**ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN**

PROYECTO FIN DE GRADO

### TÍTULO: Análisis y procesamiento de información textual utilizando generadores de compiladores

**AUTOR: Miguel Amarís Martos**

**TITULACIÓN: Grado en Ingeniería Telemática**

**TUTOR: José Luis López Presa**

**DEPARTAMENTO: Ingeniería Telemática y Electrónica**

VºBº

**Miembros del Tribunal Calificador:**

**PRESIDENTE:**

**TUTOR: José Luis López Presa**

**SECRETARIO: Javier Malagón Hernández Fecha de lectura:**

**Calificación:**

**El Secretario,**

**AGRADECIMIENTOS**

A mi madre, mi tía y mi abuela por habérmelo dado todo sin pedir nada a cambio. A mi padre, por enseñarme que las cosas no son siempre como nos las cuentan.

Al pollo, por darnos cuidados y alegría. A mis amigos y amigas del pueblo, por crecer a mi lado.

A mis amigos y amigas de Madrid, por ser mi segunda familia.

A todos los compañeros y compañeras de la carrera, por enseñarme el valor del trabajo

colectivo. A M. Azorí, por haber sido el mejor compañero de trabajo.

A José Luis y a César, por su paciencia, su tiempo y su ayuda inestimables.

A todos y todas las que lucharon para defender las libertades humanas a lo largo de la historia.

**RESUMEN**

**Análisis y procesamiento de información textual utilizando generadores de compiladores**

En el panorama actual, el procesamiento de información en diversos formatos se ha convertido en una práctica común. Para abordar esta tarea, se emplean diferentes herramientas diseñadas específicamente para analizar datos en diversos lenguajes de programación o formatos como JSON o XML. Sin embargo, la necesidad de una solución genérica que permita el análisis de una amplia variedad de gramáticas se ha vuelto evidente.

En este contexto, se han desarrollado generadores de reconocedores gramaticales, como *Java Compiler Compiler* (JavaCC). JavaCC es una herramienta que permite la generación de analizadores léxicos y sintácticos a partir de una gramática definida por el usuario. Su versatilidad y capacidad para adaptarse a diversas gramáticas lo convierten en una elección atractiva para el procesamiento de información estructurada.

Este proyecto se enfoca en explorar el potencial de JavaCC y su aplicación en la asignatura Procesamiento de la Información en Aplicaciones Telemáticas (PIAT). El objetivo principal es aprender a utilizar JavaCC, aplicarlo en prácticas de PIAT y crear una documentación que respalde su versatilidad para el análisis de gramáticas y el procesamiento de información.

***ABSTRACT***

***Alternative for the IT infrastructure of DTE laboratories based on free software***

*Over the last two decades, information and communication technologies have come to play a central role in our lives, not only personal but also professional and educational ambits.*

*Exponential developments in information storage and transmission systems have reduced drastically costs associated with these processes, that is storage and transmission of information, and thus to create a global knowledge network that a large part of the world can access for consultation and/or contribution.*

*However, due to the monopolising dynamics of the global economic system, although information wants to be free, it has become a market good, through legislation protecting (private) intellectual property.*

*In this sense, we find that a large part of the computer systems used today tend to depend on the proprietary software of large companies such as Microsoft, Apple, Google or Amazon. Their hegemony is not necessarily linked to the quality and security of their products, but rather to their great capacity for marketing, reaching agreements with public and private institutions, and, ultimately, their approach to the end user, characterised by abstracting as much as possible from the real functioning of the systems on which they run, reducing the learning curve to a minimum in many cases.*

*On the other side of the scale, and with a completely opposite philosophy, we find Free Software, with the full range of GNU/Linux systems as its maximum exponent. In this other "world", were, by the way, the participation of private institutions is not excluded, the golden rule is "[...] that users have the freedom to run, copy, distribute, study, modify and improve the software". In this sense, FOSS makes possible to exploit to the maximum the current characteristics of ICT mentioned above, allowing the creation, development and improvement of software in a transparent, collaborative and easily accessible way for anyone, regardless of their economic resources.*

*Taking this context into account, the aim of this project is to demonstrate that it is totally feasible to migrate the computer systems in a university environment (the DTE laboratories of the ETSIST, in this case) from proprietary software to free software, that such a change would not only not affect the workload of technical and teaching staff, as well as student learning, but could also, on the one hand, facilitate the management tasks of the systems and, on the other, allow students to improve their knowledge of the operation of computer systems, as well as to display greater levels of creativity.*

*In addition to all this, the use of free software instead of proprietary software would allow the educational institution to substantially alleviate its budget expendienture currenly spent on license fees.*

# ÍNDICE

[ÍNDICE 5](#_Toc152607696)

[1. INTRODUCCIÓN 12](#_Toc152607697)

[1.1 Motivación 12](#_Toc152607698)

[1.2 Objetivos 12](#_Toc152607699)

[1.3 Restricciones 12](#_Toc152607700)

[2. MARCO TECNOLÓGICO 13](#_Toc152607701)

[2.1 Introducción 13](#_Toc152607702)

[2.2 JavaCC 14](#_Toc152607703)

[2.3 Funciones principales de JavaCC 14](#_Toc152607704)

[2.4 Ejemplo de gramática en JavaCC 16](#_Toc152607705)

[2.5 Analizador Léxico 25](#_Toc152607706)

[2.6 Estados Léxicos 26](#_Toc152607707)

[2.7 Analizador Sintáctico o Semántico 32](#_Toc152607708)

[2.8 Recursividad 35](#_Toc152607709)

[3. DESARROLLO DE LA PROPUESTA 37](#_Toc152607710)

[3.1 Introducción 37](#_Toc152607711)

[3.2 Procesamiento de la Información en Aplicaciones Telemáticas (PIAT) 37](#_Toc152607712)

[3.3 Entorno de escritorio utilizado 40](#_Toc152607713)

[3.4 Herramienta de gestión y automatización utilizada 43](#_Toc152607714)

[3.5 Software actualmente instalado en los laboratorios DTE 44](#_Toc152607715)

[3.6 Análisis de archivos XML. Práctica 3 46](#_Toc152607716)

[3.7 Análisis de archivos JSON. Práctica 4 47](#_Toc152607717)

[3.8 Configuración del servidor 47](#_Toc152607718)

[3.9 Herramientas propias desarrolladas 48](#_Toc152607719)

[3.10 Entorno de desarrollo y pruebas 53](#_Toc152607720)

[4. PRUEBAS DE DESPLIEGUE 56](#_Toc152607721)

[5. RESULTADOS 58](#_Toc152607722)

[5.1 Tiempos de instalación 58](#_Toc152607723)

[5.2 Comparativa Windows y Kubuntu 61](#_Toc152607724)

[6. MANUALES 63](#_Toc152607725)

[6.1 Instalación de JavaCC 63](#_Toc152607726)

[6.2 Instalación de los clientes 69](#_Toc152607727)

[6.3 Mantenimiento de los clientes 84](#_Toc152607728)

[7. PRESUPUESTO 85](#_Toc152607729)

[8. CONCLUSIONES 85](#_Toc152607730)

[9. LÍNEAS FUTURAS 86](#_Toc152607731)

[10. BIBLIOGRAFÍA 87](#_Toc152607732)

[11. REFERENCIAS 87](#_Toc152607733)

[12. ANEXO 87](#_Toc152607734)

[SÍMBOLOS DE EXPRESIONES REGULARES EN JAVACC 87](#_Toc152607735)

[13. Preguntas Frecuentes 89](#_Toc152607736)

**ÍNDICE DE FIGURAS**

[Figura 1. Funcionamiento del Analizador Léxico 26](#_Toc151320503)

[Figura 2. Funcionamiento de un Analizador Sintáctico 33](#_Toc151320504)

[Figura 3. Compilación de Archivos JavaCC en Eclipse 65](#_Toc151320505)

**ÍNDICE DE TABLAS**

[Tabla 1. Herramientas de aprovisionamiento 45](#_bookmark47)

[Tabla 2.Comparativa de distribuciones GNU/Linux 50](#_bookmark59)

[Tabla 3. Comparativa de sabores de Ubuntu 53](#_bookmark62)

[Tabla 4. Comparativa de consumo de memoria RAM 53](#_bookmark63)

[Tabla 5. Comparativas herramientas de automatización 55](#_bookmark65)

[Tabla 6. SW de los laboratorios DTE disponible para Ubuntu 58](#_bookmark69)

[Tabla 7. SW de los laboratorios DTE parcialmente disponible para Ubuntu 60](#_bookmark71)

[Tabla 8. SW de los laboratorios DTE no disponible para Ubuntu 61](#_bookmark73)

[Tabla 9. Comparativa básica Windows y Kubuntu en sistema virgen 85](#_bookmark116)

[Tabla 10. Comparativa básica Windows y Kubuntu en sistema final 85](#_bookmark117)

[Tabla 11. Desglose presupuestario 121](#_bookmark204)

[Tabla 12. Coste de la mano de obra 121](#_bookmark205)

**LISTA DE ACRÓNIMOS**

IDE – Integrated Development Environment (Entorno de Desarrollo Integrado)

Red Hat Enterprise Linux SLES – Suse Linux Enterprise Server IDE – Entorno de desarrollo integrado CLI – Interfaz de línea de comandos GUI – Interfaz gráfica de usuario

Cuando se represente la CLI de GNU/Linux como texto, el comando comenzará con “$” si es de ejecución estándar, o con “#”, si se debe ejecutar con privilegios de administrador.

MV – Máquina virtual

CPU – Unidad central de procesamiento, equivalente a microprocesador LAN – Red de área local

# INTRODUCCIÓN

## Motivación

Desde la infancia desarrollé un interés temprano por la tecnología informática. Con el paso de los años, tras un lento aprendizaje autodidacta, mi interés se fue centrando hacia todo lo que representaba la otredad en este mundo tecnológico en constante ascenso, todo lo que escapaba a lo oficial y extensamente conocido, desde el hacking de videoconsolas y redes inalámbricas a los sistemas operativos ajenos a lo que se consideraba normal, es decir, Microsoft Windows.

