UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño de un sistema de gestión y sincronización de cuentas de usuario para Informatica MDM

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Julia Penadés Ribera

Tutor: Víctor Fuertes de la Cruz

Tutor: Jose Luis Poza Lujan

2016-2017

Agradecimientos

En primer lugar, me gustaría dar las gracias a los tutores de este trabajo de fin de grado, puesto que sin su interés y su participación no habría sido posible realizarlo. A Víctor Fuertes, por la oportunidad que me ha dado al ofrecerme la idea del proyecto y por la ayuda que me ha brindado a lo largo de su desarrollo. A Jose Luis, por haber resuelto todas mis dudas, haberme orientado en su estructura y redacción y por todas las recomendaciones y sugerencias que me ha aportado.

Por último, dar las gracias a mis compañeros, amigos y familia, por estar ahí en todo momento, apoyarme, orientarme y por todos sus consejos. Su ayuda y la confianza que han depositado en mi han sido muy imprescindibles para completar este proyecto.

Resum

En aquest projecte es descriu un sistema d'autenticació i autorització que garanteix la retroalimentació dels usuaris de l'empresa *arhis* amb la finalitat de treballar de forma més òptima amb els conjunts de dades allotjats en els servidors remots dels seus clients. En el projecte s'ha fet un estudi entre els sistemes actuals que proporcionen funcionalitats semblants, així com una comparativa entre els mateixos per garantir la seguretat en la gestió de recursos de l'aplicació desenvolupada. En la present memòria, es mostra en profunditat el disseny del programari, així com el desenvolupament del mateix juntament amb la documentació dels errors i millores detectades al llarg de la implementació. Per a l'elaboració d'aquest document s'han seguit les pautes de la norma tècnica per a la realització de projectes de sistemes de la informació del Consell de Col·legis d'Enginyers en Informàtica, basada en l'estàndard UNE 157.801.

Paraules clau: accés remot, autenticació, autorització, seguretat, servidors, usuaris

Resumen

En este proyecto se describe un sistema de autenticación y autorización que garantiza la retroalimentación de los usuarios de la empresa *arhis* para trabajar de forma más óptima con los conjuntos de datos hospedados en los servidores remotos de sus clientes. En el proyecto se ha hecho un estudio entre los sistemas actuales que proporcionan funcionalidades parecidas, así como una comparativa entre los mismos para garantizar la seguridad en la gestión de recursos de la aplicación desarrollada. En la presente memoria, se muestra en profundidad el diseño del *software*, así como el desarrollo del mismo junto con la documentación de los errores y mejoras detectadas a lo largo de la implementación. Para la elaboración de este documento se han seguido las pautas de la norma técnica para la realización de proyectos de sistemas de la información del Consejo de Colegios de Ingenieros en Informática, basada en el estándar UNE 157801.

Palabras clave: acceso remoto, autenticación, autorización, seguridad, servidores, usuarios

Abstract

This project describes an authentication and authorization system that guarantees the feedback of users of the company *arhis* to work more optimally with datasets hosted on the remote servers of its clients. In this project, a study between the current systems that provide similar functionalities has been made, as well as a comparative among them to guarantee the security in the management of resources of the developed application. In the present specification, the software design is shown in depth, as well as the development of the software together with documentation of errors and improvements detected throughout the implementation. For the elaboration of this document the guidelines of the technical standard for the realization of projects of information systems of the Council of Colleges of Engineers in Computing have been followed, based on the standard UNE 157801.

Keywords: remote access, authentication, authorization, security, servers, users

Contenido

Índice de figuras

Índice de capturas

Índice de tablas

1 Introducción

Hoy en día es imprescindible para cualquier organización, ya sean pequeña, mediana o grande empresa, una correcta gestión de su base de datos o *data base management* (DBM); es bien sabido que a nivel administrativo es de vital importancia mantener una estructura correcta y ordenada de la información de dicha organización. Se habla de la Gestión de Datos Maestros, más bien conocida como *Master Data Management* (MDM), una disciplina directamente relacionada con el mundo de la informática encargada de trabajar a alto nivel con gran cantidad de datos.

La tarea de MDM no se limita simplemente a trabajar con datos hospedados en un único servidor. Estamos hablando de casos reales en los que es necesario disponer de múltiples servidores repartidos por la *web*, protegidos de forma que se necesite un usuario y sus credenciales para acceder a ellos.

La importancia de la gestión de datos maestros viene dada por la necesidad de una compañía de alinear estrategias e identificar áreas de crecimiento. Un sistema integral de gestión de datos puede ayudar a los gerentes y ejecutivos a tener la información que necesita, cuando la necesitan y de la forma en que la necesitan. Surge la necesidad de disponer de múltiples puntos de acceso datos para garantizar que, en caso de fallo en uno de un servidor de almacenamiento, la empresa evite perder la información y pueda acceder a ella redireccionándose a otro de sus servidores.

2 Objeto del proyecto

El presente proyecto busca crear un servicio para la empresa *arhis* que permita conectarse de forma rápida y segura a los servidores de la empresa *Informatica,* con la finalidad de facilitar y automatizar el tratamiento de cuentas de sus clientes. De esta forma se consigue que un usuario, introduciendo una única vez su nombre y contraseña desde un servidor, sea capaz tanto de autentificarse remotamente y acceder a los recursos hospedados en el otro servidor como de conseguir una autorización que le permita trabajar con ellos.

Informatica dispone de una herramienta, *Informatica Master Data Manager (Informatica MDM)* para la gestión de datos maestros. No obstante, en *Informatica MDM* se realiza dicho tratamiento de cuentas sin conexión remota: para cada nueva empresa cliente que solicite sus servicios, se deben crear nuevos usuarios directamente en el servidor de *Informática* que tengan los mismos roles y permisos que los usuarios del cliente.

Así pues, haciendo uso del sistema de automatización implementado en este proyecto, se evita ese trabajo y se agiliza el proceso de gestión de usuarios, pese a encontrarse en servidores distintos. Los usuarios de *arhis* ya pueden trabajar directamente con *Informatica MDM* de forma eficiente, con un inicio de sesión único o *Single Sign On* (SSO) que dota a la aplicación de una buena seguridad y que se explica en profundidad posteriormente.

Los detalles acerca de la actividad que llevan a cabo ambas empresas, así como la información relacionada con ellas, se exponen en el Anexo 17.1 Documentación de entrada.

3 Antecedentes

En la actualidad, son pocas las empresas que carecen de un sistema de informatización para gestionar su recurso. En cualquiera de estas compañías es típico trabajar con aplicaciones no disponibles para la misma, es decir, aplicaciones que se encuentran hospedadas en servidores externos, fuera de su alcance. Con el fin de acceder a estos recursos, es necesario el uso de usuarios y contraseñas para empezar un proceso de intercambio de información y datos entre las diferentes entidades. Así pues, del hecho de tener que utilizar un nombre de usuario y una contraseña para conectarse a cada entidad remota, surgen problemas tanto a nivel de seguridad (puesto que se aumentan el número de posibilidades de que se filtren los datos), como a nivel de administración (ya que resulta más difícil gestionar muchas cuentas diferentes).

La empresa, *arhis*, es una consultoría cuyo trabajo principal se centra en la gestión de datos maestros: preparación, normalización y depuración de los mismos. Trabajando estrechamente con ella nos encontramos a la empresa estadounidense *Informatica*. Situada en el mismo sector que *arhis*, *Informatica* le proporciona las herramientas necesarias para cumplir sus objetivos, convirtiéndose de esta forma en su principal empresa asociada o *partner*.

A pesar de las múltiples aplicaciones en MDM de *Informatica*, no existe funcionalidad de sincronización de cuentas, debiéndose crear usuarios y contraseñas nuevas para cada proyecto. De esta carencia surge la oportunidad de diseñar una mejora para *Informatica* que permita a los usuarios de *arhis* una conexión segura para verificar su identidad (autenticación) y poder trabajar con los recursos dependiendo de sus roles (autorización) de forma automatizada.

En el Anexo 17.2 “Estudios con entidad propia” se explica con más detalle la diferencia entre autorización y autenticación y su importancia para el proyecto.

4 Descripción de la situación actual

4.1 Descripción del entorno actual

Muchas son las compañías que necesitan integrar un sistema de sesión único para garantizar la seguridad de las cuentas de sus usuarios y evitar la múltiple creación de las mismas. Un claro ejemplo sería el caso de Google, la famosa multinacional estadounidense que posee más de un millón de servidores y centros de datos repartidos por todo el mundo. El hecho de poder aprovechar un usuario tanto para el correo como para el drive o incluso para el YouTube, supone una gran ventaja tanto para la empresa como para su cliente.

A lo largo de estos años se han desarrollado una serie de implementaciones similares a las que se presentan en este proyecto con el objetivo de conseguir la autenticación y autorización de los usuarios para acceder a los recursos de los diferentes servidores y poder trabajar con ellos. Partimos, pues, de una estructura como la que se muestra en la *Figura 1,* que presenta un esquema básico de una red con inicio de sesión único SSO que garantiza la seguridad de los usuarios.

En el siguiente subapartado se muestran algunos casos de aplicación basados en este tipo de seguridad, así como un análisis comparativo de sus atributos más relevantes, tanto a nivel cuantitativo como cualitativo, que han ayudado en la toma de decisión respecto a los componentes que han pasado a formar parte del desarrollo e implementación del proyecto.

Figura 1. Topología de una red con SSO

- Cliente: Usuario que intenta acceder al recurso o servicio.
- Proveedor de identidad o *Identity Provider* (IdP): Proveedor de información sobre la identidad de los usuarios.
- Base de Datos o *Data Base* (DB): Almacenan de nombres y contraseñas del IdP
- Proveedor del servicio o *Service Provider* (SP): Servicios online que se le ofrecen a un cliente. Utiliza la información del IdP.

Como ejemplos de servicios online tendríamos cualquiera que incorpore un SSO, como *Dropbox*, *Google* y *Saleforce*. Así pues, algunos proveedores de identidad más conocidos serían *Heroku* o *Microsoft Azure*. *Active Directory* (AD) o *Lightweight Directory Access Protocol* (LDAP) se utiliza en la mayoría de casos como base de datos para respaldar dichos IdP.

Se habla de identidad federada cuando se permite el desacoplamiento de la autenticación y autorización, facilitando a los usuarios el uso de sus propias credenciales y el aprovechamiento los roles ya existentes. Se facilita la existencia de un almacén de identidad, el *Identity Provider,* para la verificación de dichos usuarios o se trabaja directamente con una tercera identidad de confianza, reduciendo la lógica de mantenimiento de credenciales y soportando un solo tipo de *back-end* corporativo (al ser el sistema respaldado por una base de datos LDAP o AD).

Existen tres protocolos principales para llevar a cabo este enfoque de identidad federada, que proveen a la aplicación de autorización, autentificación o ambas según convenga: Auth0, SAML o OpenID. Basándose tanto en la información anterior como en la recopilada en el apartado 17.3 Anexo 17.2 “Estudios con entidad propia” (donde se entra en detalle sobre los términos de autenticación y autorización) y teniendo en cuenta algunos términos detallados en el apartado 6.1 “Definiciones” (como el de identidad federada), se ha decidido clasificar los sistemas según el protocolo utilizado.

4.1.1 Sistema 1. OpenID

OpenID es un estándar de identificación digital descentralizado, con el que un usuario puede identificarse en una página web a través de un Localizador Uniforme de Recuros o *Uniform Resource Locator* (URL). En los sitios que soporten OpenID, los usuarios no tienen que crearse una nueva cuenta de usuario para obtener acceso. En su lugar, solo necesitan disponer de un identificador creado en un servidor que verifique OpenID, un *Identity Provider*.

En la actualidad hay muchas cuentas habilitadas por este protocolo, como por ejemplo *Facebook*, *Microsoft*, *Google*, *PayPal* o *Yahoo*. Los usuarios pueden elegir utilizar sus proveedores OpenID preferido, para iniciar sesión en sitios web que aceptan el esquema de autenticación OpenID (como *Facbook*, que acepta una cuenta de *Google* como inicio de sesión).

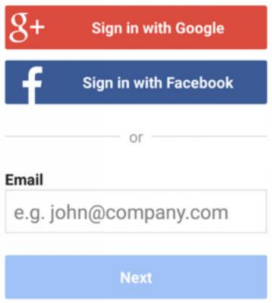


Figura 2. Single Sign On con OpenID

En una página web, el intercambio se vería como en la *Figura 2.* A diferencia de otras arquitecturas *Single Sign On*, OpenID no especifica el mecanismo de autenticación. Por lo tanto, la seguridad de una conexión OpenID depende de la confianza que tenga el cliente OpenID en el proveedor de identidad. Así pues, la especificación de OpenID define tres funciones:

- El usuario final o la entidad que busca verificar su identidad.
- La parte de confianza, que es la entidad que busca verificar la identidad del usuario final.
- El proveedor OpenID, que es la entidad que registra la URL de OpenID y puede verificar la identidad del usuario final Servicio entre identidades.

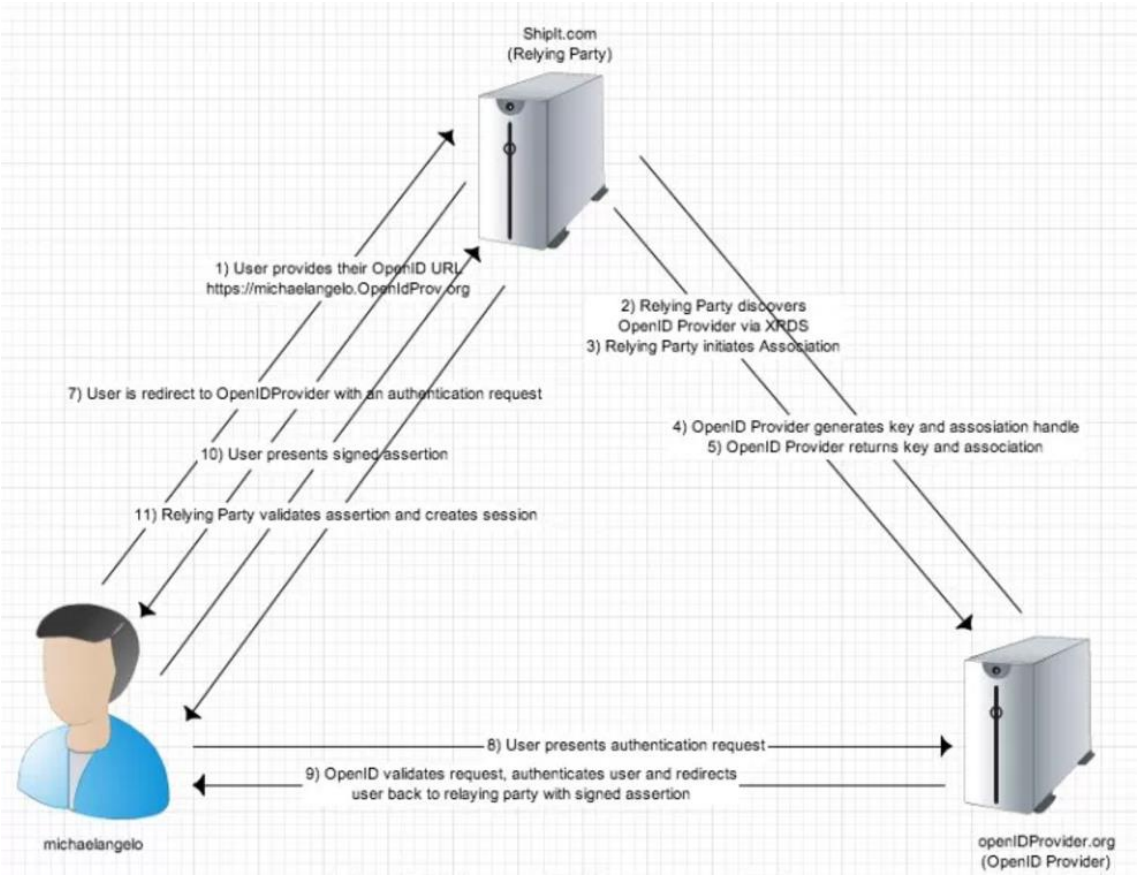


Figura 3. Flujo de trabajo de OpenID

Relacionado con el tema de seguridad, se debe remarcar que OpenID puede tener problemas de *phishing*: es posible que la parte de confianza envíe el usuario a un proveedor OpenID falso y se recopilen las credenciales del usuario.

4.1.2 Sistema 2. OAuth

A diferencia de los otros sistemas OAuth trabaja con la autorización, no con la autenticación. La empresa estadounidense *InCommon Federation* es proveedora de un marco de trabajo o *framework* para el intercambio seguro de accesos a recursos online. A través de *InCommon*, el *Identity Provider* puede dar a sus usuarios un SSO y protección de privacidad, mientras que los proveedores de servicios, *service providers* online controlan el acceso a sus recursos protegidos.

La estructura de OAuth definiría los roles que se muestran a continuación:

- El usuario final o la entidad que posee el recurso en cuestión.
- El servidor de recursos (OAuth Provider), que es la entidad que aloja el recurso.
- El cliente (OAuth Consumer), que es la entidad que busca consumir el recurso después de obtener la autorización del cliente.

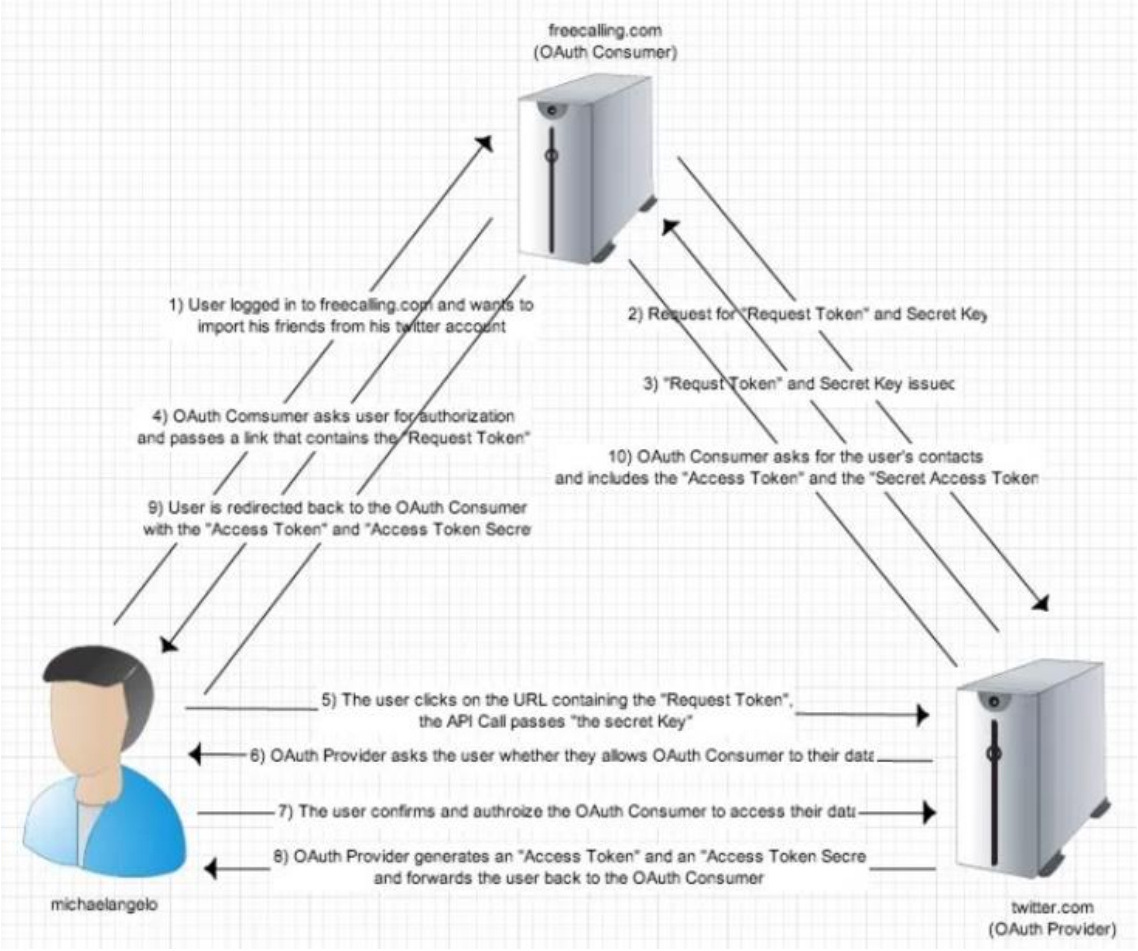


Figura 4. Flujo de trabajo de OAuth

Este sistema, no obstante, también tiene fallos de seguridad. Un atacante es capaz de fijar un *token,* y utilizarlo para robar las credenciales. OAuth no admite firma, cifrado ni cana de vinculación o verificación de clientes (autenticación). El protocolo se basa exclusivamente en la seguridad de la capa de transporte (por ejemplo, un *Secure Socket Layer* SSL o *Transport Layer Security* TLS) para proporcionar confidencialidad e integridad.

4.1.3 Sistema 3. SAML

El SAML (*Security Assertion Markup Language*) es un producto del Comité Técnico de Servicios de Seguridad de OASIS. Datado de 2001, SAML es un estándar abierto basado en el lenguaje de marcado XML para intercambiar datos de autenticación y autorización entre las partes.

La especificación SAML define tres roles:

- El director, que normalmente es el usuario que busca verificar su identidad.
- El proveedor de identidad (idP), que es la entidad que es capaz de verificar la identidad del usuario final.
- El proveedor de servicios (SP), que es la entidad que busca utilizar el proveedor de identidad para verificar la identidad del usuario final.

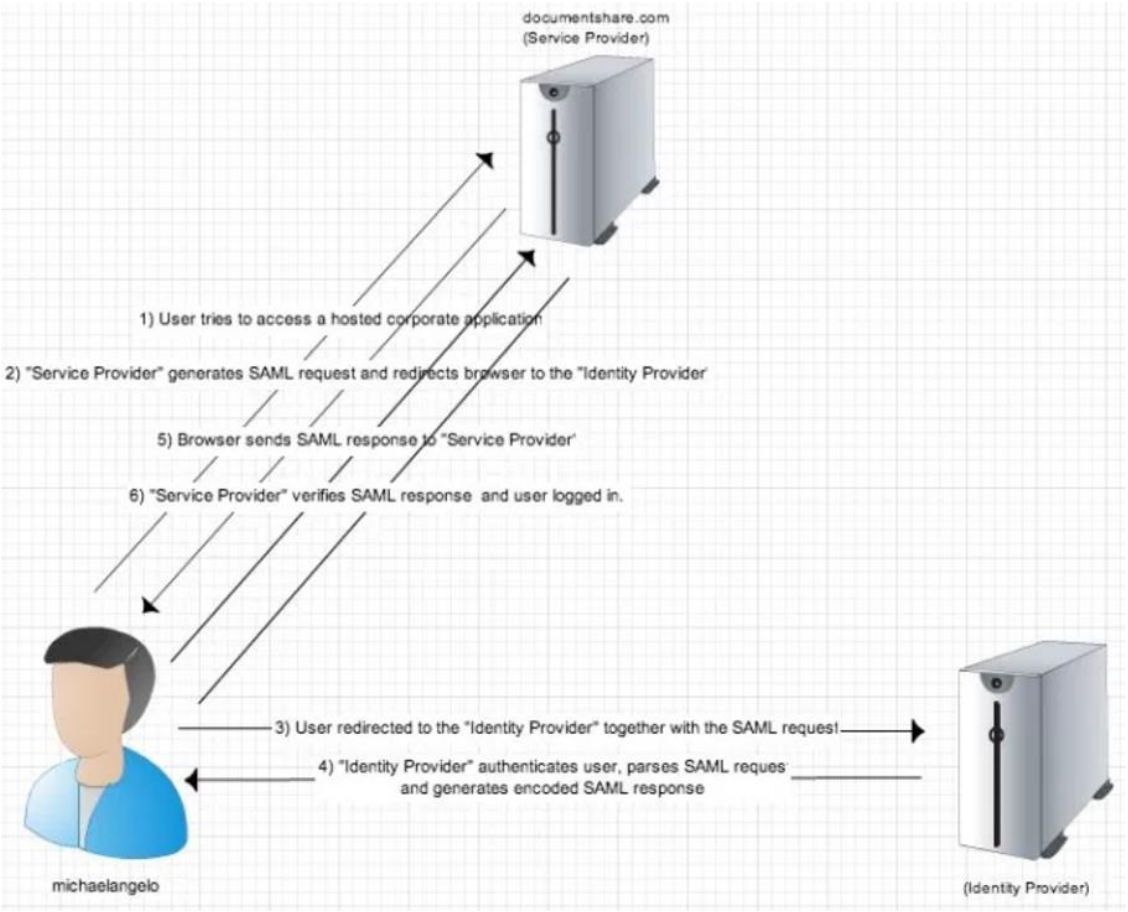


Figura 5. Flujo de trabajo de SAML

Un grupo de investigadores presentó un documento en 2011 donde utilizaron una vulnerabilidad de envoltura de firmas de XML para suplantar a cualquier usuario. Este sistema dispone de una lista de información registrada sobre vulnerabilidades de seguridad, las llamadas *Common Vulnerabilities and Exposures* (CVE) mucho más larga que su predecesor.

4.2 Resumen de las características

4.2.1 Análisis cuantitativo

A continuación, se muestra una tabla con las características medibles fácilmente (objetivas):

	Sistema 1. OpenID	**Sistema 2. OAuth**	**Sistema 3. SAML**
Protocolos utilizados	XRDS, HTTP	JSON, HTTP	SAM, XML, HTTP, SOAP
Finalidad principal	SSO para clientes	API de Autorización entre aplicaciones	SSO para usuarios de empresas
Versión actual	OpenID 2.0	OAuth 2.9	SAML 2.0
Fecha de creación	2005	2006	2001
Nº de vulnerabilidades (CVEs) relacionadas	24	3	19

Tabla 1. Análisis cuantitativo de los sistemas

4.2.2 Análisis cualitativo

En este subapartado se clasifican las características subjetivas:

	Sistema 1. OpenID	**Sistema 2. OAuth**	**Sistema 3. SAML**
Autenticación	Si	No	Si
Autorización	No	Si	Si
Nivel de seguridad	Bajo	Alto	Medio
Compatible con Linux	Si	Si	Si
Facilidad de manejo	Medio	Bajo	Alto
Facilidad de aprendizaje	Medio	Bajo	Alto
Gratuito	Si	Si	Si

Tabla 2. Análisis cualitativo de los sistemas

4.2.3 Conclusiones

En un mundo con una mayor interconectividad entre sistemas híbridos, protocolos y dispositivos, la identidad federada parece estar aquí para quedarse. Aunque la identidad federada es mucho más conveniente para los usuarios que no tienen que recordar tantos nombres de usuario y contraseñas diferentes, como es el caso de este proyecto, viene con un precio de seguridad. Así pues, la implementación adecuada de OAuth, SAML, OpenID o cualquier otro protocolo de identidad federada agrega comodidad sin superficie de amenaza adicional.

5 Normas y referencias

5.1 Disposiciones legales y normas aplicadas

En este apartado se han identificado las normas y reglamentos que han sido de aplicación en la elaboración este proyecto y la ejecución del mismo. Puesto que se ha trabajado en un marco de seguridad informática, las leyes están relacionadas directamente con la protección de datos e identidades, así como la propiedad intelectual relacionada con la redacción de la memória. Así pues, se citan algunas normativas según el Boletín Oficial del Estado (BOE).

"*La presente Ley Orgánica tiene por objeto garantizar y proteger, en lo que concierne al tratamiento de los datos personales, las libertades públicas y los derechos fundamentales de las personas físicas, y especialmente de su honor e intimidad personal y familiar*" (Ley de Protección de Datos de Carácter Personal, BOE núm. 298, de 14 de diciembre de 1999, páginas 43088 a 43099).

"*Los derechos al honor, a la intimidad personal y familiar y a la propia imagen tienen el rango de fundamentales, y hasta tal punto aparecen realzados en el texto constitucional que el artículo veinte, cuatro, dispone que el respeto de tales derechos constituya un límite al ejercicio de las libertades de expresión que el propio precepto reconoce y protege con el mismo carácter de fundamentales*" (Ley de protección civil del derecho al honor, a la intimidad personal y familiar y a la propia imagen, BOE núm. 115, de 14 de mayo de 1982, páginas 12546 a 12548).

"*Se prevé la anotación del nombre o nombres de dominio de Internet que correspondan al prestador de servicios en el registro público en que, en su caso, dicho prestador conste inscrito para la adquisición de personalidad jurídica o a los solos efectos de publicidad, con el fin de garantizar que la vinculación entre el prestador, su establecimiento físico y su "establecimiento" o localización en la red, que proporciona su dirección de Internet, sea fácilmente accesible para los ciudadanos y la Administración pública. La Ley establece, asimismo, las obligaciones y responsabilidades de los prestadores de servicios que realicen actividades de intermediación como las de transmisión, copia, alojamiento y localización de datos en la red*." (Ley de servicios de la sociedad de la información y el comercio electrónico, BOE núm. 166, de 12 de julio de 2002, páginas 25388 a 25403).

