Implantación de un SSO (Single Sign On)

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

Proyecto de fin de máster

Autor: Enrique Barquilla Méntrida

Director: Antoni González Ciria

Empresa: ANCERT (Agencia Notarial de Certificación)

Fecha de realización: 01/01/2018

Universitat Oberta de Catalunya

Universitat Autònoma de Barcelona

Universitat Rovira i Virgili

Dedicatoria

Me gustaría dedicar este trabajo de fin de máster a mi mujer Elisa, por su comprensión y su apoyo durante esta aventura en la que decidí embarcarme hace ya dos años.

A mi hijo Mario, que ha tenido que sufrir el poco tiempo del que ha dispuesto su padre durante sus dos primeros años de vida. Cuando seas mayor te explicaré lo que hacía papá por las noches cuando todos os ibais a dormir.

Resumen

El trabajo de fin de máster consiste en la implementación de un mecanismo de SSO *(Single Sign On)* mediante soluciones de código abierto. El SSO nos va a permitir acceder a distintas aplicaciones, incluso si se encuentran en distintos dominios y servidores, simplemente autenticándose en el sistema una única vez.

El sistema implementado está formado por cuatro aplicaciones web a las que podrán acceder los usuarios autenticándose en un único servidor centralizado (servidor SSO) y que estará desplegado en alta disponibilidad. El acceso podrá realizarse por dos medios:

- Login: El usuario introduce usuario y contraseña en una página de login.
- Certificado digital: El usuario tiene instalado en su navegador web un certificado digital.

Las aplicaciones presentes en nuestro sistema, tendrán dos tipologías que determinarán el modo de acceso a ellas:

- Aplicaciones de administración: Acceso con certificado digital.
    - Gestión de usuarios
- Aplicaciones de usuario: Acceso con usuario y contraseña.
    - Asignaturas UOC.
    - Biblioteca.
    - Ofertas de empleo

Además, también se definen una serie de roles de acceso al sistema, que serán gestionados desde la aplicación de gestión de usuarios y los cuales determinarán la autorización de los usuarios para acceder a los distintos recursos o aplicaciones web, que ofrece la plataforma.

El sistema definido, será implementado en un entorno de laboratorio compuesto por distintas máquinas virtuales que serán ubicadas en dos redes diferenciadas: red DMZ *(Demilitarized Zone)* y red interna.

En este documento, se explican tanto los detalles funcionales como técnicos del sistema desarrollado, así como las distintas soluciones de hardware y software que se han utilizado.

Abstract

The final master project consists of the implementation of a SSO (Single Sign On) mechanism through open source solutions. The SSO will allow us to access different applications, even if they are in different domains and servers, simply authenticating in the system only once.

The implemented system consists of four web applications that users can access by authenticating themselves in a single centralized server (SSO server) and that will be deployed in high availability. Access can be made by two means:

- Login: The user writes a username and password on a login page.
- Digital certificate: The user has a digital certificate installed in their web browser.

The applications present in our system, will have two typologies that will determine the mode of access to them:

- Administration applications: Access with digital certificate.
    - User Management
- User applications: Access with username and password.
    - UOC subjects.
    - Library.
    - Job offers.

In addition, a set of system access roles are also defined, which will be managed from the user management application and which will determine the users' authorization to access the different resources or web applications offered by the platform.

The defined system will be implemented in a laboratory environment composed of different virtual machines that will be located in two differentiated networks: DMZ network (Demilitarized Zone) and internal network.

In this document, both the functional and technical details of the developed system are explained, as well as the different hardware and software solutions that have been used.

Índice de Contenidos

Introducción

Single Sign On (SSO)[1] es un procedimiento de autenticación que habilita al usuario para acceder a varios sistemas con una sola instancia de identificación. Por ejemplo, ingresando a Gmail podemos acceder a sus diferentes utilidades web, como Google Drive, Google Maps, Google Books, Youtube, etc.

Su traducción literal sería algo como "autenticación única" o "validación única". Un sistema SSO permite delegar la autenticación de los usuarios en un proveedor externo, el cual devolverá al usuario autenticado en el sistema, un ticket que deberá ser presentado a la aplicación a la que se desea acceder. La aplicación deberá validar la veracidad del ticket haciendo uso nuevamente del proveedor externo de autenticación. El ticket será válido para todas las aplicaciones que utilicen el mismo servidor de SSO para autenticarse en el sistema, por lo que el usuario sólo deberá introducir sus credenciales una única vez, y podrá tener el mismo usuario y contraseña para todas las aplicaciones. A continuación se muestra un diagrama general del funcionamiento de un sistema SSO:

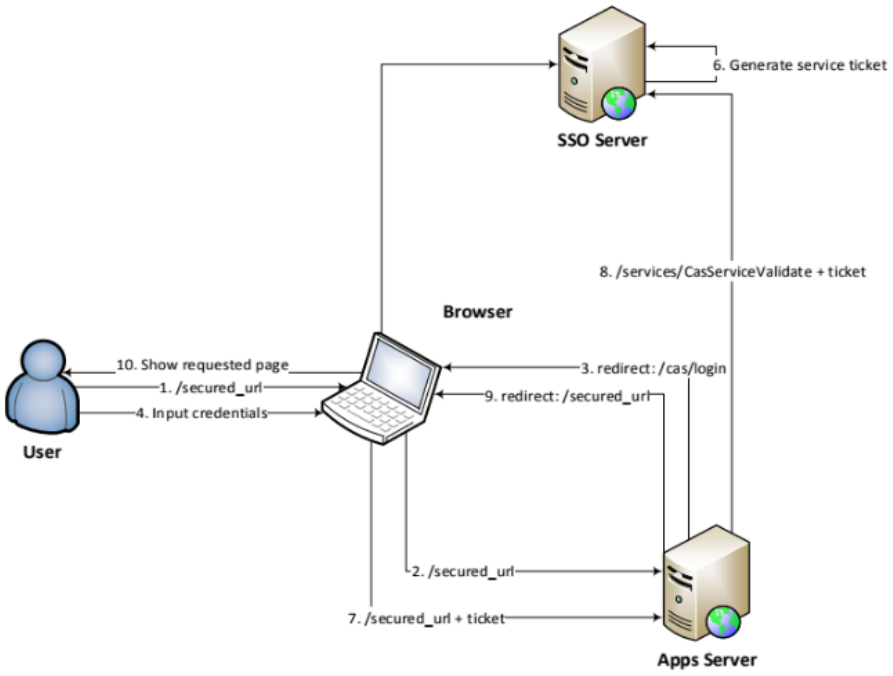

Ilustración 1. Esquema de SSO

Las principales características de un SSOCaracterísticas de un Single Sign On - https://www.chakray.com/que-es-el-single-sign-on-sso-definicion-caracteristicas-y-ventajas/[2] son las siguientes:

- Multiplataforma: Posibilidad de iniciar sesión y acceder a recursos desde distintas plataformas.
- Transparencia: El acceso a los distintos recursos del sistema para los que el usuario está autorizado, se realiza de manera transparente debido a la automatización del inicio de sesión.
- Facilidad de uso: El usuario se autentica una sola vez, por lo que evita las interrupciones producidas por la solicitud de usuario y contraseña para el acceso a diferentes recursos alojados en distintos sistemas.

- Gestión sencilla: El uso de SSO aconseja la centralización de contraseñas e información de usuarios. Esto implica la simplificación de la gestión de los recursos por parte de los administradores.
- Seguridad: En todos los casos, la información de los usuarios viajará cifrada por la red, mediante el uso de protocolos seguros.

Justificación

Implementar Single Sign On es una muy buena estrategia de autenticación ya que facilita la vida tanto al usuario (sólo tiene que memorizar un único usuario y contraseña) como al desarrollador de las aplicaciones, que puede despreocuparse de la autenticación de los usuarios al delegarla en una entidad externa.

Desde el punto de vista de la seguridad, este sistema de autenticación mejora la seguridad de la red y de las aplicaciones, identificando a un usuario inequívocamente. Toda la información proporcionada al servidor SSO, será transmitida mediante protocolo seguro, es decir, viajará cifrada por la red. Las soluciones SSO mejoran la experiencia del usuario evitando las interrupciones producidas por las solicitudes de contraseñas para acceder a sus herramientas informáticas. El usuario se autentica una única vez y el sistema le permite acceder a los recursos para los cuales está autorizado.

Un sistema de este tipo, presenta numerosas ventajas como la mejora de la seguridad, de la usabilidad del sistema por parte del usuario, la gestión sencilla por parte del administrador, etc, sin embargo, también hay que tener en cuenta su principal desventaja. Si un usuario accede a todos los sistemas con una sola combinación de credenciales, si estas son vulneradas, se obtendrá acceso a todos los sistemas que compartan el mismo servidor de SSO. Por otro lado, si el servidor SSO falla, se pierde el acceso a todos los sistemas, por este motivo, normalmente estos servidores suelen estar desplegados en alta disponibilidad o HA (High Availability)[3].

Contexto

Actualmente, se suelen distinguir cinco tipos de sistemas SSO[1]:

- Enterprise Single Sign-On (E-SSO): Esta técnica es conocida como legacy single sign-on y su principal uso es el de interactuar con sistemas que pueden deshabilitar la presentación de las pantallas de autenticación. Esta técnica es utilizada para lo que se conoce como autenticación primaria, puesto que intercepta los requisitos de autenticación del sistema y se completan con el usuario y contraseña que están almacenando en los servidores.
- **Web Single Sign On (Web-SSO):** Esta técnica es conocida como Web access management (Web-AM) y su principal uso es con aplicaciones y recursos accedidos vía Web. Los accesos son interceptados utilizando un servidor proxy o un componente encargado de esta función instalado en el servidor Web o aplicación Web de destino. Los usuarios que no han sido aún autenticados son redirigidos a un servidor de autenticación o a un servicio Web que haga el acceso. Se suelen utilizar como mecanismos de autenticación las famosas cookies, parámetros GET/POST, cabeceras del protocolo http.
- Kerberos: Es un protocolo de autenticación creado por el MIT en el que los usuarios se registran en el servidor Kerberus y reciben un identificador (ticket), el cual es utilizado en las aplicaciones para obtener el acceso.
- Identidad federada: En esta técnica las empresas comparten información sin compartir tecnologías, seguridad y autenticación, sino que se basa en el "círculo de confianza" entre las diferentes partes. Para conseguir intercambiar esta información se han utilizado soluciones basadas en XML**.**

- OpenID: Es una técnica de SSO distribuida y descentralizada donde la identidad es compilada en una URL para que cualquier aplicación o servidor pueda verificar la veracidad del usuario. Los sitios Web que soportan OpenID no requieren crearse una cuenta de usuario para acceder sino que solamente se debe proporcionar el identificador creado en un servidor OpenID.

El proyecto de fin de máster estará basado en un sistema de tipo Web Single Sign On, el cual se implementará mediante el producto de software libre CAS (Central Authentication Service). Web Single Sign On, funciona estrictamente con aplicaciones a las que se accede con un navegador web. Los usuarios no autenticados, se desvían a un servicio de autenticación (CAS) y dan acceso al recurso web solicitado únicamente después de una autenticación exitosa.

Para hacer posible la implantación de CAS, en primer lugar crearemos la infraestructura necesaria, mediante máquinas virtuales que estarán dispuestas en dos redes, una DMZ (Zona Desmilitarizada) y una interna. Desde la DMZ y a través de una capa de proxies, se podrá acceder a los servidores de la red interna, entre los que se encontrarán los servidores SSO desplegados en alta disponibilidad y el servidor de aplicaciones, donde se encontrarán desplegadas las aplicaciones web a las que dará acceso el CAS.

Como particularidad respecto a sistemas SSO más clásicos donde tanto la autorización como la autenticación se controlan desde un único sistema o aplicación (típicamente un LDAP), nuestro sistema será diseñado de tal forma que la autenticación sea controlada mediante un servidor LDAP, sin embargo, la autorización será gestionada por un componente web que hará uso de una base de datos en la que se asociarán a los usuarios dados de alta en el sistema a distintos roles, los cuales darán acceso a las distintas aplicaciones web presentes en la plataforma.

Metodología

La metodología seleccionada para la realización del proyecto ha sido un modelo incremental, por el cual, se definen una serie de entregas parciales del producto para que se puedan ir evaluando. Tendrá la particularidad de que para las fases de requerimientos, análisis y diseño, se seguirá un modelo en cascada tradicional y una vez definido el modelo funcional y el diseño del sistema, pasaremos al modelo incremental por el cual se entregarán distintos evolutivos hasta finalmente alcanzar el producto final.

Las fases serán las siguientes:

1. Definición de requisitos
2. Análisis funcional
3. Diseño
4. Desarrollo
5. Despliegue
6. Pruebas

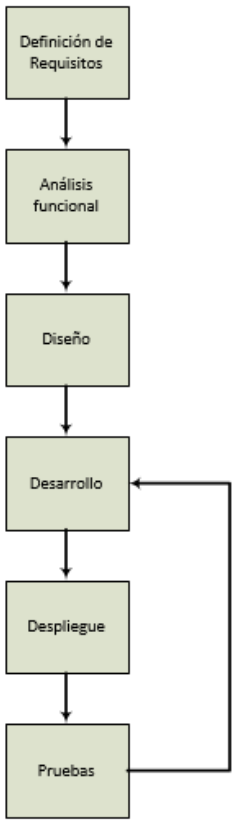

Ilustración 2. Grafo metodología

Objetivos

- Implementación de sistema SSO mediante solución de software libre CAS (Central Authentication Service).
- Implementación de autenticación en el SSO a través de distintos métodos: usuario y contraseña y certificado digital.
- Comunicaciones seguras entre todos los servidores.
- Implementación de red perimetral.
- Instalación, configuración y securización de distintos tipos de servidores.
    - Servidor web
    - Servidor de aplicaciones
    - Servidor LDAP
    - Servidor de base de datos
- Alta disponibilidad de los sistemas críticos.
- Desarrollo de aplicación web para la gestión de roles del sistema.
- Desarrollo de aplicaciones web, para validar la correcta autenticación en ellos a través del sistema implementado.

Planificación

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1		Plan de Trabajo	58,25 días	dom 20/08/17	lun 09/10/17	
2		**Entrega de PEC 1 - Plan de trabajo**	**1 día**	**lun 09/10/17**	**lun 09/10/17**	
3		Análisis funcional	4 días	mar 10/10/17	vie 13/10/17	2
4		**Diseño**	**5 días**	**sáb 14/10/17**	**mar 17/10/17**	3
5		Diseño arquitectónico	3 días	sáb 14/10/17	dom 15/10/17	
6		Diseño interfaz de aplicaciones web	2 días	lun 16/10/17	mar 17/10/17	5
7		**Preparación infraestructura**	**11 días**	**mié 18/10/17**	**vie 27/10/17**	4
8		Instalación y configuración de máquinas virtuales	3 días	mié 18/10/17	vie 20/10/17	
9		Instalación de sistemas operativos en máquinas virtuales	1 día	sáb 21/10/17	sáb 21/10/17	8
10		Configuración red DMZ	3 días	sáb 21/10/17	lun 23/10/17	9
11		Configuración red interna	2 días	mar 24/10/17	mié 25/10/17	10
12		Configuración de Firewall	2 días	jue 26/10/17	vie 27/10/17	11
13		**Instalación y configuración de servidores**	**8,25 días**	**sáb 28/10/17**	**sáb 04/11/17**	7
14		Instalación y configuración de Apache	1 día	sáb 28/10/17	sáb 28/10/17	
15		Instalación y configuración de Tomcat.	1 día	sáb 28/10/17	dom 29/10/17	14
16		Instalación y configuración de JBoss.	1 día	dom 29/10/17	dom 29/10/17	15
17		Instalación y configuración de OracleXE	1 día	lun 30/10/17	lun 30/10/17	16
18		Generación base de datos y usuario	0,25 días	lun 30/10/17	lun 30/10/17	
19		Generación de scripts	0,5 días	lun 30/10/17	lun 30/10/17	18
20		Ejecución de scripts	0,25 días	lun 30/10/17	lun 30/10/17	19

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
21		Instalación y configuración de OpenLDAP.	1,25 días	mar 31/10/17	mié 01/11/17	17
22		Generación de esquema	1 día	mar 31/10/17	mar 31/10/17	
23		Carga de datos	0,25 días	mié 01/11/17	mié 01/11/17	22
24		Instalación y configuración de CAS en Tomcat	3 días	mié 01/11/17	sáb 04/11/17	21
25		**Pruebas de integración de la plataforma sin securizar con aplicación dummy desplegada en Jboss.**	**3 días**	**sáb 04/11/17**	**lun 06/11/17**	13
26		**Entrega de PEC 2**	**1 día**	**lun 06/11/17**	**lun 06/11/17**	
27		Implementación aplicaciones web.	3 días	mar 07/11/17	jue 09/11/17	26
28		Creación de esquemas en base de datos para cada aplicación y generación de scripts dcl, ddl, y dml.	2 días	vie 10/11/17	sáb 11/11/17	27
29		Despliegue de aplicaciones web en Jboss	3,5 días	sáb 11/11/17	mar 14/11/17	28
30		**Securización de plataforma**	**15,25 días**	**mar 14/11/17**	**lun 27/11/17**	29
31		Generación de certificados digitales con OpenSSL.	1,25 días	mar 14/11/17	mié 15/11/17	
32		Certificado servidor Apache	0,25 días	mar 14/11/17	mar 14/11/17	
33		Certificado servidores Tomcat.	0,25 días	mar 14/11/17	mar 14/11/17	32
34		Certificado servidor JBoss	0,25 días	mié 15/11/17	mié 15/11/17	33
35		Certificado servidor LDAP.	0,25 días	mié 15/11/17	mié 15/11/17	34
36		Certificado de cliente.	0,25 días	mié 15/11/17	mié 15/11/17	35
37		Instalación de certificados	3 días	mié 15/11/17	sáb 18/11/17	31

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
38		Comunicación https con servidor Apache	2,5 días	sáb 18/11/17	lun 20/11/17	37
39		Comunicación https con servidores Tomcat	2,5 días	lun 20/11/17	mié 22/11/17	38
40		Comunicación https con servidores Jboss	2,5 días	mié 22/11/17	sáb 25/11/17	39
41		Comunicación LDAPS entre CAS y LDAP	3,5 días	sáb 25/11/17	lun 27/11/17	40
42		Implementación de alta disponibilidad en servidores CAS.	5 días	lun 27/11/17	sáb 02/12/17	30
43		**Pruebas de integración de la plataforma securizada**	3 días	sáb 02/12/17	lun 04/12/17	42
44		**Entrega PEC 3**	**1 día**	**lun 04/12/17**	**lun 04/12/17**	
45		**Memoria Final**	32 días	mar 05/12/17	lun 01/01/18	44
46		**Entrega PEC 4 - Memoria Final**	**1 día**	**lun 01/01/18**	**lun 01/01/18**	
47		**Presentación y vídeo**	8 días	mar 02/01/18	lun 08/01/18	46
48		**Entrega PEC 5 - Presentación y vídeo**	**1 día**	**lun 08/01/18**	**lun 08/01/18**	
49		**Defensa del TFM**	**5 días**	**lun 15/01/18**	**vie 19/01/18**	

Las variaciones que ha sufrido el desarrollo del proyecto respecto a su planificación inicial, han sido las siguientes:

- Entrega de la PEC 2 sin pruebas completas de la plataforma sin securizar. Se realizaron durante la siguiente fase del proyecto.
- Entrega de la PEC 3 sin pruebas completas de la plataforma securizada. Como en el caso anterior, se realizaron durante la siguiente fase.
- La tarea 45 (Memoria final), finalmente se ha realizado como una tarea transversal durante todas las fases del proyecto, lo que ha permitido completar en la última fase las tareas pendientes.
- La tarea 17 (Instalación y configuración de OracleXE), ha sufrido una pequeña variación que no afecta al alcance del proyecto. Finalmente el producto seleccionado como servidor de base de datos, ha sido MySQL, por su mayor facilidad de instalación y configuración, y su compatibilidad total con el objetivo definido.
- Las tareas, 12 (Configuración de Firewall), 40 (Comunicación https con servidores JBoss) y 41 (Comunicación LDAPS entre CAS y LDAP), no se han realizado por falta de tiempo.

Organización de la memoria

La memoria del trabajo de fin de máster ha sido organizada del siguiente modo:

- En primer lugar, nos encontramos con una introducción en la que se detalla brevemente en qué consiste un sistema Single Sign On y cuáles son sus principales características. Durante la introducción, se justifica el uso de este tipo de sistemas y se describen los principales tipos de SSO existentes. Finalmente, se muestra la metodología empleada y el desglose de tareas identificadas, así como su planificación.
- En un segundo bloque, se disponen una serie de secciones en las cuales se detalla el trabajo realizado. Se incluye una descripción técnica de las distintas soluciones de software utilizadas, detallando las distintas funcionalidades que presentan y las que se han utilizado para la realización del proyecto. Una vez descritos las distintas soluciones que utilizaremos, pasaremos al análisis y diseño de la plataforma desarrollada. Finalmente, describiremos la implementación y configuración de los componentes que forman el sistema.

- En la última sección de la memoria, se incluyen las conclusiones, donde se relacionan los objetivos conseguidos y no conseguidos, y las posibles mejoras y ampliaciones que se pueden realizar sobre el alcance del proyecto. Finalmente se incluyen las referencias bibliográficas utilizadas durante la realización de la memoria.
- Se incluye un anexo con distintos aspectos de la instalación y configuración de los componentes que no han sido añadidos a la memoria.

Fundamentos teóricos

A continuación se describen las soluciones de software que se han utilizado para implementar la plataforma:

Software de virtualización

El término virtualización en un contexto informático se utiliza para referirnos a la creación mediante software de una versión virtual de algún recurso tecnológico, como puede ser una plataforma de hardware, un sistema operativo, un dispositivo de almacenamiento u otros recursos de red.

El software de virtualización implementa lo que se llama un hipervisor o VMM (Virtual Machine Monitor) que consiste en una capa de abstracción entre el hardware de la máquina física (host anfitrión) y la máquina virtual (MV) formada por hardware y software virtualizado, haciendo el papel de centralita entre lo real y lo virtualizado[3].

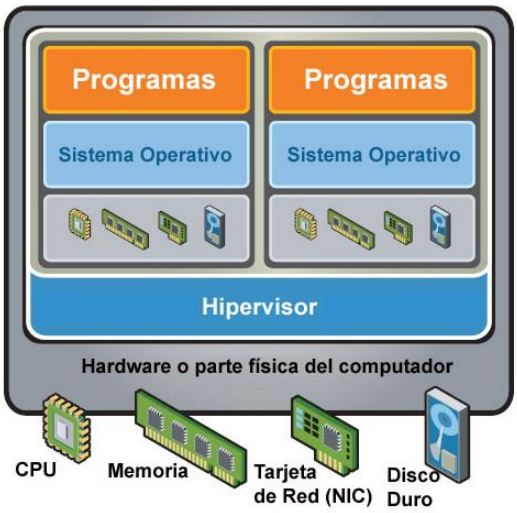

Ilustración 3. Arquitectura virtualización

Esta capa de software, el hipervisor o VMM, maneja, gestiona y arbitra los cuatro recursos principales de un ordenador (CPU, memoria, disco y tarjeta de red) y así puede repartir dinámicamente dichos recursos entre todas las máquinas virtuales creadas en el ordenador anfitrión. Esto permite que se puedan tener varias máquinas virtuales ejecutándose en el mismo ordenador físico.

Oracle VirtualBox

Para la virtualización de nuestra plataforma, se ha elegido **Oracle VirtualBox 5.2.0**. Oracle VirtualBox es un software de virtualización para arquitecturas x86/amd64, creado originalmente por la empresa alemana innotek GmbH. Actualmente es desarrollado por Oracle Corporation como parte de su familia de productos de virtualización.

La aplicación fue inicialmente ofrecida bajo una licencia de software privativo, pero en enero de 2007, después de años de desarrollo, surgió VirtualBox OSE (Open Source Edition) bajo la licencia GPL 2. Actualmente existe la versión privativa Oracle VM VirtualBox, que es gratuita únicamente bajo uso personal o de evaluación, y está sujeta a la licencia de "Uso Personal y de Evaluación VirtualBox" (VirtualBox Personal Use and Evaluation License o PUEL) y la versión Open Source, VirtualBox OSE, que es software libre, sujeta a la licencia GPL.

VirtualBox es un software de virtualización donde los usuarios pueden cargar múltiples sistemas operativos invitados en un solo sistema anfitrión. Cada invitado se puede configurar, iniciar, pausar o parar de forma independiente. El sistema operativo anfitrión y los sistemas operativos invitados pueden comunicarse entre sí a través de una serie de mecanismos, entre ellos un portapapeles común, carpetas compartidas, arrastrando y soltando ficheros, etc.

Una máquina virtual en VirtualBox puede tener hasta 8 tarjetas de red PCI Ethernet, y a cada una de ellas, de forma independiente, se les puede especificar el tipo de hardware a virtualizar y el modo de configuración. Desde el Administrador de VirtualBox se pueden configurar 4 de las 8 tarjetas de red, si necesitamos más, debemos configurarlas desde la línea de comandos con VBoxManage.

Cada tarjeta puede tener un modo de funcionamiento y éste se elige en la sección de Red de la máquina virtual, en el apartado de Conectado a.

