisitos cubiertos</i> | RS-04 |

CP-08 Resiliencia

| | |
|---------------------------|---|
| <i>Descripción</i> | Un usuario intenta acceder a una web interna. |
| <i>Precondiciones</i> | Se para uno de los nodos del sistema de SSO. |
| <i>Resultado esperado</i> | Se puede acceder a la web interna |

| | |
|-----------------------------|---|
| <i>Requisitos cubiertos</i> | RO-01 |
| CP-09 | Auditoría |
| <i>Descripción</i> | Un usuario intenta acceder a una web interna. |
| <i>Precondiciones</i> | |
| <i>Resultado esperado</i> | Se generan logs de auditoría que permiten trazar la petición del usuario. |
| <i>Requisitos cubiertos</i> | RO-02 |

3. Análisis de OpenAM

OpenAM proporciona un servicio integral de autenticación y autorización que permite controlar el acceso de sistemas online como aplicaciones y servicios web. La plataforma proporciona mecanismos para la gestión de usuarios, roles y políticas de acceso a recursos.

OpenAM centraliza la gestión de accesos haciéndose cargo de la autenticación, es decir, la identificación de un individuo, y la autorización que permite acceso a un recurso a los individuos identificados.

Para llevar a cabo la autenticación, OpenAM utiliza módulos de autenticación conectados a distintos repositorios de identidades. Los repositorios de identidades pueden estar implementados como directorios LDAP, servidores RADIUS, autenticación mediante credenciales de Windows, etc. La plataforma permite enlazar distintos módulos de autenticación para que el proceso sea más seguro. Los módulos enlazados forman cadenas o árboles de autenticación que pueden configurarse como el punto de autenticación por defecto para una parte de la plataforma.

Para ejecutar la autorización, OpenAM permite a los administradores la definición de políticas de acceso desacopladas de las aplicaciones que las definen y utilizan. Las políticas de autorización se basan en el estándar XACML (eXtensible Access Control Markup Language).

3.1 Arquitectura

La plataforma OpenAM se compone fundamentalmente de dos elementos: el frontal web y el backend (OpenDS).

El frontal web proporciona a los administradores de la plataforma las herramientas necesarias para configurar tanto la autenticación como la autorización. Así mismo, proporciona a los usuarios finales los formularios necesarios para identificarse dependiendo de la configuración definida por los administradores.

El backend, en adelante OpenDS o DS, es un producto propio de ForgeRock que puede ser instalado de manera independiente o embebido en la

configuración inicial de OpenAM. Esta última forma de configurar los DS solamente se recomienda para sistemas pequeños o pruebas de concepto. DS es un servicio de directorio, con conectividad a través de LDAP, que está desarrollado para ofrecer un alto rendimiento en operaciones de búsqueda y lectura. DS usa como base de datos una Berkeley DB Java Edition. Este tipo de bases de datos utilizan tecnología B-Tree. Se entra en detalle acerca de OpenDS más adelante en este documento.

Las instalaciones de DS pueden ser configuradas con tres propósitos distintos:

- **Almacén de identidades:** permite gestionar las identidades que pueden autenticarse en una instalación de OpenAM. Para configurar un DS con este objetivo, pueden definirse (ACIs), esquemas y políticas reseteo de contraseñas entre otras configuraciones.
- **Almacén de configuraciones:** persiste la configuración de la instalación de OpenAM. Contiene la configuración de los módulos, cadenas y árboles de autenticación así como otras configuraciones relevantes de la plataforma.
- **Almacén de tokens** (CTS o Core Token System): persiste los tokens necesarios para las sesiones de los usuarios así como los tokens de otros protocolos como OAuth2, OIDC y SAML. El tráfico en este tipo de configuración es mucho más dinámico, en cuanto a escrituras, por lo que el fabricante recomienda la configuración de uno (o varios) DS para la gestión de tokens de usuario en lugar del CTS embebido en la instalación de OpenAM.

Además de estos dos componentes principales, la arquitectura de OpenAM se complementa con otro tipo de aplicaciones satélite:

- **Amster CLI** (evolución de ssoadm). Es una herramienta de línea de comandos enfocada a la administración de la plataforma.
- **Agente de políticas java o web:** Son interceptores que se pueden desplegar junto a aplicaciones “standalone” o web y que harían las funciones PEP (XACML). Un PEP es un punto de forzado de políticas que puede obligar a un usuario a autenticarse o a contar con un nivel específico de autenticación. Los agentes desacoplan la lógica de autenticación y autorización de las aplicaciones finales de manera similar a como veremos con OpenIG.

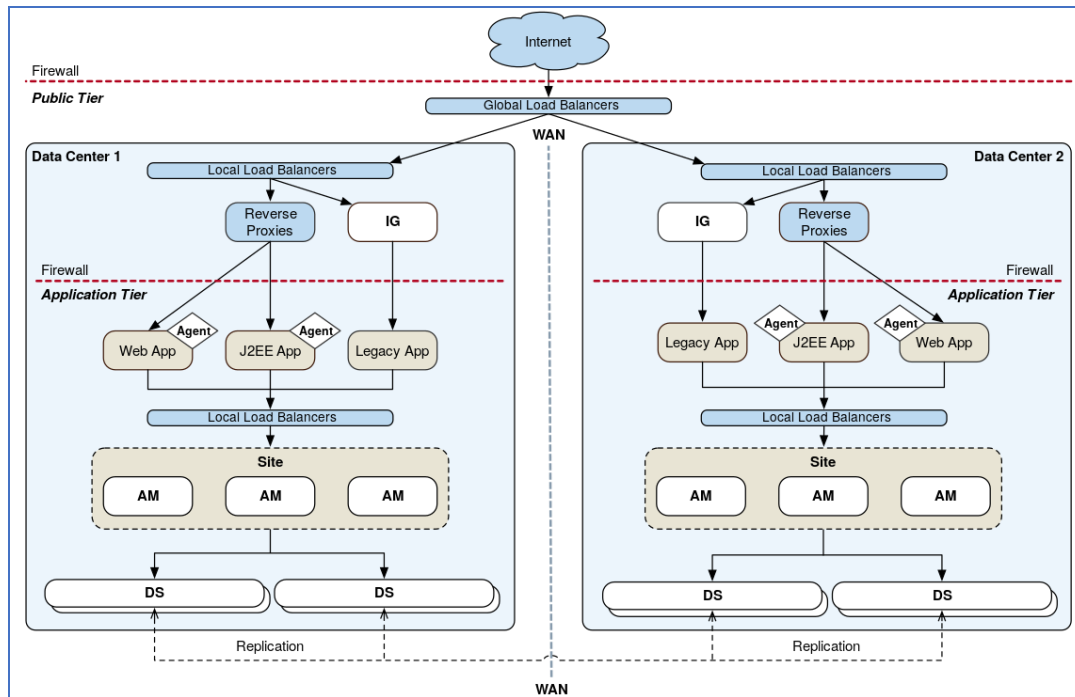


Figura 3 - Ejemplo de arquitectura OpenAM Multi-CPD

3.2 Autenticación

La autenticación es el proceso de validar las credenciales presentadas por un usuario contra un almacén de identidades.

OpenAM cuenta con diversas funcionalidades de autenticación que lo proveen de una gran flexibilidad a la hora de adaptarse a los requisitos de diferentes sistemas. Entre estas funcionalidades se puede destacar:

- **Árboles y nodos de autenticación:** AM proporciona nodos de autenticación para gestionar distintos tipos de autenticación. Los más relevantes para este trabajo son los de validación de credenciales básicas y certificados. Los nodos de AM se combinan en árboles que permiten la ejecución de distintos caminos de autenticación.
- **Módulos y cadenas de autenticación:** es un sistema análogo al de módulos y árboles aunque menos flexible. Desde la versión 6 de OpenAM se aconseja el uso de árboles y nodos.
- **Autenticación multifactor:** AM soporta autenticación multifactor que requiere a los usuarios presentar distintos tipos de credenciales. La combinación más usada para aplicaciones web es la de credenciales básicas y un OTP (One time password) generado a través de aplicaciones como Google Authenticator o RSA SecureId.
- **Nivel de autenticación:** En la configuración de módulos de autenticación, y en el caso de árboles, mediante un nodo específico, se puede definir un atributo de la sesión de usuario conocido como nivel de autenticación. Este nivel se puede utilizar para indicar el tipo de credenciales que un usuario ha presentado para identificarse. A partir de este dato, se pueden establecer políticas de acceso para restringir a los

usuarios identificados con credenciales más sencillas o básicas la visibilidad sobre ciertas aplicaciones críticas.

- **Bloqueo de cuentas:** AM implementa un sistema por el cual las cuentas se bloquean después de un determinado número de intentos de autenticación fallidos.
- **Sesiones:** AM crea una sesión cuando un usuario se ha autenticado de manera correcta. En clientes de tipo web la sesión se mantiene a través de una cookie. Este punto se comenta en más detalle en el análisis del mecanismo de SSO.
- **SSO:** AM implementa un sistema mediante el cual se permite a un usuario utilizar un único conjunto de credenciales para acceder a múltiples aplicaciones de un mismo dominio.

3.3 Sesiones y SSO

Cuando un usuario es autenticado de manera correcta AM crea una sesión para ese usuario. AM usa la información guardada en la sesión para determinar si la autenticación de ese usuario sigue siendo válida o necesita reautenticarse.

OpenAM tiene dos tipos de sesiones:

- **CTS-Based:** este tipo de sesiones se guardan en un DS configurado como CTS. Cuando un usuario se autentica correctamente se le devuelve una referencia a esta sesión. Esta referencia, conocida como token SSO, no contiene ninguna información de la propia sesión. Lo más habitual es que se envíe a un cliente web en forma de cookie pero otros tipos de cliente pueden acceder a ella como respuesta a la autenticación a través de la URL “/json/authenticate”.
- **Client-based:** en este tipo de sesiones se envía el estado en la respuesta a cada petición y este, a su vez, tiene que ser enviado en las siguientes peticiones.

Solamente las sesiones de tipo CTS-Based son modificables por OpenAM, es decir, según pasa el tiempo y ciertos eventos, sus propiedades pueden cambiar. Esto permite que, según las políticas de autorización, los accesos de un usuario puedan cambiar, por ejemplo, debido al tiempo que ha pasado desde la autenticación. Además, hay otras características que solamente están disponibles en las sesiones de tipo CTS-Based:

- Escalado de privilegios.
- Límite de sesiones activas.
- Políticas de autorización basadas en propiedades actuales de la sesión.
- SSO multidominio

La cookie que utiliza OpenAM para guardar la sesión del usuario en el navegador web se llama por defecto *iPlanetDirectoryPro*. Al guardar la cookie en el navegador, en las siguientes llamadas, AM (y OpenIG) tiene la prueba de que el usuario se ha autenticado. El usuario podrá acceder a todos los servicios cuya autenticación es gestionada por la misma instalación de OpenAM sin

necesidad de volver a autenticarse. Esta es la manera en la que OpenAM implementa su sistema SSO.

AM tiene la capacidad de dar soporte SSO dentro de un único dominio y a través de múltiples dominios:

- CDSSO (Cross-domain SSO): proporciona SSO en múltiples dominios o en uno solo. Este tipo de SSO elimina la restricción de configurar SSO dependiente del DNS de AM y de los agentes java o web.
- Classic CDSSO: análogo al anterior pero para versiones anteriores a la 5.
- Classic SDSSO (Single-domain SSO): proporciona SSO en un solo dominio.

AM permite el escalado de privilegios a través de la mejora de sesiones. La mejora de sesiones implica que el usuario presente credenciales adicionales para conseguir acceso a recursos más restringidos. Esta característica se consigue mediante una composición de los distintos servicios de la plataforma en la que entran en juego las políticas de autorización, los árboles de autenticación y los agentes web o java.

La mejora de sesiones se basa en el nivel de autenticación. El nivel de autenticación es el mecanismo que utiliza OpenAM para definir qué tipo de credenciales ha presentado el usuario. En las cadenas de autenticación, el nivel de autenticación se define en cada módulo y a la sesión se le asigna el mayor nivel de entre los módulos en los que el usuario se haya autenticado correctamente. En los árboles de autenticación existe un nodo específico para añadir niveles según el flujo de autenticación que haya tomado el usuario.

Por último, la invalidación de sesiones también depende del tipo al que pertenece la cookie. En el caso de sesiones CTS-Based hay varios mecanismos para invalidarlas ya que son guardadas en un DS que es parte del sistema. Un usuario puede decidir salir de la sesión por su cuenta o un administrador puede decidir terminar la sesión de un usuario. Las invalidaciones también pueden producirse por exceder el tiempo de vida o de inactividad.

En ambos casos, AM intentará invalidar la cookie del navegador del usuario sobrescribiéndola con datos incorrectos. En el caso de las sesiones Client-Based, AM no garantiza la invalidación de la cookie así que como medida adicional guarda un listado de sesiones inválidas.

3.4 Configuración

OpenAM agrupa la configuración y las identidades en dominios (“realms”). AM asocia cada dominio a, por lo menos, un almacén de identidades y a una cadena de autenticación o árbol de autenticación. En los dominios también se configuran las políticas de autorización y los permisos para los administradores. En definitiva, cada dominio supone una configuración específica para los servicios que presta.

En la primera configuración de OpenAM se crea un dominio raíz ("/"). Este dominio permite acceder a los usuarios configurados en el almacén de identidades de la instalación inicial. El resto de dominios creados en la instalación heredan, por defecto, la mayoría de configuraciones definidas en el dominio raíz. Los subdominios son interesantes para definir distintas necesidades de autenticación y/o proveer de distintos servicios como federación, conversiones de tokens, agentes, etc.

La configuración por dominios está ligada directamente a la funcionalidad de SSO. Cuando un usuario se autentica en un dominio distinto al que se ha autenticado previamente se desconectará del dominio de SSO actual. Es decir, cuando un usuario se autentica en un dominio, automáticamente se desconecta del dominio en el que actualmente está autenticado (invalidación de sesión).

3.5 OpenDS

En este apartado se entra un poco más en detalle sobre las características, necesidad y funcionamiento de Directory Services de ForgeRock. Como se comenta en apartados anteriores, DS sirve de backend para las distintas funcionalidades proporcionadas por OpenAM.

Los servicios de directorio de OpenDS están basados en almacenes de clave valor como en un diccionario. Además de esta organización lógica de la información, existen índices que mejoran el rendimiento al ordenar las entradas del diccionario mediante distintos atributos (nombre, teléfono, etc.). OpenDS no permite a usuarios estándar hacer búsquedas para las que no existe índice. Los administradores de cada instalación deben asegurar que la información está correctamente indexada (existen herramientas específicas para estas tareas). El acceso a OpenDS está basado en LDAP aunque también proporciona acceso mediante llamadas REST.

Las entradas se organizan de manera jerárquica dentro del directorio. Esta organización es análoga a la encontrada en un sistema de ficheros de cualquier sistema operativo moderno. Cada entrada cuenta con un identificador único (Distinguished Name - DN), este DN se compone de los DN de los elementos superiores del árbol hasta la posición de la entrada separados por comas en formato Little-endian. A continuación se puede ver un ejemplo de una entrada de directorio correspondiente a un usuario:

```
dn: uid=bjensen,ou=People,dc=example,dc=com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: top
uid: bjensen
userpassword: hifalutin
facsimiletelephonenumber: +1 408 555 1992
givenname: Barbara
cn: Barbara Jensen
cn: Babs Jensen
telephonenumber: +1 408 555 1862
sn: Jensen
roomnumber: 0209
homeDirectory: /home/bjensen
mail: bjensen@example.com
L: San Francisco
ou: Product Development
ou: People
uidNumber: 1076
gidNumber: 1000
```

Figura 4 - Entrada directorio activo

Los servicios de directorio cuentan con esquemas que definen los atributos de las entradas que contiene. Estos esquemas pueden ser consultados mediante LDAP y modificados en tiempo de ejecución, al contrario que las bases de datos relacionales. La mayoría de estos esquemas son estándar y se comparten a lo largo de distintas instalaciones y compañías. El atributo “objectClass” de cada entrada indica que tipos de atributos debe contener.

Los servicios de directorio también cuentan con control de accesos mediante ACIs (Access control instructions). Estas instrucciones indican quien, como y cuando se puede acceder a cierta información almacenada. Las propias instrucciones de acceso son guardadas en el directorio por lo que se pueden modificar en tiempo de ejecución como cualquier otra entrada.

En escenarios de alta disponibilidad suele haber configuradas varias instancias de servicios de directorio tanto para almacén de usuarios como de tokens y configuración. Para estas circunstancias OpenDS cuenta con servicios de replicación de los cambios en las entradas. Las replications están preparadas para resolver la mayoría de conflictos que se pueden dar en escenarios no estándar sin tener que recurrir a resolución manual por parte del administrador del directorio.

La replicación de los servicios de directorio OpenDS se considera como de “consistencia eventual”. Este término indica que la información será consistente en algún momento pero no tiene porqué serlo inmediatamente después de modificar una entrada.

3.6 Gestión de usuarios

Los usuarios en la plataforma OpenAM puede gestionarse de distintas maneras:

- **Identity management:** OpenAM puede hacer uso del directorio activo en organizaciones que cuenten con un sistema de gestión de identidades establecido. El directorio activo corporativo puede ser utilizado como almacén de identidades externo.

En este tipo de organizaciones, la gestión de las cuentas de usuarios se realiza a través de políticas de enrolamiento de identidades definidas por los departamentos correspondientes. Estas políticas dan acceso a diferentes tipos de servicios internos dependiendo de los grupos de directorio activo a los que pertenece cada usuario. ForgeRock cuenta con su propia herramienta de gestión de identidades OpenIDM.

- **Autoservicio:** OpenAM provee de una interfaz en su consola de administración en la que el usuario puede hacer el enrolamiento por su cuenta. Como todas las partes de un dominio, los datos que un usuario debe indicar así como las medidas de seguridad que aplican en este proceso son configurables. Los sistemas de autoservicio incluyen procesos como la recuperación de cuenta o contraseña y el proceso de baja. La interfaz de autoservicio es actualizable para que cumpla con las restricciones de diseño de interfaces de usuario en caso de haberlas.
- **Interfaz REST:** OpenAM expone los servicios REST que usa su interfaz de administración para que puedan ser integrados otros sistemas existentes.
- **Interfaz LDAP (o HTTP a través de Amster):** OpenAM proporciona herramientas para poder hacer un volcado de la información de otros sistemas de manera automática así como herramientas para generar cuentas de usuarios de prueba.

4. Análisis de CAS

El proyecto CAS (Apereo Central Authentication Service) es una solución de Single Sign On empresarial para aplicaciones web. CAS, al igual que OpenAM, es una plataforma web basada en tecnología Spring (Java) que soporta:

- Distintos tipos de autenticación.
- Múltiples protocolos: OpenID, OAuth2, CAS,...
- Autenticación multi-factor.
- Autenticación delegada.
- Gestión de passwords, notificaciones, condiciones de uso e impersonación.
- Monitorización
- Clientes multiplataforma.

4.1 Arquitectura

La arquitectura de CAS se compone principalmente de dos elementos, los servidores y los clientes.