"*Se entenderá por obras, para los efectos de la Ley de Propiedad Intelectual, todas las que se producen y puedan publicarse por los procedimientos de la escritura, el dibujo, la imprenta, la pintura, el grabado, la litografía, la est Ley de Propiedad Intelectual, ampación, la autografía, la fotografía o cualquier otro de los sistemas impresores o reproductores conocidos o que se inventen en lo sucesivo*" (Ley de propiedad intelectual, BOE núm. 250, de 6 de septiembre de 1880, páginas 763 a 766, Artículo 1).

5.2 Métodos, herramientas, modelos, métricas y prototipos

5.2.1 Métodos y herramientas

Con el fin de gestionar adecuadamente el proyecto y facilitar su implementación, se han seguido una serie de procedimientos y metodologías. En este apartado se enumeran, además, las herramientas que han facilitado el proceso, así como los recursos necesitados para el mismo.

5.2.1.1 Procedimiento

Inicialmente se ha estimado el tiempo del desarrollo del proyecto y se han listado las herramientas y programas que se han considerado importantes.

A continuación, se ha procedido a la búsqueda de información relacionada con el tema de comunicación segura entre servidores. Una vez estructurada dicha información, se han hecho estudios y comparativas y se ha dado inicio a la implementación del proyecto a partir de los resultados.

A lo largo de dicho desarrollo han surgido problemas y dudas, que han sido resueltos a base de nuevas búsquedas y comparativas. Se ha tenido acceso a todo tipo de fuentes, ya sean digitales o escritas. En caso de haber detectado algún error de implementación, se ha procedido a restaurar las máquinas virtuales (Véase el apartado 6.1 “Definiciones” sobre la Máquina Virtual para más información).

5.2.1.2 Recursos

Los recursos materiales utilizados en este TFG han sido bastante reducidos. El papel principal lo ha desempeñado el ordenador, recurso material imprescindible tanto para la búsqueda de información como para el diseño y desarrollo del proyecto. También han sido de ayuda libros relacionados con el tema (véase el apartado 17 Estudio con entidad propia: Referencias).

En cuanto a los recursos humanos, se ha obtenido información tanto a nivel práctico como teórico por parte de los tutores del proyecto, siendo el estudiante el que ha llevado a cabo el desarrollo principal.

5.2.1.3 Herramientas

En este apartado se mencionan a grandes rasgos las herramientas empleadas en el desarrollo e implementación del proyecto. Para empezar y con el objetivo de definir la estructura y organización del proyecto, se ha hecho uso de los siguientes programas:

- ***GanttProject***. Tal y como ya indica su nombre, esta herramienta se ha empleado para el diseño de un diagrama de Gantt con la planificación del proyecto. El diagrama se ha ido modificando a lo largo del desarrollo.

Figura 6. Logotipo de GanttProject

- ***yEd Graph Editor***. Programa para creación de diagramas y esquemas. Se ha utilizado para diseñar algunas de las figuras que se ven a lo largo de la memoria, como por ejemplo el diagrama de flujo del proyecto.

Figura 7. Logotipo de yED Graph Editor

- ***Virtual SVN Server.*** Es un Cliente *Subversion* para Windows, utilizado tanto en pequeñas como grandes empresas. Esta herramienta ha sido ideal para la creación del repositorio donde se encuentra guardada toda la información del proyecto (herramientas, programas, documentación, etc.).

Figura 8. Logotipo de Visual SVN Server

- ***Tortoise SVN.*** Es un sistema de control de versiones que permite manejar archivos y directorios localizados en un repositorio central. Para este proyecto, se ha creado un repositorio en el SVN Server mencionado en el punto anterior y se ha gestionado en Windows con esta herramienta. A continuación se enumeran sus características más relevantes para el proyecto:
  - Comprobación de cambios de datos y recuperación de versiones antiguas de ficheros
  - Iconos que muestran archivos/directorios que necesitan ser enviados al repositorio (para actualización y sincronización)
  - Muestra de diferencias entre documentos *Office*
  - Facilidad de acceso a comandos de *Subversion*

Figura 9. Logotipo de Tortoise SVN

- ***Oracle VM VirtualBox***. Es el *software* más popular de virtualización (máquinas virtuales) del mundo. Destaca por su gran potencia y por tener versiones tanto para Windows como para Linux o Mac OS, entre otros.
  Esta herramienta permite, además:
  - Mas de un Sistema Operativo (SO) en el mismo escritorio
  - Clientes y servidores multi- plataforma
  - Soporta cargas de trabajo de más de 32CPUs virtuales (alto rendimiento)
  - Teleportacion de la ejecución de las VM entre servidores sin interrupción
  - OVF (open virtualization) to package and deploy virtualized systems
  - Permite arrancar VMs de forma remota, utilizando un protocolo de escritorio remoto (RDP)
  - Da la posibilidad de montar imágenes ISO como un CD o DVD virtual
  - Es imposible saber si una máquina es virtual o no al revisar las conexiones de la red

Figura 10. Logotipo de Virtual Box de Oracle

- ***CygWin.*** Es un conjunto de herramientas GNU y de código abierto para Microsoft Windows que proporcionan funcionalidad similar a los sistemas UNIX. Es una DLL que proporciona funcionalidad API POSIX. Su objetivo es llevar software que ejecute el sistema POSIX a Windows con una recompilación de sus fuentes. Por ejemplo, si se conecta a una computadora Linux desde un Windows, Cygwin se encarga de "adaptar" el teclado, de modo que se puedan reconocer caracteres especiales, entre otras cosas.

Figura 11. Logotipo de CygWin

- ***Java JDK.*** JDK (*Java development kit*) es un software que proporciona un conjunto de programas y bibliotecas para el mantenimiento y desarrollo (compilación, ejecución, generación de documentación, etc.) de aplicaciones y applets en Java. Podemos ver las características principales a continuación.
  - JRE (Java Runtime Environment). Es necesario ejecutar código Java
  - Java como intérprete (cargador para aplicaciones Java)
  - Un compilador javac
  - Javadoc como el generador de documentación (genera automáticamente la documentación del código fuente.
  - Es importante saber que JDK trabaja en la plataforma Java SE (Standard Edition)

  Para más información sobre Java, ir al apartado “6.1 Definiciones”

Figura 12. Logotipo de Java JDK

- ***Notepad ++.*** Editor de texto y código gratuito para Windows. Soporta muchos lenguajes naturales y de programación, como C #, C ++, Java, Javascript, LISP, HTML o XML. Tiene, además, gran cantidad de *plugins* que añaden características útiles. Algunas de sus características más relevantes se muestran a continuación.
  - Copia automática de los ficheros que no han sido guardados
  - Codificación en formatos ASCII, Unicode, UTF-8
  - Edición simultánea
  - Pantalla dividida y comparación de datos
  - Búsqueda y reemplazo de expresiones regulares

Figura 13. Logotipo de Notepad++

- ***Eclipse NEON.*** Es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el JDK de Java.

Figura 14. Logotipo de eclipse neon

- ***CPU-Z.*** Es una utilidad *freeware* (gratis) que recoge información de algunos de los principales dispositivos del propio sistema. No es necesario instalar CPU-Z, basta con descomprimir los archivos en un directorio y ejecutar el .exe, (archivo ejecutable). El programa no copia ningún fichero en ningún directorio Windows, ni escribe en el registro.

Figura 15. Logotipo de CPU-Z

5.2.2 Modelos, métricas y prototipos

En este apartado se contemplan los métodos utilizados para desarrollar los cálculos y estimaciones del proyecto. Puesto que se pretende desarrollar un producto software, se ha utilizado el Modelo de Análisis de Puntos Función (FPA), una técnica de medición de funcionalidades. Se definen los componentes del producto software desarrollar en la tabla que se muestra a continuación.

Componente	Tipo de componente	Nivel de complejidad	Puntos de función
LDAP	Archivo lógico interno	Medio	5
SSO	Archivo externo Salida externa	Alto	12
MDM	Consulta interna	Medio	10
IDD	Entrada externa	Bajo	3
HA-Proxy	Consulta externa	Medio	9

Tabla 3. FPA de componentes funcionales del proyecto

También se han utilizado métricas para la identificación de los límites, la medición de funciones de datos, medición de funciones transaccionales, determinar y valorar la complejidad, calcular el tamaño funcional, etc. Estas mediciones se han utilizado en apartados posteriores para cálculos y estimaciones de recursos, costes y tiempos para realizar el proyecto. A continuación, se explica el cálculo realizado en la tabla.

5.2.2.1 Validación las especificaciones funcionales

IFPUG-FPA, el grupo internacional de usuarios de FPA, define 5 tipos de componentes de software que determinan su funcionalidad:

- Archivo lógico interno: El LDAP está implementado para actuar como base de datos con información sobre los usuarios.
- Archivo externo de interfaz: El SSO interactúa directamente con el LDAP y se encarga de comprobar dichos usuarios y realizar las correspondientes acciones.
- Entrada externa: En caso de validar las credenciales, el cliente accede a la página web IDD y utiliza sus credenciales para acceder al MDM.
- Salida externa: El SSO produce un valor agregado dependiendo de lo que conteste el LDAP.
- Consulta externa: Toma el contenido de la solicitud realizada por el cliente en la página web y la redirecciona a otro componente.

5.2.2.2 Medición del nivel de complejidad

Otra aplicación de los puntos de función incluye hacer un cálculo sobre la complejidad de cada componente. Para ello se han tenido en cuenta las características de los mismos: accesos a información, información que tienen, comparaciones que hacen, acciones de redirección e incluso recursos que consume cada componente. En apartados posteriores se entra en más detalle sobre la funcionalidad de cada componente.

5.2.2.3 Asignación de puntos de función según el nivel de complejidad

Por otra parte, los puntos de función tienen aplicaciones en mediciones de productividad. Esta asignación se ha realizado en función a la planificación temporal del proyecto, al estar directamente relacionado con el tiempo invertido en el desarrollo de cada componente. Otros indicadores interesantes son número de incidencias por puntos de función producidos como medición de la calidad del proceso de desarrollo. Aunque es interesante tener una visión general.

Teniendo en cuenta un margen de error a la hora de calcular los puntos de función, se ha reajustado el valor de los mismos:

- Puntos de función no ajustados: **39**
- Puntos de función con un ajuste de un 10%: +-**3 -->** un valor del intervalo **[36, 42]**.

5.2.2.4 Estimación de horas a partir de los puntos de función

Con esto, adicionalmente, se puede calibrar el factor de conversión entre puntos función y horas de trabajo invertidas. Se podría considerar, por ejemplo, que un punto equivale a unas 10h. La duración del proyecto sería entre **360h y 420h**.

No obstante, como ya sea comentado, en el apartado 14 “Planificación temporal” ya se ha llevado un seguimiento del tiempo invertido en el proyecto, por lo tanto, se ha decidido no tener en cuenta este cálculo, porque para este proyecto resulta impreciso.

6 Definiciones y abreviaturas

6.1 Definiciones

En este apartado se han añadido una serie de definiciones que ayudan al lector a entender la terminología empleada en el desarrollo de la memoria.

- ***Master data Management*:** MDM es un método que permite a una empresa vincular todos sus datos críticos a un archivo, denominado archivo maestro, que proporciona un punto de referencia común. La gestión de datos maestros agiliza el intercambio de datos entre el personal y los departamentos. Además, puede facilitar la computación en múltiples arquitecturas de sistemas, plataformas y aplicaciones.

- ***Structured Query Language:*** SQL es un lenguaje de programación estandarizado utilizado para administrar bases de datos relacionales y realizar diversas operaciones sobre los datos en ellas. Es utilizado regularmente por administradores de bases de datos, así como por desarrolladores que escriben *scripts* (un archivo de órdenes) de integración de datos e incluso analistas, que realizan consultas analíticas.

- **Base de datos**: Es una colección de información organizada para proporcionar una recuperación eficiente. La información recopilada puede estar en cualquier número de formatos (electrónicos, impresos, gráficos…). En este contexto, es un sistema computarizado que facilita la búsqueda, selección y almacenamiento de información. Podría ser tan simple como un arreglo alfabético de nombres en una libreta de direcciones o tan complejo como una base de datos que proporciona información en una combinación de formatos. Dependiendo de cómo se trabaje con los datos, las DB se clasifican como sigue:
  - Bases de datos SQL: Jerárquica (por ejemplo, el sistema de directorios de Windows), empresarial, en memoria y embebida.
  - Base de datos NoSQL: BD *Key-value*, BD Grafo, etc.

- ***Bussines to Bussines***. El marketing B2B es aquel en el que una empresa vende a otra empresa y no al consumidor final. Es decir, son todas las empresas que crean productos o servicios para que sean consumidos por otras empresas, siendo estas las que satisfacen finalmente al consumidor final

- **Oracle:** Oracle *Database* es la primera base de datos diseñada para la computación en malla (*grid computing)* empresarial, la forma más flexible y rentable de administrar la información y las aplicaciones. El *grid computing* de la empresa crea grandes grupos de almacenamiento modular y servidores estándar de la industria. Con esta arquitectura, cada nuevo sistema se puede aprovisionar rápidamente desde el conjunto de componentes. No hay necesidad de cargas de trabajo máximas, ya que la capacidad se puede agregar o reasignar fácilmente desde los grupos de recursos según sea necesario. La base de datos tiene estructuras lógicas y estructuras físicas separadas: el almacenamiento físico de datos se puede gestionar sin afectar el acceso a las estructuras de almacenamiento lógico.

Figura 16. Grid computing en Oracle

- **Java:** Es un lenguaje de programación multiplataforma que permite utilizar programas, herramientas, juegos y aplicaciones y se desarrolla para todo tipo de dispositivos: computadoras, móviles, agendas, etc. Java presenta una arquitectura simple, orientada a objetos, distribuida, interpretada, robusta, segura, neutral, portátil, de alto rendimiento, multitarea y dinámica. Además, al ser independiente de la plataforma, puede ejecutarse en cualquier computadora del mercado. Se ha creado una máquina Java para cada sistema que actúa como puente entre el sistema operativo y el programa Java, evitando así tener que desarrollar un programa diferente para cada OS.

- **Active Directory:** AD es un servicio de Microsoft que implementa un servicio de directorio en una red distribuida de computadores. Utiliza distintos protocolos, principalmente LDAP, *Domain Name System* (DNS), *Dynamic Host Configuration Protocol* (DHCP) y Kerberos. De forma sencilla se puede decir que es un servicio establecido en uno o varios servidores en donde se crean objetos tales como usuarios, equipos o grupos, con el objetivo de administrar los inicios de sesión en los equipos conectados a la red, así como también la administración de políticas en toda la red. Su estructura jerárquica permite mantener una serie de objetos relacionados con componentes de una red, como usuarios, grupos de usuarios, permisos y asignación de recursos y políticas de acceso.

- **Máquina Virtual:** Una VM es un software que imita las acciones de una *Central Processing Unit* (CPU) u otros dispositivos de hardware en la utilización de los recursos de una computadora y reside como un espacio de memoria protegido, usualmente en un disco duro. Por ejemplo, en el contexto de una Máquina Virtual Java, ejecuta applets Java en un navegador web. Mientras que diferentes plataformas informáticas pueden requerir JVM diferentes (debido a sistemas operativos diferentes), todos ellos pueden ejecutar código Java independientemente del programa fuente (el navegador) que se esté utilizando.

- **Identidad federada**: Significa enlazar y usar las identidades electrónicas que un usuario tiene a través de varios sistemas de administración de identidad. Una aplicación no necesita necesariamente obtener y almacenar las credenciales de los usuarios para autenticarlos. En su lugar, se puede utilizar un sistema de gestión de identidad que ya almacena la identidad electrónica de un usuario para autenticar al usuario.

- ***Open Source*:** En general, el código abierto se refiere a cualquier programa cuyo código fuente esté disponible para su uso o modificación como los usuarios u otros desarrolladores lo consideren apropiado. El modelo de definición de términos de distribución requiere que:
  - El software que se distribuye debe ser redistribuido a cualquier otra persona sin ninguna restricción.
  - El código fuente debe estar disponible (para que el receptor pueda mejorarlo o modificarlo).
  - La licencia puede requerir versiones mejoradas del software para llevar un nombre o una versión diferente del software original.

- ***Supplier Relationship Management***. La gestión de las relaciones con los proveedores SRM (suministradores, acreedores) describe los métodos y procesos de una empresa o una institución que compra para establecer relaciones positivas con otra empresa que le provee bienes o servicios diversos. Esto puede ser para la compra de suministros de uso interno, la compra de materias primas para el consumo durante el proceso de fabricación, o para la adquisición de bienes de inventario para ser revendidos como productos en la distribución y venta al por menor.

- **HTTP:** *HyperText Transfer Procolol* (HTTP), es el protocolo de transferencia de hipertexto utilizado por la World Wide Web (www) y que define cómo se formatean y se transmiten los mensajes y qué acciones deben realizar los servidores y navegadores web en respuesta a varios comandos. Por ejemplo, cuando se ingresa una URL en el navegador, esto realmente envía un comando HTTP al servidor Web dirigiéndolo para buscar y transmitir la página Web solicitada.

- **HTTPS.** Es un HTTP seguro (*secure* http). Aparece en la URL cuando un sitio web está protegido por un certificado SSL. Los detalles del certificado, incluyendo la autoridad emisora y el nombre corporativo del propietario del sitio web, se pueden ver haciendo clic en el símbolo del candado en la barra del navegador

- **Servlet:** Es una clase que responde a un tipo particular de petición de red, la mayoría de las veces una solicitud HTTP. Los *servlets* se usan generalmente para implementar aplicaciones web. Se ejecutan en un contenedor de *servlets* que maneja el lado de red (por ejemplo, analizar una solicitud HTTP, manejo de conexión, etc.). Uno de los contenedores de *servlets* de código abierto más conocidos es Tomcat.

- **Tomcat:** es un servidor web de código abierto y un contenedor *servlet* de Apache. Permite que el código *Java Server Pages* (JSP) se ejecute, así como Java *Servlets*. Tomcat incluye tres componentes para sus diferentes funciones: Coyote (Conector HTTP), Catalina (contenedor *servlet*) y Jasper (motor JSP). Tomcat tiene una huella de memoria más ligera (entre 60MB y 70 MB) que otros servidores, como *JBoss*.

- **JBoss*:*** Es una división del sistema operativo Red Hat que proporciona soporte para el programa de servidor de aplicaciones Java EE de código abierto y servicios de *middleware* relacionados, comercializados bajo la marca JBoss Enterprise Middleware.

JBoss es una alternativa de código abierto a las ofertas comerciales de IBM WebSphere y SAP *NetWeaver*.

- **Ethernet:** En su definición estándar, ethernet es una norma o estándar que determina el protocolo o la conexión de una red. La idea base de esta norma consiste en que todos los ordenadores que se encuentran dentro de una red reciban y envíen datos de tal manera que no se superpongan. Por esta razón los datos que se envían o se reciben a través de esta norma se deben dividir en fracciones más pequeñas y deben ser enviados mediante lo que se denomina "conmutación de paquetes".

- **RSA *layout***: En criptografía, RSA (Rivest, Shamir y Adleman) es un sistema criptográfico de clave pública desarrollado en 1977. Es el primer y más utilizado algoritmo de este tipo y es válido tanto para cifrar como para firmar digitalmente.

Figura 17. Ejemplo de encriptación RSA

6.2 Abreviaturas

Dado su uso frecuente en la definición del proyecto, se ha añadido una lista con abreviaturas con el fin de facilitar su significado.

Abreviatura	**Significado**
TFG	Trabajo de fin de curso
BOE	Boletín Oficial del Estado
FPA	Análisis de Puntos Función
DBM	*Data Base Management*
MDM	*Master data management*
DB	*Data Base*
SQL	*Structured Query Language*
SSO	*Single Sign On*
DN	*Distingued Name*
DNS	*Domain Name System*
DHCP	*Dynamic Host Configuration Protocol*
VM	*Virtual Machine*
LDAP	*Lightweight Directory Access Protocol*
IDD	*Informatica Data Director*
SVN	*Subversion*
SSH	*Secure SHell*
HTTP	*Hyper Text Transfer Protocol*
HTTPS	*Secure Hyper Text Transfer Protocol*
AD	*Active Directory*
IDaaS	*Identity as Service*
UID	*User* ID
URL	*Uniform Resource Locator*
IDP	*Identity Provider*
SSL	*Secure Socket Layer*
SAML	*Security Assertion Markup Language*
TLS	*Transport Layer Security*
CPU	*Central Processing Unic*
SRM	*Supplier Relationship Management*
CVE	*Common Vulnerabilities and Exposures*
B2B	*Bussines to Bussines*
RSA	Rivest, Shamir y Adleman
CRM	*Customer Relationship Management*
ERP	*Enterprise Resource Planing*
IP	*Internet Protocol*
IDQ	*Informatica Data Quality*
WWW	*World Wide Web*

Tabla 4. Abreviaturas

7 Requisitos iniciales

Este proyecto está formado por cuatro productos y sus respectivos servicios. La entrega de los mismo se ha realizado de forma progresiva según el cronograma establecido en el apartado 14 "Planificación temporal".

Se ha realizado un estudio previo al diseño del proyecto en el que se han recopilado los requisitos del mismo. Han sido divididos en dos tipos según su función:

- **Funcionales:** Cómo debe funcionar cada uno de los componentes.
  "Característica requerida del sistema que expresa una capacidad de acción del mismo – una funcionalidad; generalmente expresada en una declaración en forma verbal."
- **No funcionales:** Características genéricas de *performance* del componente final.
  "Característica requerida del sistema, del proceso de desarrollo, del servicio prestado o de cualquier otro aspecto del desarrollo, que señala una restricción del mismo."

7.1 Requisitos funcionales

Partiendo del análisis FPA realizado en el apartado 5.2.2 "Modelos, métricas y prototipos", se enumeran los requisitos funcionales en función de cada componente que forma el proyecto. A continuación, se muestran dichos requisitos recopilados en una tabla. Se entra en más detalle en el Anexo 17.3 "Análisis y diseño del sistema".

Producto	Requisitos funcionales
Implementación del LDAP	• Almacenamiento. DB con usuarios, contraseñas y roles • Sistema de seguridad propio • Conexión directa con *HAProxy* • Conexión indirecta con el resto de componentes
Implementación del SSO	• Página web de *Log In* • Sistema de comparativa de credenciales • Conexión directa con *HAProxy* • Conexión indirecta con el resto de componentes
Implementación de IDD y MDM	• Tratamiento de usuarios • Conexión entre IDD y MDM • Conexión directa con *HAProxy* • Conexión indirecta con el resto de componentes
Implementación del *HAProxy*	• Sistema de seguridad (tratamiento de *headers*) • Redireccionamiento de componentes (depende del *header*) • Conexión directa con todos los componentes

Tabla 5. Requisitos funcionales

7.2 Requisitos no funcionales

En este subapartado se entra en detalle sobre el producto final. Una vez integrados todos los productos, se cumplen los siguientes requisitos no funcionales:

- **Accesibilidad:** Al tratarse de una aplicación web, es accesible para la mayoría de personas. Para la aplicación que se le va a dar, los usuarios únicamente necesitan el ordenador. Además, las páginas web tienen la opción de aumentar o disminuir el tamaño de la letra en caso de tener problemas oculares.
- **Seguridad:** Esta característica formaría parte también de los requisitos funcionales. Al estar implementado el sistema de seguridad del HAProxy, la aplicación final es capaz de utilizar un sistema de bloqueo de *headers*, que permite que solo los usuarios registrados accedan a los recursos remotos.
- **Usabilidad**: El objetivo es que la aplicación final sea fácil de usar. Así pues, el proyecto se apoya en las 10 reglas heurísticas de usabilidad de Jakob Nielsen. En el Annexo 17.2 "Estudios con entidad propia" se entra explican las reglas adaptadas a este proyecto. Se definen de forma sencilla debido a la poca interacción que debe tener el usuario con la aplicación.
- **Robustez:** El sistema está preparado de forma que, si se introduce algún carácter o dato incorrecto, se devuelva un mensaje de error claro. Puesto que los únicos parámetros entrada son el usuario y la contraseña, presenta una robustez 100% fiable en este aspecto. Aun así, pueden surgir errores en la ejecución de la aplicación, sobre todo relacionados con la base de datos, que también han sido tratados a lo largo de implementación del proyecto.
- **Rendimiento:** Para una buena optimización del proyecto, ha sido necesario estudiar y reducir al máximo la carga de trabajo a la que va estar sometido. Es por eso que ha habido que tener en cuenta los recursos que utiliza para no llegar ralentizar demasiado el sistema. En el Anexo 17.3 "Análisis y diseño del sistema" se explica en los dos primeros apartados, cómo funcionaría el proyecto según como se despliegue. De forma resumida, vemos que la mejor opción es desplegar la aplicación en varios servidores para repartir equitativamente la carga de los recursos a los que va estar sometida.

8 Alcance

Al final del proyecto la aplicación es capaz de sincronizar cuentas de usuarios de forma segura, desde los servidores de la empresa de *arhis* a un servidor remoto de *Informatica*. El sistema permite la retroalimentación de dichos usuarios *y* sus respectivos roles, gracias a la autenticación y autorización de sus cuentas cuando se realiza un inicio de sesión en la aplicación remota de *Informatica*.

Para facilitar la comprensión del proyecto, se ha decidido realizar un diagrama sobre el alcance del proyecto, la famosa *"Table Top Drawing*", que tiene las características que se ven a continuación.

- Representa el alcance de forma esquemática. Se sigue un avance lineal.
- Incluye el integrante de cada parte del proyecto, en este caso solo es el alumno.
- Cada nivel está marcado de un color distinto, con el objetivo de diferenciar las tareas y las subtareas

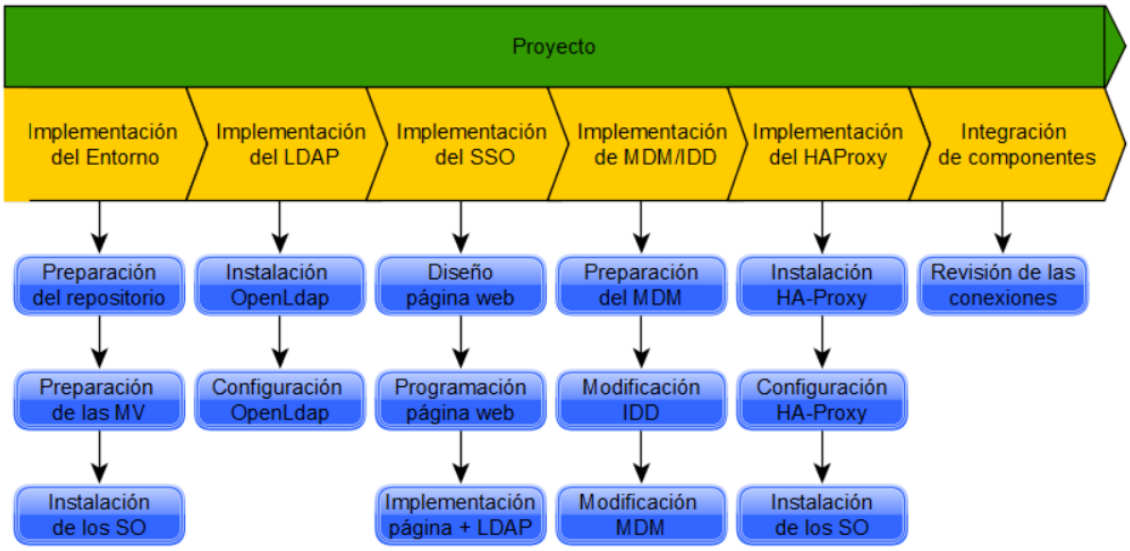


Figura 18. Alcance del proyecto

En los siguientes apartados se entra en detalle sobre la estimación de plazos y la estimación de costes del proyecto, entre otras cosas.

9 Hipótesis y restricciones

El proyecto surge de la necesidad de vincular cuentas de usuarios de la empresa de *arhis* con los servidores de *Informatica* a través de un sistema que proporcione seguridad. Se propone la implementación de un *Single Sign On* con *Active Directory* como base de datos con los usuarios que, mediante un intercambio de credenciales a través de protocolos HTTPS, ataque contra MDM y, después de esto, es capaz de validar y dar permisos a dichos usuarios.

Aunque no dista mucho del diseño del proyecto actual, las hipótesis y puntos de partida iniciales han variado a lo largo del desarrollo del proyecto, dando lugar al sistema que se explica en detalle en el apartado 11 “Descripción de la solución propuesta”. Algunos de los factores que han cambiado son, por ejemplo, el coste, los plazos de entregables y alguna funcionalidad.