Mis estudios posteriores en Ingeniería Telemática me sirvieron para profundizar en el conocimiento e interés respecto a estos asuntos, especialmente en lo referido a la arquitectura y seguridad de las redes de telecomunicación, el desarrollo de software y el funcionamiento de los sistemas operativos.

En este sentido, comencé mis prácticas curriculares en 2018, en un puesto que combinaba tareas de administrador de sistemas GNU/Linux, *DevOps* y, en menor medida, gestión de redes. Tras unos meses, finalicé las prácticas y fui contratado por la misma empresa, donde permanecí hasta hace poco tiempo, y dónde continué afianzando y ampliando mis conocimientos.

Teniendo en cuenta este contexto, el presente proyecto persigue aplicar los conocimientos y metodologías que he adquirido en mi vida personal, académica y laboral en materia de administración de sistemas, GNU/Linux, software libre y automatización de tareas a un entorno concreto dentro de la universidad, como los laboratorios del departamento DTE de la ETSIST. Este proyecto tratará de demostrar de la mejor manera posible que, aunque a día de hoy la infraestructura informática de la escuela, de la que hacen uso alumnos, profesorado y personal técnico, funcione principalmente con software privativo de Microsoft y otras compañías, sería perfectamente viable que lo hiciera mediante GNU/Linux y software libre-abierto, y que esto, además, supondría una serie de ventajas a todos los niveles.

## Objetivos

El objetivo principal del proyecto es evaluar la viabilidad de utilizar una herramienta de generadores genéricos como JavaCC para abordar las prácticas de PIAT.

Esto implica:

* Aprender a utilizar JavaCC de manera efectiva.
* Aplicar JavaCC en prácticas específicas de PIAT.
* Generar documentación que sirva como recurso para estudiantes y profesores interesados en utilizar JavaCC en proyectos relacionados con el procesamiento de información.

El proyecto busca proporcionar una solución versátil y eficaz para el análisis de gramáticas y el procesamiento de información en el ámbito de PIAT

## Restricciones

En la realización del proyecto, debemos tener en cuenta las siguientes restricciones:

* El proyecto se centrará en la utilización de JavaCC como herramienta principal.
* Debe garantizarse que la documentación generada sea comprensible y útil para aquellos que deseen aplicar JavaCC en proyectos similares.
* Las prácticas de PIAT deben servir como un contexto relevante para la aplicación de JavaCC.

# MARCO TECNOLÓGICO

## Introducción

El procesamiento de información en diversos formatos es una tarea cada vez más importante. En un mundo cada vez más digital, es necesario poder procesar datos en una amplia variedad de formatos, desde lenguajes de programación hasta formatos de documentos y datos estructurados.

En este contexto, los generadores de reconocedores gramaticales, como JavaCC, ofrecen una solución muy prometedora. Estos generadores permiten crear analizadores léxicos y sintácticos a partir de una gramática definida por el usuario. Esto los hace muy versátiles y adaptables a una amplia variedad de gramáticas.

La aplicación de JavaCC en la asignatura Procesamiento de la Información en Aplicaciones Telemáticas (PIAT) es una idea muy acertada. PIAT es una asignatura que se centra en el procesamiento de información en diversos formatos. El uso de JavaCC podría proporcionar a los estudiantes una herramienta muy valiosa para abordar las prácticas de la asignatura.

Desde que apareció la informática tal y como la conocemos, allá por la década de 1960 [1], las metodologías que se han seguido para el desarrollo del SW, así como las restricciones impuestas para su acceso han ido variando.

Hasta finales de la década de 1970, el SW no era considerado como un producto con entidad propia, sino simplemente como el complemento necesario para que los sistemas HW de la época (*mainframes*) [2] fueran funcionales [3]. En este contexto, los desarrolladores de SW solían compartir libremente sus soluciones de forma colaborativa, tanto en entornos universitarios como empresariales, sin ningún tipo de restricción de por medio.

Este paradigma cambió en la década de 1980, momento en el que se empezó a dar al SW y a los SO valor como entidades/productos en sí mismos, convirtiéndose en privados y restringiendo el acceso y modificación de código fuente por parte de las empresas que poseían la propiedad sobre dichos programas. [3]

En este contexto de incipiente privatización de la información y el conocimiento, una parte de la comunidad de desarrolladores de la época comenzó a trabajar para materializar una alternativa que permitiera el desarrollo libre, colaborativo y sin barreras, con el papel destacado de Richard Matthew Stallman.

Stallman, convencido de la urgente necesidad de crear alternativas de software libre, y con la premisa de que la pieza fundamental para ejecutar software es un sistema operativo, hizo público en 1983 el Proyecto GNU, tardando dos años más en concretar todos los detalles a través del Manifiesto GNU:

“*GNU, que significa «Gnu No es Unix», es el nombre del sistema de software completamente compatible con Unix que estoy escribiendo para entregarlo libremente a todas las personas que puedan utilizarlo. Algunos voluntarios me están ayudando. Las aportaciones de tiempo, dinero, programas y equipos son muy necesarias.”* [4]

Después, en el año 1985, Stellman y el resto de programadores involucrados en el proyecto, fundaron la *Free Software Foundation* con el objetivo de obtener los fondos y medios necesarios para el desarrollo de GNU. Desde entonces, la FSF ha seguido evolucionando, convirtiéndose en una organización sin ánimo de lucro que se encarga de fomentar y estandarizar el software libre, además de ofrecer soporte jurídico para su desarrollo y protección. En este sentido, la FSF es el garante de diferentes licencias de software libre, conocidas como licencias *copyleft*, en oposición al *copyright* privativo. La más extendida de estas licencias es la GNU GPL (*General Public License)*[5].

## JavaCC

En relación con lo anterior, aclaremos a qué hace referencia el concepto de *JavaCC* y los analizadores semánticos y sintácticos, y como se interrelacionan:

*Java Compiler Compiler*, comúnmente conocido como JavaCC, es una poderosa herramienta ampliamente utilizada para generar analizadores sintácticos en aplicaciones Java. Su función principal es transformar una especificación gramatical en un programa Java capaz de reconocer y analizar la sintaxis según la gramática proporcionada.

JavaCC se diferencia de otras herramientas similares al generar analizadores sintácticos de arriba hacia abajo (descenso recursivo) en lugar de analizadores de abajo hacia arriba. Esta característica permite el uso de gramáticas más generales y facilita la depuración, además de permitir el análisis en cualquier elemento no terminal de la gramática y la transferencia de valores (atributos) en ambas direcciones en el árbol de análisis.

Por otro lado, como comentábamos en el punto anterior, surge el Proyecto GNU como intento de crear un sistema operativo similar a Unix y totalmente compatible con él, pero de carácter libre y abierto. Es importante mencionar que GNU no hace referencia exclusivamente al sistema operativo, sino también a todo el conjunto de aplicaciones y herramientas que corren sobre él.

En el año 1990, ya se había conseguido desarrollar la mayoría de componentes del sistema GNU: intérprete de comandos, editor de textos, compilador de C, etc. Sin embargo, se estaba retrasando el desarrollo del componente fundamental: el núcleo o *kernel*. Es aquí donde entra en juego Linus Torvalds [7], un ingeniero de software finlandés-estadounidense. Torvalds había estado trabajando en un kernel compatible con sistemas Unix. En 1991 liberó la primera versión de este núcleo, con el nombre de Linux, convirtiéndolo en software libre al año siguiente.

Rápidamente se comenzó a realizar un esfuerzo cooperativo para hacer completamente funcional la combinación de GNU y Linux, que dio lugar al sistema operativo completo GNU/Linux, del que, a día de hoy, existen cientos de distribuciones destinadas a cubrir un gran abanico de casos de uso, desde ordenadores personales hasta granjas de servidores. Miles de personas, ya sea de forma individual o a través de instituciones públicas y privadas, forman la comunidad GNU/Linux que mantiene y mejora constantemente el sistema operativo y sus aplicaciones, de forma colaborativa y siguiendo los principios del software libre o, al menos, del software de código abierto. En el siguiente punto aclaramos esta cuestión.

## Funciones principales de JavaCC

JavaCC ofrece una serie de características y funcionalidades clave que lo hacen destacar como un generador de analizadores sintácticos:

* Generación de analizadores sintácticos de arriba hacia abajo.
* Resolución de ambigüedades de cambio de turno localmente.
* Generación de analizadores 100% Java puro.
* Especificaciones BNF extendidas.
* Integración de especificaciones léxicas y gramaticales en un solo archivo.
* Manejo de la entrada Unicode completa.
* Soporte para tokens no distinguibles entre mayúsculas y minúsculas.
* Herramientas adicionales como JJTree para construir árboles y JJDoc para generar documentación.
* Personalización a través de numerosas opciones.
* Informes de errores de alta calidad y mensajes de diagnóstico completos.

### ¿Cómo leer de un fichero?

Dependiendo de si pones la opción moderna ()

options

{

JAVA\_TEMPLATE\_TYPE = "modern";

}

, se podrá leer de dos diferentes maneras:

* Sin JAVA\_TEMPLATE\_TYPE = "modern" →

XMLParser parser = new XMLParser(new XMLParserTokenManager(new SimpleCharStream(new InputStreamReader(new FileInputStream(args[0]), "UTF-8"))));

* Añadiendo JAVA\_TEMPLATE\_TYPE = "modern" →

XMLParser parser = new XMLParser(new XMLParserTokenManager(new SimpleCharStream(new StreamProvider(new FileInputStream(args[0]), "UTF-8"))));

They represent somewhat different things.

The InputStream is the ancestor class of all possible streams of bytes, it is not useful by itself but all the subclasses (like the FileInputStream that you are using) are great to deal with binary data.

On the other hand, the InputStreamReader (and its father Reader) are used specifically to deal with characters (so strings) so they handle charset encodings (utf8, iso-8859-1, and so on) gracefully.

The simple answer is: if you need binary data you can use an InputStream (also a specific one like a DataInputStream), if you need to work with text use an InputStreamReader.