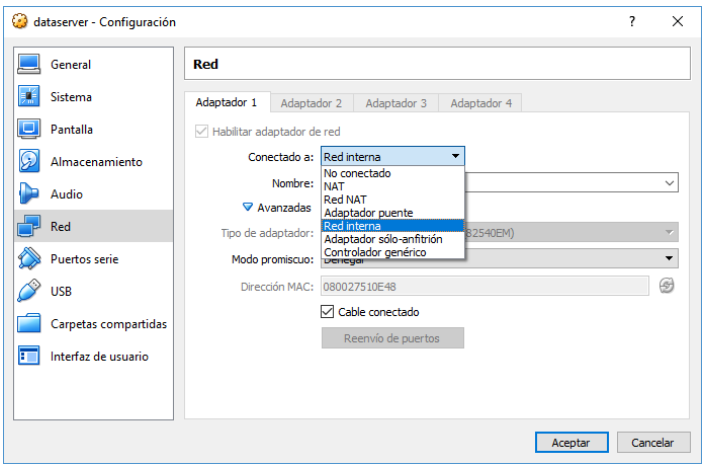

Ilustración 4. Configuración Red Virtual Box

Los distintos modos en los que puede trabajar una tarjeta de red creada mediante VirtualBox, son los siguientes[3]:

- No Conectado: En este modo, VirtualBox informa al cliente de que una tarjeta de red está presente, pero que no hay ninguna conexión, como si el cable de red no estuviera conectado a la tarjeta.
- Modo NAT: En modo NAT, VirtualBox coloca un router entre el exterior (hacia donde hace NAT) y el invitado. Dicho router posee un servidor DHCP que sirve hacia el interior. Este router mapea el tráfico desde y hacia la MV de forma transparente. Cada máquina virtual en modo NAT tendrá su propio router, por lo que estarán en redes aisladas, lo que implica, que por defecto, las máquinas virtuales que tienen su tarjeta de red en modo NAT no pueden verse entre sí.
- Modo Red NAT: El modo Red NAT, funciona como el router de nuestra casa, es decir, los equipos que estén dentro de la misma red NAT podrán comunicarse entre sí, y es aquí donde radica la diferencia con el modo NAT el cual siempre constituye una red con un único equipo y no de varios como ahora es el caso.

- Modo Adaptador puente: El modo Adaptador puente simula que la tarjeta virtual está conectada al mismo switch que la tarjeta física del anfitrión, por lo tanto, la MV se va a comportar como si fuese un equipo más dentro de la misma red física en la que está el equipo anfitrión.
- Modo Red interna: Con la configuración de tarjetas de red en modo Red interna, podemos construir redes aisladas, en las cuales sólo habrá comunicación entre las máquinas virtuales que pertenezcan a la misma red interna.
- Modo Sólo-anfitrión: El modo Sólo-anfitrión se utiliza para crear una red interna a la que pertenecerá también el equipo anfitrión, algo que no sucede en el modo Red interna.

Sistemas Operativos

Un sistema operativo es el software principal o conjunto de programas de un sistema informático que gestiona los recursos de hardware y provee servicios a los programas de aplicación de software, ejecutándose en modo privilegiado respecto de los restantes (aunque puede que parte de él se ejecute en espacio de usuario). El centro de un sistema operativo es el núcleo o kernel. El núcleo es el programa más importante en la computadora, realiza todo el trabajo básico y le permite ejecutar otros programas[6].

Debian

El sistema operativo instalado en todas las máquinas virtuales que forman la plataforma es **Debian Stretch 9.2.1.** El Proyecto Debian es una asociación de personas que han hecho causa común para crear un sistema operativo libre. Este sistema operativo que han creado se llama Debian.

Los sistemas Debian actualmente usan el núcleo de Linux o de FreeBSD. Linux es una pieza de software creada en un principio por Linus Torvalds y desarrollada por miles de programadores a lo largo del mundo. FreeBSD es un sistema operativo que incluye un núcleo y otro software.

Una gran parte de las herramientas básicas que completan el sistema operativo, vienen del proyecto GNU; de ahí los nombres: GNU/Linux, GNU/kFreeBSD, y GNU/Hurd. Estas herramientas también son libres.

Servidores

Un servidor es una aplicación en ejecución (software) capaz de atender las peticiones de un cliente y devolverle una respuesta en concordancia. Los servidores se pueden ejecutar en cualquier tipo de máquina, incluso en máquinas dedicadas. En la mayoría de los casos una misma máquina puede proveer múltiples servicios y tener varios servidores en funcionamiento. La ventaja de montar un servidor en máquinas dedicadas es la seguridad. Por esta razón la mayoría de los servidores son procesos diseñados de forma que puedan funcionar en computadoras de propósito específico.

Comúnmente los servidores proveen servicios esenciales dentro de una red, ya sea para usuarios privados dentro de una organización o compañía, o para usuarios públicos a través de Internet. Los tipos de servidores más comunes son servidor de base de datos, servidor de ficheros, servidor de correo, servidor de impresión, servidor web y servidor de aplicaciones[8].

Servidor Web

Un servidor web o servidor HTTP es un programa informático que procesa una aplicación del lado del servidor, realizando conexiones bidireccionales o unidireccionales y síncronas o asíncronas con el cliente y generando o cediendo una respuesta en cualquier lenguaje o Aplicación del lado del cliente. El código recibido por el cliente es renderizado por un navegador web[9].

HTTP Apache

El servidor web utilizado en nuestra plataforma es **HTTP Apache 2.4.25.** El servidor HTTP Apache es un servidor web HTTP de código abierto, para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.12 y la noción de sitio virtual.

El servidor Apache es desarrollado y mantenido por una comunidad de usuarios bajo la supervisión de la Apache Software Foundation dentro del proyecto HTTP Server (httpd). Tiene amplia aceptación en la red: desde 1996, Apache, es el servidor HTTP más usado. En 2009 se convirtió en el primer servidor web que alojó más de 100 millones de sitios web.

La arquitectura del servidor Apache es modular. El servidor consta de una sección core y diversos módulos que aportan mucha de la funcionalidad que podría considerarse básica para un servidor web.

A continuación, vamos a describir los módulos que ofrece Apache y que se han utilizado durante la implementación de nuestra plataforma de SSO[10]:

Virtual Host: El módulo de Apache donde se encuentra esta utilidad es su *core*, por lo que una vez instalado Apache, no tendremos que realizar ninguna acción adicional para poder utilizarla.

El término Virtual Host se refiere a la posibilidad de ejecutar más de un sitio web en una sola máquina o host. Los Virtual Hosts, pueden estar basados en IP y puerto, lo que significa que tienen una dirección IP y/o puerto diferente para cada sitio web, o basados en el nombre, lo que quiere decir que tienen varios nombres ejecutándose en una misma dirección IP y puerto.

Por ejemplo, suponiendo que tenemos el dominio www.example.com y queremos agregar el Virtual Host other.example.com que apunta a la misma dirección IP, tendríamos que agregar la siguiente configuración al fichero *httpd.conf* de Apache:

```
<VirtualHost *:80>
    # This first-listed virtual host is also the default for *:80
    ServerName www.example.com
    ServerAlias example.com
    DocumentRoot "/www/domain"
</VirtualHost>

<VirtualHost *:80>
    ServerName other.example.com
    DocumentRoot "/www/otherdomain"
</VirtualHost>
```

Si tenemos varios dominios que van a la misma dirección IP y que quieren servir varios puertos, tendríamos que configurar Apache del siguiente modo:

```
Listen 80
Listen 8080

<VirtualHost 172.20.30.40:80>
    ServerName www.example.com
    DocumentRoot "/www/domain-80"
</VirtualHost>

<VirtualHost 172.20.30.40:8080>
    ServerName www.example.com
    DocumentRoot "/www/domain-8080"
</VirtualHost>
```

Reverse Proxy: Apache puede comportarse como un servidor proxy inverso. En tales escenarios, el servidor http no aloja ni genera los datos, sino que el contenido es obtenido por uno o varios servidores backend, que normalmente no tienen conexión directa a la red externa. Cuando el servidor Apache recibe una petición, ésta es enviada a dichos servidores backend, que la procesan y devuelven la respuesta http correspondiente a Apache que se encarga de generar la respuesta http real al cliente que hizo la petición. Existen numerosas razones para el uso de un Reverse Proxy, pero

en general los fundamentos típicos se deben a la seguridad, la alta disponibilidad, el balanceo de carga y la autenticación/autorización centralizada. Es fundamental que en este tipo de arquitecturas, los servidores backend estén aislados del exterior y que el único modo de acceso sea a través de los proxies configurados en el servidor Apache. A continuación se muestra una implementación típica de un Reverse Proxy:

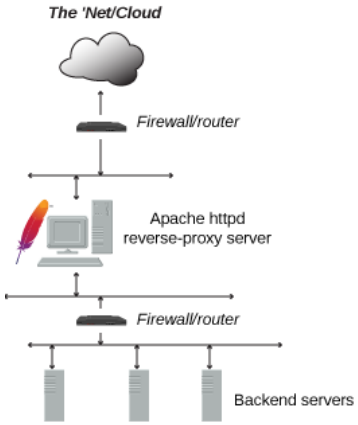

Ilustración 5. Esquema Reverse Proxy

Los módulos necesarios para poder implementar un Reverse Proxy con Apache son *mod_proxy* y *mod_proxy_http*. A continuación se muestra un ejemplo simple de uso de las directivas *ProxyPass* y *ProxyPassReverse*:

```
ProxyPass "/foo" "http://foo.example.com/bar"
ProxyPassReverse "/foo" "http://foo.example.com/bar"
```

La directiva *ProxyRequests Off* evita que el frontend sea utilizado como proxy, es decir, que usuarios puedan saltar al frontend y de ahí a cualquier otra dirección. Es muy importante dejarlo deshabilitado para evitar problemas de seguridad o incluso legales.

La directiva *ProxyPreserveHost On* permite que el salto del servidor de frontend al de backend sea transparente para el usuario. Si no estuviera habilitada, el usuario se dirigiría a http://frontend.com pero inmediatamente vería como la dirección cambia a http://backend.com. Además, como en este supuesto el servidor de backend no es visible desde Internet el usuario vería un error.

High availability: La alta disponibilidad (high availability) permite que un servicio funcione correctamente ante un fallo software o hardware. Una configuración basada en software consiste en una serie de servidores denominados nodos, conectados entre sí de tal manera que, ante un fallo de un nodo, sus servicios son repartidos a los demás nodos que forman el clúster.

Para implementar alta disponibilidad, Apache dispone del módulo *mod_proxy_balancer*, mediante el cual se pueden definir un conjunto de servidores backend que puedan procesar las solicitudes recibidas. El Reverse Proxy, será capaz de realizar un balanceo de carga entre los nodos backend y detectar la pérdida de servicio de uno de los nodos para dirigir las peticiones a los demás nodos que forman el clúster o *balancer*, como es denominado por Apache. A continuación, se muestra un ejemplo sencillo de una configuración con un balanceador y dos nodos:

```
<Proxy "balancer://mycluster">
    BalancerMember "http://192.168.1.50:80"
    BalancerMember "http://192.168.1.51:80"
</Proxy>
ProxyPass "/test" "balancer://mycluster"
ProxyPassReverse "/test" "balancer://mycluster"
```

En la actualidad, existen tres algoritmos de planificación del balanceo de carga entre los nodos del clúster: Request Counting, Weighted Traffic Counting and Pending Request Counting. Estos se controlan mediante el valor *lbmethod* en la definición del *Balancer*. Los tipos de *lbmethod* disponibles son: *mod_lbmethod_byrequests*, *mod_lbmethod_bytraffic*, *mod_lbmethod_bybusyness* y *mod_lbmethod_heartbeat*. A continuación se muestra un ejemplo con *lbmetod = bytraffic*, por el cual el nodo http://ww3.example.com:8080, recibirá tres veces más tráfico que el nodo http://ww2.example.com:8080:

```
<Proxy balancer://myset>
    BalancerMember http://www2.example.com:8080
    BalancerMember http://www3.example.com:8080 loadfactor=3 timeout=1
    ProxySet lbmethod=bytraffic
</Proxy>

ProxyPass "/images"  "balancer://myset/"
ProxyPassReverse "/images"  "balancer://myset/"
```

Si no indicamos nada, por defecto el balanceador utiliza *lbmethod_byrequest.*

Stickness[29]: El balanceador de carga de Apache permite que definamos lo que se denomina Stickness, lo cual permitirá que cuando una solicitud se envíe a un determinado nodo, todas las demás solicitudes del mismo usuario se enviarán al mismo. Muchos balanceadores de carga implementan esta característica a través de una tabla que asigna direcciones IP de clientes a servidores backend. Este enfoque es transparente para clientes y servidores, pero adolece de algunos problemas como por ejemplo la distribución de carga desigual si los clientes están ocultos detrás de los proxies, errores cuando el cliente utiliza una IP dinámica y pérdida de Stickness si la tabla de asignación se desborda.

El módulo *mod_proxy_balancer*, implementa Stickness mediante cookies, de esta manera eliminamos los problemas con las tablas de asignaciones que describimos anteriormente. Por ejemplo, una configuración utilizando la cookie de sesión JSESSIONID, sería la siguiente:

```
<Proxy "balancer://cascluster">
  BalancerMember "https://casserver1:8443/cas-server-webapp-4.0.0" route=casserver1
  BalancerMember "https://casserver2:8443/cas-server-webapp-4.0.0" route=casserver2
</Proxy>

ProxyPass /cas-server-webapp-4.0.0 balancer://cascluster stickysession=JSESSIONID
ProxyPassReverse /cas-server-webapp-4.0.0 balancer://cascluster stickysession=JSESSIONID
```

SSL: Mediante el uso del protocolo SSL v3, se consigue que las comunicaciones entre el servidor Apache y el resto de máquinas, estén cifradas. El módulo *mod_ssl*, proporciona soporte sobre los protocolos SSL V3 y TLS v1.x. El módulo puede ser configurado para proporcionar numerosa información mediante variables de entorno, por ejemplo:

- HTTPS: Flag que indica si HTTPS está siendo utilizado.
- SSL_PROTOCOL: Versión del protocolo SSL utilizado.
- SSL_CLIENT_S_DN: Valor del atributo DN del certificado de cliente.
- SSL_CLIENT_I_DN: Valor del atributo DN del emisor del certificado de cliente.
- SSL_CLIENT_CERT: Certificado de cliente.

Las principales directivas del módulo *mod_ssl* de Apache, que nos permiten trabajar con SSL, son:

- *SSLEngine On*: Activamos el uso del protocolo SSL en el virtual host donde se haya definido.

- *SSLProxyEngine On*: Activamos el uso del protocolo SSL cuando utilizamos proxies.
- *SSLCertificateFile*: Indicamos la ruta del certificado de servidor. Imprescindible para poder realizar la comunicación SSL.
- *SSLCertificateKeyFile*: Ruta del fichero con la Clave privada del certificado de servidor.
- *SSLCACertificateFile*: Ruta del fichero con la CA o autoridad certificadora que emite los certificados admitidos por el servidor.
- SSLVerifyClient: Indica si es necesario que el cliente que realiza la petición al servidor, debe disponer de un certificado de cliente. Esta directiva puede tomar os siguientes valores:
    - none: No se requiere certificado de cliente.
    - optional: El cliente puede presentar un certificado válido.
    - require: El cliente tiene que presentar un certificado válido.
    - optional_no_ca: El cliente debe presentar un certificado pero no es necesario que sea verificado con éxito.
- *SSLProxyVerify*: Cuando utilizamos proxies, activando esta directiva podemos comprobar que el servidor remoto al que accedemos, tiene un certificado válido.

Por defecto, el puerto con el que el servidor Apache trabaja con SSL es el 443 (https), pero podemos habilitar su uso en cualquier otro puerto utilizando las directivas vistas anteriormente.

Servidores de aplicaciones

Se denomina servidor de aplicaciones a un servidor en una red ordenadores que ejecuta ciertas aplicaciones. Normalmente se trata de un dispositivo de software que proporciona servicios de aplicación a las máquinas cliente. Un servidor de aplicaciones generalmente gestiona la mayor parte (o la totalidad) de las funciones de lógica de negocio y de acceso a los datos de la aplicación. Los principales beneficios de la aplicación de la tecnología de servidores de aplicación son la centralización y la disminución de la complejidad en el desarrollo de aplicaciones[11].

Apache Tomcat

En primer lugar, cabe señalar que Tomcat[12] no es realmente un servidor de aplicaciones como tal, sino que se trata de un contenedor de Servlets, capaz de ejecutar páginas web desarrolladas con JSP (JavaServer Page). Incluye el compilador Jasper, que compila JSP's convirtiéndolas en servlets. La principal diferencia es que un servlet proporciona respuestas http, es decir, está vinculado a un servicio web, mientras que un servidor de aplicaciones como WildFly que veremos a continuación, tiene soporte para EJB's, a través de los cuales se consigue acceso RMI (Remote Method Invocation), por lo que su uso no es exclusivo para web.

La versión de **Apache Tomcat** que vamos a utilizar es la **8.5.23**. Apache Tomcat es una implementación de código abierto de las tecnologías Java Servlet, JavaServer Pages, Java Expression Language y Java WebSocket. Las especificaciones Java Servlet, JavaServer Pages, Java Expression Language y Java WebSocket se desarrollan bajo Java Community Process.

El software Apache Tomcat se desarrolla en un entorno abierto y participativo y se publica bajo la versión 2 de Apache License. Apache Tomcat es una marca registrada de la Apache Software Foundation.

Para poder comunicarse con el servidor, es necesario definir conectores. El conector HTTP está configurado por defecto y listo para ser usado una vez concluida la instalación de Tomcat. Para poder disponer de un clúster de servidores Tomcat y así implementar alta disponibilidad, será necesario tener un balanceador de carga con soporte para stickiness que dirija el tráfico a los distintos servidores Tomcat del clúster. Como vimos en el apartado anterior, este tipo de balanceador está disponible con el módulo *mod_proxy_balancer* del servidor web Apache. Para que

el balanceador del servidor web funcione correctamente, será necesario indicar el nombre dado al nodo por el balanceador, en el atributo *jvmRoute* del elemento de configuración *Engine*.

```
<Engine name="Catalina" defaultHost="localhost" jvmRoute="casserver1">
```

Otra opción para conectar el servidor web con el servidor Apache Tomcat es el conector AJP. El funcionamiento es equivalente al que proporciona el conector HTTP, sin embargo, proporcionará un rendimiento superior que el HTTP con Proxy,

Podemos seleccionar el conector que más nos interese y configurarlo desde el fichero *server.xml*:

- Conector HTTP en puerto 8080:

```
<!-- A "Connector" represents an endpoint by which requests are received
     and responses are returned. Documentation at :
     Java HTTP Connector: /docs/config/http.html
     Java AJP  Connector: /docs/config/ajp.html
     APR (HTTP/AJP) Connector: /docs/apr.html
     Define a non-SSL/TLS HTTP/1.1 Connector on port 8080
-->
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
```

- Conector AJP en puerto 8009:

```
<!-- Define an AJP 1.3 Connector on port 8009 -->
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

Configuración SSL en Tomcat: Para implementar SSL, un servidor debe tener un Certificado asociado para cada interfaz externa (dirección IP) que acepte conexiones seguras. Este certificado está firmado criptográficamente por su propietario y, por lo tanto, es extremadamente difícil de falsificar. Para que el certificado sea aceptado por los navegadores de clientes sin advertencias, debe estar firmado por un tercero de confianza, lo que conocemos como Autoridades de Certificación (CA).

Java proporciona una herramienta de línea de comandos relativamente simple, llamada keytool, que puede crear fácilmente un certificado autofirmado. Los certificados autofirmados son simplemente certificados generados por el usuario que no han sido firmados por una CA reconocida y, por lo tanto, no están garantizados en absoluto como auténticos. Si bien los certificados autofirmados pueden ser útiles para algunos escenarios de prueba, no son adecuados para ninguna forma de uso de producción. Tomcat actualmente solo funciona en almacenes de claves de formato JKS, PKCS11 o PKCS12. El formato JKS es el formato estándar de Java (Java KeyStore) y es el formato creado por la herramienta de línea de comandos keytool, la cual está incluida en el JDK (Java SE Development Kit).

Una vez hemos generado e importado en el Keystore de Tomcat el certificado de servidor necesario, a continuación debemos configurar el conector correspondiente, indicando dicho Keystore y la contraseña para acceder a él:

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
<Connector
           protocol="org.apache.coyote.http11.Http11NioProtocol"
           port="8443" maxThreads="200"
           scheme="https" secure="true" SSLEnabled="true"
           keystoreFile="${user.home}/.keystore" keystorePass="changeit"
           clientAuth="false" sslProtocol="TLS"/>
```

Si queremos que el servidor Tomcat solicite certificados al cliente, tendremos que indicárselo a través del atributo *clientAuth*, el cual admite tres posibles valores:

- False: No es necesario certificado de cliente.
- Want: El cliente puede aportar un certificado de cliente válido.
- True: El cliente tiene que aportar un certificado de cliente válido.

WildFly

Se ha utilizado la versión **WildFly-9.0.1.Final**. WildFly anteriormente conocido como JBoss AS, o simplemente JBoss, es un servidor de aplicaciones Java EE de código abierto implementado en Java, lo que le permite ser utilizado en cualquier sistema operativo para el que esté disponible la máquina virtual de Java, al igual que ocurre con Apache Tomcat. WildFly es software libre y de código abierto, sujeto a los requisitos de la GNU Lesser General Public License (LGPL), versión 2.1[13].

Wildfly tiene dos modos de administración: Standalone y Domain. Cada modo le permite administrar los servidores de forma diferente utilizando distintas topologías[14].

- Modo Standalone es el modo tradicional de funcionamiento. Básicamente implica tener una instalación diferente (o un directorio independiente) para cada instancia de Wildfly. Es decir, para cada Wildfly rodando en su entorno es necesario cambiar su propio archivo de configuración, sus propias opciones de ejecución para JVM, etc.

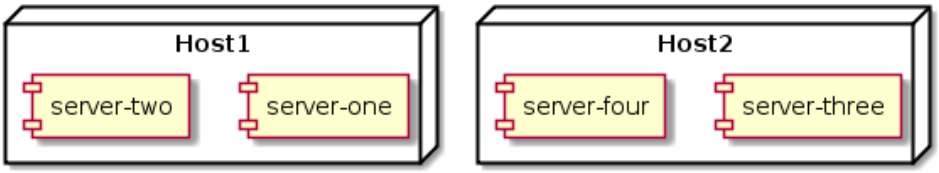

Ilustración 6. Modo Standalone WildFly

- Modo Domain es el modo que se introdujo en JBoss AS 7 donde es posible administrar un conjunto de instancias Wildfly, agrupándolas y permitiendo compartir configuraciones comunes entre ellos.

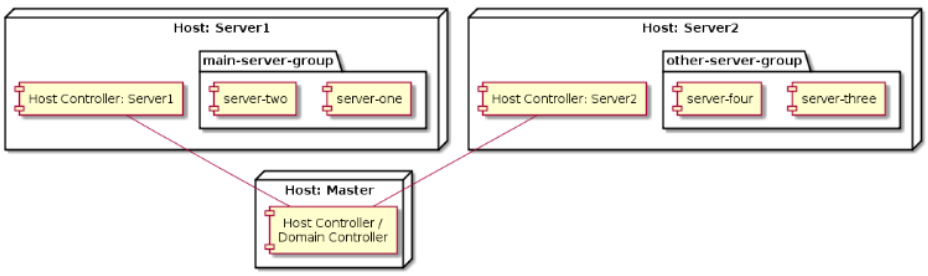

Ilustración 7. Modo Domain WildFly

Será necesario definir en el fichero de configuración los puertos en los que queremos que escuche nuestro servidor, por defecto, 8080 http y 8443 https. Se podrán definir en el elemento socket-binding-group:

```
    <socket-binding-group name="standard-sockets" default-interface="public" por
t-offset="${jboss.socket.binding.port-offset:0}">
        <socket-binding name="management-http" interface="management" port="${jb
oss.management.http.port:9990}"/>
        <socket-binding name="management-https" interface="management" port="${j
boss.management.https.port:9993}"/>
        <socket-binding name="ajp" port="${jboss.ajp.port:8009}"/>
        <socket-binding name="http" port="${jboss.http.port:8080}"/>
        <socket-binding name="https" port="${jboss.https.port:8443}"/>
        <socket-binding name="txn-recovery-environment" port="4712"/>
        <socket-binding name="txn-status-manager" port="4713"/>
        <outbound-socket-binding name="mail-smtp">
            <remote-destination host="localhost" port="25"/>
        </outbound-socket-binding>
    </socket-binding-group>
```

Configuración SSL en WildFly: Para configurar SSL en WildFly, en primer lugar, al igual que con Apache Tomcat, tendremos que generar un certificado de servidor e importarlo en un keystore. A continuación, tendremos que referenciarlo en el siguiente elemento de configuración:

```
<https-listener name="default-ssl" socket-binding="https" security-realm="SslRealm"/>

<security-realm name="SslRealm">
    <server-identities>
            <ssl>
                <keystore path="yourdomain.com.jks" relative-to="jboss.server.config.dir" keystore-password="<secret password>"/>
        </ssl>
    </server-identities>
</security-realm>
```

Servidor de base de datos

Un servidor de base de datos relacional, también conocidos como RDBMS (acrónimo en inglés de Relational DataBase Management Systems), son programas que permiten organizar datos en una o más tablas relacionadas. Los servidores de Bases de Datos se utilizan en todo el mundo en una amplia variedad de aplicaciones.