Inicialmente se estipuló una fecha de inicio y de fin para el proyecto que han tenido que ser cambiadas debido a una serie de contratiempos surgidos en la empresa. En cuanto al coste del proyecto, se ha visto afectado positivamente. Al decidirse una implementación sobre un SO gratuito y, como consecuencia, la utilización de una base de datos *open source* como es LDAP, se ha reparado en gastos al ahorrarse tanto la licencia de *Windows* como la del AD.

Pese a la retroalimentación de los usuarios y sus credenciales, actualmente no es posible encontrar una relación rol en *arhis* – rol en *Informatica*. Si es cierto que un *admin*, es un *admin* en ambas partes, *Informatica* dispone de roles de los que *arhis* carece y viceversa. Este problema surgido a la hora de sincronizar los roles de los usuarios es, por tanto, la restricción principal de este proyecto. No obstante, dicha restricción queda aislada de forma indefinida, con la intención de mejorar la aplicación en un futuro para adaptarse mejor a las necesidades de la empresa ya que, por el momento, no es necesario sincronizar estos roles minoritarios. En el proyecto hay dos tipos de roles que es obligatorio gestionar: el *admin*, con permiso de super usuario y el *user*, con permiso de usuario.

10 Estudio de alternativas y viabilidad

En cuanto a alternativas, en los estudios realizados con anterioridad se han expuesto los diferentes sistemas de la actualidad y se han analizado sus ventajas e inconvenientes. Se podría desarrollar el proyecto para que se parezca de forma más precisa al comportamiento de los sistemas anteriores, siempre ciñéndose a cumplir el objetivo del proyecto.

Así pues, en este apartado se estudia la viabilidad general del proyecto, tanto a nivel técnico como económico y entrando dentro del marco legal (Véase el apartado 5.1 “Disposiciones legales y normas para más detalle).

10.1 Viabilidad económica

En el apartado 15 “Resumen del presupuesto” se ha calculado los márgenes de beneficio sobre el coste total, mostrando la rentabilidad del proyecto en relación con las horas invertidas y el beneficio que se obtiene de él. En cuanto al cliente, que en este caso es la propia empresa, interesa que el coste se ajuste en la medida de lo posible, aunque dependiendo de las partidas económicas que se le dedique a su área de I+D, no tendría por qué considerarse una limitación.

10.2 Viabilidad técnica

Vistos los requisitos funcionales expuestos en apartados anteriores, se puede decir que los recursos tecnológicos de la actualidad relacionados con el campo de la informática permiten abordar este proyecto desde el punto de vista de técnico. Los ordenadores ofrecen grandes tasas de rendimiento, puesto que los existentes actualmente suelen llevar como mínimo cuatro núcleos físicos (hablamos, por ejemplo, de dos núcleos lógicos cuando al núcleo físico se le instala un procesador *dual core*) y una tarjeta gráfica acorde a las necesidades de virtualización de las máquinas. El tema de rendimiento se explica con más detalle en el Anexo 17.3. Las herramientas y lenguajes utilizados tampoco han resultado un problema puesto que hay gran variedad a elegir acorde con las necesidades de cada parte de la implementación.

10.3 Viabilidad legal

Al tratarse de un software de empresa, hay que facilitar en la medida de lo posible los datos de usuarios y contraseñas para poder ejecutar la aplicación. No obstante, esto quedaría dentro del marco de trabajo de la empresa *arhis,* puesto que los usuarios van a ser los mismos trabajadores. Es necesario que el proyecto siga las normas y disposiciones legales establecidas y proteja la confidencialidad de la empresa, así como de sus trabajadores.

11 Descripción de la solución propuesta

El proyecto está diseñado sobre una red amplia a la que están conectados los diferentes servidores que intervienen en el proceso de paso de cuentas de usuario y contraseñas. Mediante el protocolo Secure *Hyper Text Transfer Protocol* (HTTPS), se realiza el envío de cabeceras (*headers)* y se realiza una comparación en base de datos auxiliar: o bien se establece una conexión segura desde *arhis* hasta los servidores de *Informatica* (en el hipotético caso de introducir los usuarios con credenciales correctas) para el intercambio de información o bien se rechaza la conexión y se redirecciona a otros servidores auxiliares.

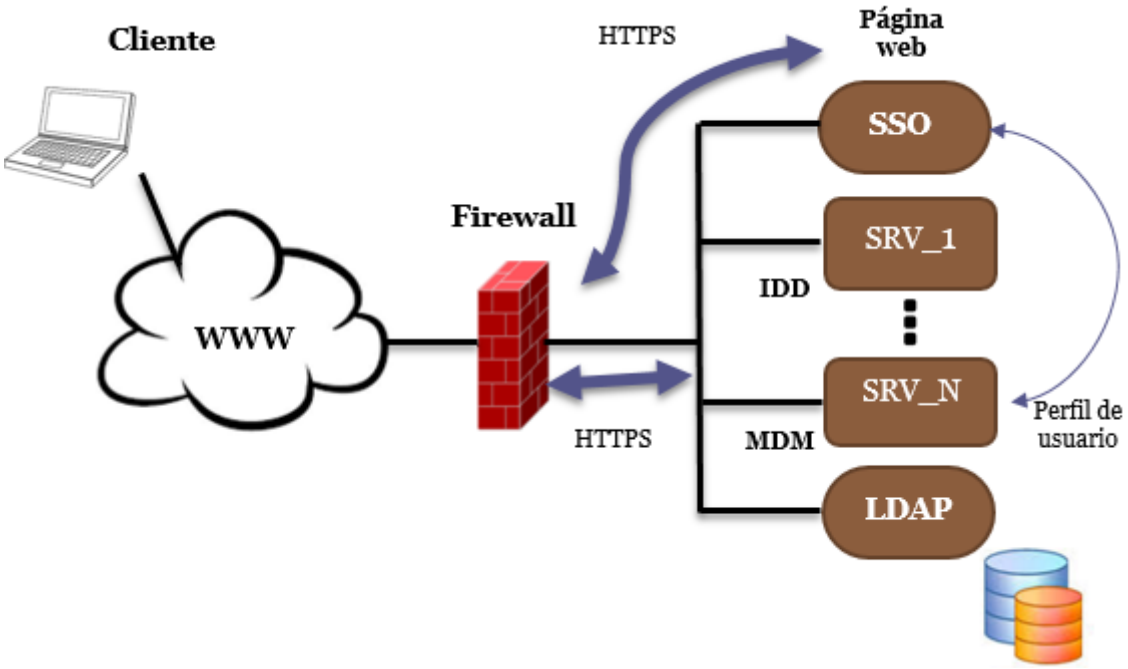


Figura 19. Topología de la red del proyecto

Para la conexión con *Informatica* el proyecto dispone de uno o dos servidores, en función del despliegue que la empresa elija. Aunque para ambos la solución es la misma, supondremos el segundo: sobre el primer servidor se ha desplegado la interfaz de usuario *Informática Data Director* (IDD), el portal desde el cual nuestros clientes pueden acceder a dicha empresa. En cuanto al segundo servidor, está diseñado para el gestor de datos maestros *Informática* MDM, base de datos en la que se encuentra la información sobre los usuarios y sus roles, entre otras cosas, a la cual se pretende acceder.

Con el fin de evitar los problemas derivados de la administración de múltiples cuentas de usuario de *arhis* para cada nuevo proyecto, se ha implementado, en forma de aplicación web, un sistema de control de acceso *Single Sign On*. El SSO es el encargado de intercambiar los perfiles de usuario con el gestor de datos maestros (MDM) de *Informatica* simulando un sistema de sesión único, haciendo uso de la información recibida a través de los *headers* mencionados anteriormente.

Otra de las claves de este proyecto es el uso de un *Lightweight Directory Access Protocol* (LDAP), un protocolo que permite el acceso a un servicio de directorio ordenado para buscar información sobre las cuentas en el entorno de la red (los distintos usuarios y sus roles), que actuará como una base de datos auxiliar. El LDAP trabaja junto con el SSO para autenticar a los clientes, mediante el intercambio de *tokens* (claves) temporales.

Para completar el proyecto se ha añadido un balanceador con un *firewall* (cortafuegos*)* que ofrece una protección y estabilidad adicional a la red y se encarga de redireccionar al usuario dependiendo de los *headers* que reciba en su solicitud (*request)*.

La solución debe estar disponible tanto para los empleados con presencia física en la oficina de la empresa, como para aquellos que trabajan remotamente, sin la sobrecarga de una conexión VPN, por lo que la aplicación debe desplegarse en un proveedor de la nube.

12 Análisis de riesgos

El siguiente paso ha consistido en la identificación de los riesgos a las que estos están expuestos los componentes del proyecto. El conjunto de amenazas es amplio y diverso por lo que se ha optado por un enfoque práctico y aplicado. Llegado a este punto y partiendo de que se dispone de los siguientes activos, vistos en el apartado anterior:

- Servidor con el LDAP
- Servidor del SSO
- Servidor con MDM (IDD incluido)
- Servidor HA-Proxy

Para cada uno se ha estimado la probabilidad de que la amenaza se materialice y el impacto sobre el negocio que esto produciría. Con el objetivo de tener una visión más general, a la hora de calcular el riesgo, se ha optado por hacer un análisis cualitativo, haciendo uso de una matriz de riesgo como la que se muestra a continuación:

		IMPACTO		
		Bajo	Medio	Alto
PROBABILIDAD	Baja	Muy bajo	Bajo	Medio
	Media	Bajo	Medio	Alto
	Alta	Medio	Alto	Muy alto

Tabla 6. Matriz cualitativa de cálculo de riesgo

Al estimar la probabilidad y el impacto se han tenido en cuenta las vulnerabilidades y salvaguardas existentes. Este proyecto, al estar desarrollado sobre diferentes servidores, se enfrenta al riesgo de que alguno de los servidores de sus activos pueda dejar de funcionar.

Aunque se ve que la caída de un servidor podría tener un impacto altamente negativo, existen soluciones de alta disponibilidad (para el caso del MDM, para el que *Informatica* tiene implementada dicha funcionalidad). Se considera entonces que el impacto será medio ya que estas medidas de seguridad harán que el servidor se recupere y, por tanto, se considera que el resto de procesos no se vean gravemente afectados.

Amenaza	**Riesgo**	**Probabilidad**	**Impacto**
Caída del servidor LDAP	Medo	Baja	Alto
Caída del servidor SSO	Medio	Baja	Alto
Caída del servidor MDM	Bajo	Baja	Medio
Caída del HA-Proxy	Medio	Baja	Alto

Tabla 7. Análisis de riesgos

No obstante, el impacto para el resto de los servidores sigue siento alto, puesto que no se ha estudiado la posibilidad de una solución de alta disponibilidad en este proyecto, al no ser un requisito para cumplir el objeto del proyecto, sino una mejora de optimización.

13 Gestión del proyecto

13.1 Gestión de requisitos

A partir de los requisitos iniciales, se ha establecido un sistema de gestión de los mismos. En función de las características demandadas por el proyecto, los requisitos han ido variando. Pese a que en este proyecto no ha surgido ningún problema, es importante saber cómo gestionar un requisito que pueda ser añadido o modificado posteriormente. Se debe tener en cuenta el esquema que se ve a continuación:

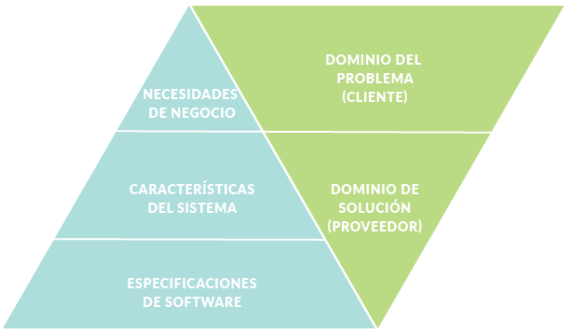


Figura 20. Gestión de requisitos

13.2 Gestión y Validación del diseño técnico y arquitectura del sistema

En cuanto al diseño y arquitectura del sistema, se ha hecho un seguimiento de la estructura del proyecto, así como de la configuración de los componentes. Para ello, se ha pasado una fase de diseño y análisis de los componentes.

1. Análisis de la situación actual. Se han analizado sistemas similares para contextualizar la idea del proyecto y hacer una recopilación de información.
2. Análisis de necesidades del proyecto. Se han tenido en cuenta los objetivos del proyecto y, a partir de esta información, se ha propuesto una arquitectura.
3. Validación del diseño. Se ha comprobado que la implementación propuesta es posible (y si no se han aplicado los cambios correspondientes) y que cumple con los objetivos propuestos.

13.3 Gestión de incidencias

A lo largo del desarrollo del proyecto se pueden tener dos tipos de incidencias: incidencias funcionales, a causa de un mal diseño de alguno de los componentes (error del diseñador/ programador) o incidencias técnicas causadas por el mal comportamiento de uno de sus componentes (error del *hardware*).

Cada incidencia estría clasificada, además, según su impacto. Para ello se utilizaría una tabla similar a la planteada en el apartado anterior 12 “Análisis de Riesgos”.

13.4 Gestión de riesgos

A lo largo del desarrollo de cualquier proyecto siempre surgen riesgos. Aunque se supone que se deben contemplar desde un principio, una cosa es la teoría y otra es la práctica. Así pues, es imprescindible tener en cuenta los siguientes puntos.

- Inventario de activos.
- Conjunto de riesgos a los que está expuesta cada activo.

- Conjunto de vulnerabilidades asociadas a cada activo (si corresponde).
- Conjunto de medidas de seguridad implantadas.

Para el hipotético caso en el que surge un riesgo no previsto, es necesario documentarlo correctamente (como algo surgido de forma imprevisible), puesto que puede servir de gran ayuda a proyectos posteriores.

13.5 Gestión y Validación de las pruebas

Para una correcta gestión las pruebas se han realizado dos fases de validación del proyecto esenciales y se ha añadido una última prueba opcional.

- Fase 1. Prueba de funcionalidad de componentes.
  - Funcionalidad del LDAP.
  - Funcionalidad del SSO.
  - Funcionalidad de MDM e IDD.
- Fase 2. Prueba de funcionalidad de la aplicación.
- Fase 3. Prueba de funcionalidad de mejora: Funcionalidad del HA-Proxy.

La segunda fase depende del buen funcionamiento de la primera, es decir, si la prueba de componentes no está validada, no se pasa a la prueba de la aplicación. A esto se le añade el hecho de repetición: Cada vez que una prueba falla, se hace una gestión de incidencias (véase del subapartado anterior) y se vuelven a realizar las pruebas de funcionalidad.

La tercera fase, que para este proyecto sí que resulta importante, también depende de las anteriores. Además, una vez realizada esta última fase, es importante volver a probar la funcionalidad de la aplicación completa (fase 2).

13.6 Gestión de plazos y presupuesto

En este subapartado se presenta una tabla con los diferentes entregables que forman el proyecto, así como a la fase que pertenecen y un cálculo de los días laborales que se tarda en llevar a cabo cada una.

Fase	**Entregable**	**Días laborables**	**Presupuesto**
Fase Inicial	Acta	8	-
Fase de Planificación	Diseño	90	-
Fase de Ejecución	Implantación	Aprox. 298	-
Fase de Cierre	Informe de Cierre	12	-

Tabla 8. Gestión de plazos y presupuesto

Nota: El presupuesto del proyecto no se especifica puesto que no se han dado datos al respecto. Se supone que se va a diseñar e implementar una aplicación gratuita. No obstante, sí se tiene en cuenta que el alumno cobrará por las horas trabajadas (véase el apartado 15 “Resumen del presupuesto”).

14 Planificación temporal

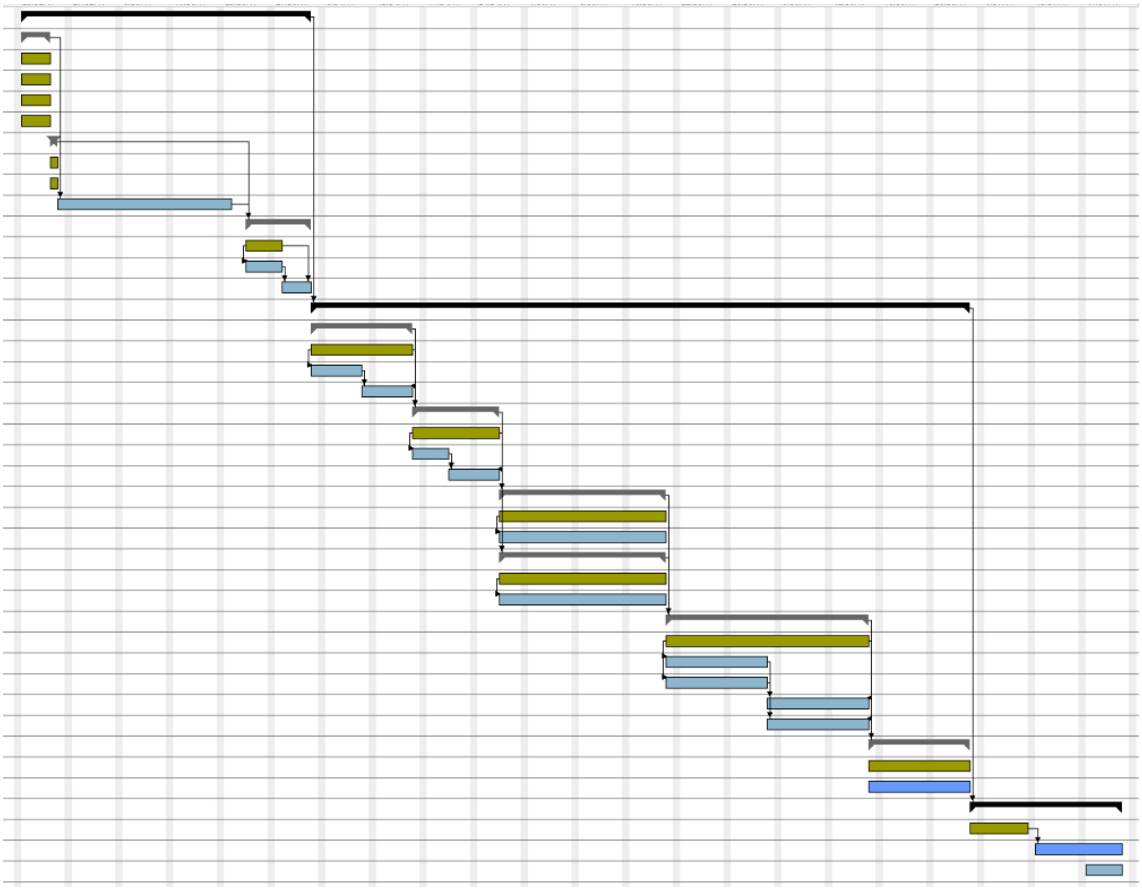
En este apartado se va a analizar el plan llevado a cabo para organizar el desarrollo, implementación y documentación del proyecto, así como su fase de pruebas. Puesto que este trabajo ha sido realizado por una sola persona, más adelante se ve que la planificación sigue un modelo lineal. Se ha utilizado la herramienta "*GanttProject*", explicada anteriormente.

En primer lugar, se van a listar las diferentes tareas principales del proyecto que conforman las fases de inicio, desarrollo y cierre del mismo, con objeto de obtener una visión general del orden en que se han realizado. Estas tareas se dividen, a su vez, en subtareas que concretan más la planificación del proyecto.

Nombre	Fecha de inicio	Fecha de fin
INICIO	20/02/17	31/03/17
Propuesta TFG	20/02/17	23/02/17
Doc Resumen	20/02/17	23/02/17
Doc Motivación	20/02/17	23/02/17
Doc Objetivos	20/02/17	23/02/17
Doc Diseño	20/02/17	23/02/17
Recursos	24/02/17	24/02/17
Doc Rec Humanos	24/02/17	24/02/17
Doc Rec Materiales	24/02/17	24/02/17
Diagrama Gantt	25/02/17	20/03/17
Entorno	23/03/17	31/03/17
Doc Entorno	23/03/17	27/03/17
Impl Máquina Virutal	23/03/17	27/03/17
Impl Red	28/03/17	31/03/17
DESARROLLO	1/04/17	30/06/17
LDAP	1/04/17	14/04/17
Doc LDAP	1/04/17	14/04/17
Impl BD	1/04/17	7/04/17
Credenciales ADFS	8/04/17	14/04/17
SSO	15/04/17	26/04/17
Doc SSO	15/04/17	26/04/17
Diseño Web Sencilla	15/04/17	19/04/17
Impl Web	20/04/17	26/04/17
MDM	27/04/17	19/05/17
Doc MDM	27/04/17	19/05/17
Impl Libreria MDM	27/04/17	19/05/17
IDD	27/04/17	19/05/17
Doc IDD	27/04/17	19/05/17
Impl Libreria IDD	27/04/17	19/05/17
Integración	20/05/17	16/06/17
Doc Integración	20/05/17	16/06/17
LDAP - SSO	20/05/17	2/06/17
LDAP - MDM	20/05/17	2/06/17
MDM - IDD	3/06/17	16/06/17
SSO - IDD	3/06/17	16/06/17
Pruebas	17/06/17	30/06/17
Doc Pruebas	17/06/17	30/06/17
Pruebas y cambios	17/06/17	30/06/17
FIN	1/07/17	21/07/17
Doc Final	1/07/17	8/07/17
Diapositivas	10/07/17	21/07/17
Presentación	17/07/17	21/07/17

Captura 1. Planificación temporal

Debido a que el diseño de este diagrama fue creado antes de empezar con el desarrollo de la aplicación, ha habido unos cambios en las fechas de inicio y fin de las tareas a partir de la segunda fase. Inicialmente el proyecto estaba propuesto para su desarrollo a partir del mes de febrero, pero finalmente se decidió retrasar su inicio dos meses. No obstante, la duración de las tareas ha sido prácticamente igual a la que se muestra en el diagrama de Gantt diseñado inicialmente y que se muestra a continuación.



Captura 2. Diagrama de Gantt

15 Resumen del presupuesto

Puesto que el coste temporal del proyecto ya está aclarado, en este apartado se calcula el presupuesto para llevar a cabo su diseño e implementación.

15.1 Coste del hardware

Tal y como se ha especificado en el apartado 5.2.1.2 “Recursos”, se han utilizado dos PCs portátiles para agilizar la virtualización de las máquinas.:

- Portátil 1 (potencia normal-alta): Aproximadamente 700€
- Portátil 2 (potencia normal-baja): Aproximadamente 400€

A continuación, se muestra una tabla con las características en detalle de los portátiles utilizados, con el objetivo de tener una idea de la potencia disponible para arrancar los servidores de la aplicación desarrollada.

Componentes	Portátil 1	Portátil 2
Disco duro	HDD: 500GB – 7200RPM	HDD: 500GB - 7200RPM
Memoria instalada (RAM)	8GBytes DDR3 Dual	DDR3 SDRAM 4GB
Tarjeta gráfica	Intel® HD Graphics 4600	Intel® HD GRAPHICS 400
Procesador	Intel® Core™ i5-5210M 2.60GHz (4 CPUs)	Intel® Core™ i3-6006U CPU 2.60GHz
Sistema Operativo	Microsoft Windows 10 Home 64 Bits	Microsoft Windows 10 Premium 64 Bits

Tabla 9. Características técnicas del hardware

Así pues, puesto que esta adquisición no formará parte del producto final, ya que se han aprovechado dos portátiles con estas potencias que ya tenía la empresa, se va a realizar una estimación de la amortización. Suponiendo que el coste total es de 1.100€, y que la vida útil a pleno rendimiento y sin verse afectado en un ordenador portátil es de 6 años:

- **Amortización**: 1.100/6 = 183,33€ anuales

15.2 Coste del software

Las herramientas utilizadas para el proyecto, así como los sistemas operativos (véase el Anexo 17.2 “Estudios con entidad propia” el apartado de SO para más información) son gratuitos. Así pues, puede ser modificable en un futuro sin coste adicional alguno y por cualquier persona de la empresa. Al ser, gratuito y de código abierto, decimos que es un proyecto *Free Open Source.* Una de las máquinas, la de MDM, está desplegada sobre un sistema operativo SRHEL, pero ya se disponía de ella, por lo tanto, tampoco se incluye como coste.

Tanto para la realización de la memoria y posterior presentación, como para el montaje de la aplicación en sí, ha sido necesario el sistema operativo de Windows y algunos de sus componentes, que vienen incluidos en la instalación. Cabe remarcar que no se tendrá en cuenta este gasto, puesto que se supone que el precio del equipo Microsoft ya iba incluido en el del portátil, visto en el apartado anterior.

15.3 Coste de la mano de obra

Tal y como se observa en el apartado anterior de planificación temporal, el proyecto dura unos 5 meses, teniendo en cuenta la presentación de la idea inicial, pero sin contar el tiempo tardado en recibir su aprobación, hasta llegar a la presentación del mismo. Se ha tenido en cuenta que el desarrollo e implementación del mismo se han llevado a cabo durante la jornada laboral, por tanto, el estudiante ha utilizado una media 20h/semanales, para el hipotético caso en el que la hora se paga a 25€. No obstante, estas horas no están repartidas equitativamente, puesto que al inicio del proyecto se han invertido menos horas que al final. A continuación, se presentan los resultados de los cálculos en una tabla.

Concepto	Mes 1	Mes 2	Mes 3	Mes 4	Mes 5	Cantidad	Precio	Total
Hardware								

Portátil 1 (Windows 10)	0	0	0	0	0	0	0	0€
Portátil 2 (Windows 10)	0	0	0	0	0	0	0	0€
Software								
Oracle Linux	0	0	0	0	0	3	0	0€
Máquina MDM (RHEL)	0	0	0	0	0	1	30€	30€
Servidor web	0	0	0	0	0	1	45€	45€
Otras herramientas	0	0	0	0	0	9	0	0€
Mano de obra								
Estudiante	60h	60h	80h	80h	120h	400h	25€	10.000€
							Sub Total	10.075€
						% IVA	21	2.115,75€
						Total Presupuesto		12.190,75€

Tabla 10. Costes del Proyecto

16 Conclusiones y trabajo futuro

En este apartado se sintetiza el trabajo realizado a lo largo del diseño e implementación del proyecto. La información se divide en subapartados en los que se describen los problemas que han surgido durante el desarrollo del proyecto, así como las aportaciones del mismo, tanto a nivel técnico como a nivel personal. Para finalizar, se ha añadido una propuesta sobre una funcionalidad que sería interesante añadir en un futuro con la intención de mejorar la aplicación.

16.1 Dificultades encontradas

El hecho de haber definido unos requisitos, el alcance del proyecto y las restricciones del mismo a partir de una hipótesis inicial ha ayudado a encaminar el proyecto para evitar, en la medida de lo posible, problemas y dificultades iniciales. A su vez, realizar un adecuado análisis de riesgos ha permitido detectar a priori posibles amenazas que podrían surgir durante el desarrollo y ralentizar el proceso de implementación. A pesar de todo, han surgido pequeñas dificultades que abarcan desde problemas por un mal uso del lenguaje de programación hasta fallos en los servidores a causa de una mala configuración.

Una de las dificultades principales ha surgido a raíz del manejo de cabeceras *headers* por parte del servidor HA-Proxy. El problema radicaba en la dificultad de desencriptación de las mismas (para poder autentificar un cliente). Dado un *header* encriptado, al *firewall* le resultaba difícil tratarlo a partir de la clave: al hacer una mala desencriptación, no reconocía la petición y la rechazaba.

Otra de las dificultades surgidas iba relacionada con el tema de autorización de los usuarios. El hecho de tener que trabajar con una máquina MDM predefinida, ha limitado la modificación de la misma para a sincronización de los roles. Tanto MDM como IDD disponen de unas librerías y un fichero java al que se le pueden hacer modificaciones. No obstante, pese a su facilidad de

modificación a nivel de componentes internos, trabajar con datos externos ha sido más complicado por temas de seguridad del propio fichero.

Por último, se han detectado también dificultades de comunicación entre servidores, relacionados con temas de optimización (*performance).* Aun así, tanto para los problemas de autorización y autenticación como para el de velocidad de respuesta de los servidores se han encontrado las correspondientes soluciones.

16.2 Aportaciones del proyecto

Pese a las dificultades encontradas durante el desarrollo, el proyecto cumple su tarea principal: autorizar y autentificar los usuarios de un servidor a otro remoto, de forma automática. El proyecto logra, de esta forma, que un trabajador de la empresa *arhis* con su propio usuario y rol sea capaz de conectarse de forma segura al servidor de la empresa asociada *Informatica* y aprovechar su cuenta y privilegios, sin importar el cliente para el que se esté trabajando.

A nivel personal ha sido un proyecto muy gratificante. El hecho de partir de una base nula con respecto al tema de seguridad en la red, así como la gestión de datos maestros y almacenamiento en la nube, ha servido para ampliar los conocimientos sobre estos conceptos. Ha servido, además, para saber, en detalle, como funciona la tecnología del SSO, un tema muy presente en la actualidad debido a lo imprescindible que es para el día a día internet.