### ¿Como funciona la coincidencia de tokens?

Todas las expresiones regulares en el estado léxico actual se consideran posibles candidatas a coincidir. El administrador de tokens consume la cantidad máxima posible de caracteres del flujo de entrada que coincidan con una de estas expresiones regulares. Es decir, el administrador del token prefiere la coincidencia más larga posible. Si hay varias coincidencias más largas de la misma longitud, la expresión regular que coincide es la que tiene el primer orden de aparición en el archivo de gramática.

Como hemos comentado en apartados anteriores, el Software libre surge de la necesidad de plantear una alternativa frente a la incipiente privatización del SW en la década de 1980, con el proyecto GNU como mayor exponente.

Según el propio proyecto GNU, el Software libre es aquel que cuenta con las siguientes características:

*“«Software libre» es el software que respeta la libertad de los usuarios y la comunidad. A grandes rasgos, significa que* ***los usuarios tienen la libertad de ejecutar, copiar, distribuir, estudiar, modificar y mejorar el software****. Es decir, el «software libre» es una cuestión de libertad, no de precio.”* [9]

Por otro lado, el concepto de Software de código abierto o *Open source* surge en 1998 como una escisión de la comunidad del Software libre. La motivación principal de este sector es la de hacer más atractivo el software libre y sus metodologías a la inversión privada, dejando a un lado los aspectos éticos y haciendo hincapié en los beneficios técnico-prácticos: permitiendo la consulta y mejora de los códigos fuente por parte de la comunidad se pueden lograr productos software de mayor calidad que aquellos que se desarrollan y mantienen de manera privativa [10].

En resumen, ambos conceptos hacen referencia a prácticamente los mismos programas, pero desde puntos de vista diferentes: mientras uno hace hincapié en los valores éticos, el otro lo hace en los beneficios técnicos. Es por esto que la FSF ha establecido unos estándares “oficiales” por los cuales un SW se puede considerar, o no, libre. Todo SW libre os abierto, pero no todo SW abierto es libre.

En el caso concreto del kernel Linux oficial, aunque en su mayoría es código libre, contiene algunos módulos firmware para la comunicación con periféricos que son código privativo. En este sentido, no se puede considerar SW libre como tal. Como alternativa 100% libre, desde el 2008, la Fundación Software Libre América Latina desarrolla y mantiene *Linux-libre*, una modificación del núcleo Linux oficial al que se le ha extraído todo el software cuyo código fuente no es público, está ofuscado o simplemente no posee una licencia de uso libre [11].

Si bien Linux-libre cumple con todos los estándares de SW libre promovidos por la FSF y esta lo recomienda oficialmente, su implementación es prácticamente marginal en comparación al núcleo Linux oficial, principalmente porque al carecer de los módulos privativos, su incompatibilidad hardware crece de manera significativa y, a diferencia de Linux oficial, no cuenta con ningún tipo de apoyo por parte de las grandes empresas tecnológicas.

Todo este panorama nos conduce a la pregunta: ¿hasta qué punto es viable implementar soluciones basadas en software 100% libre? Según la propia FSF: “*No hay límites, excepto cuando leyes como el sistema de patentes prohíben el software libre. El objetivo final es el de proporcionar software libre para realizar todas las tareas que los usuarios de ordenadores quieran llevar a cabo, y lograr así que el software privativo sea cosa del pasado.*”

Si bien podemos asumir como correcta esta respuesta en un contexto ideal, las restricciones de la realidad nos imponen algunos límites, de los que hablaremos en el siguiente capítulo de la presente memoria.

## Ejemplo de gramática en JavaCC

A continuación, se presenta un ejemplo de cómo JavaCC puede utilizarse para generar un analizador sintáctico en Java. En este ejemplo se define un lenguaje para analizar expresiones matemáticas simples. Partiendo de este ejemplo, seremos capaz de ampliar las funcionalidades del programa, y realizar una calculadora:

// Define la gramática para analizar una simple expresión matemática.

**void** **Expression**() :

{}

{

Term() ( "+" Term() | "-" Term() )\*

}

**void** **Term**() :

{}

{

Factor() ( "\*" Factor() | "/" Factor() )\*

}

**void** **Factor**() :

{}

{

<NUMBER>

| "(" Expression() ")"

}

**Función Expression()**

La función *Expression()* define la estructura general de una expresión. Una expresión consiste en un término seguido de cero o más términos conectados por operadores aritméticos. La función *Expression()* comienza con la función *Term()*. Esto significa que la primera parte de cualquier expresión debe ser un término. A continuación, la función *Expression()* utiliza la regla *)\** para especificar que puede seguir cero o más términos. Esto significa que las expresiones pueden tener cualquier longitud, desde una sola palabra hasta una expresión compleja con muchos términos.

Los términos están conectados por operadores aritméticos. La regla *Expression()* utiliza la regla *)\** para especificar que puede seguir cualquier número de operadores aritméticos. Esto significa que las expresiones pueden tener cualquier cantidad de operadores aritméticos, desde ninguno hasta muchos. Para más información acerca de operadores en JavaCC, vaya al anexo **Operadores en JavaCC.**

Los operadores aritméticos permitidos son la suma (*+*), la resta (*-*), la multiplicación (*\**) y la división (*/*).

**Función Term()**

La regla *Term()* define la estructura de un término. Un término consiste en un factor seguido de cero o más factores conectados por operadores aritméticos. La regla *Term*() comienza con la regla Factor(). Esto significa que la primera parte de cualquier término debe ser un factor. A continuación, la regla *Term*() utiliza la regla )\* para especificar que puede seguir cero o más factores. Esto significa que los términos pueden tener cualquier longitud, desde una sola palabra hasta un término complejo con muchos factores.

Los factores están conectados por operadores aritméticos. La regla *Term*() utiliza la regla )\* para especificar que puede seguir cualquier número de operadores aritméticos. Esto significa que los términos pueden tener cualquier cantidad de operadores aritméticos, desde ninguno hasta muchos.

Los operadores aritméticos permitidos son la suma (+), la resta (-), la multiplicación (\*) y la división (/)

**Función Factor()**

La regla *Factor()* define la estructura de un factor. Un factor puede ser un número o una expresión entre paréntesis. La regla Factor() tiene dos opciones. La primera opción es un número. La segunda opción es una expresión entre paréntesis. Si la opción elegida es un número, la regla Factor() utiliza la regla <NUMBER> para especificar que el factor debe ser un número. Si la opción elegida es una expresión entre paréntesis, la regla Factor() utiliza la regla "(" Expression() ")" para especificar que la expresión debe estar entre paréntesis.

**Ejemplos**

Aquí hay algunos ejemplos de expresiones que pueden ser analizadas por esta gramática:

* 1 + 2
* 3 \* 4
* (5 - 6) / 7

Aunque también se puede analizar expresiones más complejas, como:

* (1 + 2) \* (3 - 4)
* (5 \* 6) / (7 + 8)

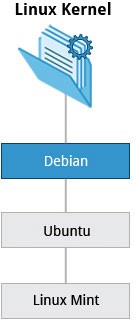
El concepto distribución GNU/Linux, comúnmente, *distro*, hace referencia a la suma de un kernel Linux compilado, las aplicaciones básicas de sistema del proyecto GNU, y a otras aplicaciones añadidas en función de los casos de uso para los que se conciba dicha distribución. Por lo general, este conjunto se entrega al usuario final compilado y parcialmente configurado en forma de imagen ISO, para poder desplegarse en Hardware *real* o en máquinas virtuales.

Antes de que estas surgieran, los usuarios que quisieran ejecutar un sistema GNU/Linux debían hacer manualmente el proceso de compilado de kernel, adición de las herramientas necesarias, etc.

Vamos a pasar a detallar brevemente las principales ramas de desarrollo que han dado lugar a las familias de distribuciones que más se implementan a día de hoy:

### Familia Debian

Dentro de la familia Debian existen cientos de distribuciones basadas directamente en Debian, o basadas en otras “hijas”. Sin embargo, nos centraremos en las más importantes: el propio Debian y Ubuntu.



*Figura 5. Esquema general de la familia Debian [12]*

### Debian

El proyecto Debian fue lanzado al público el 16 de agosto de 1993 por Ian Murdock, acompañado por otros programadores seguidores de la filosofía del Software libre. Debian surge en un momento en que las distribuciones GNU/Linux todavía no estaban extendidas, al ser su construcción y mantenimiento un trabajo complicado y que, en muchos casos, no llamaba demasiado la atención de los desarrolladores. En este sentido, el Manifiesto Debian recoge: “Las distribuciones son esenciales para el futuro de Linux. En esencia, le ahorran al usuario la necesidad de buscar, obtener, compilar, instalar e integrar correctamente un gran número de herramientas esenciales para conseguir un sistema Linux en funcionamiento. (…) A pesar de su obvia importancia, las distribuciones han atraído poco la atención de los desarrolladores. Existe una sencilla razón para ello: no son ni

fáciles ni fascinantes de construir, y requieren gran cantidad de esfuerzo continuado por parte de su creador con el fin de mantener la distribución libre de errores y además actualizada.

(…) El diseño de Debian es lo bastante abierto para asegurar que el sistema tiene la más alta calidad y que refleja las necesidades de la comunidad de usuarios. Al implicar a otras personas de diversas capacidades y bagajes, Debian puede desarrollarse de forma modular. (…) Implicar a otros asegura además que a la distribución pueden incorporarse valiosas contribuciones durante su desarrollo; de esta manera, se crea una distribución basada en las necesidades y deseos de los usuarios, en vez de las necesidades y deseos del constructor.” [13]

Siguiendo esta idea, desde el 1993 hasta nuestros días, han sido liberadas 13 versiones oficiales y estables de Debian, siendo la última “Debian 11.0 bullseye” liberada el 14 de agosto de 2021.

La característica más reseñable de esta distribución y las que están basadas en ella, es el uso del sistema de paquetes deb para instalar todo el software del sistema, con la interfaz aptitude [14] y derivados, para facilitar la instalación, desinstalación y actualización de paquetes (ya sea individualmente, en grupos o el sistema completo) resolviendo todas las dependencias y garantizando la estabilidad del sistema.