Los servidores de CAS son servlets construidos a partir de Spring Framework que tienen como responsabilidad principal autenticar usuario y permitir acceso a clientes integrados con CAS emitiendo y validando tickets.

El término cliente CAS tiene dos significados distintos. Por un lado, un cliente CAS es una aplicación que puede comunicarse con el servidor mediante protocolos. Por otro lado, un cliente cas es también un software que puede ser integrado en varias plataformas y aplicaciones para poder comunicarse con el servidor CAS a través de distintos protocolos. En general, cuando se referencia a un cliente CAS, se refiere a los componentes de integración como un cliente CAS Java más que a una aplicación que delega en un servidor CAS para gestionar autenticación y autorización.

Los servidores CAS, como se puede ver en el diagrama de arquitectura (figura-5), se pueden dividir en tres componentes distintos: Web, Ticketing y Autenticación que aglutinan todas las consideraciones de despliegue y configuración de componentes de la plataforma.

El componente web es el endpoint para la comunicación con todos los sistemas externos incluidos los clientes CAS. La capa web delega en el sistema de ticketing la generación de tickets para los clientes. La sesión SSO, como se verá en un apartado de este documento, comienza con la generación de un ticket para la generación de tickets (Ticket granting ticket, en adelante TGT) después de una autenticación exitosa. Debido a este punto, el subsistema de ticketing delega en el subsistema de autenticación. La capa de autenticación entra en juego, habitualmente, al inicio de una sesión SSO.

La capa de Ticketing se compone fundamentalmente de dos piezas:

- TicketRegistry: provee la capa de persistencia necesaria para guardar las sesiones de usuario. Existen varios tipos de TicketRegistry siendo el que gestiona tickets en memoria solamente recomendable para instalaciones más pequeñas y los basados en cachés, colas de mensajes y bases de datos los recomendables en caso de instalaciones con más demanda.
- ExpirationPolicy: provee flujos de expiración de tickets de sesión.

El proceso de autenticación en CAS está controlado por el Authentication Manager encargado de orquestar distintos gestores de autenticación. Cuando se produce un intento de autenticación el Manager recorre la lista de gestores de autenticación para comprobar cuál de ellos es capaz de manejar la autenticación en base a las credenciales presentadas por el usuario. Si se produce una autenticación correcta se intenta resolver el principal del usuario en base a los resolutores configurados. Una vez resuelto el principal ya sea a

través de un resolutor o a través del propio gestor de autenticación, se aplican las políticas de autenticación correspondientes.

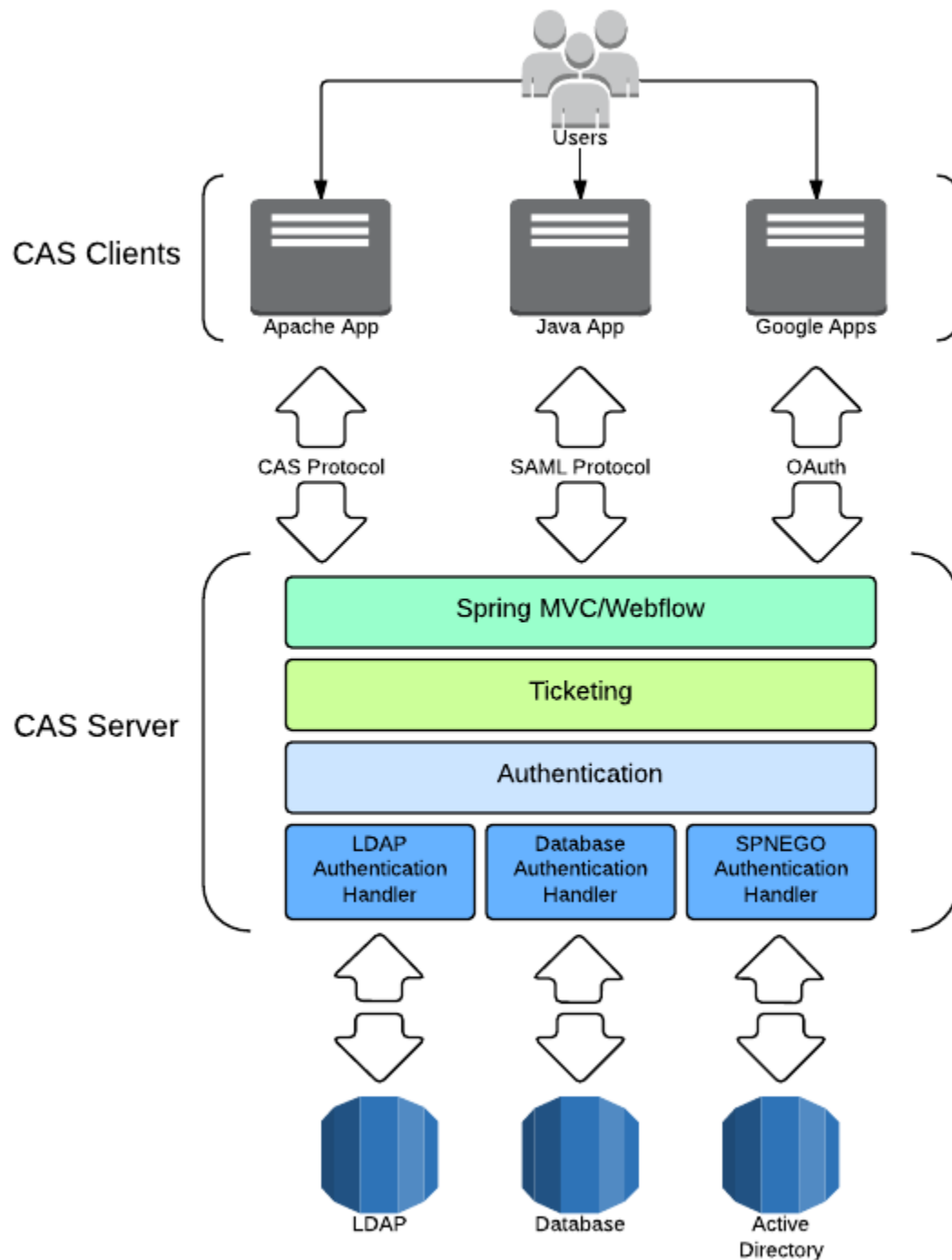


Figura 5 - Arquitectura CAS

4.2 Sesiones y SSO

Una sesión SSO se crea cuando el servidor de CAS genera un TGT para el usuario después de una autenticación correcta. El TGT se incorpora en una cookie y se establece al final del proceso de autenticación. La cookie mantiene el estado de logueo del usuario para ese cliente y, mientras tiene validez, el cliente puede presentársela a CAS en lugar de sus credenciales.

La autenticación y por ende la creación de sesiones se basa en el protocolo CAS. El protocolo CAS se basa en dos conceptos principales:

- TGT, representa la sesión del usuario.
- ST (Service Ticket), es usado por el servidor CAS para permitir acceso a una aplicación “CASificada” para un usuario específico.

Después de la creación de una sesión SSO, el servidor expide un TGT. A continuación, un ST es creado para una aplicación en base a peticiones de un usuario a través de redirecciones en el navegador usando el TGT como token. El ST es validado a través de una comunicación directa entre la aplicación a la que se quiere acceder y el servidor CAS.

Usando el mismo TGT el usuario será capaz de acceder a cualquier aplicación integrada con el servidor CAS. Se generará para cada una de estas aplicaciones su correspondiente ST que tendrán que validar contra el servidor.

CAS no pretende ser un gestor de sesiones de aplicación, la responsabilidad de la sesión de la aplicación es responsabilidad de la propia aplicación. CAS simplemente se ocupa de mantener el control de la sesión de SSO a través de la gestión del TGT.

Para resolver el “Single Sign Out” CAS puede invalidar sesiones asociadas al TGT cuando este es invalidado explícitamente. CAS intenta notificar a cada aplicación que la sesión de SSO ya no es válida y que las sesiones de aplicación deben ser invalidadas. Puede darse el caso que las aplicaciones no soporten estas notificaciones. En este caso, CAS no hace nada más al respecto.

4.3 Autorización

La autorización en CAS se lleva a cabo a través del sistema conocido como estrategias de acceso a servicios. Este sistema provee control de grano fino sobre las reglas de autorización de las aplicaciones. Describe si se permite a un servicio a utilizar el servidor CAS, participar en autenticaciones SSO, etc. Adicionalmente, puede ser configurado para obligar a que los usuarios solamente puedan acceder al servicio en base a una serie de atributos de su identidad.

5. Análisis de HAProxy

Los balanceadores de carga, como HAProxy, tienen como objetivo agregar múltiples componentes para conseguir una capacidad de proceso mayor que la de cada componente por separado. Además de la mejora de rendimiento conseguida para altos volúmenes de tráfico también se consiguen otros beneficios como alta disponibilidad al implementar uno de estos sistemas.

Un balanceador de carga puede actuar a varios niveles del modelo OSI. HAProxy actúa a nivel de capa de transporte para los propósitos de este trabajo. Habitualmente los balanceadores de carga son desplegados en un servidor específico lo que hace que no haya demasiados cambios en arquitecturas ya consolidadas.

En un sistema de alta disponibilidad, el elevado número de componentes aumenta el riesgo de fallo de alguno de ellos. Por este motivo los balanceadores de carga necesitan comprobar el estado de los servidores para los que ejercen de frontal. Si se detecta que alguno de los servidores no está respondiendo de manera adecuada se deja de enviar tráfico hasta que este servidor esté de nuevo operativo. Existen varios métodos que proporcionan este tipo de verificación de manera automática:

- Health checks: con este método el balanceador envía señales para comprobar que los servidores siguen funcionando.
- Comprobación de tráfico: con este método se revisan las respuestas de los servidores para comprobar que son las esperadas. Tiene la desventaja de que es necesario revisar todas las respuestas para comprobar el estado de los servidores. Esto implica una mayor carga para el balanceador.

Los balanceadores de carga de nivel 7 (o de aplicación) pueden ser configurados para persistencia o stickiness, es decir, la capacidad que tienen de dirigir siempre las peticiones de un mismo origen a un mismo servidor, habitualmente por motivos de rendimiento o funcionales. Este mecanismo es similar a las cookies de balanceo que utiliza OpenAM para sus “sticky sessions”.

Teniendo en cuenta todas las características que puede implementar un sistema como este, es recomendable que se implemente algún tipo de sistema de auditoría para poder diagnosticar los posibles errores.

HAProxy puede ejecutar distintos roles dentro de un sistema. Estos roles pueden ser:

- TCP proxy
- HTTP reverse-proxy
- SSL terminator
- Normalizador TCP
- Normalizador HTTP

- Corrector de HTTP
- Switch basado en contenido
- Balanceador de capa 7
- Regulador de tráfico
- Protección contra DDoS
- Punto de observación para diagnóstico de problemas de red
- Compresor de HTTP

5.1 Arquitectura

HAProxy es un motor asíncrono dirigido por eventos con un único hilo de ejecución. Combina una capa de IO muy rápida con un planificador basado en prioridades. La arquitectura de esta plataforma está optimizada para mover los datos de la manera más rápida posible, con el mínimo número de operaciones necesarias. La mayoría del procesamiento hecho por HAProxy se ejecuta en el kernel. La plataforma se encarga de que el kernel ejecute las operaciones de la manera más eficiente posible.

HAProxy solamente necesita el fichero ejecutable y el fichero de configuración para funcionar. Se recomienda que para el sistema de logs comentado anteriormente se instale, además, un daemon de syslog y se configure la rotación de logs. El fichero de configuración se analiza y parsea antes de arrancar el ejecutable. Si el fichero no es correcto, HAProxy no arranca. Una vez arrancado, el fabricante asegura que no produce ningún fallo hasta que se pare el proceso. Durante su funcionamiento HAProxy solamente ejecuta 3 funciones:

- Proceso de peticiones entrantes.
- Health checks.
- Cambio de información con otros HAProxy si los hubiese.

Para el trabajo actual, el fichero de configuración de HAProxy deberá definir un frontend y un backend. La configuración de frontend define en que puerto e IP HAProxy escuchará peticiones y a que backend debe enviarlas. La configuración de backend indica el tipo de balanceo, los servidores (IP y puerto) que forman parte de ese backend y cuáles son los controles de Health Check.

5.2 Procesado de peticiones

El proceso de peticiones entrantes es el más complejo de todos los que HAProxy lleva a cabo. Este proceso se puede resumir en varios pasos:

- Aceptar la conexión de los sockets que pertenecen al frontend.
- Aplicar las reglas del frontend a las peticiones.
- Enviar las peticiones al correspondiente backend.
- Aplicar las reglas del backend a las peticiones.
- Decidir el servidor destino de la petición.
- Aplicar las reglas del backend a las respuestas.

- Aplicar las reglas del frontend a las respuestas.
- Generar una entrada en los ficheros de log.
- En HTTP, volver al paso de escuchar conexiones, si no, cerrar la conexión.

Los conceptos de frontend y backend son considerados como medio proxies ya que cada uno se encarga de una sola parte de una conexión de punto a punto. Los frontends solamente se encargan de los clientes y los backends se encargan de los servidores. HAProxy también soporta proxies completos que son el resultado de unir un frontend con un backend.

5.3 Funcionalidades

De entre las funcionalidades de HAProxy destacamos las que más afectan a este trabajo: Alta disponibilidad, balanceo de carga y stickiness.

HAProxy pone mucho esfuerzo en la alta disponibilidad de los servicios a los que hace de proxy. Se encarga de que solamente los servidores válidos sean usados y aquellos que necesiten ser detenidos lo puedan hacer de una manera ordenada para no afectar a ninguna petición. Los servidores de backup se añaden al balanceo de forma automática cuando los principales no están disponibles. HAProxy tiene la capacidad de devolver un estado general de no error cuando varios de los servidores de backend no están activos.

En cuanto al balanceo de carga, HAProxy soporta nueve tipos distintos, desde el round robin más básico hasta los apoyados en cabeceras de la petición o en las URIs solicitadas. Independientemente del algoritmo, se pueden aplicar pesos a cada servidor en caso de que sea necesario desviar más o menos tráfico a alguno de ellos. Además, en algunos tipos de balanceo como round robin se pueden aplicar pesos dinámicos.

El balanceo de carga no sería posible sin la capacidad de stickiness pegajosidad. HAProxy proporciona una lista bastante amplia de posibilidades para mantener las peticiones de un cliente contra un mismo servidor, incluso cuando los servidores son parados y arrancados. En general se pueden aplicar reglas de adhesión a un servidor en función de cualquier parte de la petición.

6. Análisis de OpenIG

Un reverse proxy es un servidor proxy que recupera recursos de varios servidores en nombre de un cliente. Estos recursos son devueltos al cliente como si el que resolviese las peticiones fuese el propio reverse proxy. OpenIG, como reverse proxy, oculta todos los servidores establecidos detrás de él, dando la sensación de que es el propio server el que completa las peticiones. Debido a esta capacidad, es un sistema perfecto para ser instalado en una red perimetral y así actuar como primera línea de un sistema completo de seguridad.

Además de las características de seguridad que aporta a cualquier sistema, OpenIG permite interceptar las peticiones y respuestas al sistema transformándolas o ejecutando tareas transversales a todas las aplicaciones como por ejemplo la autenticación. Esta posición intermedia en todas las llamadas a un sistema, le permite proporcionar servicios a aplicaciones que por su tecnología o diseño no pueden integrarse con nuevas plataformas como OpenAM.

Algunas de las funcionalidades transversales que proporciona OpenIG son:

- Integración con gestión de accesos (OpenAM)
- Seguridad en aplicaciones y APIs
- Replay de credenciales
- Soporte para los protocolos OAuth, OIDC, SAML y SSO
- Control de tráfico de red
- Captura de peticiones y respuestas
- Reescritura de peticiones y respuestas

6.1 Arquitectura

Al igual que OpenAM, el componente principal de OpenIG es una aplicación web. El fichero que contiene la aplicación puede ser desplegado en varios tipos de servidores como por ejemplo Apache Tomcat.

La configuración que indica a OpenIG cuáles son las transformaciones o manipulaciones sobre peticiones y respuestas se guardan en ficheros de configuración en formato JSON. Según la configuración definida en estos ficheros, OpenIG instancia una serie de objetos en memoria que se encargará de los procesos sobre las peticiones.

La configuración se divide en múltiples carpetas dependiendo de su objetivo:

- .openig/config: ficheros de configuración administrativos.
- .openig/config/routes: ficheros de configuración de rutas y filtros.
- .openig/config/SAML: configuración de federación SAML.
- .openig/config/scripts/groovy: scripts de groovy para filtros y handlers.
- .openig/config/tmp: ficheros temporales.

Los Routers (enrutadores) de OpenIG son componentes que ejecutan las siguientes tareas:

- Definen los directorios de rutas y cargan su configuración en IG.
- Escanean los directorios de rutas y recargan la configuración si se produce algún cambio.
- Enrutan peticiones a la primera ruta cuya condición se satisfaga.

Las rutas son ficheros de configuración que se añaden a IG para gestionar peticiones. Estos ficheros se pueden añadir a mano, a través de un interfaz REST o a través de Studio. Todas las rutas deben delegar en un handler para procesar la petición y producir una respuesta. Las condiciones establecidas en una ruta indican las peticiones que puede gestionar. Esta configuración es

jerárquica lo que implica que se pueden definir configuraciones globales que afectan a cualquier ruta.

OpenIG cuenta con una interfaz de administración (Studio) que permite definir configuraciones. Esta interfaz provee una manera sencilla de probar OpenIG creando rutas, capturando mensajes, recolectando estadísticas, etc.

6.2 Procesado de peticiones

Las peticiones, respuestas y su contexto son manejados a través de handlers y filtros. Estas dos entidades se enlazan en cadenas.

Los handlers pueden delegar en otro handler o producir una respuesta. Una manera de producir una respuesta es la de redirigir una petición y devolver la respuesta del servicio llamado. También se pueden producir respuestas de manera estática convirtiendo a OpenIG en un servidor.

Los filtros pueden transformar los datos de la petición, la respuesta, el contexto o ejecutar una acción cuando una petición pasa por ellos (redirección a autenticación). También puede hacer de proxy sin modificar ningún dato como por ejemplo cuando escribe en fichero la información de las peticiones que pasan por él.

Un caso particular de filtro es la cadena de filtros. Este conjunto de elementos se convierte en una entidad que puede ser configurada como cualquier otro filtro sin tener que redefinir su estructura interna. Las cadenas procesan la petición a través de una serie de filtros y al finalmente a través de un handler. Una cadena de filtros es totalmente sustituible por un filtro y viceversa.

7. Plataformas seleccionadas

En base a los análisis vistos y a los estados del arte las plataformas seleccionadas para este TFM son:

- Plataforma SSO: OpenAM
- Balanceador de carga: HAProxy
- Proxy reverso: OpenIG

Para el rol de la plataforma SSO, principal componente de este TFM, se ha seleccionado OpenAM por diversos motivos:

- Aunque los métodos de autenticación de usuarios son muy similares en cuanto a protocolos soportados, OpenAM ofrece una mayor flexibilidad a la hora de establecer flujos de autenticación en base a distintas necesidades.
- Las capacidades de autorización de OpenAM se basan no solo en el estándar XACML sino que además de ABAC permite autorización basada en múltiples factores no basados en el propio usuario y circunstanciales.

