Universitat Oberta de Catalunya

uoc.edu

Implementación de un sistema de Single Sign-On (SSO)

Nombre Estudiante: José Manuel Rejón Santiago
Plan de Estudios: Máster Universitario en Ciberseguridad y Privacidad
Área del trabajo final: Seguridad empresarial

Nombre Consultor/a: Antoni González Ciria
Nombre Profesor/a responsable de la asignatura: Víctor García Font

1 de Junio de 2021

Agradecimientos:

- *A mi familia, pareja y amigos, por apoyarme durante estos dos años dedicados a la obtención del máster y por empujarme en esta recta final con la elaboración del TFM.*

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Implementación de un sistema de Single Sign-On (SSO)*
Nombre del autor:	*José Manuel Rejón Santiago*
Nombre del consultor/a:	*Antoni González Ciria*
Nombre del PRA:	*Víctor García Font*
Fecha de entrega (mm/aaaa):	06/2021
Titulación:	*Máster Universitario en Ciberseguridad y Privacidad*
Área del Trabajo Final:	*Seguridad empresarial*
Idioma del trabajo:	Castellano
Palabras clave	*SSO, Autenticación, Alta disponibilidad*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo.*

El objetivo de este trabajo es el de implementar un sistema de autenticación y autorización para permitir el acceso de los usuarios (empleados) a las aplicaciones web que les permiten desempeñar su función laboral diaria. En concreto, el sistema de autenticación estará basado en la capacidad que ofrece Single Sign-On (SSO), de modo que se disponga de un solo inicio de sesión entre aplicaciones.

A lo largo de este documento se contará con una descripción detallada sobre qué aporta un sistema de SSO, por qué y cuándo es conveniente aplicarlo y qué diferentes opciones de implementación se pueden elegir.

Tras realizar el análisis de la posibilidades que ofrece la autenticación mediante un sistema de SSO, se mostrará cómo llevar a cabo la implementación y las configuraciones típicas de este modelo.

Abstract (in English, 250 words or less):

The main objective of this project is to implement an authentication and authorization system which allows access to the users (employees) to the web applications of the organization in which they work. In particular, the chosen authentication system is based on a Single Sign-on scheme. This type permits a single authentication process between different apps.

This document there will include a detailed description about what is a SSO, why and when this authentication scheme would be a great choice and a study about the different SSO types.

Once the analysis about SSO types is done, we will learn how to implement this authentication system and the typical configurations of this model.

Índice

1. Introducción **8**
1.1 Contexto y justificación del Trabajo 8
1.2 Objetivos del Trabajo 8
1.3 Enfoque y método seguido 9
1.4 Planificación del Trabajo 9
1.5 Breve descripción de los otros capítulos de la memoria 10

2. Análisis y diseño **11**
2.1 Arquitectura del proyecto 11
2.2 Entorno de despliegue 12
2.3 Componentes de la solución 13
2.3.1 Lenguajes de programación 13
2.3.2 Servidor Web 13
2.3.3 Servidor de datos - LDAP 13
2.3.4 Servidor de aplicaciones 14
2.3.5 Servidor de Single Sign-On 14
2.3.6 Firewalls 20
2.4 Funcionalidades del proyecto 20
2.4.1 Aplicaciones de la organización 20
Escritorio Único 20
Herramientas de desarrolladores 21
Formación 21
2.4.2 AuthN & AuthZ 21
Autenticación 21
Autorización 22

3. Desarrollo e implantación **23**
3.1 Desarrollo de las aplicaciones 23
3.2 Crear directorio de empresa 26
3.3 Configuración de la red 28
3.4 Integración SSO 31
3.4.1 Primera interacción 31
3.4.2 Integración CAS y OpenLDAP 33
3.4.3 WebServer 34
3.4.4 Protección de los recursos 37
3.5 Securización de las comunicaciones 39
3.5.1 Generación de certificados 39
3.5.2 Configuración del servidor web 41
3.5.3 Configuración de los servidores de SSO 41
3.5.4 Configuración LDAP 43

4. Pruebas **45**
4.1 SSO entre aplicaciones 45
4.2 Acceso a recursos según permisos 47
4.3 Alta disponibilidad 48

5. Conclusiones **50**
5.1 Conclusiones del trabajo 50
5.2 Logro de los objetivos 50
5.3 Análisis crítico 50

6. Glosario **51**

7. Bibliografía **53**

8. Anexos **57**
8.1 Configuración Máquinas VirtualBox 57
8.2 Configuración inicial de los servidores 57
8.3 Instalación Apache HTTP Server Project 58
8.4 Instalación OpenLDAP 59
8.5 Instalación Apache Tomcat 9 60
8.6 Instalación de Central Authentication Service (CAS) 65
8.7 Bean's del fichero deployerConfigContext.xml 66

1. Introducción

1.1 Contexto y justificación del Trabajo

Vivimos en una sociedad tecnológica en la que las personas dependemos de diversos servicios informáticos para desarrollar nuestra vida, tanto en el ámbito laboral como en el ámbito personal. Todos estos servicios que usamos a diario (llegando incluso a ocupar la mayor parte de nuestro día a día) normalmente requerirán de un registro de la persona en los sistemas que prestan el servicio.

Según las características del servicio tendremos que realizar un registro en la aplicación que solicite una determinada cantidad de datos personales del usuario como pudieran ser nombre y apellidos, DNI, email, dirección postal... pero hay un mínimo de datos que siempre se va a requerir para hacer uso de un servicio que requiera de autenticación, que serán las credenciales del usuario.

Las credenciales del usuario típicamente estarán formadas por un nombre único del usuario en el sistema y una contraseña, donde este nombre único de usuario en el sistema podría ser un username, un DNI, un email, un código alfanumérico o cualquier otra opción que identifique de forma unívoca a un usuario, de modo que éste, mediante la combinación de ese nombre único más la contraseña, pueda consumir el servicio en cuestión de forma autenticada.

A modo de símil, tenemos que entender que las credenciales que usamos para el uso de distintas aplicaciones son como las llaves de nuestra casa, de nuestro vehículo o de una caja fuerte que contenga algo de mucho valor.

Aplicando lo anteriormente expuesto a la vida laboral, se presentan varios casos en los que los empleados requieren de distintas aplicaciones para desempeñar su función, teniendo como resultado credenciales distintas para cada herramienta que deben usar. Por ejemplo, si un empleado hace uso de 4 herramientas distintas al día, implicará que tenga que memorizar 4 credenciales distintas y realizar el proceso de autenticación 4 veces cada día que se conecte para realizar su trabajo.

Ante esta situación y dado que la digitalización es un hecho más que evidente en estos tiempos, tiene sentido que se apliquen medidas desde un punto de vista de la seguridad informática para proteger las credenciales, en este caso, de los empleados que acceden a diversas herramientas para desempeñar su función laboral.

1.2 Objetivos del Trabajo

A través de este Trabajo Fin de Máster[1] se pretende ofrecer una solución de autenticación para aquellas empresas que hagan uso de diversas herramientas por parte de sus empleados y que requieran de credenciales para identificar al usuario que accede a éstas. Para ello, se hará uso de un sistema de autenticación unificado, también conocido como Single Sign-On (SSO).

Single Sign-On (SSO)[2] ofrece la capacidad de que un usuario disponga de unas credenciales únicas y realizar con éstas un único proceso de login para autenticarse en las distintas aplicaciones que conforman el ecosistema de herramientas de una empresa.

De este modo, el empleado introducirá una única vez las credenciales para autenticarse en el sistema y esta autenticación será válida para el resto de aplicaciones integradas con SSO, evitando así realizar un proceso de autenticación por cada aplicación con la que trabaja y disponer de distintas credenciales para cada herramienta.

Los objetivos que se persiguen tras lo expuesto serían:

- Analizar y comprender las soluciones disponibles de las que se dispone para implementar un sistema de SSO.
- Analizar y comprender distintas soluciones disponibles para implementar el sistema de gestión de identidades.
- Desarrollar las aplicaciones que requerirán autenticación en el sistema de la empresa.
- Implementar el sistema de gestión de identidades.
- Implementar un sistema de SSO en alta disponibilidad.

1.3 Enfoque y método seguido

Para la elaboración de este trabajo no se dispone de ningún desarrollo ya implementado, por lo que el coste será alto ya que hay que realizar todas las tareas desde cero. Se tendrán dos partes claramente diferenciadas:

- Enfoque teórico: donde realizaremos las tareas de investigación sobre distintas soluciones de Single Sign-On, gestión de identidades, perfilado de usuarios o la elección del software adecuado para montar la solución.
- Enfoque práctico: instalación, desarrollo y configuración de los productos seleccionados en el apartado teórico.

Mediante esta estrategia se realizará un esfuerzo grande en la parte teórica en cuanto a estudio y análisis pero, tras realizar este estudio de las posibles opciones que disponemos para montar un sistema de este tipo, podremos abordar la fase práctica de forma directa.

1.4 Planificación del Trabajo

A continuación se expone un diagrama de Gantt para realizar la planificación temporal del trabajo fin de máster. Para hacer este diagrama nos hemos basado en las distintas entregas de las PEC para identificar los bloques y líneas de tiempo principales:

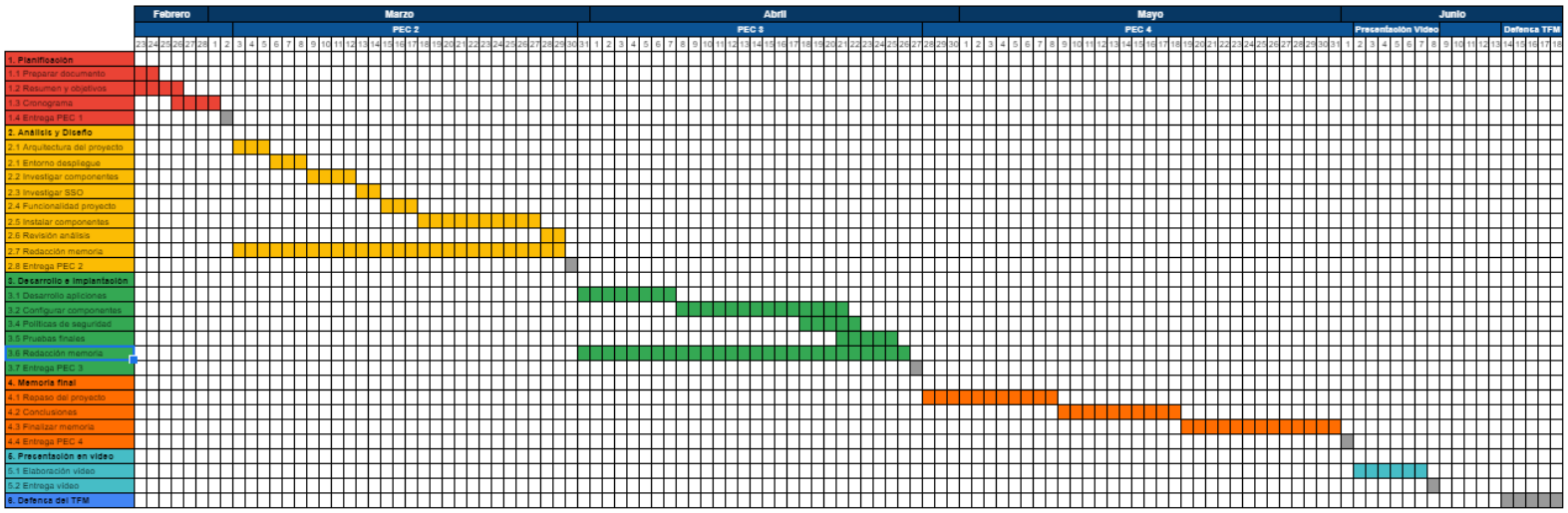

Imagen 1.1. Diagrama de Gantt

1.5 Breve descripción de los otros capítulos de la memoria

- **Capítulo 2. Análisis y diseño:** a lo largo del capítulo 2 se realizará el análisis y diseño de la plataforma. Analizaremos las piezas que conforman la arquitectura de la plataforma, como servidores, firewalls, redes, etc. Tras este capítulo, tendremos una foto clara de qué se quiere montar.
- **Capítulo 3. Desarrollo e implantación:** este capítulo aterrizará el análisis anteriormente realizado, es decir, es cuando damos lógica a todas las piezas del diseño. Se desarrollan los programas y se configuran las redes y servidores, de modo que obtenemos el producto final.
- **Capítulo 4. Pruebas:** en este apartado, como su propio nombre indica, haremos una serie de pruebas para demostrar el correcto funcionamiento de la plataforma.
- **Capítulo 5. Conclusiones:** podremos ver una serie de comentarios relacionados con las lecciones aprendidas en la elaboración del proyecto, una reflexión sobre los logros obtenidos así como objetivos que no se han conseguido.
- **Capítulo 6. Glosario:** incluirá un listado con la definición de los términos y acrónimos más relevantes utilizados en el documento.
- **Capítulo 7. Bibliografía:** capítulo que reflejará todas las fuentes consultadas para poder acometer este proyecto, ya sean páginas webs, libros, etc.
- **Capítulo 8.Anexos:** este punto contendrá todas las instalaciones de los distintos componentes utilizados (máquinas virtuales, servidores, ...) que son demasiados extensas para contar en la propia memoria.

2. Análisis y diseño

2.1 Arquitectura del proyecto

A continuación se expone la arquitectura del proyecto a alto nivel, la cual ha sido representada en un diagrama realizado con la herramienta web diagrams.net:

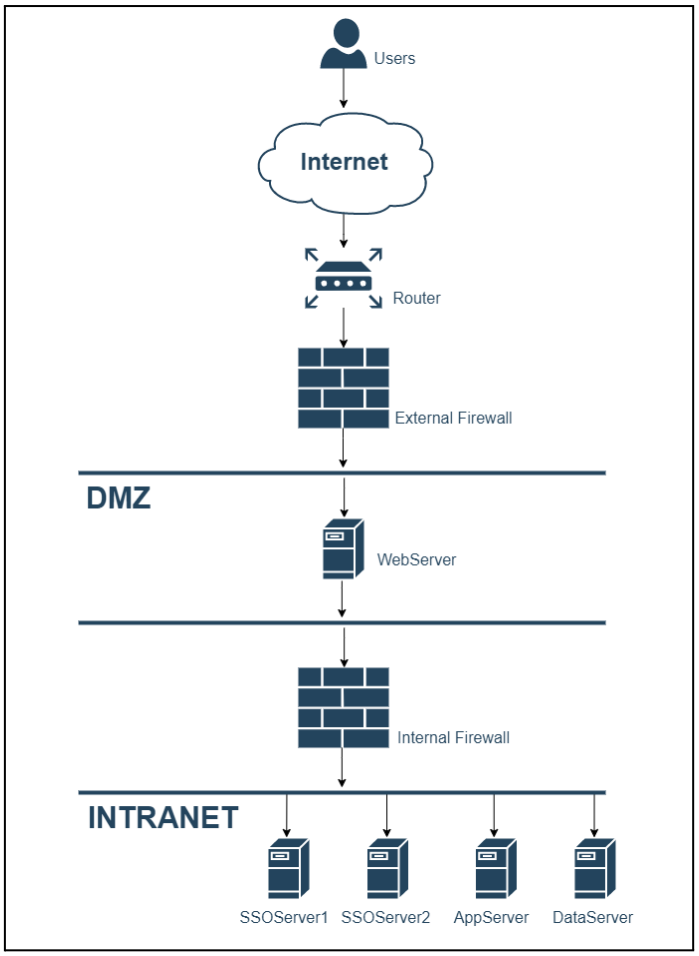

Imagen 2.1. Diagrama con la arquitectura del proyecto

Los elementos que conforman la arquitectura son:

- **Usuarios:** se trata de los empleados que acceden al sistema para desempeñar sus funciones a través de las herramientas disponibles.
- **Router**[13]**:** dispositivo que conecta y permite el tráfico de datos entre redes. En este caso, conectará la DMZ con Internet.
- **Servidores:** para realizar el proyecto necesitaremos 5 servidores:
    - **WebServer:** se trata del servidor web, que estará desplegado en la DMZ de la organización.
    - **SSOServer1:** servidor en el que se desplegará el sistema de SSO. Lo ubicamos en la intranet de la organización.

  - **SSOServer2:** réplica del anterior servidor, ambos coexisten para ofrecer un sistema en Alta Disponibilidad.
  - **AppServer:** desplegado también en la intranet, se trata del servidor de aplicaciones. Contendrá las aplicaciones de la plataforma.
  - **DataServer:** servidor que contiene el LDAP de empleados (usuarios del sistema) y podría contener las bases de datos. También lo ubicamos en la intranet de la organización.
- **Redes:** como hemos podido apreciar llegados a este punto, la organización estará compuesta por dos redes distintas: DMZ (o zona desmilitarizada) y la Intranet.
- **Firewalls:** también disponemos de dos firewalls para proteger los accesos al sistema:
  - Firewall externo: con el que protegeremos las comunicaciones que van desde Internet hacia la DMZ.
  - Firewall interno: con el que protegeremos las comunicaciones que van desde la DMZ hacia la Intranet.

2.2 Entorno de despliegue

Para desplegar la aplicación se va a hacer uso de distintas **máquinas virtuales**[3,4]. En este contexto, una máquina virtual es la simulación (virtualización) de un equipo con su sistema operativo. El uso de esta tecnología nos brinda una serie de beneficios como:

- Particionamiento:
  - Ejecutar varios sistemas operativos en una sola máquina real (física).
  - Repartir los recursos entre las máquinas virtualizadas.
- Aislamiento:
  - Proporciona seguridad de modo que los problemas que puedan presentar a una máquina no afecten a otra.
- Encapsulamiento:
  - El estado de las máquinas se guarda en ficheros y la acción de mover o copiar máquinas virtuales se limita a simplemente reubicar ficheros.
- Independencia Hardware:
  - Capacidad de migrar la máquina virtual a cualquier otra máquina física independientemente de su hardware.

Como sistema de virtualización para la realización de este proyecto se ha elegido el producto de Oracle VirtualBox, debido a que es una herramienta que nos proporciona todo lo necesario para crear una máquina virtual y está disponible de forma gratuita como Open Source Software (software de código abierto).

Como sistema operativo que soporte las capacidades de este proyecto se ha elegido Ubuntu Server en su versión 20.04.2 LTS. Decantarse por Ubuntu Server ha sido fácil, ya que es un sistema operativo que se utiliza con bastante frecuencia como servidor de aplicaciones, por lo que dispondrá de todas las herramientas necesarias para montar

nuestro proyecto, así como de una amplia y actualizada documentación. Al igual que VirtualBox, Ubuntu Server está disponible de forma gratuita como Open Source Software.

2.3 Componentes de la solución

Para este proyecto será necesario hacer uso de distintos componentes para montar la solución, desde los lenguajes de programación usados hasta los servidores involucrados. En los próximos apartados se hará mención a los distintos componentes.

2.3.1 Lenguajes de programación

Se hará uso de los siguientes lenguajes de programación para la elaboración del proyecto:

- **Java**[5] → Hacemos uso de Java para el desarrollo del backend de las aplicaciones web que conforman el ecosistema de herramientas de nuestra empresa. Se ha optado por el uso de Java tanto por tratarse de un lenguaje muy extendido en el desarrollo de aplicaciones web, como por el propio conocimiento de este lenguaje.
    - En concreto, hablamos de **JavaServer Pages (JSP)**. JSP es una tecnología que usa Java para crear páginas webs dinámicas basadas en HTML y XML.
- **HTML**[6] → Con HTML5 haremos el desarrollo de los frontales (o vistas) de las aplicaciones web.
- **CSS**[7] → Nos apoyaremos en CSS3 para dotar de estilo (o personalización) a los frontales de las aplicaciones web.

2.3.2 Servidor Web

El servidor web que vamos a utilizar es **Apache HTTP Server Project** (Apache Server)[8]. Este servidor web se distribuye bajo licencia Open Source y está disponible para poder utilizarse en sistemas operativos UNIX, como es nuestro caso.

Se puede seguir el proceso de instalación de este producto en el anexo 8.3 Instalación Apache HTTP Server Project.

2.3.3 Servidor de datos - LDAP

EL **LDAP**[9,10] (Lightweight Directory Access Protocol) o Protocolo Ligero de Acceso a Directorios se refiere a un protocolo que permite el acceso a un directorio ordenado y distribuido para buscar información. Por directorio nos referimos a un tipo de base de datos que almacena la información en estructura de árbol.

LDAP proporciona un punto interesante con respecto a la seguridad del sistema, ya que permite utilizarse para gestionar la información de los usuarios que deben registrarse para acceder al sistema. Permite almacenar distinta información del usuario (como podrían ser los datos de contacto o los permisos de éste) pero la que más nos interesa es que almacena el usuario y contraseña utilizado para realizar el proceso de autenticación en los sistemas. Por lo tanto, LDAP nos quedaría como el protocolo de acceso unificado a un conjunto de datos de los usuarios registrados.