Debian está considerada como una de la distribuciones más estables y seguras, equilibrada a su vez con métodos de instalación y configuración relativamente sencillos. Cuenta con una enorme colección de software disponible y una de las mayores comunidades de desarrollo actualmente activas. Aunque no está oficialmente patrocinada por la FSF, por cuestiones concretas relacionadas con lo expuesto en el punto 2.3 de esta memoria, el proyecto Debian está comprometido activamente con la filosofía de software libre [15], y detrás de su desarrollo no participa ninguna entidad privada con ánimo de lucro, si no que sus miles de desarrolladores se asocian libremente, formando la estructura organizativa del proyecto que podríamos calificar de democrática [16].



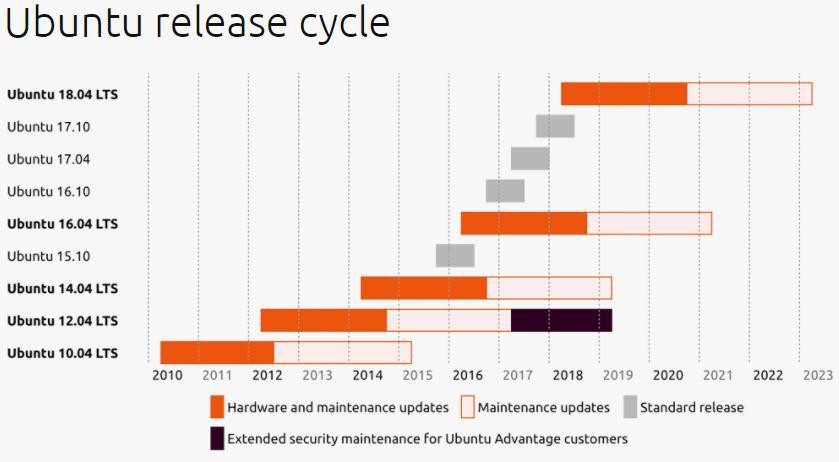
### Ubuntu

Como ya hemos comentado, múltiples distribuciones han surgido tomando a Debian como base. Sin lugar a dudas, la más popular y extendida de todas es Ubuntu.

Ubuntu fue lanzado al público en el año 2004 por parte de la empresa Canonical [18], con el sudafricano Mark Shuttleworth a la cabeza. El objetivo principal que persigue Ubuntu es el de ser una distribución GNU/Linux lo más amigable posible para el usuario estándar, que no tiene por qué poseer conocimientos informáticos avanzados y que provienen de otros sistemas privativos, generalmente Windows de Microsoft, aunque en los últimos años Canonical también está invirtiendo grandes cantidades de recursos para el desarrollo de productos orientados a servidores y *cloud*. En este sentido, Ubuntu ha conseguido afianzarse como una de las distribuciones que año tras año aparecen en el *ranking* de las más usadas [19], además de servir de base para otras distribuciones muy populares a día de hoy, como Linux Mint [20], que siguen una filosofía similar.

Ubuntu, al partir de Debian, hereda la paquetería “deb”, aunque cuenta con sus propios repositorios y otro tipo de paquetes propios conocidos como **Snap**. La popularidad de Ubuntu estriba en, además de la robustez y estabilidad propias de la familia Debian, en su instalador, gráfico e intuitivo, la instalación y configuración por defecto de una gran cantidad de SW útil en el día a día del usuario medio. También pone las cosas bastante fáciles para elegir entre los diferentes entornos de escritorio para GNU/Linux existentes, con imágenes pre-configuradas conocidas como **sabores** o **flavours** [21].

En cuanto al ciclo de actualizaciones, en Ubuntu está claramente definido y se cumple con exactitud. Se liberan versiones cada 6 meses, en abril y octubre. Las versiones de abril de los años pares se conocen como LTS (*Long Time Service*) con un soporte garantizado de hasta 5 años. El resto de liberaciones se consideran versiones “intermedias”, que cuentan con SW más avanzado, pero que son menos estables y el soporte oficial para ellas es de 9 meses. Son versiones enfocadas a experimentar y mejorar los problemas que van ocurriendo de cara a la siguiente versión LTS.



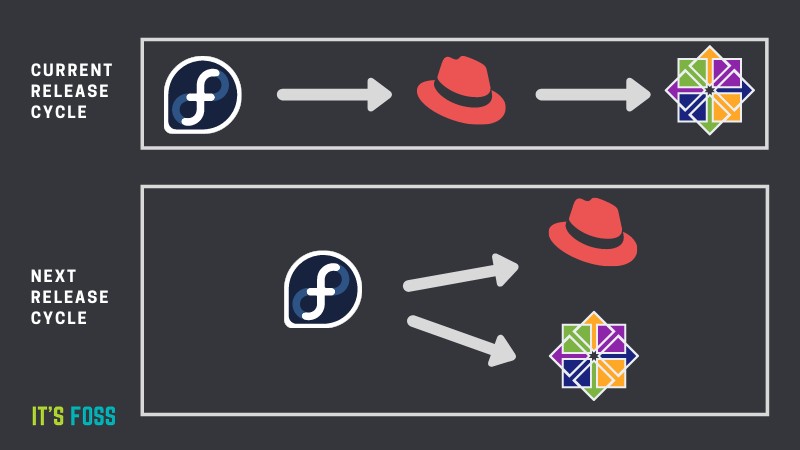
*Figura 7. Diagrama de liberación de versiones de Ubuntu [22]*

Por último, Ubuntu nos da la opción, de forma sencilla, de instalar drivers y otro software privativo que permiten sacar el máximo partido a determinados periféricos o poner las cosas fáciles a la hora de realizar ciertas actividades, como la reproducción o producción de contenido multimedia.

Este hecho, unido al de que la responsable del proyecto sea una empresa privada con ánimo de lucro, alejan a Ubuntu de la “pureza” del software libre promulgada por la FSF. Pese a que Ubuntu se distribuye bajo licencia libre GNU GPL y que da la posibilidad de contribuir al proyecto a desarrolladores ajenos a Canonical, nos encontramos con un caso claro en el que se hace hincapié en la usabilidad y el resultado final por encima de las consideraciones éticas que comentamos en puntos anteriores. De nuevo nos encontramos con la paradoja entre lo ideal y los límites de lo real.  


### Familia Red Hat

Después de la familia Debian, la que más popularidad y consolidación ha adquirido con el paso del tiempo es la familia Red Hat, caracterizada principalmente por el uso del gestor de paquetes RPM (Red Hat Package Manager) [24] y el instalador Anaconda [25].



*Figura 9. Esquema de la familia Red Hat antes y después de los cambios anunciados en 2019 [26]*

Esta familia de distribuciones es desarrollada y mantenida por la empresa estadounidense Red Hat, fundada en 1993 y adquirida recientemente por IBM. El objetivo principal de Red Hat es el de ofrecer soluciones Open Source altamente fiables para entornos empresariales contando, además de sus distribuciones, con una amplia gama de productos relacionados con la automatización de infraestructura, computación en la nube, virtualización, contenedores, etc. [27]

A diferencia de la familia Debian, la historia de la evolución de las distribuciones de la familia Red Hat y su interrelación es, por decirlo de alguna forma, caótica, hecho motivado principalmente por decisiones corporativas de la empresa Red Hat. Entendiendo que este tipo de asuntos escapan al alcance del proyecto, pasaremos a desarrollar la estructura existente a día de hoy que, como veremos, recientemente ha sufrido algunos cambios respecto a lo que ha estado vigente durante más de una década.

#### Fedora

Fedora es la distribución *upstream* de la familia, es decir, la que cuenta con las versiones de software y kernel más actualizadas. Fedora recibe una nueva versión cada 6 meses, y cada versión cuenta con actualizaciones regulares de paquetes hasta 13 meses después de su liberación [28]. Aunque es considerada una distribución con un gran nivel de estabilidad, no está recomendada para su uso en entornos de producción. Si está recomendada para usarse en entornos personales, para desarrolladores, estudiantes o, en definitiva, usuarios que quieran experimentar y contar con el SW más actualizado de manera inmediata.

La distribución es mantenida por el Proyecto Fedora, en el que parte de los miembros son designados por Red Hat y otra parte por la comunidad [29]. Aunque la comunidad supone una parte muy importante en el proyecto, y desde este mismo afirman por defender los valores del SW libre y *Open source*, lo cierto es que las grandes decisiones estratégicas y el rumbo del proyecto son marcadas por Red Hat.

Fedora está en el ranking de las 10 distribuciones más usadas [19].



*Figura 10. Logotipo de Fedora [30]*

#### Red Hat Enterprise Linux

RHEL es la distribución comercial de la empresa Red Hat. Se desarrolla a partir de Fedora, una vez se han pasado una serie de pruebas en las partes del SW elegidas de esta, añadiendo correcciones y actualizaciones de seguridad. Este proceso se lleva a cabo de forma interna y no transparente por la empresa.

RHEL no se puede adquirir de forma libre y gratuita, sino que requiere el pago previo de una licencia con la que, además de la distribución, el cliente adquiere beneficios en materia de soporte, formación, documentación, etc. [31]

El objetivo principal de la distribución es ofrecer un producto altamente eficaz y estable, que cuenta con una serie de certificaciones de calidad, destinado a entornos corporativos de producción, alta disponibilidad, etc.

En este sentido, y a diferencia de Fedora, RHEL no cuenta con las versiones más actualizadas de SW ni de kernel, sino con aquellas que, como hemos comentado, han pasado todas las pruebas pertinentes para considerarse oficialmente estables en base a los estándares de calidad que la empresa Red Hat marca.

Red Hat libera una nueva versión de RHEL en intervalos de entre 18 y 24 meses, aunque cada una de ellas cuenta con numerosas actualizaciones parciales que se liberan cada 3-6 meses [32].

A pesar de que las versiones compiladas de RHEL no se pueden adquirir de manera libre, el tener una licencia GPL obliga a publicar su código fuente, mediante el cual desde la comunidad se desarrollan múltiples clones de libre acceso.