- OpenAM cuenta con OpenDS como almacenamiento específico de tokens en lugar de tener que integrarse con soluciones de terceros como bases de datos, colas de mensajes, etc. Además, OpenDS cuenta con un sistema de replicación ad hoc que le otorga capacidades de alta disponibilidad. En el caso de CAS esta alta disponibilidad debe ser implementada en el ticket registry seleccionado y requiere de un expertise independiente al buscado en este TFM.
- El equipo de desarrollo cuenta con más de dos años de experiencia en la gestión y mantenimiento de un sistema basado en componentes OpenAM lo que reducirá la complejidad del proyecto y los tiempos de ejecución.

Los otros dos sistemas, complementarios al objetivo principal del TFM, son:

- HAProxy es una solución que cuenta con una buena reputación y con muy buena documentación. Es un sistema open source con una puesta en marcha sencilla y una configuración inicial sencilla para cumplir el objetivo de este TFM.
- OpenIG es del mismo fabricante que OpenAM por lo que se espera una integración sencilla y rápida. Además, el equipo de desarrollo usará este TFM como prueba de concepto de esta plataforma para el ámbito profesional.

Además de estas plataformas, se escoge el lenguaje de desarrollo Java y el Framework Spring para la implementación de la aplicación piloto. Esta elección está motivada por la familiaridad del equipo de desarrollo con el Framework.

8. Diseño

Tras la fase de análisis en las que se ha comentado la arquitectura y funcionamiento básico de las tres plataformas principales, se definen ahora, mediante diagramas UML, las interacciones entre estos tres elementos para conseguir los objetivos marcados por los requisitos. Además, se muestra en esta fase las necesidades de despliegue y todos los componentes de la arquitectura definida así como los razonamientos que originan a este diseño.

Como se ha visto en puntos anteriores de este documento, los actores principales de este sistema son:

- OpenIG como Gateway y punto de entrada.
- HAProxy como balanceador de carga del sistema de gestión de accesos.
- OpenAM como sistema de gestión de accesos (OpenDS como backend).
- Aplicaciones de prueba para proveer la funcionalidad.
- Usuarios finales.
- Administradores de la plataforma.

8.1 Diagramas de secuencia

Los diagramas de secuencia permiten modelar la interacción entre los diferentes actores de un sistema mostrando las llamadas que se producen entre ellos. En los siguientes diagramas se muestran los flujos de información necesarios para cumplir los requisitos RF-01 y RF-02 definidos en este documento.

RF-01 Mostrar datos de acceso de usuario (nivel básico)

En este diagrama de secuencia se muestran las interacciones entre los distintos actores que tienen lugar para mostrar la web que necesita nivel básico de acceso.

Cuando el usuario intenta acceder a la web de acceso básico, OpenIG intercepta la petición y comprueba si el usuario ya tiene una sesión asociada. Al no ser así, redirige la petición a OpenAM. HAProxy intercepta la petición y, según la configuración de balanceo, escoge el servidor de OpenAM al que corresponde atender la petición. OpenAM devuelve al navegador del usuario el formulario de autenticación con los controles para que pueda introducir usuario y contraseña. Una vez que el usuario informa de sus credenciales, estas son verificadas contra el backend de usuarios instalado. Si las credenciales son correctas, OpenAM genera una sesión y el token SSO correspondiente (cookie) que se establecerá para todas las aplicaciones de ese dominio. Una vez establecida la cookie se redirige al usuario a la página web de acceso básico.

OpenIG envía a OpenAM la información del usuario y de la sesión mediante cabeceras http a la web de acceso básico para que pueda parsear su contenido y mostrar esta información en la página de inicio.

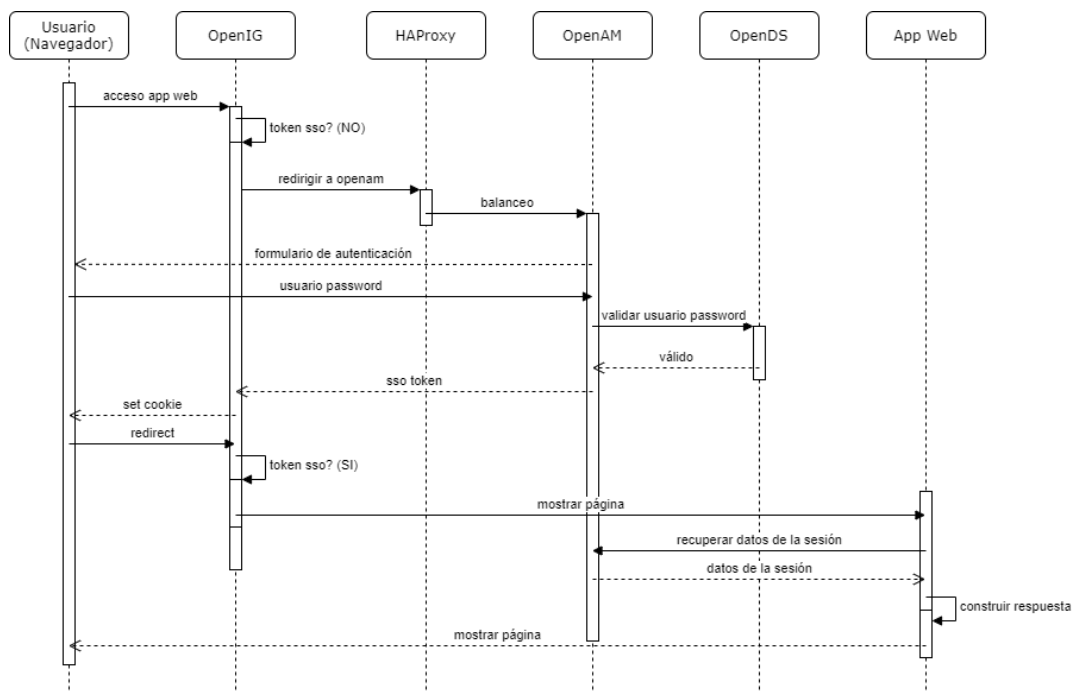


Figura 6 - Diagrama de secuencia RF-01

RF-02 Mostrar datos de acceso de usuario (nivel avanzado)

De manera análoga al punto anterior, el siguiente diagrama de secuencia muestra las interacciones entre los distintos actores en el caso de la autenticación basada en certificados. El punto diferencial entre los dos diagramas es la presentación al usuario de un selector de certificado al acceder a los Tomcat en los que se despliega cada instancia de OpenAM. OpenAM validará que el certificado no está revocado, que tiene una CA en la que confía y que el DN del usuario del certificado se encuentra en el almacén de identidades. La otra diferencia con RF-01 es que el nivel de autenticación proporcionado a la sesión en este caso será 2. En RF-01 el nivel de autenticación es 1.

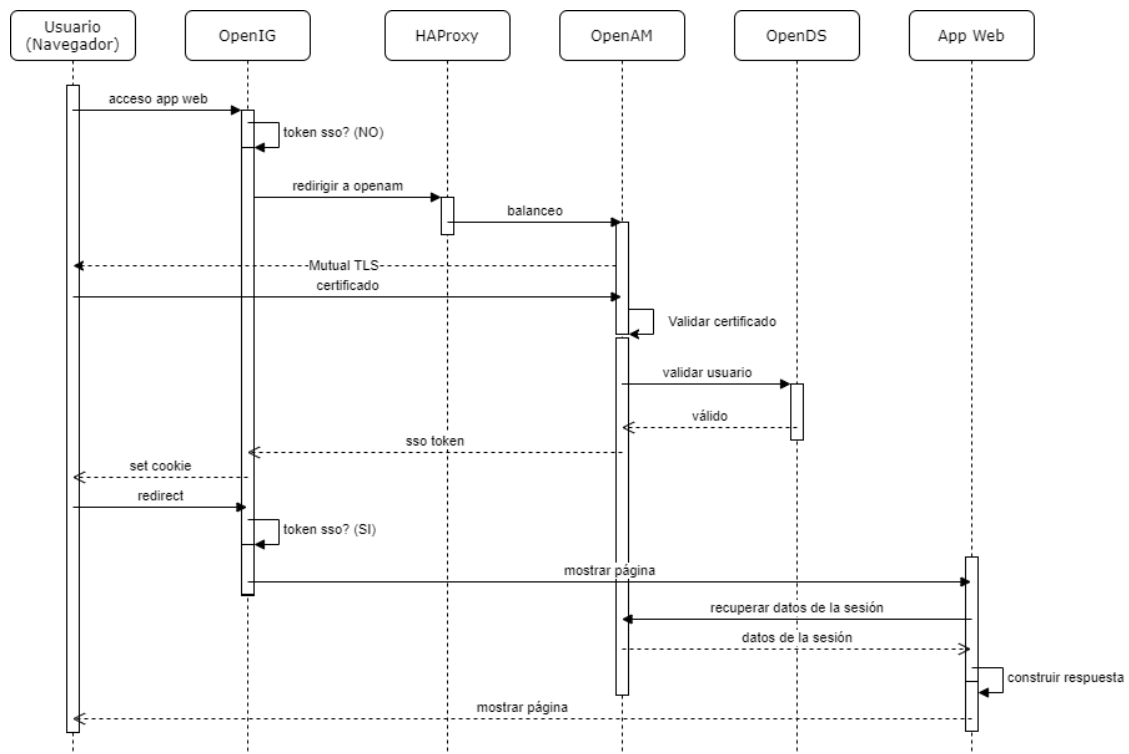


Figura 7 - Diagrama de secuencia RF-02

8.2 Diagrama de componentes

El siguiente diagrama muestra los componentes necesarios para el correcto funcionamiento del sistema.

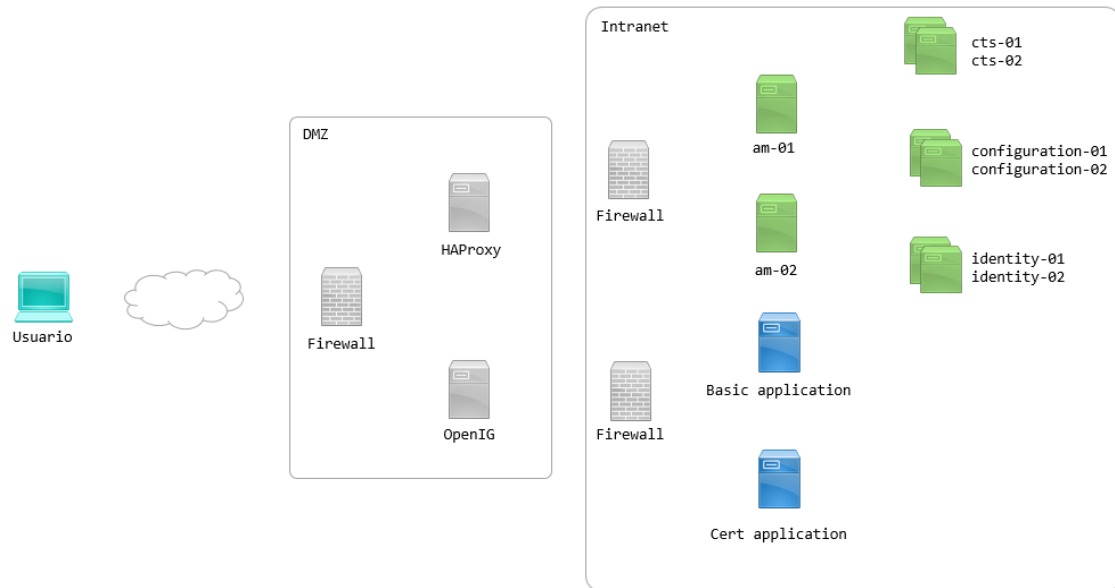


Figura 8 - Diagrama de componentes

En este diagrama destacan varios puntos de la arquitectura:

- Existen dos tipos de redes diferenciadas claramente. La DMZ o red perimetral está compuesta por los sistemas a los que los usuarios externos a la organización pueden acceder. Esta red, en una instalación empresarial productiva, debe estar protegida por un firewall que ayude en las medidas de seguridad. En la DMZ se desplegará OpenIG y HAProxy detrás del firewall. A través de una segunda interfaz de red en la máquina donde se despliega OpenIG se puede acceder a la red Interna.

En la intranet se desplegarán todas las aplicaciones corporativas. Esto incluye tanto las que dan servicio transversal como OpenAM (y sus sistemas adyacentes) como las aplicaciones funcionales. También puede optarse por desplegar OpenAM en una red interna distinta a la que se despliegan las aplicaciones corporativas para un mayor nivel de seguridad.

En este caso se toman las siguientes decisiones de diseño en cuanto a las redes definidas:

- OpenAM y sus aplicaciones complementarias se establecerán en una red interna.
- OpenIG se comunicará a través de comunicaciones TLS con OpenAM. Esto evitaría que un atacante que se haga con una de

las máquinas en esa red pueda esnifar tráfico sin encriptar entre OpenIG y OpenAM teniendo acceso a credenciales de usuario

- OpenIG se comunicará a través de conexiones no cifradas con las aplicaciones corporativas. En este tipo de conexiones no se transmitirán más que los token de sesión y no hay riesgo de que se puedan filtrar credenciales de usuario.
- Todos los componentes de OpenAM cuentan con dos instancias. Además, los componentes basados de OpenDS como la configuración, los CTS y los almacenes de identidades tienen replicación entre sí. Esta configuración pretende dotar al sistema de alta disponibilidad ya que todos los componentes cuentan con un sistema de backup.

8.3 Diagrama de despliegue

A continuación se muestra el diagrama de despliegue del sistema. Se observa que la implementación se llevará a cabo en una máquina virtual. La máquina virtual tendrá instalado Docker para simular distintas redes dentro de la intranet.

También se pueden comprobar dos tipos de componentes distintos:

- Las aplicaciones web que necesitan de un servidor como Tomcat para poder funcionar.
- Los procesos como OpenDS o HAProxy que no están basados en tecnologías web.

NOTA: En este trabajo no se implementan los firewalls que si son recomendables si este sistema se despliega en producción.

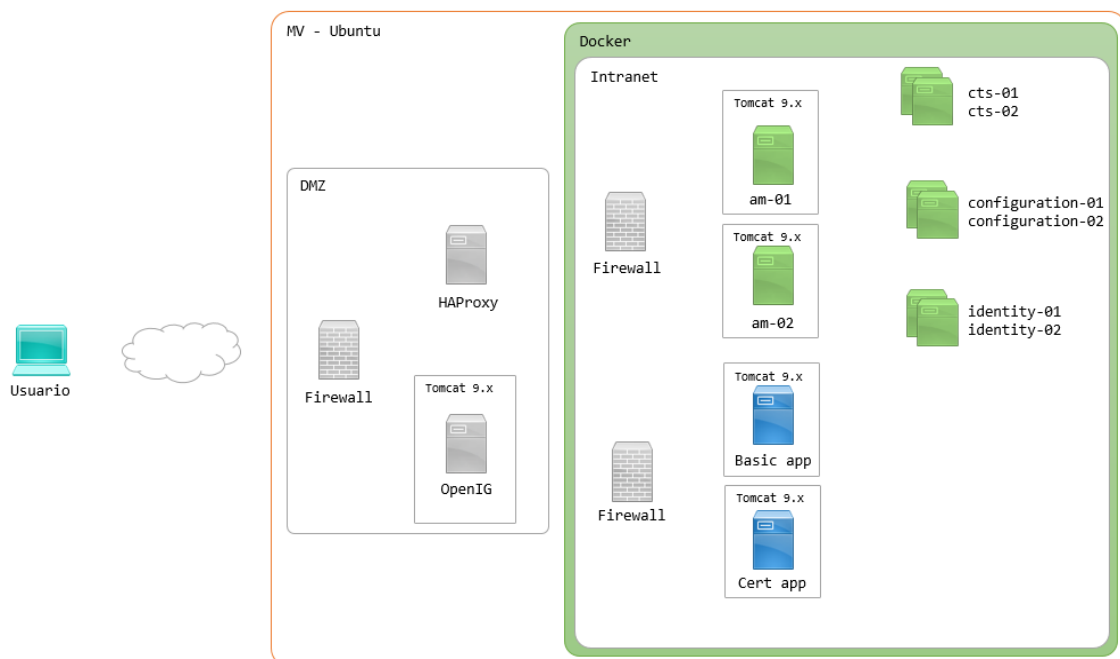


Figura 9 - Diagrama de despliegue

8.4 Diagrama de red

A continuación se muestra el diagrama de red con las interfaces correspondientes a cada aplicación tanto dentro de la máquina virtual Ubuntu como dentro de las redes Docker.

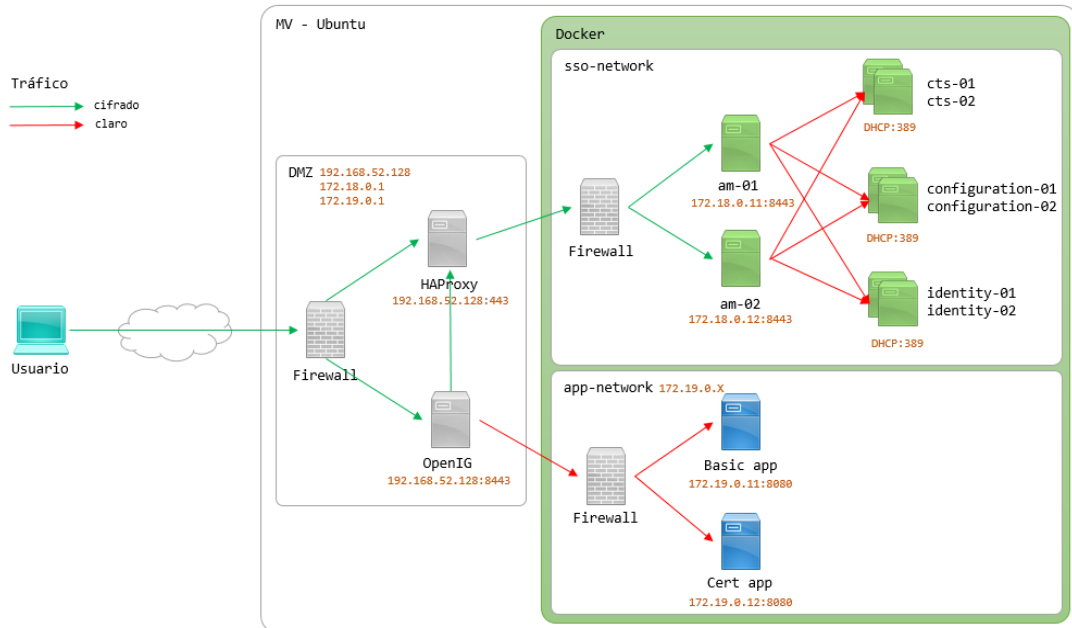


Figura 10 - Diagrama de red

Se pueden destacar varios puntos en este diagrama:

- La máquina virtual Ubuntu cuenta con varias interfaces de red. Una correspondiente a la red que comparte con el host Windows y otras dos correspondientes a las redes definidas en Docker.
- Docker cuenta con dos redes distintas para separar los componentes del sistema de autenticación de las aplicaciones funcionales. Estos dos tipos de componentes no deberían tener visibilidad entre ellos.
- Definir redes personalizadas en Docker permite que los contenedores se encuentren a través de un sistema de DNS basado en su nombre. De ahí que los contenedores que contienen las instancias de OpenDS no tengan IP predefinida y se le asigne una aleatoria cada vez que se crean.
- Como los frontales de OpenAM y las aplicaciones funcionales deben ser referenciadas desde la DMZ se les establece una IP predefinida.
- Se cifra el tráfico a la DMZ y a los frontales de OpenAM. El resto se deja sin cifrar.