16.3 Ampliaciones futuras

Con el objetivo de mejorar el rendimiento del proyecto, sería interesante añadir una implementación que mejorara la operabilidad de los servidores. Al igual que el de MDM proporcionado por *Informatica* ofrece servidores que respaldan al principal, tanto el HA-Proxy como el LDAP podrían tener sus propios respaldos. Se consigue de esta forma que, en el hipotético caso de que un servidor tenga un mal funcionamiento, la aplicación sea capaz de redireccionar las peticiones de forma automática a otro servidor y evitar la pérdida de la información que facilita el funcionamiento del sistema.

Tal y como se ha mencionado en apartados anteriores, en este proyecto no se ha implementado la funcionalidad que facilita el trabajo con usuarios menos habituales. A parte de *admin* y *user*, *Informatica* posee una serie de roles que van incluidos en sus usuarios predefinidos e incluso da la posibilidad de crear nuevos y agregarlos. Sería pues una buena idea añadir una automatización para incluir dicha variedad de tipos de cuentas de usuario. Cabe remarcar que, en caso de ser aceptada, la implementación de la mejora no entraría en vigor inmediatamente. En la actualidad, los clientes para los que trabaja *arhis* no demandan unas funcionalidades especiales en cuanto a los roles de sus usuarios. Todos ellos trabajan exclusivamente con *admin* o *user* y la mejora no sería de ninguna utilidad. Aun así, puede facilitar trabajos futuros.

17 Anexos

17.1 Anexo: Documentación de entrada

Este proyecto ha sido diseñado e implementado en la sede de València de la gestoría *arhis,* por la autora del mismo (Julia Penadés), alumna de la Universitat Politècnica de València (UPV). La aplicación se ha llevado a cabo con herramientas tanto de *arhis* como de su principal empresa asociada (*partner*), una plataforma de gestión de datos llamada *Informatica*. Así pues, el

proyecto va dirigido a *arhis* con el objetivo de poderse conectar a los servidores de *Informatica* de forma más óptima y mejorar el trabajo con sus clientes a largo plazo.

17.1.1 arhis

Tal y como se ha mencionado al inicio de la memoria, *arhis Competency Center* es una gestoría multinacional especializada en gestión de datos maestros y cuya cita "*Going beyond data*" realza su principal característica. La empresa dispone de sedes desde países europeos, como España, Francia o el reino Unido, hasta de países de Asia, como China. Ha dirigido proyectos MDM para empresas conocidas, como *Carrefour* e *Iberia*, entre otras.

Fundada en el año 2003, *arhis* es un proveedor dinámico e innovador de servicios de consultoría en tecnología de la información. Pese que su enfoque inicial fue proporcionar servicios y soporte para las instalaciones de los portales *Supplier Relationship Management* (SRM) y *Bussines to Bussines* (B2B) de *Commerce One*, terminó apostando por una gestión de datos maestros MDM.

Figura 21. Logotipo de le empresa arhis

Con la aparición de la tecnología de integración de datos de clientes (CDI), *arhis* decidió invertir en esta nueva tecnología. CDI más tarde se convirtió en MDM, y *arhis* es por lo tanto uno de los primeros adoptantes de lo que ahora se llama soluciones MDM.

A continuación, se enumeran sus características más destacadas:

- **Expertos en MDM:** Preparación de la información, limpieza y normalización de datos, así como actualización gracias a la centralización de los mismos (se almacenan en un repositorio central y se ponen a disposición de todos los demás sistemas que lo utilizan)
- ***arhis* MDM Factory**: Produce artefactos de software pre-configurados y pre-ensamblados como documentos XML estandarizados. Estos pueden ser fácilmente distribuidos e incorporados en la implementación MDM en el sitio
- Soporte al cliente después de la implementación (***post implementation support***)
- **Servicios de educación**. Se pretende que sus clientes tengan los conocimientos y las habilidades necesarias para participar en el despliegue, mantenimiento y actualización de las instalaciones MDM. Disponen de clases iniciales (*training)* como se explica más adelante.
- ***Partnership***. Afiliación con otras empresas.

Haciendo referencia a este último punto, cabe destacar que el principal *partner* es la mencionada anteriormente *Informatica*, empresa cuya información se explican con más detalle en el siguiente apartado.

Para más información sobre *arhis* se recomienda consultar su página oficial: http://www.arhis.com/

17.1.2 Informatica

Al igual que *arhis*, *Informatica* es una consultora especializada en el sector de gestión de datos maestros. La empresa estadounidense cuenta con que es líder mundial en el manejo de datos en la nube, es decir, una especialista en *Enterprise Cloud Data Management* que cuenta con sedes en América, Europa y Asia.

Figura 22. Logotipo de Informatica

Destaca por algunos de los siguientes aspectos:

- Gran versatilidad en ***Cloud* MDM** y ***Multidomain MDM*** y sus opciones de integración son muy probablemente sus tres puntos más destacados.

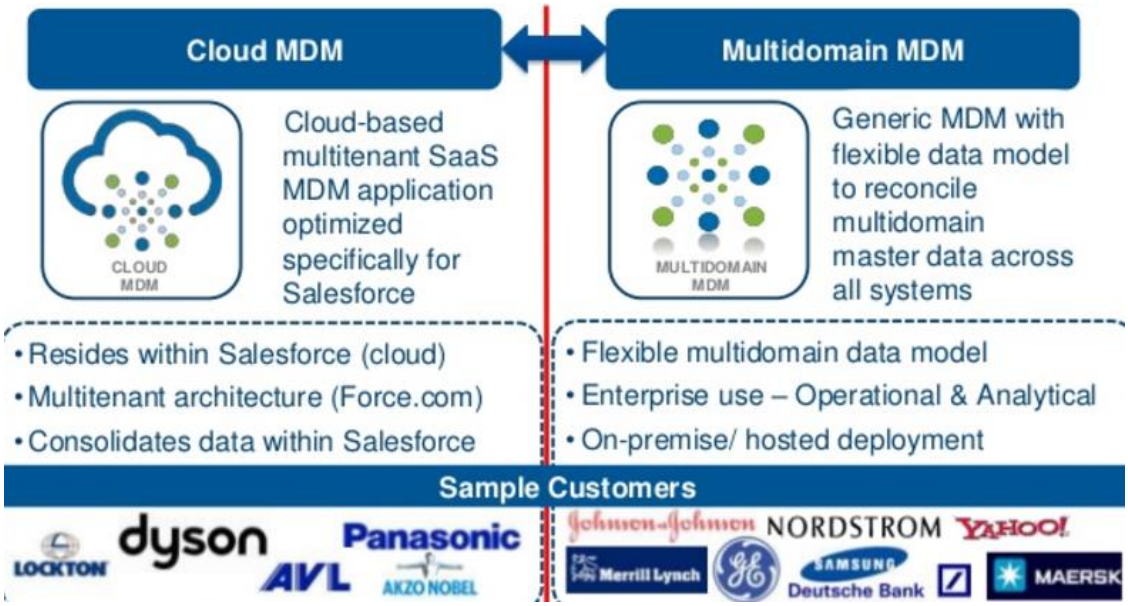


Figura 23. MDM en la nube vs MDM multi-dominio

- Dispone de un gran abanico de **productos** y **herramientas** en la nube para facilitar el trabajo de sus clientes.
- Garantiza **soporte global** al cliente.
- Dispone de **Informatica University**, un centro de formación especializado.
- Trabaja con ***partners*** de cualquier sector o categoría.

Véase la página oficial de *Informatica* para más información: https://www.informatica.com/.

En el caso de su relación con la empresa *arhis*, interesa destacar uno de sus productos de gestión de datos: ***Informatica MDM – Product 360***. Esta herramienta ha sido imprescindible tanto para este proyecto como para otros proyectos en la empresa. En el apartado siguiente se entra más en detalles.

17.2 Anexo: Estudios con entidad propia

17.2.1 Informatica MDM

17.2.1.1 Definición

La gestión de datos maestros (MDM) de *Informatica* es un marco de confianza único que proporciona datos empresariales críticos consolidados y fiables con una gestión completa del

flujo de trabajo. Utiliza un modelo de negocio flexible para satisfacer las necesidades del negocio y mejorar la transparencia. Está diseñado para reducir el tiempo de búsqueda y conciliar los datos mejorando la eficiencia, integrando las adquisiciones y gestionando el cumplimiento normativo para reducir el riesgo. MDM permite que la administración de la jerarquía revele relaciones valiosas entre los datos.

Figura 24. Logotipo del producto de Informatica

Informatica MDM en su versión 10.1 (versión con la que se trabaja actualmente) es uno de los productos más reconocidos del segmento MDM. De acuerdo con el sitio web de *Informatica* su solución *"Crea una visión única e integral de los clientes, productos u otras entidades de negocio"*. Entre otras, se destacan las funcionalidades autónomas que facilitan una solución completa mediante la inclusión en el software de herramientas de integración de datos, calidad de datos y gestión de procesos de negocio.

También se alude a la búsqueda inteligente para recuperar rápidamente información de datos maestros, a la creación de aplicaciones compatibles con MDM, a una visión completa e integral de las entidades de negocio, a una seguridad inteligente (*smart security)* y a los controles de datos. En este último aspecto, se menciona una fluida incorporación de componentes de datos maestros avanzados a aplicaciones empresariales como *Enterprise Resource Planing* (ERP) o *Customer Relationship Management* (CRM).

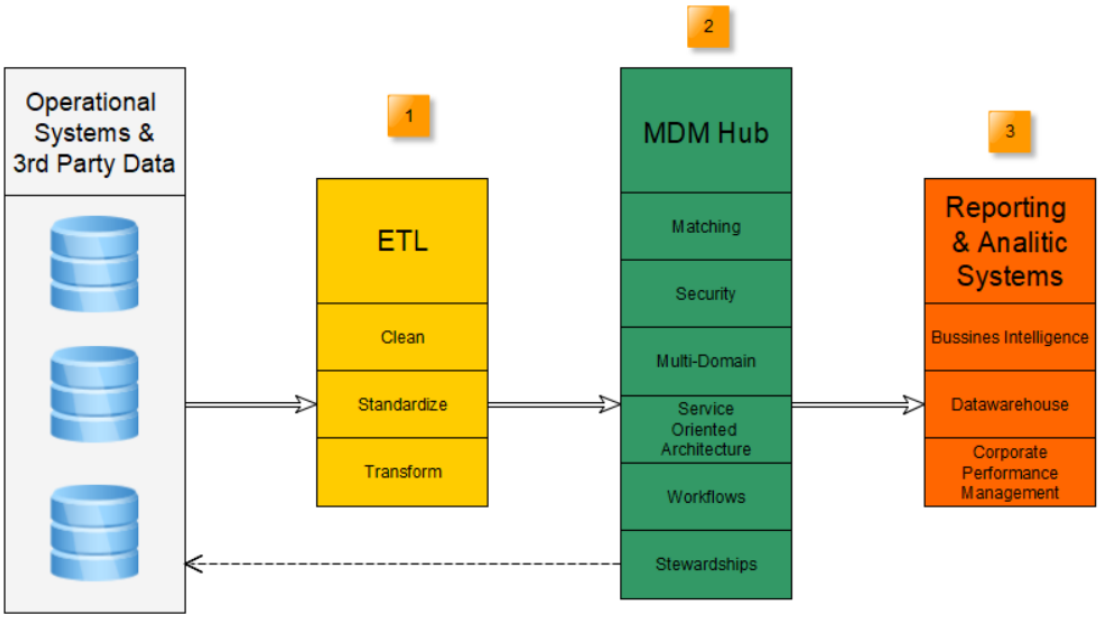


Figura 25. Tratamiento de datos en Informatica MDM

Basándose en la *Figura 25,* los pasos que sigue el tratamiento de datos de MDM (de forma resumida y esquemática) sería el que se muestra a continuación.

1. Limpieza y mejora los datos en el sistema existentes mediante transformaciones y estandarizaciones.

2. Registro de datos mejorados en en el sistema y aplicación de la lógica y reglas establecidas en el sistema de *Informatica* MDM sobre ellos, como por ejemplo:
    a. Flujos de datos (*workflows)* que implementen dichas lógicas y reglas, gracias al uso de gracias a herramientas como *Informatica Data Quality* (IDQ) o *Informatica Analyst* (IA) para la intervención manual.
    b. Detección de duplicados (*match)*.
    c. Fusión de las entradas duplicadas (*merge*).
    d. Sistemas de administración de datos (*Data Stewardships)* que permiten a un usuario utilizar las herramientas para consolidar y gestionar los datos de la organización.
3. Reporte y análisis de los datos tratados.

17.2.1.2 Hub de MDM

17.2.1.2.1 Estructura del Hub

Pese a que se podría entrar en mucho detalle, para este proyecto interesa hacer hincapié exclusivamente en el Hub de MDM. El Hub está dividido, a su vez en tres capas principales: la capa de aplicación, la de la consola y la capa de base de datos.

Así pues, es interesante centrarse en la tercera capa mencionada. Para ello, a continuación se hace una explicación basándose en un esquema facilitado por la misma empresa de *Informatica,* en la que se muestra de forma detallada la distribución de los servicios en la capa.

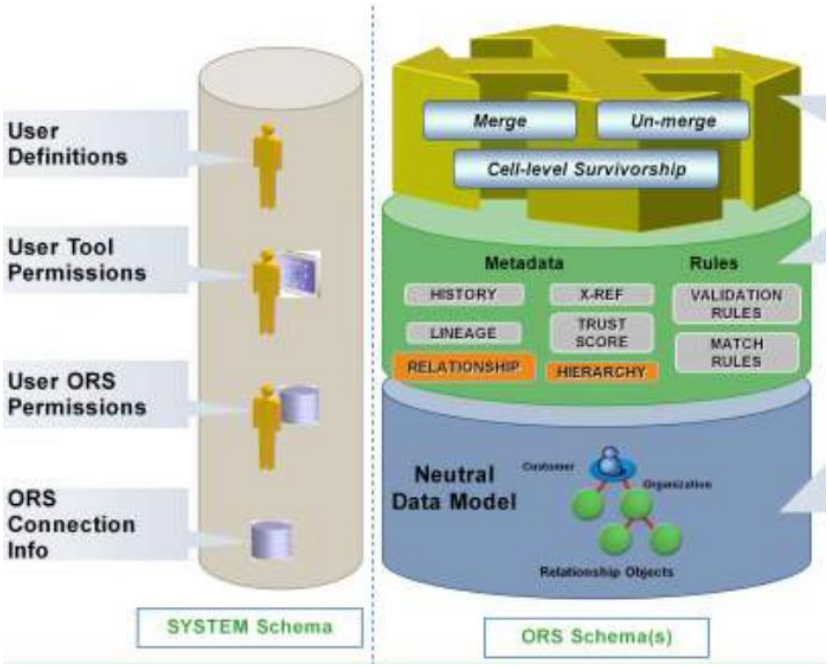


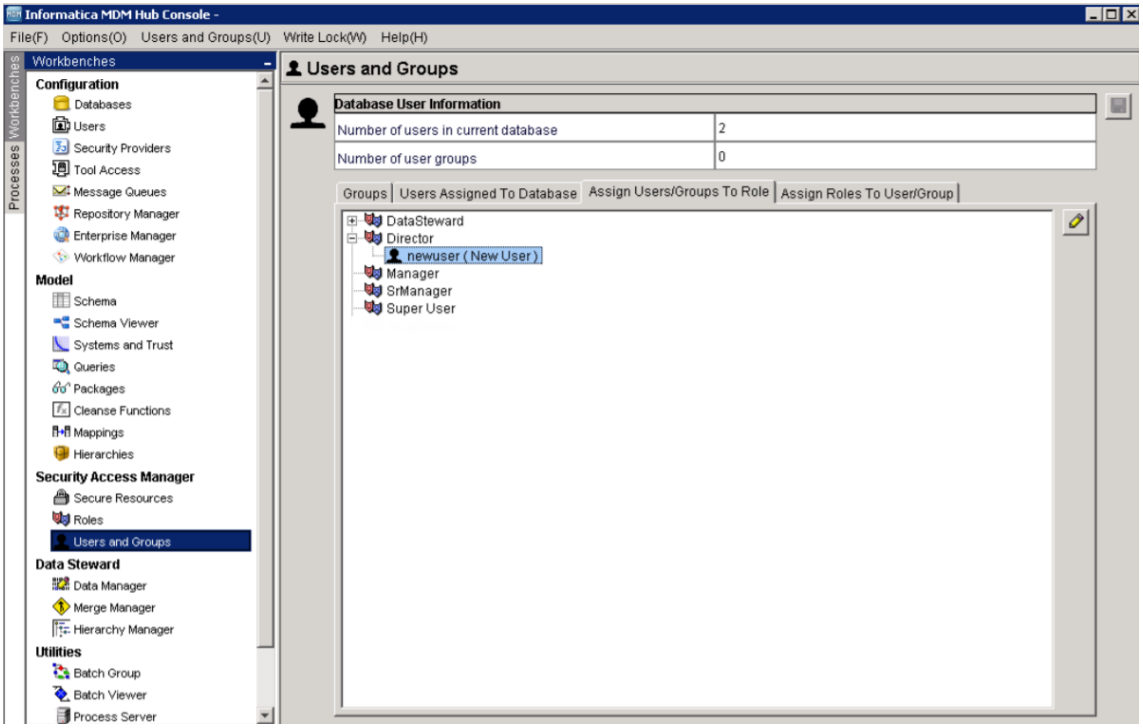
Figura 26. Capa de base de datos MDM

La capa de BD es donde se almacenan todos los datos y metadatos del negocio. Está implementada siguiendo el modelo de bases de datos relacionales, en este caso en Oracle. A parte de ser donde se guarda toda la información de la empresa, esta BD está dividida en dos partes para agilizar su gestión:

- El *Operational Reference Store* (**esquema ORS**). Contiene los objetos de datos, repositorio y la lógica y gestión de referencias a datos maestros.
- El **esquema del sistema**. Se utiliza para registrar información común sobre las ORS, como por ejemplo los ajustes de conexión para cada una, así como los usuarios que pueden acceder a MDM, y los esquemas y herramientas a los que cada usuario puede acceder.

17.2.1.2.2 Usuarios y roles

Volviendo la atención al proyecto, es necesario acceder a la parte de SYSTEM para obtener la información sobre los usuarios y sus permisos. En la captura que se muestra a continuación, vemos que en el Hub existen unos roles predefinidos y se les puede añadir usuarios (para que actúen con estos roles).



Captura 3. Usuarios y roles de la MDM Hub Console

Vemos que desde el Hub también es posible:

- Crear grupos de usuarios.
- Gestionar los usuarios existentes para saber a que bases de datos tienen acceso.
- Añadir grupos o usuarios a un rol y viceversa.
- Crear nuevos roles a partir de los roles predefinidos.

17.2.1.2.3 IDD

Otro de los pilares que forman el proyecto es *Informatica Data Director* (IDD). Es una aplicación web de gestión de datos que permite soluciones de datos maestros que son efectivas para todos: usuarios empresariales, administradores de datos y administradores de tecnologías de la información (TI).

IDD es altamente configurable, con una interfaz fácil de usar basado en el modelo de datos de su organización y permite a los usuarios empresariales:

- Crear datos maestros de alta calidad, trabajando individualmente o de forma colaborativa en toda su empresa.
- Administrar duplicados, resolver coincidencias, aprobar y administrar actualizar sus datos maestros, crear y asignar tareas a los usuarios.
- Buscar todos los datos maestros desde una ubicación central y ver sus detalles.
- Supervisar el linaje y la historia de la pista, así como personalizar el panel de control.

En resumen, IDD sería como un sub apartado dentro de MDM, una forma de acceder a los datos desde una aplicación web. En los apartados siguientes se entra en más detalles sobre cómo configurar IDD y MDM para adaptarlos a las necesidades del proyecto.

17.2.2 Autenticación y Autorización

Ambos términos surgen de la necesidad de este proyecto de encontrar una solución a la retroalimentación de cuentas de usuarios. Se pretende que un usuario de *arhis* pueda reconocerse directamente en el servidor de *Informatica.* Se necesita, para ello, que el sistema autentique al usuario para acceder a los recursos y lo autorice para que pueda trabajar con ellos.

17.2.2.1 Autenticación vs Autorización

La autenticación tiene que ver con la identidad: verificando que el usuario es realmente quien dice ser. La autorización, por otro lado, es decidir qué recursos un usuario debe tener acceso a, y qué se les debe permitir hacer con esos recursos, es decir, los roles de cada usuario.

Aplicados ambos conceptos al ámbito del proyecto, como ejemplos básicos tendríamos los usuarios y administradores. Un usuario en *arhis,* existe en *Informatica.* Lo mismo pasa con un administrador. En cuanto a sus roles:

- Alguien con la función de usuario en *arhis* puede, desde el Hub de MDM (de *Informatica*):
  - Agregar tablas (que próximamente incorporaran datos).
  - Modificar sus propias entradas: solo puede tener acceso de lectura a las tablas creadas por otros usuarios.
- El rol de administrador de *arhis* también tiene el mismo derecho que un usuario. A parte incorpora más privilegios, como por ejemplo:
  - Creación, modificación y eliminación de cualquier tabla del Hub, sin importar que tipo de usuario sea el propietario.
  - Eliminar, modificar y crear nuevos usuarios y definir los roles. Esta opción no interesa puesto que se pretende sincronizar las cuentas, no crear nuevas remotamente.
- A parte de las acciones que se pueden realizar en el Hub de MDM, el rol también tiene acciones más allá: En IDD, por ejemplo, un administrador se diferencia de un usuario porque el primero puede visualizar los *reports* (estadísticas con información sobre los datos), mientras que el segundo no tiene acceso a ellos

17.2.2.2 Modelos de autenticación

Se parte de la existencia de dos formas de básicas de autenticación de cuentas, tal y como se muestran en la *Figura 24* que se ve a continuación. Pese a que la autenticación directa es la más común y sencilla de implementar, dista mucho de cumplir con los objetivos de este proyecto puesto que carece de seguridad. Debido a su sencilla estructura, el intercambio de contraseña entre servidores puede peligrar, ya que resulta fácil que un tercer usuario pueda adquirir dicha

contraseña y acceder al recurso. Así pues, para este proyecto interesa centrarse en sistemas que apliquen el segundo modelo: la autenticación federacional.

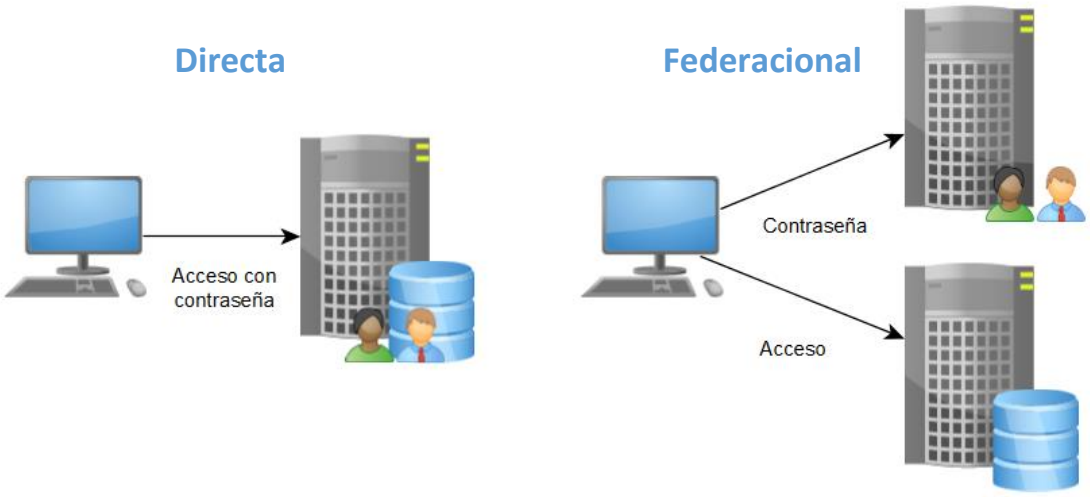


Figura 27. Modelos de autenticación

Así pues, en un sistema federacional, primero se accede a un servidor de nombres cuya función es autenticar al usuario y luego se nos garantiza el acceso a la base de datos del servidor, donde se hospedan los recursos.

17.2.2.3 Modelos de autorización

La autorización se sitúa en un paso intermedio entre la autenticación y el acceso a los recursos. Una vez autorizado un usuario, es necesario saber sus roles. Superada ya esta fase, el usuario podrá acceder a los recursos y realizar operaciones de acuerdo con su nivel de autoridad.

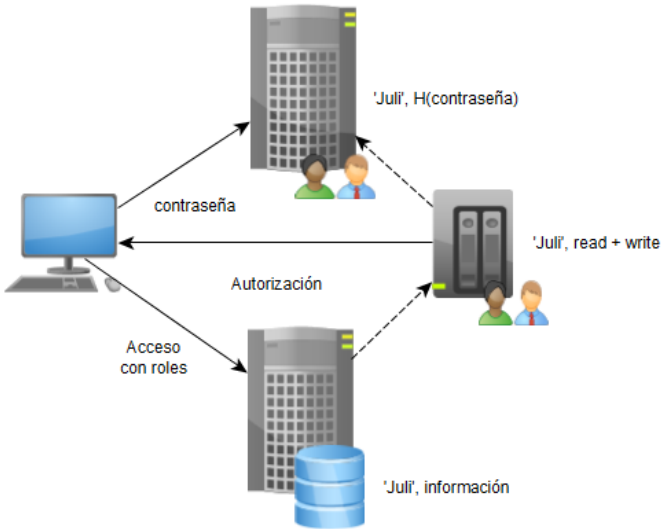


Figura 28. Flujo de la autorización

Así pues, con el fin de gestionar los clientes, se necesita tener un servicio que disponga de un listado con los clientes y sus respectivos roles y que actúe como proveedor de dicha información, tal y como se observa en la *Figura 24*. Entra en juego el *Identity as Service* (IDaaS), un servicio

basado en la nube para la gestión de identidad y acceso. Los servicios ofrecidos por este proveedor incluyen, entre otros, identidad federada y administración de contraseñas.

La identidad federada significa enlazar y usar las identidades electrónicas que un usuario tiene a través de varios sistemas de administración de identidad. En términos más simples, una aplicación no necesita necesariamente obtener y almacenar las credenciales de los usuarios para autenticarlos. En su lugar, la aplicación puede utilizar un sistema de gestión de identidad que ya almacena la identidad electrónica de un usuario para autenticar al usuario, dado, por supuesto, que la aplicación confía en ese sistema de gestión de identidad.

Basándose en la información anterior, se ha procedido a esquematizar las ideas básicas de autenticación y autorización, reflejadas en la *Figura 26,* con el objetivo de tener claras las diferencias entre los términos y comprender, más adelante, las decisiones tomadas en el desarrollo de la aplicación.

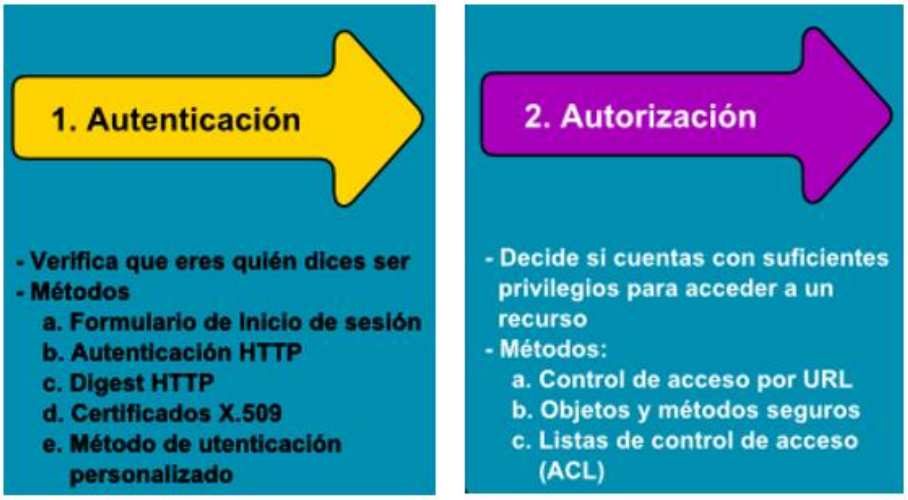


Figura 29. Autenticación vs Autorización

17.2.3 Reglas heurísticas de usabilidad

Para estudiar uno de los requisitos no funcionales de este proyecto, se ha decidido recurrir a las reglas de J. Nielsen, el "gurú" de la usabilidad. A continuación, se muestra un resumen de las mismas adaptadas a este proyecto.

1. **Visibilidad del estado del sistema**: la aplicación está capacitado de un sistema de mensajes que se muestran tanto por consola como por la red. De esta forma el usuario sabe en todo momento lo que está ocurriendo.
2. **Relación entre el sistema y el mundo real**: el sistema habla con el lenguaje de los usuarios, deja la información clara y concisa. Además, la tarea del usuario es simplemente introducir sus credenciales, no da lugar a dudas.
3. **Control y libertad para el usuario**: el usuario tiene control total sobre la aplicación y todas las operaciones que puede hacer. No hay distinciones ni permisos especiales.
4. **Consistencias y estándares**: La única acción que se realiza es la de insertar usuario/contraseña, un *Log In* estándar, que no da lugar confusiones al utilizarlo.
5. **Prevención de errores**: La aplicación facilita el tratamiento de errores en caso de su mal funcionamiento.