MySQL (MariaDB)

El sistema gestor de base de datos utilizado ha sido la versión **10.1.26** de **MariaDB**. MariaDB es un sistema de gestión de bases de datos derivado de MySQL con licencia GPL (General Public License). Tiene una alta compatibilidad con MySQL ya que posee las mismas órdenes, interfaces, APIs y bibliotecas, siendo su objetivo poder cambiar un servidor por otro directamente. Surge a raíz de la compra de Sun Microsystems, compañía que había comprado previamente MySQL AB, por parte de Oracle. MariaDB es un fork directo de MySQL que asegura la existencia de una versión de este producto con licencia GPL.

Para poder acceder al servidor de base de datos desde otra máquina, es necesario modificar el fichero de configuración *50-server.cnf*. En él, existen las dos propiedades siguientes: *skip-external-locking*, la cual comentaremos para que nos permita hacer login en las bases de datos creadas desde máquinas externas y *bind-address*, donde tendremos que indicar las direcciones IP de las máquinas a las que se permite el acceso. Si comentamos esta propiedad, se podrá acceder desde cualquier máquina.

Servidor LDAP

LDAP son las siglas de Lightweight Directory Access Protocol que hacen referencia a un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red. Habitualmente, almacena la información de autenticación (usuario y contraseña) y es utilizado para autenticarse aunque es posible almacenar otra información (datos de contacto del usuario, ubicación de diversos recursos de la red, permisos, certificados, etc)[16].

El LDAP Data Interchange Format (LDIF) es un formato que se utiliza para la importación y exportación de datos independientemente del servidor LDAP que se esté utilizando. Cada servidor

LDAP tiene una o varias maneras de almacenar físicamente sus datos, por este motivo LDIF proporciona un método de unificar el tratamiento de los datos y así poder migrar de un servidor a otro sin importar el tipo de LDAP con el que estemos trabajando.

OpenLDAP

El LDAP seleccionada para incorporarlo a nuestra plataforma SSO se trata de **OpenLDAP 2.4.44**. OpenLDAP es una implementación de código abierto del protocolo Lightweight Directory Access Protocol (LDAP) desarrollada por el proyecto OpenLDAP. Está liberado bajo su propia licencia OpenLDAP Public License[17].

La suite de OpenLDAP, incluye[18]:

- Slapd: Servidor LDAP standalone.
- Liberías que implementan el protocolo LDAP.
- Aplicaciones cliente para operar con el LDAP, como *slapcat* para mostrar el contenido del LDAP en formato LDIF, *ldapsearch* para buscar una determinada entrada en el directorio, *ldapadd* para añadir una nueva entrada, *ldapdelete* para eliminar una entrada…

Backends: La arquitectura del servidor OpenLDAP está dividida en dos niveles: *frontend* que maneja las conexiones de redes y el procesamiento del protocolo, y una base de datos *backend* que se ocupa del almacenamiento de los datos. El servidor slapd puede utilizar arbitrariamente varios backends a la vez y tener arbitrariamente varias instancias de cada backend (por ejemplo varias bases de datos) activas por vez.

Overlays: Generalmente una petición LDAP es recibida por el frontend, decodificada y luego transferida a un backend para su procesamiento. Cuando el backend completa la petición, devuelve un resultado al frontend, quien luego envía el resultado al cliente LDAP. Un overlay es una pieza de código que puede ser insertada entre el frontend y el backend. Es entonces capaz de interceptar peticiones y lanzar otras acciones en ellas antes de que el backend las reciba, y puede también actuar sobre los resultados del backend antes de que éstos alcancen el frontend. Algunos overlays incluidos en el core de OpenLDAP son:

- *Accesslog*: Servidor de registro de actividades en otra base de datos LDAP.
- D*eref*: Devuelve información acerca de entradas referenciadas en resultados de búsqueda dados.
- *Dyngroup*: Soporte simple de grupos dinámicos.
- *Memberof*: Soporte de memberOf y atributos similares.
- *Refint*: Integridad referencial.
- *Trace*: Lleva un registro de peticiones y respuestas LDAP.

Generación de certificados X.509

Un certificado de clave pública es un punto de unión entre la clave pública de una entidad y uno o más atributos referidos a su identidad. El certificado garantiza que la clave pública pertenece a la entidad identificada y que la entidad posee la correspondiente clave privada. La entidad identificada se denomina sujeto del certificado. Para que un certificado digital sea válido, tendrá que ser emitido por alguna Autoridad Certificadora (Certification Authority o CA), ya que si uno se certifica a sí mismo no hay ninguna garantía de que su identidad sea la que anuncia, y por lo tanto, no debe ser aceptada por un tercero que no lo conozca. Para evitar la falsificación de certificados, la entidad certificadora después de autentificar la identidad de un sujeto, firma el certificado digitalmente.

El formato de certificados X.509 es un estándar del ITU-T (International Telecommunication Union-Telecommunication Standarization Sector) y el ISO/IEC (International Standards Organization / International Electrotechnical Commission) que se publicó por primera vez en 1988. El formato de la versión 1 fue extendido en 1993 para incluir dos nuevos campos que permiten soportar el control de acceso a directorios. Después de emplear el X.509 v2 para intentar desarrollar un estándar de correo electrónico seguro, el formato fue revisado para permitir la extensión con campos adicionales, dando lugar al X.509 v3, publicado en 1996[19].

OpenSSL y Keytool

Para la generación de los certificados digitales necesarios para securizar la comunicación entre los servidores de la plataforma, vamos a utilizar las herramientas OpenSSL y Keytool.

OpenSSL: OpenSSL[20] es un proyecto de software libre basado en SSLeay, desarrollado por Eric Young y Tim Hudson. Consiste en un robusto paquete de herramientas de administración y bibliotecas, que suministran funciones criptográficas a otros paquetes como OpenSSH y navegadores web (para acceso seguro a sitios mediante el protocolo HTTPS). Estas herramientas ayudan al sistema a implementar el Secure Sockets Layer (SSL), así como otros protocolos relacionados con la seguridad, como el Transport Layer Security (TLS). OpenSSL también permite crear certificados digitales que pueden aplicarse a un servidor, por ejemplo Apache.

Keytool: Keytool[21] es una utilidad que encontramos en la JDK de Java y que nos permite administrar los Keystore o almacenes de claves con los que trabaja Java.

A continuación se describe el proceso para crear una CA propia y los certificados necesarios firmados por dicha CA, para poder securizar la comunicación entre clientes y servidores:

1. Generación de CA
   a. Generación de clave
      ```
      openssl genrsa -out cakey.pem 1024
      ```
   b. Petición de firma (csr)
      ```
      openssl req -new -key cakey.pem -out cacsr.csr
      ```
   c. Generación de certificado firmado con la clave generada con un periodo de validez de 365 días
      ```
      openssl x509 -req -days 365 -in cacsr.csr -signkey cakey.pem -out cacert.pem
      ```
2. Generación de certificado de servidor
   a. Generación de clave
      ```
      openssl genrsa -out serverkey.pem 1024
      ```
   b. Petición de firma (csr)
      ```
      openssl req -new -key serverkey.pem -out server.csr
      ```
   c. Firma del csr por la CA creada anteriormente
      ```
      openssl x509 -req -days 365 -CA cacert.pem -CAkey cakey.pem -in server.csr -CAcreateserial -out server.pem
      ```
3. Generación de certificado de cliente
   a. Generación de clave
      ```
      openssl genrsa -out clientkey.pem 1024
      ```
   b. Petición de firma (csr)
      ```
      openssl req -new -key clientkey.pem -out client.csr
      ```
   c. Firma del csr por la CA creada anteriormente

```
openssl x509 -req -days 365 -CA cacert.pem -CAkey cakey.pem -
in client.csr -CAcreateserial -out client.pem
```

d. Exportar certificado a formato p12 para poder ser importado en un navegador

```
openssl pkcs12 -export -clcerts -in client.csr -inkey
clientkey.pem -out client.p12
```

4. Generar keystore con el certificado de servidor generado
   a. Generación de keystore y certificado autofirmado que será sustituido posteriormente

```
keytool -genkey -alias server -keyalg RSA
```

   b. Importar la cadena del certificado dentro del keystore

```
keytool -import -alias root -trustcacerts -file cacert.pem
```

   c. Importar el certificado de servidordentro del keystore, sustituyendo al creado en primer lugar para crear el almacén

```
Keytool -import -alias server -file server.pem
```

Desarrollo de aplicaciones web

Para el desarrollo de las distintas aplicaciones web que se encuentran desplegadas en los servidores de aplicaciones de la plataforma, hemos utilizado el siguiente software:

IDE Eclipse

La versión utilizada de Eclipse ha sido **Eclipse Java EE IDE for Web Developers. Versión Neon 3, Release 4.6.3**. Eclipse es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido". Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el propio Eclipse)[22].

Java

El código de las aplicaciones y su compilación se han realizado con la **JDK 1.8.0_122** de Java. Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos, que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como WORA, o "write once, run anywhere"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Esto es posible gracias a la JVM (Java Virtual Machine), que se encarga de encarga de ejecutar el código Java compilado en bytecode en cualquier plataforma[23].

Apache Maven

La versión de Maven que se ha utilizado, ha sido **Apache Maven 3.3.9.** Se ha utilizado integrado en Eclipse a través de los plugins de integración disponibles. Maven es un software de gestión de proyectos y una herramienta de compresión. Maven utiliza un Project Object Model (POM) para describir el proyecto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos[24].

Apache CXF

Se ha utilizado la versión de **Apache CXF 2.2.12**. Apache CXF es un framework completo, de código abierto para servicios web. Incluye un amplio conjunto de características, entre las que destaca el soporte de estándares de servicios web como SOAP, WSAddresing o WSSecurity, y las APIs para el

desarrollo de servicios web JAX-WS (Java y wsdl) y servicios web de tipo RESTful JAX-RS. Proporciona una serie de herramientas para la generación de código (*wsdl2java, wsdl2js* y *java2js*), generación de wsdl (Web Service Description Language) (*java2ws, xsd2wsdl* y *idl2wsdl*) y para uso con Maven (*Maven Java2WS plugin*), entre otras[25].

Framework Spring

La versión utilizada ha sido **Spring-framework 4.1.1**. Spring es un framework para el desarrollo de aplicaciones Java. Spring dispone de multitud de librerías que proporcionan un gran número de servicios al desarrollador, como por ejemplo, control del ciclo de vida de los objetos Java, programación orientada a aspectos, acceso a datos, gestión de transacciones, acceso remoto, implementación del modelo vista-controlador, etc.

Spring Framework utiliza el concepto de inyección de objetos mediante el Inversion of Control (IoC). El Contenedor se encarga de gestionar los ciclos de vida de objetos de los objetos inyectados: la creación de estos objetos, llamando a sus métodos de inicialización, y configurando estos objetos. Los objetos creados por IoC también se denominan beans. IoC puede configurar mediante la carga de archivos XML o la detección de anotaciones Java específicas sobre la configuración de las clases. Estas fuentes de datos contienen las definiciones que proporcionan la información necesaria para la creación de las beans[26].

Spring Security

Se ha utilizado la versión **Spring Security 3.1.3**. Spring Security[27] es un framework que se encuentra dentro del ecosistema de Spring y está enfocado en proporcionar autenticación y autorización a las aplicaciones Java. Como todos los proyectos de Spring, el poder real de Spring Security se encuentra en la facilidad con la que se puede personalizar e integrar con distintos sistemas, por ejemplo, con un servidor CAS mediante las librerías *spring-security-cas*.

JQuery

Se ha utilizado la versión **JQuery 3.2.1**. JQuery[28] es una biblioteca multiplataforma de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privados. Al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

Bootstrap

Se ha utilizado la versión **Bootstrap 3.3.7**. Bootstrap[29] es un framework web de código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales adicionales.

Software SSO

Actualmente en el mercado, existen una gran variedad de implementaciones del mecanismo Single Sign On. A continuación se muestran algunas de las más destacadas[31]:

Producto	Propietario	Licencia	Descripción
Xpress Sign-On	Ilantus Technologies	Commercial Software	Admite SSO para cualquier tipo de aplicación, incluyendo Web SSO. Soporte para gestión de contraseñas,

			gestión del ciclo de vida del usuario y control de accesos.
Access One	Pirean	Propietary	Gestión de accesos e identidades, inicio de sesión en la nube con registro de usuario. Admite web SSO y SSO federado (SAML, OAuth2, OpenID, JWT, etc)
ORY Hydra	ORY GmbH	Free Software (Apache 2.0)	Implementación de OAuth2 y OpenID Connect (SSO federado)
Portal Guard	Pistol Star, Inc.	Propietary	Paquete de autenticación integrada, todo en uno: SSO, 2FA, SSPR, etc.
Enterprise SSO, Web Access Manager	Evidian	Commercial Software	Admite Enterprose SSO, web SSO y SSO federado. Proporciona una plataforma de gestión de identidad global.
Accounts & SSO	Intel, Canonical Ltd, KDE	Free Software	Arquitectura basada en plugins que trabaja con distintas interfaces de usuario, backends de almacenamiento y niveles de seguridad.
Open Athens	Eduserv	Propietary	Servicio de gestión de identidad y acceso. Los proveedores de identidad (IdP), pueden mantener las identificaciones de usuarios en la nube, en local o en ambos. Admite integración con ADFS, LDAP o SAML. Admite múltiples plataformas.
CAS (Central Authentication Service)	Apereo	Free & Open Source (Apache 2.0)	Implementación de SSO con soporte para SAML1, SAML2, Outh2, SCIM, OpenID Connect WS-Fed. Proveedor de identidad y de servicios junto con otras funciones auxiliares.
Enterprise Sign On Engine	Queensland University of Technology	Free Software	Es una plataforma de código abierto para SSO, control de acceso y federación. El core está desarrollado en Java y la conectividad entre servicios se logra mediante SAML.
Facebook Connect	Facebook	Propietary	Se trata del SSO de Facebook para terceros habilitados. Es una API de autenticación de Facebook que los desarrolladores pueden usar para ayudar a sus usuarios a conectarse a su sitio web o aplicación.
IBM Enterprise Identity Mapping	IBM	Free Software	IBM Enterprise Identity Mapping (EIM) es un framework de IBM que permite el mapeo de diferentes identidades en diversas plataformas, repositorios de usuarios y aplicaciones a una única identidad.
JBoss SSO	Red Hat	Free Software	Es un producto de la suite JBoss SOA para permitir el SSO y el acceso federado a múltiples aplicaciones y recursos informáticos en la red.
JOSSO	JOSSO	Free Software	Java Open Single Sign On (JOOSO) es una plataforma de gestión de acceso e identidad (IAM) para SSO en la nube y basado en estándares, seguridad de servicios web y aprovisionamiento.
Kerberos	MIT	Es un protocolo	Es un protocolo de autenticación de red que trabaja mediante el uso de tickets para permitir que los nodos que se comunican a través de una red no seguro prueben su identidad entre sí de forma segura.

Microsoft Account	Microsoft	Propietary	Se trata del servicio SSO que ofrece Microsoft para logarse en sitios web, como Outllook, dispositivos como un PC con Windows 10, y aplicaciones como Visula Studio, todas ellas desarrolladas por la empresa.
MyOneLogin	VMWare	Propietary	Es un SSO en la nube. Todos los servicios están completamente integrados y se suministran bajo demanda sin necesidad de implementar hardware, software o modificar aplicaciones.
OpenAM	ForgeRock	Open Source	Se trata de un producto para la implementación de entornos SSO. Nacido fruto de un "fork" del producto OpenSSO. Ofrece autenticación y autorización de usuarios, servicios de federación, SSO, mecanismos de alta disponibilidad y API de desarrollo.
WSO2 Identity Server	WSO2	Free & Open Source (Apache 2.0)	WSO2 Identity Server, es capaz de conectar y gestionar múltiples identidades en aplicaciones, APIs, la nube, dispositivos móviles e IoT, independientemente de los estándares en los que se basan.

Ilustración 8. Productos SSO

CAS (Central Authentication Service)

El sistema escogido para la implantación de nuestro SSO es la solución de software libre **CAS (Central Authentication Service)**[32] en su versión **4.0.0**. CAS ha sido seleccionado por tener licencia Open Source, por ser una de las implementaciones de SSO más extendidas en todo el mundo, por su facilidad de configuración y por las múltiples posibilidades que ofrece. CAS dispone de una comunidad que contribuye y da soporte activamente al proyecto. Ofrece las siguientes características destacadas:

- Trabaja con un protocolo abierto y bien documentado.
- Está desarrollado con código Java disponible con licencia Open Source.
- Fácil integración con el framework Spring Security.
- Tiene soporte para la autenticación mediante LDAP, base de datos, certificados X.509, 2-factor).
- Soporte para múltiples protocolos (CAS, SAML, OAuth, OpenID).
- Una librería de clientes para Java, .Net, PHP, Perl, Apache, uPortal, etc.
- Se integra con uPortal, BlueSocket, TikiWiki, Mule, Liferay, Moodle, etc.
- Documentación y soporte bajo una amplia comunidad de desarrolladores.

Arquitectura

La arquitectura del sistema CAS está formada por el servidor CAS y los clientes CAS que se comunican mediante distintos protocolos.

El **servidor CAS**, es un Java servlet implementado con Spring Framework, cuya responsabilidad principal es autenticar y dar acceso a los clientes del CAS, mediante la emisión y validación de tickets. Una sesión de SSO se crea cuando el servidor emite un ticket (TGT) a un usuario al iniciar sesión correctamente. Un ticket de servicio (ST) se emite a un servicio a petición de un usuario mediante la redirección del navegador utilizando el TGT como token. El ST se valida posteriormente en el servidor CAS.

El **cliente CAS**, puede referirse a cualquier aplicación habilitada para comunicarse con el servidor CAS o a un paquete de software que se puede integrar con varias plataformas y aplicaciones para comunicarse con el servidor CAS.

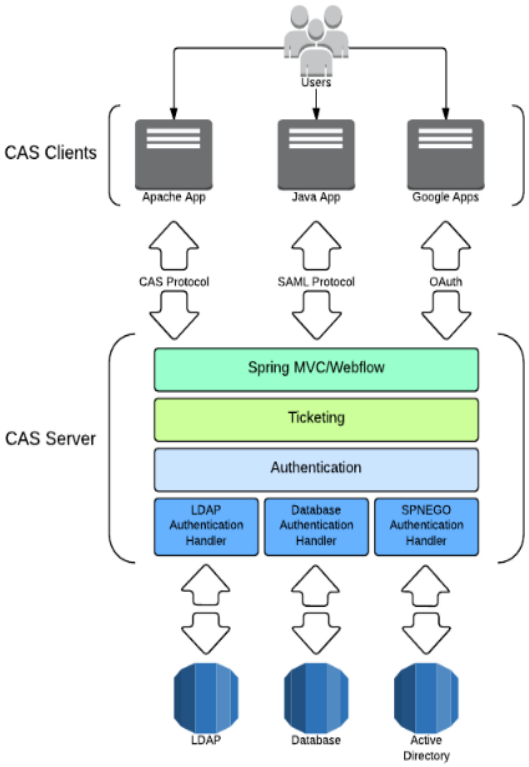

Ilustración 10. Arquitectura CAS

A continuación se muestra el diagrama de secuencia del protocolo CAS 3.0:

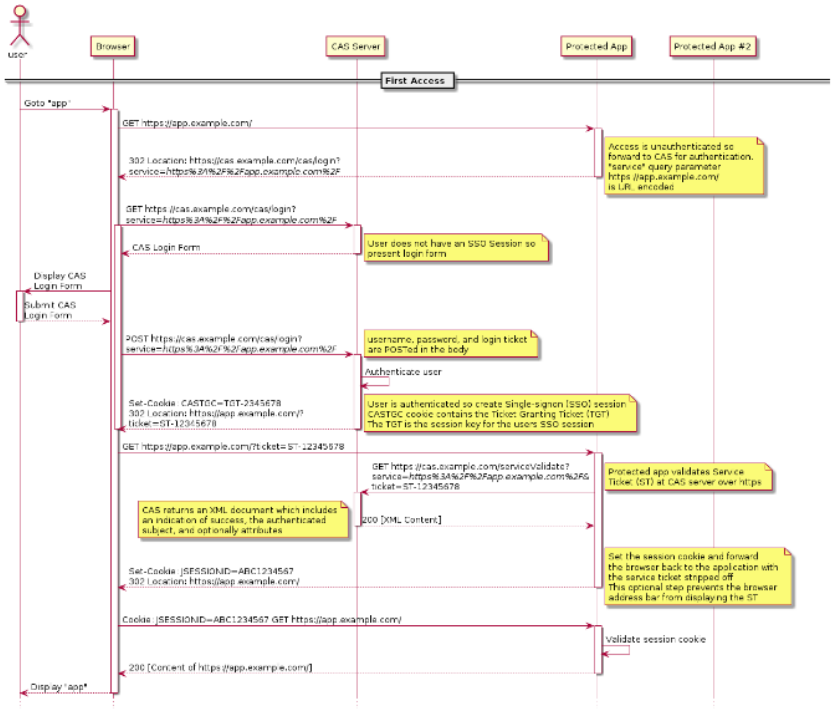

Ilustración 11. Protocolo CAS, primer acceso

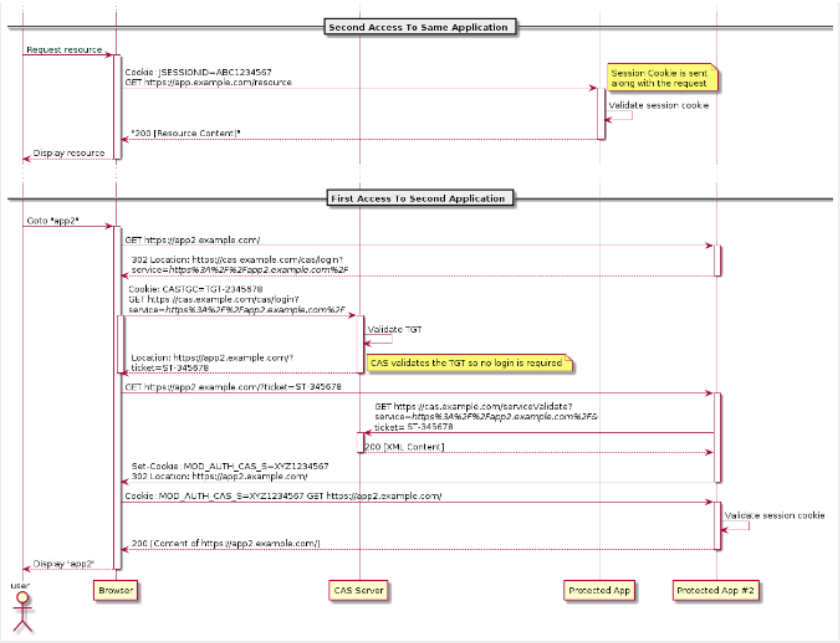

Ilustración 12. Protocolo CAS, segundo acceso

Seguridad

CAS es un software de seguridad que proporciona SSO seguro para aplicaciones basadas en web. El uso de CAS generalmente mejora el entorno de seguridad, pero hay varias consideraciones sobre la configuración, la política y la implementación de CAS que deben considerarse para lograr una seguridad adecuada.

Protocolo seguro (https): Toda comunicación con el servidor CAS debe tener lugar a través de un canal seguro, es decir, SSLv3, TLSv1. Hay dos justificaciones principales para este requisito:

- El proceso de autenticación requiere la transmisión de credenciales de seguridad.
- El ticket que genera el servidor CAS es un token de acceso al portador.

Dado que la divulgación de cualquiera de los datos permitiría ataques de suplantación, es de vital importancia asegurar el canal de comunicación entre los clientes CAS y el servidor CAS.

Conexiones a sistemas dependientes: CAS comúnmente requiere conexiones a otros sistemas, como directorios LDAP, bases de datos y servicios de almacenamiento en caché. En general, se recomienda utilizar transporte seguro (SSL / TLS, IPSec) a esos sistemas siempre que sea posible.

Componentes de autenticación

A continuación de describen los componentes de CAS que intervienen en el proceso de autenticación[33]:

Authentication Manager: Es el componente que se ocupa de comprobar que las credenciales de acceso introducidas por un cliente cumplen con las políticas de autenticación definidas. Se pueden definir las siguientes políticas de autenticación:

- *AnyAuthenticationPolicy*: Cualquiera de los handlers, que veremos a continuación, definido ha sido satisfecho.
- *AllAuthenticationPolicy*: Todos los handlers definidos, tienes que haber sido satisfechos.
- *RequiredHandlerAuthenticationPolicy*: Un determinado handler debe ser saltisfecho.