9. Instalación de plataformas y tecnologías base

En la implementación de este trabajo se usan dos tecnologías base que permiten la instalación y ejecución de varias máquinas virtuales para formar todo el sistema. Estas tecnologías son VMWare Workstation Player y Docker.

VMWare Workstation Player permite la ejecución de máquinas virtuales dentro de una máquina host Windows. La máquina virtual elegida para este trabajo es un Ubuntu 19.10 eoan en su variante Budgie. Se descarga e instala la máquina virtual correspondiente a este sistema operativo de la página web osboxes.com.

Una vez arrancada la máquina virtual Ubuntu, se instala la herramienta Docker. Docker es una tecnología de contenedores que permite la creación y el uso de contenedores de Linux. Docker usa el kernel de Linux y las funciones de este, como Cgroups y namespaces, para segregar los procesos, de modo que puedan ejecutarse de manera independiente. El propósito de los contenedores es esta independencia: la capacidad de ejecutar varios procesos y aplicaciones por separado para hacer un mejor uso de su infraestructura y, al mismo tiempo, conservar la seguridad que tendría con sistemas separados.

Docker ofrece un modelo de implementación basado en imágenes. Esto permite compartir una aplicación, o un conjunto de servicios, con todas sus dependencias en varios entornos. Docker también automatiza la implementación de la aplicación (o conjuntos combinados de procesos que constituyen una aplicación) en este entorno de contenedores.

El uso de estas dos tecnologías permite tener, dentro de una sola máquina física, varias redes de máquinas que de otra manera tendrían que ser instaladas, configuradas y ejecutadas en máquinas físicas independientes. Como se ve más adelante en este documento, bajo el amparo de máquinas virtuales y contenedores se simulan tres redes, dos interna y otra perimetral, que dan forma a la implementación final de este trabajo.

La instalación de las plataformas necesarias para la ejecución del sistema se divide en dos partes.

Por un lado, las herramientas, instaladas como contenedores de Docker, que formarán parte de las redes internas del sistema. Estas herramientas son fundamentalmente los componentes de la plataforma SSO y las aplicaciones finales que contienen los procesos funcionales.

La creación de un contenedor de docker se basa en un fichero llamado Dockerfile. Este fichero contiene una lista de instrucciones que indican cómo se debe construir la imagen en la que se basa el contenedor. La imagen parte, habitualmente, de una base Linux (Alpine, Ubuntu, Debian) sobre la que se ejecutan diferentes comandos de configuración y/o instalación de herramientas.

Finalmente se define que comando se ejecuta en el arranque de un contenedor basado en esa imagen.

Cada contenedor en el momento de ser arrancado a partir de una imagen puede definir:

- Red a la que pertenece dentro de las definidas en Docker.
- Puertos que expone dentro de esa red y en el host (Ubuntu).
- Nombre del contenedor. Docker implementa un sistema de DNS basado en los nombres de contenedor que hace fácil descubrir las IPs de las máquinas existentes.
- Volúmenes. Debido a la naturaleza volátil de los contenedores y, para no perder los datos almacenados en ellos si se para y borra el contenedor, en el momento de ejecutar el comando *run* se configura el contenedor para que persista esos datos volátiles en volúmenes de Docker que se mantienen más allá de la vida de este. Estos volúmenes son mapeados a una parte del sistema de ficheros del contenedor.

Por otro lado, las herramientas instaladas en la propia máquina virtual, que formarán parte de la red perimetral del sistema. Estas herramientas son la puerta de entrada a todo el sistema y están en contacto directo con las máquinas cliente que consumen los servicios expuestos. HAproxy como balanceador de las máquinas OpenAM y OpenIG como reverse proxy.

9.1 Instalación de OpenDS

OpenDS forma parte del backend de OpenAM tal y como se define en puntos anteriores de este documento. Para este trabajo se instala, en varios contenedores de Docker, las tres variantes de esta plataforma: configuración, almacén de tokens y almacén de identidades.

Las tres instalaciones están basadas en el siguiente Dockerfile:

```
FROM krallin/ubuntu-tini:bionic
FROM adoptopenjdk/openjdk8:debian-slim

RUN apt-get update && apt-get install -y unzip && apt-get clean all
COPY opens.zip /var/tmp/opens.zip

WORKDIR /var/tmp

RUN unzip -q opens.zip && \
    mv opendj /opt/opendj

WORKDIR /opt/opendj
```

La imagen está basada en un Ubuntu-tiny y un debian-slim con jdk8. Como en todas las imágenes que se verán más adelante en este documento, primero se actualizan los repositorios de herramientas y a continuación se copia de la carpeta ds-base a */var/tmp* el archivo zip OpenDS. Se cambia de directorio y se

descomprime la herramienta en `/opt/opensj`. Finalmente se cambia de directorio de trabajo a `/opt/opensj`.

Para la creación de esta imagen se ejecuta el siguiente comando. A partir de ahora, todos los comandos de creación de imágenes y arranque de contenedores se ejecutarán, si no se indica lo contrario, desde esta carpeta.

```
juanma@osboxes:~$ docker build -t tfm/ds-base ds-base
```

El parámetro indicado por el atributo `-t` indica el nombre de la imagen generada por Docker. `ds-base` indica la carpeta donde se encuentra el fichero Dockerfile.

OpenDS de configuración

El OpenDS de configuración persiste todos los datos relacionados con las políticas de autenticación y autorización necesarias para cumplir con los requisitos del sistema. Almacena también, según como se configure, las referencias a los almacenes de identidades y tokens y otras configuraciones de OpenAM establecidas por los administradores de la plataforma.

El siguiente Dockerfile se basa la imagen `ds-base` y tiene los siguientes comandos:

```
FROM tfm/ds-base

EXPOSE 4444 389 636 8080 8443

RUN ./setup directory-server \
    --rootUserDn "cn=Directory Manager" \
    --rootUserPassword password \
    --monitorUserPassword password \
    --hostname opens.uoc.edu \
    --adminConnectorPort 4444 \
    --ldapPort 389 \
    --ldapsPort 636 \
    --httpPort 8080 \
    --httpsPort 8443 \
    --profile am-config \
    --set am-config/amConfigAdminPassword:password \
    --acceptLicense \
    --doNotStart

CMD ["/bin/start-ds", "--nodetach"]
```

Expone los puertos necesarios para acceder a la herramienta: administración, LDAP, LDAPS, HTTP y HTTPS. Ejecuta el comando de configuración de OpenDS como almacén de configuraciones. Este comando define el usuario base, el DN donde se almacenan todas las configuraciones, los puertos que expone, el nombre de host y el tipo de configuración de OpenDS: `am-config`. Establece el comando de ejecución de OpenDS. Se indica el parámetro `--`

nodetach para que la ejecución del contenedor no finalice al ejecutar el comando ya que por defecto devuelve el control a la línea de comandos.

Para ejecutar un contenedor de Docker basado en esta imagen se deben ejecutar dos comandos:

- Creación de la imagen a partir del Dockerfile:

```
juanma@osboxes:~$ docker build -t tfm/ds-configuration configuration
```

- Ejecución de un contenedor basado en esa imagen:

```
juanma@osboxes:~$ docker run -d --name configuration-ds-1 -v configuration-ds-1:/opt/opensj/db --network sso-network tfm/ds-configuration
```

Este comando tiene las siguientes particularidades que serán comunes en el resto de aplicaciones creadas en Docker:

- **-d**: indica que la ejecución del contenedor debe ser “detached”, es decir, se indica que se vuelva a la línea de comandos después de finalizar la ejecución.
- **--name**: indica el nombre del contenedor creado.
- **-v**: indica el volumen y el mapeo al sistema de archivos interno del contenedor.
- **--network**: indica la red interna de docker a la que este contenedor pertenece.
- **tfm/ds-configuration**: indica la imagen en la que se basa este contenedor.

Una vez ejecutado un contenedor por primera vez este podrá ser parado e iniciado tantas veces como sea necesario sin perder la configuración con la que se ha definido.

OpenDS de identidades

La instalación, ejecución y configuración de OpenDS como almacén de identidades es muy similar a la vista para configuración.

```
FROM tfm/ds-base
```

```
EXPOSE 4444 389 636 8080 8443
```

```
RUN ./setup directory-server \  
    --rootUserDn "cn=Directory Manager" \  
    --rootUserPassword password \  
    --monitorUserPassword password \  
    --hostname opens.uoc.edu \  
    --adminConnectorPort 4444 \  
    --ldapPort 389 \  
    --ldapsPort 636 \  
    --httpPort 8080 \  
    --httpsPort 8443 \  
    --profile am-identity-store \  
    --set am-identity-store/amIdentityStoreAdminPassword:password \  
    --acceptLicense \  
    --doNotStart
```

```
CMD ["/bin/start-ds", "--nodetach"]
```

La diferencia con el caso anterior radica en el valor del parámetro profile que en este caso es *am-identity-store*. De manera análoga, la construcción de la imagen para este tipo de OpenDS y su comando de ejecución de un contenedor son los siguientes:

```
juanma@osboxes:~$ docker build -t tfm/ds-identity identity
```

```
juanma@osboxes:~$ docker run -d --name identity-ds-1 -v identity-ds-1:/opt/opensds/db --network sso-network tfm/ds-identity
```

OpenDS de tokens

La instalación, ejecución y configuración de OpenDS como almacén de tokens es muy similar a las vistas anteriormente.

```
FROM tfm/ds-base
```

```
EXPOSE 4444 389 636 8080 8443
```

```
RUN ./setup directory-server \  
    --rootUserDn "cn=Directory Manager" \  
    --rootUserPassword password \  
    --monitorUserPassword password \  
    --hostname opens.uoc.edu \  
    --adminConnectorPort 4444 \  
    --ldapPort 389 \  
    --ldapsPort 636 \  
    --httpPort 8080 \  
    --httpsPort 8443 \  
    --profile am-cts \  
    --set am-cts/amCtsAdminPassword:password \  
    --set am-cts/tokenExpirationPolicy:am-sessions-only \  
    --acceptLicense \  
    --doNotStart
```

```
CMD ["/bin/start-ds", "--nodetach"]
```

La diferencia con el caso anterior radica en el valor del parámetro `profile` que en este caso es `am-cts`. De manera análoga, la construcción de la imagen para este tipo de OpenDS y su comando de ejecución de un contenedor son los siguientes:

```
juanma@osboxes:~$ docker build -t tfm/ds-cts cts
```

```
juanma@osboxes:~$ docker run -d --name cts-ds-1 -v cts-ds-1:/opt/opensj/db --network sso-network tfm/ds-cts
```

Para cumplir con el requisito no funcional de alta disponibilidad se crearán dos contenedores de cada tipo. Los contenedores de OpenDS disponibles en la red interna serán:

- configuration-ds-1
- configuration-ds-2
- identity-ds-1
- identity-ds-2
- cts-ds-1
- cts-ds-2

9.2 Instalación de OpenAM

OpenAM es la aplicación web que funciona como frontal de la Plataforma SSO proporcionando una interfaz de usuario que hace la configuración del sistema más fácil para los administradores. De manera similar a la configuración de OpenDS, la configuración de OpenAM se basa en una imagen y dos contenedores de Docker.

El fichero Dockerfile de OpenAM cuenta con más detalle que los anteriores:

```
FROM tomcat:jdk8-openjdk-slim

RUN apt-get update && apt-get install -y unzip && apt-get clean all
COPY openam.war /var/tmp/openam.war

RUN rm -fr "$CATALINA_HOME"/webapps/* && \
    mkdir "$CATALINA_HOME"/webapps/openam && \
    (cd "$CATALINA_HOME"/webapps/openam; jar -xvf /var/tmp/openam.war) && \
    rm -f /var/tmp/openam.war

ENV FORGEROCK_HOME /home/forgerock

RUN apt-get update && apt-get install -y unzip curl bash wget \
    && addgroup --gid 11111 forgerock \
    && adduser --shell /bin/bash --home "$FORGEROCK_HOME" --uid 11111 --ingroup root --disabled-
password --gecos "" -q forgerock \
    && chown -R forgerock:root "$CATALINA_HOME" \
    && chown -R forgerock:root "$FORGEROCK_HOME" \
    && chmod -R g+rxw "$CATALINA_HOME" \
    && apt-get clean all

COPY server.xml "$CATALINA_HOME"/conf/server.xml
COPY openam-server-keystore.jks "$CATALINA_HOME"/conf/openam-server-keystore.jks

COPY certificate-node-1.0.0.jar "$CATALINA_HOME"/webapps/openam/WEB-INF/lib

EXPOSE 8080 8443

USER forgerock
```

Los puntos más destacables de este Dockerfile son los siguientes:

- La imagen base es de un Tomcat con una JVM versión 8 que a su vez se basa en una imagen de openjdk basada en Linux Alpine.
- Se actualizan los repositorios de herramientas como en el caso de OpenDS.
- Se copia el war de OpenAM y se extrae en la carpeta webapps de Tomcat.
- Se establece la carpeta home del usuario ForgeRock para el que posteriormente se crean y se configuran sus permisos de forma que pueda ejecutar tomcat.
- Se copia el server.xml. Esto es muy importante porque en él se define el conector HTTPS y la keystore de java. Este tema se ve más en detalle en el punto Gestión de certificados.
- Se copia la keystore de java.
- Debido a que los árboles de autenticación no tienen los nodos correspondientes a certificados en la instalación base de OpenAM, es necesario acudir al Marketplace de ForgeRock y descargar el jar que contiene esta funcionalidad. Hay que incluir este jar en la carpeta `/webapps/openam/WEB-INF/lib` del Tomcat donde se despliegan los OpenAM.
- Se exponen los puertos 8080 y 8443.
- Se cambia el usuario a forgerock.

Este Dockerfile tiene la particularidad de que no define un comando de arranque ya que es suficiente con el definido en la imagen en la que se basa. Al hacer esto se delega en el comando de arranque que ejecuta el fichero `$CATALINA_HOME/bin/catalina.sh`.

Para ejecutar un contenedor de Docker basado en esta imagen se deben ejecutar dos comandos:

- Creación de la imagen a partir del Dockerfile:

```
juanma@osboxes:~$ docker build -t tfm/am am
```

- Ejecución de un contenedor basado en esa imagen:

```
juanma@osboxes:~$ docker run -d --name am-01 -p 9081:8080 -p 9443:8443 --network sso-network --ip 172.18.0.11 -v am-01:/home/forgerock tfm/am
```

Para cumplir con el requisito no funcional de alta disponibilidad se crearán dos contenedores de cada tipo. Los contenedores de OpenAM disponibles en la red interna serán:

- am-01
- am-02

9.3 Instalación de aplicaciones funcionales

Las aplicaciones desarrolladas para probar la funcionalidad del sistema se basan en el framework de desarrollo java Spring Framework. En sus últimas versiones este framework está muy orientado al despliegue de aplicaciones en

contenedores. Este hecho encaja perfectamente con lo que se buscaba para este trabajo, simpleza de despliegue y desarrollo.

El resultado de construir los desarrollos de esta aplicación es un fichero jar (fat jar) que ya incorpora las librerías necesarias para ejecutar un Tomcat de manera embebida. Por este motivo el fichero Dockerfile necesario para definir las imágenes de las aplicaciones es muy sencillo:

```
FROM openjdk:8-jdk-alpine

COPY basic-demo-app.jar /var/tmp/app.jar

WORKDIR /var/tmp|

EXPOSE 8080

ENTRYPOINT ["java", "-jar", "app.jar"]
```

La imagen se basa en una imagen de java, basada a su vez en un Linux Alpine. En este Dockerfile, la copia del fichero ejecutable y la exposición del puerto 8080 son suficientes para poder ejecutar la aplicación con el comando java indicado en ENTRYPOINT.

- Creación de las imágenes a partir del Dockerfile:

```
juanma@osboxes:~$ docker build -t tfm/basic-app basic-app
```

```
juanma@osboxes:~$ docker build -t tfm/certificate-app certificate-app
```

- Ejecución de un contenedor basado en esa imagen:

```
juanma@osboxes:~$ docker run -d --name basic-app --network app-network tfm/basic-app
```

```
juanma@osboxes:~$ docker run -d --name certificate-app --network app-network tfm/certificate-app
```

9.4 Instalación de HAProxy

HAProxy será el balanceador de carga que hará de frontal de las dos instancias de OpenAM para poder cumplir con el requisito de alta disponibilidad. Como se comenta anteriormente en este documento HAProxy consta principalmente de un fichero de configuración haproxy.cfg y un proceso ejecutable.

Para la instalación de HAProxy en la máquina virtual deben ejecutarse los siguientes comandos:

```
juanma@osboxes:~$ sudo apt-get -y install haproxy
```

Para validar la instalación ejecutamos el siguiente comando:

```
juanma@osboxes:~$ haproxy -v
HA-Proxy version 2.0.5-1ubuntu0.4 2020/04/03 - https://haproxy.org/
```

La instalación de HAProxy mediante este método instala un servicio en la máquina virtual Ubuntu y lo habilita para arrancar en el inicio de sesión en la máquina.

9.5 Instalación de OpenIG

OpenIG al igual que OpenAM es una aplicación web en formato war que se despliega en un contenedor de servlets como Tomcat. Por este motivo, debe instalarse un Tomcat en la máquina virtual Ubuntu.