Existen diferentes soluciones para implementar un LDAP como son Active Directory, OpenLDAP o Apache Directory Server, entre otros. En nuestro caso nos vamos a decantar por hacer uso de **OpenLDAP**.

OpenLDAP

OpenLDAP[11,12] es una implementación Open Source de LDAP. OpenLDAP es una implementación muy extendida, motivo que junto con el hecho de ser software libre hace que nos decantemos por esta solución, así como la capacidad de integrarse con los componentes listados hasta ahora.

Se puede seguir el proceso de instalación de este producto en el anexo 8.4 Instalación OpenLDAP.

2.3.4 Servidor de aplicaciones

El servidor de aplicaciones[14], como su propio nombre indica, es el servidor de la plataforma en el que se desplegarán las aplicaciones del sistema de las que harán uso los usuarios registrados. Como servidor de aplicaciones se analizó la posibilidad de usar **WildFly**[32] (antiguo JBoss), el cual es un servidor de aplicaciones Java EE[15] implementado en Java y es de código abierto.

Finalmente, dada la alta capacidad de Wildfly (que dispone de varios módulos) y que aporta una mayor complejidad al proyecto desvinculada del objetivo de éste, se decide hacer uso de **Apache Tomcat**. Apache Tomcat es un servidor web y contenedor de servlets que se utiliza para la presentación de aplicaciones Java. Podemos ver los pasos de la instalación en el anexo 8.5 Instalación Apache Tomcat 9.

2.3.5 Servidor de Single Sign-On

Single Sign-On (SSO)

Single Sign-On (SSO)[17] es un método de autenticación segura que proporciona la capacidad de acceso a diferentes aplicaciones mediante el uso de unas credenciales únicas.

Contamos con una serie de ventajas al hacer uso de este sistema:

- Mitigación del riesgo en accesos de terceras partes dado que las credenciales no se almacenan externamente.
- Al reducir a uno el número de credenciales (*user* + *pass*) no es necesario memorizar diferentes contraseñas para cada acceso.
- Se reducen tiempos, costes y errores en la gestión de la autenticación del sistema.

Por lo tanto, con SSO centralizamos la autenticación de forma segura en un punto para que el usuario acceda a sus herramientas con una única credencial, que será introducida una sola vez, es decir, no será necesario introducir esta credencial para cada aplicación del sistema.

Un esquema básico de cómo funcionaría un sistema de SSO sería el siguiente[16]:

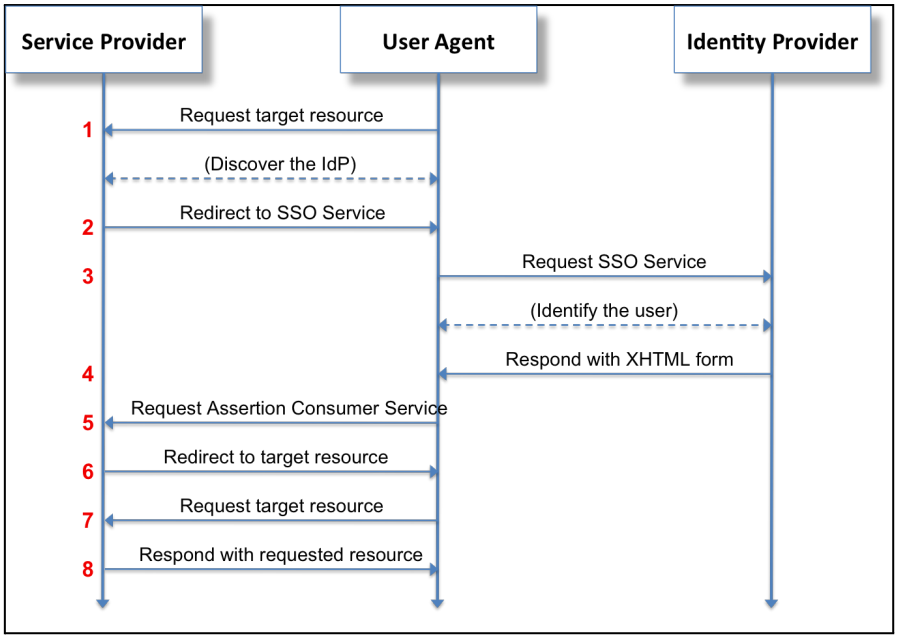

Imagen 2.2. Flujo de comunicación de un sistema de SSO

Dado que la base del proyecto gira en torno a la implementación de un sistema de Single Sign-On para la autenticación de los usuarios, haremos foco en este apartado y analizaremos las distintas soluciones posibles que existen a día de hoy.

Tipos de SSO

Antes de elegir un SSO, tenemos que tener en cuenta que existen distintos tipos:

- Gestión de Identidad Federada (Federated Identity Management - FIM): se refiere a la relación de confianza que se crea entre varios dominios o sistemas de gestión de identidad. SSO es una característica que está disponible en la arquitectura del FIM.
- OAuth 2.0: se trata de un framework que también podría ser considerado parte de un FIM. OAuth se centra en una relación de confianza que permite compartir la información de identidad del usuario entre diferentes dominios.
- OpenID Connect: es un protocolo de autenticación construido sobre OAuth 2.0 para proveer la funcionalidad de SSO.
- Security Access Markup Language (SAML): se trata de un estándar abierto que define un esquema XML para el intercambio de datos de autenticación y autorización y que también está diseñado para proveer la funcionalidad de SSO.
- Kerberos: es un método de autenticación por el cual los usuarios se registran en el servidor Kerberos y reciben un ticket. Posteriormente, se presentará ese ticket a las aplicaciones para obtener acceso al servicio.

Implementaciones de SSO

Para abordar este punto nos servimos del artículo de Wikipedia "*List of single sign-on implementations*" que muestra una tabla con diferentes implementaciones de SSO. De esta lista nos quedaremos solamente con las opciones que cumplan las siguientes premisas que nos hemos marcado:

- Open Source Software: el producto de SSO debe ser Free & Open Source tal y como venimos haciendo con todos los componentes elegidos hasta el momento.
- Compatibilidad con Ubuntu Server: al ser Ubuntu Server el sistema operativo elegido para desplegar, es necesario que el producto de SSO seleccionado sea compatible con éste.
- Compatibilidad con Java: al ser Java el lenguaje de programación elegido para realizar los desarrollos, es necesario que el producto de SSO seleccionado sea compatible con dicho lenguaje.

Los productos que cumplen con estas premisas son[18]:

- **Central Authentication Service (CAS)[19]:** CAS es un Enterprise SSO (E-SSO) que actúa interceptando los requisitos de login presentados por las aplicaciones para completarlos con el usuario y contraseña. Entre sus ventajas podemos decir que:
  - Es un protocolo altamente documentado.
  - Dispone de un componente Java Server basado en Open Source.
  - Soporte para autenticaciones con LDAP, bases de datos, certificados X.509, o doble factor de autenticación.
  - Soporte para distintos protocolos: CAS, SAML, OAuth, OpenID.
  - Librería para Java
- **Distributed Access Control System (DACS)[20]:** es un sistema *light-weight single sign-on* y dispone de un acceso de control basado en reglas para servidores web. Provee SSO para los servidores de una organización y limita el acceso a recursos.
- **FreeIPA[21]:** ofrece un sistema SSO para autenticación de los sistemas, servicios y aplicaciones de una organización. Hace uso de Kerberos para implementar las políticas de autenticación y autorización.
- **IBM Enterprise Identity Mapping[22]:** ofrece una solución SSO de IBM basa en Kerberos.
- **Java Open Single Sign On (JOSSO)[23]:** SSO Server con soporte para los protocolos SAML, OpenID, OpenID Connect, OAuth y WS-Federation, así como soporte también para LDAP y autenticación de doble factor.
- **Keycloak[24]:** solución que también ofrece soporte para LDAP, así como los protocolos OpenID Connect, OAuth 2.0 o SAML 2.0.
- **OpenAM[25]:** se trata de otro producto completo que nos brinda servicios de autenticación, autorización o proveedor de identidad, usando SAML, OAuth 2.0 u OpenID Connect.
- **Shibboleth[26]:** ofrece soporte para SAML.
- **WSO2[27]:** permite accesos Web y Móvil, incluye soporte para estándares como OIDC, SAML, OAuth o FIDO2.

Tras este análisis, bajo mi punto de vista y en el contexto que estamos trabajando, veo interesantes los productos CAS, JOSSO, Keycloak, WSO2. Finalmente se opta por hacer uso de **Central Authentication Service (CAS)** por los siguientes motivos:

1. Es un protocolo altamente documentado.
2. Estrechamente ligado a Java, que es el lenguaje elegido para el desarrollo.
3. Soporte para autenticaciones con LDAP y doble factor de autenticación.
4. Personalmente ya he tenido un acercamiento a esta tecnología en una asignatura pasada, por lo que ya se tiene algo de recorrido sobre este producto y se conocen sus capacidades.

Los pasos que hay que seguir para instalar el sistema de SSO con CAS los podemos ver en el anexo 8.6 Instalación de Central Authentication Service (CAS).

Antes de abordar la instalación del servidor de Single Sign-On, se precisa instalar **Apache Tomcat**. Apache Tomcat es un servidor web y contenedor de servlets que se utiliza para la presentación de aplicaciones Java. Podemos ver los pasos de la instalación en el anexo 8.5 Instalación Apache Tomcat 9.

Protocolo CAS

El **protocolo CAS**[45] es un protocolo basado en *tickets* y está desarrollado exclusivamente para CAS. Permite al usuario acceder a múltiples aplicaciones mientras que solamente se necesita ingresar unas credenciales únicas y una sola vez.

El sistema se compone por un servidor (CAS Server) que es el responsable de autenticar a los usuarios y otorgarles acceso a la aplicación, y uno o varios clientes (Aplicaciones CASificadas o CAS Services) que recuperan la identidad del usuario logado desde el CAS Server.

Los conceptos claves de este protocolo son:

- **TGT (Ticket Granting Ticket):** se almacena en la cookie **CASTGC** y representa la sesión SSO de un usuario.
- **ST (Service Ticket):** se transmite como parámetro GET en la url y representa el acceso otorgado por el CAS Server a la aplicación CASificada para un usuario específico.

En la siguiente imagen se muestra el flujo de una autenticación mediante el protocolo CAS tanto para un primer acceso (el usuario no dispone de sesión), como para un segundo acceso a la aplicación (el usuario ya se ha autenticado en el CAS Server) o un acceso a una segunda aplicación CASificada (disponiendo de la sesión anterior):

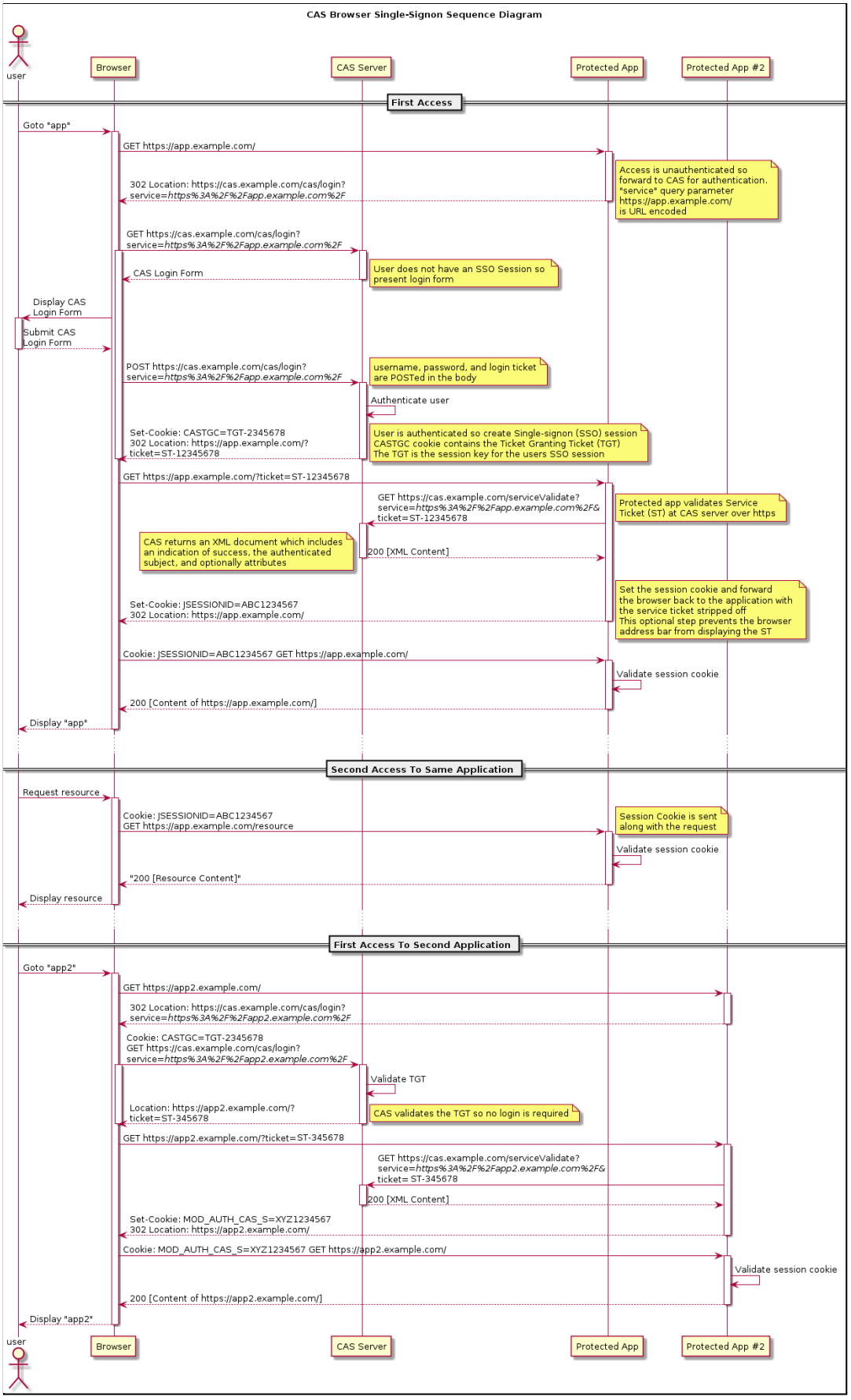

Imagen 2.3. Diagrama de flujo del protocolo CAS

Una de las características más potentes de este protocolo es la capacidad de actuar como proxy de otro servicio CAS, transmitiendo la identidad del usuario. A continuación se muestra el diagrama de flujo:

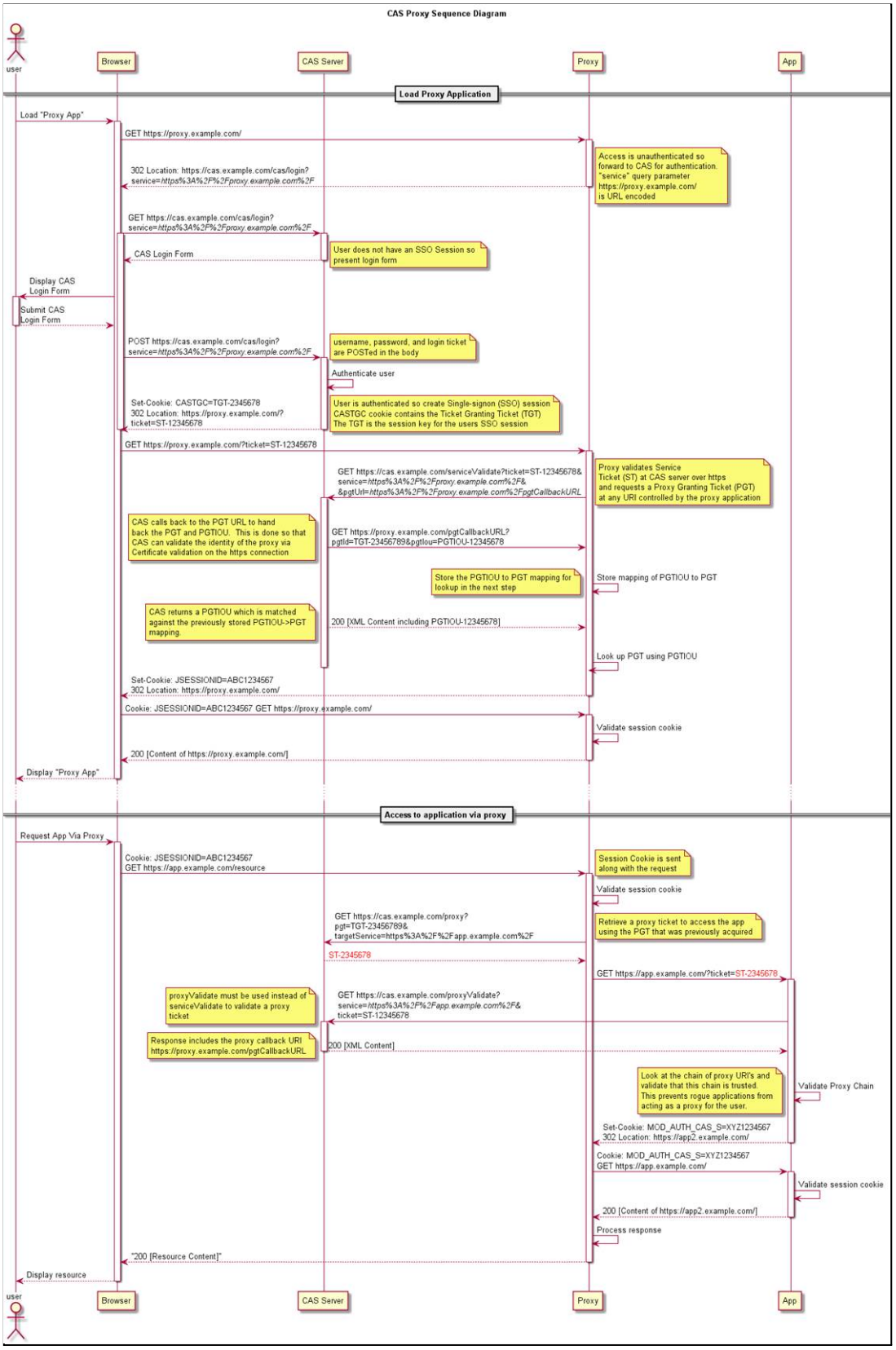

Imagen 2.4. Diagrama de flujo del protocolo CAS - Proxy

2.3.6 Firewalls

Los firewalls[54] (o cortafuegos) son una herramienta software o hardware que se emplea para bloquear accesos a un sistema informático o a una red informática.

En nuestro caso vamos a hacer uso de **UFW (Uncomplicated Firewall)**[55] para el servidor web. Por defecto, UFW está configurado para denegar todas las conexiones entrantes y permitir todas las conexiones salientes. Tenemos que añadir al firewall las reglas para los accesos HTTP y HTTPS, de modo que consultando la lista de permisos de acceso quede tal que así:

```
useradmin@webserver:/etc/apache2/certssl$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
Apache                     ALLOW       Anywhere
Apache Full                ALLOW       Anywhere
Apache (v6)                ALLOW       Anywhere (v6)
Apache Full (v6)           ALLOW       Anywhere (v6)
```

Imagen 2.5. Firewall UFW

2.4 Funcionalidades del proyecto

Para la elaboración del trabajo final de máster vamos a suponer un escenario ficticio para dotar de un contexto al proyecto que se está realizando. Para ello, vamos a suponer un escenario en el que una empresa tecnológica realiza los desarrollos de software para un Banco.

2.4.1 Aplicaciones de la organización

Los empleados van a requerir el acceso de distintas herramientas, pero en concreto serán tres las que solicitarán de autenticación y son objeto de la necesidad de montar un sistema de Single Sign-On:

1) **Escritorio Único:** plataforma central del banco que aglutina las distintas herramientas que debe utilizar el empleado en su día a día.
2) **Herramientas de desarrolladores:** plataforma complementaria que empaqueta las herramientas de desarrollo.
3) **Formación:** herramienta complementaria de formación del empleado.

Escritorio Único

Como se indicaba anteriormente, el *Escritorio Único* es la plataforma central del banco que aglutina las distintas herramientas que debe utilizar el empleado en su día a día según su puesto. Por ejemplo, para el caso de los desarrolladores, proporciona acceso a manuales, a la herramienta de gestión de usuarios o a la herramienta de registro de jornada, así como los accesos directos a las aplicaciones de *Herramientas de desarrolladores* y *Formación*.