Authentication Handler: CAS tiene soporte para múltiples tipos de sistemas de autenticación. Los componentes de tipo Authentication Handler, son los encargados de implementar dicho soporte. A continuación se muestran los handlers disponibles:

- *Database Authentication*: Autenticación mediante base de datos. Se proporcionan tres handlers:
    - *QueryDatabaseAuthenticationHandler*: Autentica a un usuario comparando la contraseña introducida con la almacenada en un registro determinado por una consulta a base de datos.
    - *SearchModeSearchDatabaseAuthenticationHandler*: Busca un registro de usuario consultando con un nombre de usuario y una contraseña. El usuario se autentica si se encuentra al menos un resultado.
    - *BindModeSearchDatabaseAuthenticationHandler*: Autentica a un usuario intentando crear una conexión de base de datos usando el nombre de usuario y la contraseña.
- *JAAS Authentication*: JAAS es una API de autenticación y autorización estándar de Java. Se configura a través de un fichero de configuración de texto plano externo. El handler para su uso es *JaasAuthenticationHandler*.
- *LDAP Authentication*: Autenticación mediante un directorio LDAP como Active Directory u OpenLDAP. El handler para su uso es *LdapAuthenticationHandler*.
- *Legacy Authentication*: Proporciona soporte para adaptarse a diferentes necesidades de autenticación heredadas de versiones anteriores de CAS. El handler para su uso es *LegacyAuthHandler*.
- *OAuth/OpenID Authentication:* CAS dispone de soporte para trabajar con servidores OAuth/OpenID. Para su configuración, es necesario añadir el *OAuth20WrapperController*.
- *Radius Authentication:* El handler RADIUS, acepta credenciales de nombre de usuario y contraseña y delega la autenticación a uno o más servidores RADIUS. El handler para su uso es *RadiusAuthenticationHandler*.
- *SPNEGO Authentication*: SPNEGO es una tecnología de autenticación que se utiliza principalmente para proporcionar autenticación CAS transparente a los navegadores que se ejecutan en Windows con las credenciales de dominio de Active Directory. El handler para su uso es JCIFSSSpnego*AuthenticationHandler*.
- *Trusted Authentication*: Brinda soporte para la autenticación de confianza realizada por algún otro componente en la cadena de manejo de solicitudes HTTP. Los proxies (incluido Apache en un escenario de reverse proxy) son los componentes más comunes que realizan la autenticación frente a CAS. El handler para su uso es Trusted*AuthenticationHandler*.
- *X.509 Authentication*: Proporciona un mecanismo de autenticación para usuarios que presentan un certificado de cliente durante la comunicación SSL/TLS. El handler para su uso es X509Credentials*AuthenticationHandler*.

Principal resolution: Un *Principal* contiene un identificador único por el cual el usuario autenticado será conocido por todos los servicios solicitantes. También contiene atributos opcionales que pueden transmitirse a los servicios para admitir autorizaciones y personalizaciones. La resolución del *Principal* es una fase necesaria del proceso de autenticación que ocurre después de la autenticación de credenciales. Los componentes *PrincipalResolver* que se encuentran disponibles en CAS, son los siguientes:

- *PersonDirectoryPrincipalResolver*: Utiliza la librería Jasig Person Directory para proporcionar servicios flexibles a partir de un número indeterminado de fuentes de datos.

- *OpenIdPrincipalResolver*: Ectensión de *PersonDirectoryPrincipalResolver* para uso con credenciales de OpenID. Se configura del mismo modo que el *PersonDirectoryPrincipalResolver.*
- *SpnegoPrincipalResolver*: Extensión de *PersonDirectoryPrincipalResolver* que es específicamente para uso con credenciales de SPNEGO. La configuración es la misma que la de *PersonDirectoryPrincipalResolver*, pero con una propiedad adicional, *transformPrincipalId*, que proporciona una transformación de un caso simple en el Principal ID.
- *X509SubjectPrincipalResolver*: Crea un Crea una Principal ID a partir de una cadena de formato compuesta por componentes del nombre del sujeto.
- X509SubjectDN*PrincipalResolver*: Crea un Principal ID con el DN del sujeto del certificado.

Requisitos de la plataforma de SSO

Requisitos funcionales

RF001. Autenticación en sistema mediante usuario y contraseña.

Cuando un usuario intenta acceder a una aplicación de la plataforma por primera vez y no ha iniciado una sesión a nivel de SSO, se presentará una pantalla de login en la que deberá introducir usuario y contraseña. Una vez autenticado en el sistema, el usuario podrá acceder a las distintas aplicaciones sin necesidad de volver a introducir sus credenciales de acceso.

RF002. Autenticación en sistema mediante certificado digital.

Para acceder a las aplicaciones de administración de la plataforma, el usuario deberá tener instalado en su navegador un certificado digital firmado por una CA reconocida por el sistema. En este caso, no se mostrará la pantalla de login, sino que el usuario accederá directamente a la aplicación.

RF003. Implementación de aplicación web de gestión de usuarios.

Se desarrollará una aplicación web para la gestión de los usuarios de la plataforma. La aplicación mostrará un listado con los usuarios dados de alta en el sistema y sus roles asociados. Se podrán realizar las siguientes acciones desde la aplicación:

- Alta de nuevo usuario: Se mostrará un formulario donde se introducirán los datos del usuario y sus roles asociados.
- Baja de usuario: Se podrá eliminar cualquiera de los usuarios que muestra el sistema.
- Modificación de usuario: Se podrá modificar los datos y roles asociados de cualquier usuario.

RF004. Implementación de aplicación web de consulta de asignaturas MISTIC.

Se desarrollará una aplicación web que mostrará un listado con las asignaturas del MISTIC dadas de alta en la base de datos de la aplicación.

RF005. Implementación de aplicación web de consulta de libros en biblioteca.

Se desarrollará una aplicación web que mostrará un listado con los libros dados de alta en la base de datos de la aplicación.

RF006. Implementación de aplicación web de consulta de ofertas de empleo.

Se desarrollará una aplicación web que mostrará un listado con las ofertas de empleo dadas de alta en la base de datos de la aplicación.

RF007. Implementación de Punto de Acceso Único.

Se desarrollará una aplicación web que servirá de punto de acceso a todas las demás aplicaciones web de la plataforma.

RF008. Autorización de acceso a aplicaciones web.

Se definirán roles de acceso a las distintas aplicaciones. Una vez que el usuario se haya autenticado en el sistema por cualquiera de los dos métodos disponibles (usuario y contraseña o certificado digital), el sistema consultará que el usuario autenticado tiene los permisos necesarios para acceder a la aplicación. Se definen los siguientes roles:

- Rol administrador: Tendrá permiso de acceso a cualquier aplicación de la plataforma, en la cuales, podrá ejecutar cualquier acción.
- Rol usuario: Tendrá permiso de acceso a cualquier aplicación del sistema excepto a las aplicaciones de administración.
- Rol cliente aplicación consulta asignaturas MISTIC: Permiso de acceso a la aplicación de consulta de asignaturas del MISTIC.
- Rol cliente aplicación consulta de libros: Permiso de acceso a la aplicación de consulta de libros del sistema.
- Rol cliente aplicación consulta de ofertas de empleo: Permiso de acceso a la aplicación de consulta de ofertas de empleo.

RF009. Nombre de usuario autenticado.

En todas las aplicaciones que forman parte de la plataforma, se mostrará el identificador del usuario autenticado en la parte superior derecha de cada pantalla.

RF010. Logout de aplicaciones

En todas las aplicaciones que forman parte de la plataforma, el usuario podrá hacer logout mediante un botón situado en la parte superior derecha de cada pantalla.

Requisitos no funcionales

RNF001. Implementación de seguridad en las comunicaciones.

Las comunicaciones entre los distintos servidores y con el usuario final de las aplicaciones, se realizarán mediante protocolo seguro.

RNF002. Implementación de alta disponibilidad en servidores CAS.

Se configurará el servidor CAS en alta disponibilidad en dos servidores siguiendo un esquema activo-activo.

RNF003. Se implementará una red perimetral.

La arquitectura definida debe impedir el acceso desde una red externa a los servidores de aplicaciones donde se encuentran desplegadas las aplicaciones web. Para ello se configurará una red perimetral.

RNF004. Usuarios en servidor LDAP.
Los usuarios y sus contraseñas encriptadas, serán almacenados en un servidor LDAP.

RNF005. Roles en base de datos.
Los roles de los usuarios se almacenarán en base de datos.

Análisis funcional

Casos de uso

Actores
- Administrador: Actor que tendrá permisos sobre todas las aplicaciones de la plataforma.
- Usuario: Actor con permisos para acceder a las aplicaciones de usuario de la plataforma.
- CAS: Aplicación que implementa el Single Sign On.

Diagrama general de casos de uso
A continuación se muestra el diagrama general de casos de uso de las aplicaciones de la plataforma:

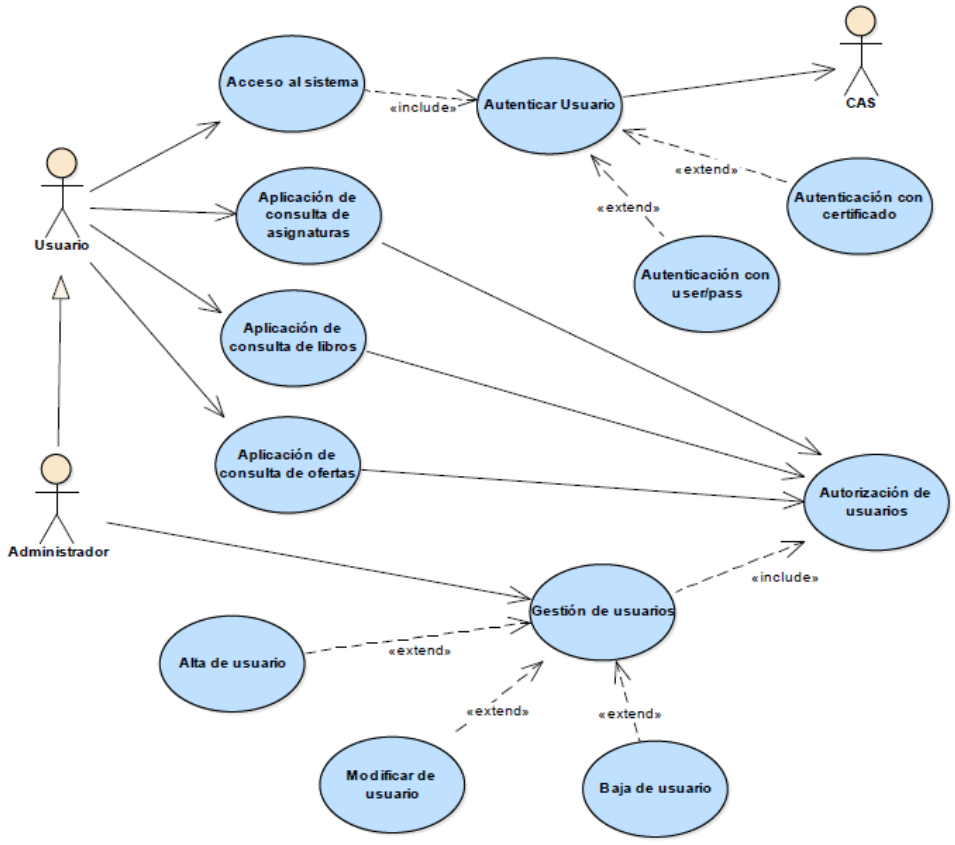

Ilustración 13. Diagrama general de casos de uso

Acceso al sistema con usuario y contraseña
Flujo Principal:

Paso 1: El usuario introduce la url de una de las aplicaciones del sistema

Paso 2: Se muestra al usuario la pantalla de login.

Paso 3: El usuario introduce su usuario y contraseña.

Paso 4: El sistema comprueba que el usuario y la contraseña son correctos.

Paso 5: El sistema comprueba que el usuario tiene permisos para acceder a la aplicación, es decir, que su rol es el adecuado.

Paso 6: El usuario accede a la aplicación.

Flujo alternativo: Usuario y contraseña incorrectos.

Paso 3.1: El sistema devuelve un mensaje de error y vuelve a la pantalla de login.

Flujo alternativo: Usuario sin permisos de acceso a la aplicación.

Paso 5.1: El sistema devuelve un mensaje de error y vuelve a la pantalla de login.

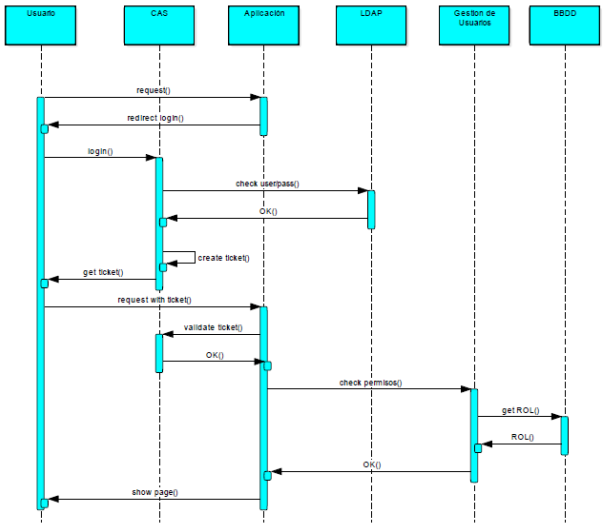

Ilustración 14. Diagrama de secuencia, acceso con usuario y password

Acceso al sistema con certificado digital

Flujo Principal:

Paso 1: El usuario introduce la url de una de las aplicaciones de administración de la plataforma.

Paso 2: El sistema valida el certificado digital instalado en el navegador del cliente.

Paso 3: El sistema comprueba que el usuario tiene permisos para acceder a la aplicación, es decir, que su rol es el adecuado.

Paso 4: El usuario accede a la aplicación.

Flujo alternativo: Certificado no válido

Paso 2.1: El sistema devuelve un mensaje de error de autenticación.

Flujo alternativo: Usuario sin permisos de acceso a la aplicación.

Paso 3.1: El sistema devuelve un mensaje de error de autorización.

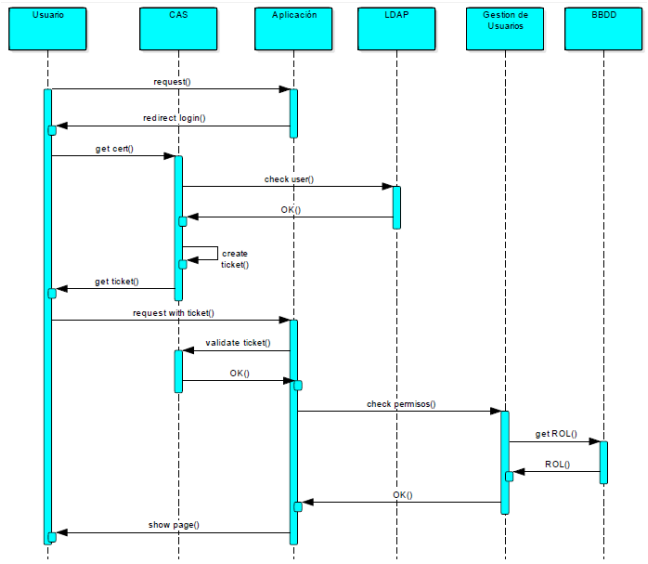

Ilustración 15. Diagrama de secuencia, acceso con certificado digital

Alta de usuario:

Flujo Principal:

Paso 1: El usuario introduce la url de la aplicación de gestión de usuarios.

Paso 2: El sistema valida el certificado digital instalado en el navegador del cliente (caso de uso 2.1.4).

Paso 3: El sistema comprueba que el usuario tiene permisos para acceder a la aplicación, es decir, que su rol es el adecuado.

Paso 4: El usuario accede a la aplicación.

Paso 5: El usuario selecciona la opción alta de usuario.

Paso 6: El sistema muestra un formulario de alta para rellenar.

Paso 7: El usuario introduce los datos y acepta.

Paso 8: El usuario se almacena en el sistema.

Paso 9: El sistema muestra el listado de usuarios.

Flujo alternativo: Certificado no válido

Paso 2.1: El sistema devuelve un mensaje de error de autenticación.

Flujo alternativo: Usuario sin permisos de acceso a la aplicación.

Paso 3.1: El sistema devuelve un mensaje de error de autorización.

Flujo alternativo: Error de validación de los datos.

Paso 7.1: El sistema devuelve un error de validación.

Modificación de usuario

Flujo Principal:

Paso 1: El usuario introduce la url de la aplicación de gestión de usuarios.

Paso 2: El sistema valida el certificado digital instalado en el navegador del cliente (caso de uso 2.1.4).

Paso 3: El sistema comprueba que el usuario tiene permisos para acceder a la aplicación, es decir, que su rol es el adecuado.

Paso 4: El usuario accede a la aplicación.

Paso 5: El usuario selecciona un usuario y selecciona la opción modificar.

Paso 6: El sistema muestra un formulario con los datos del usuario editables.

Paso 7: El usuario modifica los datos y acepta.

Paso 8: El usuario modificado se almacena en el sistema.

Paso 9: El sistema muestra el listado de usuarios.

Flujo alternativo: Certificado no válido

Paso 2.1: El sistema devuelve un mensaje de error de autenticación.

Flujo alternativo: Usuario sin permisos de acceso a la aplicación.

Paso 3.1: El sistema devuelve un mensaje de error de autorización.

Flujo alternativo: Error de validación de los datos.

Paso 7.1: El sistema devuelve un error de validación

Baja de usuario

Flujo Principal:

Paso 1: El usuario introduce la url de la aplicación de gestión de usuarios.

Paso 2: El sistema valida el certificado digital instalado en el navegador del cliente (caso de uso 2.1.4)..

Paso 3: El sistema comprueba que el usuario tiene permisos para acceder a la aplicación, es decir, que su rol es el adecuado.

Paso 4: El usuario accede a la aplicación.

Paso 5: El usuario selecciona un usuario y selecciona la opción borrar.

Paso 6: El sistema elimina el usuario.

Paso 7: El sistema muestra el listado de usuarios.

Flujo alternativo: Certificado no válido

Paso 2.1: El sistema devuelve un mensaje de error de autenticación.

Flujo alternativo: Usuario sin permisos de acceso a la aplicación.

Paso 3.1: El sistema devuelve un mensaje de error de autorización.

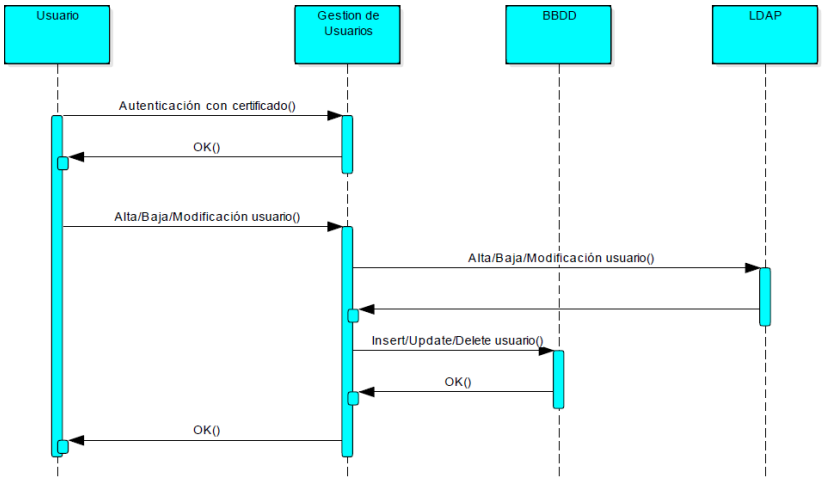

Ilustración 16. Diagrama de secuencia, alta, baja y modificación de usuario

Diagrama de clases de las aplicaciones web

Para todas las aplicaciones web, las cuales están desarrolladas siguiente el patrón modelo vista controlador (MVC), se define la siguiente estructura de paquetes:

- Auth → Clases necesarias para autenticarnos en la plataforma haciendo uso del web service de gestión de usuarios. Incluye cliente de ws generado con Apache CXF.
- Controller → Controlador del flujo de la aplicación.
- Dao → Acceso a datos.
- Dto → Objetos de transferencias de datos.
- Model → Modelo de datos de la aplicación.
- Service → Servicios de la aplicación.
- WebService (sólo en la aplicación Gestión de usuarios) → Implementación de servicios web mediante Apache CXF (WS de gestión de usuarios).
- Webapp → Pantallas (vista) de la aplicación web.

Todas las aplicaciones hacen uso de los Framework Spring, Spring Security y Apache CXF. A continuación se muestra el diseño que siguen todas las aplicaciones desarrolladas:

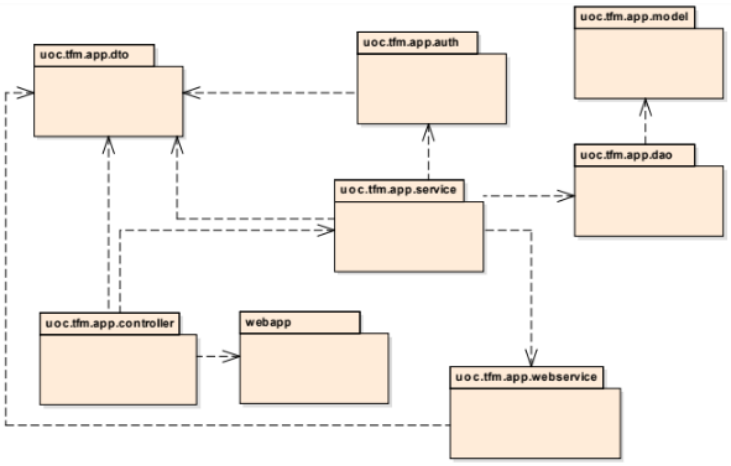

Ilustración 17. Diagrama de clases de aplicaciones web

Todas las aplicaciones de la plataforma tendrán una relación de uso contra la aplicación Gestión de usuarios, al contener ésta el servicio web de autenticación de usuarios.

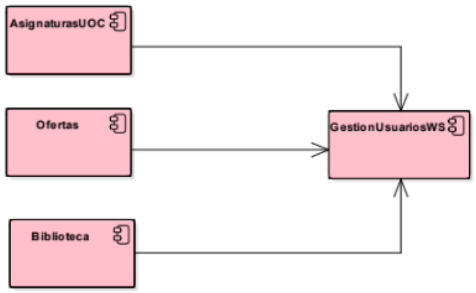

Ilustración 18. Diagrama de componentes de aplicaciones web

Modelo de Entidad-Relación de las aplicaciones web

Aplicación Gestión de usuarios

Se definen tres entidades:

Entidad USUARIOS con relación *n:m* sobre la entidad ROLES

Entidad ROLES con relación *n:m* sobre la entidad USUARIOS y con relación n:m sobre la entidad APLICACIÓN.

Entidad APLICACIONES con relación *n:m* sobre la entidad ROLES.

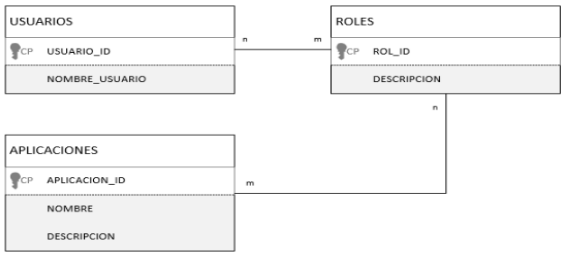

Ilustración 19. Diagrama E-R de Gestión de usuarios

Con este modelo de datos, podremos asociar varios roles a un mismo usuario y cada rol podrá estar asociado a varias aplicaciones. A su vez, los usuarios podrán tener varios roles y las aplicaciones podrán ser accedidas por varias roles.

Aplicación Asignaturas UOC

Se define una única entidad ASIGNATURAS, que contendrá los datos de las asignaturas que se mostrarán en la aplicación:

Ilustración 20. Diagrama E-R de Asignaturas UOC

Aplicación Ofertas de empleo

Se define una única entidad OFERTAS, que contendrá los datos de las ofertas de empleo que se mostrarán en la aplicación:

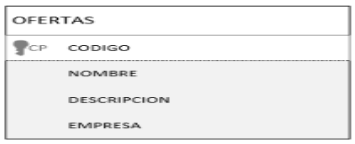

Ilustración 21. Diagrama E-R de Ofertas de empleo

Aplicación Biblioteca

Se define una única entidad LIBROS, que contendrá los datos de los libros que se mostrarán en la aplicación:

Ilustración 22. Diagarma E-R de Biblioteca

Interfaz gráfico de aplicaciones web

Aplicación PAU (Punto de Acceso Único)

Desde la aplicación PAU, podremos acceder al resto de aplicaciones de la plataforma, aportando las credenciales necesarias. Se mostrarán dos secciones: Aplicaciones de administración, para las que serán necesario disponer de certificado digital para acceder y Aplicaciones de usuario, cuyo acceso será únicamente aportando un usuario y una contraseña válidos:

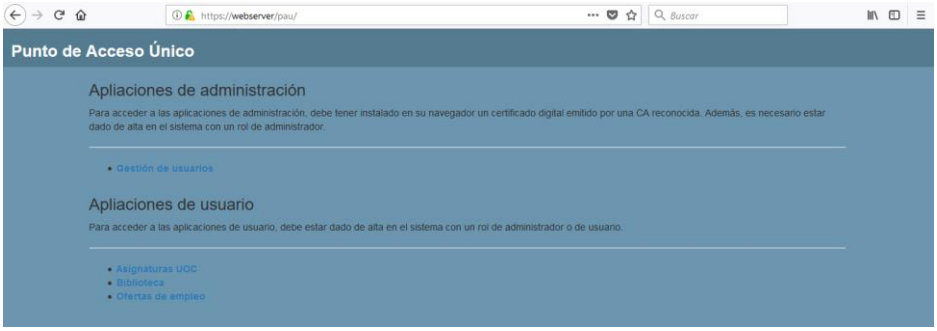

Ilustración 23. Aplicación PAU

Aplicación Gestión de usuarios

La pantalla principal de la aplicación Gestión de usuarios tendrá el siguiente aspecto:

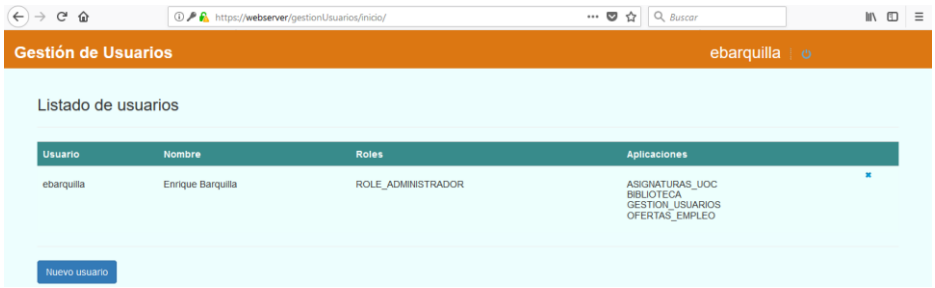

Ilustración 24. Aplicación Gestión de usuarios

Al pulsar doble click sobre cualquiera de la usuarios que muestra el listado, se mostrará el siguiente formulario, desde el cual podremos modificar los datos del usuario y sus roles asociados.