Esta instalación se compone de los siguientes pasos:

- Instalación de la jvm de Java.
- Creación del grupo y usuario de tomcat.
- Descarga y descompresión del fichero de instalación Tomcat.
- Actualizar los permisos del usuario.
- Creación del fichero que dará lugar al servicio de Tomcat.
- Recarga del daemon de servicios, habilitación del servicio de Tomcat para arrancar al iniciar la máquina virtual y comprobación de estado del servicio

```
sudo apt install default-jdk
```

```
sudo groupadd tomcat
```

```
sudo useradd -s /bin/false -g tomcat -d /opt/tomcat tomcat
```

```
curl -O https://apache.brunneis.com/tomcat/tomcat-8/v8.5.54/bin/apache-tomcat-8.5.54.tar.gz
```

```
sudo mkdir /opt/tomcat
```

```
sudo tar xzvf apache-tomcat-*tar.gz -C /opt/tomcat --strip-components=1
```

```
cd /opt/tomcat
```

```
sudo chgrp -R tomcat /opt/tomcat
```

```
sudo chmod -R g+r conf
```

```
sudo chmod g+x conf
```

```
sudo chown -R tomcat *
```

```
sudo nano /etc/systemd/system/tomcat.service
```

```
sudo systemctl daemon-reload
```

```
sudo systemctl start tomcat
```

```
sudo systemctl status tomcat
```

```

[Unit]
Description=Apache Tomcat Web Application Container
After=network.target

[Service]
Type=forking

Environment=JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64
Environment=CATALINA_PID=/opt/tomcat/temp/tomcat.pid
Environment=CATALINA_HOME=/opt/tomcat
Environment=CATALINA_BASE=/opt/tomcat
Environment='CATALINA_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC'
Environment='JAVA_OPTS=-Djava.awt.headless=true -Djava.security.egd=file:/dev/./urandom'

ExecStart=/opt/tomcat/bin/startup.sh
ExecStop=/opt/tomcat/bin/shutdown.sh

User=tomcat
Group=tomcat
UMask=0007
RestartSec=10
Restart=always

[Install]
WantedBy=multi-user.target

```

Finalmente se descarga la aplicación web de OpenIG de la web del fabricante y se le modifica el nombre a ROOT.war. Se copia este fichero en la carpeta webapps de Tomcat y se reinicia el servicio. OpenIG debe ser configurada en Tomcat como root para que pueda cumplir con sus funciones.

Al arrancar Tomcat por primera vez y acceder a OpenIG se crea la carpeta *.openig* en */opt/tomcat* y es necesario actualizar los permisos sobre esa carpeta para el usuario tomcat previamente creado.

10. Configuración de la arquitectura

Este punto define las configuraciones necesarias en los contenedores de Docker y en la máquina virtual para que todas las plataformas del sistema sean capaces de interactuar entre sí y pueda pasarse a la fase de configuración puramente funcional.

10.1 OpenDS Replicación

La replicación entre los distintos nodos de OpenDS es importante de cara a garantizar la estabilidad del sistema en condiciones fuera de lo normal. Para esta configuración se debe acceder a los contenedores de docker correspondientes a los distintos pares de OpenDS: configuración, identidades y tokens.

Para acceder a un contenedor en ejecución se puede utilizar, por ejemplo, el siguiente comando de Docker:

```

juanma@osboxes:~$ docker exec -it identity-ds-2 /bin/bash

```

El comando exec de docker tiene los siguientes atributos:

- **-it**: accede a la consola de manera interactiva.
- **name**: nombre del contenedor al que se quiere acceder.
- **comando**: comando que se quiere ejecutar inicialmente

La ejecución de este comando abre una consola bash en el contenedor elegido. Una vez aquí dentro (seleccionamos los contenedores con el número 2 de cada par) ejecutaremos los siguientes comandos:

```

juanma@osboxes:~$ ./bin/dsreplication \
> configure \
> --adminUID admin \
> --adminPassword password \
> --baseDN ou=identities \
> --host1 identity-ds-1 \
> --port1 4444 \
> --bindDN1 "cn=Directory Manager" \
> --bindPassword1 password \
> --replicationPort1 8989 \
> --host2 identity-ds-2 \
> --port2 4444 \
> --bindDN2 "cn=Directory Manager" \
> --bindPassword2 password \
> --replicationPort2 8989 \
> --trustAll \
> --no-prompt

```

```

juanma@osboxes:~$ ./bin/dsreplication \
> initialize \
> --adminUID admin \
> --adminPassword password \
> --baseDN ou=identities \
> --hostSource identity-ds-1 \
> --portSource 4444 \
> --hostDestination identity-ds-2 \
> --portDestination 4444 \
> --trustAll \
> --no-prompt

```

El primer comando configura el sistema de réplica con el que cuenta OpenDS. En él se indican los servidores origen y destino de la réplica así como los usuarios de administración de cada servidor, los puertos de réplica y el DN base que se monitorizará de cara a replicar sus entradas.

Una vez configurada la replicación se puede hacer la comprobación de su estado con el siguiente comando:

```

juanma@osboxes:~$ ./bin/dsreplication status --adminUID admin --adminPassword password --hostname localhost --port 4444 --trustAll

```

Este comando da como resultado la tabla de la imagen:

```

root@436698762d9c:/opt/opensdj# ./bin/dsreplication status --adminUID admin --adminPassword password --hostname localhost --port 4444 --trustAll
The equivalent non-interactive command-line is:
dsreplication status \
  --hostname localhost \
  --port 4444 \
  --adminUID admin \
  --adminPassword ***** \
  --trustAll \
  --no-prompt
Suffix DN      : Server      : Entries : Replication enabled : DS ID : RS ID : RS Port (1) : Delay (ms) : Security (2)
-----
ou=identities : identity-ds-1:4444 : 11      : true                : 4085  : 20312 : 8989        : 0           : false
ou=identities : identity-ds-2:4444 : 11      : true                : 18417 : 29297 : 8989        : 0           : false
uid=Monitor   : identity-ds-1:4444 : 1        :                      :        :        :              :             :
uid=Monitor   : identity-ds-2:4444 : 1        :                      :        :        :              :             :
[1] The port used to communicate between the servers whose contents are being replicated.
[2] Whether the replication communication through the replication port is encrypted or not.

```

En él se puede observar que ambas instalaciones cuentan con un total de 11 entradas bajo el dn seleccionado para la réplica (*ou=identities*). Este hecho hace que, aunque uno de los dos DS de identidades no se encuentre operativo, el otro contenga las mismas entradas y pueda continuar con el servicio. Más adelante se ve como configurar OpenAM para que tenga los dos OpenDS en cuenta y acuda a uno o al otro en función del que se encuentre operativo.

Además del número de entradas puede verse el retardo en la copia de entradas entre los dos OpenDS. En este caso Delay 0ms indica que en cuanto una entrada se crea o modifica en uno de los dos servicios es instantáneamente creada o modificada en el otro.

De manera análoga se ejecutan estos comandos para los otros dos pares de OpenDS: configuración y tokens.

10.2 Configuración de OpenAM

Existe una ligera diferencia entre la configuración de un nodo de OpenAM cuando este es el primero que se configura o cuando ya forma parte de un Site (cluster de servidores OpenAM). Al arrancar por primera vez el contenedor y acceder a la url base de OpenAM se presenta la siguiente pantalla:



Figura 11 - Configuración inicial OpenAM

OpenAM ofrece la posibilidad de hacer una configuración predeterminada que hará uso de una instancia de OpenDS local a la máquina para que funcione

como almacén de configuración y tokens. También ofrece la posibilidad de establecer una configuración avanzada que permite seleccionar instancias de OpenDS independientes. En el Anexo I se pueden ver los pasos del asistente de configuración personalizada así como dos configuraciones extra que son necesarias una vez finalizada la instalación.

10.3 Configuración de HAProxy

Como ya se ha comentado en puntos anteriores de este documento, la configuración de HAProxy se hace a partir de un fichero de configuración HAproxy.cfg. La parte más destacable del fichero de configuración, que se encuentra en /etc/haproxy/haproxy.cfg, se puede ver en este extracto:

```
frontend http-in
    bind *:80
    bind *:443
    option tcplog
    mode tcp
    default_backend openam-servers

backend openam-servers
    mode tcp
    balance roundrobin
    server node1 am-01:8443 check fall 3 rise 2
    server node2 am-02:8443 check fall 3 rise 2

listen stats
    bind :32700
    stats enable
    stats uri /
    stats hide-version
    stats auth admin:password
```

Este fichero configura HAProxy en modo SSL passthrough lo que implica que no hará nada más que redirigir todas las peticiones https al correspondiente backend a nivel de capa de transporte.

Para este propósito se define un frontend que escuchará cualquier petición en el puerto 443 y la redirigirá las peticiones a los backends configurados en *openam-servers*.

El backend define un balanceo de tipo roundrobin (alterna entre los dos nodos) para los servidores de openam configurados en los pasos previos. Estos dos nodos escuchan peticiones de tipo https en el puerto 8443. Se ha configurado una verificación del estado de los nodos por la que, si falla tres veces la verificación de que la interfaz de red no está disponible, se dejan de enviar peticiones a ese nodo. Por el contrario, si esa verificación vuelve a ser positiva dos veces, se envían peticiones de nuevo a ese nodo.

La referencia a los nodos desde la máquina virtual Ubuntu no se resuelve a través del DNS de Docker por lo que es necesario configurar la máquina virtual para que sepa dónde están esos servidores. Esta configuración puede verse en el siguiente punto.

Además del frontend y backend se configura el puerto 32700 para poder acceder a una consola de monitorización del balanceador. En ella se puede ver el estado de los nodos de backend, número de peticiones atendidas, etc.

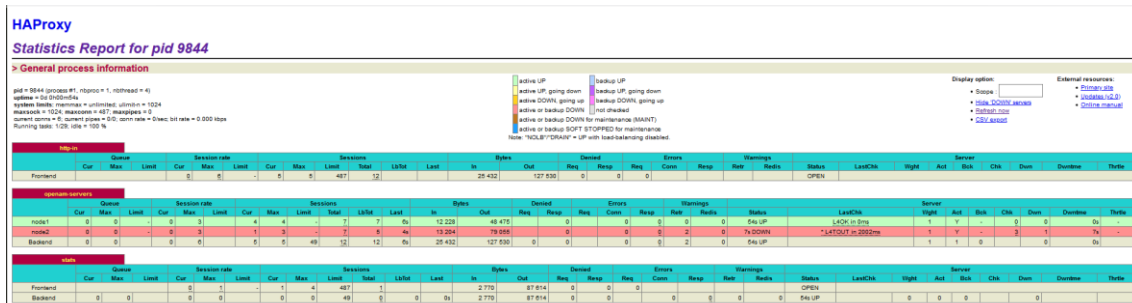


Figura 12 - Monitorización estado HAProxy

10.4 Configuraciones adicionales

La máquina virtual necesita ser configurada para acabar de establecer la conectividad entre todos los elementos de la arquitectura.

Es necesario establecer las IPs de los contenedores de Docker a los que se harán peticiones desde la máquina, es decir, los dos OpenAM y las aplicaciones de prueba. Además es necesario acceder a la consola de administración de OpenAM a través de un FQDN con lo que se indica aquí también esa configuración. Se modifica el archivo local `/etc/hosts` para hacer el mapeo entre las IPs y los FQDN correspondientes.

```

juanma@osboxes:~$ cat /etc/hosts
127.0.0.1      localhost openig.example.com openam.example.com
127.0.1.1     osboxes

172.19.0.11   basicapp.example.com
172.19.0.12   certificateapp.example.com
172.18.0.11   am-01
172.18.0.12   am-02
    
```

Una configuración análoga debe hacerse desde la máquina Windows que actuará como cliente de estos servicios desde una red externa. Al no contar con un DNS que resuelva estos nombres de host, es necesario configurar el fichero `C:\Windows\System32\Drivers\etc\hosts`:

```

127.0.0.1      localhost
192.168.52.128 openig.example.com openam.example.com
    
```

192.168.52.128 corresponde a la IP de la máquina virtual Ubuntu en la red local que comparte con el host Windows. Para probar esta configuración en otro host, debe cambiarse esta IP por la que corresponda a la máquina virtual de Ubuntu.

11. Configuración funcional

Este punto define las configuraciones necesarias en las plataformas ya instaladas para cumplir con los requisitos funcionales del sistema.

11.1 OpenAM

OpenAM ofrece la posibilidad de separar las configuraciones para distintos sistemas en dominios. En un dominio se pueden establecer flujos de autenticación, políticas de autorización, protocolos como OAuth2 y SAML y muchos otros componentes de manera independiente. Para este sistema se ha creado un dominio llamado Basic. Este apartado enumera las principales configuraciones que se han llevado a cabo en OpenAM en el dominio Basic.

Autenticación

Como se comenta en apartados anteriores de este documento, a partir de la versión 6.0.0 de la plataforma, el fabricante recomienda hacer una migración de las cadenas y módulos a los árboles y nodos de autenticación. Para este trabajo se han probado las dos opciones y se ha comprobado que las dos tienen la misma funcionalidad. La diferencia entre las dos reside en la facilidad de configuración y en la flexibilidad que ofrecen a la hora de añadir o quitar elementos en el flujo.

Finalmente se ha optado por configurar un árbol de autenticación llamado "Intranet" y se ha configurado como el flujo principal de autenticación para este dominio (Dominio Basic > Authentication > Settings > Configuración de autenticación de organización > Intranet).

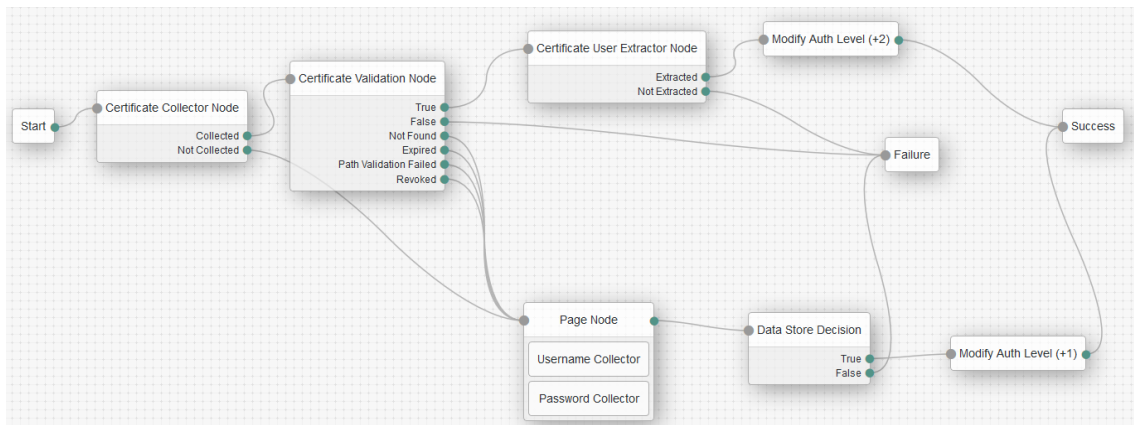


Figura 13 - Árbol de autenticación intranet

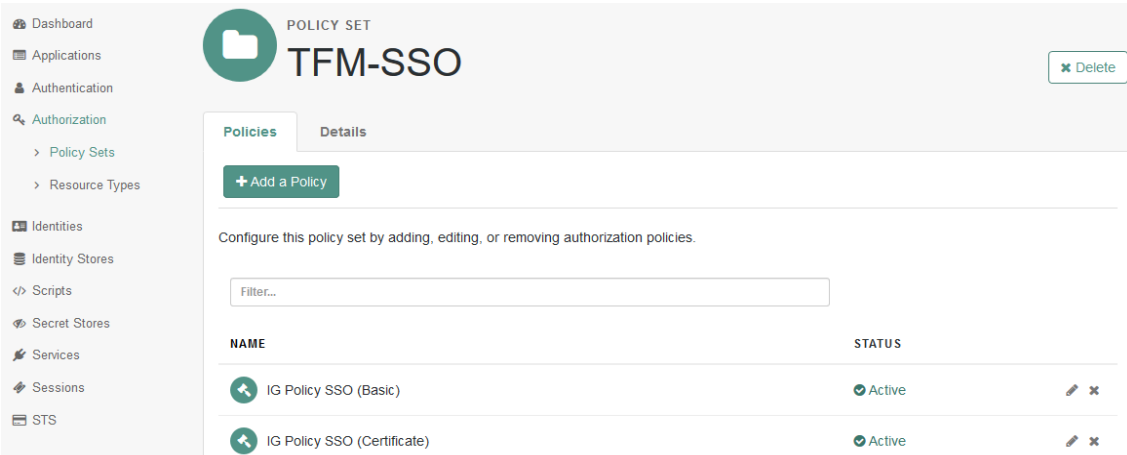
En la imagen anterior, se ve el árbol de autenticación que se utiliza para los usuarios del sistema creado para este trabajo. Por un lado están los nodos que se encargan de recoger, validar y extraer el usuario del certificado presentado. Si esta parte del árbol se cumple sin problemas, se le asignará al usuario un nivel de autenticación 2, el mayor posible en este sistema. La validación del certificado se hace a partir del CN configurado en el mismo. El valor de este CN será verificado contra el almacén de identidades configurado en este dominio,

es decir, el valor del CN (juanmaro en el caso del certificado de pruebas) debe existir como usuario.

Si alguno de estos tres pasos falla entonces comienza a ejecutarse la otra rama del árbol en la cual se recogen credenciales básicas a partir de un formulario y se comprueban contra el almacén de identidades configurado. En este caso se asigna al usuario un nivel de autenticación 1, el más bajo en este sistema para usuarios autenticados.

Autorización

OpenAM basa su sistema de autenticación en políticas XACML. Para cubrir los objetivos funcionales del sistema se ha definido un conjunto de políticas (TFM-SSO) que contiene dos políticas distintas (Dominio Basic > Authorization > Policy sets):



The screenshot displays the OpenAM interface for configuring a Policy Set named 'TFM-SSO'. The left sidebar contains navigation links for Dashboard, Applications, Authentication, Authorization (with sub-links for Policy Sets and Resource Types), Identities, Identity Stores, Scripts, Secret Stores, Services, Sessions, and STS. The main content area shows the 'Policies' tab for the 'TFM-SSO' policy set. It includes an 'Add a Policy' button, a filter input field, and a table of existing policies.





| NAME | STATUS | |
|-----------------------------|--------|---|
| IG Policy SSO (Basic) | Active |   |
| IG Policy SSO (Certificate) | Active |   |

Figura 14 - Políticas de autorización

Dentro del set de políticas de autorización se definen dos políticas distintas que protegerán a las aplicaciones de prueba: IG Policy SSO (Basic) e IG Policy SSO (Certificate). Estas políticas serán utilizadas por el reverse proxy (OpenIG) para saber que tiene que hacer con las peticiones a las aplicaciones de intranet.