El *Escritorio Único*, previa autenticación, presentará una serie de iconos que se corresponden con herramientas que harán o no SSO según su naturaleza. Dividimos entonces en dos las aplicaciones:

- **Aplicaciones que no requieren SSO:** son aplicaciones que vienen integradas con el *Escritorio Único*. Estas aplicaciones son:
    - **Herramienta de conocimiento del banco:** facilita al empleado información relacionada con la arquitectura del banco, manuales de desarrollo, estructura departamental, etc.
    - **Registro de jornada:** permite al empleado registrar sus horas de inicio y fin de la jornada laboral.
- **Aplicaciones que sí requieren SSO:** son las aplicaciones externas de *Formación* y *Herramientas de desarrolladores*, que disponen de un acceso directo en el *Escritorio Único* y requerirán de hacer SSO.

Herramientas de desarrolladores

Herramientas de desarrolladores es la plataforma que permite el acceso a las aplicaciones que utilizan los desarrolladores en su día a día. Esta plataforma se integra con el sistema de SSO y, una vez dentro, ofrece un listado de aplicaciones a desarrolladores que pueden variar según su rol.

Formación

La herramienta *Formación* es la última aplicación que conforma el sistema de SSO. Esta aplicación ofrece formación a desarrolladores necesaria para realizar su labor profesional dentro del banco.

2.4.2 AuthN & AuthZ

En este apartado vamos a analizar los requisitos de autenticación / autorización[28] de las herramientas que conforman el sistema de SSO, que recordemos que son *Escritorio Único, Herramientas de desarrolladores* y *Formación*.

Autenticación

Mediante la autenticación buscamos verificar la identidad de los empleados en el sistema.

Autenticación basada en usuario y contraseña: como método de autenticación nos basaremos en el acceso mediante usuario y contraseña:

- **Usuario:** ha de ser un identificador único para diferenciar a los empleados. Como usuario se va utilizar el email de los empleados.
- **Contraseña:** para ofrecer robustez a este sistema de autenticación, la contraseña contará con unos requisitos mínimos para hacerla más segura ante posibles ataques de fuerza bruta o de diccionario. Los requisitos serán:
    - Longitud mínima de 8 caracteres.
    - Ha de incluir caracteres (mayúsculas y minúsculas), números y caracteres especiales.

Autorización

Mediante el proceso de autorización verificamos los permisos que corresponden a cada empleado. Para ello, dotaremos al proyecto un sistema de Control de Acceso Basado en Roles (RBAC - Role Based Access Control)[29]. RBAC se refiere a la idea de asignar permisos a los usuarios de un de un sistema según su rol, de modo que podemos gestionar los privilegios de los empleados por grupos y no de forma nominal, cosa que nos permite realizar la asignación de permisos de forma sistemática y previene que usuarios que cambien de rol durante su vida en la empresa hereden privilegios que no pertenecen a su rol.

En nuestro proyecto vamos a contar con los siguientes roles:

- **Desarrollador:** se trata del rol base (o rol más bajo). Permite el acceso a las herramientas de desarrollo, formación, documentación o registro de jornada.
- **Editor:** además de las acciones del Desarrollador, podrá subir nueva documentación al sistema.
- **Supervisor:** engloba los roles anteriores y le permite subir el código a Producción.

3. Desarrollo e implantación

En este apartado trabajaremos los puntos relativos al desarrollo de las aplicaciones y las configuraciones necesarias para hacer funcionar la estructura que estamos montando.

Recordemos que la aplicaciones que se van a desarrollar son las siguientes:

- **Escritorio Único:** plataforma central del banco que aglutina las distintas herramientas que debe utilizar el empleado en su día a día.
- **Herramientas de desarrolladores:** plataforma complementaria que empaqueta las herramientas de desarrollo.
- **Formación:** herramienta complementaria de formación del empleado.

Para el desarrollo de las aplicaciones nos vamos a apoyar en el framework **Spring**, ya que con **Spring Security** disponemos de una herramienta potente para implementar la capa de de autorización necesaria para trabajar con los roles de usuario. Además, Spring Security se integra fácilmente con el servidor de SSO **Central Authentication Service (CAS)**, por lo que de esta unión saldrá nuestro sistema de autenticación y autorización.

3.1 Desarrollo de las aplicaciones

En este punto comentaremos los aspectos más relevantes del desarrollo de las aplicaciones web que se pondrán a disposición de los empleados.

Para desarrollar se hará uso del editor de textos **Sublime Text 3**[40,41].

Para los desarrollos de las aplicaciones web se ha seguido una estructura similar:

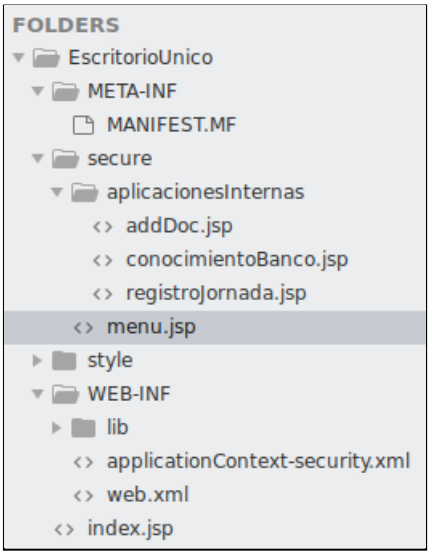

Imagen 3.1. Estructura aplicación Escritorio Único

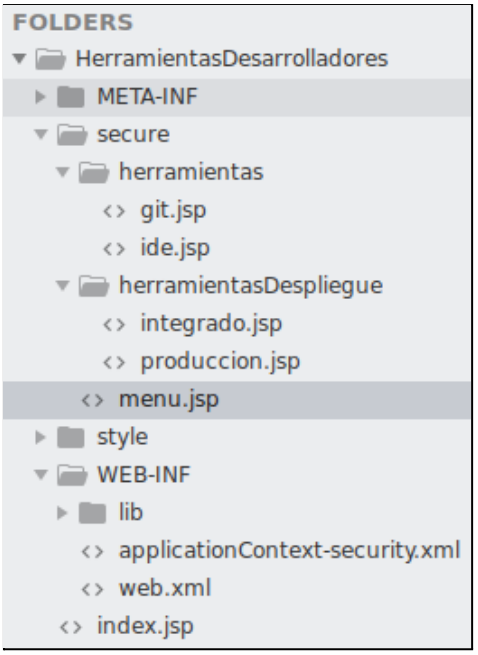

Imagen 3.2. Estructura aplicación Herramientas de desarrolladores

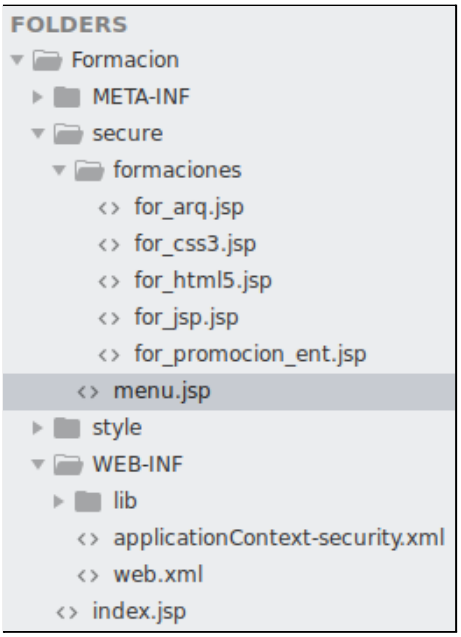

Imagen 3.3. Estructura aplicación Formación

Estos desarrollos se han hecho haciendo uso de los lenguajes JSP, HTML y CSS, además del framework Spring, cuyos ficheros de configuración están basados en XML.

De estos ficheros de configuración de Spring destacamos el fichero *"applicationContext-security.xml"*. En este fichero distinguimos distintas secciones importantes como podrían ser:

- **casAuthenticationEntryPoint:** es la sección en las que indicamos cuáles son los recursos protegidos de la aplicación, qué roles son los que tienen permisos para acceder a los recursos y también cargamos los filtros, como por ejemplo, el de logout:

```
<security:http entry-point-ref="casAuthenticationEntryPoint" auto-config="true">
    <security:intercept-url pattern="/secure/formaciones/for_promocion_ent.jsp" access="ROLE_SUPERVISOR"></
    security:intercept-url>
    <security:intercept-url pattern="/secure/**" access="ROLE_SUPERVISOR, ROLE_EDITOR, ROLE_DESARROLLADOR"></
    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>
    <security:custom-filter ref="singleLogoutFilter" before="CAS_FILTER"></security:custom-filter>
</security:http>
```

Imagen 3.4. casAuthenticationEntryPoint

- **Bean serviceProperties:** esta sección se usa para identificar el servicio al que se redirige la sesión de login de CAS:

```
<bean id="serviceProperties" class="org.springframework.security.cas.ServiceProperties">
    <property name="service" value="http://webserver/Formacion/j_spring_cas_security_check"></property>
    <property name="sendRenew" value="false"></property>
</bean>
```

Imagen 3.5. Bean serviceProperties

- **Bean casAuthenticationEntryPoint y bean casAuthenticationProvider:** el primero es usado para interceptar las peticiones de autenticación en CAS y redireccionar a la página de login de CAS. El segundo indica la gestión del proceso del ticket de autenticación de 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>

<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>
```

Imagen 3.6. Bean casAuthenticationEntryPoint y bean casAuthenticationProvider

- **Bean singleLogoutFilter:** gestiona el cierre de sesión (logout) de la aplicación:

```
<bean id="singleLogoutFilter" class="org.jasig.cas.client.session.SingleSignOutFilter" />
<bean id="requestSingleLogoutFilter" class="org.springframework.security.web.authentication.logout.LogoutFilter
">
    <constructor-arg value="http://webserver/cas-server-webapp-4.0.0/logout" />
    <constructor-arg>
        <bean class="org.springframework.security.web.authentication.logout.SecurityContextLogoutHandler" />
    </constructor-arg>
    <property name="filterProcessesUrl" value="/j_spring_security_logout" />
</bean>
```

Imagen 3.7. Bean singleLogoutFilter

Para todos los proyectos se ha optado por crear una carpeta llamada "*secure*" que es la que engloba todos los recursos que van a estar protegidos. En este caso no se dispone de ningún recurso que sea público, salvo la ventana de login para poder alcanzarlos.

Las tres aplicaciones dispuestas para los usuarios empleados de la organización están desplegadas en Tomcat para su acceso.

3.2 Crear directorio de empresa

Hemos elegido OpenLDAP como producto para el repositorio de identidades (usuarios) que van a acceder al sistema. Para añadir información a LDAP nos apoyaremos en la definición de ficheros *".ldif"*, que son ficheros de texto plano que sirven para representar el contenido del directorio LDAP.

Antes de definir los usuarios que formarán parte de la aplicación, se ha de definir la estructura del LDAP. El siguiente pantallazo muestra la definición de la nueva unidad organizativa (ou) *"empleados"*, que se creará dentro del dominio que ya registramos durante la instalación en el anexo 8.4, es decir, *"banco.org"*:

```
#Crear la unidad organizativa empleados
dn: ou=empleados, dc=banco, dc=org
objectClass: organizationalUnit
ou: empleados
```

Imagen 3.8. Fichero .ldif de creación de unidad organizativa

A través de la consola comprobamos que la creación del unidad organizativa se ha realizado correctamente:

```
dn: ou=empleados,dc=banco,dc=org
objectClass: organizationalUnit
ou: empleados
structuralObjectClass: organizationalUnit
entryUUID: 42fa05f4-2a86-103b-9a4f-27d7d5efc4bb
creatorsName: cn=admin,dc=banco,dc=org
createTimestamp: 20210405181243Z
entryCSN: 20210405181243.858359Z#000000#000#000000
modifiersName: cn=admin,dc=banco,dc=org
modifyTimestamp: 20210405181243Z
```

Imagen 3.9. Comprobación de creación de unidad organizativa

Una vez definida la unidad organizativa, ya podemos asignar al LDAP los usuarios empleados que se van a poder registrar en las aplicaciones disponibles. Volvemos a hacer uso de un fichero *".ldif"* para indicar los usuarios que se registrarán en el LDAP (se muestra una parte del fichero):

```
dn: uid=joserejon, ou=empleados, dc=banco, dc=org
objectClass: inetOrgPerson
uid: joserejon
cn: Jose
sn: Rejon
title: Supervisor
userPassword: {SHA}xkJJ5Jc6YmAuCi5+Iq7XsTVqycA=
mail: joserejon@banco.com

dn: uid=pacoperez, ou=empleados, dc=banco, dc=org
objectClass: inetOrgPerson
uid: pacoperez
cn: Paco
sn: Perez
title: Editor
userPassword: {SHA}Mx1tMkAVPWT1H/i3ftnBwtkN01Y=
mail: pacoperez@banco.com
```

Imagen 3.10. Fichero .ldif de creación de usuarios empleados

Y de nuevo, a través de la consola comprobamos que la creación de los usuarios se ha realizado correctamente:

```
dn: uid=joserejon,ou=empleados,dc=banco,dc=org
objectClass: inetOrgPerson
uid: joserejon
cn: Jose
sn: Rejon
title: Supervisor
userPassword:: e1NIQX14a0pKNUpjNlltQXVDaTUrSXE3WHNUVnF5Y0E9
mail: joserejon@banco.com
structuralObjectClass: inetOrgPerson
entryUUID: 882aac4e-2a88-103b-9a50-27d7d5efc4bb
creatorsName: cn=admin,dc=banco,dc=org
createTimestamp: 20210405182858Z
entryCSN: 20210405182858.934551Z#000000#000#000000
modifiersName: cn=admin,dc=banco,dc=org
modifyTimestamp: 20210405182858Z

dn: uid=pacoperez,ou=empleados,dc=banco,dc=org
objectClass: inetOrgPerson
uid: pacoperez
cn: Paco
sn: Perez
title: Editor
userPassword:: e1NIQX1NeDF0TWtBVlBXVDFIL2kzZnRuQnd0a04wMVk9
mail: pacoperez@banco.com
structuralObjectClass: inetOrgPerson
entryUUID: 882c5dbe-2a88-103b-9a51-27d7d5efc4bb
creatorsName: cn=admin,dc=banco,dc=org
createTimestamp: 20210405182858Z
entryCSN: 20210405182858.945645Z#000000#000#000000
modifiersName: cn=admin,dc=banco,dc=org
modifyTimestamp: 20210405182858Z
```

Imagen 3.11. Comprobación de creación de usuarios empleados

Tras realizar estos pasos, tendremos en LDAP una arquitectura de empleados similar a la del siguiente esquema:

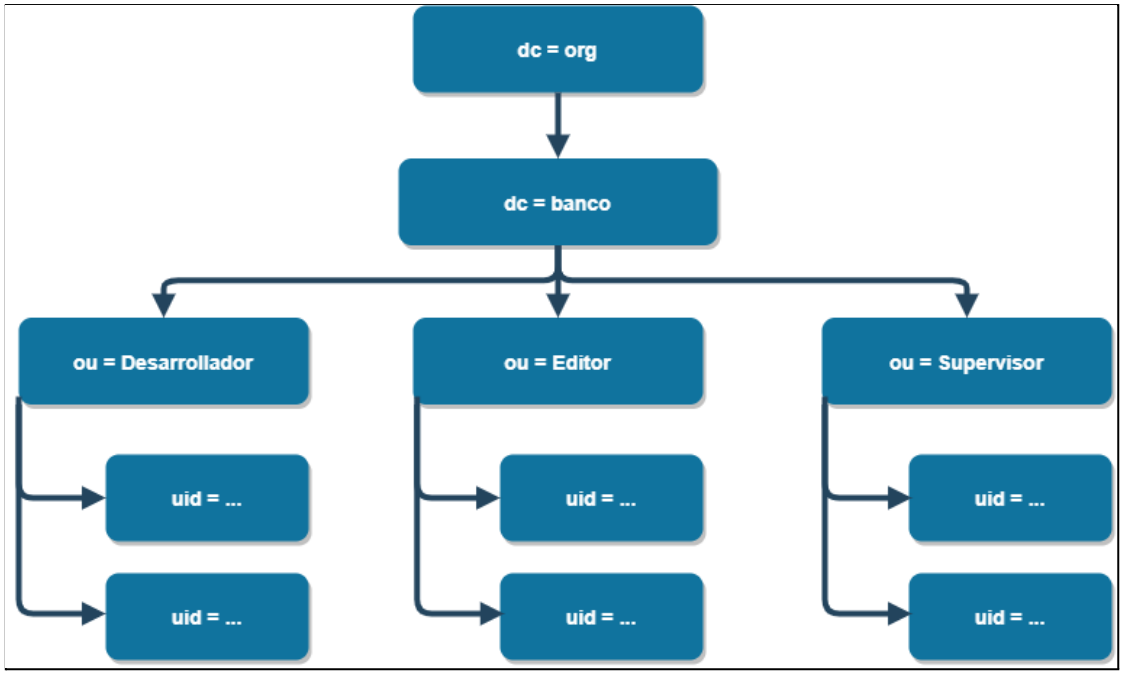

Imagen 3.12. Arquitectura del LDAP

Observamos de la imagen anterior que dentro del dominio *"banco.org"* tenemos tres perfiles diferenciados, que son los que ya se mencionaban en un punto anterior de la memoria y que son los siguientes:

- **Desarrollador:** se trata del rol base (o rol más bajo). Permite el acceso a las herramientas de desarrollo, formación, documentación o registro de jornada.
- **Editor:** además de las acciones del Desarrollador, podrá subir nueva documentación al sistema.
- **Supervisor:** engloba los roles anteriores y le permite subir el código a Producción.

3.3 Configuración de la red

Será necesario establecer una serie de comunicaciones entre los distintos equipos que conforman la estructura de la empresa. Para ello, desde VirtualBox tendremos que establecer las configuraciones oportunas para que los equipos se hablen entre sí.

La máquina *webserver* contará con dos adaptadores de red para conectarse a Internet (adaptador puente) y a la Red Interna (adaptador red interna):

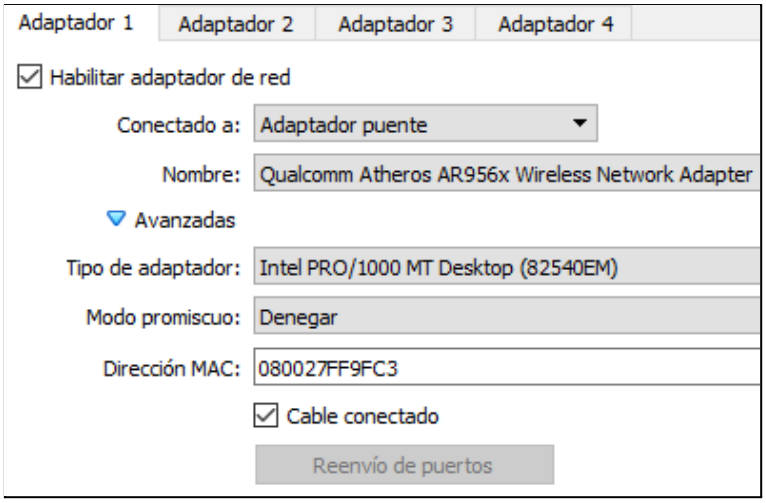

Imagen 3.13. Configuración adaptador 1 de webserver

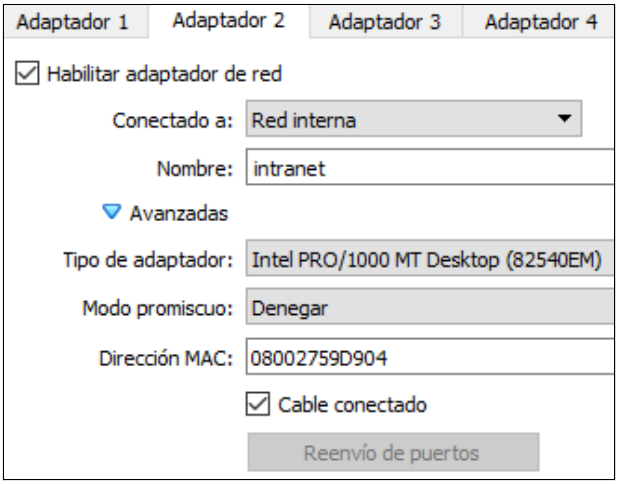

Imagen 3.14. Configuración adaptador 2 de webserver

El resto de máquinas (*ssoserver1, ssoserver2, appserver, dataserver*) contarán con un solo adaptador de Red Interna (adaptador red interna):

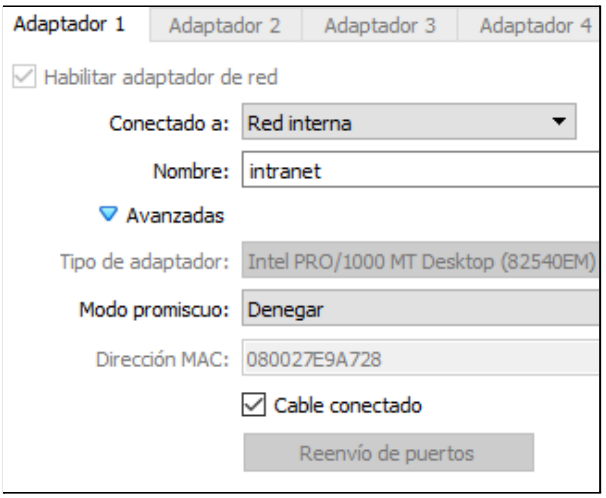

Imagen 3.15. Configuración adaptador 1 de ssoserver1, ssoserver2, appserver, dataserver

De este modo tendremos la posibilidad de tener una red aislada en red interna, que hemos llamado intranet, que es donde se desplegarán los servidores que no han de estar expuestos a internet. Por otro lado, nos queda una red que sí lo estará (DMZ) que es donde colocamos nuestro servidor web (*webserver*) para canalizar las peticiones que venga desde internet. La arquitectura del proyecto con las redes ya definidas quedaría de la siguiente forma:

- **WebServer:**
  - Adaptador puente: *DMZ - 192.168.1.173*
  - Adaptador red interna: *Intranet - 10.10.0.2*
- **SSOServer1:**
  - Adaptador red interna: *Intranet - 10.10.0.5*
- **SSOServer2:**
- Adaptador red interna: *Intranet - 10.10.0.6*
- **AppServer:**
  - Adaptador red interna: *Intranet - 10.10.0.3*
- **DataServer:**
  - Adaptador red interna: *Intranet - 10.10.0.4*

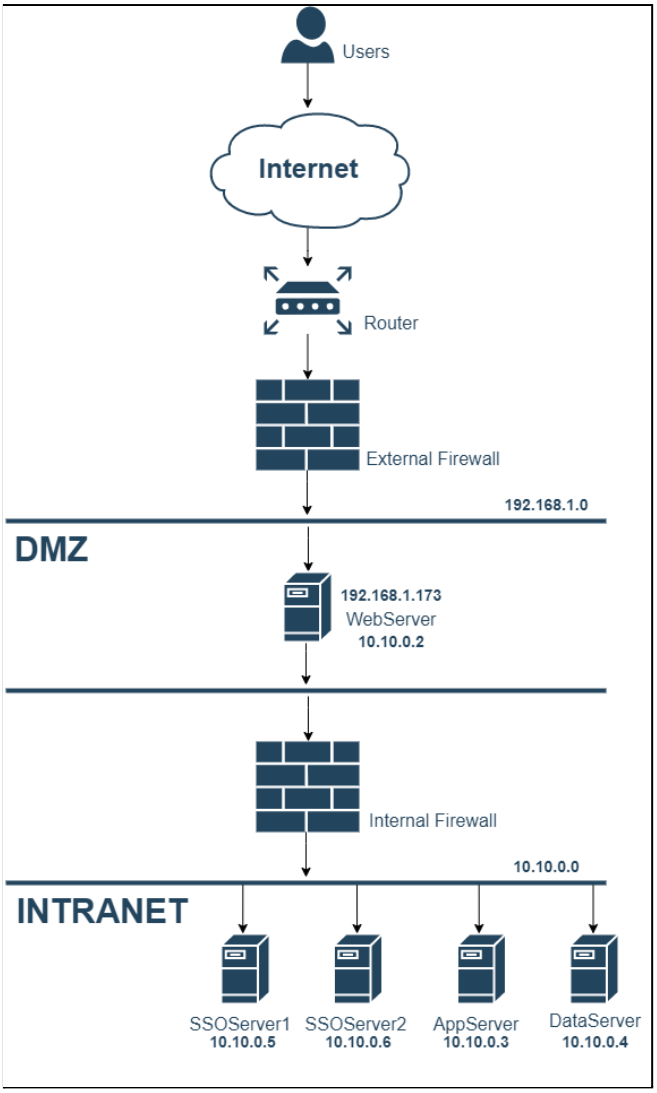

Imagen 3.16. Arquitectura del proyecto actualizada con direcciones IP

3.4 Integración SSO

En este punto vamos a llevar a cabo la integración del servidor de SSO[36,37,38,39] con las aplicaciones desarrolladas que queremos securizar. Este apartado es amplio, ya que requiere de distintas configuraciones que iremos haciendo de forma progresiva, por lo que dividiremos este punto en diferentes subapartados con los que lograremos un hito diferente.

3.4.1 Primera interacción

La primera interacción la hemos realizado entre la aplicación *Escritorio Único*, ubicada en la máquina virtual *appserver* y el servidor **CAS** ubicado en la máquina virtual *ssoserver1*.

Para ello, ha sido necesario modificar el fichero *"web.xml"* (de la aplicación de *Escritorio Único*) para conectar con el servidor CAS. Este fichero se ha editado para incluir la configuración necesaria para hacer uso del framework Spring, como indicar la ruta al fichero *"applicationContext-security.xml"* de configuración de Spring, añadir el listener que inicializa el contexto, etc:

```
<display-name>Escritorio Único</display-name>

<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>
        /WEB-INF/applicationContext-security.xml
    </param-value>
</context-param>

<context-param>
    <param-name>webAppRootKey</param-name>
    <param-value>tutorial.root</param-value>
</context-param>

<filter>
    <filter-name>springSecurityFilterChain</filter-name>
    <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
</filter>

<filter-mapping>
  <filter-name>springSecurityFilterChain</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>

<listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>

<listener>
  <listener-class>org.springframework.security.web.session.HttpSessionEventPublisher</listener-class>
</listener>

 <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
```

Imagen 3.17. Configuración del fichero fichero web.xml

Después tenemos que crear y configurar el fichero "*applicationContext-security.xml*" (de la aplicación de *Escritorio Único*). En este fichero se establecen las configuraciones de Spring tales como los accesos a los recursos, los roles de acceso, así como el ***casAuthenticationEntryPoint***, que indica a qué url se tiene que redirigir para hacer el login cuando se intenta acceder a un recurso protegido. Para esta primera interacción, indicaremos en *casAuthenticationEntryPoint* la url a la cual se debe acceder para hacer login. Como vemos en el siguiente pantallazo, le indicamos que vaya a *http://10.10.0.5:8080/cas-server-webapp-4.0.0*, que es la dirección del servidor CAS de la máquina virtual *ssoserver1*:

```
<bean id="casAuthenticationEntryPoint" class="org.springframework.security.cas.web.CasAuthenticationEntryPoint">
    <property name="loginUrl" value="http://10.10.0.5:8080/cas-server-webapp-4.0.0/login"></property>
    <property name="serviceProperties" ref="serviceProperties"></property>
</bean>
```

Imagen 3.18. Configuración dirección login CAS Server

Como recurso protegido y de forma temporal, tenemos el interior de la carpeta ***secure*** y bajo un solo rol, ya que estamos haciendo la primera interacción con el usuario de ejemplo de CAS Server:

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

Imagen 3.19. Configuración roles

Llegados a este punto, con esta mínima configuración, tenemos el acceso restringido a la aplicación *Escritorio Único* y solo el usuario por defecto de CAS Server tiene acceso a ella.

Repetimos este proceso para las aplicaciones *Herramientas de desarrolladores* y *Formación* para estar empatados en esta primera toma de contacto en la integración del servidor SSO con las aplicaciones.

Es importante resaltar que en estos momentos estamos trabajando bajo el protocolo HTTP y no HTTPS. Es importante tenerlo en cuenta dado que la cookie **CASTGC** por defecto viene configurada como *cookieSecure=true*, por lo que no se enviará si no estamos usando HTTPS. Para las pruebas en este punto hemos tenido que modificar el valor y establecerlo a *false* para que se envíe por HTTP. Esta configuración se realiza en el fichero *"WEB-INF/spring-configuration/ticketGrantingTicketCookieGenerator.xml"* de los servidores CAS:

```
<bean id="ticketGrantingTicketCookieGenerator" class="org.jasig.cas.web.support.CookieRetrievingCookieGenerator"
        p:cookieSecure="false"
        p:cookieMaxAge="-1"
        p:cookieName="CASTGC"
        p:cookiePath="/cas" />
```

Imagen 3.20. Configuración cookieSecure

3.4.2 Integración CAS y OpenLDAP

Como segundo hito en el apartado de la integración del sistema de Single Sign-On afrontaremos la unión de **CAS Server** y **OpenLDAP**[42,43,44] para realizar la autenticación. Se busca que el servidor CAS (*ssoserver1* y *ssoserver2*) se conecte al servidor OpenLDAP (*dataserver*) para consultar las credenciales de los usuarios y realizar la autenticación en el sistema.

El primer paso para que el servidor CAS interactúe con el servidor OpenLDAP es añadir las librerías *"cas-server-support-ldap-4.0.0.jar"* y *"ldaptive-1.1.0.jar"* en el del directorio del servidor CAS "*/opt/tomcat/webapps/cas-server-webapp-4.0.0/WEB-INF/lib*".

Para agregar las dependencias LDAP en el servidor CAS, es necesario modificar el fichero *"/opt/tomcat/webapps/cas-server-webapp-4.0.0/META-INF/maven/org.jasig.cas/cas-server-webapp/**pom.xml**"* añadiendo el siguiente código:

```
<parent>
  <groupId>org.jasig.cas</groupId>
  <artifactId>cas-server</artifactId>
  <version>4.0.0</version>
</parent>
<modelVersion>4.0.0</modelVersion>
<artifactId>cas-server-webapp</artifactId>
<packaging>war</packaging>
<name>Jasig CAS Web Application</name>
<dependencies>
  <dependency>
      <groupId>org.jasig.cas</groupId>
      <artifactId>cas-server-support-ldap</artifactId>
  <version>${cas.version}</version>
  </dependency>
```

Imagen 3.21. Agregar dependencias LDAP en servidor CAS

Una vez añadidas las librerías y establecida la dependencia en *pom.xml*, toca proceder a modificar el fichero *"deployerConfigContext.xml"*, ubicado en

"*/opt/tomcat/webapps/cas-server-webapp-4.0.0/WEB-INF/*", que es uno de los ficheros de configuración del CAS.

Se procede a modificar en el fichero *"deployerConfigContext.xml"* la llamada que se realiza a primaryAuthenticationHandler, que tiene la autenticación por defecto:

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

Imagen 3.22. Bean primaryAuthenticationHandler

Y se sustituye por el bean **ldapAuthenticationHandler**:

```
<bean id="ldapAuthenticationHandler"
    class="org.jasig.cas.authentication.LdapAuthenticationHandler"
    p:principalIdAttribute="uid"
    c:authenticator-ref="authenticator">
    <property name="principalAttributeMap">
    <map>
        <entry key="uid" value="uid" />
        <entry key="cn" value="cn" />
    </map>
    </property>
</bean>
```

Imagen 3.23. Bean ldapAuthenticationHandler

Adicionalmente se incluyen una serie de **bean's** adicionales que se pueden consultar en el anexo 8.7 Bean's del fichero deployerConfigContext.xml.

Para finalizar con la integración es necesario modificar el fichero ***cas.properties***, el cual se encuentra ubicado en el directorio *"/opt/tomcat/webapps/cas-server-webapp-4.0.0/WEB-INF/"*, añadiendo una serie de valores que son los que se cargarán en la configuración establecida en el fichero *"deployerConfigContext.xml"*. Algunos de estos valores son (entre otros):

- **ldap.url** → Establecemos la url del servidor donde está instalado el directorio ldap. En este caso sería *ldap://10.10.0.4*.
- **ldap.authn.baseDn** → Indicamos dónde se ubican los usuarios que hay que buscar. En este caso indicamos que busque en ***ou=empleados,dc=banco,dc=org***.
- **ldap.authn.managerPassword** → Indicamos la contraseña del usuario administrador de OpenLDAP para hacer las peticiones.

3.4.3 WebServer

Es turno de realizar las configuraciones en el servidor web Apache HTTP, ubicado en la máquina virtual *webserver*.

Modificar fichero hosts[46]

La primera modificación que vamos a realizar es la de modificar el fichero *"/etc/hosts"*. Se trata de un archivo de texto que almacena nombres de máquinas (hosts) y su correspondiente IP:

```
127.0.0.1 localhost
127.0.1.1 webserver

10.10.0.3 appserver
10.10.0.4 dataserver
10.10.0.5 ssoserver1
10.10.0.6 ssoserver2
```

Imagen 3.24. Configuración fichero /etc/hosts

Como vemos en la imagen anterior, se han añadido las direcciones IP de las máquinas virtuales que forman la estructura de nuestra organización y se les ha otorgado un nombre. Gracias a esto, en cierta manera podemos decir que tenemos una especie de servidor DNS pero a nivel interno de la máquina webserver. De este modo podemos llamar a una máquina por su nombre, como por ejemplo vemos en la siguiente captura para hacer un ***ping***:

```
useradmin@webserver:~$ ping ssoserver1
PING ssoserver1 (10.10.0.5) 56(84) bytes of data.
64 bytes from ssoserver1 (10.10.0.5): icmp_seq=1 ttl=64 time=2.08 ms
64 bytes from ssoserver1 (10.10.0.5): icmp_seq=2 ttl=64 time=0.798 ms
64 bytes from ssoserver1 (10.10.0.5): icmp_seq=3 ttl=64 time=0.879 ms
^C
--- ssoserver1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 0.798/1.252/2.080/0.586 ms
```

Imagen 3.25. Ejemplo utilidad /etc/hosts

Configurar Proxy Inverso

Un proxy inverso[47,48] es un tipo de servidor que recupera recursos en nombre de un cliente, siendo estos recursos devueltos al cliente como si fuera el propio servidor proxy quien los ha originado. Esto permite ocultar toda la red detrás del servidor proxy ya que, lo único de lo que se tiene visibilidad desde el exterior es de dicho servidor, quedando apantallado todo lo que exista detrás.

Disponer de un proxy inverso nos aportará seguridad, alta disponibilidad (entre *ssoserver1* y *ssoserver2*), balanceo de carga y centralización de la autenticación/autorización.

Para realizar esta configuración tenemos que editar el fichero "*/etc/apache2/sites-available/000-default.conf*" del servidor *webserver*. En la siguiente imagen se muestra como quedaría el fichero:

```
ServerName webserver
<VirtualHost *:80>

    #Máquina appserver - Aplicación Escritorio Único
    ProxyPass /EscritorioUnico http://appserver:8080/EscritorioUnico
    ProxyPassReverse /EscritorioUnico http://appserver:8080/EscritorioUnico

    #Máquina appserver - Aplicación Herramientas de desarrolladores
    ProxyPass /HerramientasDesarrolladores http://appserver:8080/HerramientasDesarrolladores
    ProxyPassReverse /HerramientasDesarrolladores http://appserver:8080/HerramientasDesarrolladores

    #Máquina appserver - Aplicación Formación
    ProxyPass /Formacion http://appserver:8080/Formacion
    ProxyPassReverse /Formacion http://appserver:8080/Formacion


    ProxyPass /cas-server-webapp-4.0.0 balancer://ssoset
    ProxyPassReverse /cas-server-webapp-4.0.0 balancer://ssoset

    #Balanceador
    <Proxy "balancer://ssoset">
        BalancerMember "http://ssoserver1:8080/cas-server-webapp-4.0.0"
        BalancerMember "http://ssoserver2:8080/cas-server-webapp-4.0.0"
    </Proxy>

</VirtualHost>
```

Imagen 3.26. Configuración del fichero 000-default.conf

Como podemos comprobar del pantallazo anterior, se establecen las reglas para los distintos servidores que forman la plataforma.

Con la directiva **ProxyPass** se especifica el mapeo de las peticiones entrantes a los servidores backend (*EscritorioUnico*, HerramientasDesarrolladores, Formacion). También hacemos uso de la directiva **ProxyPassReverse** para asegurarnos de que las cabeceras *location* generadas en los backend's apuntan al proxy inverso en vez de a los propios backend's.

Para cerrar el bloque **VirtualHost** y con ello la configuración de este fichero, incluimos las directivas *Proxy balancer* y *BalancerMember*. Con el esquema *balancer://* le decimos a *httpd* que estamos creando un grupo de balanceo llamado *ssoset*, el cual incluye los dos servidores de SSO que disponemos: *ssoserver1* y *ssoserver2*.

Configurar appserver

Para que la configuración anterior tenga sentido es necesario realizar unos sencillos cambios en el fichero *"applicationContext-security.xml"* de los tres programas que hemos desarrollado.

En el siguiente pantallazo vemos que para las directivas *casAuthenticationEntryPoint* y *casAuthenticationProvider* se indican las url's con el dominio *webserver*. Esto nos permitirá el balanceo propuesto en la definición del servidor web ya que no estamos especificando a qué servidor de SSO estamos llamando:

```
<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>

<!--
    Handles the CAS ticket processing.
 -->
<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>
```

Imagen 3.27. Configuración del fichero applicationContext-security.xml

3.4.4 Protección de los recursos

Protección a nivel de rol

Aplicamos el perfilado a nivel de grano fino, de modo que los accesos a los recursos estén protegidos por los roles de los usuarios. Como se indicaba en la fase de análisis, disponemos de tres roles distintos:

- **Desarrollador:** se trata del rol base (o rol más bajo). Permite el acceso a las herramientas de desarrollo, formación, documentación o registro de jornada. Lo declaramos como ***ROLE_DESARROLLADOR***.
- **Editor:** además de las acciones del Desarrollador, podrá subir nueva documentación al sistema. Lo declaramos como ***ROLE_EDITOR***.
- **Supervisor:** engloba los roles anteriores y le permite subir el código a Producción. Lo declaramos como ***ROLE_SUPERVISOR***.

Ubicados en el fichero de configuración de Spring *"applicationContext-security.xml"* aplicamos las reglas de acceso mediante el filtro ***casAuthenticationEntryPoint***. Desde esta parte del código y a través de los ***security:intercept-url*** indicamos qué recurso (url) se quiere proteger y a qué roles se les permite acceder. Es importante resaltar que el orden en el que se introducen las restricciones juega un papel muy importante, ya que si la restricción más general se pone en primer lugar, de nada sirven el resto. Por lo tanto, las sentencias más restrictivas deben ir en primer lugar.

Por ejemplo, para la aplicación *Escritorio Único* tendremos unas restricciones tal que así:

```
<security:http entry-point-ref="casAuthenticationEntryPoint" auto-config="true">
    <security:intercept-url pattern="/secure/aplicacionesInternas/addDoc.jsp" access="ROLE_SUPERVISOR, ROLE_EDITOR"></security:intercept-url>
    <security:intercept-url pattern="/secure/**" access="ROLE_SUPERVISOR, ROLE_EDITOR, ROLE_DESARROLLADOR"></security:intercept-url>
    <security:custom-filter position="CAS_FILTER" ref="casAuthenticationFilter"></security:custom-filter>
</security:http>
```

Imagen 3.28. Escritorio Único - Acceso a recursos según roles

Con estas restricciones le estamos diciendo a la aplicación dos cosas:

1. Que solo los roles *ROLE_SUPERVISOR* y *ROLE_EDITOR* tienen permisos para acceder al documento *"addDoc.jsp"*, es decir, a la capacidad de subir nueva documentación.

2. Que todo lo que esté dentro del directorio *secure* estará restringido para los tres roles definidos, o dicho de otra forma, todos los recursos dentro de *secure* solamente serán accesibles para un usuario autenticado (ya que no disponemos de otro cuarto rol).

Para la aplicación *Formación* tendremos estas restricciones:

```
<security:http entry-point-ref="casAuthenticationEntryPoint" auto-config="true">
    <security:intercept-url pattern="/secure/formaciones/for_promocion_ent.jsp" access="ROLE_SUPERVISOR"></security:intercept-url>
    <security:intercept-url pattern="/secure/**" access="ROLE_SUPERVISOR, ROLE_EDITOR, ROLE_DESARROLLADOR"></security:intercept-url>
    <security:custom-filter position="CAS_FILTER" ref="casAuthenticationFilter"></security:custom-filter>
</security:http>
```

Imagen 3.29. Formación - Acceso a recursos según roles

Con estas restricciones le estamos diciendo a la aplicación dos cosas:

1. Que solo el rol *ROLE_SUPERVISO*R tiene permisos para acceder a la formación de promoción de código entre entornos.
2. Que todo lo que esté dentro del directorio *secure* estará restringido para los tres roles definidos, o dicho de otra forma, todos los recursos dentro de *secure* solamente serán accesibles para un usuario autenticado (ya que no disponemos de otro cuarto rol).

Y por último para la aplicación *Herramientas de desarrolladores* indicamos lo siguiente:

```
<security:http entry-point-ref="casAuthenticationEntryPoint" auto-config="true">
    <security:intercept-url pattern="/secure/herramientasDespliegue/**" access="ROLE_SUPERVISOR"></security:intercept-url>
    <security:intercept-url pattern="/secure/**" access="ROLE_SUPERVISOR, ROLE_EDITOR, ROLE_DESARROLLADOR"></security:intercept-url>
    <security:custom-filter position="CAS_FILTER" ref="casAuthenticationFilter"></security:custom-filter>
</security:http>
```

Imagen 3.30. Herramientas de desarrollo - Acceso a recursos según roles

Con estas restricciones le estamos diciendo a la aplicación dos cosas:

1. Que solo el rol *ROLE_SUPERVISO*R tiene permisos para acceder a las herramientas de promoción de código entre entornos.
2. Que todo lo que esté dentro del directorio *secure* estará restringido para los tres roles definidos, o dicho de otra forma, todos los recursos dentro de *secure* solamente serán accesibles para un usuario autenticado (ya que no disponemos de otro cuarto rol).

Protección a nivel de FrontEnd

En el apartado anterior hemos establecido la protección de recursos a nivel de roles de usuarios, ahora lo que vamos a hacer es ocultar los recursos a nivel de vista de las aplicaciones (basándonos en los roles) para que los usuarios que no tengan permisos no vean los recursos que no pueden acceder. Para restringir la vista por roles lo hacemos con una llamada al método ***isUserInRole(rol)*** de Spring, con el que le pasamos por parámetro el rol que queremos comprobar.

A continuación vemos un ejemplo de cómo se usa esta restricción para mostrar las herramientas de despliegue de código entre entornos:

```
<%if (request.isUserInRole("ROLE_SUPERVISOR")) { %>
<h3>Herramientas despligue</h3></td>
<p><a href="herramientasDespliegue/produccion.jsp">Producción</a></p>
<p><a href="herramientasDespliegue/integrado.jsp">Integrado</a></p>
<% } %>
```

Imagen 3.31. Ejemplo de uso de isUserInRole(rol)

3.5 Securización de las comunicaciones

Proteger las comunicaciones en una plataforma web es una necesidad muy importante, sobretodo cuando estamos tratando datos sensibles/personales de los usuarios que hacen uso de la aplicación.

En nuestro caso estamos hablando de aplicaciones que usan los empleados de un banco y las cuales contienen información confidencial de la empresa, así como datos de los empleados como el nombre, apellidos, contraseñas, etc.

Hasta ahora venimos trabajando en el portal con el protocolo HTTP. Este protocolo permite transferencias de información vía web y se basa en el esquema petición-respuesta entre cliente y servidor. Queremos dotar a estas comunicaciones de privacidad, seguridad y protección de datos, por lo que tendremos que evolucionar este protocolo a **HTTPS**. El protocolo **HTTPS**[49] es una variante del protocolo HTTP, al que se le añade una capa extra de seguridad al encriptar los datos a través de **SSL/TLS**[50].

Gracias a esto, las comunicaciones son más seguras y evitamos que un atacante pueda interceptar la comunicación y capturar los datos que se intercambian.

Una de las cosas que tenemos que tener en cuenta para securizar la comunicación es que necesitamos disponer de un certificado SSL/TLS para ello. El certificado nos aporta:

1. Cifrado de la información en la transmisión.
2. Autenticar al sitio web que expone la información, de modo que el cliente sepa que no se trata de un sitio falso.

Aterrizando esto al proyecto, queremos securizar todas las comunicaciones entre el navegador del cliente, el servidor web (*webserver*), las aplicaciones CASificadas (*appserver*), los servidores CAS (*ssoserver1* y *ssoserver2*) y el servidor OpenLAP (*dataserver*).

NOTA: este apartado (*3.5 Securización de las comunicaciones*) no se ha conseguido implementar debido a una serie de fallos que no ha dado tiempo a corregir. Se incluye en la memoria algunos de los pasos dados para dejar constancia de la intención y los esfuerzos realizados para esta parte del proyecto. Por lo tanto, el proyecto funcionará sobre **HTTP** sin SSL/TLS.

3.5.1 Generación de certificados

Para la generación de certificados vamos a hacer uso de las dos herramientas siguientes:

- **OpenSSL**[51]**:** se trata de un paquete de herramientas de administración y bibliotecas de criptografía que ofrece funciones criptográficas a navegadores web (o a otros paquetes) para securizar el protocolo HTTP.

- **Keytool[52]:** administra un keystore de claves criptográficas, cadenas de certificados X.509 y certificados de confianza.

El primer certificado que se ha creado ha sido el de nuestra propia CA (Certification Authority) ya que haremos uso de certificados autofirmados. En la siguiente imagen se puede apreciar cómo se crean tanto la **clave privada** (*rootCA.key*) como la **parte pública** (*rootCA.pem*) haciendo uso de la herramienta OpenSSL:

```
useradmin@webserver:/etc/apache2/certssl$ sudo openssl genrsa -out rootCA.key
Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
..+++++
e is 65537 (0x010001)
useradmin@webserver:/etc/apache2/certssl$ sudo openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 730 -out rootCA.pem
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Granada
Locality Name (eg, city) []:Granada
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Banco
Organizational Unit Name (eg, section) []:Banco
Common Name (e.g. server FQDN or YOUR name) []:Banco
Email Address []:
```

Imagen 3.32. Generación certificado CA

Generamos ahora el certificado para el servidor web (*webserver*) y para ello tenemos que comenzar generando la parte privada del certificado (*server_cer.key*) y posteriormente el fichero *.csr*, que es el fichero Certificate Signing Request (Solicitud de Firma de Certificado):

```
useradmin@webserver:/etc/apache2/certssl$ sudo openssl genrsa -out server_cert.key
Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
.....+++++
e is 65537 (0x010001)
useradmin@webserver:/etc/apache2/certssl$ sudo openssl req -new -key server_cert.key -out server_cert.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Granada
Locality Name (eg, city) []:Granada
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Banco
Organizational Unit Name (eg, section) []:Banco
Common Name (e.g. server FQDN or YOUR name) []:Banco
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:serverpasscert
An optional company name []:
```

Imagen 3.33. Generación certificado webserver

Para finalizar con el certificado del servidor, vemos en la siguiente imagen como se genera el certificado para *webserver* (*server_cert.crt*), el cual ha sido firmado con la CA que se ha definido al principio:

```
useradmin@webserver:/etc/apache2/certssl$ sudo openssl x509 -req -in server_cert.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateseri
al -out server_cert.crt
Signature ok
subject=C = ES, ST = Granada, L = Granada, O = Banco, OU = Banco, CN = Banco
Getting CA Private Key
```

Imagen 3.34. Generación certificado webserver

3.5.2 Configuración del servidor web

Tras la generación de los certificados tenemos que realizar ciertas configuraciones[53] en el servidor web. Para ello comenzamos modificando el archivo de configuración "*/etc/apache2/sites-available/000-default.conf*".

Dentro de este documento vamos a modificar el bloque de configuración VirtualHost para redirigir el tráfico del puerto 80 al 443, que es el puerto habilitado para SSL/TLS. Además de hacer la redirección mediante la directiva **Redirect**, tenemos que indicar al fichero la ruta en la que tiene que buscar los certificados, de modo que el fichero resultaría tal que así:

```
ServerName webserver

<VirtualHost *:80>
    Redirect / https://webserver/
</VirtualHost>

<VirtualHost *:443>

    SSLEngine on

    SSLCertificateFile  /etc/apache2/certssl/server_cert.crt
    SSLCertificateKeyFile   /etc/apache2/certssl/server_cert.key
```

Imagen 3.35. Configuración del fichero 000-default.conf

Como tenemos habilitado el firewall ufw, tenemos que revisar la configuración para permitir el tráfico SSL. Con el comando **sudo ufw status** vemos la siguiente salida:

```
useradmin@webserver:/etc/apache2/certssl$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
Apache                     ALLOW       Anywhere
Apache Full                ALLOW       Anywhere
Apache (v6)                ALLOW       Anywhere (v6)
Apache Full (v6)           ALLOW       Anywhere (v6)
```

Imagen 3.36. Estado del firewall

Vemos que al final del pantallazo tenemos la entrada HTTPS habilitada, **Apache Full**, por lo que a nivel de firewall no tendríamos que introducir ninguna configuración nueva.

3.5.3 Configuración de los servidores de SSO

Comenzamos creando una nueva CA (como en el apartado anterior) para poder generar las Claves, certificados y Key Store para Tomcat. Una vez creada, es el turno de crear el Key Store, donde se almacenará la parte privada del Tomcat y la parte pública de la CA que se usarán para cifrar las conexiones del Tomcat. En la imagen se puede ver cómo se genera el Key Store con la herramienta keytool:

```
useradmin@ssoserver1:/opt/tomcat/certssl$ sudo keytool -genkey -alias KS_SSO -keyalg RSA -keystore ssocert.jks
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  ssoserver
What is the name of your organizational unit?
  [Unknown]:  banco
What is the name of your organization?
  [Unknown]:  banco
What is the name of your City or Locality?
  [Unknown]:  Granada
What is the name of your State or Province?
  [Unknown]:  Granada
What is the two-letter country code for this unit?
  [Unknown]:  ES
Is CN=ssoserver, OU=banco, O=banco, L=Granada, ST=Granada, C=ES correct?
  [no]:  yes
```

Imagen 3.37. Generación del Key Store

Posteriormente se crea el fichero de solicitud de certificado para tomcat:

```
useradmin@ssoserver1:/opt/tomcat/certssl$ sudo keytool -certreq -keyalg RSA -alias KS_SSO -file ssocert.csr -keystore ssocert.jks
Enter keystore password:
```

Imagen 3.38. Solicitud de certificado

Y se finaliza la generación de certificados creando el certificado de Tomcat, el cual también ha sido firmado con la CA que se ha definido unos pasos atrás:

```
useradmin@ssoserver1:/opt/tomcat/certssl$ sudo openssl x509 -req -in ssocert.csr -CA ssoCA.pem -CAkey ssoCA.key -CAcreateserial -o
ut ssocert.crt -days 730 -sha256
Signature ok
subject=C = ES, ST = Granada, L = Granada, O = banco, OU = banco, CN = ssoserver
Getting CA Private Key
```

Imagen 3.39. Generación del certificado

Importamos primero la parte pública de la CA al Key Store para luego poder importar el certificado del Tomcat con el comando ***keytool -import -alias SSO_CA -keystore ssocert.jks -trustcacerts -file ssoCA.pem***, que nos solicitará la password del Key Store.

Después, se importa el certificado de Tomcat también con el comando ***keytool -import -alias KS_SSO -keystore ssocert.jks -file ssocert.crt***, solicitando una vez más la password del Key Store:

```
useradmin@ssoserver1:/opt/tomcat/certssl$ sudo keytool -import -alias SSO_CA -keystore ssocert.jks -trustcacerts -file ssoCA.pem
Enter keystore password:
Owner: CN=Banco, OU=Banco, O=Banco, L=Granada, ST=Granada, C=ES
Issuer: CN=Banco, OU=Banco, O=Banco, L=Granada, ST=Granada, C=ES
Serial number: 11da81faa73d7a0bde2379fe0d66915cf80fa6b4
Valid from: Sun Apr 18 11:46:41 UTC 2021 until: Tue Apr 18 11:46:41 UTC 2023
Certificate fingerprints:
```

Imagen 3.40. Importar parte pública de la CA al Key Store

```
useradmin@ssoserver1:/opt/tomcat/certssl$ sudo keytool -import -alias KS_SSO -keystore ssocert.jks -file ssocert.crt
Enter keystore password:
Certificate reply was installed in keystore
```

Imagen 3.41. Importar certificado de Tomcat

Por último se crea el Trust Store con el certificado de la CA tal que así: ***keytool -import -file ssoCA.pem -alias ssoCAcert -keystore truststore.jks***, que solicitará generar una password para proteger el acceso y preguntará si se confía en la CA adjunta.

```
useradmin@ssoserver1:/opt/tomcat/certssl$ sudo keytool -import -file ssoCA.pem -alias ssoCAcert -keystore truststoresso.jks
Enter keystore password:
Re-enter new password:
Owner: CN=Banco, OU=Banco, O=Banco, L=Granada, ST=Granada, C=ES
Issuer: CN=Banco, OU=Banco, O=Banco, L=Granada, ST=Granada, C=ES
Serial number: 11da81faa73d7a0bde2379fe0d66915cf80fa6b4
Valid from: Sun Apr 18 11:46:41 UTC 2021 until: Tue Apr 18 11:46:41 UTC 2023
Certificate fingerprints:
```

Imagen 3.42. Creación del Trust Store

Una vez tenemos el Key Store y Trust Store preparados, tenemos que configurar Tomcat para que vaya sobre SSL, teniéndose que modificar el fichero "*tomcat/conf/server.xml*" definiendo el conector tal que así:

```
<Connector
    port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
    maxThreads="150" secure="true" SSLEnabled="true" scheme="https"
    keystoreFile="/opt/tomcat/certssl/ssocert.jks" keystorePass="[redacted]"
    truststoreFile="/opt/tomcat/certssl/truststoresso.jks" truststorePass="[redacted]"
    clientAuth="false" sslProtocol="TLS" />
```

Imagen 3.43. Conector Tomcat HTTPS

También y muy importante, además de incluir el Trust Store junto con su password, es añadirlo al fichero "*tomcat/bin/catalina.sh*", sustituyendo el valor de la variable JAVA_OPTS por lo siguiente: ***JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.trustStore=opt/cert/truststore.jks -Djavax.net.ssl.trustStorePassword=truststorep"***

```
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.trustStore=opt/tomcat/certssl/truststoresso.jks -Djavax.net.ssl.trustStorePassword=[redacted]"
```

Imagen 3.44. Configurar variable JAVA_OPTS

Quedaría finalmente modificar el fichero "*applicationContext-security.xml*" y *"000-default"* para cambiar todas las llamadas http:// por **https://**, así como cambiar a los puertos correctos.

3.5.4 Configuración LDAP

Generamos la parte pública y privada de la CA y después se procede a generar el certificado para LDAP. Se genera primero la parte privada y se genera también un fichero .csr, que es el fichero de Certificate Signing Request (Solicitud de Firma de Certificado).

Para securizar la comunicación en LDAP, hay que incluir las rutas a los certificados (parte pública y privada del certificado de LDAP) y la ruta al certificado de la CA. Esta configuración se debe cargar en LDAP a través de un fichero *".ldif"*, que quedaría de la siguiente manera:

```
dn: cn=config
changetype: modify
replace: olcTLSCACertificateFile
olcTLSCACertificateFile: /etc/ldap/certssl/ldapCA.pem

dn: cn=config
changetype: modify
replace: olcTLSCertificateKeyFile
olcTLSCertificateKeyFile: /etc/ldap/certssl/ldapcert.key

dn: cn=config
changetype: modify
replace: olcTLSCertificateFile
olcTLSCertificateFile: /etc/ldap/certssl/ldapcert.crt
```

Imagen 3.45. Definir rutas certificados LDAP

El fichero definido se carga en LDAP a través del comando: ***ldapmodify -Y EXTERNAL -H ldapi:/// -f addPath.ldif***