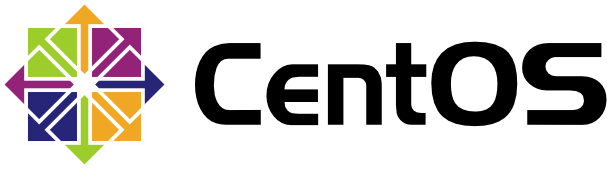


### CentOS

CentOS (*Community Enterprise Operating System*) surge como un clon a partir del código publicado de RHEL que comentábamos anteriormente. Su objetivo es ofrecer un sistema fiable para entornos empresariales de producción, de la misma forma que lo hace RHEL, pero de forma totalmente gratuita y abierta, y con soporte/mantenimiento por parte de la comunidad [34]. De forma similar a lo que ocurre en Fedora, existe el proyecto CentOS que mantiene la distribución [35]. En él participan miembros de la comunidad independientes y empleados de Red Hat, además desde el 2015 CentOS está patrocinado oficialmente por Red Hat y lo posee en forma de marca registrada. Esto se traduce en que las decisiones estratégicas más importantes las toma la empresa y no la comunidad. En este sentido, Red Hat anunció en 2019 un cambio de rumbo fundamental en el proyecto CentOS [36].

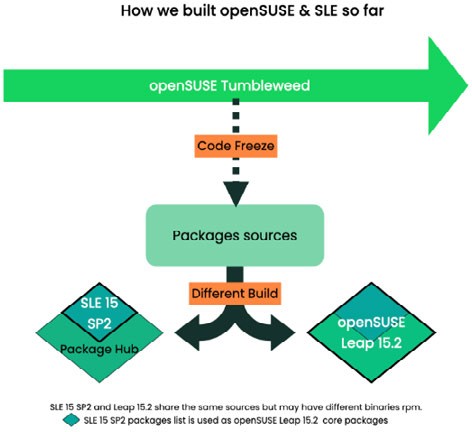
CentOS pasará a denominarse CentOS Stream, y dejará de ser el clon binario de RHEL que hemos comentado. Pasará a ser una distribución *Rolling Release*, es decir, con liberación de actualizaciones constantes en lugar de versiones puntuales [37], situándose *entre* Fedora y RHEL, de forma que, según Red Hat “(…) *trasladar toda nuestra inversión a CentOS Stream es la mejor manera de impulsar aún más la innovación de Linux al brindar a la comunidad de ecosistemas en general una conexión más cercana con el desarrollo de RHEL. CentOS Stream ahora se encuentra entre la innovación del sistema operativo del Proyecto Fedora y la estabilidad de producción de RHEL”* [38]. Resumiendo, Red Hat pretende con este cambio implicar en mayor grado a la comunidad para el desarrollo de las versiones de RHEL, poniendo un escalón transparente entre Fedora y RHEL, y mejorando con esto la calidad del producto.

Sin embargo, este cambio de paradigma no ha sentado muy bien a gran parte de la comunidad CentOS, que se han puesto a desarrollar alternativas con la filosofía de lo que ha sido este sistema hasta ahora. Han surgido diferentes propuestas, una de las más interesantes ha sido anunciada por parte del propio fundador del proyecto CentOS, Gregory Kurtzer [39]. Su propuesta se llama Rocky Linux [40], un clon a nivel binario de RHEL 8.4 que pretende ocupar el espacio que deja libre CentOS con el cambio de rumbo anunciado por Red Hat.



### Familia SUSE

Otra de las familias de distribuciones GNU/Linux más consolidadas a día de hoy es SUSE, a cargo de la empresa con el mismo nombre, que a lo largo de los años ha pasado por diferentes propietarios, pero siempre manteniendo un alto nivel de independencia [42]. SUSE, de forma similar a Red Hat, tiene como objetivo principal ofrecer soluciones comerciales de alta fiabilidad para entornos corporativos, contando entre sus productos con la/s distribución/es que mantiene, así como otros orientados a la virtualización, SAP, cloud computing, etc. [43], repartiéndose la cuota de mercado mundial con Red Hat para este tipo de entornos.



*Figura 13. Familia SUSE [44]*

Entre las características genéricas más destacadas de la familia SUSE se encuentran el uso de paquetes RPM, aunque no compatibles con Red Hat, y el instalador y configurador YaST [45], que proporciona una interfaz sencilla desde la que poder realizar todo tipo de operaciones sobre el sistema.

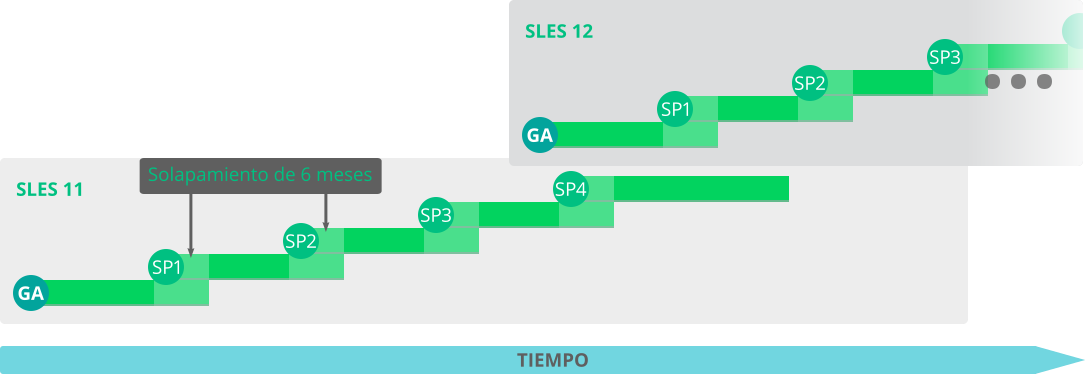
#### SUSE Linux Enterprise

SLE es la versión comercial y de pago de SUSE, orientada a clientes empresariales. Existen diferentes variantes según el propósito para el que se adquiera, aunque, de forma genérica, la más extendida es SUSE Linux Enterprise Server (SLES) [43].

Como hemos mencionado, entre sus características técnicas más relevantes están el uso de paquetes RPM con el gestor de paquetes Zypper [46], y el gestor de configuración YaST, dos de sus principales señas de identidad.

Al estar pensada mayoritariamente para entornos de producción, de forma similar a RHEL, suele priorizar la estabilidad y la tolerancia a errores por encima de la inclusión rápida de nuevas versiones de SW. En este sentido, las versiones principales de SUSE son liberadas cada 4 años, mientras las versiones parciales (*Service Packs*) se lanzan aproximadamente cada año. Con cada versión, SUSE ofrece entre 10 y 13 años de soporte y asistencia oficial, según el producto [47].

Como se puede observar SLES y RHEL tienen características y objetivos similares, de forma que compiten en el mismo mercado, aunque tradicionalmente, por su origen estadounidense RHEL obtiene mayor cuota de mercado en EEUU y SLES, de origen alemán, la obtiene en Europa [48].



*Figura 14. Ejemplo de ciclo de actualizaciones de SLES [47]*

#### openSUSE

La distribución openSUSE es la variante comunitaria y de acceso gratuito que ofrecre SUSE, de manera similar a como Red Hat hace con Fedora y CentOS.

Existen dos variantes:

* + - * 1. **openSUSE Leap**: distribución basada en el código fuente de SEL pero con el añadido de la participación de la comunidad en su desarrollo y mantenimiento [49]. Posee características muy similares a su *hermano* comercial en cuanto a herramientas, procedimientos y estabilidad, aunque con un ciclo de actualizaciones más rápido y un ciclo de vi8a mucho más corto en cada versión, sobre 3 años [50].
        2. **openSUSE Tumbleweed**: se trata de una distribución rolling release [37], que es continuamente actualizada con lo más novedoso de los repositorios SUSE. No es una distribución enfocada para entornos de producción, ya que las continuas actualizaciones pueden provocar fallos en el sistema, problemas con drivers, etc. Por sus características, ofrece la última tecnología en SW disponible, por lo que es una buena opción para usuarios personales como desarrolladores, que deseen estar al día en cuanto al SW instalado [51].



### Otras distribuciones

Como comentamos en puntos anteriores, existen cientos de distribuciones GNU/Linux actualmente mantenidas [53]. Nos hemos centrado en presentar brevemente las que, tras el paso de los años, tienen una mayor consolidación, estabilidad, comunidad y SW disponible.

Sin embargo, hay otras distribuciones “menores” con un número muy significativo de usuarios:

* **Linux Mint** [20]: como comentábamos en el apartado 2.4.1.2, es una distribución basada en Ubuntu enfocada en hacer aún más fácil la experiencia GNU/Linux para usuarios novatos que provienen de otros sistemas.
* **Kali Linux** [54]: es una distribución basada en Debian, especializada en el pentesting y auditoría de redes. Es la distribución por excelencia en el mundo del Hacking Ético.
* **Tails** [55]: también basada en Debian, esta distribución está pensada para la navegación anónima en internet. Se ejecuta desde una memoria externa, como *pen drive*, y está preparada para redirigir todo el tráfico a través de la red Tor [56], entre otras características.
* **Slackware** [57]: se trata de la distribución GNU/Linux más antigua actualmente mantenida. Aunque es potente y estable, su instalación y configuración requieren de conocimientos avanzados, por lo que está enfocada a usuarios expertos. Fue la base inicial para el desarrollo de SUSE.
* **Arch Linux** [58]: Arch es una distribución *rolling release* desarrollada bajo el enfoque KISS (*Keep It Simple, Stupid*) [59], por lo que también requiere de ciertos conocimientos para su instalación y configuración, por lo que es una buena opción para usuarios avanzados que deseen adaptar el sistema según sus necesidades y prescindir de todo aquello ajeno a sus propósitos.
* **Manjaro** [60]: es una distribución basada en Arch y con un gran número de usuarios a día de hoy [19], que comparte con Arch características en cuanto a simplicidad y actualizaciones, aunque está pensada para facilitar un poco las cosas en cuanto a instalación y configuración respecto a Arch, por ejemplo instalando por defecto un entorno de escritorio.