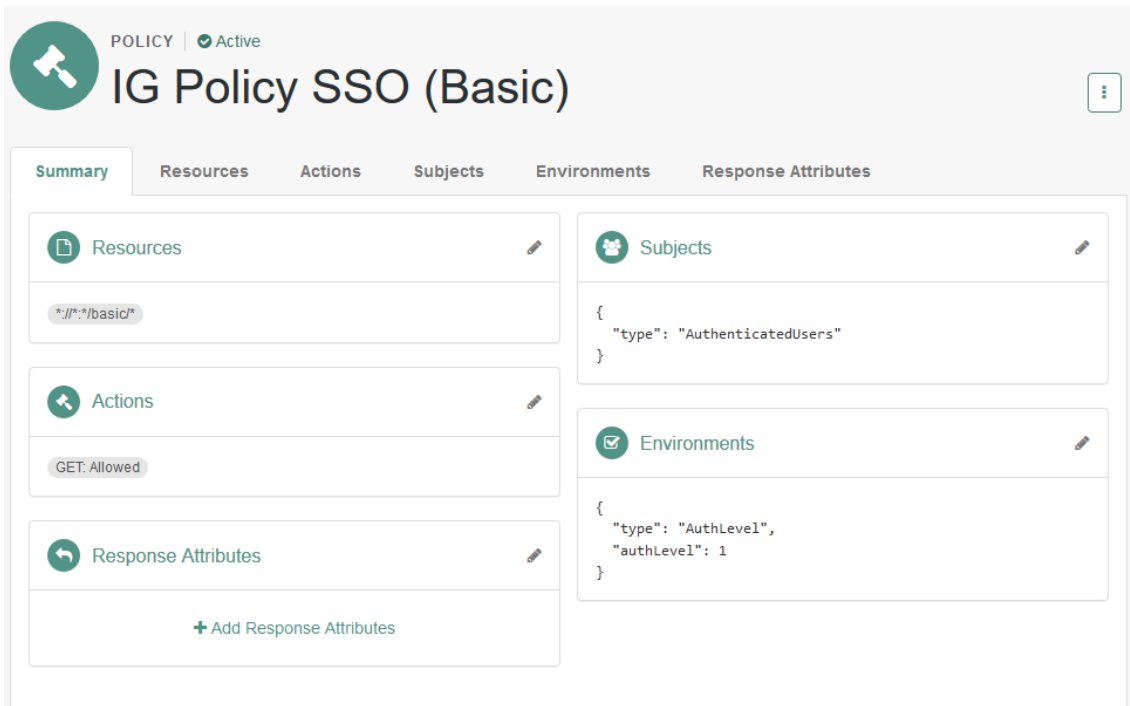


Figura 15 - Política de autorización básica

IG Policy SSO (Basic) define que está permitido ejecutar un GET a cualquier url que comience por /basic siempre y cuando el usuario que lo ejecute se haya autenticado en OpenAM y tenga un nivel de autenticación de por lo menos 1.

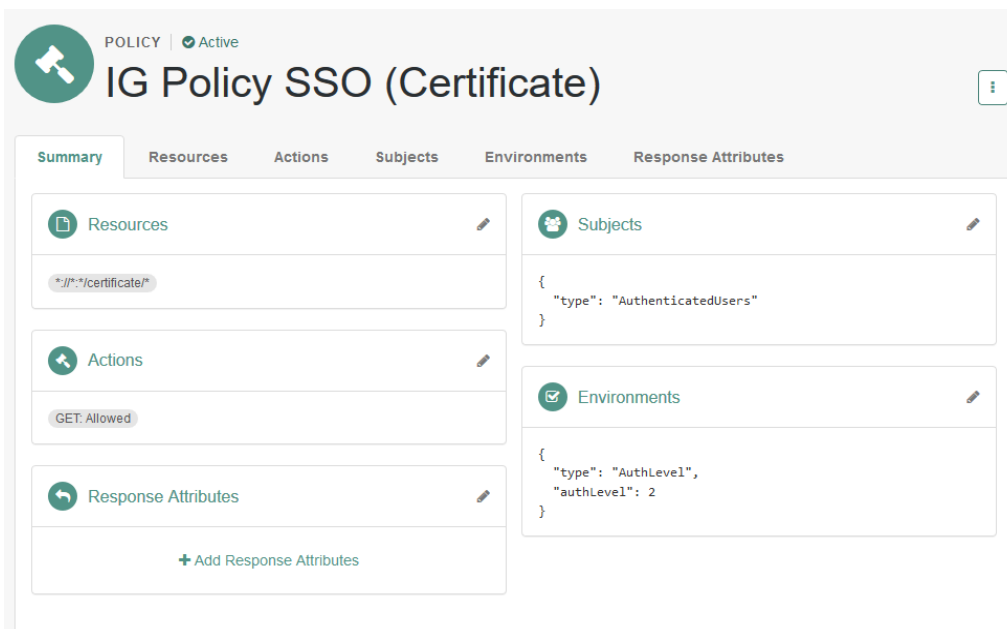


Figura 16 - Política de autorización certificado

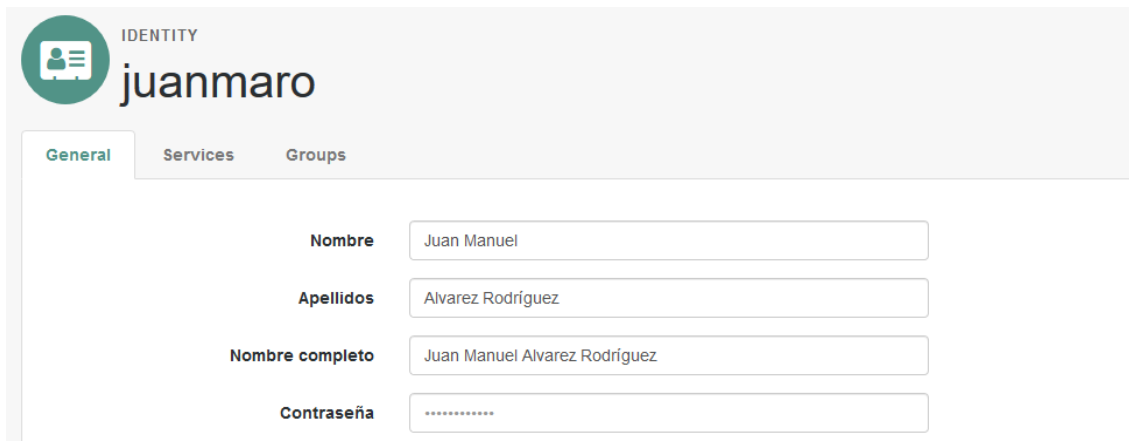
IG Policy SSO (Certificate) define que está permitido ejecutar un GET a cualquier url que comience por /certificate siempre y cuando el usuario que lo ejecute se haya autenticado en OpenAM y tenga un nivel de autenticación de por lo menos 2.

Otras configuraciones

Hay una serie de configuraciones menores que se deben establecer para dar cumplimiento a todos los requisitos.

Es necesario crear un agente java para que OpenIG pueda conectarse a la plataforma OpenAM y consultar las políticas de autorización comentadas anteriormente. Para esto se ha creado el agente *ig_agent* en Dominio Basic > Applications > Agents > Java.

Debe establecerse un almacén de identidades (con el configurado en la instalación es suficiente) y deben crearse identidades para poder hacer las pruebas de los requisitos funcionales definidos.



The screenshot displays the 'IDENTITY' management interface for a user named 'juanmaro'. It features three tabs: 'General', 'Services', and 'Groups'. The 'General' tab is active, showing a form with the following fields:

| Field | Value |
|-----------------|-------------------------------|
| Nombre | Juan Manuel |
| Apellidos | Alvarez Rodriguez |
| Nombre completo | Juan Manuel Alvarez Rodriguez |
| Contraseña | ***** |

Figura 17 - Ejemplo de usuario en el almacén de identidades

11.2 OpenIG

La primera configuración obligatoria antes de hacer ninguna otra en OpenIG es la de establecer el entorno como entorno de desarrollo. Para esto debemos crear el fichero `admin.json` en la carpeta base de configuración (`/opt/tomcat/openig/config`) con el siguiente contenido:

```
{  
  "mode": "DEVELOPMENT"  
}
```

A partir de esta configuración existen dos maneras de definir las rutas en OpenIG. A través de Studio, una interfaz web que facilita su creación o creandolas a mano a través de ficheros json. A la hora de poner este sistema en producción será necesario reestablecer el modo a modo producción eliminando el fichero `admin.json` para que las rutas queden fijadas.

Para el sistema implementado en este trabajo se han definido dos rutas que se corresponden con las dos aplicaciones de prueba.

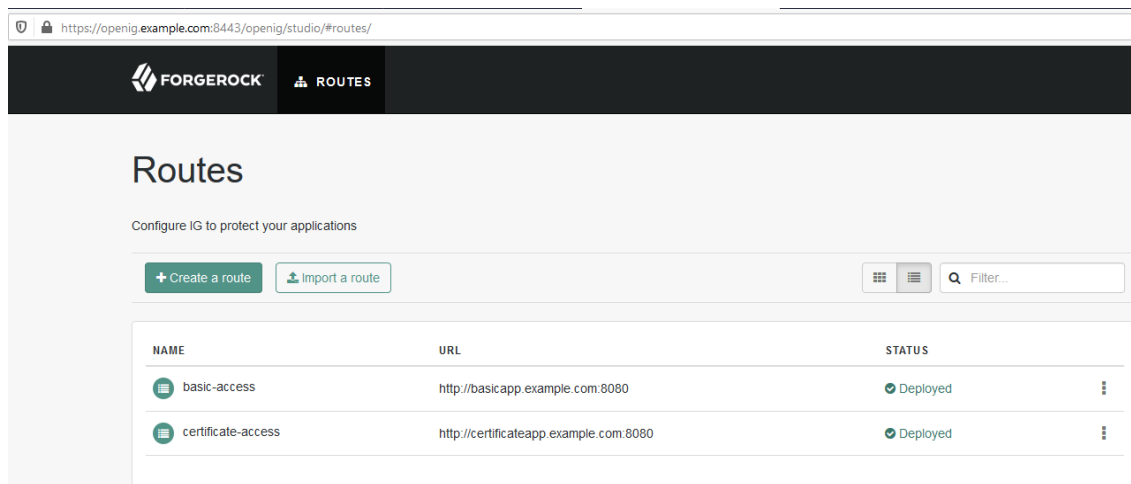


Figura 18 - Rutas de OpenIG

Ruta de aplicación básica

La ruta básica protege la aplicación <http://basicapp.example.com:8080/basic/home>. Para eso utiliza la siguiente cadena de filtros y handlers cuya configuración se puede ver más en detalle en el Anexo II: Configuración de filtros de OpenIG.

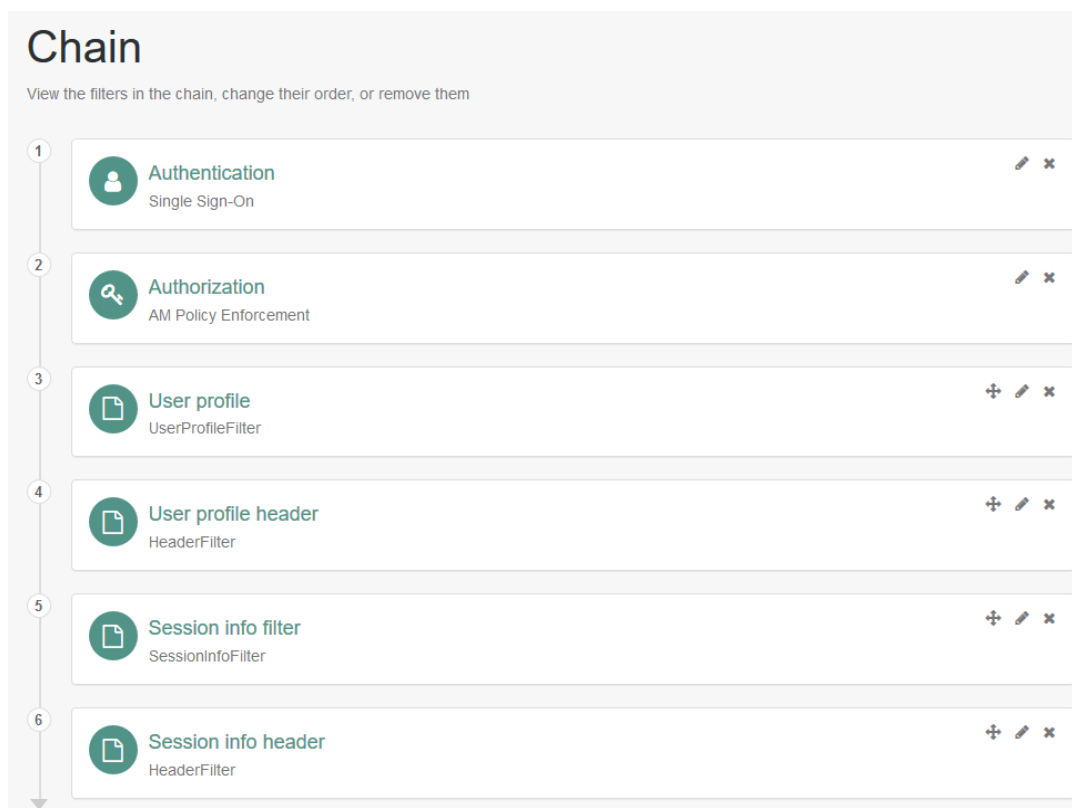


Figura 19 - Cadena de filtros ruta básica

En la consola de administración de OpenIG se crea un servicio de Autenticación utilizando la instalación de OpenAM. En este servicio se ha usado el agente creado en la configuración del dominio *basic ig_agent*.

Se configura, así mismo, un servicio de autorización que partiendo de la URL de la ruta que se está protegiendo, enviará el token de autenticación del usuario para que valide si cumple con la política que aplica, en este caso IG Policy SSO (Basic).

Además se han incluido otros filtros que se integran a pares. El primer par extrae información del perfil del usuario y lo incluye en una cabecera. El segundo par hace lo mismo pero con la información de la sesión. Usando estos cuatro filtros, las aplicaciones de prueba son capaces de mostrar la información relacionada con el usuario y la sesión.

Ruta de aplicación de certificados

Esta ruta protege la aplicación <http://basicapp.example.com:8080/certificate/home>. Para eso utiliza una cadena de autenticación análoga a la vista en el apartado anterior y cuya configuración se puede ver más en detalle en el Anexo II: Configuración de filtros de OpenIG.

Auditoría

Para los requisitos de auditoría se establece un gestor de eventos que escribirá en la carpeta /audit de la instalación en formato .json.

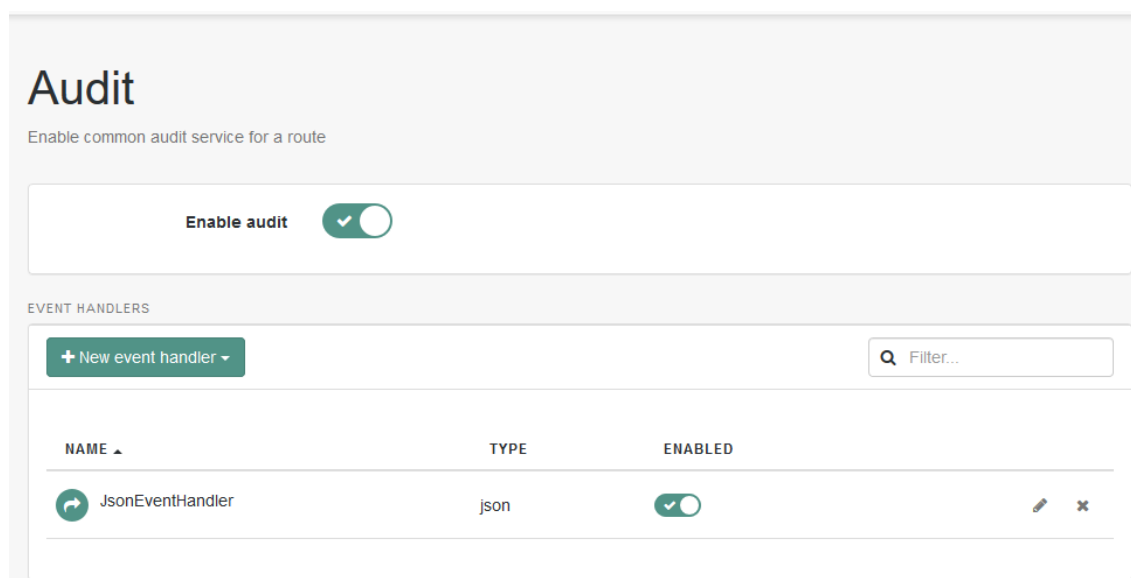


Figura 20 - Gestor de eventos de auditoría OpenIG

Este gestor de eventos de auditoría está disponible tanto para la ruta básica como para la de certificados.

12. Gestión de certificados

Parte de la seguridad de un sistema como el implementado en este trabajo es la capacidad que tiene de comunicarse de forma segura entre sus componentes. Para comunicarse de manera segura, teniendo en cuenta que la mayoría de comunicaciones son a través del protocolo HTTP, se han creado los certificados necesarios para que los servidores y las personas que los usan puedan comunicarse a través de HTTPS.

En este diagrama de red, visto en la fase de diseño, se aprecia que la comunicación cifrada se hace desde el exterior de la máquina virtual hacia el Tomcat de OpenIG, a través de HAProxy configurado como SSL passthrough y con los Tomcat de los nodos de OpenAM. Por esto, se necesitarán 2 certificados de servidor (1 OpenIG y 1 OpenAM), un certificado personal para el usuario de pruebas y un certificado de CA para poder generar y establecer como confiables los tres certificados anteriores.