```
useradmin@dataserver:/etc/ldap/certssl$ sudo ldapmodify -Y EXTERNAL -H ldapi:/// -f addPathCert.ldif
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
modifying entry "cn=config"
ldap_modify: Other (e.g., implementation specific) error (80)
```

Imagen 3.46. Cargar rutas certificados LDAP

4. Pruebas

4.1 SSO entre aplicaciones

Para hacer las pruebas tendremos que reproducir distintas casuísticas atendiendo a que los roles de los usuarios les van a permitir ver distintas capacidades.

Comenzamos la primera prueba accediendo a la aplicación *Escritorio Único* a través del enlace *http://webserver/EscritorioUnico* y, dado que la página principal es un recurso protegido, nos mostrará la ventana de login:

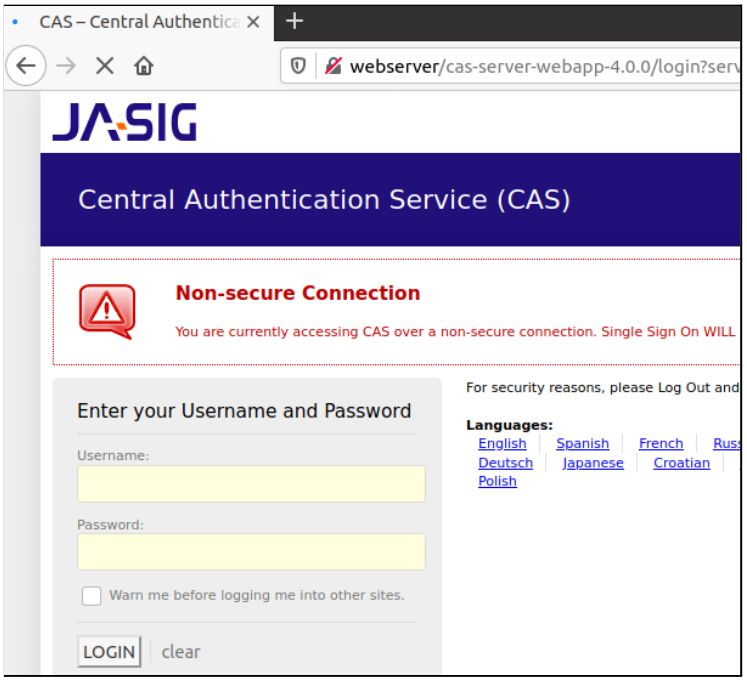

Imagen 4.1. Ventana de login

Accedemos con el usuario con el rol de supervisor (*ROLE_SUPERVISOR*) y nos aparece la ventana principal de la aplicación de *Escritorio Único*:

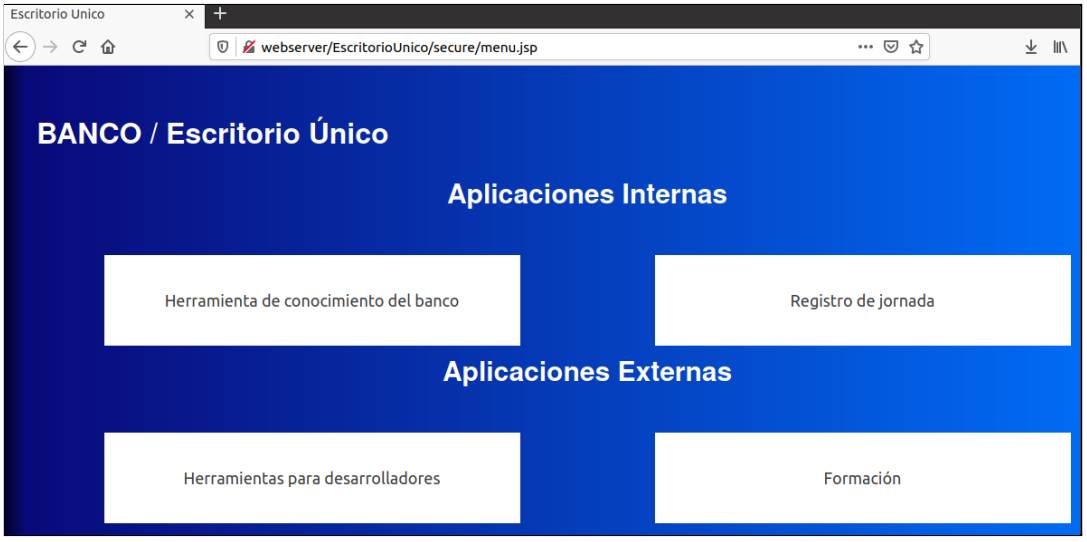

Imagen 4.2. Página principal de Escritorio Único

Vemos que se muestran 2 secciones diferentes, que son los accesos a aplicaciones internas y los accesos a aplicaciones externas. Las aplicaciones externas son aquellas que hemos configurado para que hagan SSO entre ellas, por lo que para probarlo accedemos a *Herramientas de desarrolladores*:

Imagen 4.3. Página principal de Herramientas de desarrolladores

Consultamos en el navegador la cookie **CASTGC,** que veíamos en uno de los apartados anteriores de la memoria que es la cookie que almacena el **TGT (Ticket Granting Ticket)**, o dicho de otra forma, la sesión SSO de un usuario, y se puede apreciar que tienen el mismo valor:

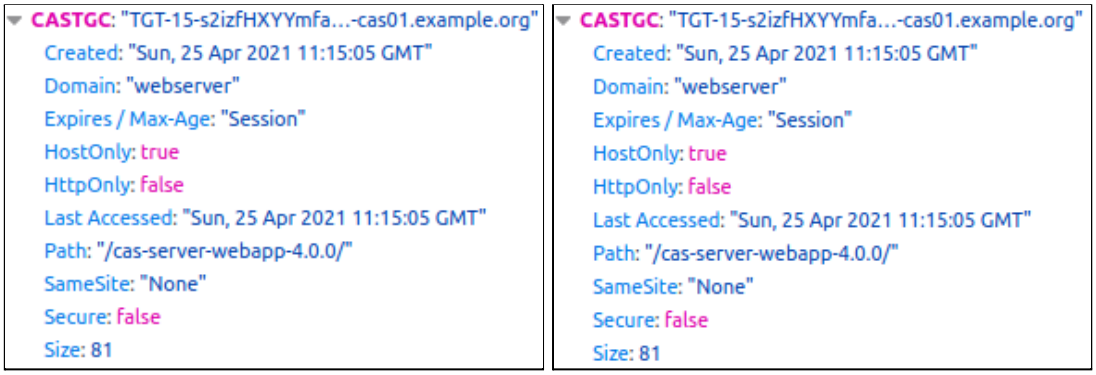

Imagen 4.4. Página principal de Herramientas de desarrolladores

Esta cookie nos ha dado acceso a la segunda aplicación sin necesidad de volver a hacer login, dado que CAS ha dado por válida la primera sesión realizada al acceder a *Escritorio Único*. Si accedemos a la herramienta de *Formación* sucede igual.

4.2 Acceso a recursos según permisos

Viendo que funciona el SSO, hacemos una prueba para ver que los recursos están correctamente protegidos según los roles. Para esta prueba accederemos con un usuario supervisor (*ROLE_SUPERVISOR*) y un usuario desarrollador (*ROLE_DESARROLLADOR*). En el caso del usuario supervisor, si accedemos a la aplicación *Herramientas de desarrolladores* nos encontramos con una sección llamada "Herramientas despliegue". Esta sección tiene acceso a los recursos que permitirían realizar los despliegue de códigos entre los dos entornos (producción e integrado) que se disponen en la empresa:

Imagen 4.5. Página principal de Herramientas de desarrolladores

Si accedemos por ejemplo al despliegue en producción vemos que el recurso es alcanzable:

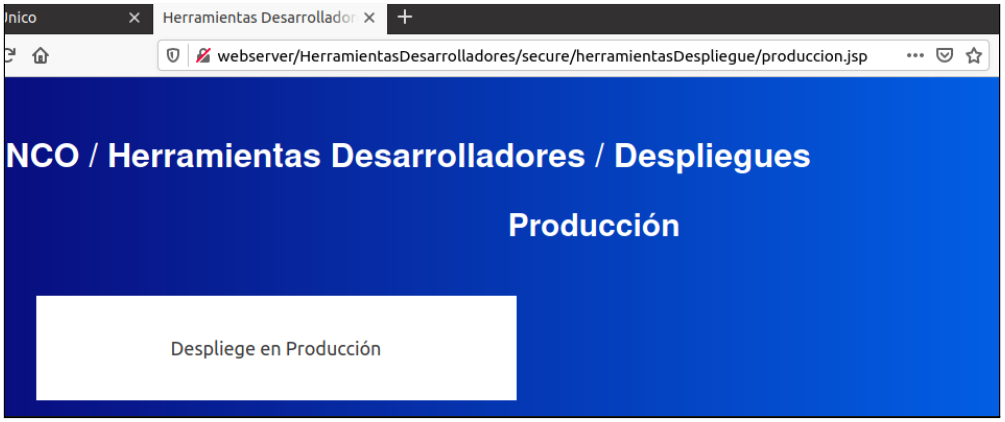

Imagen 4.6. Página de despliegue en producción

Accedemos ahora con el usuario con rol de desarrollador, que no tiene permisos para acceder a este recurso y vemos que a nivel de vista se omite y no aparecen las funcionalidades:

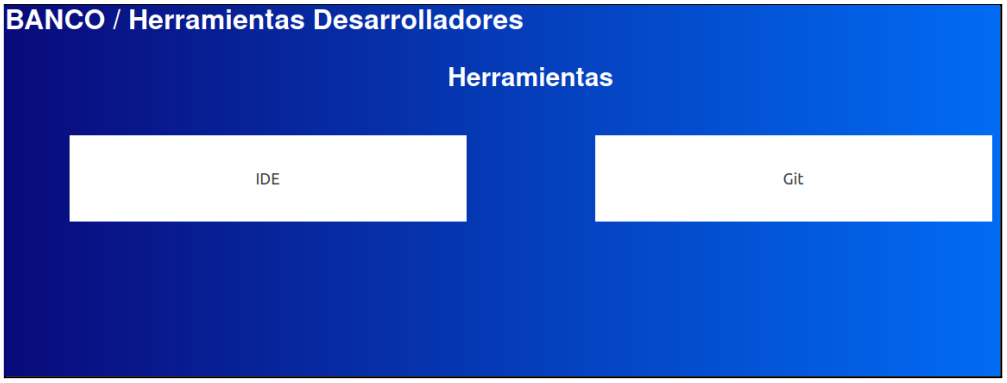

Imagen 4.7. Página principal de Herramientas de desarrolladores

Para estar seguros de que el usuario no puede acceder, tratamos de alcanzar directamente el recurso que ofrece la página de subida de código a producción que es *http://webserver/HerramientasDesarrolladores/secure/herramientasDespliege/produccion.jsp* y la aplicación nos devuelve lo siguiente:

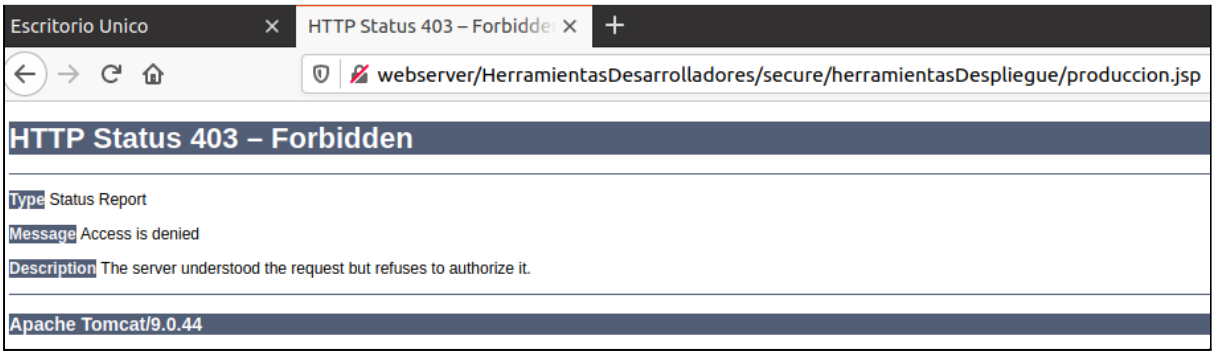

Imagen 4.8. Página de error por permisos

CAS nos devuelve una página HTML que marca un error 403, es decir, acceso denegado al recurso. Podemos confirmar por lo tanto que el acceso está correctamente perfilado para los usuarios.

4.3 Alta disponibilidad

Para la prueba de alta disponibilidad queremos comprobar que el balanceador hace su trabajo y ante una caída de uno de los servidores de SSO (*ssoserver1* o *ssoserver2*) el otro funciona y no deja sin servicio de login a las aplicaciones.

Apagamos uno de los servidores y vemos que al acceder a la aplicación nos da un error ya que el servicio está apagado:

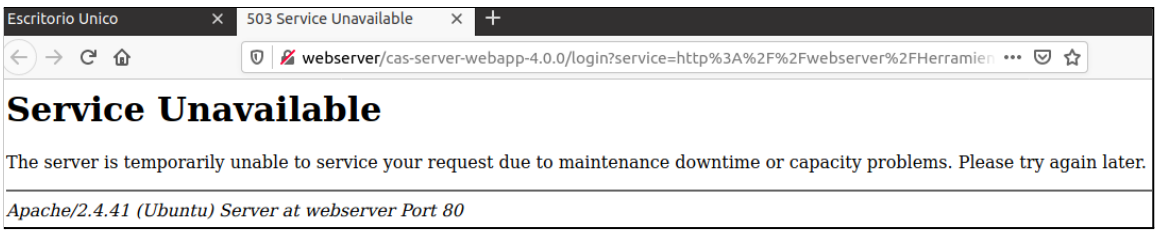

Imagen 4.9. Página de error por servicio apagado

Encendemos el otro servidor y vemos que ya sí que alcanza el recurso:

Imagen 4.10. Página principal de Herramientas de desarrolladores

5. Conclusiones

5.1 Conclusiones del trabajo

De este trabajo se han aprendido las virtudes y facilidades que aportan a una empresa un sistema de Single Sign-On. El hecho de disponer de un punto centralizado para realizar la autenticación, dota de gran flexibilidad a las aplicaciones de las empresas, en el sentido de que, por ejemplo, podemos crecer en aplicaciones sin que tengamos que reparar en montar un sistema de autenticación exclusivo para ellas.

Además de las ventajas a nivel de empresa, para los empleados también supone un punto extra de comodidad, tanto desde el punto de vista de memorizar distintas credenciales, como del de evitar tener que autenticarse una y otra vez todos los días para poder realizar sus labores diarias.

5.2 Logro de los objetivos

De los objetivos que nos habíamos marcado se han podido conseguir prácticamente todos. Hemos conseguido desarrollar las aplicaciones que conforman la estructura de la empresa y las hemos interconectado a través de un sistema de SSO con CAS Server.

El objetivo que no se ha alcanzado es el de securizar las comunicaciones HTTP sobre SSL/TLS. Proteger las comunicaciones es una necesidad muy importante desde el punto de vista de la seguridad, pero se han dado una serie de errores que han impedido conseguir que el proyecto funcionara con esta capacidad.

5.3 Análisis crítico

Con respecto al desarrollo del proyecto considero que desde un primer momento se ideó una correcta planificación (sujeta a algunas pequeñas modificaciones posteriores) de los hitos que se querían abordar, por lo que no ha habido problemas para seguirla día a día y abordar todo el trabajo con los tiempos marcados desde un principio.

Uno de los puntos que hubo que cambiar para garantizar cumplir con los plazos fue la decisión de no usar WildFly como servidor de aplicaciones. Este servidor fue la opción que a priori parecía más acertada pero, dada la alta capacidad de Wildfly y que aporta una mayor complejidad al proyecto desvinculada del objetivo de éste, se decidió usar Apache Tomcat, que se centraba más en la necesidad del proyecto y se trataba de un producto conocido.

6. Glosario

Listado con la definición de los términos y acrónimos más relevantes utilizados dentro de la Memoria:

1. **Autenticación:** proceso de verificación de la identidad de un usuario.
2. **Autorización:** valida si el usuario tiene permiso para acceder a un recurso.
3. **Credenciales:** combinación de nombre de usuario (username) y contraseña para autenticarse en un sistema.
4. **SSO:** Single Sign-On. Método de autenticación segura que proporciona la capacidad de acceso a diferentes aplicaciones mediante el uso de unas credenciales únicas.
5. **Login:** proceso de autenticación en un sistema.
6. **Logout:** proceso de cierre de sesión en un sistema.
7. **Software:** conjunto de programas y rutinas que permiten a la computadora realizar determinadas tareas.
8. **Hardware:** conjunto de elementos físicos o materiales que constituyen una computadora o un sistema informático.
9. **Diagrama de Gantt:** gráfica que muestra el tiempo de dedicación previsto para diferentes tareas a lo largo de un tiempo determinado.
10. **Servidor[56]:** conjunto de computadoras capaz de atender las peticiones de un cliente y devolverle una respuesta en concordancia.
11. **Firewall:** elemento informático que trata de bloquear el acceso, a una red privada conectada a Internet, a usuarios no autorizados.
12. **Redes[57]:** conjunto de sistemas informáticos independientes conectados entre sí, de tal forma que posibilitan un intercambio de datos.
13. **Máquina virtual:** simulación (virtualización) de un equipo con su sistema operativo.
14. **Router:** dispositivo que conecta y permite el tráfico de datos entre redes.
15. **DMZ[58]:** red aislada dentro de la red interna de la organización. En ella se encuentran ubicados los recursos que deben ser accesibles desde Internet, como el servidor web.
16. **Intranet:** red interna de una organización.
17. **Alta Disponibilidad:** replicar sistemas informáticos para evitar la pérdida de servicio ante una indisponibilidad.
18. **LDAP:** (Lightweight Directory Access Protocol) o Protocolo Ligero de Acceso a Directorios. Se refiere a un protocolo que permite el acceso a un directorio ordenado y distribuido para buscar información.
19. **OpenLDAP:** implementación Open Source de LDAP.

20. **Open Source Software[59]:** software que cualquier persona puede inspeccionar, modificar y mejorar.
21. **Apache Server:** servidor web.
22. **(CAS):** Central Authentication Service. Actúa interceptando los requisitos de login presentados por las aplicaciones para completarlos con el usuario y contraseña.
23. **Java:** lenguaje de programación.
24. **JPS:** tecnología que usa Java para crear páginas webs dinámicas basadas en HTML y XML.
25. **Tickets:** cadena criptográfica que representa la sesión de un usuario.
26. **UFW:** (Uncomplicated Firewall). Interfaz para iptables orientada a simplificar el proceso de configuración de un firewall.
27. **Spring:** framework de desarrollo basado en Java.
28. **Servidor DNS[60]:** recurre a la base de datos de un DNS para responder a las peticiones que guardan relación con el espacio de nombres de dominio.
29. **DNS:** Sistema de Nombres de Dominio o Domain Name Server. Sistema que se ocupa de la administración del espacio de nombres de dominio.
30. **Backend[61]:** capa de acceso a datos de una aplicación.
31. **FrontEnd[61]:** capa de presentación de una aplicación.
32. **HTTP[62]:** Hypertext Transfer Protocol (HTTP) (o Protocolo de Transferencia de Hipertexto) es un protocolo de la capa de aplicación para la transmisión de documentos hipermedia, como HTML.
33. **SSL[63]:** tecnología estandarizada que permite cifrar el tráfico de datos entre equipos.
34. **TLS[63]:** versión actualizada y más segura de SSL.
35. **HTTPS[63]:** HTTP sobre una conexión SSL/TLS.
36. **CA[64]:** (Certification Authority). Entidad de confianza responsable de emitir y revocar los certificados, utilizando en ellos la firma electrónica, para lo cual se emplea la criptografía de clave pública.
37. **Bean[65]:** componente de Java que permite encapsular el contenido y la reutilización de código mediante una estructura sencilla.

7. Bibliografía

1. **Iria Da Cunha.** "El trabajo de fin de grado y de máster: Redacción, defensa y publicación". Publicado en Octubre de 2015 y consultado en Marzo de 2021.
2. **Authenticationworld.com.** "Single Sign-On". URL: https://archive.is/XHlfF. Consultado en Marzo de 2021.
3. **Smith, James E.; Nair, Ravi.** "The Architecture of Virtual Machines". URL: https://minds.wisconsin.edu/handle/1793/11154. Publicado en 2005.
4. **vmware.com**. "HOW VIRTUALIZATION WORKS". URL: https://www.vmware.com/solutions/virtualization.html. Consultado en Marzo de 2021.
5. **oracle.com**. "JavaServer Pages Technology". URL: https://www.oracle.com/java/technologies/jspt.html. Consultado en Abril de 2021.
6. **html.spec.whatwg.org.** "HTML". URL: https://html.spec.whatwg.org/#html-vs-xhtml. Consultado en Marzo de 2021.
7. **w3.org.** "CSS". URL: https://www.w3.org/standards/webdesign/htmlcss#whatcss. Consultado en Marzo de 2021.
8. **httpd.apache.org.** "Apache HTTP Server Project". URL: hhttps://httpd.apache.org/. Consultado en Marzo de 2021.
9. **es.wikipedia.org**. "LDAP - Protocolo ligero de acceso a directorios". URL: https://es.wikipedia.org/wiki/Protocolo_ligero_de_acceso_a_directorios. Consultado en Marzo de 2021.
10. **php.net**. "LDAP". URL: https://www.php.net/manual/es/intro.ldap.php. Consultado en Marzo de 2021.
11. **openldap.org**. "OpenLDAP". URL: https://www.openldap.org/. Consultado en Marzo de 2021.
12. **es.wikipedia.org**. "OpenLDAP". URL: https://es.wikipedia.org/wiki/OpenLDAP. Consultado en Marzo de 2021.
13. **wikipedia.org**. "Router". URL: https://es.wikipedia.org/wiki/Router. Consultado en Marzo de 2021.
14. **wikipedia.org**. "Servidor de aplicaciones". URL: https://es.wikipedia.org/wiki/Servidor_de_aplicaciones. Consultado en Marzo de 2021.
15. **oracle.com**. "Java EE". URL: https://www.oracle.com/es/java/technologies/java-ee-glance.html. Consultado en Marzo de 2021.
16. **onelogin.com**. "How does single sign-on work?". URL: https://www.onelogin.com/learn/how-single-sign-on-works. Consultado en Marzo de 2021.
17. **wikipedia.org**. "Single Sign-On". URL: https://en.wikipedia.org/wiki/Single_sign-on. Consultado en Marzo de 2021.
18. **wikipedia.org**. "List of single sign-on implementations". URL: https://en.wikipedia.org/wiki/List_of_single_sign-on_implementations. Consultado en Marzo de 2021.
19. **apereo.org**. "CAS". URL: https://www.apereo.org/projects/cas. Consultado en Marzo de 2021.
20. **dacs.dss.ca**. "DACS". URL: https://dacs.dss.ca/. Consultado en Marzo de 2021.
21. **freeipa.org**. "FreeIPA". URL: https://www.freeipa.org/page/Main_Page. Consultado en Marzo de 2021.
22. **ibm.com**. "Single sign-on". URL: https://www.ibm.com/support/knowledgecenter/ssw_ibm_i_73/rzamz/rzamzsso.htm. Consultado en Marzo de 2021.

23. **josso.org**. "Simplified Identity and Access". URL: http://www.josso.org/. Consultado en Marzo de 2021.
24. **keycloak.org**. "Open Source Identity and Access Management". URL: https://www.keycloak.org/. Consultado en Marzo de 2021.
25. **openidentityplatform.org**. "OpenAM". URL: https://www.openidentityplatform.org/openam. Consultado en Marzo de 2021.
26. **shibboleth.net**. "Secure Identity Management Solutions". URL: https://www.shibboleth.net/products/. Consultado en Marzo de 2021.
27. **wso2.com**. "Identity Server". URL: https://wso2.com/identity-and-access-management/. Consultado en Marzo de 2021.
28. **redeszone.net**. "Diferencias entre la autenticación y la autorización". URL: https://www.redeszone.net/tutoriales/seguridad/diferencias-autenticacion-autorizacion/. Publicado en Junio de 2020 y consultado en Marzo de 2021.
29. **auth0.com**. "Role Based Access Control". URL: https://auth0.com/docs/authorization/rbac. Consultado en Marzo de 2021.
30. **digitalocean.com**. "Configuración inicial del servidor con Ubuntu 20.04". URL: https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-20-04-es. Publicado en Mayo de 2020 y consultado en Marzo de 2021.
31. **digitalocean.com**. "Cómo instalar el servidor web Apache en Ubuntu 20.04". URL: https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-ubuntu-20-04-es. Publicado en Mayo de 2020 y consultado en Marzo de 2021.
32. **wildfly.org**. "WildFly". URL: https://www.wildfly.org/. Consultado en Marzo de 2021.
33. **linux.com**. "How to Install OpenLDAP on Ubuntu Server 18.04". URL: https://www.linux.com/topic/desktop/how-install-openldap-ubuntu-server-1804/. Publicado en Marzo de 2019 y consultado en Marzo de 2021.
34. **digitalocean.com**. "Cómo instalar Apache Tomcat 9 en Ubuntu 18.04". URL: https://www.digitalocean.com/community/tutorials/install-tomcat-9-ubuntu-1804-es. Publicado en Enero de 2020 y consultado en Marzo de 2021.
35. **apereo.github.io**. "WAR Overlay Installation". URL: https://apereo.github.io/cas/4.2.x/installation/Maven-Overlay-Installation.html#gradle. Consultado en Marzo de 2021.
36. **github.com**. "CAS Maven Overlay". URL: https://github.com/apereo/cas-overlay-template/tree/4.2. Publicado en Enero de 2017 y consultado en Marzo de 2021.
37. **apereo.github.io/cas**. "Configuring Authentication Components". URL: https://apereo.github.io/cas/4.0.x/installation/Configuring-Authentication-Components.html. Consultado en Marzo de 2021.
38. **spring.io**. "Spring Framework". URL: https://spring.io/projects/spring-framework. Consultado en Abril de 2021.
39. **spring.io**. "CAS Authentication". URL: https://docs.spring.io/spring-security/site/docs/3.2.0.CI-SNAPSHOT/reference/html/cas.html. Consultado en Abril de 2021.
40. **sublimetext.com**. "Sublime Text 3". URL: https://www.sublimetext.com/3. Consultado en Abril de 2021.
41. **linuxize.com**. "How to Install Sublime Text 3 on Ubuntu 20.04". URL: https://linuxize.com/post/how-to-install-sublime-text-3-on-ubuntu-20-04/. Publicado en Agosto de 2020 y consultado en Abril de 2021.
42. **apereo.github.io**. "CAS - LDAP Authentication". URL: https://apereo.github.io/cas/4.0.x/installation/LDAP-Authentication.html. Consultado en Abril de 2021.

43. **mvnrepository.com**. "LDAPTIVE CORE 1.1.0". URL: https://mvnrepository.com/artifact/org.ldaptive/ldaptive/1.1.0. Publicado en Octubre de 2015 y consultado en Abril de 2021.

44. **mvnrepository.com**. "CAS Server Support LDAP 4.0.0". URL: https://mvnrepository.com/artifact/org.jasig.cas/cas-server-support-ldap/4.0.0. Publicado en Mayo de 2014 y consultado en Abril de 2021.

45. **apereo.github.io**. "CAS protocol". URL: https://apereo.github.io/cas/4.2.x/protocol/CAS-Protocol.html. Consultado en Abril de 2021.

46. **redeszone.net**. "Que es y para qué sirve el archivo hosts en un equipo". URL: https://www.redeszone.net/tutoriales/internet/que-es-archivo-hosts/. Publicado en Diciembre de 2019 y consultado en Abril de 2021.

47. **httpd.apache.org**. "Guía de Proxy Inverso". URL: https://httpd.apache.org/docs/trunk/es/howto/reverse_proxy.html. Consultado en Abril de 2021.

48. **digitalocean.com**. "How To Use Apache as a Reverse Proxy with mod_proxy on Ubuntu 16.04". URL: https://www.digitalocean.com/community/tutorials/how-to-use-apache-as-a-reverse-proxy-with-mod_proxy-on-ubuntu-16-04. Publicado en Febrero de 2017 y consultado en Abril de 2021.

49. **adslzone.net**. "HTTP vs HTTPS, diferencias entre protocolos". URL: https://www.adslzone.net/reportajes/internet/ipv6-vs-ipv4/. Publicado en Enero de 2021 y consultado en Abril de 2021.

50. **tomcat.apache.org**. "SSL/TLS Configuration HOW-TO". URL: http://tomcat.apache.org/tomcat-8.0-doc/ssl-howto.html. Publicado en Junio de 2018 y consultado en Abril de 2021.

51. **openssl.org**. "OpenSSL Cryptography and SSL/TLS Toolkit". URL: https://www.openssl.org/. Consultado en Abril de 2021.

52. **docs.oracle.com**. "keytool". URL: https://docs.oracle.com/javase/8/docs/technotes/tools/unix/keytool.html. Consultado en Abril de 2021.

53. **digicert.com**. "Apache SSL de instalación del certificado". URL: https://www.digicert.com/es/instalar-certificado-ssl-apache.htm. Consultado en Abril de 2021.

54. **softwarelab.org**. "¿Qué es un firewall?". URL: https://softwarelab.org/es/que-es-un-firewall/. Consultado en Abril de 2021.

55. **digitalocean.com**. "Cómo configurar un firewall con UFW en Ubuntu 18.04". URL: https://www.digitalocean.com/community/tutorials/como-configurar-un-firewall-con-ufw-en-ubuntu-18-04-es. Publicado en Diciembre de 2019 y consultado en Abril de 2021.

56. **wikipedia.org**. "Servidor". URL: https://es.wikipedia.org/wiki/Servidor. Publicado en Abril de 2021 y consultado en Mayo de 2021.

57. **ionos.es**. "Los tipos de redes más conocidos". URL: https://www.ionos.es/digitalguide/servidores/know-how/los-tipos-de-redes-mas-conocidos/. Publicado en Julio de 2019 y consultado en Mayo de 2021.

58. **incibe.es**. "Qué es una DMZ y cómo te puede ayudar a proteger tu empresa". URL: https://www.incibe.es/protege-tu-empresa/blog/dmz-y-te-puede-ayudar-proteger-tu-empresa. Publicado en Septiembre de 2019 y consultado en Mayo de 2021.

59. **opensource.com**. "What is open source?". URL: https://opensource.com/resources/what-open-source. Consultado en Mayo de 2021.

60. **ionos.es**. "El servidor DNS y la resolución de nombres en Internet". URL: https://www.ionos.es/digitalguide/servidores/know-how/que-es-el-servidor-dns-y-como-funciona/. Publicado en XXXX de xxxx y consultado en XXXX de 2021.

61. **wikipedia.org**. "Front end y back end". URL: https://es.wikipedia.org/wiki/Front_end_y_back_end. Publicado en Noviembre de 2020 y consultado en XXXX de 2021.

62. **developer.mozilla.org**. "HTTP". URL: https://developer.mozilla.org/es/docs/Web/HTTP. Consultado en Mayo de 2021.
63. **digicert.com**. "¿QUÉ SON SSL, TLS Y HTTPS…?". URL: https://www.digicert.com/es/what-is-ssl-tls-https/. Consultado en Mayo de 2021.
64. **wikipedia.org**. "Autoridad de certificación". URL: https://es.wikipedia.org/wiki/Autoridad_de_certificaci%C3%B3n. Publicado en Febrero de 2021 y consultado en Mayo de 2021.
65. **javadesde0.com**. "Introducción a los Beans". URL: https://javadesde0.com/introduccion-a-los-beans/. Publicado en Julio de 2019 y consultado en Mayo de 2021.

8. Anexos

8.1 Configuración Máquinas VirtualBox

Como sabemos, se han creado 5 servidores distintos:

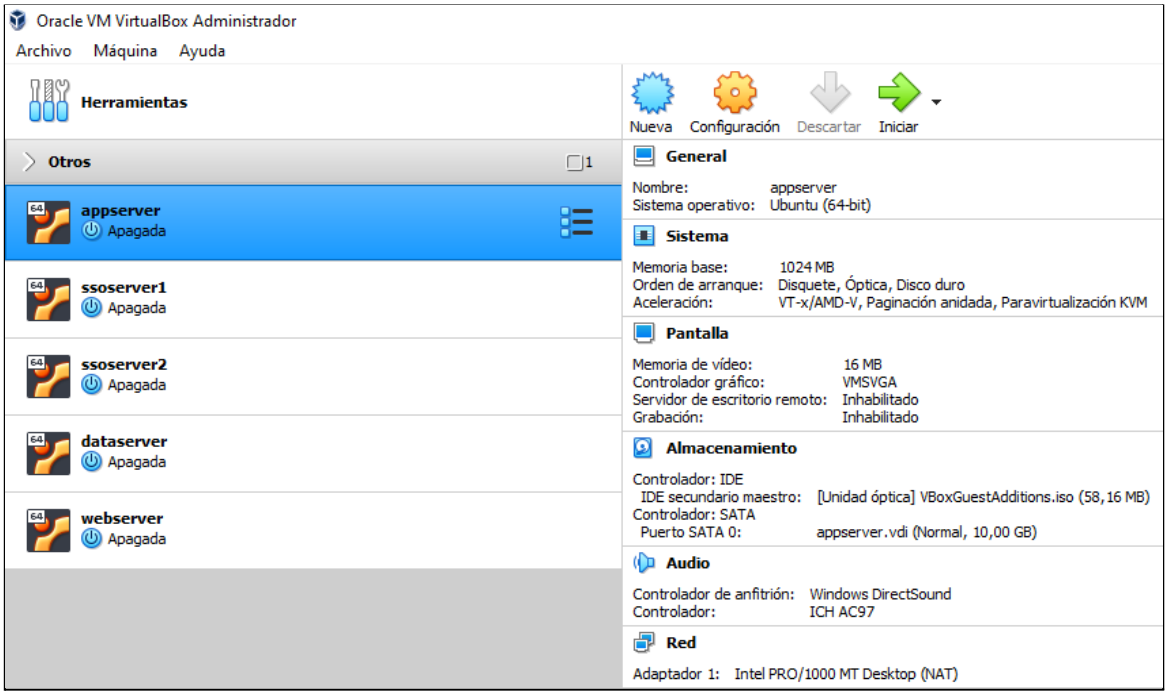

Imagen 8.1. Máquinas virtuales del proyecto

Estos 5 servidores conservan las mismas características, que son:

- Sistema Operativo Ubuntu Server 64 bits
- Procesador Intel Core i7 (el de la máquina anfitrión)
- 1024MB de memoria RAM
- 10GB de almacenamiento

8.2 Configuración inicial de los servidores

Junto con la instalación de Ubuntu Server en las máquinas virtuales, vamos a tener en cuenta una serie de configuraciones básicas[30] para reforzar la seguridad del servidor.

Comenzaremos por crear un nuevo usuario para hacer uso del sistema y que no sea el usuario *root*. Para ello, mediante el comando **adduser** crearemos el usuario:

```
administrador@webserver:~$ sudo adduser useradmin
```

Imagen 8.2. Creación de usuario

En ocasiones será necesario que nuestro nuevo usuario disponga de permisos administrativos para realizar ciertas tareas, por lo que debemos añadirlo al grupo sudo

mediante el comando **sudo usermod -aG sudo useradmin**. Cuando iniciemos sesión con el usuario **useradmin** tendremos privilegios de superusuario siempre que usemos delante *sudo*. Al iniciar una sesión con este usuario la propia terminal muestra ese mensaje:

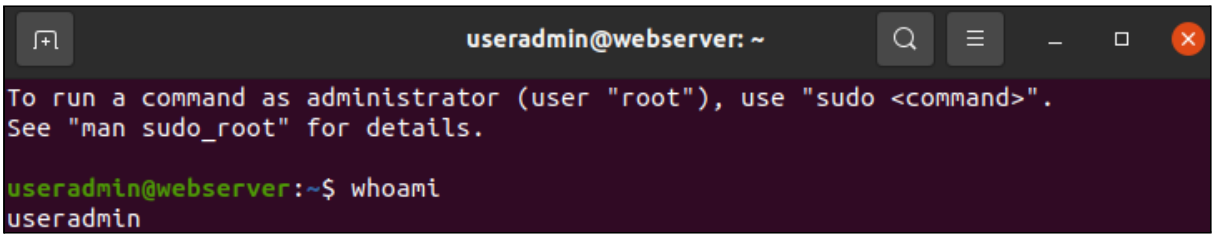

Imagen 8.3. Comprobar creación de usuario

8.3 Instalación Apache HTTP Server Project

Instalamos el servidor web Apache HTTP[31] en la máquina virtual *webserver* mediante la ejecución de **sudo apt install apache2**.

Tras la instalación, vamos a configurar un firewall básico con **UFW** y comenzamos listando los perfiles de aplicación UFW ejecutando **sudo ufw app list**:

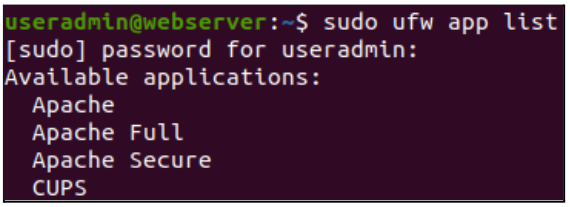

Imagen 8.4. Perfiles de aplicación UFW

De los 4 perfiles disponibles vemos que 3 de ellos son de Apache:

- Apache: este perfil abre solo el puerto 80 (tráfico web)
- Apache Full: este perfil abre el puerto 80 y el puerto 443 (tráfico TLS/SSL cifrado)
- Apache Secure: este perfil abre solo el puerto 443

La idea es que se permita el tráfico tanto por el puerto 80 como por el puerto 443, por lo que ejecutaremos el comando **sudo ufw allow 'Apache Full'** para implementarlo y luego el comando **sudo ufw enable** para habilitar el firewall.

Lanzamos el comando **sudo ufw statuts** para comprobar que se han aplicado correctamente los cambios y vemos que solamente se permitirá el tráfico que queríamos:

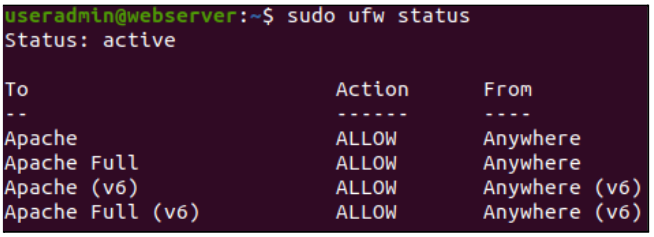

Imagen 8.5. Estados de firewall UFW

Apache se inicia solo tras la instalación, pero lo comprobaremos ejecutando el comando **sudo systemctl status apache2**, que muestra la siguiente salida corroborando que Apache HTTP server está activo:

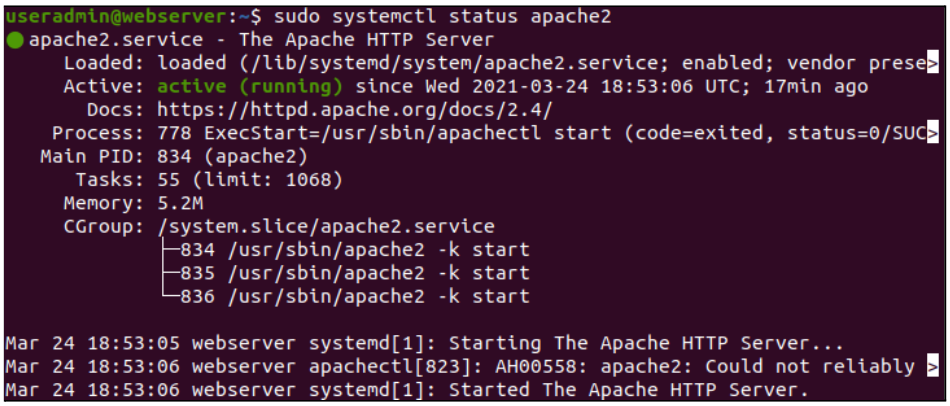

Imagen 8.6. Comprobar estado Apache HTTP

Otra forma de comprobar que está funcionando correctamente es yendo a un navegador y escribiendo localhost en la barra de búsqueda. Ésto nos debería mostrar una web de bienvenida de Apache como vemos en la imagen:

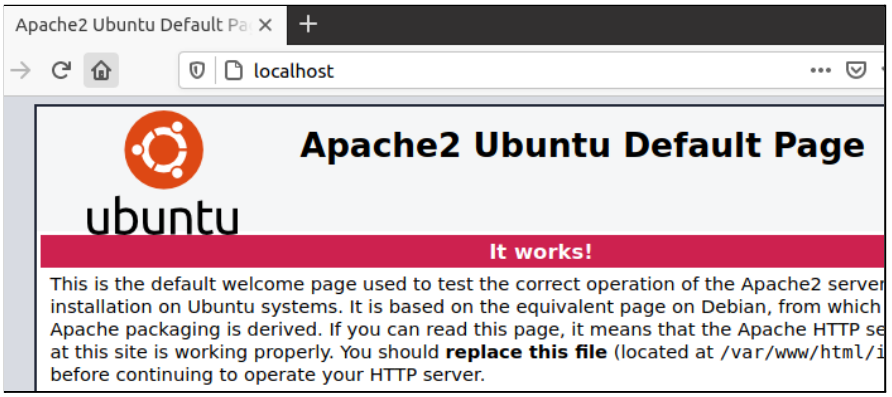

Imagen 8.7. Comprobar funcionamiento Apache HTTP

8.4 Instalación OpenLDAP

Instalamos OpenLDAP[33] en la máquina virtual *dataserver*, que es la máquina que contiene el servidor de datos, en nuestro caso, contendrá el LDAP de empleados. Para instalar OpenLDAP en nuestro servidor de datos usamos el comando **sudo apt instal slapd ldap-utils**. Durante el proceso de instalación nos pedirá una contraseña de administrador de OpenLDAP:

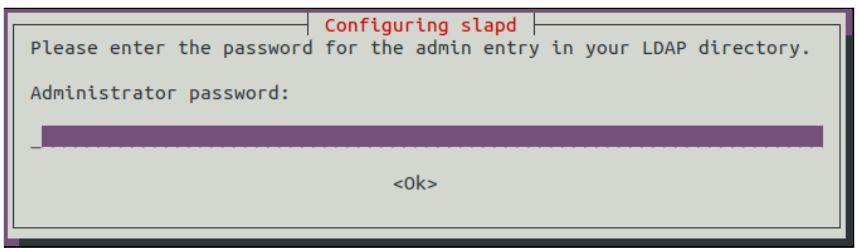

Imagen 8.8. Solicitud de contraseña de administrador de OpenLDAP

Una vez la instalación se haya completado, pasamos a configurar OpenLDAP. Para ello nos serviremos del comando **sudo dpkg-reconfigure slapd**. En el proceso nos solicitará introducir un nombre para el dominio DNS:

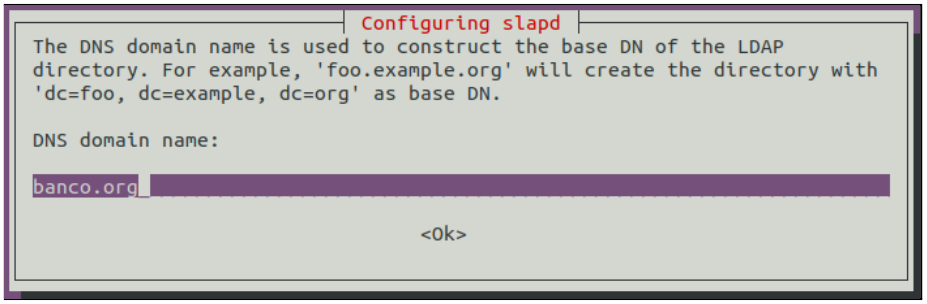

Imagen 8.9. Configuración dominio dominio DNS

Y comprobamos que la instalación haya ido correctamente con el comando **/etc/init.d/slapd status**, que nos debería mostrar algo como lo siguiente:

```
useradmin@dataserver:~$ /etc/init.d/slapd status
● slapd.service - LSB: OpenLDAP standalone server (Lightweight Directory Access
Protocol)
     Loaded: loaded (/etc/init.d/slapd; generated)
    Drop-In: /usr/lib/systemd/system/slapd.service.d
             └─slapd-remain-after-exit.conf
     Active: active (running) since Fri 2021-03-26 08:21:45 UTC; 1min 52s ago
       Docs: man:systemd-sysv-generator(8)
    Process: 18591 ExecStart=/etc/init.d/slapd start (code=exited, status=0/SUCC
ESS)
      Tasks: 3 (limit: 1068)
     Memory: 3.3M
     CGroup: /system.slice/slapd.service
             └─18598 /usr/sbin/slapd -h ldap:/// ldapi:/// -g openldap -u openl…
```

Imagen 8.10. Comprobar estado LDAP

8.5 Instalación Apache Tomcat 9

En este apartado procederemos a instalar Apache Tomcat 9[34] en las máquinas virtuales *ssoserver1, ssoserver2* y *appserver*, por lo tanto, los pasos que se indican a continuación se han de realizar en estas tres máquinas.

Empezamos instalando la JDK de Java con el comando **sudo apt install default-jdk**. Tras instalar la JDK y antes de instalar Tomcat, sería deseable por temas de seguridad crear un usuario sin privilegios para Tomcat. Crearemos entonces un nuevo grupo y posteriormente un nuevo usuario para que lo ejecuten con los comando:

- **sudo groupadd tomcat**
- **sudo useradd -s /bin/false -g tomcat -d /opt/tomcat tomcat**

En este paso procedemos a instalar Tomcat, que descargaremos su última versión binaria y luego se configurará manualmente. Comenzamos entonces yendo al directorio /tmp con el comando **cd /tmp** y situados en esta carpeta ejecutamos la descarga con **curl -O**

https://downloads.apache.org/tomcat/tomcat-9/v9.0.44/bin/apache-tomcat-9.0.44.tar.gz
Tras esta ejecución se nos descargará el contenido comprimido en un fichero *tar.gz* que vemos que se corresponde con la versión v9.0.44 de Apache Tomcat:

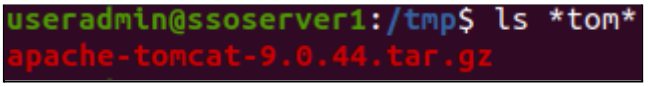

Imagen 8.11. Fichero tar.gz versión v9.0.44 de Apache Tomcat

La idea es instalar Apache Tomcat en un nuevo directorio que crearemos con **sudo mkdir /opt/tomcat** y luego extraeremos el archivo *tar.gz* con **sudo tar xzvf apache-tomcat-*tar.gz -C /opt/tomcat --strip-components=1**.

Es turno de dar permisos al usuario *tomcat* que creamos previamente para tener acceso a la instalación. Empezamos por posicionarnos en la carpeta de la instalación con **cd /opt/tomcat** y le otorgamos la propiedad al grupo *tomcat* al directorio con **sudo chgrp -R tomcat /opt/tomcat**. Continuamos el proceso dando permisos de lectura al grupo *tomcat* sobre el directorio *conf* y sus subdirectorios, y acceso de ejecución al directorio:

- **sudo chmod -R g+r conf**
- **sudo chmod g+x conf**

Nos aseguraremos de que el usuario *tomcat* sea el propietario de los directorios webapps, work, temp y logs con **sudo chown -R tomcat webapps/ work/ temp/ logs/**.

Procedemos a dar la capacidad a Tomcat de ejecutarse como servicio *systemd*. El primer paso es saber dónde está instalado Java (ruta conocida como *JAVA_HOME*). Comprobamos ésto con el comando **sudo update-java-alternatives -l** y en mi caso muestra lo siguiente:

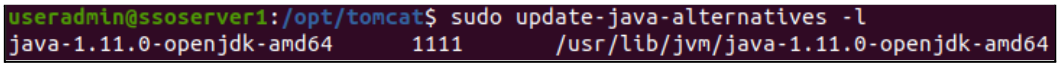

Imagen 8.12. Ruta JAVA_HOME

Sabiendo esto ya podemos crear el archivo de servicio *systemd*. Ejecutamos el comando **sudo gedit /etc/systemd/system/tomcat.service** para abrir un editor de textos (podría ser gedit, nano, u otro editor a gusto de la persona) y copiamos el siguiente contenido:

```
[Unit]
Description=Apache Tomcat Web Application Container
After=network.target
[Service]
Type=forking
Environment=JAVA_HOME=/usr/lib/jvm/java-1.11.0-openjdk-amd64
Environment=CATALINA_PID=/opt/tomcat/temp/tomcat.pid
Environment=CATALINA_HOME=/opt/tomcat
Environment=CATALINA_BASE=/opt/tomcat
Environment='CATALINA_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC'
Environment='JAVA_OPTS=-Djava.awt.headless=true -Djava.security.egd=file:/dev/./urandom'
ExecStart=/opt/tomcat/bin/startup.sh
ExecStop=/opt/tomcat/bin/shutdown.sh
User=tomcat
Group=tomcat
UMask=0007
RestartSec=10
Restart=always
[Install]
WantedBy=multi-user.target
```

Imagen 8.13. Configuración /etc/systemd/system/tomcat.service

Notamos que la variable *JAVA_HOME* contiene el valor **/usr/lib/jvm/java-1.11.0-openjdk-amd64**, que era el que obteníamos anteriormente.

Una vez creado el archivo en ambas máquinas, volvemos a cargar el *daemon systemd* para que tenga en cuenta esta nueva información que hemos proporcionado. Para realizarlo ejecutamos el comando **sudo systemctl daemon-reload**.

Ya podemos iniciar el servicio Tomcat con **sudo systemctl start tomcat** y probar que se ha iniciado correctamente comprobando su estado con **sudo systemctl status tomcat**:

```
useradmin@ssoserver1:/opt/tomcat$ sudo systemctl start tomcat
useradmin@ssoserver1:/opt/tomcat$ sudo systemctl status tomcat
● tomcat.service - Apache Tomcat Web Application Container
     Loaded: loaded (/etc/systemd/system/tomcat.service; disabled; vendor preset: enabled)
     Active: active (running) since Sat 2021-03-27 11:50:09 UTC; 58s ago
    Process: 4208 ExecStart=/opt/tomcat/bin/startup.sh (code=exited, status=0/SUCCESS)
   Main PID: 4226 (java)
      Tasks: 30 (limit: 1067)
     Memory: 151.9M
     CGroup: /system.slice/tomcat.service
             └─4226 /usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -Djava.util.logging.config.file=>

Mar 27 11:50:09 ssoserver1 systemd[1]: tomcat.service: Scheduled restart job, restart counter is at >
Mar 27 11:50:09 ssoserver1 systemd[1]: Stopped Apache Tomcat Web Application Container.
Mar 27 11:50:09 ssoserver1 systemd[1]: Starting Apache Tomcat Web Application Container...
Mar 27 11:50:09 ssoserver1 systemd[1]: Started Apache Tomcat Web Application Container.
Mar 27 11:50:09 ssoserver1 startup.sh[4208]: Tomcat started.
lines 1-15/15 (END)
```

Imagen 8.14. Comprobar estado de Tomcat

Ya que el servidor está ejecutándose nos queda probarlo para garantizar que todo está correctamente. Antes de ellos, vamos a realizar unos ajustes a nivel de firewall para permitir que las solicitudes lleguen al servicio. Para ello ejecutamos **sudo ufw allow 8080** y probamos desde un navegador que funcione ingresando en la barra de búsqueda la siguiente dirección: http://localhost:8080. Tras realizar estos pasos, el navegador nos debería mostrar una página como la siguiente:

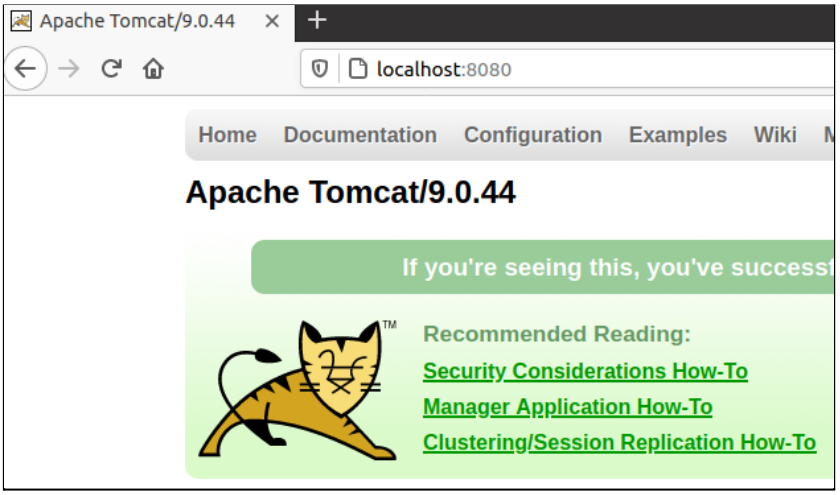

Imagen 8.15. Comprobar funcionamiento de Tomcat

Para finalizar la instalación es necesario tener en cuenta que para usar la aplicación de administración web de Tomcat hay que añadir un inicio de sesión al servidor. Esto se hace mediante la edición del fichero **tomcat-users.xml**.

Abrimos el fichero con **sudo gedit /opt/tomcat/conf/tomcat-users.xml** y vemos que vienen comentados ejemplos de cómo se define un usuario:

```
<!--
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <user username="tomcat" password="<must-be-changed>" roles="tomcat"/>
  <user username="both" password="<must-be-changed>" roles="tomcat,role1"/>
  <user username="role1" password="<must-be-changed>" roles="role1"/>
-->
```

Imagen 8.16. Definición de usuarios en /opt/tomcat/conf/tomcat-users.xml

Nosotros queremos un usuario que pueda acceder a *manager-gui* y *admin-gui*, por lo que añadiremos una línea al fichero tal que así:

```
<tomcat-users . . .>
        <user username="admin" password="password" roles="manager-gui,admin-gui"/>
</tomcat-users>
```

Imagen 8.17. Usuario de ejemplo de Tomcat

Sería idóneo que los valores de username y password no fueran unos básicos como los que se exponen en el anterior pantallazo a modo de ejemplo.

Comprobamos que podemos acceder a **Manager App**. Para ello, accedemos desde el navegador al enlace http://localhost:8080/manager/html, nos pedirá las credenciales del usuario definido en el paso previo y nos mostrará una página similar a esta:

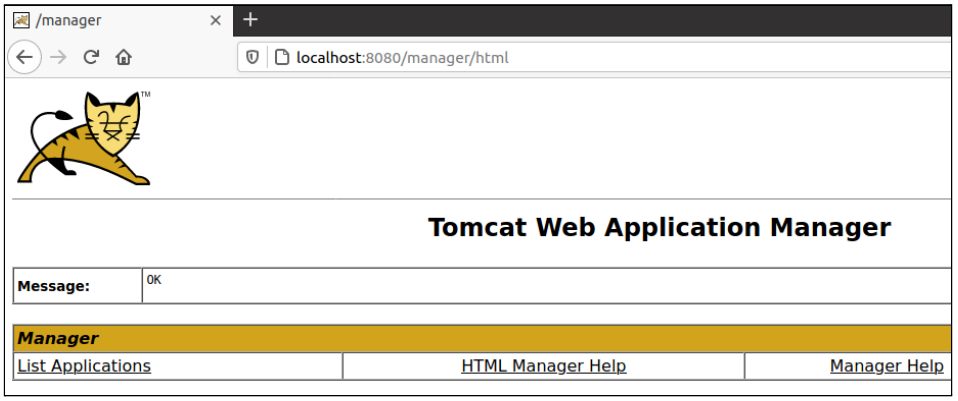

Imagen 8.18. Comprobar acceso a Manager App

Y también comprobamos que podemos acceder a **Host Manager**, que lo haremos accediendo a http://localhost:8080/host-manager/html y nos mostrará una página como la siguiente:

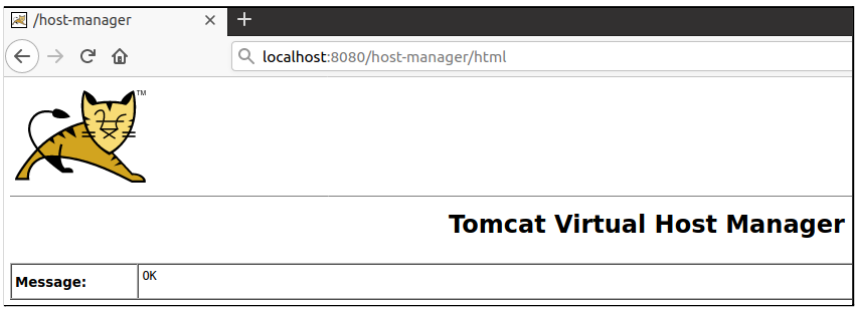

Imagen 8.19. Comprobar acceso a Host Manager

Es importante recordar que todos estos pasos se han realizado tanto en la máquina virtual *ssoserver1* como en la máquinas virtuales *ssoserver2* y *appserver*.

8.6 Instalación de Central Authentication Service (CAS)

En este apartado procederemos a instalar el servidor de SSO **Central Authentication Service (CAS)**[35] en las máquinas virtuales *ssoserver1* y *ssoserver2*, por lo tanto, los pasos que se indican a continuación se han de realizar en ambas máquinas.

Para la instalación de Cas Server seguiremos una documentación facilitada por la asignatura de Identidad Digital cursada en el máster: Material soporte Identidad Digital (MISTIC). Esta documentación está basada en el artículo 38 referenciado en la bibliografía.

Comenzamos entonces por realizar la descarga de **Cas Server 4.0.0** en formato *.zip* desde GitHub. Una vez descargado, descomprimimos el *.zip* y buscamos el fichero **cas-server-webapp-4.0.0.war**, que se encuentra en la carpeta **/modules** del *.zip* descargado. Localizado este fichero, lo que tenemos que hacer es copiarlo en la carpeta **/webapps** de Apache Tomcat. En este caso hacemos la copia desde la consola, ubicados en la carpeta extraída, mediante el comando **sudo cp modules/cas-server-webapp-4.0.0.war /opt/tomcat/webapps/**. Al copiar el fichero en tal destino, lo que ocurrirá es que el **cas-server-webapp-4.0.0.war** se descomprimirá automáticamente al arrancar el Apache Tomcat, quedando la carpeta con el CAS Server ubicada en **/opt/tomcat/webapps/**. Reiniciamos Tomcat con **sudo systemctl restart tomcat** y comprobamos que efectivamente se ha generado una carpeta para Cas Server con **sudo ls -l /opt/tomcat/webapps/ | grep cas**:

```
useradmin@ssoserver1:~$ sudo systemctl restart tomcat
useradmin@ssoserver1:~$ sudo ls -l /opt/tomcat/webapps/ | grep cas
drwxr-x---  7 tomcat tomcat     4096 Mar 29 15:43 cas-server-webapp-4.0.0
-rw-r--r--  1 root   root   25038970 Mar 29 15:43 cas-server-webapp-4.0.0.war
```

Imagen 8.20. Generación carpeta Cas Server en Tomcat

Para comprobar que funciona correctamente, ingresamos en la barra de búsqueda de un navegador la url http://localhost:8080/cas-server-webapp-4.0.0/login y deberá aparecernos algo tal que así:

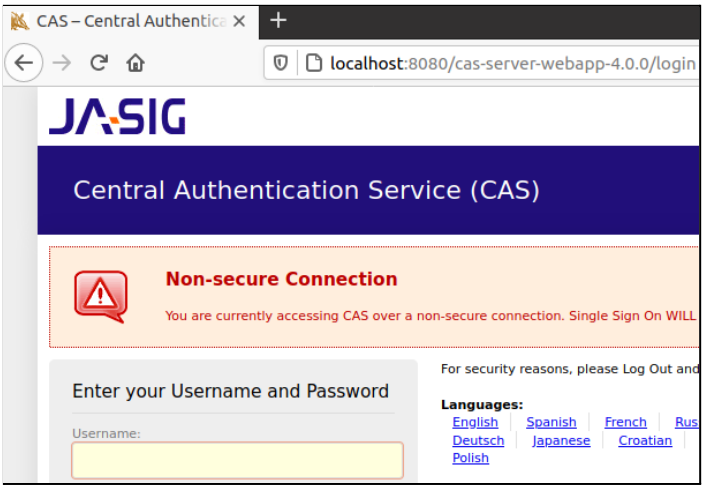

Imagen 8.21. Comprobar funcionamiento CAS

Por defecto se crea un usuario con las credenciales ***casuser*** / ***Mellon***, que podemos comprobarlo en el fichero **/WEB-INF/deployerConfigContext.xml** de Cas Server:

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

Imagen 8.22. usuario de ejemplo de CAS

Probamos a autenticarnos para ver que funciona correctamente. La página que se debería mostrar sería una como la que vemos en el siguiente pantallazo:

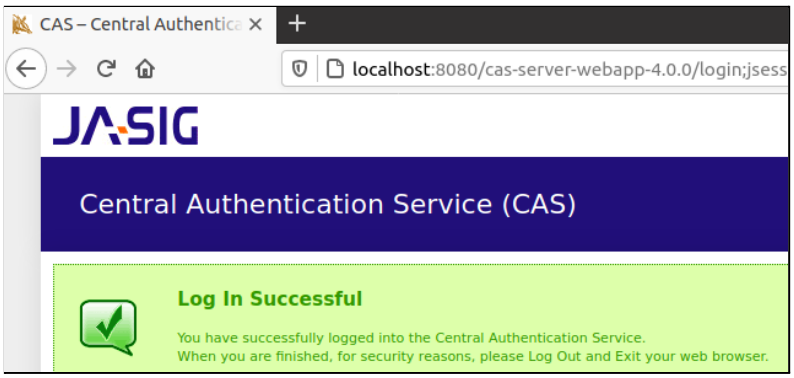

Imagen 8.23. Comprobar autenticación CAS

Con esto, ya habríamos completado la instalación del servidor de SSO a falta de la configuración necesaria para adaptarlo a las necesidades de nuestro proyecto.

Es importante recordar que todos estos pasos se han realizado tanto en la máquina virtual **ssoserver1** como en la máquina virtual **ssoserver2**.

8.7 Bean's del fichero deployerConfigContext.xml

Bean Authenticator:

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

Imagen 8.24. Bean Authenticator

Bean dnResolver

```
<bean id="dnResolver" class="org.ldaptive.auth.PooledSearchDnResolver"
    p:baseDn="${ldap.authn.baseDn}"
    p:subtreeSearch="true"
    p:allowMultipleDns="false"
    p:connectionFactory-ref="searchPooledLdapConnectionFactory"
    p:userFilter="${ldap.authn.searchFilter}" />
```

Imagen 8.25. Bean dnResolver

Bean searchPooledLdapConnectionFactory

```
<bean id="searchPooledLdapConnectionFactory"
      class="org.ldaptive.pool.PooledConnectionFactory"
      p:connectionPool-ref="searchConnectionPool" />
```

Imagen 8.26. Bean searchPooledLdapConnectionFactory

Bean searchConnectionPool

```
<bean id="searchConnectionPool" parent="abstractConnectionPool"
    p:connectionFactory-ref="searchConnectionFactory" />
```

Imagen 8.27. Bean searchConnectionPool

Bean searchConnectionFactory

```
<bean id="searchConnectionFactory"
    class="org.ldaptive.DefaultConnectionFactory"
    p:connectionConfig-ref="searchConnectionConfig" />
```

Imagen 8.28. Bean searchConnectionFactory

Bean searchConnectionConfig

```
<bean id="searchConnectionConfig" parent="abstractConnectionConfig"
    p:connectionInitializer-ref="bindConnectionInitializer" />
```

Imagen 8.29. Bean searchConnectionConfig

Bean bindConnectionInitializer

```
<bean id="bindConnectionInitializer"
      class="org.ldaptive.BindConnectionInitializer"
      p:bindDn="${ldap.authn.managerDN}">
    <property name="bindCredential">
        <bean class="org.ldaptive.Credential"
              c:password="${ldap.authn.managerPassword}" />
    </property>
</bean>
```

Imagen 8.30. Bean bindConnectionInitializer

Bean abstractConnectionPool

```
<bean id="abstractConnectionPool" abstract="true"
      class="org.ldaptive.pool.BlockingConnectionPool"
      init-method="initialize"
      p:poolConfig-ref="ldapPoolConfig"
      p:blockWaitTime= "${ldap.pool.blockWaitTime}"
      p:validator-ref="searchValidator"
      p:pruneStrategy-ref="pruneStrategy"
      p:failFastInitialize="false" />
```

Imagen 8.31. Bean abstractConnectionPool

Bean abstractConnectionConfig

```
<bean id="abstractConnectionConfig" abstract="true"
    class="org.ldaptive.ConnectionConfig"
    p:ldapUrl="${ldap.url}"
    p:connectTimeout="${ldap.connectTimeout}"
    p:useStartTLS="false"
    />
```

Imagen 8.32. Bean abstractConnectionConfig

Bean ldapPoolConfig

```
<bean id="ldapPoolConfig" class="org.ldaptive.pool.PoolConfig"
    p:minPoolSize="${ldap.pool.minSize}"
    p:maxPoolSize="${ldap.pool.maxSize}"
    p:validateOnCheckOut="${ldap.pool.validateOnCheckout}"
    p:validatePeriodically="${ldap.pool.validatePeriodically}"
    p:validatePeriod="${ldap.pool.validatePeriod}" />
```

Imagen 8.33. Bean ldapPoolConfig

Bean pruneStrategy

```
<bean id="pruneStrategy" class="org.ldaptive.pool.IdlePruneStrategy"
    p:prunePeriod="${ldap.pool.prunePeriod}"
    p:idleTime="${ldap.pool.idleTime}" />
```

Imagen 8.34. Bean pruneStrategy

Bean searchValidator

```
<bean id="searchValidator" class="org.ldaptive.pool.SearchValidator" />
```

Imagen 8.35. Bean searchValidator

Bean authHandler

```
<bean id="authHandler" class="org.ldaptive.auth.PooledBindAuthenticationHandler"
      p:connectionFactory-ref="bindPooledLdapConnectionFactory" />
```

Imagen 8.36. Bean authHandler

Bean bindPooledLdapConnectionFactory

```
<bean id="bindPooledLdapConnectionFactory"
      class="org.ldaptive.pool.PooledConnectionFactory"
      p:connectionPool-ref="bindConnectionPool" />
```

Imagen 8.37. Bean bindPooledLdapConnectionFactory

Bean bindConnectionPool

```
<bean id="bindConnectionPool" parent="abstractConnectionPool"
      p:connectionFactory-ref="bindConnectionFactory" />
```

Imagen 8.38. Bean bindConnectionPool

Bean bindConnectionFactory

```
<bean id="bindConnectionFactory"
      class="org.ldaptive.DefaultConnectionFactory"
      p:connectionConfig-ref="bindConnectionConfig" />
```

Imagen 8.39. Bean bindConnectionFactory

Bean bindConnectionConfig

```
<bean id="bindConnectionConfig" parent="abstractConnectionConfig" />
```

Imagen 8.40. Bean bindConnectionConfig