6. **Reconocimiento antes que recuerdo**: El usuario no necesita en ningún momento aprenderse de memoria la información que aparece. El encadenamiento de los movimientos tras ejecutar el *Login* son mecánicos.
7. **Flexibilidad y eficiencia de uso**: En un *login* comun, todo se hace de forma rápida y mostrando la información correspondiente.
8. **Estética y diseño minimalista**: Página web sencilla y accesible, con diálogos mínimos.
9. **Ayudar a los usuarios a reconocer:** La aplicación lanza mensajes de error claros. En caso de ser un error de la base de datos, el usuario lo sabrá, pero no lo podrá solucionar.
10. **Ayuda y documentación**: No es necesario ningún manual de usuario para utilizar la aplicación.

17.2.4 Configuraciones red

La implementación de la red que forma el proyecto ha sido una parte del desarrollo que ha necesitado de la búsqueda de información relacionada con las configuraciones red para elegir el tipo más conveniente. Pese que al final se ha optado por una implementación de máquinas concreta (véase el anexo que sigue, 17.3 “Análisis y diseño del sistema”), es interesante añadir el estudio que se ha realizado para llegar a dicha conclusión.

17.2.4.1 Adaptador puente (Bridge)

Un *bridge* añade un nivel de inteligencia a una conexión entre redes. Conecta dos segmentos de red iguales o distintos. Se puede ver un *bridge* como un clasificador de correo que mira las direcciones de los paquetes y los coloca en la red adecuada. En este modo, se crea una tarjeta de red virtual en el anfitrión que intercepta el tráfico de red y puede inyectar paquetes en la red, de manera que el huésped se configura como si estuviera conectado por un cable a la tarjeta de red virtual del anfitrión (Se simula una conexión física real a la red, asignando una IP al sistema operativo huésped).

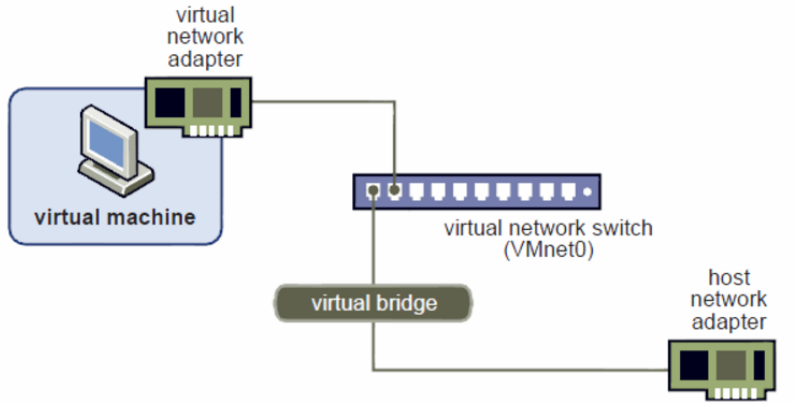


Figura 30. Adaptador puente

17.2.4.2 NAT (Traducción de Direcciones de Red)

La *Network Address Translation* (NAT) como bien indica su nombre, sirve para traducir lo *Internet Protocol* (IP) privada de la red en una IP pública para que la red pueda enviar paquetes al exterior; y traducir luego esta IP pública, de nuevo a la IP privada del pc que envió el paquete, para que pueda recibirlo una vez que llega la respuesta. Este modo permite al huésped navegar por Internet, descargar ficheros y leer el correo electrónico sin necesidad de configurar el sistema operativo huésped. La máquina virtual puede conectarse a otras redes mediante un *router* virtual proporcionado por VirtualBox.

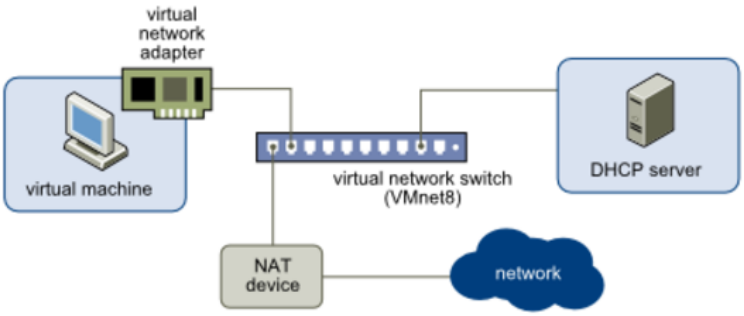


Figura 31. Adaptador NAT

17.2.4.3 Red Interna

Esta implementación se ha elegido para el *back-end* del sistema implementado. En este modo se crea una red virtual visible entre las máquinas virtuales, pero invisible al anfitrión o a máquinas externas a la red. Similar al Adaptador puente, se puede comunicar directamente con el mundo exterior con la salvedad de que ese mundo exterior está restringido a las máquinas virtuales conectadas en la misma red interna. Esta limitación viene justificada por seguridad y velocidad. Para más información, véase el apartado 17.3 "Análisis y diseño del sistema", donde se explica el diseño del HA-Proxy.

17.2.4.4 Adaptador Sólo-Anfitrión

En este modo se crea una tarjeta de red virtual en el anfitrión que puede conectarse con las máquinas virtuales sin necesitar que el anfitrión tenga una tarjeta de red. Es una mezcla entre los tipos "Adaptador puente" e "interna".

17.2.5 Sistemas operativos

En la actualidad existen gran cantidad de sistemas operativos. Cada uno de ellos presenta unas características específicas. Entre los sistemas más conocidos destacan algunos como Windows, OS X, UNIX, Linux, Solaris. A continuación se muestra información general sobre ellos.

17.2.5.1 Windows

17.2.5.1.1 Distribuciones Windows comunes

En el mercado se encuentran todo tipo de drivers para Windows, así como hardware compatible. Esto se debe a que, puesto que Windows tiene una gran demanda, existen gran cantidad de desarrolladores interesados en la plataforma de Microsoft. Por estas razones, entre otras, se pueden encontrar por la red muchos ejemplos de LDAP junto con AD implementados en estos sistemas operativos. Algunas versiones para win7, win8, win8.1, win10 se muestran a continuación.

- ***Windows 10 Home***: es la versión instalada en la mayoría de PCs de casa, incluye las aplicaciones universales.
- ***Windows 10 Pro***: Equivalente a Windows 8 Pro, y permitirá a los puestos y equipos empresariales conectarse a servidores orientados a estos escenarios además de tener acceso al programa *Windows Update for Business*.
- ***Windows 10 Enterprise***: Se basa en Windows 10 Pro y le añade características destinadas a ser utilizado en grandes implantaciones. Hay acceso a actualizaciones de seguridad y mejoras de forma más rápida.
- ***Windows 10 Education***: Derivada de la anterior, orientada a entidades educativas y académicas en la que haya requisitos especiales para la plantilla, los administradores, los profesores y los estudiantes.

17.2.5.1.2 Distribuciones Windows empresariales: Windows Server

A pesar de estar disponible en varias ediciones, las más interesantes para este Proyecto son la edición *Standard* y la *Datacenter*. La diferencia principal entre ambas es el número de máquinas virtuales (VM's) por *host* físico para las que tienen licencia: La versión Standard tiene 2 VM's mientras que la *Datacenter* tiene ilimitadas.

- ***Server* 2008 R2**: SO de *Microsoft* para empresas que proporciona características para virtualización, ahorro de energía, manejabilidad y acceso móvil.
- ***Server* 2002, Server 2003, *Server* 2012**…

17.2.5.1.3 Ventajas y desventajas

Ventajas: Intuitivo, visual, implementación sencilla.
Desventajas: Licencia de pago, limitado, difícil optimización.

17.2.5.2 Linux

Linux es un Sistema Operativo que tiene origen en Unix. Es el más conocido de entre los *open source* (código abierto), es decir, su software distribuido y desarrollado libremente. Es por esta razón que Linux se ofrece como una alternativa a Windows, mucho menos limitada y con software equivalente.

- **Linux generales:** Ubuntu, Debian, Fedora, Mint, KDE neon
- **Linux empresarial:** Red Hat Enterprise Linux (RHEL) , CentOS, SUSE Linux Enterprise Server, Oracle Linux, Mandriva Enterprise Server.

Para este Proyecto, no obstante, el en apartado siguiente vamos a centrarnos en algunas de sus distribuciones más conocidas para empresas.

17.2.5.2.1 Red Hat Enterprise Linux (RHEL)

Red Hat es una distribución comercial de las más recomendadas para empresas y servidores. Actualmente, es la más popular y con más experiencia en este sector. Se puede, además, instalar el paquete de programas Samba, para proveer archivos y servicios a clientes de Windows (conectar Linux a AD). No obstante, se necesita una licencia de pago para garantizar su soporte.

Características principales:

- Robustez y estabilidad
- Servidor de archivos (*Clustered file system*)
- Virtualización
- Seguridad
- *Clusters* de alta disponibilidad (ideal para grandes empresas)
- Soporte y actualizaciones regulares (de pago)

Figura 32. Logotipo del SO redhat

17.2.5.2.2 CentOS

CentOS es otra distribución de Linux, basada en RHEL. Es parte del trio Red Hat, junto Con RHEL y Fedora. CentOS es 100% binariamente comptatible con RHEL, es decir, Cualquiera que soporte Red Hat puede soportar también CentOS. A diferencia que RHEL, CentOS posee soporte gratuito de comunidad, lo que significa que tiene un nivel extra de pruebas, pero también que se necesita una persona dedicada al soporte del sistema.

Figura 33. Logotipo del SO centOS

Características principales:

- Robustez y estabilidad
- Virtualización (VMware)
- Calidad y fiabilidad
- Seguridad
- Soporte de comunidad (gratuito)

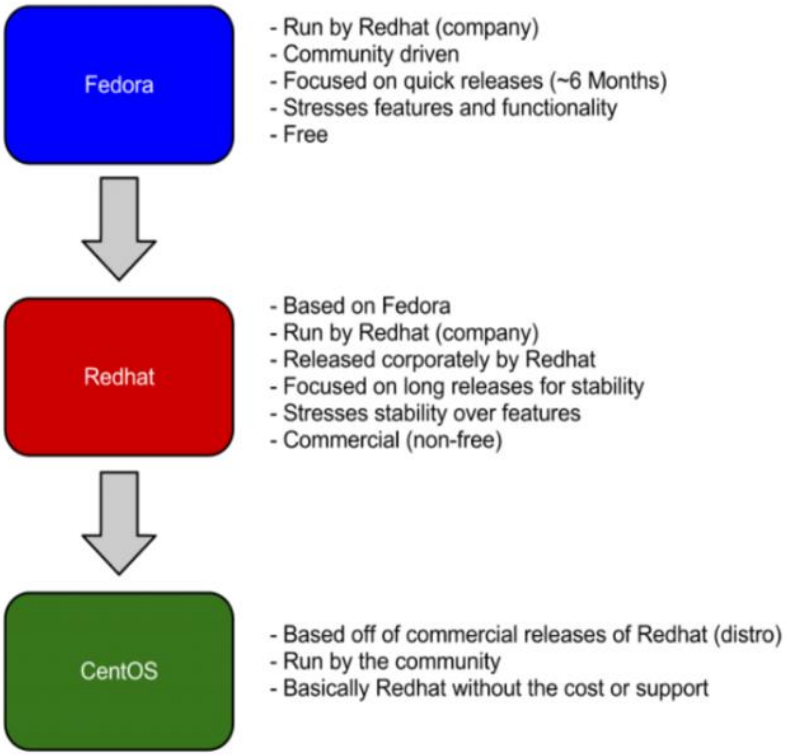


Figura 34. Relaciones entre sistemas operativos

17.2.5.2.3 SUSE Linux Enterprise Server(SLES)

SUSE Linux es una distribución perteneciente a Micro Focus que, pese a no tener tanto prestigio como Red Hat, también es capaz de manejar gran cantidad de entornos de empresa situados en servidores, siendo más eficaz a pequeña escala. SUSE cuenta con una interfaz gráfica que lo hace más intuitivo y "fácilmente" configurable. Licencia de distribución de pago (debido a su interfaz

gráfica, inútil para un servidor). Además, SUSE recibe recomendaciones de Microsoft, SAP y VMware.

Características principales:

- Robustez y estabilidad
- Virtualización
- Seguridad (Firewall, con buena integración de AD)
- Servidor de archivos (Cloud Computing)
- Soporte y actualizaciones regulares (de pago)

Figura 35. Logotipo del SO SUSE

17.2.5.2.4 Oracle Unbrekable Linux

Oracle Linux también es binariamente compatible con RHEL. Puesto que esta distro dispone de un equipo de desarrolladores dedicados exclusivamente al soporte del SO al igual que RHEL, se necesita una licencia, pero más barata.

Características principales:

- Kernel Irrompible
- Servidor de archivos (*Clustered file system*)
- Seguridad
- Virtualización
- Soporte y actualizaciones regulares (de pago)

Figura 36. Logotipo del SO Oracle Linux

17.2.5.2.5 Ventajas y desventajas

Generalmente, las características de las diferentes distribuciones de Linux se podrían resumir de la siguiente forma:
Ventajas: Licencia de distribución gratuita, optimizable, eficiente, rápido, *open source*
Desventajas: Complejo

17.2.5.3 Síntesis y conclusiones

Puesto que el proyecto va enfocado a una aplicación empresarial, una opción viable sería Windows Enterprise. No obstante, dado que el objetivo es la implementación de servidores, el sistema operativo (SO) ideal sería un Windows server, como por ejemplo el Windows Server 2008 R2.

Así pues, como se ha comentado con anterioridad, una alternativa a SO de Windows sería utilizar un SO de Linux. Los cuatro ejemplos que se han comentado cubrirían las necesidades del proyecto.

Características de los SO	RHEL	CentOS	SLES	Oracle Linux
Robustez y estabilidad	✓	✓	✓	✓
Servidor de archivos	✓	✓	✓	✓
Virtualización	✓	✓	✓	✓
Seguridad	✓	✓	✓	✓
Soporte de comunidad	✓	✓	✓	✓
Licencia de distribución de pago			✓	
Licencia de soporte de pago	✓		✓	✓
Soporte y actualizaciones regulares	✓		✓	✓

Tabla 11. Comparativa de Sistemas Operativos

La máquina de MDM que se utiliza lleva como SO el RHEL. Lo ideal habría sido que todas las máquinas llevaran el mismo SO y evitar con seguridad problemas de compatibilidad. No obstante, puesto que no entra en el presupuesto del proyecto el pago de una licencia por un sistema operativo, se ha decidido trabajar con Oracle Linux. Cabe remarcar también que en *Arthis* se trabaja con bases de datos de Oracle, lo que nos facilita su integración en el SO.

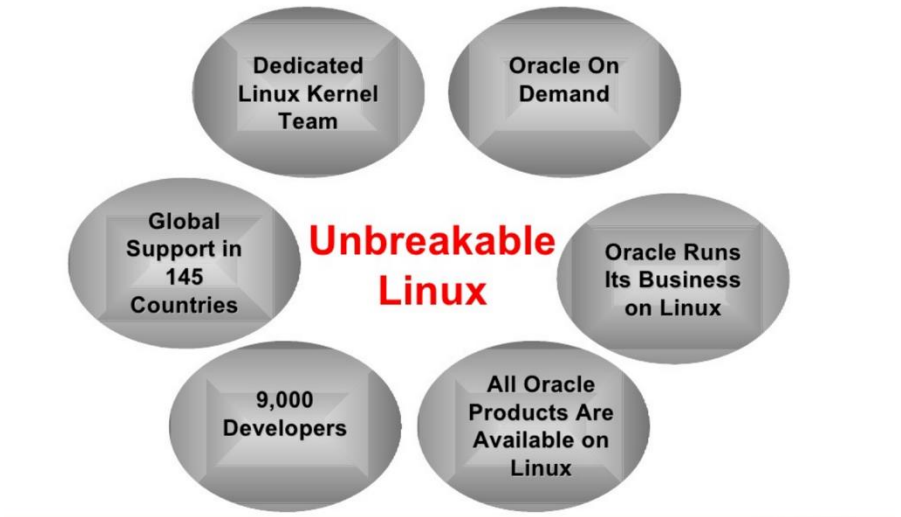


Figura 37. Características de Unbrekable Linux

17.3 Anexo: Análisis y diseño del sistema

A partir de la información recopilada en el apartado anterior y partiendo de la comparativa entre los sistemas que existen en la actualidad, capaces de garantizar la autenticación y autorización de los usuarios, se ha realizado el diseño del proyecto. Más adelante, en el Anexo 17.5: Implementación, se ha hecho un seguimiento completo del desarrollo de este diseño.

17.3.1 Flujo del sistema

La idea del funcionamiento del sistema se basa en el intercambio de información entre los diferentes componentes que forman el proyecto. Más adelante se explica el diseño de cada uno de ellos y su funcionalidad detallada. Así pues, tal y como se ha visto en el apartado 11. "Descripción de la solución propuesta" este sistema está compuesto por tres elementos clave:

- **SSO**: Módulo de seguridad implementado sobre una página web dinámica de *Login*.
- **LDAP**: Base de datos en la que se almacenan los usuarios de la empresa *arhis*.
- **MDM** e **IDD**: Servidor al que se intenta acceder y contra el que se tiene que autenticar y autorizar los usuarios.
- **HA-Proxy**: Proxy de seguridad (*secure proxy)* que hace de intermediario entre los otros componentes, encargado de gestionar las peticiones del usuario.

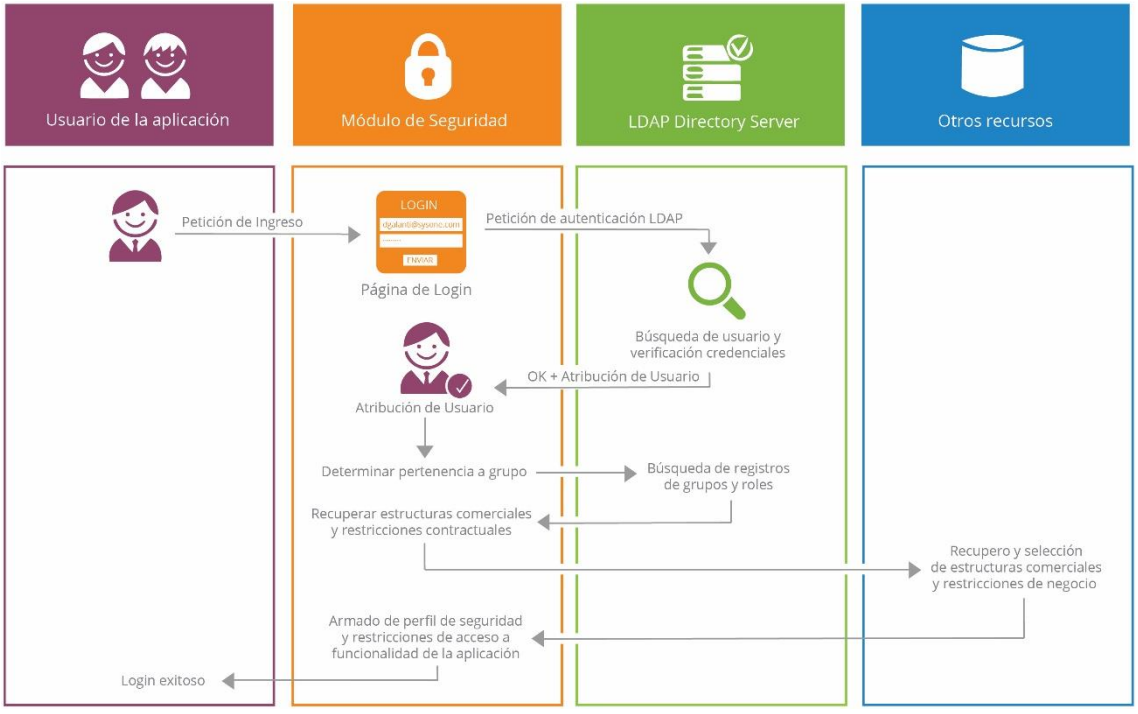


Figura 38. Flujo de un sistema de single sign on

En la *Figura 35* ya se muestra de forma aproximada el flujo de trabajo que se sigue este proyecto, sin especificar los componentes del mismo. A continuación, se supone el siguiente caso de uso con la intención de entrar en detalle del funcionamiento de la aplicación:

Se es un usuario trabajador de la empresa de *arhis* que necesita conectarse a la herramienta de *Informatica MDM*, puesto que se le ha asignado el trabajo de tratamiento de datos de la empresa cliente "X". El trabajador sigue, por tanto, los pasos que se ven a continuación.

1. El usuario, desde el servidor de *arhis,* intenta conectar con IDD, la interfaz gráfica que permite acceder a los datos de MDM desde una web.
    a. La petición (*request)* es interceptada por el servidor HA-Proxy, que actúa como *firewall* entre los componentes.
    b. El firewall se encarga de tratar la cabecera (*header)* de la *request* para redireccionarla en función del resultado.
2. Como se trata de la primera conexión, el *firewall* envía al usuario a la página web del SSO, el módulo de seguridad.
3. El usuario introduce su cuenta y contraseña y hace un envío (*submit)* de dichos datos. El SSO se encarga de empaquetarlos en su *header* y enviar la petición al LDAP.
4. En el LDAP se comparan el usuario y contraseña introducidos, con los existentes en la base de datos.
    a. En caso de ser correcto, al usuario se le da una clave (*token*) que actúa como un certificado para garantizar el acceso a MDM.
    b. En caso de no existir las credenciales, se redirecciona a una página de error con el mensaje "error de *user*/*password*".
5. Una vez obtenida la clave, esta se guarda en el *header* con una encriptación de tipo *hash** (par de clave, valor) y se envía en forma de respuesta (*response*) al *firewall*.
6. El HA-Proxy (*secure* proxy) reconoce el formato hash y redirecciona, esta vez sí, la petición a IDD.
7. IDD dispone del *Login Provider,* una funcionalidad facilitada por el MDM que se encarga de comprobar en el *header* que el par clave, valor es el esperado:
    a. En caso afirmativo, IDD acepta la conexión y se conecta con la base de datos de MDM.
    b. En caso negativo (ya sea por el *timeout** del token o por no corresponder al usuario) se rechazaría la *request*.
8. MDM garantiza el acceso al usuario: se facilita la autenticación y autorización del trabajador.

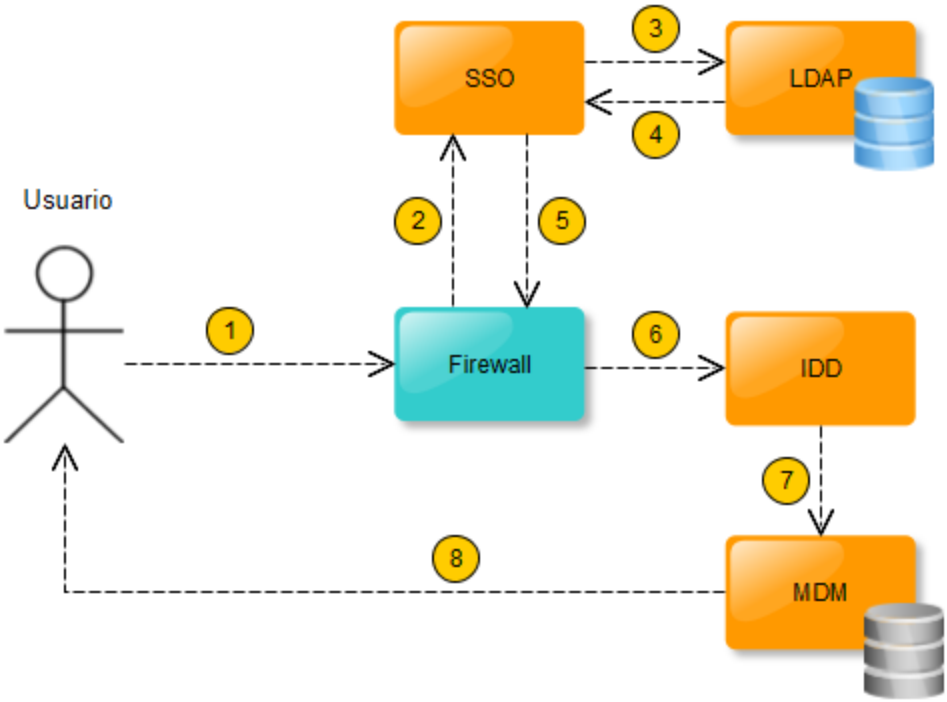


Figura 39. Flujo de trabajo del Sistema diseñado

* Hash <String, {token, *timeout*}> = <*user*: X, {*token*: lksdjfhla4nNSs46dD, 10}>.
* El *token* tiene, además, una fecha de caducidad (*timeout*) que actúa como contador, como por ejemplo de 10 minutos. Así pues, en caso de tardar en recibir la *response*, el *token* no se puede validar y el servidor lo rechaza.

17.3.2 Diseño inicial: Una máquina

Con el objetivo de tener una visión básica de la estructura del proyecto, se observa un diseño inicial en el que se ha distribuido los componentes del proyecto en una misma máquina. De esta forma se presentan el sistema compacto y se facilita el poder trabajar con cada componente para adquirir una primera visión.

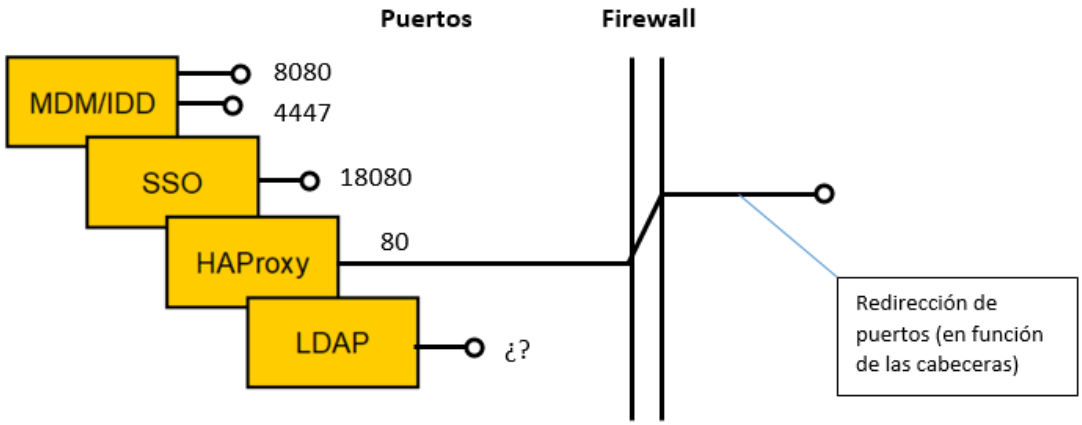


Figura 40. Diseño del sistema en una máquina

Así pues, tal y como se observa en la figura anterior, cada uno de los componentes que forman la aplicación está accesible a través de un puerto. Esto ha sido posible gracias a la función de la máquina virtual con configuración de red NAT, explicada en el anexo de 17.2 “Estudios con entidad propia” en el apartado sobre “Configuraciones red”.

En resumen y puesto que este no ha sido el diseño definitivo, se ha recurrido a una alternativa que se ciñe más a los requisitos del sistema. Cabe remarcar, no obstante, que la funcionalidad y el flujo del trabajo sigue siendo el mismo de un diseño a otro viéndose afectada, exclusivamente, la productividad del producto final.

17.3.3 Diseño final: Varias máquinas

Puesto que en apartado anterior se ha llegado a la conclusión que el despliegue de la aplicación sobre una única máquina no resulta factible, a no ser que se disponga de una máquina muy potente (en el apartado 15 “Resumen de presupuesto” se entra en detalle sobre el *hardware* utilizado), se ha propuesto un diseño final: una distribución del sistema sobre diferentes máquinas.

En la figura que se ve a continuación se muestra el esquema del diseño que cumple dichas características. Simulara una red de componentes conectados por *ethernet*, cada uno en un servidor diferente pero dentro de una misma área local. A su vez, se dispone de la posibilidad del HA-Proxy de comunicarse con la amplia web exterior, puesto que actúa como *firewall,* tal y como se ha mencionado en apartados anteriores. Se consigue de esta forma adaptarse en su totalidad al diseño propuesto en la solución propuesta y, además, la mejora de *performance* que se buscaba.