Creación

El primer paso en el proceso de creación de certificados es el de crear la autoridad certificadora raíz de todos los certificados que se crearán. Para este proceso primero creamos una clave de firma rsa de 4096 bits y con ella generamos el certificado raíz de la CA.

```
juanma@osboxes:~/TFM$ openssl genrsa -des3 -out rootCA.key 4096
Generating RSA private key, 4096 bit long modulus (2 primes)
.....+++++
.....+++++
e is 65537 (0x010001)
```

Figura 21 - Creación clave de firma CA

```
juanma@osboxes:~/TFM$ openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 1024 -out rootCA.crt
Enter pass phrase for rootCA.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:A Coruña
Locality Name (eg, city) []:Santiago de Compostela
Organization Name (eg, company) [Internet Widgits Pty Ltd]:UOC
Organizational Unit Name (eg, section) []:MISTIC
Common Name (e.g. server FQDN or YOUR name) []:Plataforma SSO Root CA
Email Address []:juanmaalvarez@uoc.edu
```

Figura 22 - Creación certificado CA

Una vez generados estos dos elementos, se crean los otros tres certificados. Este proceso es el mismo para los tres certificados así que solamente se indica la creación del certificado del servidor OpenIG. Como en el caso de la CA se crea una clave de firma RSA, en este caso más pequeña, de 2048 bits.

```
juanma@osboxes:~/TFM/certificates$ openssl genrsa -out openig.example.com.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
.....+++++
e is 65537 (0x010001)
```

Figura 23 – Creación clave de firma servidor OpenIG

Con la clave de firma se crea una Certificate Signing Request (CSR) que, en caso de un ejemplo de la vida real, debe enviarse a una CA reconocida como Verisign o DigiCert para que generen el certificado.

```

juanma@osboxes:~/TFM/certificates$ openssl req -new -key openig.example.com.key -out openig.example.com.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:A Coruña
Locality Name (eg, city) []:Santiago de Compostela
Organization Name (eg, company) [Internet Widgits Pty Ltd]:OUC
Organizational Unit Name (eg, section) []:MISTIC
Common Name (e.g. server FQDN or YOUR name) []:openig.example.com

```

Figura 24 - Creación CSR servidor OpenIG

Como poseemos el certificado y la clave de firma de la CA podemos generar nosotros mismos el certificado a partir de la CSR.

```

juanma@osboxes:~/TFM/certificates$ openssl X509 -req -in openig.example.com.csr -CA rootCA.crt -CAkey rootCA.key -CAcreateserial -out openig.example.com.crt -days 500 -sha256
Signature ok
subject=C = ES, ST = A Coru(C)83(C)2)B1a, L = Santiago de Compostela, O = OUC, OU = MISTIC, CN = openig.example.com
Getting CA Private Key
Enter pass phrase for rootCA.key:

```

Figura 25 - Creación certificado OpenIG

Instalación en máquinas y navegadores

Antes de poder instalar los certificados en las máquinas virtuales y en los navegadores de los usuarios que harán uso de la plataforma se deben pasar los certificados a formato pfx y jks para poder importarlos en Firefox y Tomcat respectivamente.

```

juanma@osboxes:~/TFM/certificates$ openssl pkcs12 -export -out personal.pfx -inkey personal.key -in personal.crt
Enter Export Password:
Verifying - Enter Export Password:

```

Figura 26 - Conversión certificado crt a pfx

```

juanma@osboxes:~/TFM/certificates$ keytool -importkeystore -destkeystore openig-server-keystore.jks -srckeystore openig.example.com.p12 -srcstoretype pkcs12 -alias openig.example.com
Importando el almacén de claves de openig.example.com.p12 a openig-server-keystore.jks...
Introduzca la contraseña de almacén de claves de destino:
Volver a escribir la contraseña nueva:
Introduzca la contraseña de almacén de claves de origen:
Warning:
El almacén de claves JKS utiliza un formato propietario. Se recomienda migrar a PKCS12, que es un formato estándar del sector que utiliza "keytool -importkeystore -srckeystore openig-server-kesto
re.jks -destkeystore openig-server-keystore.jks -deststoretype pkcs12".
juanma@osboxes:~/TFM/certificates$ keytool -import -trustcacerts -alias uoc-root -keystore openig-server-keystore.jks -file rootCA.crt
Introduzca la contraseña del almacén de claves:
Propietario: EMAILADDRESS=juannaalvarez@uoc.edu, CN=Plataforma SSO Root CA, OU=MISTIC, O=UOC, L=Santiago de Compostela, ST=A Coruña, C=ES
Alias: EMAILADDRESS=juannaalvarez@uoc.edu, CN=Plataforma SSO Root CA, OU=MISTIC, O=UOC, L=Santiago de Compostela, ST=A Coruña, C=ES
Número de serie: 1536689e9cb179b488543b9c2c15a81be53f3158
Válido desde: Sat Apr 25 17:28:11 CET 2020 hasta: Mon Feb 13 16:28:11 CET 2023
Huellas digitales del certificado:
MD5: 08:5C:80:4E:32:74:AA:FF:4C:8C:ED:29:A8:6C:74:BB
SHA1: 07:CB:30:1D:FE:E7:C5:C0:BS:AE:90:32:7B:80:B6:D7:3B:A7:72:63
SHA256: F7:9D:69:EB:51:1A:51:2F:3E:BE:DF:1A:3D:71:00:BD:00:AB:4E:93:3A:63:B3:29:00:05:21:20:D9:13:4D:54
Nombre del algoritmo de firma: SHA256withRSA
Algoritmo de clave pública de asunto: Clave RSA de 4096 bits
Versión: 3
Extensiones:
#1: ObjectID: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: B5 36 E7 8B FE FA 5A 02 87 CA 0B D6 8D 16 D1 02 ..6....Z.....
0010: B9 FA 43 46 ..CF
]
]
#2: ObjectID: 2.5.29.19 Criticality=true
BasicConstraints:
CA:true
PathLen:2147483647
]
#3: ObjectID: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: B5 36 E7 8B FE FA 5A 02 87 CA 0B D6 8D 16 D1 02 ..6....Z.....
0010: B9 FA 43 46 ..CF
]
]
¿Confíar en este certificado? [no]: si
Se ha agregado el certificado al almacén de claves
Warning:
El almacén de claves JKS utiliza un formato propietario. Se recomienda migrar a PKCS12, que es un formato estándar del sector que utiliza "keytool -importkeystore -srckeystore openig-server-kesto
re.jks -destkeystore openig-server-keystore.jks -deststoretype pkcs12".

```

Figura 27 - Conversión e importación a jks

Este último paso es muy importante porque los certificados que el usuario puede seleccionar a la hora de autenticarse de esta manera dependen de los certificados incluidos en la keystore que haga de trustcacerts.

Una vez generados todos los ficheros se debe importar el fichero personal.pfx y rootCA.crt en el Firefox o navegador que se vaya a utilizar para hacer las pruebas. Así mismo, como ya se comenta en puntos anteriores, debe configurarse el server.xml de los Tomcat para establecer el conector que dará servicio HTTPS.

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol" SSLEnabled="true"
  maxThreads="150" scheme="https" secure="true"
  keystoreFile="conf/openam-server-keystore.jks" keystorePass="changeit"
  truststoreFile="conf/openam-server-keystore.jks" truststorePass="changeit"
  clientAuth="want" sslProtocol="TLS" />
```

Este conector tiene varios puntos a destacar: la configuración de los parámetros keystore y truststore y sus correspondientes contraseñas, la configuración de SSL habilitado y el más importante en el caso de los servidores de OpenAM, clientAuth="want" que solicita al usuario que presente un certificado válido para autenticarse (autenticación mutua TLS).

13. Desarrollo aplicaciones de prueba

Las aplicaciones de prueba se desarrollan utilizando el Spring Framework con la sencillez como objetivo. Su única funcionalidad será la de proporcionar una respuesta cuando se haga un GET sobre una url determinada. La respuesta devuelve una página web en la que se muestran las cabeceras de la petición.

Se muestra un extracto del código que gestiona la petición GET y del código html que se devuelve. En ambas aplicaciones el código es similar, solamente se modifican un par de configuraciones de la URL a la que atienden y el título del HTML que se devuelve.

```
@Controller
@RequestMapping("/basic")
public class WebController {

    private final static String USER_INFO_HEADER = "x-ig-userprofileinfo";

    @GetMapping("/home")
    public String getHomePage(HttpServletRequest request, @RequestHeader Map<String, String> headers, Model model)
        throws JsonMappingException, JsonProcessingException {

        String userInfo = Optional.ofNullable(headers.get(USER_INFO_HEADER)).orElse("");

        User user = parseUser(userInfo);

        model.addAttribute("user", user);
        model.addAttribute("cookies", request.getCookies());
        model.addAttribute("headers", headers);

        return "home";
    }
}
```

Figura 28 - Controlador de Spring Framework

```
<body>
<section class="header">
  <div class="center-header">
    <p>Welcome, <span th:text="${user.cn}" /></p>
  </div>
</section>
```

Figura 29 - Html creada con Thymeleaf

USER INFORMATION

Emp. # 150220
Contact juanmaro@plataformasso.com - 682982982

REQUEST INFORMATION