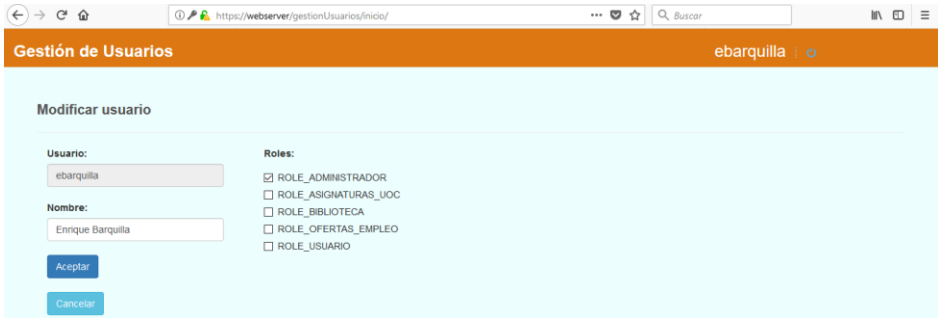

Ilustración 25. Crear/modificar usuario en Gestionar Usuarios

Para eliminar un usuario, se deberá pulsar el "aspa" ✖ que aparece a la derecha de cada usuario en el listado. La aplicación mostrará un mensaje de confirmación.

El nombre del usuario autenticado se mostrará en la parte superior derecha de la pantalla, así como un botón que permita al usuario hacer *logout*.

Aplicaciones de usuario

Las aplicaciones *Asignaturas UOC, Ofertas de empleo* y *Biblioteca*, tendrán un aspecto similar, consistente en un interfaz web simple en el que se mostrará un listado con los registros recuperados del esquema de base de datos correspondiente.

Al igual que en la aplicación Gestión de usuarios, el nombre del usuario autenticado se mostrará en la parte superior derecha de la pantalla, así como un botón que permita al usuario hacer *logout*.

Aplicación Asignaturas MISTIC

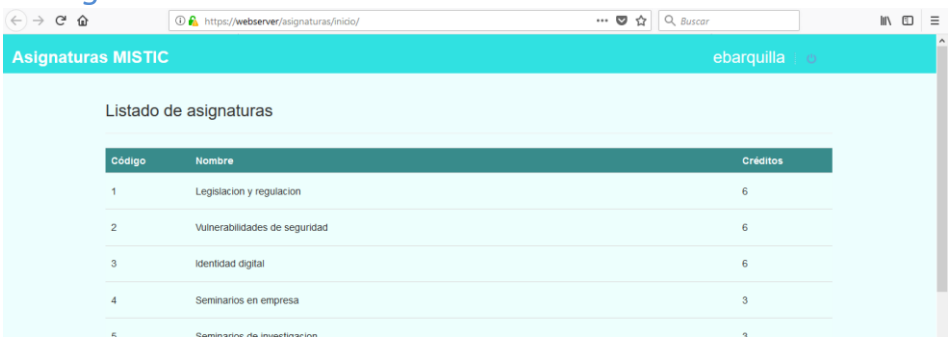

Ilustración 26. Aplicación Asigntauras MISTIC

Aplicación Ofertas de empleo

Ilustración 27. Aplicación Ofertas de empleo

Aplicación Biblioteca

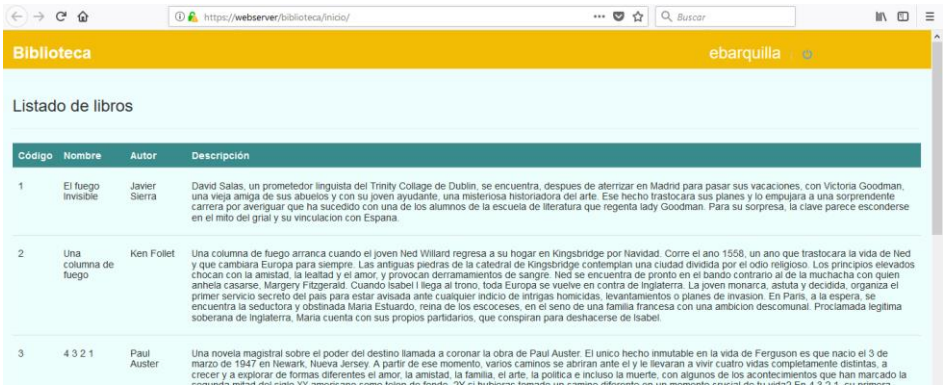

Ilustración 28. Aplicación Biblioteca

Estructura del directorio LDAP de la plataforma

Se define el dominio tfm.int y una única unidad organizativa *users* de la que "colgarán" los usuarios de la plataforma.

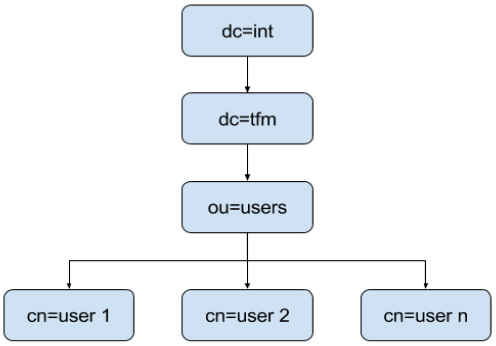

Ilustración 29. Esquema del LDAP de la plataforma SSO

Arquitectura de la plataforma SSO

A continuación se muestra el diseño de la arquitectura de nuestra plataforma SS0. Está compuesta por los siguientes elementos:

Router: Se encarga de encaminar paquetes entre nuestra red y otras externas.

Firewall: Es el encargado de permitir o denegar los paquetes que viajan por la red LAN de la plataforma SSO. Será necesario configurarlo para que permita conexiones sólo a los puertos 80 y 443 desde el exterior y que tan sólo permita conexiones a la red interna desde la DMZ, en concreto a los puertos:

- 8080 (http Tomcat y Wildfly). Máquinas casserver1, casserver2 y appsserver.
- 8443 (https Tomcat y Wildfly). Máquinas casserver1, casserver2 y appsserver.
- 8009 (ajp Tomcat). Máquinas casserver1 y casserver2.
- 386 (ldap). Máquina dataserver.
- 636 (ldaps). Máquinas dataserver.
- 3306 (mysql). Máquina dataserver.

Servidor web (webserver): Se encontrará configurado en la red DMZ y a través de Proxies dará acceso a las máquinas de la red interna. Contendrá el software HTTP Apache.

Servidores CAS (casserver1 y casserver2): Se encontrarán desplegados en alta disponibilidad en la red interna. Contendrá el software Tomcat, donde se desplegarán los servidores CAS.

Servidor de aplicaciones (appsserver): Se encuentra en la red interna y contiene todas las aplicaciones web de la plataforma, tantos las administrativas como las de usuario. El software instalado es el servidor de aplicaciones WildFly.

Servidor de datos (dataserver): Se encuentra en la red interna y en él se instalan el servidor de base de datos (mysql) y el servidor ldap (OpenLDAP).

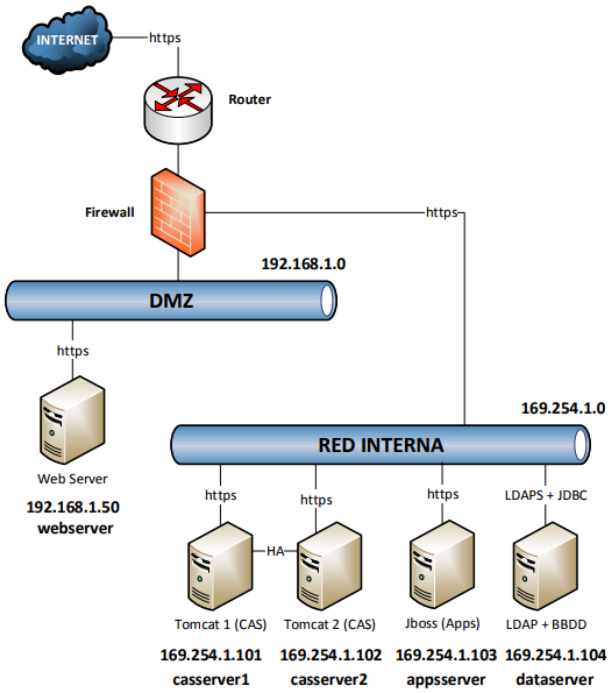

Ilustración 30. Arquitectura plataforma SSO

Instalación y configuración de la plataforma SSO

A continuación se va a mostrar el proceso que se ha seguido para conseguir la implantación de la plataforma SSO. Los detalles de las instalaciones se pueden encontrar en los anexos de este documento.

Infraestructura

En primer lugar, ha sido necesario crear la infraestructura de máquinas y redes de la plataforma. Para ello, se ha utilizado el software de virtualización Oracle Virtual Box 5.2.0. Las máquinas virtuales creadas han sido las siguientes:

- Webserver: Memoria 1 GB, disco 20 GB, 2 procesadores i5-3210M 2,5Ghz.
- Casserver1: Memoria 1 GB, disco 20 GB, 2 procesadores i5-3210M 2,5Ghz.
- Casserver2: Memoria 1 GB, disco 20 GB, 2 procesadores i5-3210M 2,5Ghz.
- Appsserver: Memoria 1 GB, disco 20 GB, 2 procesadores i5-3210M 2,5Ghz.
- Dataserver: Memoria 2 GB, disco 20 GB, 2 procesadores i5-3210M 2,5Ghz.

Para ser estrictos con la arquitectura definida, hubiera sido necesario crear una máquina virtual adicional con el software necesario para hacer de Firewall, sin embargo, en el entorno de laboratorio creado para probar nuestra plataforma, no se ha creado por falta de tiempo y de recursos hardware. Debido a esto, el diseño de la red se ha modificado creando dos interfaces de red en la máquina Webserver, uno para comunicarse con el exterior y otro con la red interna. Otra variación respecto a lo definido inicialmente es que la comunicación con el servidor de aplicaciones *appsserver* y con el servidor ldap dataserver, se va a realizar con protocolo no seguro (*http y ldap*). De esta forma, la topología de red del entorno de laboratorio, es la siguiente:

- Red DMZ: 192.168.1.0
- Red Interna: 169.254.1.0

- Webserver:
  - Adaptador 1 (Adaptador puente): 192.168.1.50
  - Adaptador 2 (Red interna): 169.254.1.100

- Casserver1:
  - Adaptador 1 (Red interna): 192.168.1.101

- Casserver2:
  - Adaptador 1 (Red interna): 192.168.1.102

- Appsserver:
  - Adaptador 1 (Red interna): 192.168.1.103

- Dataserver:
  - Adaptador 1 (Red interna): 192.168.1.104

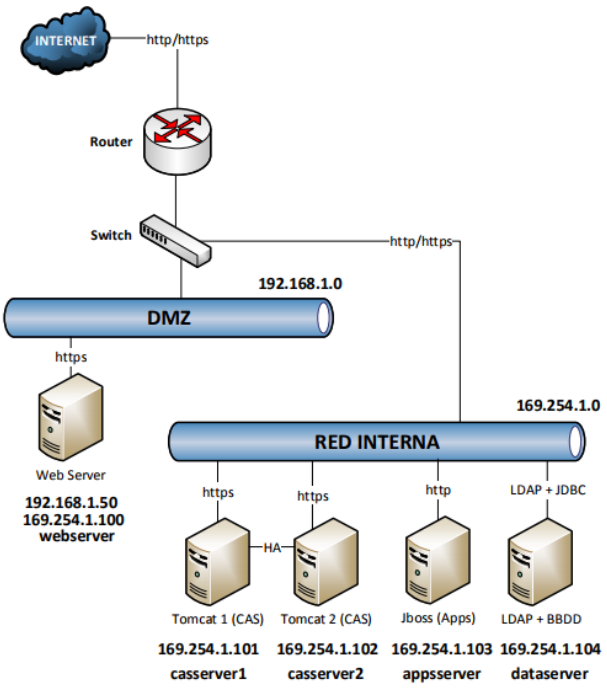

Ilustración 31. Arquitectura SSO en entorno de laboratorio

El sistema operativo instalado en todas las máquinas virtuales ha sido Debian Stretch 9.2.1.

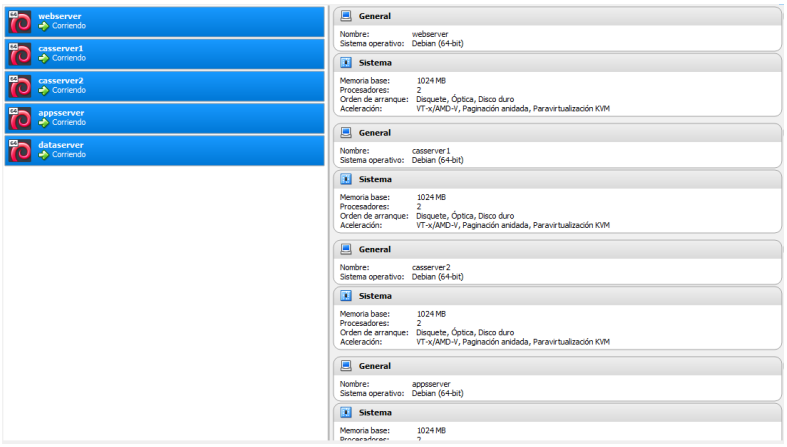

Ilustración 32. Máquinas virtuales instaladas

Para poder acceder a la máquina webserver desde Internet, tendremos que hacer uso de NAT, configurando nuestro router que nos da salida a Internet de forma que redirija todas las peticiones al puerto 443, al servidor webserver (192.168.1.50).

Cabe destacar que al no disponer de servidor DNS interno, se han creado las correspondientes entradas de todas las máquinas de la plataforma, en los ficheros */etc/hosts*. A continuación se muestra a modo de ejemplo, el contenido del fichero */etc/hosts* de la máquina *webserver*:

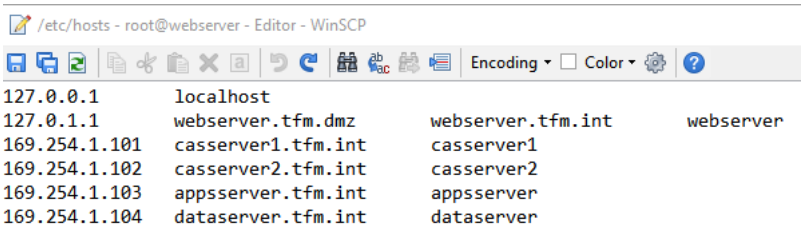

Ilustración 33. Fichero /etc/hosts de webserver

Configuración de servidores

Configuración de HTTP Apache

La versión instalada del servidor web Apache es la 2.4.25, incluida en el sistema operativo Debian Stretch 9.2.1. La configuración del servidor web Apache, se ha realizado en primer lugar creando dos Virtual Host que escuchan en los puertos 80 y 443. En el puerto 80, añadimos la directiva *Redirect / https://webserver* con lo que conseguiremos que todo el tráfico que entre por el puerto 80, sea redirigido al puerto seguro 443:

```
<VirtualHost *:80>
    ServerName webserver
    DocumentRoot /var/www/html

    Redirect / https://webserver/
</VirtualHost>
```

A continuación se muestra la configuración del Virtual Host del puerto 443:

```
<VirtualHost *:443>

  ServerName webserver

  SSLProxyEngine on

  <Location "/gestionUsuarios">
    SSLVerifyClient require
  </Location>

  SSLCertificateFile /etc/apache2/ssl/webserver.pem
  SSLCertificateKeyFile /etc/apache2/ssl/webserverkey.pem
  SSLCACertificateFile /etc/apache2/ssl/ca.pem
```

En primer lugar se muestran las directivas necesarias para poder activar el motor SSL en el puerto 443. Previamente, se han generado el certificado de servidor *webserver.pem* mediante la aplicación OpenSSL como se indicó en el apartado *Generación de certificados X.509* de este documento. Cabe destacar la directiva *<Location>*, por la cual se consigue que todo acceso a */gestionUsuarios* deba realizarse presentando un certificado digital de cliente válido (directiva *SSLVerifyClient*).

A continuación se muestran la configuración de los *ReverseProxy* que permiten comunicarse con las máquinas de la red interna y del *balanceador de carga*, con el que implementamos la alta disponibilidad de los servidores CAS:

```
ProxyPass /dummyCasApp http://appsserver:8080/dummyCasApp
ProxyPassReverse /dummyCasApp http://appsserver:8080/dummyCasApp

ProxyPass /gestionUsuarios http://appsserver:8080/gestionUsuarios
ProxyPassReverse /gestionUsuarios http://appsserver:8080/gestionUsuarios

ProxyPass /asignaturas http://appsserver:8080/asignaturas
ProxyPassReverse /asignaturas http://appserver:8080/asignaturas

ProxyPass /biblioteca http://appsserver:8080/biblioteca
ProxyPassReverse /biblioteca http://appsserver:8080/biblioteca

ProxyPass /ofertas http://appsserver:8080/ofertas
ProxyPassReverse /ofertas http://appsserver:8080/ofertas

<Proxy "balancer://cascluster">
  BalancerMember "https://casserver1:8443/cas-server-webapp-4.0.0" route=casserver1
  BalancerMember "https://casserver2:8443/cas-server-webapp-4.0.0" route=casserver2
</Proxy>

ProxyPass /cas-server-webapp-4.0.0 balancer://cascluster stickysession=JSESSIONID
ProxyPassReverse /cas-server-webapp-4.0.0 balancer://cascluster stickysession=JSESSIONID

ProxyPass /gestionUsuarios http://appsserver:8080/gestionUsuarios
ProxyPassReverse /gestionUsuarios http://appsserver:8080/gestionUsuarios

ProxyPass /pau http://appsserver:8080/pau
ProxyPassReverse /pau http://appsserver:8080/pau
```

Comprobamos que hacemos uso de *Stickness* en el balanceador haciendo uso de la cookie de sesión *JSESSIONID*.

Instalación y configuración del servidor CAS

En primer lugar, vamos a proceder a habilitar el puerto 8443 del servidor Tomcat. Para ello, generaremos un certificado de servidor mediante la herramientas OpenSSL y a continuación, lo importaremos en el almacén de claves (keytool) de Tomcat, tal y como se indicó en el apartado *Generación de certificados X.509* de este documento.

La versión del servidor CAS que vamos a instalar en los dos servidores Tomcat de las máquinas casserver1 y casserver2 es la 4.0.0. Una vez desplegada la aplicación web que encontramos en el directorio *modules* del CAS (*cas-server-webapp-4.0.0.war*), vamos a configurar el Authentication Handler *LDAPAuthenticationHandler,* con el que trabajaremos y que nos permitirá la integración de CAS con el servidor LDAP. Necesitaremos incluir en el classpath de la aplicación web del CAS, las librerías *cas-server-support-ldap-4.0.0.jar*, slf4j-*api-1.7.12.jar* y *ldaptive-1.1.0.jar*

```
<bean id="ldapAuthenticationHandler"
      class="org.jasig.cas.authentication.LdapAuthenticationHandler"
      p:principalIdAttribute="uid"
      c:authenticator-ref="authenticator">
</bean>

<bean id="authenticator" class="org.ldaptive.auth.Authenticator"
      c:resolver-ref="dnResolver"
      c:handler-ref="authHandler" />

<!--
   | The following DN format works for many directories, but may need to be
   | customized.
   -->
<bean id="dnResolver"
      class="org.ldaptive.auth.FormatDnResolver"
      c:format="uid=%s,${ldap.authn.baseDn}" />
```

Modificamos el fichero *deployerConfigContext.xml* del CAS, incluyendo el *LDAPAuthenticationHandler* y su configuración necesaria. Entre otros parámetros de configuración, tendremos que indicar la url del LDAP *ldap://dataserver*, el DN base del ldap donde se buscará al usuario *ou=users,dc=tfm,dc=int* y el flitro para buscarlo *(uid={user})*. Si se hubiera securizado la plataforma, también tendríamos que haber idicado la ruta del certificado de servidor en el *bean <bean id="sslConfig" class="org.ldaptive.ssl.SslConfig">*

```
<bean id="connectionConfig" class="org.ldaptive.ConnectionConfig"
      p:ldapUrl="${ldap.url}"
      p:connectTimeout="${ldap.connectTimeout}"
      p:useStartTLS="${ldap.useStartTLS}"
      p:sslConfig-ref="sslConfig" />

<bean id="sslConfig" class="org.ldaptive.ssl.SslConfig">
    <property name="credentialConfig">
        <bean class="org.ldaptive.ssl.X509CredentialConfig"
              p:trustCertificates="${ldap.trustedCert}" />
    </property>
</bean>
```

Para trabajar con certificados digitales X.509, CAS dispone del Authentication Handler *X509CredentialsAuthenticationHandler*. Para nuestro entorno de laboratorio, no se ha utilizado.

Para que CAS haga uso de todos los authentication handler definidos, tendremos que añadirlos al *Authetication Manager* junto con la política de autenticación que se seguirá, en este caso *AnyAuthenticationPolicy,* del siguiente modo:

```
<bean id="authenticationManager" class="org.jasig.cas.authentication.PolicyBasedAuthenticationManager">
    <constructor-arg>
        <map>
            <entry key-ref="ldapAuthenticationHandler" value-ref="primaryPrincipalResolver" />
        </map>
    </constructor-arg>
    <property name="authenticationPolicy">
        <bean class="org.jasig.cas.authentication.AnyAuthenticationPolicy" />
    </property>
</bean>
```

Para designar los valores correspondientes a los atributos de los distintos beans definidos en el fichero *deployerConfigContext.xml*, utilizaremos el fichero *cas.properties*.

También modificaremos el fichero *cas-servlet.xml* del CAS, para permitir la redirección a una página predeterminada (en nuestro caso la página de login de CAS), cuando el usuario haga *logout*. Para ello, modificamos la propiedad *followServiceRedirects* del bean *logoutAction*, asignándole el valor *yes*:

```
<bean id="logoutAction" class="org.jasig.cas.web.flow.LogoutAction"
      p:servicesManager-ref="servicesManager"
      p:followServiceRedirects="${cas.logout.followServiceRedirects:yes}"/>
```

Configuración de aplicaciones web y despliegue en WildFly

Spring Security: Para poder integrar las aplicaciones web con el servidor CAS, se ha utilizado el framework Spring Security 3.1.3, que incluye soporte para la integración con CAS mediante las librerías *spring-security-cas* y *spring-security-cas-client*.