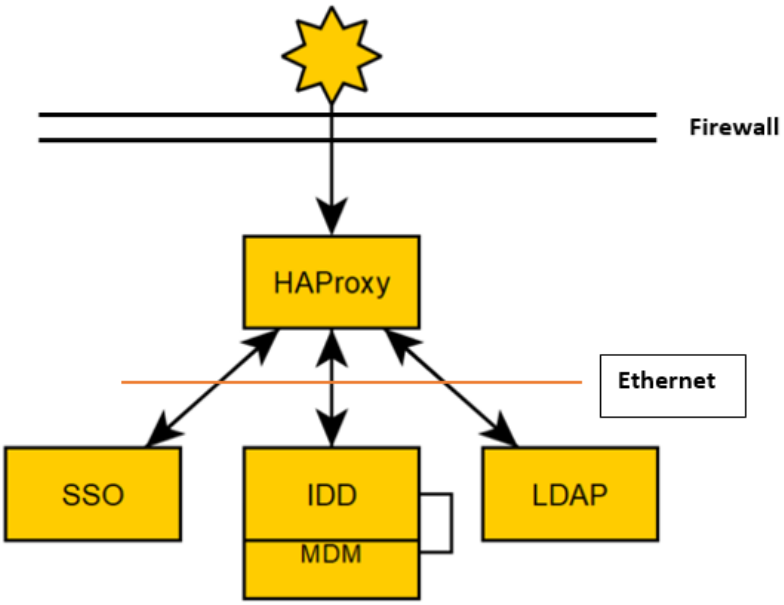


Figura 41. Diseño del sistema en múltiples máquinas

Más adelante se entra en detalle sobre cada componente. En los siguientes apartados se aclaran las dudas relacionadas con cada parte del sistema específica, así como la relación entre sus componentes, las dependencias y otros datos de interés para el proyecto.

17.3.4 Dependencias entre sistemas

Una vez seguido el flujo de trabajo del sistema y habiendo expuesto los diseños del mismo, (tanto a nivel de una máquina única en su diseño inicial como a nivel de muchas máquinas en su diseño final), se procede a hacer un análisis de las dependencias entre los sistemas con el objetivo de establecer un orden de implementación de los componentes.

Se muestra, a continuación, la tabla que marca los tiempos de implementación. Cabe remarcar que se trata de un análisis previo a la implementación y se ha dado el caso de tener que, por ejemplo, añadir funcionalidades al LDAP después de estar trabajando en la implementación del IDD.

T1	T2	T3	T4
LDAP			
	SSO		
	MDM		
		IDD	HA-Proxy

Tabla 12. Orden cronológico de implementación

En los siguientes apartados se hace un seguimiento del diseño del sistema, empezando por la implementación del servidor de nombres que se ha montado sobre un LDAP y siguiendo con el

desarrollo de la página web dinámica que forma el SSO junto con la implementación del *Login provider* de MDM, hasta llegar a la implementación final de IDD.

17.3.4.1 Lightweight Directory Access Protocol

Cuando se habla de un servidor LDAP, se habla de un Protocolo de Acceso a Directorios a nivel de aplicación. Se hace referencia a una base de datos cuyos nombres están estructurados de forma jerárquica, en directorios y sub directorios, con el objetivo de proveer una información ordenada y facilitar el acceso a ellos.

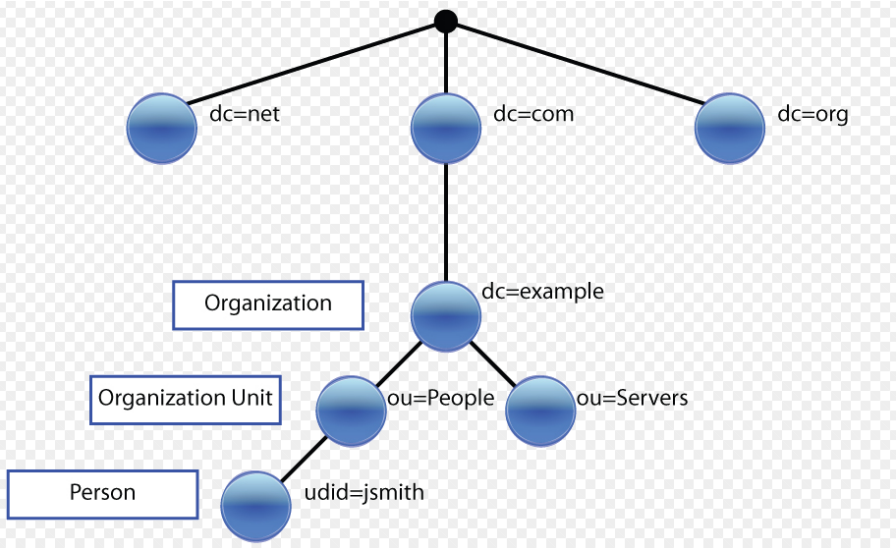


Figura 42. Árbol de directorios LDAP

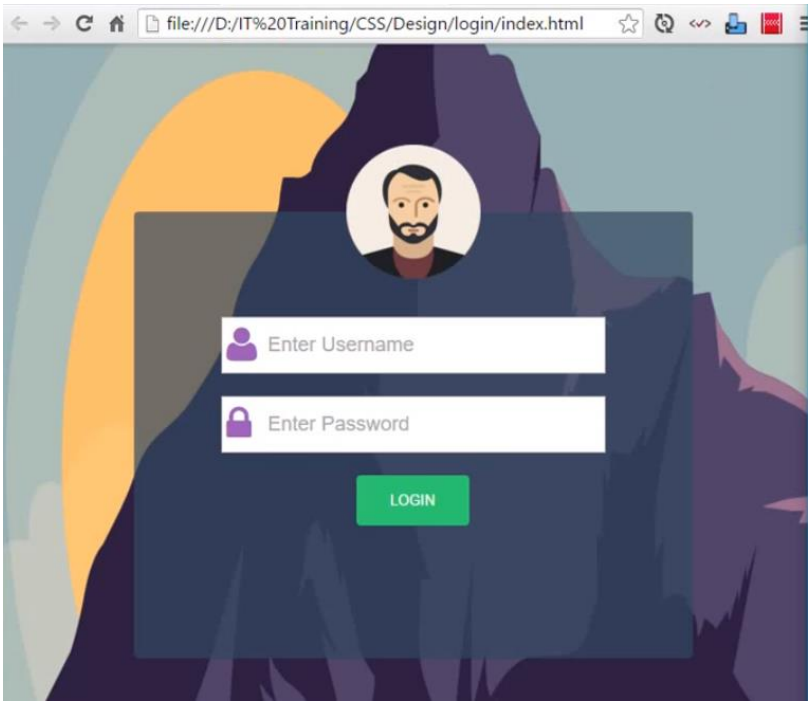
Para el desarrollo de este proyecto, la funcionalidad del LDAP se ha implementado sobre el primer servidor que forma la aplicación (**srv1**), una máquina virtual con el sistema operativo de Oracle Linux. Este LDAP se basa exclusivamente en el almacenamiento de cuentas de usuarios pertenecientes a la empresa de *arhis* con el objetivo de contrastar dicha información con la que se pueda encontrar en la base de datos MDM sobre los usuarios de la empresa de *Informatica* y poder verificar la autenticidad de los trabajadores.

17.3.4.2 Single Sign On

Para el desarrollo de la página web segura que permite acceso a una base de datos se ha desarrollado un proyecto en eclipse, programado en código java y haciendo uso de una serie de herramientas que se facilitan para trabajar con dicho lenguaje, es interesante el apartado 18 “Especificaciones del sistema” para entrar en más detalle sobre java y sus librerías.

Así pues, desplegada sobre un SO de Oracle Linux en un MV que actúa como servidor (**srv2**), al igual que su componente predecesor, la página web se estructura de forma sencilla, se basa en el simple “introduzca usuario + contraseña” que se puede encontrar en cualquier Login de internet en la actualidad, variando únicamente el diseño en función del código *css* que se haya utilizado.

El SSO del proyecto es por tanto una página web dinámica desplegada sobre un servidor respaldado por el contenedor de servlets T*omcat.* Los *servlets* han sido un elemento clave que ha permitido el empaquetamiento de cabeceras, para guardar la información sobre el usuario, y el envío de *requests* y *responses* contra el LDAP para comprobarlas, tal y como se ha mencionado en el apartado anterior de flujo del sistema.



Captura 4. Página web dinámica SSO

La implementación se ha situado en segundo en la línea temporal de la implementación del proyecto, puesto que depende directamente del servidor de nombres.

17.3.4.3 Master Data Manager

Entrando en el uno de los últimos apartados se llega a la parte de la empresa asociada, *Informatica.* El componente del proyecto que conforma el MDM se ha montado sobre el tercer servidor (**srv3**) y el sistema operativo de Red Hat Enterprise.

Ha sido posible llevar a cabo su implementación junto con la del SSO mencionado anteriormente, puesto que no tienen ninguna dependencia entre ellos. No obstante, en la tabla de dependencias entre sistemas se observa también que MDM y LDAP están "relacionados" indirectamente. Esto es debido a que el objetivo del proyecto no es otro que la autenticación de los usuarios. El LDAP trabaja como una entidad de confianza del MDM de forma que, aunque un servidor no puede ver al otro y viceversa, ambos comprueban sus datos de forma indirecta a través de los otros componentes que forman la aplicación.

Para la implementación de esta parte se ha utilizado una máquina facilitada por la misma empresa *Informatica* para el *training* de MDM, con su SO predefinido, pero 100% compatible con el resto de servidores de Oracle Linux. De esta forma se ha ahorrado en costes puesto que no ha sido necesario adquirir una nueva máquina y, a su vez, se ha decidido aprovechar una máquina obsoleta y convertirla en una parte del proyecto desarrollado.

A continuación, se presenta una figura que muestra cómo se estructura la máquina de *trainging* de MDM a nivel de directorios.

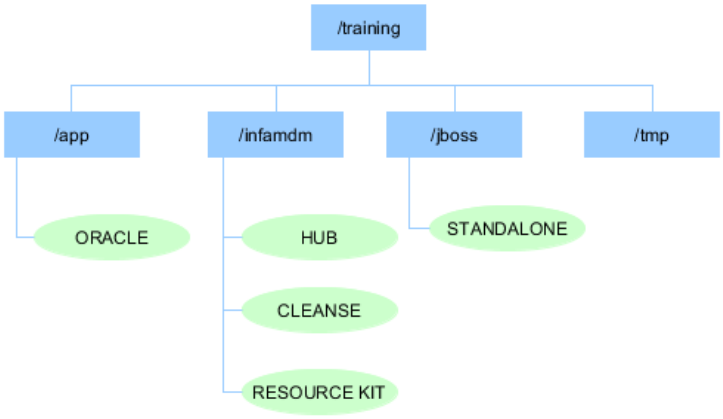


Figura 43. Arquitectura de la máquina de MDM

Se observa que la máquina tiene un directorio exclusivo para la aplicación de la empresa, *Informatica MDM*. Se divide en subdirectorios que conforman la base de datos de la aplicación. Para entrar en más detalle se puede revisar el anexo 17.2 "Estudios con entidad propia" donde se habla sobre dicha herramienta. Es interesante tener en cuenta que se dispone de un subdirectorio de Oracle y otro con el jboss, que proporciona soporte al servidor de MDM. En el apartado de 6.1 "Definiciones" se detallan más estos términos.

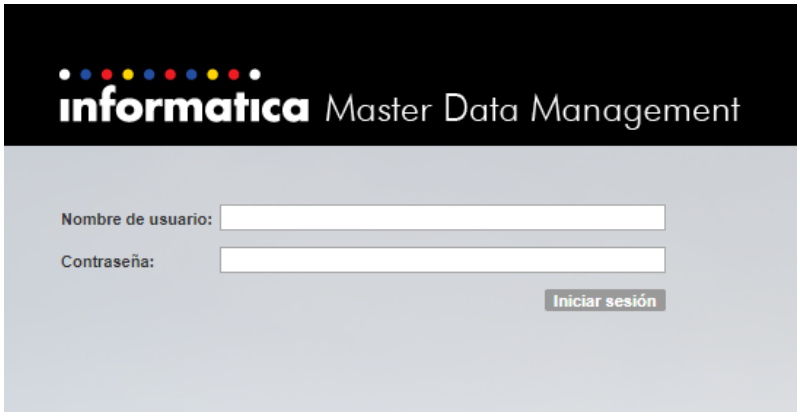
En el subapartado del HUB se puede encontrar un fichero del tipo .jar que ha sido modificado con el objetivo de adaptar MDM al proyecto. Este fichero es el llamado *Login Provider,* que ha dado la opción de facilitar el acceso a la base de datos de MDM y comparar los usuarios con los del LDAP, autorizando, de esta forma, a los trabajadores de la empresa de *arhis.*

17.3.4.4 Informatica Data Directory

Para acceder a la base de datos de Informática MDM, se dispone de una interfaz que actúa como puente entre el usuario y MDM. IDD es, por tanto, como un subconjunto de MDM que funciona de puente entre el usuario y la base de datos de la misma. Inicialmente estaba pensado que IDD trabajara en la MV número cuatro (srv4). No obstante, despúes de un análisis se ha llegado a la conclusión de colocar tanto IDD como MDM en el mismo servidor (**srv3**).

Como se ha comentado en apartados anteriores, IDD vuelve a autentificar el usuario y contraseña. Por tanto, inicialmente IDD actúa, también, como una página web de Login. En este proyecto, IDD también se modifica desde el *Login Provider* facilitado por MDM. Así pues, se han realizado una serie de cambio convenientes (que se explican a continuación en la fase de implementación) para que el trabajador no tenga que volver a introducir sus credenciales: Estas ya van incluidas en la cabecera de su petición, que IDD trata y envía a MDM, dejando acceder al usuario o no, en función del estado del *header* de la *request.* A continuación, se ve una imagen de la misma.

Nota: Cabe remarcar que el hecho de desplegar MDM e IDD en servidores separados solo afectaría en tener que replicar el *Login Provider* y cambiar la redirección.



Captura 5. Pantalla Login de IDD

17.3.4.5 Balanceador

En este apartado se introduce el concepto de balanceador, que ha sido implementado en la máquina virtual número cinco (***srv5***), con el objetivo de organizar los servidores de nuestra red de forma horizontal. Uno de los softwares más comunes para este tipo de implementación es el conocido HA-Proxy. Además de ser *opensource*, HA-Proxy nos proporciona la funcionalidad necesaria para el correcto funcionamiento de nuestro proyecto.

El servidor con el balanceador dispone de dos adaptadores de red. Con el primer adaptador (llámese *A*) se pretende que el *srv5* tenga una conexión directa con internet. Por otra parte, el segundo adaptador (llámese *B*) será el encargado de conectar el balanceador con el resto de los servidores, situados en una red virtual aislada, invisible para el resto de las máquinas.

De esta forma, el HA-Proxy hace de intermediario entre internet y dicha red privada que forman nuestros servidores. Más adelante se explica la gestión de HTTP *headers* que lleva a cabo el balanceador para enviar información entre el SSO y el LDAP.

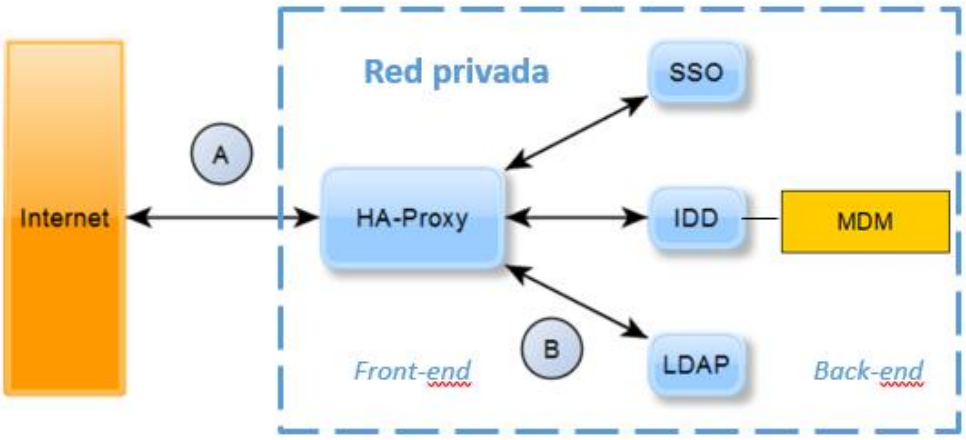


Figura 44. Conexiones con el balanceador

Cabe remarcar que la implementación de un proxy de seguridad es, sin más, una mejora que aporta robustez y eficiencia al proyecto. Es por esta razón que la implementación de la misma se sitúa en último lugar en la línea temporal y que, inicialmente, no se había ni siquiera contemplado la posibilidad de añadir dicha funcionalidad. A terminado siendo una parte imprescindible del proyecto ya que ha mejorado en gran medida la calidad del producto final.

17.4 Anexo: Implementación

17.4.1 Implementación inicial. El entorno de trabajo

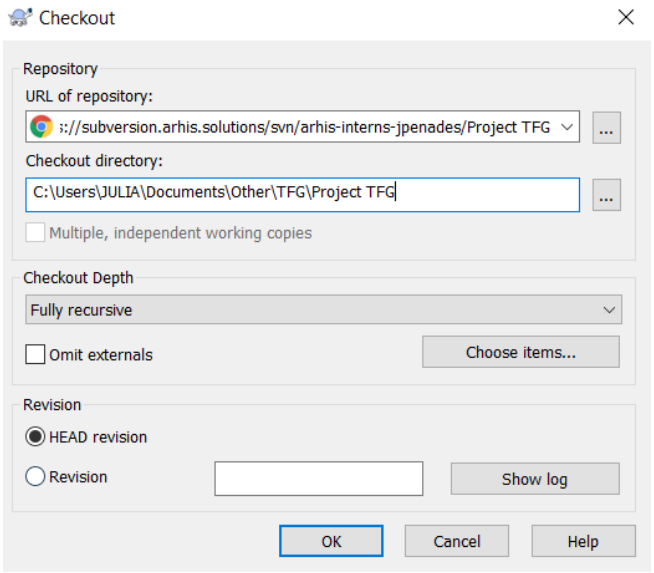
17.4.1.1 Preparación del repositorio

En la empresa de *arhis* se trabaja con VisualSVN Server, un Cliente *Subversion*. Así pues, en primer lugar, se ha llevado a cabo la creación del repositorio donde se encuentra guardada toda la información del proyecto (herramientas, programas, documentación, etc.). Para acceder se ha creado una cuenta propia con su usuario y contraseña, proporcionados por la misma empresa:

- usuario*:* jpenades
- contraseña*:* ********

Captura 6. Repositorio del proyecto

A su vez, para agilizar el acceso a dicha información, se ha hecho uso de la herramienta *Tortoise SVN,* un sistema de control de versiones que permite sincronizar las carpetas del ordenador con las que se encuentran en el servidor. Así pues, se ha elegido una ruta conveniente y se ha utilizado la función "SVN *Checkout*": Se introduce el URL del repositorio del proyecto en SVN Server y este se sincroniza.



Captura 7. Checkout del repositorio del proyecto

17.4.1.2 Preparación de las Máquinas Virtuales

Oracle VM VirtualBox ha sido una herramienta imprescindible para simular el sistema de forma que sus componentes estén divididos en servidores distintos. Así pues, para la simulación de la red y los servidores que la componen se ha hecho uso de este programa. Se han implementado

cuatro máquinas, que inicialmente eran cinco, pero se ha procedido a trabajar con MDM e IDD conjuntamente. Sobre cada una de ellas se ha montado cada uno de los servidores que intervienen en la aplicación. A continuación, se muestran las características que van a tener nuestras máquinas:

- Tipo: Linux, Versión: Oracle (64-bit).
- Tamaño de memoria: 1024MB de RAM. Puesto que el trabajo que van a realizar las máquinas no es muy costoso, con un mínimo de RAM funcionará a la perfección. Conseguimos, además, reducir el tiempo de respuesta de las mismas y, como consecuencia, aumentar su velocidad.
- Disco duro: Irrelevante, por defecto la máquina se crea con 12GB.

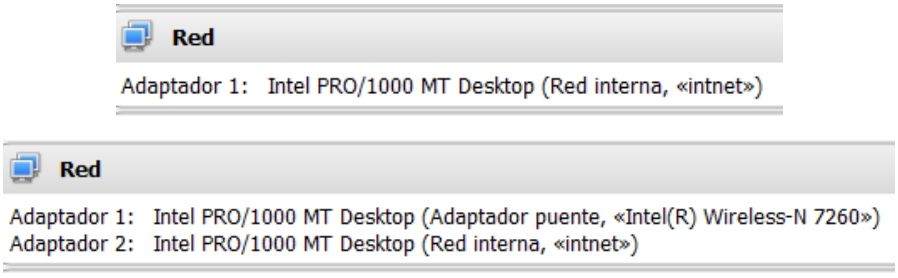
La memoria RAM para la máquina de MDM sí que es más grande, puesto que ahí se almacena toda la información de usuarios y roles. No obstante, como ya se ha mencionado anteriormente, para el MDM se dispone de una máquina que ha sido facilitada por la misma empresa de *Informatica* y que lleva el sistema operativo RHEL, por tanto, no se va a entrar en detalles al respecto.

Nota: Para el funcionamiento de la aplicación, es necesario que todas las máquinas estén encendidas.

Otro elemento imprescindible para la implementación del proyecto ha sido la herramienta *Cygwin.* Ha ayudado a conectarse con las máquinas de forma eficiente y utilizar comandos de Linux para completar las tareas de desarrollo (sobre todo en el HA-Proxy, como se ve más adelante).

Es imprescindible la implementación de la red que se ha comentado en los apartados anteriores. Para ello, se ha accedido a las configuraciones de cada una de las máquinas. Se han seguido los siguientes pasos:

- Montar red virtual aislada (local) donde solo se ven SSO, LDAP, MDM/IDD y HA-Proxy: todos los componentes tienen un Adaptador de Red interna (inet).
- Modificar HA-PROXY como punto de entrada a todas las conexiones del entorno, añadiendo una ethernet para internet: El HA-Proxy tiene, además, un adaptador que lo conecta con el exterior (para que se pueda acceder remotamente).



Captura 8. Configuración de la red de las MV

En la captura se observa la primera red: un adaptador 1, configurado para el SSO, LDAP y MDM/IDD. La segunda red corresponde al Ha-Proxy, con dos adaptadores (el de la red interna, que lo conecta con el resto de componentes, y el Wireless, que lo conecta con el exterior). En la *Figura 43* del apartado anterior se ve clara la estructura final.

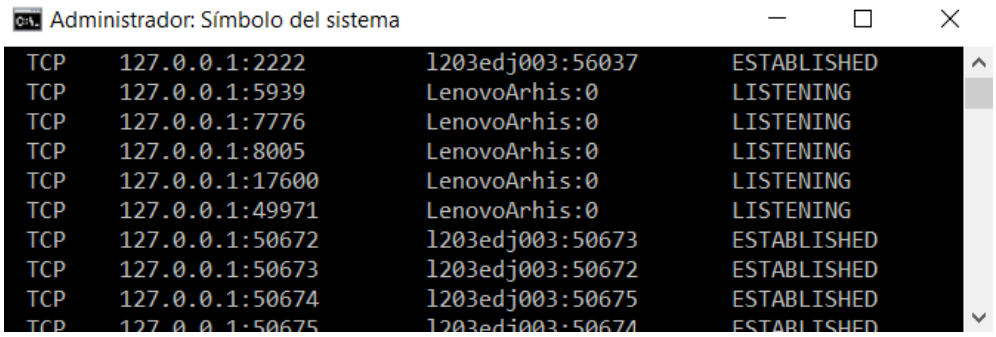
Nota: Para el diseño inicial, se había llevado a cabo una configuración NAT. Para ello, ha sido necesario el uso de puertos. Se ha procedido a buscar puertos de la red local (*localhost*) disponibles pero ocupados (*established)* y aquellos que están en escuchando (*listening*), para poder activar la conexión remota. Se necesita ejecutar el siguiente comando en la consola de Windows:

- Puertos escucha:

```
netstat -tlpn
```

- Puertos en funcionamiento:

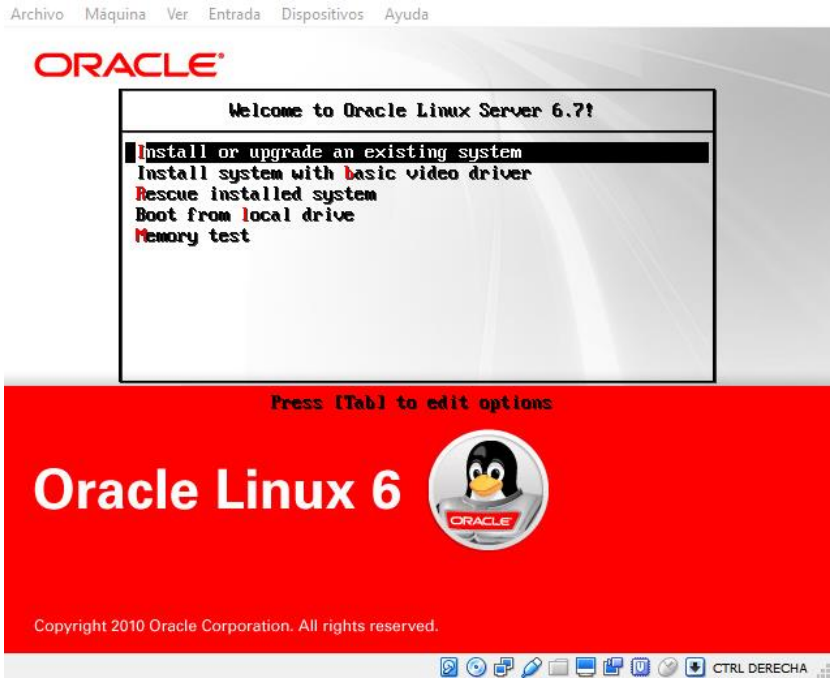
```
netstat -tpn
```



Captura 9. Puertos de la red local

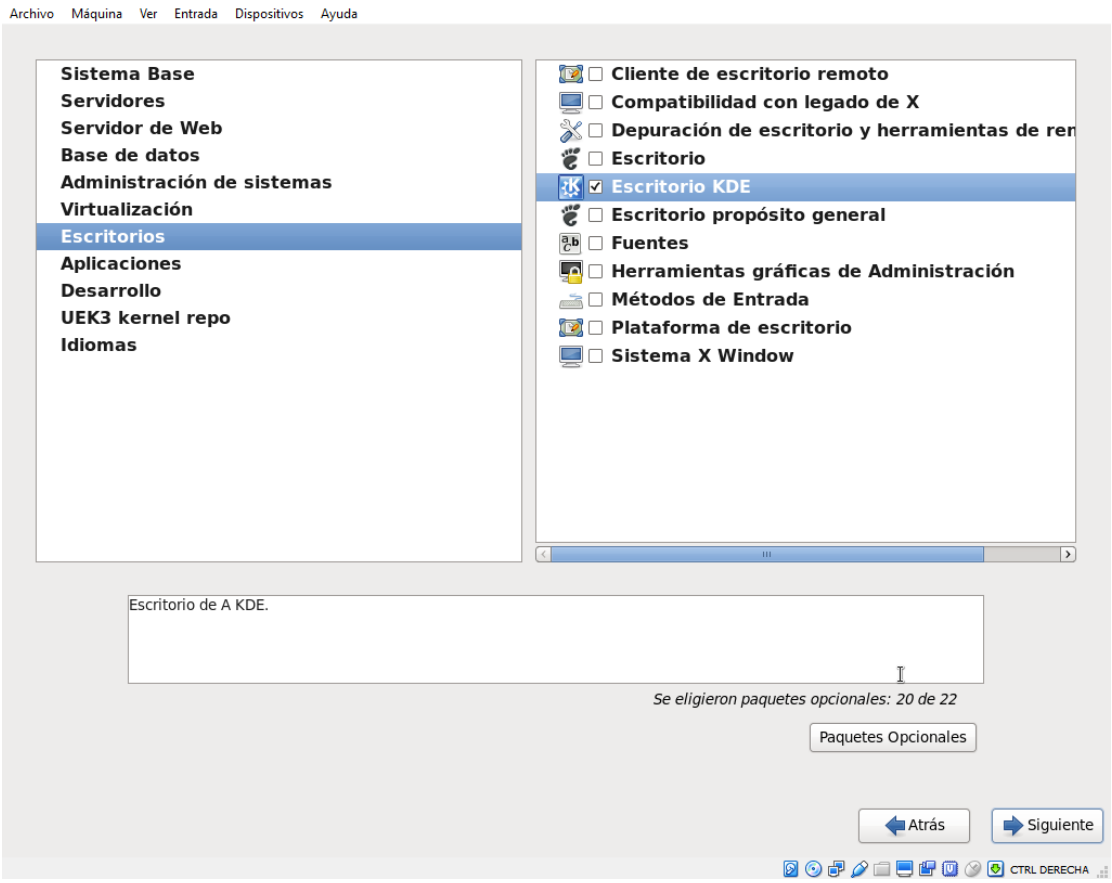
17.4.1.3 Instalación de los Sistemas Operativos

Una vez preparadas las máquinas, se ha pasado a la instalación del Sistema Operativo (SO). Con la intención de seguir en la línea de productos de Oracle, cada uno de los servidores cuenta con un SO de Linux, concretamente Oracle Linux en su versión 6.7 (Excepto MDM, que lleva por defecto RHEL). A continuación, se ve una guía rápida de la configuración elegida para los SO.