```

Headers accept = text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
accept-encoding = gzip, deflate, br
accept-language = es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
cookie = amlbcookie=04; iPlanetDirectoryPro=OUF0FHTeu-tekvLqgRk9W0ulo2s.*AAJTSQACMDIAA1NLABxIYk9xK0
JGU1hyNy9VW1JJdk4zc3ZYaTUSNGM9AAR0eXB1AANDVFMAA1MxAAIwNA..*
dnt = 1
host = openig.example.com:8443
referer = https://openam.example.com/
upgrade-insecure-requests = 1
user-agent = Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:75.0) Gecko/20100101 Firefox/75.0
x-ig-sessioninfo = username: juanmaro, realm: /basic, properties: {AMCtxId=ea05f79f-8342-4034-bb9d-
7ed6880eb5ef-2982, AuthLevel=1}
x-ig-userprofileinfo = username: juanmaro, rawInfo: {_id=juanmaro, _rev=878448038, employeeNumber=
[150220], mail=[juanmaro@plataformasso.com], givenName=[Juan Manuel], sn=[Alvarez Rodriguez], telep
honeNumber=[682982982], cn=[Juan Manuel Alvarez Rodriguez], username=juanmaro}
connection = Keep-Alive

```

Figura 30 - Captura aplicación basic

14. Pruebas

En este apartado se ejecutarán las pruebas necesarias para verificar los requisitos funcionales y no funcionales definidos en este documento. Se ejecutarán los casos de prueba definidos desde una máquina externa, representada en este caso por un navegador en la máquina host Windows.

| | |
|------------------|---|
| CP-01 | Acceso de nivel básico correcto |
| Pasos | <p>Se accede a https://openig.example.com:8443/basic/home</p> <p>Se redirige a la consola de OpenAM donde se descarta seleccionar certificado</p> <p>Se muestra el formulario de usuario y contraseña</p> <p>Se introducen usuario y contraseña</p> <p>Se redirige a la web “basic” donde se muestran los datos de acceso.</p> |
| Resultado | OK |
| CP-02 | Acceso de nivel avanzado correcto |
| Pasos | <p>Se accede a https://openig.example.com:8443/certificate/home</p> <p>Se redirige a la consola de OpenAM donde se selecciona el certificado personal</p> <p>Se redirige a la web “cetificate” donde se muestran los datos de acceso.</p> |
| Resultado | OK |
| CP-03 | Acceso mediante credenciales de tipo certificado |
| Pasos | <p>Se accede a https://openig.example.com:8443/certificate/home</p> <p>Se redirige a la consola de OpenAM donde se selecciona el certificado personal</p> <p>Se redirige a la web “cetificate” donde se muestran los datos de acceso.</p> |

| | |
|--------------------|---|
| Resultado | OK |
| CP-04 | Acceso mediante credenciales de usuario |
| Pasos | Se accede a https://openig.example.com:8443/basic/home Se redirige a la consola de OpenAM donde se descarta seleccionar certificado Se muestra el formulario de usuario y contraseña Se introducen usuario y contraseña Se redirige a la web "basic" donde se muestran los datos de acceso. |
| Resultado | OK |
| CP-05 | Acceso incorrecto |
| Pasos | Se accede a https://openig.example.com:8443/basic/home Se redirige a la consola de OpenAM donde se descarta seleccionar certificado Se muestra el formulario de usuario y contraseña Se introducen usuario y/o contraseña incorrectos. Se produce un error de login |
| Resultado | OK |
| CP-06 | Escalado de privilegios |
| Pasos | Se accede a https://openig.example.com:8443/certificate/home Se redirige a la consola de OpenAM donde se descarta seleccionar certificado Se muestra el formulario de usuario y contraseña Se introducen usuario y/o contraseña Se redirige al usuario a la autenticación mediante certificado |
| Resultado | KO |
| Comentarios | Aunque el resultado de esta prueba es técnicamente correcto. Tomcat no vuelve a requerir al usuario el certificado con lo que nunca podrá conseguir autenticarse con más nivel si no cierra y vuelve a abrir el navegador. |
| CP-07 | Acceso a directo a intranet |
| Pasos | Intento de acceso a la máquina am-01:8443 o 178.18.0.11 o openam.example.com:9443 Se muestra la interfaz de uno de los servidores de am. |
| Resultado | KO |
| Comentarios | Con esos nombres de máquina no se puede llegar pero si se accede a Ubuntu al puerto 9443 si se muestra la interfaz de openam del servidor am-01. Tal y como está diseñada esta implementación, los nodos de openam en docker asocian sus puertos a la máquina virtual de Ubuntu, por eso son accesibles desde el Host Windows. |
| CP-08 | Resiliencia |
| Pasos | Parar distintos contenedores de docker. Ejecutar los pasos de un caso de uso básico. |

| | |
|------------------|--|
| Resultado | OK |
| CP-09 | Auditoría |
| Pasos | <p>Ejecutar un caso de uso básico</p> <p>Se comprueba que en la carpeta /opt/tomcat/.openig/audit existen logs de auditoria mostrando el proceso de protección de la petición al backend.</p> <p>Se comprueba que en el contenedor de am en la carpeta /home/forgerock/openam/openam/logs existen logs de auditoría mostrando el proceso de autenticación.</p> |
| Resultado | OK |

15. Conclusiones

De este proyecto saco las siguientes conclusiones:

- OpenAM es un producto muy completo que proporciona una gran flexibilidad y facilidad de configuración a la hora de definir métodos de autenticación y políticas de autorización. El backend OpenDS es fácil de configurar y ofrece un buen rendimiento y cantidad de herramientas “out of the box” que ayudan en su instalación, configuración y monitorización.
- Por el contrario OpenIG parece un producto mucho más inmaduro para los usuarios administradores. Ciertamente, tiene una integración casi inmediata con OpenAM por ser de la misma casa. Le falta trabajo hasta llegar al punto de madurez en el que se encuentra OpenAM pero sus funcionalidades son suficientes para un proyecto de esta envergadura. Buscaría hacer una prueba de concepto con otras opciones como Kong o Apigee para implantarlo en un entorno productivo.
- HAproxy cumple su cometido con solvencia y una configuración muy resumida y específica. Tiene una curva de aprendizaje alta si hay necesidades más allá de las básicas.
- En cuanto a tecnologías auxiliares, este ha sido el primer trabajo con Docker en el que he puesto varios conceptos en práctica a la vez. Es muy intuitivo, cómodo y rápido poder establecer nuevos contenedores a partir de imágenes preestablecidas. Lo usaré más en proyectos futuros.
- Aunque funcionalmente los requisitos están conseguidos en su totalidad, los requisitos no funcionales no se han cumplido de todo. En proyectos en los que involucran partes de arquitectura de sistemas es bueno contar con varias máquinas físicas y otros elementos intermedios como DNS y firewalls para que el despliegue se asemeje lo más posible al real.
- La planificación no se ha podido seguir de manera adecuada sobre todo en la fase de análisis y diseño por motivos ajenos al proyecto. Aquí entra en juego el riesgo identificado en la fase inicial (Riesgo 6 –

Indisponibilidad del equipo de trabajo) para el que se definió una estrategia de contención. Gracias a esto, en la fase de implementación se pudo recuperar el tiempo perdido y se llegó a tiempo al hito de la tercera entrega.

16. Posibles extensiones

Existen varios puntos a tener en cuenta para extender este proyecto antes de una posible puesta en producción.

- Configuración de un sistema de migración/importación masivo de usuarios. En una empresa con empleados ya configurados y con un sistema implantado desde hace tiempo, sería necesario establecer un procedimiento por el cual esos empleados fuesen migrados a OpenDS. Otra opción sería la integración de OpenAM con un directorio activo ya existente.
- Configuración de una ruta pública en OpenIG para aplicaciones al alcance de cualquier usuario. Estas aplicaciones tendrían que estar, idealmente, en una red distinta a las aplicaciones de backoffice.
- Servidor de replicación específico. En este proyecto se ha configurado la segunda instancia de cada OpenDS para que ejecute una réplica contra su contraparte. En sistemas con más carga sería preferible establecer un OpenDS que se encargue únicamente de las réplicas. Esto hace que no se pierda rendimiento de los que se encargan de almacenar configuración, identidades o tokens.
- Autodescubrimiento de servidores en la configuración de HAProxy. Para evitar tener que configurar los ficheros de host debería configurarse HAProxy para descubrir automáticamente nuevos contenedores de OpenAM.
- Establecimiento de canales seguros de comunicación entre todos los elementos del sistema.
- Establecimiento de firewalls y hardening de todas las máquinas del sistema. Estas dos extensiones están más enfocadas a la seguridad en general que a un sistema de SSO pero aquí también tienen cabida. Deberían establecerse por lo menos dos firewall: uno entre internet y la red perimetral y otro entre la red perimetral y la red de docker.

17. Glosario

Autenticación: Procedimiento informático que permite asegurar que un usuario de un sitio web u otro servicio similar es auténtico o quien dice ser.

Autorización: Proceso por el que se permite o deniega la ejecución de una acción en función de quien intenta ejecutarla y/o las circunstancias en las que se intenta ejecutar.

Reverse proxy: tipo de servidor proxy que recupera recursos en nombre de un cliente desde uno o más servidores. Estos recursos son devueltos al cliente como si se originaran en el propio servidor Web.

TLS: Seguridad de la capa de transporte y su antecesor Secure Sockets Layer son protocolos criptográficos, que proporcionan comunicaciones seguras a través de una red.

XACML (eXtensible Access Control Markup Language): es un estándar que define un lenguaje declarativo de políticas de control de acceso implementado en XML y un modelo de procesamiento que describe cómo evaluar peticiones de acceso según las reglas definidas en las políticas

ABAC: es un paradigma de control de acceso mediante el cual se otorgan derechos de acceso a los usuarios mediante el uso de políticas que combinan atributos de su identidad.

Aplicación “CASified”: aplicación en la que se ha integrado un componente de software capaz de comunicarse con un servidor CAS. Habitualmente estas aplicación se conocen como CAS services y los componente embebidos como CAS clients.

18. Bibliografía

- CAS *Documentation*. (s.f.). Obtenido de Apereo:
<https://apereo.github.io/cas/6.1.x>
- Documentación OpenAM y OpenIG*. (s.f.). Obtenido de ForgeRock backstage:
<https://backstage.forgerock.com/>
- Ellinwood, J. (21 de Enero de 2020). *Como instalar Apache Tomcat 9 enm Ubuntu 18.04*. Obtenido de Digital Ocean:
<https://www.digitalocean.com/community/tutorials/install-tomcat-9-ubuntu-1804-es>
- HAProxy 2.2 starter guide*. (s.f.). Obtenido de HAProxy documentation:
<http://cbonte.github.io/haproxy-dconv/2.2/intro.html>
- Install Docker Engine on Ubuntu*. (s.f.). Obtenido de Docker docs:
<https://docs.docker.com/engine/install/ubuntu>
- jjuglans. (s.f.). *Install and configure HAProxy Load Balancer on Ubuntu 16.04*. Obtenido de Ionos: <https://devops.ionos.com/tutorials/install-and-configure-haproxy-load-balancer-on-ubuntu-1604>
- Lavanya, N., & Malarvizhi, T. (2008). *Risk analysis and management: a vital key to effective project management*. Obtenido de Paper presented at PMI® Global Congress 2008—Asia Pacific, Sydney, New South Wales, Australia. Newtown Square, PA: Project Management Institute:
<https://www.pmi.org/learning/library/risk-analysis-project-management-7070>
- Post-installation steps for Linux*. (s.f.). Obtenido de Docker docs:
<https://docs.docker.com/engine/install/linux-postinstall>
- Takahashi, K., & Bertino, E. (2010). *Identity Management: Concepts, Technologies, and Systems*. Artech House.
- Thadeshwar, S. (s.f.). *User & Device Identity for Microservices @ Netflix Scale*. Obtenido de InfoQ: <https://www.infoq.com/presentations/netflix-user-identity>
- Tomcat 8 Documentation*. (s.f.). Obtenido de Apache software foundation:
<http://tomcat.apache.org/tomcat-8.0-doc>
- Ubuntu*. (s.f.). Obtenido de OSBoxes: <https://www.osboxes.org/ubuntu/#ubuntu-19-10-vmware>
- Wang, C., Jan, S. T., Hu, H., Bossart, D., & Wang, G. (Marzo de 2018). *The Next Domino To Fall: Empirical Analysis of User Passwords across Online Services*. Obtenido de In Proceedings of The ACM Conference on Data and Applications Security and Privacy (CODASPY):
<https://people.cs.vt.edu/gangwang/pass>
- Weaver, P. (2008). The meaning of risk in an uncertain world. *PMI® Global Congress 2008—EMEA*. St. Julian's: Malta.

20. Anexo I: Instalación de OpenAM

20.1 Primer nodo

El asistente de instalación de OpenAM cuenta con los siguientes pasos después de aceptar la licencia de uso:

Configurador de OpenAM

Opciones de configuración personalizadas

General

2. Preferencias del servidor

3. Almacén de configuración

4. Almacén de usuarios

5. Configuración del sitio.

6. Resumen

Paso 1: General

Introduzca la contraseña del usuario predeterminado, amAdmin. La contraseña debe tener una longitud mínima de 8 caracteres. Si esta configuración va a formar parte de una implementación existente, la contraseña introducida debe coincidir con la de la implementación original.

* Indica un campo obligatorio

Contraseña de usuario predeterminada

Usuario predeterminado [amAdmin]

* Contraseña Correcto

* Contraseña (confirmar)

Anterior Siguiente Cancelar

Figura 31 - Creación de contraseña usuario amadmin

Configurador de OpenAM

Opciones de configuración personalizadas

1. General

Preferencias del servidor

3. Almacén de configuración

4. Almacén de usuarios

5. Configuración del sitio.

6. Resumen

Paso 2: Configuración del servidor

Confirme la siguiente configuración que se utilizará para el servidor.

* Indica un campo obligatorio

Preferencias del servidor

* URL del servidor

Dominio de cookies Correcto

* Idioma de plataforma

* Directorio de configuración

Anterior Siguiente Cancelar

Figura 32 - Configuración básica de servidor

Configurador de OpenAM

Opciones de configuración personalizadas

1. General
2. Preferencias del servidor
- Almacén de configuración
4. Almacén de usuarios
5. Configuración del sitio.
6. Resumen

Paso 3: Configuración del almacén de datos de configuración

Si no existe ninguna otra instancia de OpenAM en el entorno, elija la primera instancia. Si existen una o más instancias de OpenAM en el entorno, seleccione Agregar a implementación existente.

Primera instancia ¿Desea agregarla a una implementación existente? * Indica un campo obligatorio

Detalles del almacén de configuración

Almacén de datos de configuración OpenAM OpenDJ

* SSL habilitado

* Nombre de host

* Puerto Correcto

* Clave de cifrado Correcto

* Sufijo raíz Correcto

* Id. de inicio de sesión Correcto

* Contraseña Correcto

Figura 33 - Configuración del OpenDS de configuración

Configurador de OpenAM

Opciones de configuración personalizadas

1. General
2. Preferencias del servidor
3. Almacén de configuración
- Almacén de usuarios
5. Configuración del sitio.
6. Resumen

Paso 4: Configuración del almacén de usuario

Puede utilizar el almacén de datos incluido con el almacén de datos de configuración de OpenAM, o bien utilizar un almacén de datos de usuario diferente. Un procedimiento recomendado para configurar entornos de producción consiste en utilizar un almacén de datos de usuario externo diferente al almacén de datos de usuario de OpenAM. Tenga en cuenta que el servicio de directivas y el módulo de autenticación de LDAP estarán configurados para utilizar el ND del administrador de directorios y la contraseña facilitadas aquí.

Almacén de datos de usuario de OpenAM
 Otro almacén de datos de usuario * Indica un campo obligatorio

Detalles del almacén de usuario

* Tipo de almacén de datos del usuario OpenDJ Oracle Directory Server Enterprise Edition AD con nombre de dominio Active Directory con host y puerto Servidor de directorios IBM Tivoli Modo de aplicación de Active Directory

* SSL habilitado

* Nombre de directorio

* Puerto Correcto

* Sufijo raíz Correcto

* Id. de inicio de sesión Correcto

* Contraseña Correcto

Figura 34 - Configuración del almacén de usuarios

Configurador de OpenAM X

Opciones de configuración personalizadas

1. General
2. Preferencias del servidor
3. Almacén de configuración
4. Almacén de usuarios
- Configuración del sitio.
6. Resumen

Paso 5: Configuración del sitio 🔍

¿Se implementará esta instancia detrás de un equilibrador de carga como parte de la configuración del sitio?

No
 Sí

* Indica un campo obligatorio

Detalles de la configuración del sitio

Ésta es la primera instancia de OpenAM y no hay configuraciones de sitio que utilizar. Para crear una nueva configuración del sitio, proporcione la siguiente información

* Nombre del sitio Correcto

* URL del equilibrador de carga Correcto

Anterior
Siguiente
Cancelar

Figura 35 - Configuración del Site

Configurador de OpenAM X

Opciones de configuración personalizadas

1. General
2. Preferencias del servidor
3. Almacén de configuración
4. Almacén de usuarios
5. Configuración del sitio.
- Resumen

Detalles del resumen del configurador

Dedique un momento a revisar la configuración siguiente. Si alguno de los valores no es correcto, puede volver y modificarlos antes de realizar la configuración.

Detalles del resumen del configurador

Detalles del almacén de configuración [editar...](#)

| | |
|--|----------------------|
| SSL habilitado | No |
| Nombre de host | configuration-ds-01 |
| Puerto de escucha | 389 |
| Sufijo raíz | ou=am-config |
| Nombre de usuario | cn=Directory Manager |
| Nombre de directorio/home/forgerock/openam | |

Detalles del almacén de usuario [editar...](#)

| | |
|--------------------------------------|----------------------|
| SSL habilitado | No |
| Nombre de host | cts-ds-01 |
| Puerto de escucha | 389 |
| Sufijo raíz | ou=identities |
| Nombre de usuario | cn=Directory Manager |
| Tipo de almacén de datos del usuario | OpenDJ |

Detalles de la configuración del sitio [editar...](#)

| | |
|-------------------------------|----------------------------|
| Nombre del sitio | SSO |
| URL del equilibrador de carga | https://openam.example.com |

Anterior
Crear configuración
Cancelar

Figura 36 - Resumen de configuración

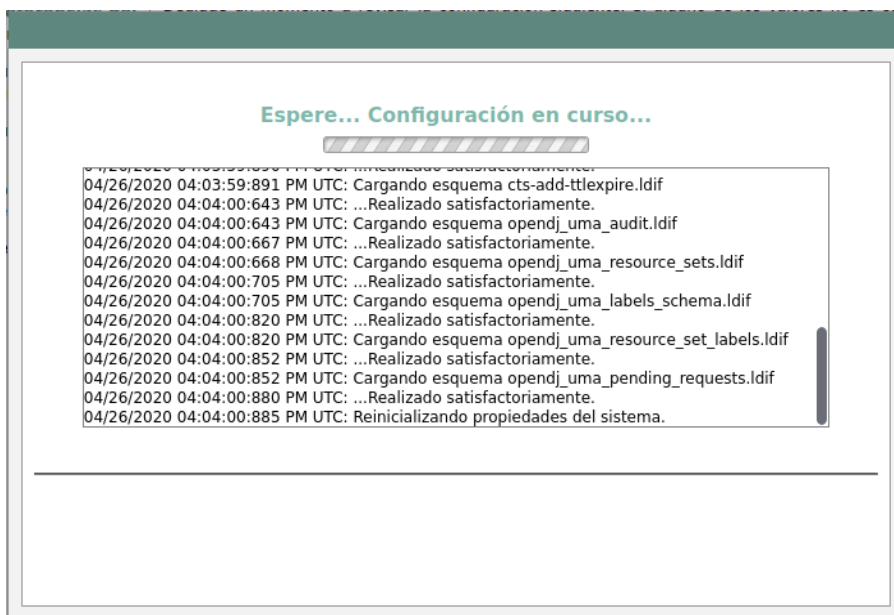


Figura 37 - Ejecución de configuración

Una vez finalizada la instalación, el último punto que se debe configurar es el almacén de tokens que sirve como soporte a los servicios de autenticación y a los distintos protocolos que implementa OpenAM. Para esto, abrimos el menú *Configuration > Server defaults > CTS > (Pestaña) External store configuration*, e incluimos los valores necesarios para referenciar las dos instancias de los OpenDS de tokens.

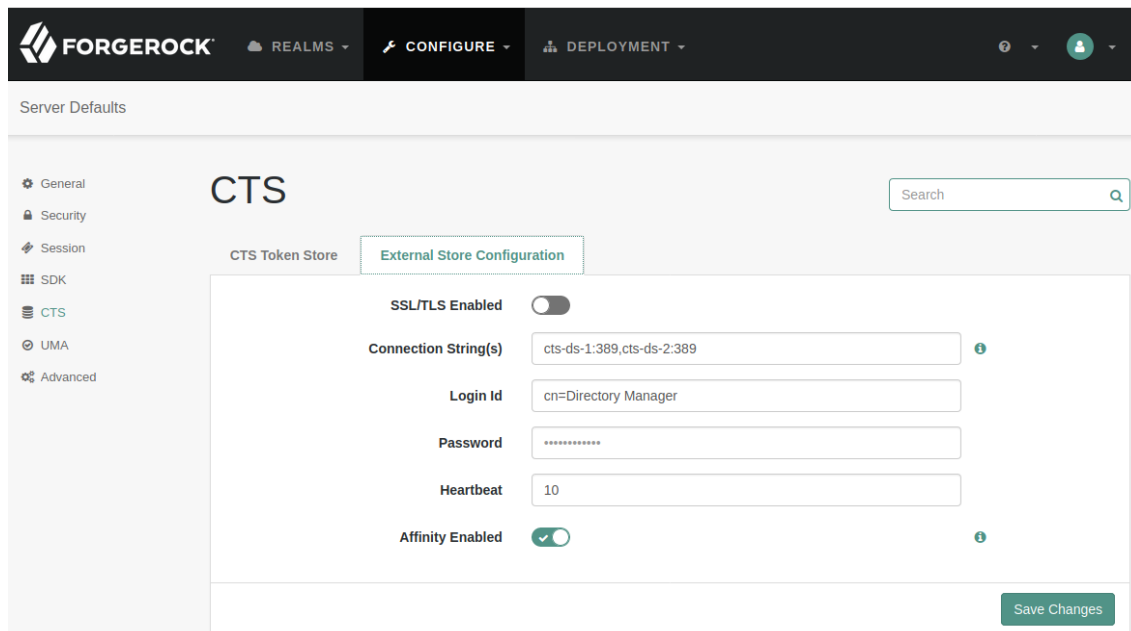


Figura 38 - Configuración de los almacenes de tokens

En la pestaña contigua (*CTS Token Store*) seleccionamos el valor “*External token store*” del desplegable “*Store mode*”. Como se puede ver en la captura, la cadena de conexión indica ambas instancias de los OpenDS de tokens de manera que si una de ellas no está disponible utilizará la otra. En el caso de los OpenDS de configuración e identidades la configuración de instancias en alta disponibilidad es similar.

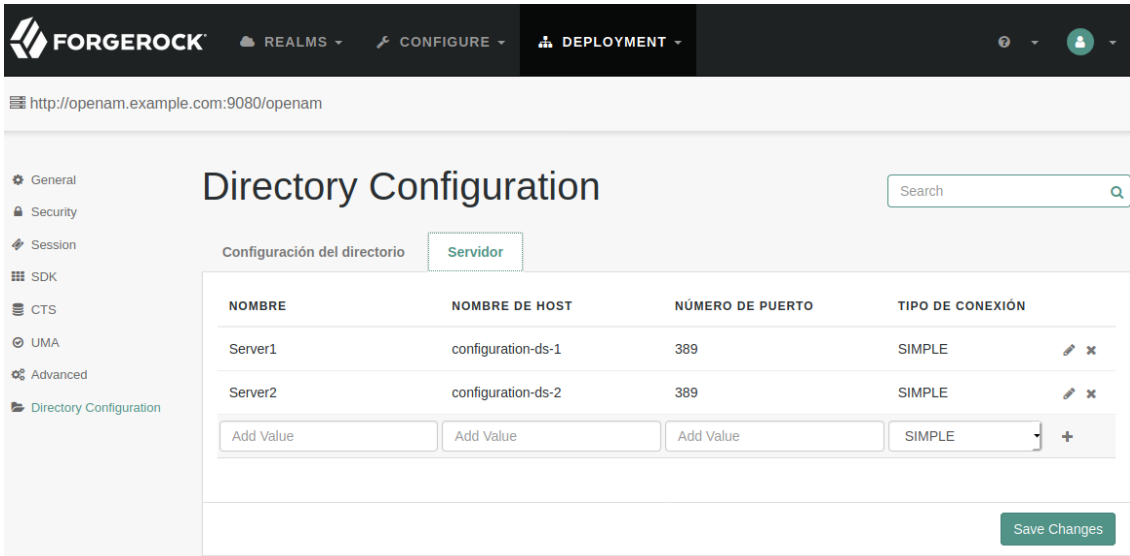


Figura 39 - Alta disponibilidad de almacén de configuración

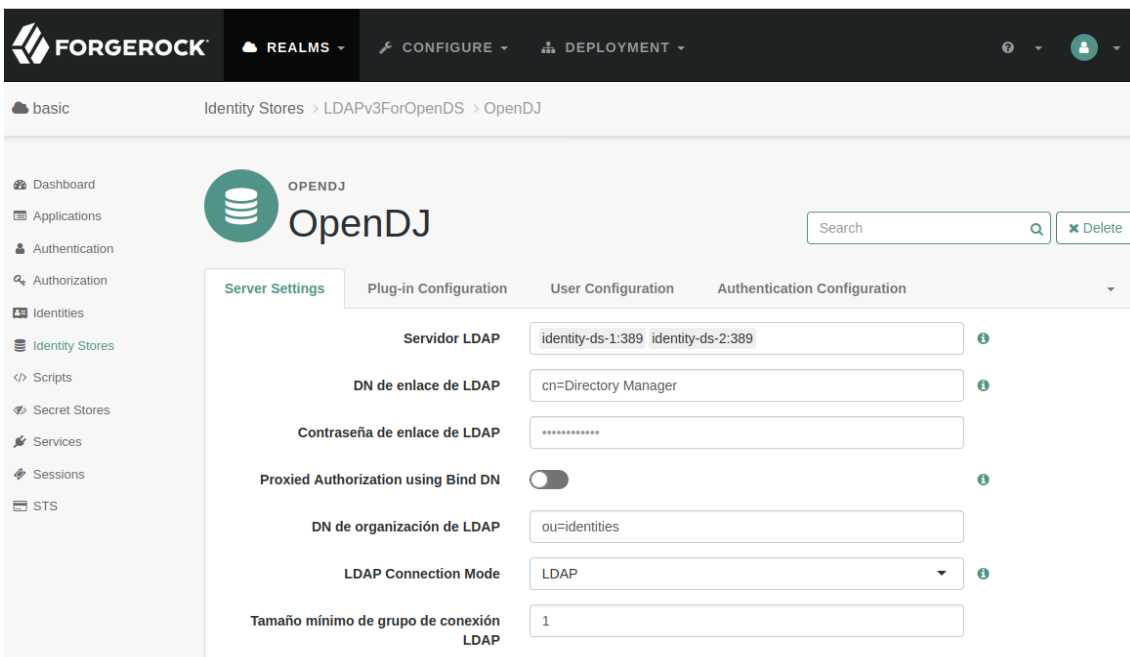


Figura 40 - Alta disponibilidad almacén de identidades

20.2 Segundo nodo y siguientes

La configuración de los siguientes nodos de OpenAM es muy similar a la ya vista para el primero pero con algún cambio en el asistente de configuración. En el paso 3, a la hora de indicar el almacén de configuración, debe seleccionarse la opción de incluir esta instancia de OpenAM en una implementación existente indicando la URL del OpenAM ya configurado.

Es destacable la manera de hacer referencia a otras aplicaciones ya configuradas en este asistente. Todas las referencias deben hacerse a los nombres de contenedor y puertos definidos dentro de la red personalizada de Docker (*ssso-network*) para hacer uso de su DNS interno.

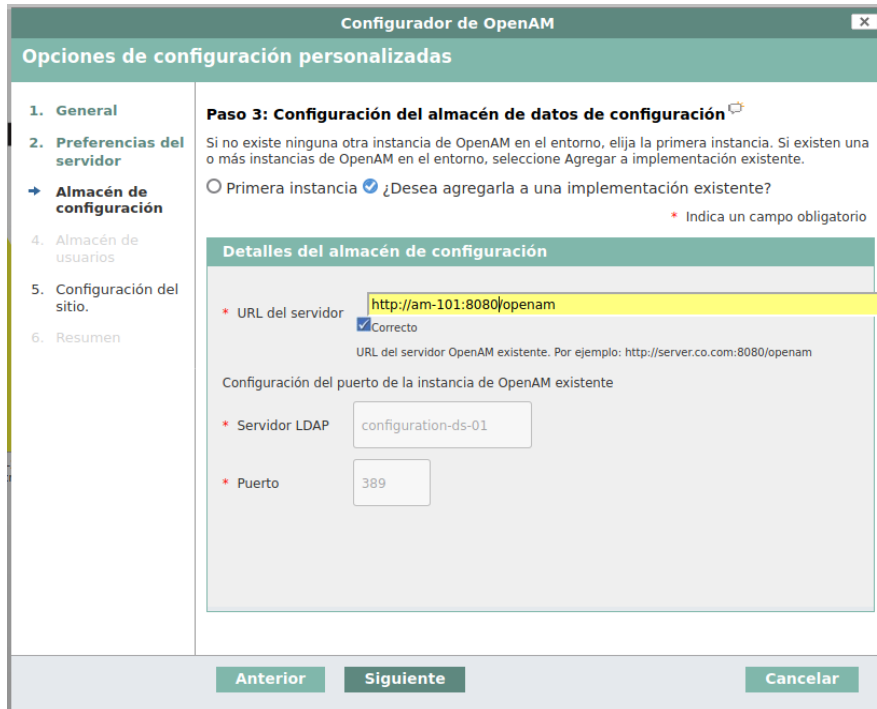


Figura 41 - Configuración de OpenAM en una instalación ya existente

21. Anexo II: Filtros de OpenIG

A continuación se exponen las configuraciones de cada uno de los filtros auxiliares de la cadena de filtros de OpenIG.

Session info filter

Config (optional)

```

1 {
2   "amService": "AmService-1"
3 }
```

Session info header

Config (optional)

```

1 {
2   "messageType": "REQUEST",
3   "add": {
4     "X-IG-SessionInfo": [ "username:
5     ${contexts.amSession.username}, realm:
6     ${contexts.amSession.realm}, properties:
7     ${contexts.amSession.properties}" ]
8   }
9 }
```

User profile

Config (optional)

```
1 {
2   "username" : "${contexts.ssoToken.info.uid}",
3   "userProfileService" : {
4     "type" : "UserProfileService",
5     "config" : {
6       "amService" : "AmService-1",
7       "profileAttributes" : [ "employeeNumber",
8 "mail", "givenName", "sn", "telephoneNumber", "cn"
9     ]
10  }
```

User profile header

Config (optional)

```
1 {
2   "messageType" : "REQUEST",
3   "add" : {
4     "X-IG-UserProfileInfo" : [ "username:
5 ${contexts.userProfile.username}, rawInfo:
6 ${contexts.userProfile.rawInfo}" ]
7   }
```