En los ficheros de contexto de spring, vamos a definir los beans necesarios para realizar autenticación y autorización integradas con el servidor CAS. El el fichero *application-context-security* de cada aplicación, definiremos los siguientes beans:

En primer lugar, establecemos el *EntryPoint*, con el que estableceremos los filtros necesarios para que cuando un usuario quiera hacer login en una aplicación, lo haga a través del servidor CAS. También definiremos los roles de acceso que deberá tener el usuario para poder acceder a la aplicación: Por ejemplo, a continuación se muestra el *EntryPoint* definido para la aplicación de Gestión de Usuarios, en el cual, también se define el acceso webservice de autenticación, accesible desde la url *https://webserver/gestionUsuarios/services/gestionusuariosservice?wsdl*:

```
<security:http entry-point-ref="casAuthenticationEntryPoint" auto-config="false" use-expressions="true">
    <security:intercept-url pattern="/services/**" access="permitAll"></security:intercept-url>
    <security:intercept-url pattern="/**" access="hasRole('ROLE_ADMINISTRADOR')"></security:intercept-url>
    <security:custom-filter position="CAS_FILTER" ref="casAuthenticationFilter"></security:custom-filter>
    <security:custom-filter ref="requestSingleLogoutFilter" before="LOGOUT_FILTER" />
    <security:custom-filter ref="singleLogoutFilter" before="CAS_FILTER"/>
</security:http>
```

En los *EntryPoints* de las aplicaciones, se definirán los roles necesarios para acceder a ellas, por ejemplo, a continuación vemos el *EntryPoint* de la aplicación de Asignaturas UOC, con roles de acceso *ROLE_ADMINISTRADOR, ROLE_USUARIO* y *ROLE_ASIGNATURAS_UOC*:

```
<security:http entry-point-ref="casAuthenticationEntryPoint" auto-config="false">
    <security:intercept-url pattern="/**" access="ROLE_ADMINISTRADOR,ROLE_USUARIO,ROLE_ASIGNATURAS_UOC"></security:intercept-url>
    <security:custom-filter position="CAS_FILTER" ref="casAuthenticationFilter"></security:custom-filter>
    <security:custom-filter ref="requestSingleLogoutFilter" before="LOGOUT_FILTER" />
    <security:custom-filter ref="singleLogoutFilter" before="CAS_FILTER"/>
</security:http>
```

A continuación, se definen los beans necesarios para hacer *logout* en las aplicaciones*:*

```
  <!-- This filter handles a Single Logout Request from the CAS Server -->
<bean id="singleLogoutFilter" class="org.jasig.cas.client.session.SingleSignOutFilter"/>
<!-- This filter redirects to the CAS Server to signal Single Logout should be performed -->
<bean id="requestSingleLogoutFilter"
      class="org.springframework.security.web.authentication.logout.LogoutFilter">
  <constructor-arg value="https://webserver/cas-server-webapp-4.0.0/logout?service=https://webserver/gestionUsuarios"/>
  <constructor-arg>
    <bean class=
        "org.springframework.security.web.authentication.logout.SecurityContextLogoutHandler"/>
  </constructor-arg>
  <property name="filterProcessesUrl" value="/j_spring_security_logout"/>
</bean>
```

Y para hacer *login*:

```
<bean id="casAuthenticationEntryPoint" class="org.springframework.security.cas.web.CasAuthenticationEntryPoint">
    <property name="loginUrl" value="https://webserver/cas-server-webapp-4.0.0/login"></property>
    <property name="serviceProperties" ref="serviceProperties"></property>
</bean>
```

Para seleccionar el modo en el que los usuarios se autenticarán en las aplicaciones, tendremos que definir el bean *userService*. En el caso de la aplicación de Gestión de usuarios, utilizaremos la clase de *Spring-security* `org.springframework.security.core.userdetails.jdbc.JdbcDaoImpl`. Con esta clase recuperamos los roles de base de datos. Para su configuración, necesitamos indicar el *dataSource* o cadena de conexión a la base de datos de Gestión de usuarios, la cual se define mediante un bean en el fichero de configuración de Spring, y las queries necesarias para acceder a la información del usuario en las propiedades *usersByUsernameQuery* y *authoritiesByUsernameQuery:*

```
<bean id="userService" class="org.springframework.security.core.userdetails.jdbc.JdbcDaoImpl">
    <property name="dataSource" ref="dataSource" />
    <property name="usersByUsernameQuery">
        <value>select usuario_id, 'none', 1 enabled from usuarios where upper(usuario_id)=UPPER(?)</value>
    </property>
    <property name="authoritiesByUsernameQuery">
        <value>select usuario_id, rol_fk from usuarios inner join usuarios_roles on usuario_fk = usuario_id where upper(usuario_id) = upper(?)</value>
    </property>
</bean>
```

En las aplicaciones de usuarios de la plataforma (Asignaturas UOC, Ofertas de empleo y Biblioteca), la autenticación se realizará mediante un servicio web presente en la aplicación de Gestión de usuarios. Para poder acceder al servicio, en cada aplicación se crear una clase de autenticación que implementa la clase de Spring-Security `org.springframework.security.core.userdetails.UserDetailsService` y que, a través de un cliente del web service *gestionusuariosservice*, podrá recuperar los roles asociados al usuario:

```
<bean id="userService" class="uoc.tfm.asignaturas.auth.UserDetailsServiceImpl">
</bean>
```

El código Java de la clase *UserDetailsServiceImpl* es el siguiente:

```
public class UserDetailsServiceImpl implements UserDetailsService {

    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        List<GrantedAuthority> lstAuth = new ArrayList<GrantedAuthority>();
        URL url=null;
        try {
            url = new URL("https://webserver/gestionUsuarios/services/gestionusuariosservice?wsdl");
        } catch (MalformedURLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        GestionUsuariosServiceImplService gestionUsuariosServiceImplService = new GestionUsuariosServiceImplService(url);
        List<RolDTO> roles = gestionUsuariosServiceImplService.getGestionUsuariosServiceImplPort().getRolesUsuario(username);

        for (RolDTO rol : roles){
            GrantedAuthority auth = new SimpleGrantedAuthority(rol.getNombre());
            lstAuth.add(auth);
        }
        return new UserDetailImpl(username, null, true, true, true, true, lstAuth);

    }

}
```

Finalmente, definimos el bean *casAuthenticationProvider*, en el que referenciaremos el bean *UserService* correspondiente y la url del servicio de validación de tickets:

```
<bean id="casAuthenticationProvider" class="org.springframework.security.cas.authentication.CasAuthenticationProvider">
    <property name="userDetailsService" ref="userService"></property>
    <property name="serviceProperties" ref="serviceProperties"></property>
    <property name="ticketValidator">
        <bean class="org.jasig.cas.client.validation.Cas20ServiceTicketValidator">
            <constructor-arg index="0" value="https://webserver/cas-server-webapp-4.0.0"></constructor-arg>
        </bean>
    </property>
    <property name="key" value="an_id_for_this_auth_provider_only"></property>
</bean>
```

Apache CXF: Para la implementación del servicio web *gestionusuariosservice,* se ha utilizado el framework Aapche CXF 2.2.12 y la herramienta que proporciona dicho framework *java2wsdl*, mediante la cual podemos generar un servicio web y su definición *wsdl*, a partir de una clase Java. Para la generación de los clientes que consumen el servicio, se ha utilizado el mismo framework y la herramienta también disponible en CXF, *wsdl2Java*, mediante la cual, partiendo de un *wsdl* dado, podemos generar las clases necesarias para comunicarlos con web service que define.

Despliegue en WildFly: Todas las aplicaciones web se compilan y empaquetan como ficheros war (web application archive) mediante Apache Maven 3.3.9. Se despliegan en modo *Standalone* en el servidor *WildFly* de la máquina *appsserver*. Se utiliza el conector http (8080), por lo que, en este caso, no se securiza la comunicación.

El servicio web de Gestion de Usuario que permitirá la autenticación desde las aplicaciones de usuario, quedará accesible en la url *https://webserver/gestionUsuarios/service/gestionusuariosservice?wsdl*:

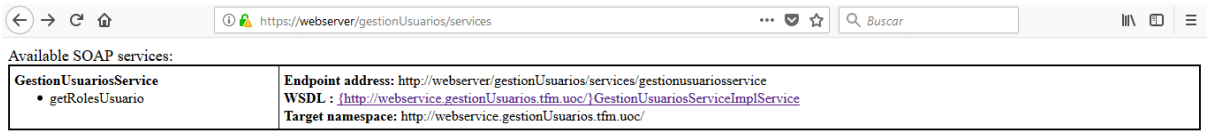

Ilustración 34. Servicio web gestionusuariosws

Configuración de MySQL

Se ha utilizado la versión 10.1.26 de MariaDB. En primer lugar, para poder acceder al servidor de base de datos desde otra máquina, es necesario modificar el fichero de configuración *50-server.cnf*. En él, existen las dos propiedades siguientes: *skip-external-locking*, la cual comentaremos para que nos permita hacer login en las bases de datos creadas desde máquinas externas y *bind-address*, donde tendremos que indicar las direcciones IP de las máquinas a las que se permite el acceso. Si comentamos esta propiedad, se podrá acceder desde cualquier máquina. En nuestro caso indicaremos la IP del servidor de aplicaciones *appsserver (169.254.1.103)*.

A continuación, creamos las bases de datos correspondientes a cada aplicación:

```
create database gestion_usuarios;
create database asignaturas_uoc;
create database ofertas_empleo;
create database biblioteca;
```

Y los usuarios y permisos correspondientes:

```
create user 'gestion_usuarios'@'appsserver.tfm.int' IDENTIFIED
BY 'gestion_usuarios';
GRANT SELECT ON gestion_usuarios.* TO
gestion_usuarios@appsserver.tfm.int;
GRANT INSERT ON gestion_usuarios.* TO
gestion_usuarios@appsserver.tfm.int;
GRANT UPDATE ON gestion_usuarios.* TO
gestion_usuarios@appsserver.tfm.int;
GRANT DELETE ON gestion_usuarios.* TO
gestion_usuarios@appsserver.tfm.int;

create user 'asignaturas_uoc'@'appsserver.tfm.int' IDENTIFIED
BY 'asignaturas_uoc';
GRANT SELECT ON asignaturas_uoc.* TO
asignaturas_uoc@appsserver.tfm.int;

create user 'ofertas_empleo'@'appsserver.tfm.int' IDENTIFIED
BY 'ofertas_empleo';
GRANT SELECT ON ofertas_empleo.* TO
ofertas_empleo@appsserver.tfm.int;

create user 'biblioteca'@'appsserver.tfm.int' IDENTIFIED BY
'biblioteca';
GRANT SELECT ON biblioteca.* TO
biblioteca@'appsserver.tfm.int';
```

Los scripts ejecutados con el detalle de las tablas generadas, se pueden consultar en los anexos de este documento.

Configuración de OpenLDAP

La versión de OpenLDAP utilizada ha sido la 2.4.44. Durante la instalación de OpenLDAP, se crea el dominio *tfm.int*. Para crear la estructura del directorio LDAP definida, vamos a crear la unidad organizativa *users*, que colgará del dominio creado. Para ello generamos el siguiente fichero *LDIF (ou_users.ldif)*:

```
dn:ou=users,dc=tfm,dc=int
objectClass:organizationalUnit
ou:users
```

Y lo cargamos en el LDAP mediante la ejecución del comando:

ldapadd –D cn=admin, dc=tfm, dc=int –w admin –f ou_users.ldif –c

Para almacenar los usuarios en el ldap, utilizamos el elemento *inetOrgPerson.* Las contraseñas encriptadas se almacenan en el atributo *userPassword*. Para su generación, se ha utilizado la utilidad *slappasswd* de openLDAP y el algoritmo de encriptación *SHA1*, mediante el comando *slappasswd -s "pass"*.

Pruebas básicas de funcionamiento de la plataforma

A continuación se muestran una serie de pruebas básicas del funcionamiento de la plataforma. La demostración completa de toda la funcionalidad, se puede consultar en el vídeo adjunto a la memoria. Cabe señalar que el requisito funcional RF002. Autenticación en sistema mediante certificado digital, se ha desarrollado parcialmente, ya que, a la hora de acceder a la aplicación de Gestión de usuarios, se nos solicita tanto el certificado como el usuario y la contraseña. Esto cumple con el requisito de la necesidad disponer de un certificado digital, pero según lo definido en tiempo de análisis, el usuario debería acceder directamente a la aplicación, sin necesidad de pasar por el Login.

Para realizar las pruebas de la plataforma y poder comprobar su correcto funcionamiento, vamos a utilizar los siguientes datos de prueba:

- Usuarios:
  - Enrique Barquilla: ROLE_ADMINISTRADOR.
  - Elisa Iglesias: ROLE_USUARIO.
- Roles:
  - ROLE_ADMINISTRADOR: Acceso a todas las aplicaciones.
  - ROLE_USUARIO: Acceso a las aplicaciones de usuario de la plataforma.
  - ROLE_ASIGNATURAS_UOC: Acceso a la aplicación Asignaturas UOC.
  - ROLE_OFERTAS_EMPLEO: Acceso a la aplicación Ofertas de empleo.
  - ROLE_BIBLIOTECA: Acceso a la aplicación Biblioteca.

1. Acceso a aplicación PAU (Punto de Acceso Único): **Resultado OK**

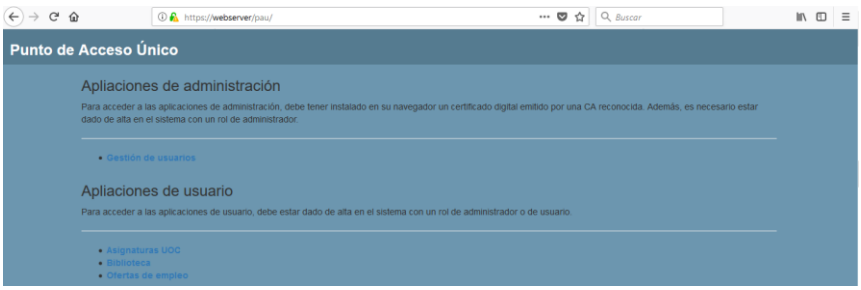

Ilustración 35. Aplicación PAU

2. Acceso a aplicación Gestión de usuarios (usuario ebarquilla, con certificado): **Resultado OK**
   a. Solicitud de certificado (certificado generado para ebarquilla con CA propia)

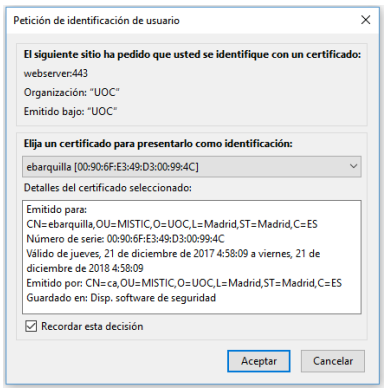

Ilustración 36. Certificado digital de cliente

   b. Login CAS (introducimos credenciales de ebarquilla)

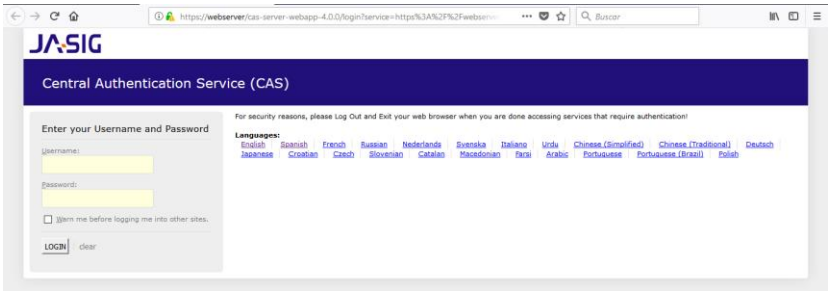

Ilustración 37. Login de servidor CAS

c. Aplicación Gestión de usuarios (se genera Cookie **CASTGC**= TGT-3-v2PBjUaOh6byb2CgZMCIddeMm40C5UTaQ90C0tS3HggpdkyDjA-casserver2.tfm.int con el ticket generado).

Nombre	Dominio	Último acceso el	Valor
CASTGC	webserver	Mon, 01 Jan 2018 01:25:50 GMT	TGT-3-v2PBjUaOh6byb2CgZMCIddeMm40C5UTaQ90C0tS3HggpdkyDjA-casserver2.tfm.int

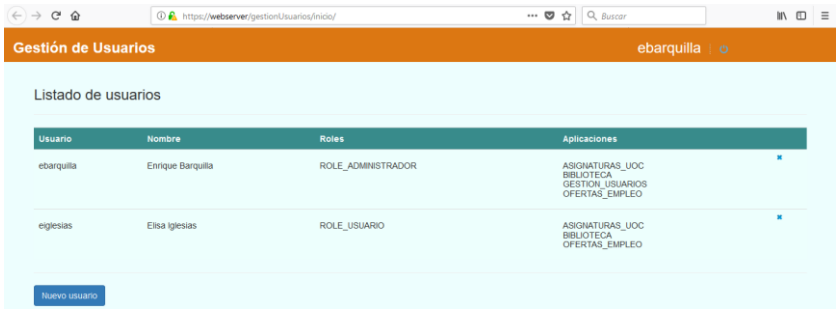

Ilustración 38. Acceso a aplicación Gestión de usuarios

d. Acceso a aplicación Asignaturas UOC con el ticket generado anteriormente, sin necesidad de volver a introducir las credenciales del usuario:

Nombre	Dominio	Último acceso el	Valor
CASTGC	webserver	Mon, 01 Jan 2018 01:50:0...	TGT-3-v2PBjUaOh6byb2CgZMCIddeMm40C5UTaQ90C0tS3HggpdkyDjA-casserver2.tfm.int

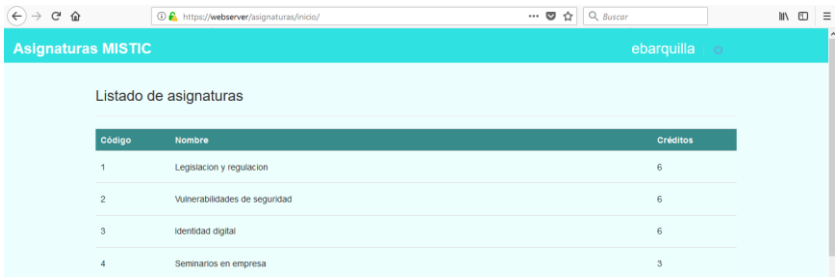

Ilustración 39. Acceso a aplicación Asignaturas UOC

3. Acceso a aplicación Gestión de usuarios (usuario eiglesias, sin certificado): **Resultado: Error de autenticación. Se necesita certificado.**

Ilustración 40. Error de autenticación

4. Acceso a aplicación Gestión de usuarios (usuario eiglesias, con certificado): **Resultado: Acceso denegado. El usuario no tiene autorización**

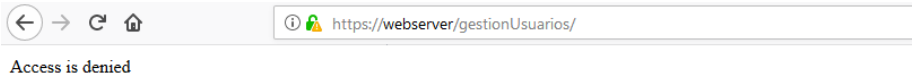

Ilustración 41. Error de autorización

Conclusiones

Una plataforma de SSO, nos ofrece múltiples ventajas sobre las autenticaciones clásicas en las que los módulos de autenticación y autorización se encontraban en el core de la propia aplicación. De manera totalmente transparente al usuario, podemos incorporar aplicaciones, incluso de distintos dominios, que compartan un mismo servidor de autenticación, facilitando la vida tanto a los usuarios (sólo tendrán que recordar unas únicas credenciales de acceso), como a los administradores, que podrán realizar una gestión más sencilla de los credenciales de los usuarios almacenadas en un único sistema centralizado.

Como hemos podido comprobar durante el desarrollo del TFM, el uso del software CAS como implementación de SSO, es una opción con gran flexibilidad, facilidad de integración y que ofrece multitud de opciones de integración con diversos tipos de sistema. Una vez implantado, el tiempo para integrar las aplicaciones web desplegadas en el servidor de aplicaciones ha sido muy inferior al tiempo que hubiéramos empleado para desarrollar un módulo de autenticación para cada una de ellas. Incorporado a una plataforma segura, con una correcta distribución de servidores y una comunicación cifrada entre cada uno de ellos, podemos decir que nos encontramos ante un sistema de autenticación con muchas ventajas sobre sistemas con autenticación descentralizada.

Objetivos cumplidos

Los objetivos cumplidos han sido los inicialmente identificados, con alguna modificación respecto al enfoque inicial:

- Implementación de sistema SSO mediante solución de software libre CAS (Central Authentication Service).
- Implementación de autenticación en el SSO a través de distintos métodos: usuario y contraseña, y certificado digital. En este punto, hay que concretar que cuando accedemos con certificado digital, el sistema también nos solicita usuario y contraseña, es decir, se realiza un doble chequeo de credenciales. Esto incumple el requisito funcional RF002. Autenticación en sistema mediante certificado digital. En el siguiente apartado se explica el motivo del nuevo diseño.
- Comunicaciones seguras entre redes externas y DMZ (servidor Apache), y entre servidor Apache y servidor Tomcat, a través del protocolo https.
- Implantación de red perimetral en una plataforma virtualizada.
- Instalación y configuración de servidor web, servidores de aplicaciones, servidor LDAP y servidor de base de datos.
- Implementación de alta disponibilidad en los sistemas críticos (servidores CAS).
- Desarrollo de aplicación web para gestionar roles de usuarios (autenticación). Se ha desarrollado la aplicación Gestión de usuarios.
- Desarrollo de aplicaciones web de usuario, para validar la autenticación en ellas haciendo uso del servicio web presente en el componente de Gestión de usuarios.

Objetivos no cumplidos

El requisito funcional RF002. Autenticación en sistema mediante certificado digital, sólo ha sido implementado parcialmente. Mediante el desarrollo realizado, se ha conseguido disponer de aplicaciones con dos niveles de seguridad en el acceso, uno (aplicaciones de usuario) con usuario y contraseña únicamente, y otro (aplicaciones de administración), con certificado digital instalado en el navegador del cliente, sin embargo, la verificación del certificado se hace en el servidor web, lo que implica que una vez pasado este primer filtro, se vuelva a solicitar usuario y contraseña. Esta

modificación ha sido necesaria por los problemas surgidos a la hora de transmitir el certificado de cliente desde el Reverse Proxy de Apache hasta el Tomcat. El motivo es que la petición el usuario, cuando llega al Reverse Proxy, se transforma en una petición nueva que es enviada al Tomcat. Esto implica que Tomcat vuelva a solicitar el certificado a Apache, provocando un error. Esto se puede corregir, desactivando el requerimiento de certificado de cliente en Tomcat y enviando los datos del certificado a través de variables de entorno SSL.

Por otro lado, el requisito no funcional RNF001. Implementación de seguridad en las comunicaciones, tampoco se ha realizado completamente. Se han securizado las comunicaciones con los servidores Apache y Tomcat, sin embargo, las comunicaciones con el resto de máquinas que contienen los servidores de aplicaciones, base de datos y LDAP, se realizar por protocolo no seguro.

En el entorno de laboratorio implementado, no se ha incluido el firewall contemplado en la arquitectura.

Posibles ampliaciones

Además de los objetivos no cumplidos detallados en el apartado anterior, las posibles mejoras que han quedado fuera del alcance de este TFM, son:

- Control de integridad entre base de datos y LDAP: Para crear un nuevo usuario en el sistema, es necesario crear por un lado el registro en el LDAP y por otro en base de datos. Una mejora en el sistema, sería que, al dar de alta un usuario desde la aplicación Gestión de usuarios, éste automáticamente se almacenará en el LDAP. Si incorporamos el atributo *memberOf* en los usuarios del LDAP, también podríamos almacenar los roles a los que pertenece.
- Personalización de pantallas: Las pantallas de login y de error, se pueden personalizar para que el usuario reciba una mejor experiencia.

Referencias bibliográficas y citas

1. Definición y tipos de Single Sign On - https://en.wikipedia.org/wiki/Single_sign-on
2. Características de un Single Sign On - https://www.chakray.com/que-es-el-single-sign-on-sso-definicion-caracteristicas-y-ventajas/
3. Qué es Single Sign On - https://www.apereo.org/projects/cas/about-cas
4. Definición de virtualización y características de Virtual Box - http://fpg.x10host.com/VirtualBox/index.html
5. Definición de Virtual Box - https://es.wikipedia.org/wiki/VirtualBox
6. Definición de sistema operativo - https://es.wikipedia.org/wiki/Sistema_operativo
7. Definición de Debian - https://www.debian.org/
8. Definición de servidor - https://es.wikipedia.org/wiki/Servidor
9. Definición de servidor web - https://es.wikipedia.org/wiki/Servidor_web
10. Documentación de HTTP Apache - https://httpd.apache.org/
11. Definición de servidor de aplicaciones - https://es.wikipedia.org/wiki/Servidor_de_aplicaciones
12. Documentación de Apache Tomcat - http://tomcat.apache.org/index.html
13. Documentación de WildFly - http://wildfly.org/
14. Topologías de WildFly - https://jboss-books.gitbooks.io/wildfly/content/estrutura/modo_standalone_x_domain.html
15. Definición de MariaDB - https://es.wikipedia.org/wiki/MariaDB
16. Definición de LDAP - https://es.wikipedia.org/wiki/Protocolo_Ligero_de_Acceso_a_Directorios
17. Definición de OpenLDAP - https://es.wikipedia.org/wiki/OpenLDAP
18. Documentación de OpenLDAP - https://www.openldap.org/
19. Certificados digitales X.509 - https://www.uv.es/sto/articulos/BEI-2003-11/certificados_digitales.html
20. Documentación OpenSSL - https://www.openssl.org/
21. Keytool en Java 8 - https://docs.oracle.com/javase/8/docs/technotes/tools/windows/keytool.html
22. Definición Eclipse - https://es.wikipedia.org/wiki/Eclipse_(software)
23. Definición Java - https://en.wikipedia.org/wiki/Java_(programming_language)
24. Documentación de Apache Maven - https://maven.apache.org/
25. Documentación de Apache CXF - http://cxf.apache.org/
26. Documentación de Spring - https://spring.io/
27. Documentación de Spring Security - https://projects.spring.io/spring-security/
28. Definición de JQuery - https://es.wikipedia.org/wiki/JQuery
29. Definición de Bootstrap - https://es.wikipedia.org/wiki/Bootstrap_(framework)
30. Configuración de Stickness en Apache con Tomcat - https://stackoverflow.com/questions/6018428/sticky-session-with-apache-web-server-and-tomcat-servers
31. Lista de implementaciones de Sigle Sign On - https://en.wikipedia.org/wiki/List_of_single_sign-on_implementations
32. Documentación de CAS 4.0.0 - https://apereo.github.io/cas/4.0.x/index.html
33. Componentes de autenticación de CAS - https://apereo.github.io/cas/4.0.x/installation/Configuring-Authentication-Components.html

Anexo – Instalación y configuración

Instalación de máquinas virtuales

Software utilizado, Virtual Box 5.2.0. Se han creado las siguientes máquinas virtuales:

- Máquina virtual *webserver*
  - Memoria: 1GB
  - Disco 20GB
  - Procesadores: 2 procesadores i5-3210M 2,5 Ghz
- Máquina virtual *casserver1*
  - Memoria: 1GB
  - Disco 20GB
  - Procesadores: 2 procesadores i5-3210M 2,5 Ghz
- Máquina virtual *casserver2*
  - Memoria: 1GB
  - Disco 20GB
  - Procesadores: 2 procesadores i5-3210M 2,5 Ghz
- Máquina virtual *appserver*
  - Memoria: 1GB
  - Disco 20GB
  - Procesadores: 2 procesadores i5-3210M 2,5 Ghz
- Máquina virtual *dataserver*
  - Memoria: 2GB
  - Disco 20GB
  - Procesadores: 2 procesadores i5-3210M 2,5 Ghz

El sistema operativo instalado en todas las máquinas virtuales, ha sido Debian Stretch 9.2.1.