Captura 10. Instalación del SO

Con el objetivo de sobrecargar al Sistema operativo con la mínima información posible, se ha decidido hacer una instalación limpia, únicamente con el Escritorio KDE, puesto que se puede trabajar únicamente mediante comandos y garantizamos, además, la fluidez del sistema.



Captura 11. Configuración inicial del SO

Inicialmente se ha utilizado para cada uno de los servidores un usuario base, para ahorrar posibles errores futuros y puesto que el proyecto no va ser comercializado nada más acabarlo (se trata de una versión en fase de prueba). Se dispone del siguiente usuario y contraseña:

- User: root
- Password: 123456

Para el LDAP se han añadido nuevos usuarios con el objetivo de poderlos utilizar en un futuro. Para ello el sistema de Oracle Linux utiliza unos comandos, disponibles solo desde su usuario root (*admin*). Facilita, también, la posibilidad de crear grupos y añadir los usuarios a dichos grupos:

```
useradd jpenades
passwd jpenades
123456
groupadd developers
```

Los detalles de los usuarios están visibles en el fichero de la ruta predefinida por el sistema operativo "/etc/passwd". Si no se especifica ningún *User* ID (UID), se asigna el siguiente UID más grande. También se crea un nuevo grupo con un nombre de grupo que coincida con el nombre

de usuario. De forma predeterminada, los usuarios de inicio directorio se crea bajo el directorio "/ home" y el shell es "/ bin / bash".

17.4.2 Implementación del LDAP

En el primer servidor que forma este proyecto, llamado **srv1.localdomain**, se ha montado el LDAP en el que se van a almacenar las cuentas de usuarios de *arhis.* El srv1, tal como se concluyó en el apartado anterior, está construido sobre una máquina virtual con Oracle Linux como sistema operativo.

Con el objetivo de configurarlo para que trabaje con el SSO (implementado en el siguiente apartado) se han tenido que facilitar permisos. En caso de trabajar con direcciones IP (algunas veces ha resultado necesario) se tiene que configurar para que la máquina sea capaz de reconocerlas. Para el objeto de este proyecto, puesto que se trabaja con una conexión *bridge* interesa apagarla para que no bloqueo.

1. Comprobar el estado de las IP:

```
service iptables status
```

2. En caso de estar encendida, apagarla:

```
service iptables stop
```

3. En caso de no querer apagarlo, También es posible modificar su configuración para que acepte las conexiones que se quiera agregar:

```
vi /etc/sysconfig/iptables
service iptables relod
iptables -nvL
```

El SO Oracle Linux dispone de su propio *firewall* llamado *Selinux*. Este *software* proporciona una seguridad que no permite hacer cambios en documentos de forma remota, por tanto, es necesario apagarlo. Para ello se han seguido los pasos que se muestran a continuación.

```
vi /etc/selinux/config
SELINUX=disabled
Reboot
```

Cabe remarcar la importancia de reiniciar el sistema cada vez que se han realizado cambios importantes.

Después de una búsqueda de información, se ha decidido hacer uso del *software* OpenLdap, puesto que tiene gran variedad de opciones y facilita el trabajo de implementación del servidor de nombres en Linux. En el último apartado de bibliografía se hace referencia a las páginas que han proporcionado información sobre este programa.

17.4.2.1 OpenLdap

En el apartado de diseño se ha presentado la estructura que tiene un directorio LDAP. No obstante, es interesante revisar las diferentes partes que lo forman. OpenLDAP facilita la siguiente estructura orientada a objetos en LDAP:

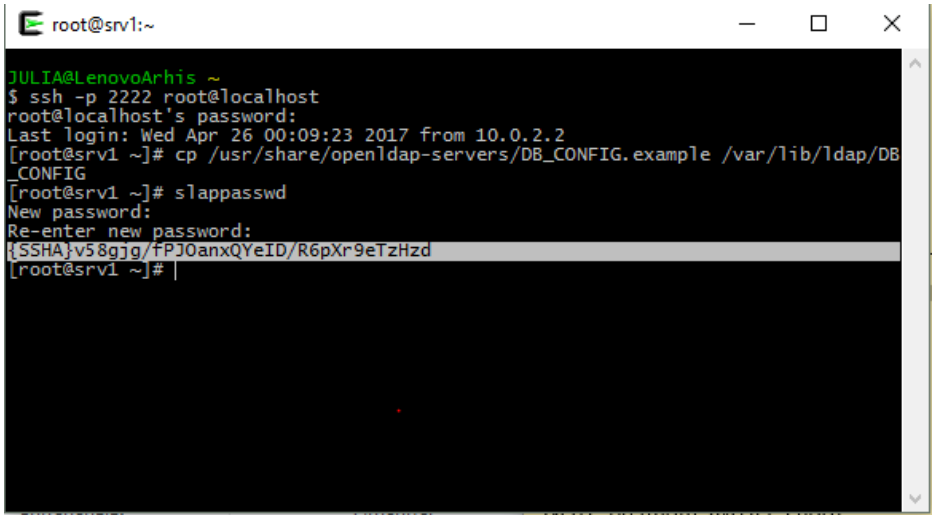
- Clases: Se definen los objetos y sus características. Por ejemplo, el tipo de objeto que se va a definir y los atributos que va a contener en función del tipo de objeto. En el esquema se define cada clase con los atributos que serán obligatorios y opcionales para cada entrada creada.
- Objetos: los objetos son entradas en el directorio. Los objetos son instancias creadas a partir de una determinada clase o de varias, en función de los atributos necesarios para un objeto. Todo el directorio va a estar compuesto por objetos (usuarios, grupos, unidades organizativas, …).
- Atributos: Son los campos asociados a cada objeto creado y definen las características del mismo. Cada atributo tiene un valor (información). Por ejemplo, el nombre del usuario, apellidos, número de teléfono, etc.
- DN: el campo DN es el nombre distinguido (*Distingued Name*) para identificar de forma única a un determinado objeto en el directorio. Es decir, cada entrada definida es única en todo el directorio.

A continuación, se muestran una serie de comandos que se han utilizado para la creación de los directorios, así como una serie de capturas con los resultados obtenidos.

Para la instalación, desde los privilegios de usuario root:

```
yum -y install openldap openldap-clients openldap-servers
```

A continuación, ha sido necesaria la creación de una contraseña para el usuario *admin* del LDAP que se creará más tarde. Para garantizar la seguridad de la misma, se ha utilizado una forma de encriptación como la que se ve en la captura siguiente.

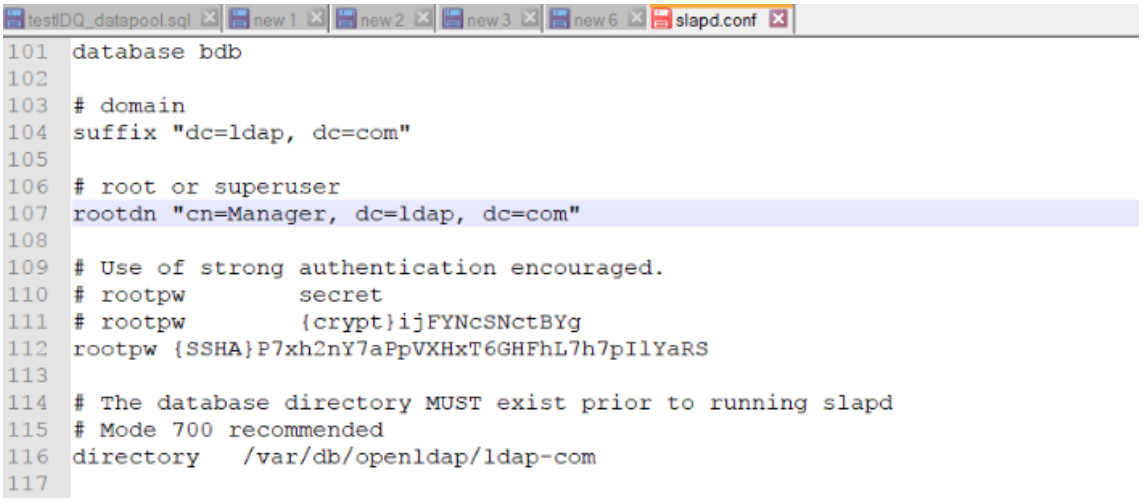


Captura 12. Clave encriptada del admin en LDAP

Seguidamente, se ha configurado el LDAP en función al objeto proyecto, para ello ha sido necesario recurrir al archivo *.conf.* Así pues, es posible acceder a dicho fichero mediante el siguiente comando:

```
vi /etc/openldap/slapd.conf
```

A continuación, se muestra una imagen de la estructura que tendría visto desde el editor del SO. La captura corresponde a la configuración inicial del LDAP, en el que se usa el usuario y contraseña de Manager (*admin*) y la contraseña (creada anteriormente con encriptación). Una vez editado este fichero, el OpenLdap ofrece la opción de trabajar con sus comandos para añadir nuevos usuarios y grupos.



Captura 13. Fichero configuración de openLdap

Una vez configurado el fichero slapd, en el que, como ya se ha mencionado, se dan los permisos, se puede proceder a añadir nuevos usuarios y roles. Para ello es necesario introducir el dominio en el que se quiere guardar (tfg-ldap.com) así como la contraseña sin encriptar (se ha utilizado también la de por defecto, 123456).

```
ldapadd -f tfg-ldap.ldif-D cn=Manager,dc=tfg-ldap,dc=com -w 123456
```

OpenLdap ofrece, además, la posibilidad de acceder a la información almacenada en su base de datos en cualquier momento. La estructura que devuelve es, también muy esquemática y fácil de interpretar.

```
ldapsearch -x -LLL -D cn=Manager,dc=tfg-ldap,dc=com -w 123456 -b dc=tfg-ldap,
dc=com
```

A continuación, se muestra una captura con información sobre la base de datos después de haber creado grupos y clases. En la captura se observa la distribución esquemática de los usuarios. Se puede ver en que grupos se integran y sus datos, como el equipo al que pertencen (que sirve para identificar el rol de cada persona) y la contraseña del usuario, entre otras (todo depende también de la información que se desea añadir). Inicialmente solo interesa tener el usuario y contraseña.

Captura 14. Estructura de directorios del LDAP

Nota: Otros comandos de búsqueda que también son interesantes:

```
ldapsearch -x -LLL -b dc=tfg-ldap,dc=com
ldapsearch -H ldapi:// -Y EXTERNAL -b "cn=config" "(olcRootDN=*)"
```

Una vez configurado el directorio, es necesario activar el LDAP. Para ello se hace uso del servicio ofrecido por el mismo software. Se introducen los comandos que se muestran a continuación.

```
vi /etc/openldap/slapd.d/cn\=config/olcDatabase\=\{
/etc/rc.d/init.d/slapd restart
ps auwx | grep slapd
```

Así pues, una vez se comprueba con el comando “grep” que el sldap está en funcionamiento, ya se puede pasar a la fase de implementación del SSO para empezar enviar peticiones y a comparar credenciales.

Nota: Si el slapd no está en funcionamiento, es suficiente con introducir los siguientes comandos:

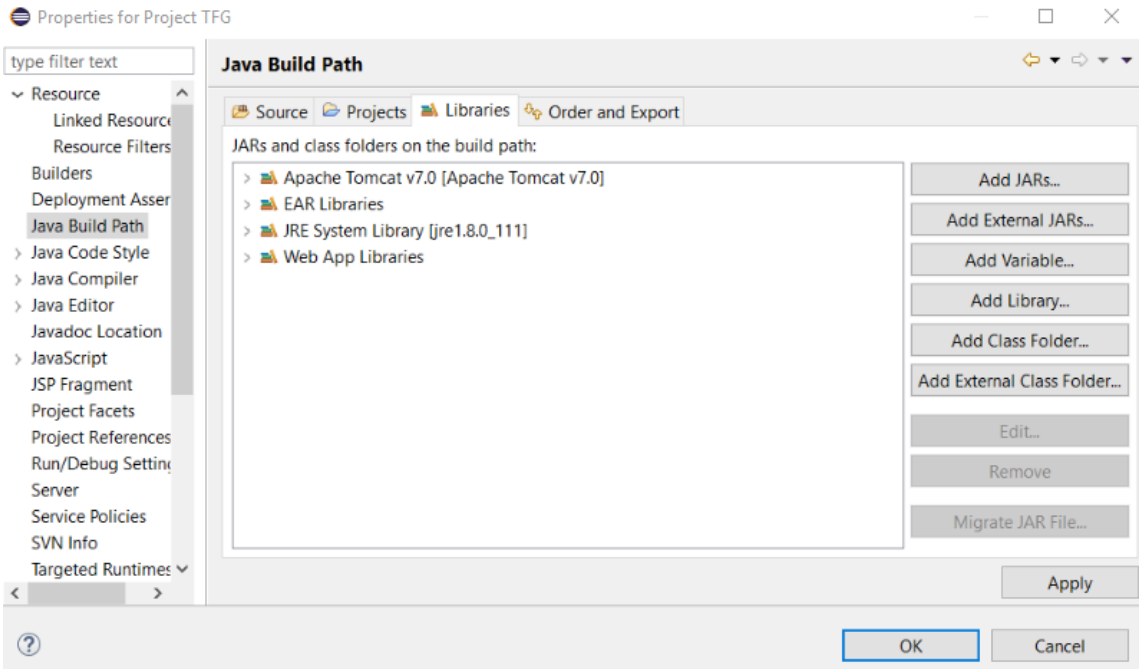
```
chkconfig slapd on
service slapd start
```

17.4.3 Implementación del SSO

El sistema de sesión único SSO, estará hospedado en el segundo servidor, al cual se ha decidido nombrar srv2.localdomain acorde con su predecesor el servidor del LDAP. Al igual que en el apartado anterior, se ha hecho una instalación del Sistema Operativo de Linux Con un Set Minimal y con escritorio KDE.

Para el correcto funcionamiento de este sistema, se ha desplegado una aplicación en código java, que ha sido diseñada con la herramienta eclipse (véase el Anexo 17.2, Estudios con entidad, para más información).

Para lo que importa a la empresa, en JAVA se pueden programar páginas web dinámicas, con acceso a bases de datos, utilizando XML. No obstante, para ello ha sido necesario añadir diversas librerías y funcionalidades al proyecto. Inicialmente, una vez cargado en eclipse, ha sido necesario añadir la siguiente configuración:



Captura 15. Liberías del Proyecto SSO

- Java Build path: La ruta en la que está instalado el java, en una de sus versiones más recientes(jre1.8)
- Apache Tomcat v7.0: Imprescindible para el funcionamiento del LDAP. Para ello ha sido necesario instalar el paquete y activar el servidor. Se han seguido los siguientes pasos:
    - *Target Runtimes* (tiempo de ejecución) --> Apache Tomcat v7.0 (Se ha elegido esta versión por ser una de las mejor documentadas).
    - *View* --> Show view --> "Servers" --> Create a new server.
    - Apache --> Tomcat v7.0 Server --> Seleccionar el nombre del servidor (srv2) --> *download* and *install.*

Una vez preparado el entorno y teniendo el LDAP en marcha (muy importante) se ha procedido al diseño del SSO. Este consiste en dos partes: El diseño de la página web en código *html* ***(Login.jsp)*** y la configuración de la misma (**SSOLogin.java**) escrita en código java.

17.4.3.1 Diseño del Login.jsp

El diseño de la página web ***Login.jsp*** tiene una implementación sencilla. A continuación, se muestra una captura de pantalla en el que se muestra el código escrito, con el objetivo de explicar su funcionalidad.

SSOLogin.java | http://localhost:8080/Project_TFG/SSOLogin | config.properties | Login.jsp

```
<%@page contentType="text/html" pageEncoding="ISO-8859-1" %>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="ISO-8859-1">
        <title>Authentication Service</title>
    </head>
    <body>
        <h1>Login</h1>
        <form method=post action="SSOLogin">

            <input type="text" placeholder="Enter Username" name="user" required="required"/>
            <br/><input type="password" placeholder="Enter Password" name="pass" required="required"/>
            <br/><input type="submit" value="Let me in"/>

        </form>
    </body>
</html>
```

Captura 16. Login.jsp

Primero se encarga de leer la información que se le introduce en sus *placeholders.* Como se ha visto en el apartado anterior, los login disponen de unas "casillas" en las que introducir dicha información.

- Insert user: para introducir el usuario.
- Insert password: para introducir la contraseña.

Así pues, una vez hecho esto, el Login dispone de un botón de *submit* cuya función es el envío de la información que se introduce, este caso el par de valores usuario, contraseña.

Una vez llevado a cabo este paso, entra en funcionamiento el archivo de configuración del SSO (**SSOLogin.java**). Este archivo está directamente relacionado con el Login, y se encarga de que la información con la que se ha hecho el *submit* sea tratada. Para que todo esto sea posible, es necesario importar las siguientes funcionalidades:

```
package com.arhis.tfg.julia;

import java.io.IOException;
import java.io.InputStream;
import java.util.Hashtable;
import java.util.Properties;

import javax.naming.Context;
import javax.naming.NamingEnumeration;
import javax.naming.NamingException;
import javax.naming.directory.*;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

Captura 17. Imports para el SSO

- El fichero *properties* trabaja con los datos para estructurarlos de la misma forma que el LDAP y poderlos validar. Lo mismo pasa con los *naming.*

- La *Hashtable*, como se ha comentado en apartados anteriores, sirve para trabajar con pares de datos (clave, valor).
- Los *servlet* son imprescindibles para el intercambio de *request* y *response*. Facilitan que el SSO trabaje con el LDAP y le envíe la información en forma de peticiones.

A continuación, se muestra una primera implementación con las funciones que ofrece la librería de *servlet*.

```
/**
 * @see HttpServlet#HttpServlet()
 */
public SSOLogin() {
    super();
    // TODO Auto-generated constructor stub
}

/**
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    // TODO Auto-generated method stub

    if(this.isValidUser(userName, password)){

        Cookie token = new Cookie("ARHIS_USER", userName); // Better use headers to send
        information (more common)
        response.addCookie(token);

        response.addHeader("Auth",
        "SECRET_TOKEN=23423423423423423wqerqwerqwerqwer324rwefqwerqwer");
        request.getRequestDispatcher("/IDD.jsp").forward(request, response);

        //response.sendRedirect("http://google.com"); // Header blocked by the proxy: it
        is not the header the page is looking for

    }else{

        if (userName != null && password != null){
            response.addHeader("User", userName);
            response.addHeader("Auth", "ERROR, Wrong User/password, try again");
            request.getRequestDispatcher("/LoginErr.jsp").forward(request, response);
        }
    }
}
```

Captura 18. Tratamiento de la request/response

En esta captura se muestra como el código comprueba si un usuario es válido o no, y lo trata. En caso de ser válido, añade la información como *cookie* y como *header*. Se decidió tener en cuenta las dos posibilidades para poder tener más opciones a la hora de tratar los datos enviados.

En caso de rechazarse la información, se envía un mensaje de error y se redirecciona a una página llamada LoginErr.jsp. Puesto que la implementación de la misma es muy similar a la observada para el Login.jsp, pero mostrando un error de autenticación, no se va entrar en detalles.

17.4.3.2 Diseño del SSOLogin.java

A continuación, se va a entrar en detalle sobre la implementación del método de validación. En esta parte el SSO pasa por dos fases:

- Se realiza una conexión contra el LDAP. En la *Captura 19* se puede ver parte del código en el que se conecta con el servidor de LDAP y se busca su información. El código comentado se muestra de color verde, con el objetivo de poder hacer un seguimiento del funcionamiento del método.

- Se accede a la información del LDAP y se intenta organizar la misma, desplazándose entre los diferentes niveles que la conforman (en este caso solo se dispone de un grupo con usuarios amlacenados).

```
Properties props = new Properties();
props.load(in);

// Second, create the context
DirContext context = null;

try{

    // Use the service user to authenticate
    Hashtable<String, String> env = new Hashtable<String, String>();
    env.put(Context.INITIAL_CONTEXT_FACTORY, props.getProperty("INITIAL_CONTEXT_FACTORY"
    ));
    env.put(Context.PROVIDER_URL, props.getProperty("PROVIDER_URL"));
    env.put(Context.REFERRAL, props.getProperty("SECURITY_REFERRAL")); // Default:
    follow (To get rid of the PartialResultException when using AD)
    env.put(Context.SECURITY_AUTHENTICATION, props.getProperty("SECURITY_AUTHENTICATION"
    ));
    env.put(Context.SECURITY_PRINCIPAL, props.getProperty("SECURITY_PRINCIPAL"));//
    adminuser - User con privilegios especiales, dn user
    env.put(Context.SECURITY_CREDENTIALS, props.getProperty("SECURITY_CREDENTIALS"));//
    dn user password
    context = new InitialDirContext(env);

    // For now, we only need the identifier (uid)
    String[] attributeFilter = {props.getProperty("IDENTIFYING_ATTR")};
    SearchControls sc = new SearchControls();
    sc.setReturningAttributes(attributeFilter);
    sc.setSearchScope(SearchControls.SUBTREE_SCOPE);

    // Use filter to find the user we need to authenticate (BASE)
    String searchFilter = "(" +  props.getProperty("IDENTIFYING_ATTR") + "=" + userName +
     ")";
    NamingEnumeration<SearchResult> results = context.search(props.getProperty("BASE"),
    searchFilter, sc);
```

Captura 19. Conexión con el LDAP

- Una vez conectado con el LDAP y obtenida su información, el programa intenta autenticar al usuario. En este momento es cuando se tiene en cuenta la información suministrada por el Login.jsp.
    1. El *servlet* busca dentro del DN con el objetivo de acceder a los usuarios.
    2. Los reslutados son guardados en una tabla Hash <clave, valor>, que en este caso corresponde al <usuario, contraseña>.
    3. Pasa lo mismo con los usuarios que se están intentando autenticar.
    4. Se compara los resultados. En caso de ser válido, se procede a pasar al método que se ha comentado al inicio del apartado (en el que se genera el *token* y se guarda la información en la cabecera de la petición)
    5. En caso de no ser válido, saldrá un mensaje de error de autenticación (Que también se ha comentado anteriormente.

```
        if (results.hasMore()) {

            // Get the users DN (distinguishedName) from the result
            SearchResult result = results.next();
            String distName = result.getNameInNamespace();

            // Attempt another authentication, now with the user
            env = new Hashtable<String, String>();
            env.put(Context.INITIAL_CONTEXT_FACTORY, props.getProperty(
            "INITIAL_CONTEXT_FACTORY"));
            env.put(Context.PROVIDER_URL, props.getProperty("PROVIDER_URL"));
            env.put(Context.SECURITY_PRINCIPAL, distName);
            env.put(Context.SECURITY_CREDENTIALS, password);
            context = new InitialDirContext(env);

            System.out.println("Authentication successful");
            return true;
        }

    } catch (Exception e) {

    } finally {
        if (context != null) {
            try {
                context.close();
            } catch (NamingException e) {
                e.printStackTrace();
            }
        }
    }
    System.err.println("Authentication failed");
    return false;
```

Captura 20. Autenticación contra el LDAP

Para poder gestionar mejor el tema las propiedades del Proyecto, se ha decidido crear un archive (*config.properties*) que se puede modificar según convenga. Este fichero es cargado directamente en el proyecto, facilitando el tratamiento del LDAP, en caso de realizar alcun cambio sobre el mismo.

SSOLogin.java | http://localhost:8080/Project_TFG/SSOLogin | config.properties

```
#Project TFG Properties

INITIAL_CONTEXT_FACTORY=com.sun.jndi.ldap.LdapCtxFactory
PROVIDER_URL=ldap://localhost:3389
SECURITY_REFERRAL=follow
SECURITY_AUTHENTICATION=simple
SECURITY_PRINCIPAL=cn=Manager,dc=tfg-ldap,dc=com
SECURITY_CREDENTIALS=123456

BASE=ou=Users,dc=tfg-ldap,dc=com
IDENTIFYING_ATTR=uid
```

Captura 21. SSO config.properties

17.4.4 Implementación de MDM e IDD

Como se ha explicado en el Anexo 17.3 en el apartado “Dependencias entre sistemas”, tanto IDD como MDM están directamente conectados. Así pues, en este apartado se explica el procedimiento llevado a cabo para la activación de ambos sistemas.

17.4.4.1 Encendido del MDM

MDM está montado sobre una Máquina Virutal ya proporcionada por la empresa, llamada *arhis-training*. Para que IDD pueda funcionar, es necesario que todos los servicios encargados de

levantar el MDM estén activos desde el principio. Para ello, se han seguido los pasos que se ven a continuación.

Para facilitar la interacción con la máquina, se vuelve a hacer uso de la herramienta *CygWin*. Se ha encendido la máquina y accedido a ella mediante una conexión *ssh* con el siguiente comando:

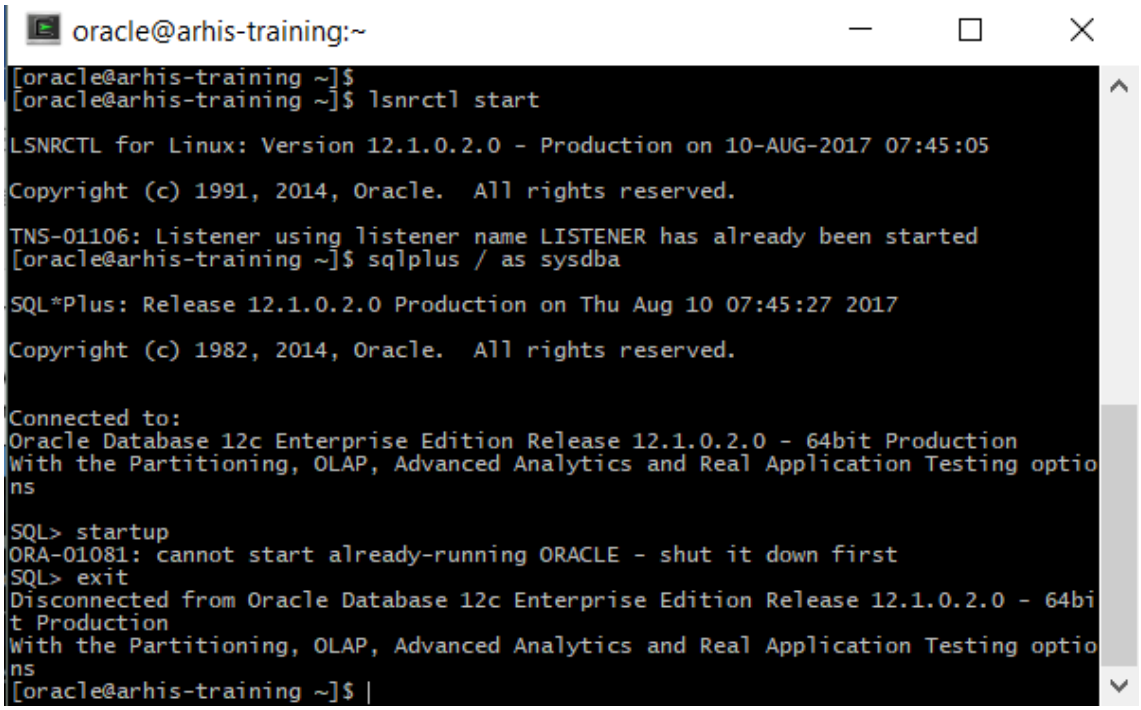
```
ssh oracle@ip-maqujna-MDM
```

Una vez autentificados como el usuario con privilegios *Oracle*, se deben realizar las siguientes tareas para tener operativo el MDM.

1. Es importante asegurarse de que Oracle está encendido correctamente, para ello se utilizan los siguientes comandos:

```
lsnrctl start
sqlplus / as sysdba
startup
exit
```

   - Se activa el *listener*
   - Se entra a SQL como *sysdba*
   - Dentro de SQL, se activa Oracle
   - Una vez Oracle ha sido activado (o si ya lo estaba) se debe salir de SQL



Captura 22. Listener y Oracle activados

2. Al igual que el SSO que se ha implementado en este proyecto MDM necesita el funcionamiento de una aplicación servidor, en este caso JBoss.