*Figura 16. Árbol genealógico con algunas de las distribuciones GNU/Linux más importantes [61]*

## Analizador Léxico

GramaticaTokenManager es literalmente el analizador lexico

SimpleCharStream se usa para leer los tokens. Usa por debajo clases como Reader

Diagrama

Descripción generada automáticamente

Figura 1. Funcionamiento del Analizador Léxico

## Estados Léxicos

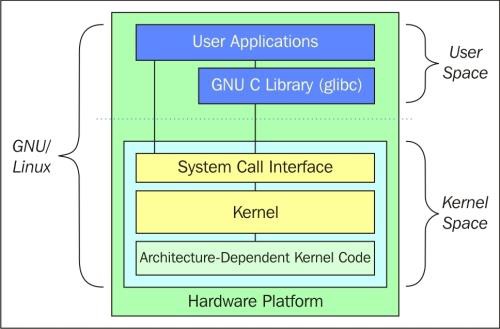
Los estados léxicos permiten efectuar un conjunto diferente de producciones de expresiones regulares en un caso concreto. Dicho de otra manera, los estados léxicos sirven para que un token este definido de una forma, y en cierto momento del análisis, pueda estar definido de otra forma.

Supongamos que desea escribir un procesador JavaDoc para extraer los comentarios JavaDoc de un documento. La mayor parte de Java está tokenizada de acuerdo con las reglas ordinarias regulares de Java. Pero dentro de los comentarios JavaDoc, se aplica un conjunto diferente de reglas en las que las palabras clave deben ser reconocidas y donde las nuevas líneas son significativas. /\*\*\*/@param

Para resolver este problema, podríamos usar dos estados léxicos: uno para la tokenización regular de Java y otro para la tokenización dentro de los comentarios de JavaDoc.

### Introducción

Como habíamos comentado anteriormente, un sistema operativo GNU/Linux se compone, como mínimo, del núcleo Linux y de las aplicaciones y bibliotecas básicas de GNU. Luego, por encima, tendríamos toda la serie de aplicaciones de usuario que, como se ha visto anteriormente, variarán en función de la distribución elegida.



*Figura 17. Arquitectura básica GNU/Linux [62]*

En la capa más alta de la estructura nos encontramos con la interfaz de usuario, de la que podemos encontrar dos grandes tipos:

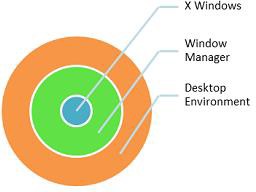
* La **interfaz de línea de comandos** (CLI). Es una interfaz en modo texto, mediante la cual el usuario interactúa con el sistema escribiendo comandos de forma manual con el teclado. Sus ventajas principales son el bajo consumo de recursos, su rapidez y la precisión a la hora de realizar las operaciones. La desventaja es que requiere un nivel de conocimiento más avanzado para su manejo.
* La **interfaz gráfica de usuario** (GUI). En este caso disponemos de un entorno gráfico que permite interactuar al usuario con el sistema, haciendo uso del teclado y el ratón, a través de diferentes elementos gráficos como ventanas, iconos, imágenes, etc. Consume más recursos que la CLI y tarda más en realizar las operaciones, además de ser menos precisa. Sin embargo, es mucho más intuitiva y no requiere de grandes conocimientos técnicos para su manejo.



*Figura 18. Ejemplos de CLI y GUI en GNU/Linux [63]*

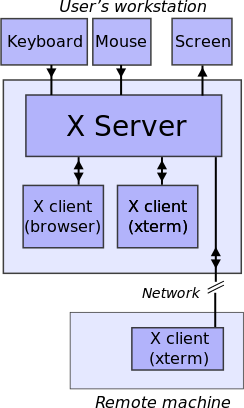
Según el propósito para el que se instala un sistema GNU/Linux, se suele instalar sólo la CLI para el caso de servidores, y la GUI en el caso de sistemas de escritorio, aunque la GUI provee fácilmente acceso a la CLI mediante aplicaciones llamadas emuladores de terminal.

La arquitectura GUI de GNU/Linux se compone de varias partes, pasamos a describirlas de más bajo a más alto nivel:



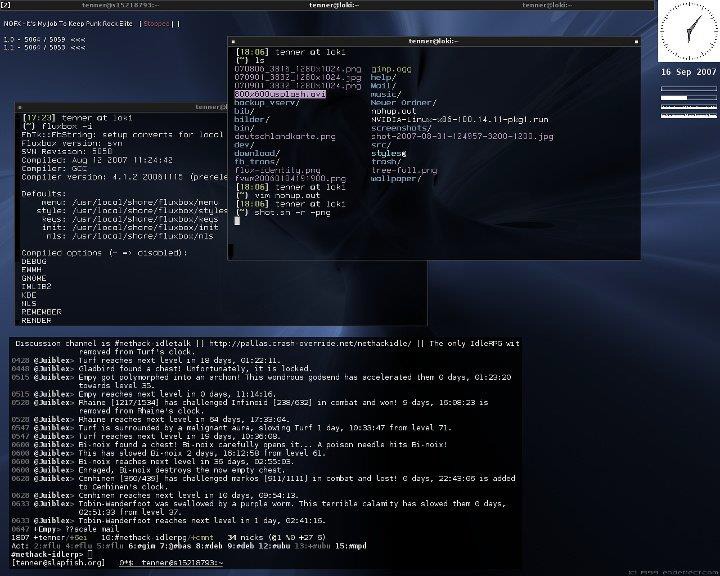
*Figura 19. Niveles de la arquitectura GUI de GNU/Linux [64]*

* + ***Sistema de ventanas X***: es la parte fundamental e imprescindible de la arquitectura. Se trata de una especificación (biblioteca y protocolo) que permite dibujar elementos gráficos en la pantalla. La implementación consolidada desde hace dos décadas es desarrollada por la X.Org [65], y se conoce popularmente como X11. Utiliza un modelo cliente-servidor, siendo el cliente las aplicaciones que solicitan recursos gráficos y el servidor los periféricos de entrada del usuario, como el teclado y el ratón. Las aplicaciones (clientes) se pueden ejecutar de forma local, o de forma remota a través de la red [66]. Es la implementación por defecto en GNU/Linux, aunque los últimos años ha surgido una alternativa que pretende desplazarla, Wayland [67], aunque su implementación todavía es muy reducida en comparación a X.



*Figura 20. Sistema de ventanas X [68]*

* + ***Gestor de ventanas*** [69]: Se trata de una aplicación (cliente) X con características particulares, que permite añadir funcionalidades al pintado de las ventanas y elementos gráficos: mover las ventanas, redimensionarlas, minimizarlas, aplicar estilos, desplegar menús contextuales, etc. Existen una gran variedad de gestores de ventanas, que pueden ejecutarse forma independiente o sirviendo de soporte para un entorno de escritorio. Algunos ejemplos gestores de ventanas que se suelen usar de forma independiente son Fluxbox [70] y Openbox [71].



*Figura 21. Fluxbox 1.3.6 [72]*

* Entorno de escritorio: se trata de otra aplicación especial, que se ejecuta sobre el sistema de ventanas y utiliza al gestor de ventanas, al que complementa proporcionando el resto de características que permiten tener una interfaz de usuario completa de forma unificada: menús de aplicaciones y configuración, iconos, widgets, temas, etc. Además, suelen instalar consigo un conjunto de aplicaciones y herramientas integradas para mejorar la experiencia de usuario [73].



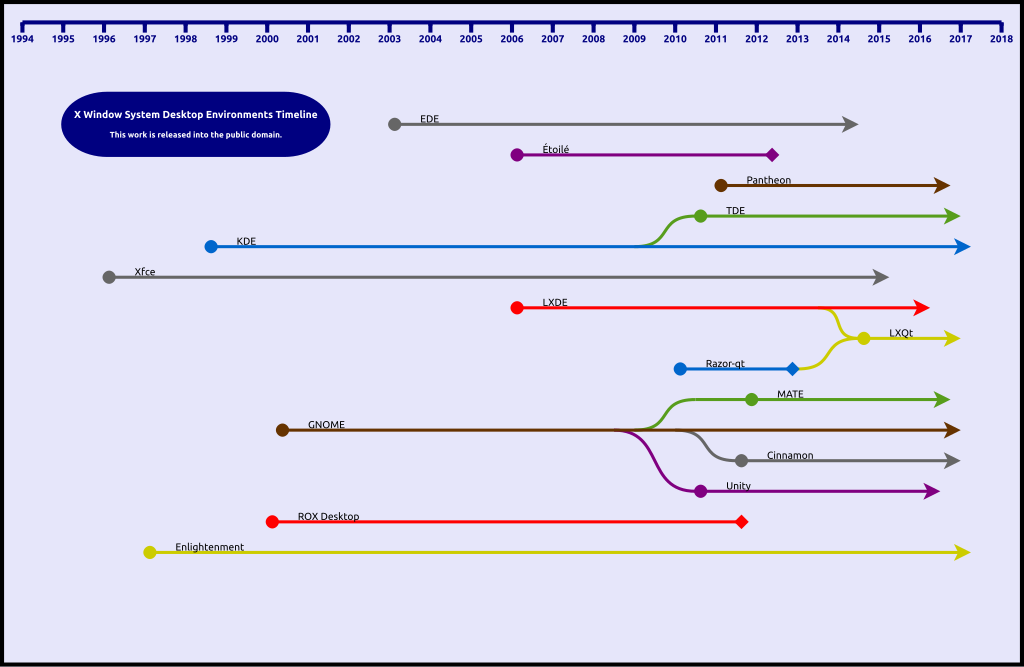
*Figura 22. Entorno de escritorio GNOME ejecutándose en Debian [74]*

### Principales entornos de escritorio GNU/Linux

El año 1995 supuso un punto de inflexión en cuanto al uso extendido de ordenadores por parte de usuarios no expertos, con la salida al mercado de Windows 95 [75]. Hasta entonces, los equipos que habían contado con un entorno de escritorio más accesible para usuarios no avanzados eran los Macintosh System de Apple, aunque su elevado precio hacía que el número de usuarios fuera considerablemente reducido [76]. El mayor valor añadido de la nueva versión del sistema operativo de Microsoft fue su amigable e intuitiva interfaz de usuario, que hizo que fuera el sistema operativo que más rápidamente vio crecer su número de usuarios hasta la fecha.