En *webserver*, al ser una máquina destinada a comportarse como servidor web, la partición con mayor tamaño es la montada sobre el directorio */var*:

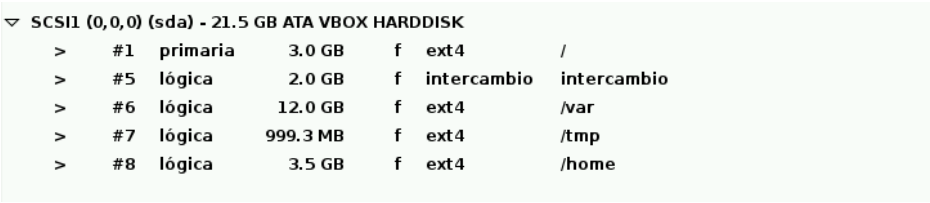

Ilustración 42. Particiones de disco de la máquina virtual webserver

Durante la instalación, seleccionamos la colección predefinida "web server", lo que instalará el servidor web Apache:

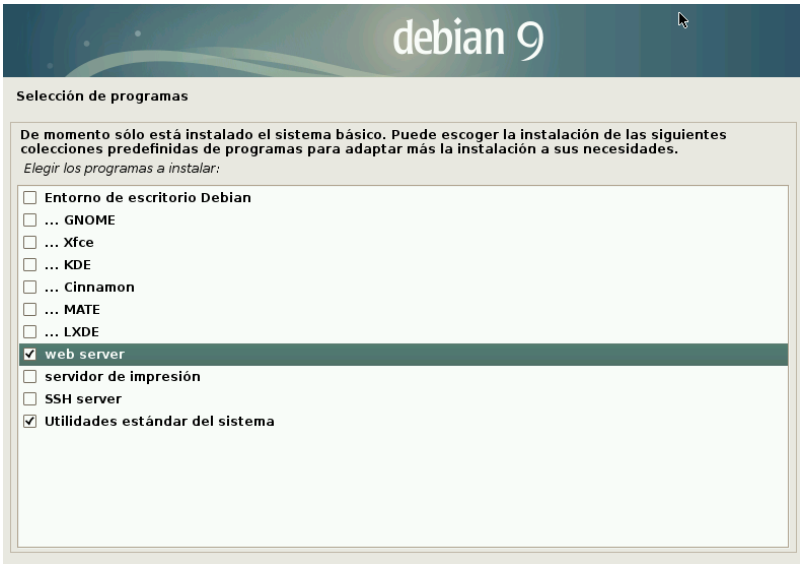

Ilustración 43. Instalación de Debian con la colección "web server"

En las máquinas virtuales *casserver1, casserver2, appsserver y dataserver*, se selecciona la misma estructura de particiones para las tres máquinas virtuales. Se asigna la partición mayor a la montada sobre el directorio */opt*, en el cual se instalarán los distintos servidores:

Ilustración 44. Particiones de disco de casserver1, casserver2, appsserver y dataserver

Una vez configuradas todas las máquinas virtuales e instalado en ellas el sistema operativo Debian Stretch 9.2.1, podemos comenzar a configurar la red perimetral:

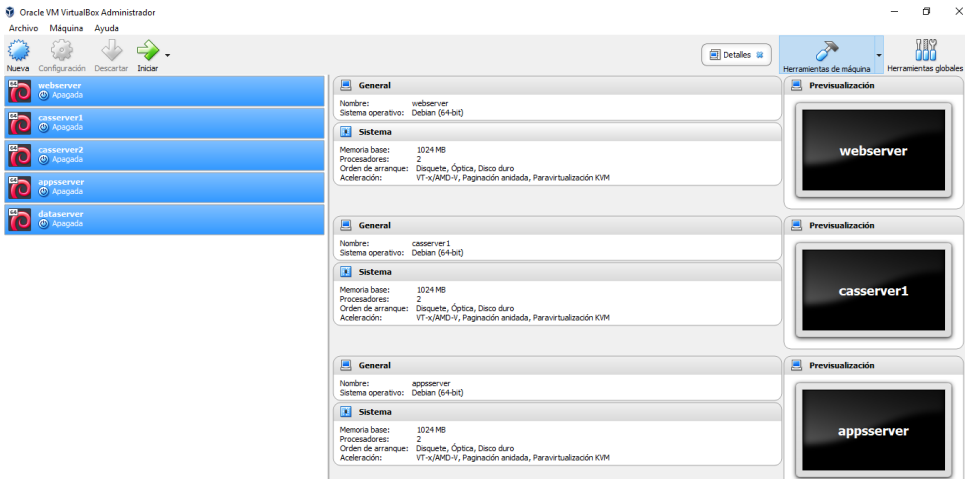

Ilustración 45. Máquinas virtuales creadas para plataforma SSO

Para la máquina virtual *webserver*, configuramos dos adaptadores de red, uno para la conexión con el exterior y otro para comunicarse con la red interna. Para la comunicación con Internet, configuramos el Adaptador 1 como "Adaptador puente". El modo Adaptador puente simula que la tarjeta virtual está conectada al mismo switch que la tarjeta física del anfitrión, por lo tanto, la

máquina virtual se va a comportar como si fuese un equipo más dentro de la misma red física en la que está el equipo anfitrión.

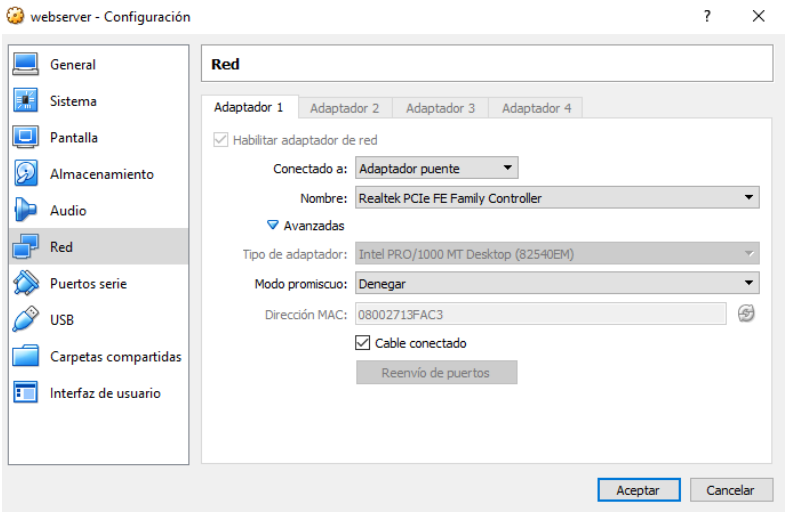

Ilustración 46. Configuración de adaptador puente de *webserver*

Para la red interna configuramos el Adaptador 2 como "Red interna". Con la configuración de tarjetas de red en modo Red interna, podemos construir redes aisladas, en las cuales solo habrá comunicación entre las máquinas virtuales que pertenezcan a la misma red interna:

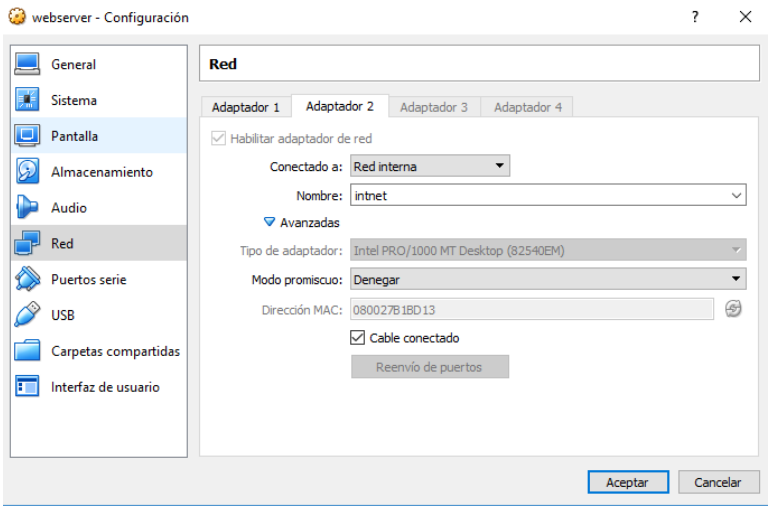

Ilustración 47. Configuración de adaptador Red interna de webserver

Configuramos las tarjetas de red con direcciones IP estáticas, modificando el fichero */etc/network/interfaces*:

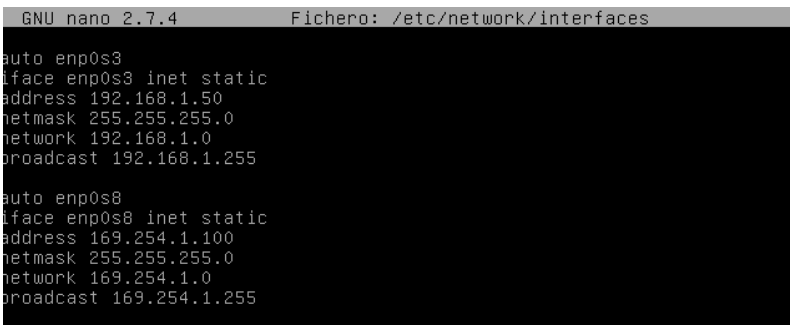

Ilustración 48. Configuración de tarjetas de red de *webserver*

El resto de máquinas virtuales, serán configuradas con adaptadores de tipo “Red interna” y se configurarán direcciones IP estáticas del mismo modo que hicimos con *webserver*:

```
# The primary network interface
auto enp0s3
allow-hotplug enp0s3
iface enp0s3 inet static
address 169.254.1.101
netmask 255.255.255.0
network 169.254.1.0
broadcast 169.254.1.255
```

Ilustración 49. Configuración de tarjeta de red de *casserver1*

```
# The primary network interface
auto enp0s3
allow-hotplug enp0s3
iface enp0s3 inet static
address 169.254.1.102
netmask 255.255.255.0
network 169.254.1.0
broadcast 169.254.1.255
```

Ilustración 50. Configuración de tarjeta de red de casserver2

```
# The primary network interface
auto enp0s3
allow-hotplug enp0s3
iface enp0s3 inet static
address 169.254.1.103
netmask 255.255.255.0
broadcast: 169.254.255.255
```

Ilustración 51. Configuración de tarjeta de red de appsserver

```
# The primary network interface
auto enp0s3
allow-hotplug enp0s3
iface enp0s3 inet static
address 169.254.1.105
netmask 255.255.255.0
network 169.254.1.0
broadcast 169.254.1.255
```

Ilustración 52. Configuración de tarjeta de red de dataserver

La asignación de direcciones IP estáticas, ha sido la siguiente:

- Webserver (dos tarjetas de red):
  - 192.168.1.50 (red DMZ)
  - 169.254.1.100 (red INTERNA)
- Casserver1:
  - 169.254.1.101 (red INTERNA)
- Casserver2:
  - 169.254.1.102 (red INTERNA)
- Appsserver:
  - 169.254.1.103 (red INTERNA)
- Dataserver:
  - 169.254.1.104 (red INTERNA)

Al no disponer de DNS interno, ha sido necesario modificar los ficheros */etc/hosts* de cada máquina, incorporando los *hostnames* y direcciones IP de todos las máquinas de la plataforma.

Para poder acceder al servidor web de nuestra máquina virtual desde el exterior, configuramos NAT en el router que da salida a Internet, para que la peticiones a los puertos 80 y 443 se redirijan a la máquina virtual *webserver*.

Instalación de Servidor HTTP Apache

El servidor web Apache se ha instalado en la máquina *webserver* durante la instalación del sistema operativo como vimos anteriormente. A continuación comprobamos la versión instalada::

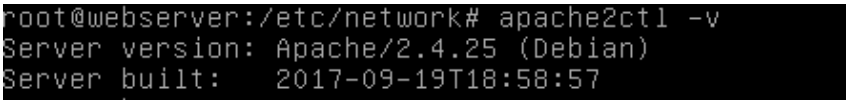

Ilustración 53. Versión de servidor HTTP Apache

Por último, comprobamos con NMAP que el servidor se encuentra corriendo en los puertos 80 y 443:

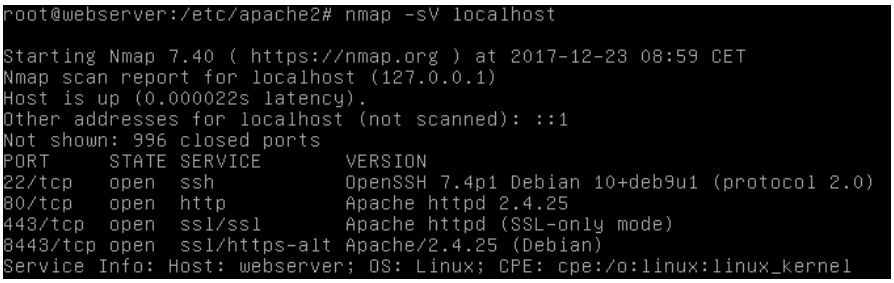

Ilustración 54. Salida NMAP en webserver

Para instalar los módulos de Apache necesarios para nuestra plataforma SSO, utilizamos el comando a2enmod:

Instalación de Servidor Tomcat

En primer lugar realizamos la instalación en la máquina casserver1:

Descargamos en el directorio /opt/tomcat mediante wget, la versión 8.5.23 de Apache tomcat:

Ilustración 55. Descarga de Apache Tomcat 8.5.23

A continuación lo descomprimimos en el directorio */opt/tomcat*, con el usuario *tomcat*. Comprobamos que tenemos instalada la máquina virtual de java, mediante el comando *java -version*:

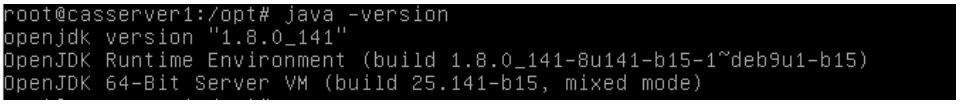

Ilustración 56. Versión JDK en servidor *casserver1*

Arrancamos el servidor ejecutando el script */opt/tomcat/apache-tomcat-8.5.23/bin/startup.sh* con el usuario *tomcat* creado anteriormente.

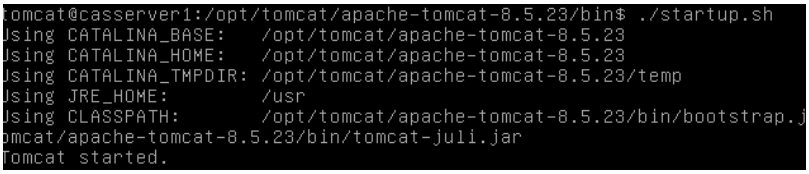

Ilustración 57. Tomcat arrancado en *casserver1*

Finalmente, comprobamos que se ha levantado correctamente accediendo desde otra máquina de la red, en este caso desde *webserver*:

Ilustración 58. Prueba de conectividad a *http://casserver1:8080*

Una vez finalizada la instalación en la máquina *casserver1*, efectuamos los mismos pasos en la máquina *casserver2*.

Instalación de Servidor WildFly

Descargamos en el directorio /opt/jboss mediante wget, la versión WildFly-9.0.1.Final de WildFly:

Ilustración 59. Descarga de servidor WildFly

A continuación lo descomprimimos en el directorio */opt/jboss*, con el usuario *jboss*. Comprobamos que tenemos instalada la máquina virtual de java, mediante el comando java -version:

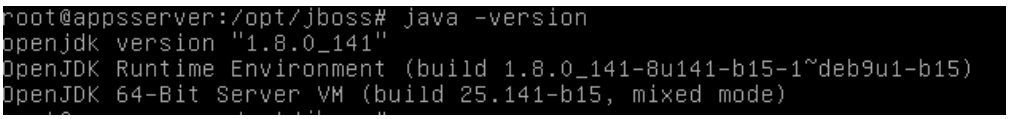

Ilustración 60. Versión de JDK en appsserver

Finalmente, arrancamos el servidor con el usuario *jboss* creado anteriormente, en modo *standalone*:

Ilustración 61. Servidor WildFly arrancado en *appsserver*

Instalación de OpenLDAP

En primer lugar vamos a instalar la versión 2.4.44 de openLDAP mediante apt-get install slapd ldap-utils. Durante la instalación, se nos pedirá que introduzcamos una contraseña para el administrador:

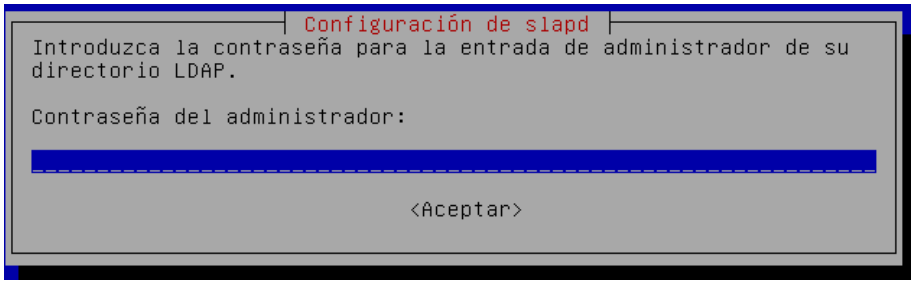

Ilustración 62. Configuración de slapd - password Admin

A continuación, configuramos openLDAP mediante el comando dpkg-reconfigure slapd. Creamos el nombre del dominio:

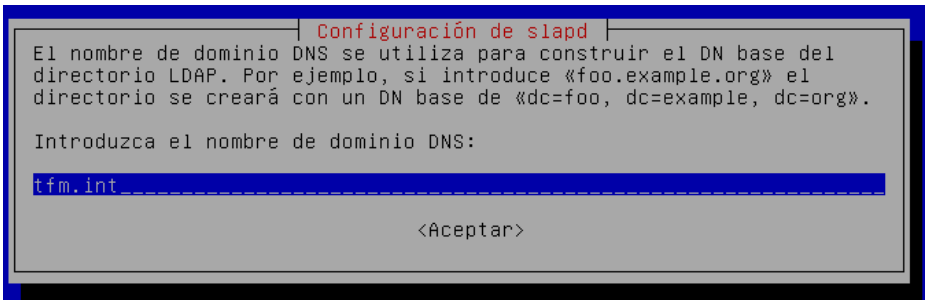

Ilustración 63. Creación de nombre de dominio en OpenLDAP

Indicamos el nombre de la organización:

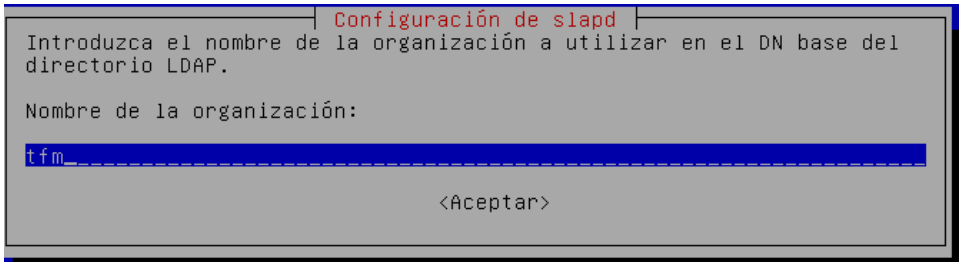

Ilustración 64. Nombre de la organización en OpenLDAP

Comprobamos que el servicio *slapd* ha arrancado correctamente, mediante el comando slapd status:

```
root@dataserver:/etc/ldap# /etc/init.d/slapd status
● slapd.service - LSB: OpenLDAP standalone server (Lightweight Directory Access
Protocol)
   Loaded: loaded (/etc/init.d/slapd; generated; vendor preset: enabled)
   Active: active (running) since Mon 2017-11-20 00:42:20 CET; 15min ago
     Docs: man:systemd-sysv-generator(8)
  Process: 1182 ExecStop=/etc/init.d/slapd stop (code=exited, status=0/SUCCESS)
  Process: 1216 ExecStart=/etc/init.d/slapd start (code=exited, status=0/SUCCESS
)
    Tasks: 3 (limit: 4915)
   CGroup: /system.slice/slapd.service
           └─1222 /usr/sbin/slapd -h ldap:/// ldapi:/// -g openldap -u openld….d

nov 20 00:42:20 dataserver systemd[1]: Starting LSB: OpenLDAP standalone se…)...
nov 20 00:42:20 dataserver slapd[1221]: @(#) $OpenLDAP: slapd  (Aug 10 2017 …) $
                                                Debian OpenLDAP Maintainers …rg>
nov 20 00:42:20 dataserver slapd[1222]: slapd starting
nov 20 00:42:20 dataserver slapd[1216]: Starting OpenLDAP: slapd.
nov 20 00:42:20 dataserver systemd[1]: Started LSB: OpenLDAP standalone ser…ol).
Hint: Some lines were ellipsized, use -l to show in full.
```

Ilustración 65. Servicio slapd arrancado

Durante el proceso de instalación, se ha creado el usuario *openldap*, que será con el que será ejecutado el proceso.

Instalación de MySQL

Para realizar la instalación de MySQL (versión 10.1.26-MariaDB), seguimos los siguientes pasos:

1. Descargamos la configuración del repositorio de mysql: wget https://dev.mysql.com/get/mysql-apt-config_0.8.6-1_all.deb
2. Instalamos la configuración del repositorio mediante la herramienta gbdeb: gdebi mysql-apt-config_0.8.6-1_all.deb
3. Actualizamos los repositorios: apt update
4. Instalamos mysql: apt install mysql-server

Finalmente, comprobamos que la instalación se ha realizado correctamente:

```
root@dataserver:/opt# mysql -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 4
Server version: 10.1.26-MariaDB-0+deb9u1 Debian 9.1

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

Ilustración 66. MySQL arrancado en dataserver

Para poder acceder desde otra máquina, en este caso desde nuestro servidor de aplicaciones *appsserver*, al servidor de base de datos MySQL, es necesario modificar el fichero de configuración *50-server.cnf*. Comentaremos las propiedades *skip-external-locking* y *bind-address*.

Instalación de servidor CAS y prueba de concepto

Vamos a proceder a la instalación del software del CAS en las máquinas virtuales *casserver1* y *casserver2*.

En primer lugar vamos a descargar la versión 4.0.0 de CAS Server del repositorio github: *https://github.com/Jasig/cas/releases/download/v4.0.0/cas-server-4.0.0-release.zip*.

A continuación descomprimimos el fichero y desplegamos en el servidor tomcat la aplicación *cas-server-webapp-4.0.0.war* del directorio */modules*.

Para probar que la aplicación CAS funciona correctamente, vamos a desarrollar una pequeña aplicación en la cual se implementará autenticación mediante el CAS instalado en la máquina *casserver1*. Utilizaremos el sistema de autenticación por defecto del servidor CAS, es decir, *AcceptUsersAuthenticationHandler*, por el cual, los usuarios y contraseñas de acceso se establecen en el fichero *deployerConfigContext.xml*. La aplicación simplemente consistirá en una pantalla estática a la que se podrá acceder autenticándose en el cas con el usuario casuser y contraseña Mellon.

```
<bean id="primaryAuthenticationHandler"
      class="org.jasig.cas.authentication.AcceptUsersAuthenticationHandler">
    <property name="users">
        <map>
            <entry key="casuser" value="Mellon"/>
        </map>
    </property>
</bean>
```

En la aplicación *dummyCasApp,* vamos a utilizar el framework *spring-security* para configurar la autenticación mediante CAS:

En primer lugar, vamos a establecer un *entry-point*. En este caso, mediante el patrón /** filtramos todas las peticiones:

```
<security:http entry-point-ref="casAuthenticationEntryPoint" auto-config="true">
    <security:intercept-url pattern="/**" access="ROLE_USER"></security:intercept-url>
    <security:custom-filter position="CAS_FILTER" ref="casAuthenticationFilter"></security:custom-filter>
</security:http>
```

Todas las peticiones filtradas por el entry-point, son redirigidas a la página de login del CAS:

```
<bean id="casAuthenticationEntryPoint" class="org.springframework.security.cas.web.CasAuthenticationEntryPoint">
    <property name="loginUrl" value="http://webserver/cas-server-webapp-4.0.0/login"></property>
    <property name="serviceProperties" ref="serviceProperties"></property>
</bean>
```

Establecemos la url del servicio de validación de tickets del CAS:

```
<bean id="casAuthenticationProvider" class="org.springframework.security.cas.authentication.CasAuthenticationProvider">
    <property name="userDetailsService" ref="userService"></property>
    <property name="serviceProperties" ref="serviceProperties"></property>
    <property name="ticketValidator">
        <bean class="org.jasig.cas.client.validation.Cas20ServiceTicketValidator">
            <constructor-arg index="0" value="http://webserver/cas-server-webapp-4.0.0"></constructor-arg>
        </bean>
    </property>
    <property name="key" value="cas"></property>
</bean>
```

Será necesario establecer un *security-user-service*. En este caso, definimos un único usuario con acceso a la aplicación, que coincide con el que hemos indicado en el fichero de configuración del CAS.