```
service jboss start
```

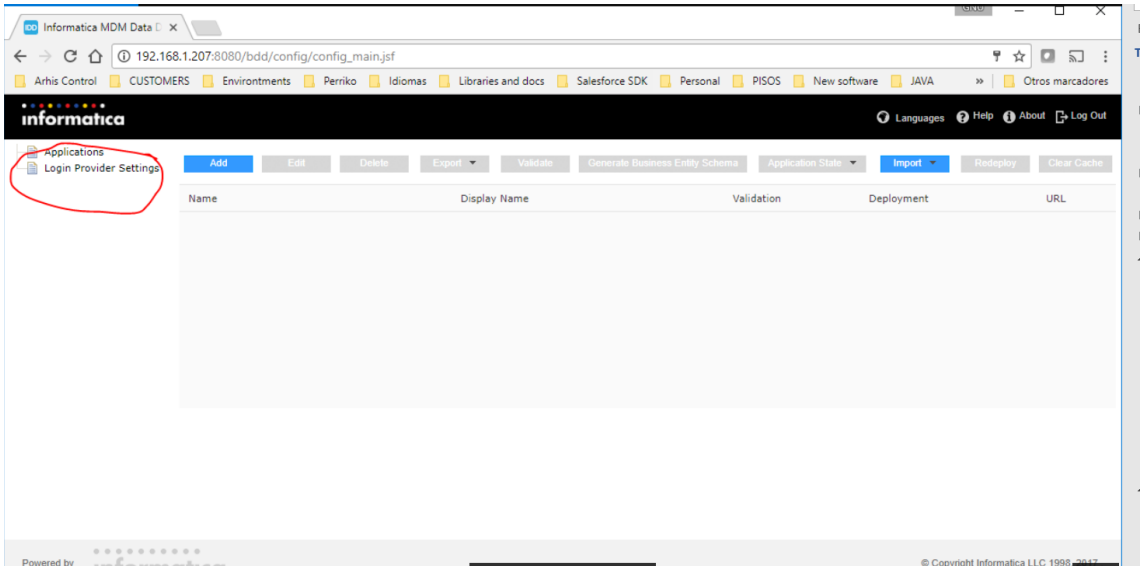
Para acceder a IDD: http://ip-maquina-MDM:8080/bdd. Si todo va bien, seguirá el flujo (redireccionará a SSO si es la primera vez).

La implementación de MDM consiste en el tratamiento del *Login Provider,* funcionalidad que ofrece *Informatica.* No obstante, puesto a que es información que pertenece a la empresa asociada, no ha sido posible entrar en detalle. Aun así, es interesante comentar algunos aspectos. A continuación, se muestra una captura con la estructura del *provider*, similar al diseño del Single Sign-on, en el que se muestra una parte el tratamiento del usuario y contraseña para ser validado.

```
private boolean isValidUser(String username, String password)
{
    boolean result = false;

    //Check login user.
    SpringSessionHolder sessionHolder = (SpringSessionHolder)SpringLookup.lookupBean(
    CommonSessionBeanRef.SPRING_SESSION_HOLDER);

    SiperianClient sipClient = sessionHolder.getSipClient();

    AuthenticateRequest request = new AuthenticateRequest();
    AuthenticateResponse response = null;

    //Set user name.
    request.setUsername(username);

    //Set user password.
    {
        Password passwordObject = new Password();
        passwordObject.setPassword(password);
        request.setPassword(passwordObject);
    }

    try
    {
        //Call SIF to check user.
        response = (AuthenticateResponse) sipClient.process(request);
        log.warn("User: " + username + " is valid");
        result = true;
    }catch(Exception ex)
    {
        log.warn("User: " + username + " is not valid");
        result = false;
    }

    return result;
}
```

Captura 23. Código del método isValidUser

Una vez terminada la modificación en el *provider* que comprueba la información de los roles y autentica y autoriza a los usuarios de la empresa *arhis,* es necesario cargar el fichero en MDM. Para ello se accede a la siguiente URL http://ip-maquina-MDM:8080/bdd/config, donde se ha cargado dicho fichero.



Captura 24. Carga del Login Provider de MDM

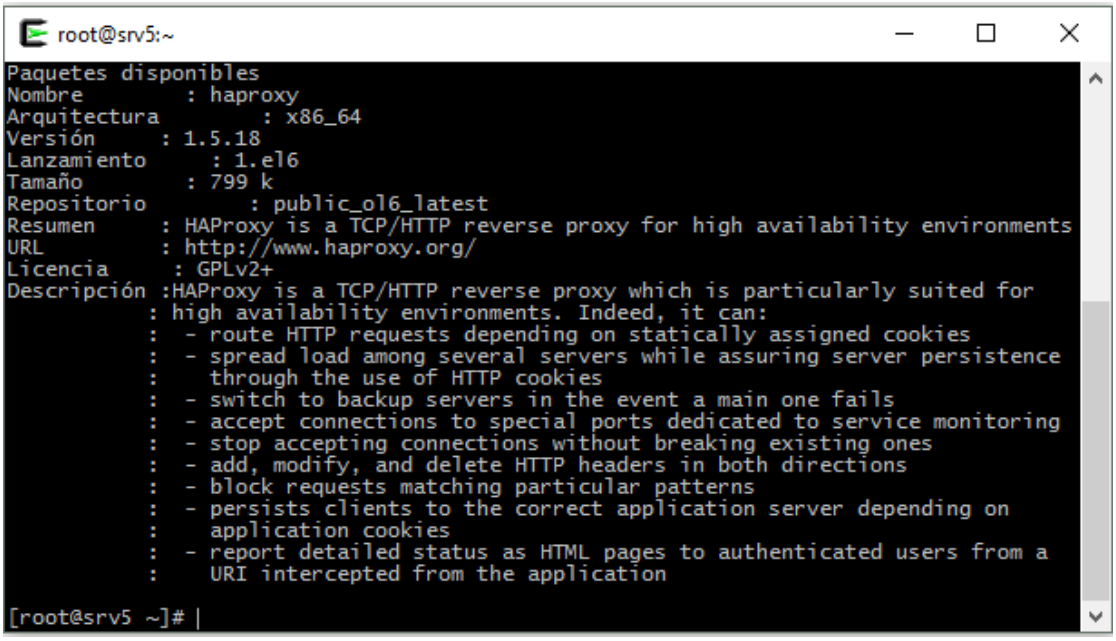
17.4.5 Implementación del Balanceador

En este apartado se procede a la explicación sobre la instalación y configuración del balanceador del proyecto, que hará de puente entre internet y la red de servidores aislados.

17.4.5.1 Instalación del HA-Proxy

Antes de empezar la instalación se ha comprobado la última versión del software en los repositorios estándares, con la intención de que el sistema esté actualizado. Para ello se ha hecho uso del comando "info". Aprovechamos también para obtener toda la información que necesitamos sobre el software.

```
yum info haproxy
```



Captura 25. Información sobre el HAProxy

Puesto que se trabaja con un usuario *root*, no es necesario el uso de "sudo" al inicio del comando. En la captura anterior se observa el resultado mostrado por pantalla.

A parte de observar que, en efecto, este software es ideal para la función de balanceador del proyecto, se puede ver el URL de su página oficial. Accediendo a esta, se ha podido revisar la última versión estable. Así pues, se ha decidido proceder a la instalación desde la fuente (*source*) para asegurar que se ha instalado la última versión del software, puesto que esta no se encuentra en los repositorios (en este caso, en la captura vemos la versión 1.5.18 y en la *source* está ya la 1.7).

Para llevar a cabo la instalación, primero ha sido necesario instalar la librería *Perl Compatible Regular Expressions* (PCRE), que nos ha facilitado las herramientas necesarias para la descarga y compilación del programa.

```
yum install wget gcc pcre-static pcre-devel -y
```

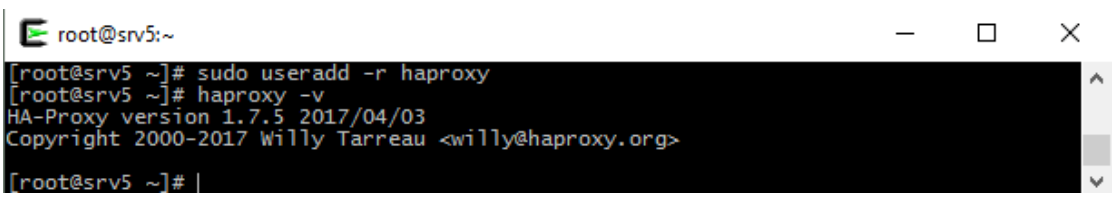
Para llevar a cabo la instalación, primero ha sido necesario instalar la librería *Perl Compatible Regular Expressions* (PCRE), que nos ha facilitado las herramientas necesarias para la descarga y compilación del programa. A continuación, se ha procedido, mediante al uso de comandos similares, a la instalación del software base del balanceador.

```
wget http://www.haproxy.org/download/1.6/src/haproxy-1.6.3.tar.gz -O ~/haprox
y.tar.gz
```

Nótese que se ha descargado un archivo rar. Para la instalación del mismo se han utilizado comandos de descompresión del archivo previa instalación del mismo. Todo esto ha sido posible gracias a la instalación del anteriormente mencionado PCRE. Para facilitar la gestión del software se han creado, además, una serie de directorios y ficheros estáticos para mejorar la funcionalidad del HA-Proxy.

```
mkdir -p /etc/haproxy
mkdir -p /run/haproxy
mkdir -p /var/lib/haproxy
touch /var/lib/haproxy/stats
```

Una vez terminada la instalación, se ha añadido un nuevo usuario para facilitar la gestión del balanceador. Se ha comprobado la correcta instalación del HA-Proxy: véase en la *Captura X* su versión (que es, en efecto, la más actual y estable), así como la fecha y derechos de autor.

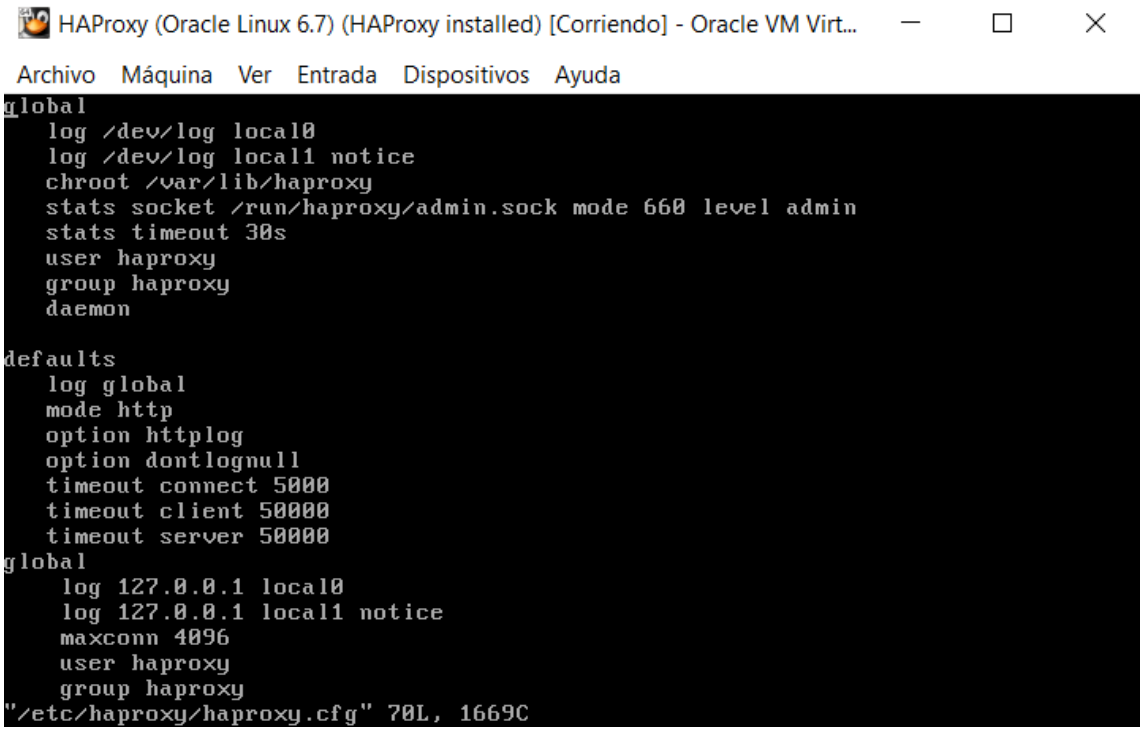


Captura 26. Usuarios del HA-Proxy

17.4.5.2 Configuración del HA-Proxy

La configuración del balanceador ha seguido un proceso lineal con unos objetivos claros explicados en Anexo 17.3 "Análisis y diseño del sistema": determinar a qué tipo de conexiones ha necesitado escuchar el HA-Proxy, así como a los servidores a los que está conectado.

Los datos de configuración se han incluido en un fichero de extensión *.cfg* con la información inicial que se muestra en la siguiente captura.



Captura 27. Ejemplo de configuración del proxy

La posibilidad que ha sido elegida es la de configurar el balanceador para que funcione en la capa 7. Esto resulta útil puesto que este proyecto se encuentra dividido en diferentes hosts. Esto puede lograrse condicionando la transferencia de conexión, en este caso, mediante una dirección URL.

Para la aplicación diseñada, el *frontend* declara una regla ACL llamada url_blog que se aplica a todas las conexiones con rutas que comienzan con / blog. Use_backend define que las conexiones que coincidan con la condición url_blog deben ser servidas por el *backend* llamado blog_back, mientras que todas las demás solicitudes son manejadas por el *backend* predeterminado. Para la configuración relacionada con este proyecto, es necesario recordar el nombre de los servidores:

- Srv1: LDAP
- Srv2: SSO
- Srv3: MDM/IDD
- Srv5: HA-Proxy

```
frontend http_front
   bind *:80
   stats uri /haproxy?stats
   acl url_blog path_beg /blog
```

```
   use_backend blog_back if url_blog
   default_backend http_back
backend http_back
   balance roundrobin
   server <srv2> <private IP>:80 check
   server <srv3> <private IP>:80 check
backend blog_back
   server <srv5> <private IP>:80 check
```

En el *backend*, la configuración configura dos grupos de servidores, http_back como antes y el nuevo llamado blog_back que sirve específicamente conexiones a example.com/blog. En este caso, el blog_back correspondería a una redirección al mismo HAProxy.

Después de realizar cada configuración, ha sido necesario reiniciar el servidor mediante el siguiente comando. De esta forma, una vez reiniciado el sistema, se han quedado los datos guardados y el *secure* proxy ya es capaz de tratar las cabeceras de las peticiones que le lleguen y redireccionarlas.

```
sudo systemctl restart haproxy
```

17.5 Anexo: Fase de pruebas

17.5.1 Test de componentes

Si ha obtenido cualquier error o se ha visto un mal funcionamiento en algún momento, se ha procedido a comprobar la configuración del componente en cuestión.

Además, con el objetivo de testear cada uno de los componentes, se ha ido haciendo uso de comandos desde el *Cygwin* para conectarse directamente a los servidores, de forma independiente, con el objetivo de probar las máquinas por separado antes del test final.

Así pues, p*ara* poder conectarnos remotamente via ssh, ha sido suficiente con introducir un comando y sus correspondientes credenciales (se recuerda que se ha utilizado, de modo genérico el usuario *root* con contraseña 123456 para evitar confusiones).

```
ssh root@maquinaIP
```

Se han realizado testeos a nivel de red (comprobar conexiones con *netstat*, revisar funcionalidad de la red, etc.) hasta testeo a nivel de configuración de componentes.

No se entra en detalle puesto que los componentes se han ido desarrollado paso a paso y no se ha llegado a detectar ningún error que no se pudiera corregir en ese mismo instante. No obstante, si que ha habido alguna observación que comenta en el siguiente apartado.

17.5.2 Posibles errores detectados

Una forma de tratar los errores ha sido recuperar versiones anteriores en las máquinas virtuales, sobre todo para el caso del LDAP. Las instantáneas (*snapshots*) consiguen congelar el estado actual de una MV para poder regresar a él en cualquier otro momento. Son instantáneas basadas en archivos del estado, los datos del disco y la configuración de una MV en un momento

determinado. Existe la posibilidad de tomar varias instantáneas e incluso revertir la MV a uno de los estados anteriores mediante la aplicación de una instantánea.

La misma eficacia y flexibilidad que hacen que estas instantáneas sean útiles en determinados escenarios han provocado consecuencias no deseadas en otros. En este caso, el proyecto se ha visto afectado

- En cuanto a productividad: El hecho de tener múltiples versiones e instantáneas ha hecho que las MV redujeran su rendimiento. Es por esta razón que el escenario ideal es solo durante desarrollo y pruebas, como la utilización de una máquina virtual como servidor de almacenamiento provisional para probar actualizaciones y revisiones antes de implementarlas en los servidores de producción.
- El uso de instantáneas no se recomienda en máquinas virtuales que proporcionen servicios que dependen del tiempo como servicios de LDAP o cuando el almacenamiento o la disponibilidad de espacio de almacenamiento es una cuestión crítica.

18 Especificaciones del sistema

En este apartado se explican aspectos que han sido tenidos en cuenta con respecto a la programación y montaje de la aplicación inicial, así como las librerías que se han utilizado. Esta información se ha sacado de dos puntos importantes:

- En el apartado 4 “Descripción de la situación actual”, se ha llevado a cabo una búsqueda de información para saber cómo están diseñados e implementados los sistemas de la actualidad y se ha hecho una comparativa.
- En el Anexo 17.2 “Estudios con entidad propia” se ha realizado una recopilación de información que ha resultado útil tanto para describir la situación actual como para definir las especificaciones en función a los recursos de la empresa *arhis*.

18.1 Lenguajes de programación

Para la implementación del SSO, se ha utilizado Java, puesto que es un lenguaje muy conocido actualmente y, además, muy utilizado en *arhis*. Como ya se ha mencionado en otros apartados, Java dispone de funcionalidad para trabajar con Oracle, una empresa cuya BD es muy utilizada en *arhis* y bastante famosa a nivel internacional.

En el código se puede entrar en detalle sobre algunos ejemplos de aplicación. En el apartado de definiciones también se entra en detalle sobre el paquete de herramientas java, *Java Development Kit* (JDK). A continuación, se muestra una imagen que resumiría las funcionalidades de este lenguaje.

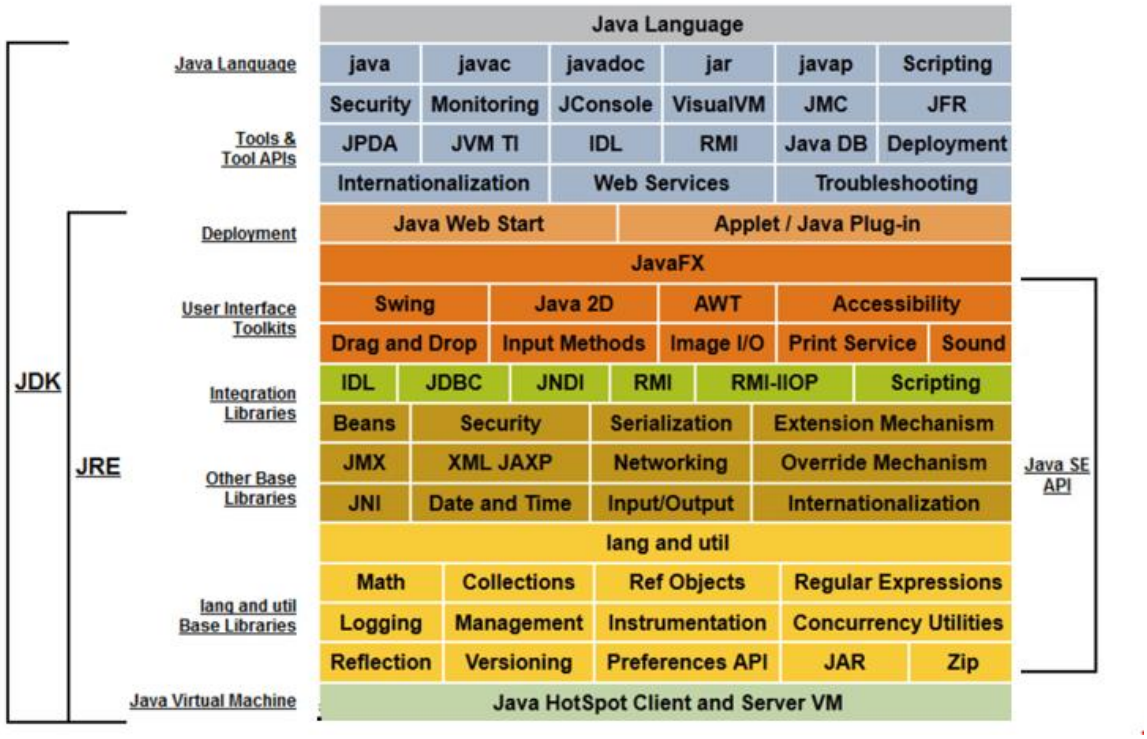


Figura 45. Set de herramientas de Java

18.2 Librerías

Puesto que no ha habido ninguna especificación sobre cómo desarrollar el proyecto, se ha optado, como se ha dicho en el apartado anterior, por utilizar el lenguaje java. Java tiene gran cantidad de funcionalidades que vienen definidas en diferentes librerías. Así pues, es interesante hacer hincapié en este tema, puesto que ha facilitado una conexión entre los componentes que forman el sistema diseñado.

Java EE (*Enterprise Edition*) dispone de una librería de *servlet* que ha sido imprescindible para el diseño e implementación del SSO. Ha permitido crear la página web que demanda las credenciales de los clientes y está conectada directamente al servidor LDAP, mediante el intercambio de *requests* y *responses*. Java también ha permitido modificar el código de implementación de MDM e IDD y ha facilitado la compatibilidad entre los servidores, gracias a una funcionalidad de *login provider* (proveedor de inicio de sesión) facilitada por MDM.

Por último, cabe remarcar el uso de las librerías facilitadas por el SO de Oracle Linux Enterprise, sobre el que se han montado el LDAP, SSO y HA-Proxy. Linux se caracterizan por facilitar un software gratuito y accesible para todo el mundo (modificable). No obstante, al haber previsto desde un inicio el uso de material de Oracle, se han descargado directamente los módulos que han ayudado a unir todo el proyecto.

Como ya se ha comentado, véase el anexo 17.3 y 17.5 para entrar en más detalle sobre el diseño e implementación del sistema.

19 Referencias: Bibliografia

Economides, N., & Katsamakas, E. (2005). Linux vs. Windows: A Comparison of Innovation Incentives and a Case Study.

Martinelli, S., Nash, H., & Topol, B. (2015). *Identity, Authentication, and Access Management in OpenStack: Implementing and Deploying Keystone*. " O'Reilly Media, Inc.".

Maldonado Torres, J. F. (2015). *Implementación de una arquitectura de alta disponibilidad de Servicio de Base de Datos Oracle 11g y sistema operativo Oracle Enterprise Linux 6, aplicado en la Empresa Redpartner* (Bachelor's thesis, PUCE).

Pashalidis, A., & Mitchell, C. (2003). A taxonomy of single sign-on systems. In *Information security and privacy* (pp. 219-219). Springer Berlin/Heidelberg.

Teesloane on GitHub (2017). *Become an Auth Boss. Learn about different authentication methodologies on the web*. Recuperado de https://github.com/teesloane/Auth-Boss.

Kassem, N., & Team, E. (2000). *Designing enterprise applications: Java 2 platform*. Addison-Wesley Longman Publishing Co., Inc..

Dietrich, S. W., Urban, S. D., & Kyriakides, I. (2002, February). JDBC demonstration courseware using Servlets and Java Server Pages. In *ACM SIGCSE Bulletin* (Vol. 34, No. 1, pp. 266-270). ACM.

Reed, P. R. (2001). *Developing Applications with JAVA and UML*. Addison-Wesley Longman Publishing Co., Inc..

Gupta, R., & Kumar, A. (2006). *U.S. Patent No. 7,035,846*. Washington, DC: U.S. Patent and Trademark Office.

Patricio Pérez Pinto on YouTube (2014). *Java web COMPLETO ejercicio Servlet, Jsp y HttpSession*. Recuperado de https://www.youtube.com/watch?v=_1HHUJOfoTY.

Shoaib Khan on YouTube (2012). *Creating a Web Application with Eclipse IDE.* Recuperado de https://www.youtube.com/watch?v=_1HHUJOfoTY.

John Bradley on YouTube (2012). *Designing a Single Sign-on system using OAuth2.0.* Recuperado de http://www.thread-safe.com/2012/02/designing-single-sign-on-system-using.html.

IBM Knowledge Center. *Creating a Web SSO configuration document.* Recuperado de https://www.ibm.com/support/knowledgecenter/en/SSKTMJ_9.0.1/admin/conf_creatingawebssoconfigurationdocument_t.html.

Butcher, M. (2007). Mastering OpenLDAP. *Packt Publishing*.

Limsico, C. T. (2003). *U.S. Patent No. 6,662,228*. Washington, DC: U.S. Patent and Trademark Office.

ORACLE Help Center. *Configuring Single Sign-on with Web Browsers and HTTP clients.* Recuperado*:* https://docs.oracle.com/cd/E21764_01/web.1111/e13707/saml.htm#SECMG252.

Answered by BalusC on *stackoverflow* (2017). *¿What is the difference between JSF, Servlet and JSP?* Recuperado de: https://stackoverflow.com/questions/2095397/what-is-the-difference-between-jsf-servlet-and-jsp?noredirect=1&lq=1.

Ha, Y. G. (2009). Dynamic integration of zigbee home networks into home gateways using OSGI service registry. *IEEE Transactions on Consumer Electronics*, *55*(2).

Saikia, L. P., & Devi, Y. L. (2014). FAULT TOLEREANE TECHNIQUES AND ALGORITHMS IN CLOUD COMPUTING. *International Journal of Computer Science & Communication Networks*, *4*(1), 1.

Answered by Cyril Beschi on *stackoverflow* (2016). *Adding header for HttpURLConnection.* Recuperado: https://stackoverflow.com/questions/2095397/what-is-the-difference-between-jsf-servlet-and-jsp?noredirect=1&lq=1.

Fang, Y., Kao, I. L., Milman, I. M., & Wilson, G. C. (2001). *U.S. Patent No. 6,243,816*. Washington, DC: U.S. Patent and Trademark Office.

Setting up OpenLDAP on CentOS 6: Installing and configuring OpenLDAP Recuperado de: http://docs.adaptivecomputing.com/viewpoint/hpc/Content/topics/1-setup/installSetup/settingUpOpenLDAPOnCentos6.htm.

From ZyTrax, Inc., on *zytrax.open* (2017). *Chapter 5. OpenLDAP Samples: Securing the Directory.* Recuperado de: http://www.zytrax.com/books/ldap/ch5/step2.html#step2.

He, W. (2000). Single sign on. *networks*, *33*, 51-58.

Admin user, on DeepakGaikwad.net (2009). *LDAP Query: Retrive Basic User Attributes in Java.* Recuperado de http://www.deepakgaikwad.net/index.php/2009/09/24/retrieve-basic-user-attributes-from-active-directory-using-ldap-in-java.html.

Cunha, L. D. A. P., Pellizzer, E. P., Verri, F. R., & Pereira, J. A. (2008). Evaluation of the influence of location of osseointegrated implants associated with mandibular removable partial dentures. *Implant dentistry*, *17*(3), 278-287.

Geddes, M., Weaver, F., Jethwa, P., Ginn, C., McConnell, V., & Anderson, D. (2006). *U.S. Patent No. 7,142,840*. Washington, DC: U.S. Patent and Trademark Office.

Bachmann, D. W., Corn, C. F., Fichtner, L. G., Mancisidor, R. A., & Shi, S. B. (2000). *U.S. Patent No. 6,085,188*. Washington, DC: U.S. Patent and Trademark Office

Hoyos, C. A., Sánchez, E. G., Lorenzo, M. L. B., Pérez, J. I. A., Calleja, A. R., & Gorgojo, G. V. Hacia el single sign-on en la integración de herramientas externas en Entornos de Aprendizaje Virtual.

Brown, M. D. (2005). *U.S. Patent Application No. 11/667,738*.

Norman, J. M., Mashayekhi, C., & Ford, K. E. (2008). *U.S. Patent Application No. 12/023,401*.

Lahiri, T., Neimat, M. A., & Folkman, S. (2013). Oracle TimesTen: An In-Memory Database for Enterprise Applications. *IEEE Data Eng. Bull.*, *36*(2), 6-13.

Miller, S. P., Neuman, B. C., Schiller, J. I., & Saltzer, J. H. (1987). Kerberos authentication and authorization system. In *In Project Athena Technical Plan*.

Neuman, B. C., & Ts'o, T. (1994). Kerberos: An authentication service for computer networks. *IEEE Communications magazine*, *32*(9), 33-38.

Kent, S. (2005). IP authentication header.

Diffie, W., & Hellman, M. E. (1979). Privacy and authentication: An introduction to cryptography. *Proceedings of the IEEE*, *67*(3), 397-427.

Bucholtz, M. (2003). Sociolinguistic nostalgia and the authentication of identity. *Journal of sociolinguistics*, *7*(3).

Maes, S. H., & Sedivy, J. (2000). *U.S. Patent No. 6,016,476*. Washington, DC: U.S. Patent and Trademark Office.

Griffiths, P. P., & Wade, B. W. (1976). An authorization mechanism for a relational database system. *ACM Transactions on Database Systems (TODS)*, *1*(3), 242-255.

Buhle, G., & Wessman, R. R. (2001). *U.S. Patent No. 6,286,104*. Washington, DC: U.S. Patent and Trademark Office.

Ng, R. K. (2008). *U.S. Patent No. 7,404,203*. Washington, DC: U.S. Patent and Trademark Office.

Geddes, M., Weaver, F., Jethwa, P., Ginn, C., McConnell, V., & Anderson, D. (2006). *U.S. Patent No. 7,043,230*. Washington, DC: U.S. Patent and Trademark Office.