Esto motivó a la comunidad Unix y GNU/Linux a buscar alternativas similares para sus sistemas.

A diferencia de lo que ocurre en sistemas privativos, la versatilidad que permiten los sistemas libres GNU/Linux se ha traducido en que haya disponibles múltiples entornos de escritorio, que el usuario puede elegir en función de sus gustos o necesidades.



*Figura 23. Evolución de los entornos de escritorio GNU/Linux [77]*

Como hemos comentado anteriormente, los entornos de escritorio se apoyan en el sistema de ventanas X para funcionar, aunque poco a poco también ofrecen compatibilidad con Wayland. Cada entorno viene con un gestor de ventanas integrado por defecto, aunque es posible cambiarlo si se deseara, no es lo recomendable.

Las diferencias básicas entre entornos de escritorio se pueden resumir en: biblioteca con la que se han desarrollado, su portabilidad, el consumo de recursos, la apariencia, el nivel de personalización y el conjunto de aplicaciones integradas.

Pasamos a desarrollar las características de algunos de los entornos de escritorio GNU/Linux más consolidados a día de hoy:

#### KDE

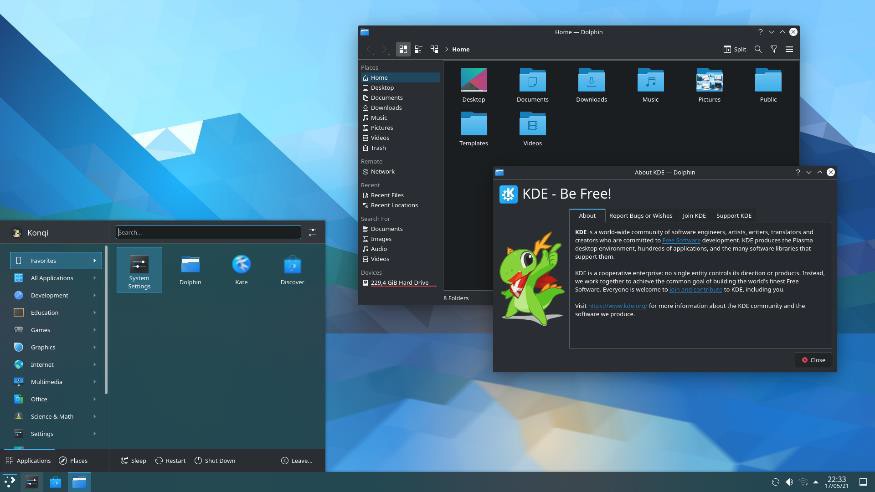
El proyecto KDE surge en el año 1996 como intento de construir un entorno de escritorio amigable para usuarios de sistemas Unix y GNU/Linux, tras el éxito de Windows 95 y buscando superar el primitivo y privativo entorno de escritorio común de los sistemas Unix hasta ese momento, el CDE (*Common Desktop Environment*) [78].

Aunque desde sus inicios el proyecto KDE persiguió crear un entorno de escritorio enmarcado en el software libre, bajo licencia GPL, el uso de las bibliotecas de Qt [79], por entonces de carácter privativo, crearon cierta controversia y no fue oficialmente respaldado por el proyecto GNU. Sin embargo, las bibliotecas de Qt para GNU/Linux pasaron a ofrecerse con licencia GPL a partir del año 2000 [80], lo que sofocó los miedos de parte de la comunidad del software libre, que no veían con buenos ojos que uno de los escritorios más usados en los sistemas libres contuviera código propietario y cerrado.

Desde entonces, KDE ha continuado su desarrollo con la liberación de versiones en intervalos de 2 a 6 años, y manteniéndose como uno de los tres escritorios más usados en GNU/Linux.

A partir del año 2014, el proyecto cambió drásticamente su estructura, dejando el nombre KDE para referirse a la comunidad explícitamente [81], y pasando a llamarse el entorno de escritorio “Plasma” [82], siendo KDE Plasma 5 la quinta versión oficial del entorno de escritorio.

Plasma 5 fue lanzado el 15 de julio de 2014, con importantes cambios a nivel de usuario y de desarrollo, como la migración completa a QML [83]. En general, este nuevo enfoque fue bien recibido por la comunidad, y algunas de sus características principales son:

* El menor consumo de recursos respecto a sus predecesores y a otros escritorios como GNOME 3.x.
* Totalmente personalizable por parte del usuario.
* El uso de widgets llamados *plasmoids* [84].
* Pose un conjunto de aplicaciones integradas que permiten llevar a cabo todas las tareas de un usuario medio, y que proveen una gran cantidad de opciones [85].
* Gestor de ventanas KWin [86].
* Es el escritorio por defecto en múltiples distribuciones principales: Kubuntu, openSUSE, KDE Neon, Fedora KDE y Manjaro KDE [69].

#### Xfce

Xfce surge en el año 1996 de la mano del programador Olivier Fourdan, con el objetivo, al igual que los entornos descritos previamente, de proporcionar una alternativa libre y amigable para sistemas GNU/Linux frente al CDE de Unix.

En sus primeras versiones, pese a liberarse con licencia GPL, fue escrito con el lenguaje XForms [99], que por entonces era de código cerrado, lo que impidió el patrocinio oficial de Xfce por parte de la comunidad del software libre y entidades como la FSF y el proyecto GNU. Sin embargo, a partir de su versión 4 liberada en 2003 todo el código se había migrado a GTK+, por lo que pasó a catalogarse oficialmente como software libre.

La popularidad de Xfce comenzó a crecer de forma exponencial a partir del año 2011, motivado este hecho en parte por el gran salto entre GNOME 2.x y GNOME 3.x, que empujó a muchos usuarios a buscar otros entornos de escritorio alternativos [100].

El desarrollo de Xfce es lento, ya que son pocos los programadores que participan activamente en el proceso. Sin embargo, es indiscutiblemente uno de los entornos de escritorio con mayor nivel de aceptación entre la comunidad GNU/Linux debido, entre otras cuestiones, a su ligereza, su estabilidad y su coherencia entre versiones, huyendo de cambios bruscos, a diferencia de los entornos mencionados previamente [101].

La última versión disponible de Xfce es la 4.16, disponible desde diciembre de 2020. Las características principales de este entorno de escritorio son [102]:

* Es muy ligero, requiere un bajo consumo de recursos.
* Pese a su ligereza, guarda razonablemente bien el equilibrio con la presentación estética.
* Alto nivel de estabilidad.
* Ampliamente personalizable.
* Fácil de manejar e intuitivo.
* Modular.
* Cuenta con aplicaciones integradas, aunque en menor medida y menos potentes que KDE y GNOME.
* Alto seguimiento de los estándares, por lo que es capaz de ejecutar muchas aplicaciones pensadas para otros entornos.
* Es el entorno por defecto de Debian.

#### Otros entornos

De forma similar a lo que ocurría con las distribuciones, la versatilidad en el mundo del software libre ha propiciado que existan decenas de entornos de escritorio y gestores de ventanas. Nos hemos centrado en los que tienen un mayor número de usuarios a la vez que una posición consolidada con el paso de los años, lo que garantiza unos ciertos estándares de calidad, estabilidad y soporte comunitario.

Por mencionar otras alternativas, además de los ya comentados Mate y Cinnamon, *forks* de GNOME 2.x, podemos destacar:

* Pantheon: se trata de un entorno desarrollado por los creadores de la distribución Elementary OS, basada en Ubuntu y que pretende ofrecer un escritorio en el que prima lo estético y la simplicidad, tomando como referencia a MacOS [104].
* LXDE: escritorio ultra-ligero diseñado para equipos antiguos y/o con pocos recursos [105].
* Unity: fue desarrollado por Canonical y se convirtió en el entorno oficial de Ubuntu durante algunas versiones, en reemplazo de GNOME tras su salto a la versión 3.x. Sin embargo, a partir de la versión 18.04, Ubuntu volvió a GNOME y Unity fue prácticamente descontinuado por Canonical [106].



*Figura 28. Entorno Pantheon ejecutándose sobre Elementary OS [104]*

## Analizador Sintáctico o Semántico

Nos dice como tienen que estar acomodadas las cosas en el código fuente que tenemos para hacerlo un programa valido.

Diagrama

Descripción generada automáticamente

Figura 2. Funcionamiento de un Analizador Sintáctico

### Introducción

Una vez explicadas las características del software libre y open source, del sistema GNU/Linux, y de haber repasado algunas de las principales distribuciones y entornos de escritorio, vamos a pasar a comentar algunas de las herramientas que existen para instalar y configurar las distribuciones cuando es necesario mantener muchos equipos a la vez y realizar tareas repetitivas.

La automatización de procesos no es algo nuevo, sino que está ligado estrechamente al desarrollo de los sistemas informáticos, desde simples scripts que contienen un conjunto de comandos hasta los instaladores de las distribuciones GNU/Linux.

Sin embargo, especialmente en la última década, este concepto ha cobrado una mayor importancia, con el crecimiento exponencial de las redes de telecomunicaciones, de internet y, en definitiva, del uso diario de los dispositivos informáticos. Este proceso ha ido de la mano de un aumento significativo en el número de sistemas y de su complejidad, necesarios para mantener toda la infraestructura tecnológica de nuestros días, tanto en entornos de desarrollo, pruebas y producción.

En este sentido, los sistemas han ido avanzando hacia una mayor virtualización y automatización, surgiendo conceptos clave como los contenedores, el cloud computing, la integración/distribución continua, DevOps o la automatización TI (infraestructura tecnológica), todos ellos estrechamente relacionados [107]–[111]

Centrándonos en la parte que nos ocupa, la de la automatización, vamos a mencionar algunas de las herramientas más usadas a día de hoy para la automatización de la TI en entornos en grandes entornos corporativos como en granjas de servidores y centros de datos, ya sean físicos, virtuales, locales o en la nube. Estas herramientas son conocidas como gestoras de configuración [112] y, cuando se trata de entornos extremadamente grandes, suele ser necesario, además, herramientas de orquestación [113].