```
<security:user-service id="userService">
    <security:user name="casuser" authorities="ROLE_USER"></security:user>
</security:user-service>
```

Generamos los ProxyPass y ProxyPassReverse en el Apache de la máquina webserver, tanto para el CAS como para la aplicación web *dummyCasApp*:

```
ProxyPass /dummyCasApp http://appsserver:8080/dummyCasApp
ProxyPassReverse /dummyCasApp http://appsserver:8080/dummyCasApp

ProxyPass /cas-server-webapp-4.0.0 http://casserver1:8080/cas-server-we$
ProxyPassReverse /cas-server-webapp-4.0.0 http://casserver1:8080/cas-se$
```

Finalmente accedemos a la aplicación web y comprobamos que nos salta la pantalla del *login* del CAS:

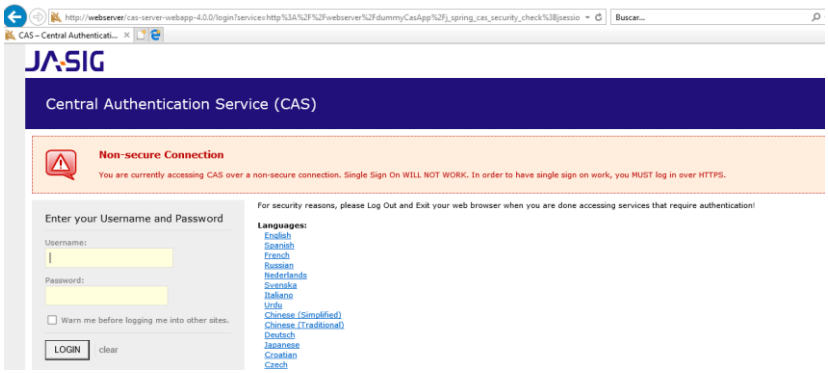

Ilustración 67. Panatalla de login de CAS

Introducimos el *casuser* y la contraseña *Mellon* y accedemos correctamente a nuestra aplicación *dummy*:

Ilustración 68. Aplicación *dummy* desplegada en *appsserver*

Hay que señalar que esta prueba de concepto no es válida para la implementación del SSO, ya que la comunicación se ha hecho por *http*, y así se indica en el mensaje que muestra el *login* de CAS:

Ilustración 69. Aviso de conexión no segura a CAS

Bases de datos de aplicaciones web

Se crea una base de datos para la aplicación de gestión de usuarios y otra para cada una de las aplicaciones que se instalan en nuestro sistema SSO (asignaturas UOC, ofertas de empleo y biblioteca):

```
create database gestion_usuarios;
create database asignaturas_uoc;
create database ofertas_empleo;
create database biblioteca;
```

A continuación, creamos los usuarios y permisos de cada base de datos:

```
create user 'gestion_usuarios'@'appsserver.tfm.int' IDENTIFIED BY 'gestion_usuarios';
GRANT SELECT ON gestion_usuarios.* TO gestion_usuarios@appsserver.tfm.int;
```

```
GRANT INSERT ON gestion_usuarios.* TO gestion_usuarios@appsserver.tfm.int;
GRANT UPDATE ON gestion_usuarios.* TO gestion_usuarios@appsserver.tfm.int;
GRANT DELETE ON gestion_usuarios.* TO gestion_usuarios@appsserver.tfm.int;

create user 'asignaturas_uoc'@'appsserver.tfm.int' IDENTIFIED BY 'asignaturas_uoc';
GRANT SELECT ON asignaturas_uoc.* TO asignaturas_uoc@appsserver.tfm.int;

create user 'ofertas_empleo'@'appsserver.tfm.int' IDENTIFIED BY 'ofertas_empleo';
GRANT SELECT ON ofertas_empleo.* TO ofertas_empleo@appsserver.tfm.int;

create user 'biblioteca'@'appsserver.tfm.int' IDENTIFIED BY 'biblioteca';
GRANT SELECT ON biblioteca.* TO biblioteca@'appsserver.tfm.int';
```

Creamos las tablas de la aplicación Gestión de usuarios:

```
create table usuarios (
  usuario_id char(32) not null,
  nombre_usuario char (64),
  primary key (usuario_id)
);

create table roles (
  rol_id char(32) not null,
  descripcion char(100),
  primary key (rol_id)
);

create table usuarios_roles (
  usuario_fk char(32) not null,
  rol_fk char(32) not null,
  foreign key (usuario_fk) references usuarios(usuario_id),
  foreign key (rol_fk) references roles(rol_id),
  primary key (usuario_fk,rol_fk)
);

create table aplicaciones (
  aplicacion_id char(32) not null,
  nombre char(64),
  descripcion char(100),
  primary key (aplicacion_id)
);

create table roles_aplicaciones (
  rol_fk char(32) not null,
  aplicacion_fk char(32) not null,
  foreign key (rol_fk) references roles(rol_id),
  foreign key (aplicacion_fk) references aplicaciones(aplicacion_id),
  primary key (rol_fk,aplicacion_fk)
);
```

Creamos las tablas de las aplicaciones de usuario:

```
create table asignaturas (
  codigo int not null,
  nombre char(100) not null,
  creditos int not null,
  primary key (codigo)
);

create table ofertas (
```

```
  codigo int not null,
  nombre char(64) not null,
  descripcion char(128) not null,
  empresa char(32) not null,
  primary key (codigo)
);

create table libros (
  codigo int not null,
  nombre char(32) not null,
  autor char(64),
  descripcion text,
  primary key (codigo)
);
```

Finalmente, cargamos los datos de prueba en las bases de datos:

Aplicación Gestión de Usuarios:

```
INSERT INTO usuarios (USUARIO_ID,NOMBRE_USUARIO)
VALUES ('ebarquilla', 'Enrique Barquilla');

INSERT INTO roles (ROL_ID,DESCRIPCION)
VALUES ('ROLE_ADMINISTRADOR','Rol con permisos de acceso a todas las aplicaciones del sistema');
INSERT INTO roles (ROL_ID,DESCRIPCION)
VALUES ('ROLE_USUARIO','Rol con permisos de acceso a todas las aplicaciones del sistema excepto gestion de
usuarios');
INSERT INTO roles (ROL_ID,DESCRIPCION)
VALUES ('ROLE_ASIGNATURAS_UOC','Rol con permisos de acceso a la aplicación asignaturas uoc');
INSERT INTO roles (ROL_ID,DESCRIPCION)
VALUES ('ROLE_OFERTAS_EMPLEO','Rol con permisos de acceso a la aplicación ofertas de empleo');
INSERT INTO roles (ROL_ID,DESCRIPCION)
VALUES ('ROLE_BIBLIOTECA','Rol con permisos de acceso a la aplicación biblioteca');

INSERT INTO usuarios_roles (USUARIO_FK,ROL_FK)
VALUES ('ebarquilla','ROLE_ADMINISTRADOR');

INSERT INTO aplicaciones (APLICACION_ID, NOMBRE, DESCRIPCION)
VALUES ('GESTION_USUARIOS', 'Gestión de usuarios', 'Aplicaciones para gestionar los roles de los usuarios con
acceso al sistema');
INSERT INTO aplicaciones (APLICACION_ID, NOMBRE, DESCRIPCION)
VALUES ('ASIGNATURAS_UOC', 'Asignaturas UOC', null);
INSERT INTO aplicaciones (APLICACION_ID, NOMBRE, DESCRIPCION)
VALUES ('OFERTAS_EMPLEO', 'Ofertas de empleo', null);
INSERT INTO aplicaciones (APLICACION_ID, NOMBRE, DESCRIPCION)
VALUES ('BIBLIOTECA', 'Biblioteca', null);

INSERT INTO roles_aplicaciones (ROL_FK, APLICACION_FK)
VALUES ('ROLE_ADMINISTRADOR','GESTION_USUARIOS');
INSERT INTO roles_aplicaciones (ROL_FK, APLICACION_FK)
VALUES ('ROLE_ADMINISTRADOR','ASIGNATURAS_UOC');
INSERT INTO roles_aplicaciones (ROL_FK, APLICACION_FK)
VALUES ('ROLE_ADMINISTRADOR','OFERTAS_EMPLEO');
INSERT INTO roles_aplicaciones (ROL_FK, APLICACION_FK)
VALUES ('ROLE_ADMINISTRADOR','BIBLIOTECA');
INSERT INTO roles_aplicaciones (ROL_FK, APLICACION_FK)
VALUES ('ROLE_USUARIO','ASIGNATURAS_UOC');
INSERT INTO roles_aplicaciones (ROL_FK, APLICACION_FK)
```

```
VALUES ('ROLE_USUARIO','OFERTAS_EMPLEO');
INSERT INTO roles_aplicaciones (ROL_FK, APLICACION_FK)
VALUES ('ROLE_USUARIO','BIBLIOTECA');
INSERT INTO roles_aplicaciones (ROL_FK, APLICACION_FK)
VALUES ('ROLE_ASIGNATURAS_UOC','ASIGNATURAS_UOC');
INSERT INTO roles_aplicaciones (ROL_FK, APLICACION_FK)
VALUES ('ROLE_OFERTAS_EMPLEO','OFERTAS_EMPLEO');
INSERT INTO roles_aplicaciones (ROL_FK, APLICACION_FK)
VALUES ('ROLE_BIBLIOTECA','BIBLIOTECA');
```

Aplicación Asignaturas UOC:

```
INSERT INTO asignaturas (CODIGO, NOMBRE, CREDITOS)
VALUES (1,'Legislacion y regulacion',6);
INSERT INTO asignaturas (CODIGO, NOMBRE, CREDITOS)
VALUES (2,'Vulnerabilidades de seguridad',6);
INSERT INTO asignaturas (CODIGO, NOMBRE, CREDITOS)
VALUES (3,'Identidad digital',6);
INSERT INTO asignaturas (CODIGO, NOMBRE, CREDITOS)
VALUES (4,'Seminarios en empresa',3);
INSERT INTO asignaturas (CODIGO, NOMBRE, CREDITOS)
VALUES (5,'Seminarios de investigacion',3);
INSERT INTO asignaturas (CODIGO, NOMBRE, CREDITOS)
VALUES (6,'Trabajo fin de master',6);
INSERT INTO asignaturas (CODIGO, NOMBRE, CREDITOS)
VALUES (7,'Seguridad en redes',6);
INSERT INTO asignaturas (CODIGO, NOMBRE, CREDITOS)
VALUES (8,'Seguridad en sistemas operativos',6);
INSERT INTO asignaturas (CODIGO, NOMBRE, CREDITOS)
VALUES (9,'Seguridad en bases de datos',6);
INSERT INTO asignaturas (CODIGO, NOMBRE, CREDITOS)
VALUES (10,'Comercio electronico',6);
INSERT INTO asignaturas (CODIGO, NOMBRE, CREDITOS)
VALUES (11,'Programacion de codigo seguro',6);
INSERT INTO asignaturas (CODIGO, NOMBRE, CREDITOS)
VALUES (12,'Biometria',6);
INSERT INTO asignaturas (CODIGO, NOMBRE, CREDITOS)
VALUES (13,'Sistemas de gestion de la seguridad',6);
INSERT INTO asignaturas (CODIGO, NOMBRE, CREDITOS)
VALUES (14,'Auditoria tecnica',6);
INSERT INTO asignaturas (CODIGO, NOMBRE, CREDITOS)
VALUES (15,'Analisis forense',6);
INSERT INTO asignaturas (CODIGO, NOMBRE, CREDITOS)
VALUES (16,'Tecnicas de marcado de la informacion',6);
INSERT INTO asignaturas (CODIGO, NOMBRE, CREDITOS)
VALUES (17,'Direccion estrategica de sistemas y tecnologias de la informacion',6);
```

Aplicación Ofertas de empleo:

```
INSERT INTO ofertas (codigo, nombre, descripcion, empresa)
VALUES (1, 'Analista Ciberseguridad Junior', 'Te buscamos a TI. A ti, que acabas de terminar una etapa de tu vida, te
acabas de licenciar en: Informatica o Telecomunicaciones. Y que tengas especial interes en el area de Ciberseguridad.
A ti que piensas en... Trabajar en la mejora de la experiencia de usuario de la nueva generacion de tiendas y canales
digitales.', 'MINSAIT');
INSERT INTO ofertas (codigo, nombre, descripcion, empresa)
```

VALUES (2, 'Junior Ciberseguridad', 'Capgemini, en pleno proceso de expansion de sus operaciones en Ciberseguridad en Espana, busca a personas creativas y dinamicas, que tengan un caracter emprendedor y perseverante, que aporten su propia vision e iniciativas al desarrollo de la practica y el offering de la compania en Espana.', 'Capgemini');
INSERT INTO ofertas (codigo, nombre, descripcion, empresa)
VALUES (3, 'Consultor/a ciberseguridad', 'Precisamos incorporar un/a Consultor/a de ciberseguridad para nuestra oficina de Madrid en el departamento de Risk and Advisory Services - auditoria informatica.', 'BDO');
INSERT INTO ofertas (codigo, nombre, descripcion, empresa)
VALUES (4, 'Consultor/a de Ciberseguridad (Tres Cantos)', ' Consultor/a de Ciberseguridad con experiencia entre 2 y 3 anos en la definicion, homogeneizacion y optimizacion de procesos de Seguridad que permitan mejorar la coordinacion entre las diversas Administraciones de Seguridad.', 'Aaron');
INSERT INTO ofertas (codigo, nombre, descripcion, empresa)
VALUES (5, 'Operador de red/Ciberseguridad', 'GRUPO SEIDOR, multinacional espanola en plena expansion, y con mas de 3.400 profesionales en su plantilla precisa incorporar un Operador tecnico en el area de Ciberseguridad para su Cybersecurity Operations Center en su sede de Barcelona.', 'Seidor');
INSERT INTO ofertas (codigo, nombre, descripcion, empresa)
VALUES (6, 'Consultor/a Ciberseguridad', 'Actualmente buscamos consultores junior de Ciberseguridad, con aproximadamente 3 anos de experiencia en proyectos de ciberseguridad, donde colaborara en aspectos relativos a la ejecucion de procesos de analisis y gestion de riesgos tecnologicos, desarrollo de cuerpo normativo, asi como su implantacion/comunicacion, adecuacion a estandares internacionales', 'Eulen');
INSERT INTO ofertas (codigo, nombre, descripcion, empresa)
VALUES (7, 'Auditor de Ciberseguridad de Codigo Fuente', 'En el area de Outsourcing estamos buscando un Auditor de seguridad de codigo fuente o en su defecto un desarrollador experto', 'S21sec');
INSERT INTO ofertas (codigo, nombre, descripcion, empresa)
VALUES (8, 'Consultor Ciberseguridad', 'Factum IT selecciona para su departamento de Seguridad un Consultor de Ciberseguridad con altos conocimientos de sistemas con al menos tres anos de experiencia.', 'Factum');
INSERT INTO ofertas (codigo, nombre, descripcion, empresa)
VALUES (9, 'Ciberseguridad senior (CLIENTE FINAL)', 'Empresa de CIBERSEGURIDAD precisa para su Laboratorio de desarrollo a expertos programadores en CIBERSEGURIDAD', 'Seleccion IT');
INSERT INTO ofertas (codigo, nombre, descripcion, empresa)
VALUES (10, 'Tecnico de Sistemas y Ciberseguridad', 'Empresa dedicada a la Ciberseguridad, Arquitectura y Diseno de Sistemas, con experiencia de mas de 20 anos, incorporara un puesto tecnico para sus oficinas en Las Palmas.Se requieren 2 anos minimo de experiencia probada como Tecnico de Sistemas y Ciberseguridad.', 'Can Be Cloud');

Aplicación Biblioteca:

INSERT INTO libros (codigo, nombre, autor, descripcion)
values (1, 'El fuego Invisible', 'Javier Sierra', 'David Salas, un prometedor linguista del Trinity Collage de Dublin, se encuentra, despues de aterrizar en Madrid para pasar sus vacaciones, con Victoria Goodman, una vieja amiga de sus abuelos y con su joven ayudante, una misteriosa historiadora del arte. Ese hecho trastocara sus planes y lo empujara a una sorprendente carrera por averiguar que ha sucedido con una de los alumnos de la escuela de literatura que regenta lady Goodman. Para su sorpresa, la clave parece esconderse en el mito del grial y su vinculacion con Espana.');
INSERT INTO libros (codigo, nombre, autor, descripcion)
values (2, 'Una columna de fuego', 'Ken Follet', 'Una columna de fuego arranca cuando el joven Ned Willard regresa a su hogar en Kingsbridge por Navidad. Corre el ano 1558, un ano que trastocara la vida de Ned y que cambiara Europa para siempre. Las antiguas piedras de la catedral de Kingsbridge contemplan una ciudad dividida por el odio religioso. Los principios elevados chocan con la amistad, la lealtad y el amor, y provocan derramamientos de sangre. Ned se encuentra de pronto en el bando contrario al de la muchacha con quien anhela casarse, Margery Fitzgerald. Cuando Isabel I llega al trono, toda Europa se vuelve en contra de Inglaterra. La joven monarca, astuta y decidida, organiza el primer servicio secreto del pais para estar avisada ante cualquier indicio de intrigas homicidas, levantamientos o planes de invasion. En Paris, a la espera, se encuentra la seductora y obstinada Maria Estuardo, reina de los escoceses, en el seno de una familia francesa con una ambicion descomunal. Proclamada legitima soberana de Inglaterra, Maria cuenta con sus propios partidarios, que conspiran para deshacerse de Isabel.');
INSERT INTO libros (codigo, nombre, autor, descripcion)
values (3, '4 3 2 1', 'Paul Auster', 'Una novela magistral sobre el poder del destino llamada a coronar la obra de Paul Auster. El unico hecho inmutable en la vida de Ferguson es que nacio el 3 de marzo de 1947 en Newark, Nueva Jersey. A partir de ese momento, varios caminos se abriran ante el y le llevaran a vivir cuatro vidas completamente distintas, a crecer y a explorar de formas diferentes el amor, la amistad, la familia, el arte, la politica e incluso la muerte, con

algunos de los acontecimientos que han marcado la segunda mitad del siglo XX americano como telon de fondo. ?Y si hubieras tomado un camino diferente en un momento crucial de tu vida? En 4 3 2 1, su primera novela despues de siete anos, Paul Auster explora de forma magistral los limites del azar y las consecuencias de nuestras decisiones. Porque todo suceso, por minimo que parezca, abre unos caminos y cierra otros. ');
INSERT INTO libros (codigo, nombre, autor, descripcion)
values (4, 'Origen', 'Dan Brown', 'Robert Langdon, profesor de simbologia e iconografia religiosa de la universidad de Harvard, acude al Museo Guggenheim Bilbao para asistir a un trascendental anuncio que <cambiara la faz de la ciencia para siempre>. El anfitrion de la velada es Edmond Kirsch, un joven multimillonario cuyos visionarios inventos tecnologicos y audaces predicciones lo han convertido en una figura de renombre mundial. Kirsch, uno de los alumnos mas brillantes de Langdon anos atras, se dispone a revelar un extraordinario descubrimiento que dara respuesta a las dos preguntas que han obsesionado a la humanidad desde el principio de los tiempos. ?DE DoNDE VENIMOS? ?ADoNDE VAMOS?');
INSERT INTO libros (codigo, nombre, autor, descripcion)
values (5, 'Mas alla del invierno', 'Isabel Allende', 'Isabel Allende parte de la celebre cita de Albert Camus -<en medio del invierno aprendi por fin que habia en mi un verano invencible>- para urdir una trama que presenta la geografia humana de unos personajes propios de la America de hoy que se hallan <en el mas profundo invierno de sus vidas>: una chilena, una joven guatemalteca ilegal y un maduro norteamericano. Los tres sobreviven a un terrible temporal de nieve que cae en pleno invierno sobre Nueva York y acaban aprendiendo que mas alla del invierno hay sitio para el amor inesperado y para el verano invencible que siempre ofrece la vida cuando menos se espera. Mas alla del invierno es una de las historias mas personales de Isabel Allende: una obra absolutamente actual que aborda la realidad de la emigracion y la identidad de la America de hoy a traves de unos personajes que encuentran la esperanza en el amor y en las segundas oportunidades.');
INSERT INTO libros (codigo, nombre, autor, descripcion)
values (6, 'Patria', 'Fernando Aramburu', 'El retablo definitivo sobre mas de 30 anos de la vida en Euskadi bajo el terrorismo. El dia en que ETA anuncia el abandono de las armas, Bittori se dirige al cementerio para contarle a la tumba de su marido el Txato, asesinado por los terroristas, que ha decidido volver a la casa donde vivieron. ?Podra convivir con quienes la acosaron antes y despues del atentado que trastoco su vida y la de su familia? ?Podra saber quien fue el encapuchado que un dia lluvioso mato a su marido, cuando volvia de su empresa de transportes? Por mas que llegue a escondidas, la presencia de Bittori alterara la falsa tranquilidad del pueblo, sobre todo de su vecina Miren, amiga intima en otro tiempo, y madre de Joxe Mari, un terrorista encarcelado y sospechoso de los peores temores de Bittori. ?Que paso entre esas dos mujeres? ?Que ha envenenado la vida de sus hijos y sus maridos tan unidos en el pasado?');
INSERT INTO libros (codigo, nombre, autor, descripcion)
values (7, 'Eva', 'Arturo Perez-Reverte', 'Marzo de 1937. Mientras la Guerra Civil sigue su tragico curso, una nueva mision lleva a Lorenzo Falco hasta Tanger, turbulenta encrucijada de espias, traficos ilicitos y conspiraciones, con el encargo de conseguir que el capitan de un barco cargado con oro del Banco de Espana cambie de bandera. Espias nacionales, republicanos y sovieticos, hombres y mujeres, se enfrentan en una guerra oscura y sucia en la que acabaran regresando peligrosos fantasmas del pasado. Tras el exito internacional deFalco, realidad y ficcion vuelven a enlazarse magistralmente con el talento literario de Arturo Perez-Reverte en esta asombrosa novela de lectura fascinante.');
INSERT INTO libros (codigo, nombre, autor, descripcion)
values (8, 'Niebla en Tanger', 'Cristina Lopez Barrio', 'El 24 de diciembre de 1951 Paul Dingle desaparecio en el puerto de Tanger sin que se llegara a saber que fue de el. Sesenta y cuatro anos despues, Flora Gascon sospecha que es el mismo hombre con el que ha tenido una aventura en Madrid y del que se ha enamorado. El nexo entre ellos: Niebla en Tanger, la novela que Paul tenia sobre su mesilla de noche. Flora viajara hasta esta ciudad magica y llena de secretos en busca de la autora de la novela, la unica que puede decirle quien es en verdad su amante y como encontrarlo. Pronto se da cuenta de que es ella misma quien debe escribir el final de la historia, pues en esa aventura tambien esta en juego su identidad; es un viaje al fondo de si misma. Un amante fugaz. Una ciudad magica. Un misterio olvidado en el viento.');
INSERT INTO libros (codigo, nombre, autor, descripcion)
values (9, 'Los pacientes del Doctor Garcia', 'Almudena Grandes', 'Tras la victoria de Franco, el doctor Guillermo Garcia Medina sigue viviendo en Madrid bajo una identidad falsa. La documentacion que lo libro del paredon fue un regalo de su mejor amigo, Manuel Arroyo Benitez, un diplomatico republicano al que salvo la vida en 1937. Cree que nunca volvera a verlo, pero en septiembre de 1946, Manuel vuelve del exilio con una mision secreta y peligrosa. Pretende infiltrarse en una organizacion clandestina, la red de evasion de criminales de guerra y profugos del Tercer Reich que dirige desde el barrio de Argüelles una mujer alemana y espanola, nazi y falangista, llamada Clara Stauffer. Mientras el doctor Garcia se deja reclutar por el, el nombre de otro espanol se cruza en el destino de los dos amigos. Adrian Gallardo Ortega, que tuvo su momento de gloria como boxeador profesional antes de alistarse en la Division

Azul, para seguir luchando como voluntario de las SS y participar en la ultima defensa de Berlin, malvive en Alemania, ignorando que alguien pretende suplantar su identidad para huir a la Argentina de Peron.');
INSERT INTO libros (codigo, nombre, autor, descripcion)
values (10, 'Asterix en Italia', 'Rene Goscinny', 'El 19 de octubre vuelve la magia de Asterix: los famosos personajes creados por Goscinny y Uderzo regresan en su aventura numero 37, la tercera firmada por Jean-Yves Ferri y Didier Conrad. Hablamos de Asterix en Italia, que llevara a nuestros heroes hasta la peninsula italiana. Esto les permitira conocer a fondo la sorprendente Italia antigua y a los numeros pueblos italicos que viven alli.');