Las tres herramientas de gestión de configuración más usadas a día de hoy son: Puppet, Chef y Ansible [114]–[116]. Cada una con sus propias características técnicas, pero con el objetivo común de automatizar la configuración y gestión de los sistemas, reduciendo tareas repetitivas a un solo click. Todas están relacionadas con las metodologías open source.

Esto es un resumen del paradigma existente en entornos empresariales. Sin embargo, en entornos educativos y públicos no está muy extendido, ni en la práctica para sus sistemas ni en los planes de enseñanza, bien sea porque las necesidades son diferentes o bien porque los cambios se producen de forma más lenta, por diferentes factores [117].

### Herramienta de aprovisionamiento para equipos físicos

Como hemos comentado, las herramientas citadas están pensadas especialmente para entornos virtualizados, por lo general en la nube, donde crear nuevos equipos con el sistema operativo y las herramientas necesarias para su gestión instalados es tan fácil como darlo de alta a través de una interfaz de comandos o gráfica.

Sin embargo, cuando hablamos de equipos físicos, necesitamos algunas funcionalidades extra: reconocimiento de las características hardware, particionado de discos, arranque por red, instalación del sistema, configuración de la red, etc.

En este sentido, existen una serie de herramientas que permiten instalar y configurar equipos físicos, y controlar su ciclo de vida completo: instalación, pos instalación, mantenimiento y reinstalación. En la siguiente tabla resumimos algunas las herramientas creadas para este propósito:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nombre | Mantenido  Actualmente | Distribuciones  compatibles | Curva de aprendizaje | Interfaz | Licencia | Comentarios |
| **Cobbler [118]** | Si | Familia Red Hat | Baja-media | CLI/GUI | GPL | La más antigua centrada en Red Hat. |
| **FAI [119]** | Si | Debian, Ubuntu, CentOS | Media | CLI | GPL | La más antigua centrada en Debian. |
| **Foreman [120]** | Si | La gran mayoría | Alta | CLI/GUI | GPL | La más potente. Permite combinar equipos físicos con máquinas virtuales y nube. |
| **MaaS [121]** | Si | Centrado en Ubuntu, pero ofrece compatibilidad con CentOS, RHEL y Windows | Baja-media | CLI/GUI | AGPL | No permite instalar por defecto versiones del sistema con entorno de escritorio. |
| **m23 [122]** | Si | Debian, Ubuntu y Linux Mint, con todos los entornos de escritorio disponibles | Baja | GUI | GPL | La más fácil de usar. |
| **Razor [123]** | No | La gran mayoría | Media | CLI | Apache | Descontinuado desde julio de 2020 |
| **Spacewalk [124]** | No | Familia Red Hat | Media | CLI/GUI | GPL | Descontinuado desde mayo de 2020 |
| **xCAT [125]** | Si | La gran mayoría | Media-alta | CLI/GUI | EPL |  |

*Tabla 1. Herramientas de aprovisionamiento*

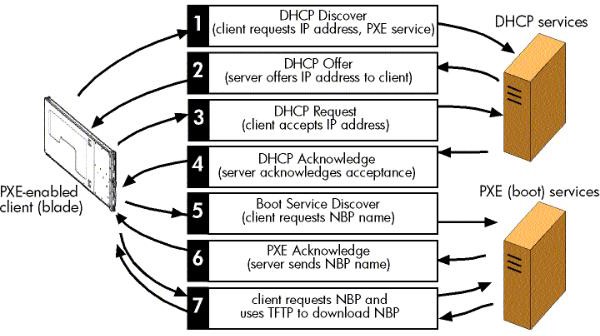
En próximos apartados desarrollaremos las características de las herramientas concretas que se han evaluado como candidatas para la realización de este proyecto.

### Protocolos y bibliotecas auxiliares

Como se ha dicho anteriormente, para controlar el ciclo de vida de equipos físicos hacen falta una serie de tecnologías auxiliares, que las herramientas analizadas se encargan de administrar, de forma más o menos transparente:

- PXE (*Preboot Execution Environment*) [126]: es un entorno que permite arrancar equipos informáticos a través de la red, en lugar dispositivos de memoria como CDROM, disco duro o pen drive. Se basa en un modelo cliente-servidor, siendo el cliente el equipo que queremos instalar, al que se le habilita el arranque PXE a nivel de BIOS. El equipo servidor es el que proporcionará al cliente lo necesario para instalar el nuevo sistema operativo: imagen, instrucciones, etc.

* DHCP (*Dynamic Host Configuration Protocol*) [127]: es un protocolo que permite asignar direcciones IP de forma dinámica. En el entorno PXE se utiliza para asignar una IP al equipo que arranca por red, además de indicarle otros parámetros, como el servidor en el que se encuentra el contenido necesario para la instalación del SO.
* TFTP (*Trivial file transfer Protocol*) [128]: se trata de un protocolo de transferencia de archivos. En el arranque PXE se utiliza para enviar desde el servidor al cliente todos los ficheros necesarios para comenzar la instalación del SO.
* NBP (*Network Bootstrap Program*) [129]: es el programa que ejecuta el cliente tras el arranque PXE y que se encarga de ejecutar el SO remoto que le proporciona el servidor.
* PXELINUX [130]: es el cargador de arranque utilizado en las instalaciones de sistemas GNU/Linux a través de la red. Es una variante del proyecto syslinux [131] específica para este propósito.



*Figura 29. Diagrama del proceso de arranque PXE [132]*

## Recursividad

Como se ha comentado en apartados anteriores, a día de hoy una herramienta fundamental tanto para el desarrollo de SW como para la administración de sistemas, es el SW de virtualización. Existen diferentes tecnologías de virtualización actualmente extendidas, como los contenedores

[108] o, las más clásicas, máquinas virtuales que es el tipo de virtualización que se ha utilizado en este proyecto.

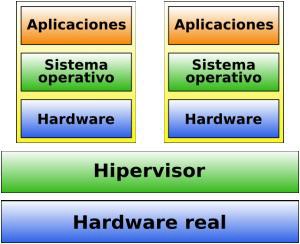
Según [133], la definición de máquina virtual es: “*Una máquina virtual (VM) es un entorno que funciona como un sistema informático virtual con su propia CPU, memoria, interfaz de red y almacenamiento, pero el cual se crea en un sistema de hardware físico*”.

Las ventajas que esta tecnología nos aporta son claras: ponemos disponer de varios sistemas virtuales, que, a grandes rasgos, funcionan como si fueran reales, haciendo uso de un único sistema hardware.

La parte fundamental de la virtualización de sistemas es el hipervisor. Se trata del SW que actúa de capa intermedia entre el sistema virtualizado o huésped y los recursos HW del anfitrión. De esta forma, se encarga de asignar los recursos compartidos a cada huésped, y así permite que en un mismo equipo físico se ejecuten de forma simultánea varios sistemas virtualizados sin interferir entre ellos [134].

Existen dos grandes tipos de hipervisores [135]:

1. **Hipervisor de tipo 1** o nativo: es aquel que se instala directamente sobre el equipo anfitrión sin necesidad de un sistema operativo, ya que ocupa su lugar de este. Al no contar con un sistema operativo como tal, este tipo de hipervisor se usa en equipos destinados exclusivamente a la virtualización, generalmente en entornos empresariales. Sus principales ventajas, al tener el control directo sobre el HW, tienen que ver con la cantidad de recursos disponibles para asignar a los huéspedes, así como una asignación dinámica de estos. Algunos ejemplos relevantes de hipervisores de tipo 1 son: VMware vSphere ESX, Microsoft Hyper-V Server o Citrix XenServer.



*Figura 30. Capas de un hipervisor tipo 1 [136]*

1. **Hipervisor de tipo 2** o alojado**:** en este caso, el hipervisor se ejecuta sobre un sistema operativo anfitrión previamente instalado, funcionando a nivel de aplicación. Por tanto, es útil en equipos que están destinados a diferentes usos además de la virtualización. Esta es su principal ventaja, además de que, al contar con la capa del SO anfitrión por debajo, la compatibilidad HW suele estar más asegurada que en el tipo 1. Algunos ejemplos de este tipo de hipervisor son VMware Workstation y VirtualBox.



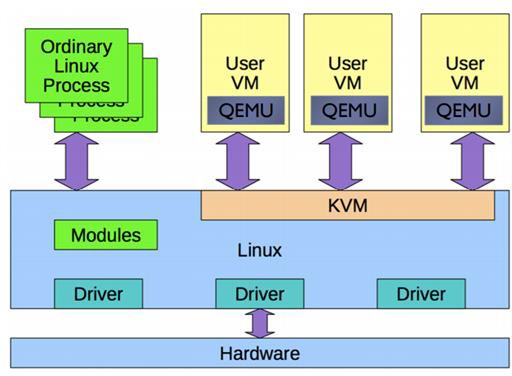
*Figura 31. Capas de un hipervisor tipo 2 [137, p. 2]*

Aunque esta es la clasificación general del SW de virtualización, existe otro tipo intermedio, que el que se implementa por defecto con GNU/Linux.

Este tipo de virtualización consta de dos partes diferenciadas:

1. **KVM** (Máquina Virtual basada en Kernel): se trata de un módulo interno del núcleo Linux que convierte a este en un hipervisor, proporcionando interfaces para que los sistemas huésped accedan directamente a los recursos HW que este núcleo controla, principalmente la CPU y la memoria RAM [138].
2. **QEMU**: es el SW que se ejecuta a nivel de usuario y permite crear y gestionar las máquinas virtuales, además de virtualizar los diferentes dispositivos necesarios para su funcionamiento. Utiliza el módulo KVM para el acceso al núcleo y los recursos HW del anfitrión [138].

Por tanto, la virtualización QEMU/KVM es un tipo mixto de hipervisor, ya que una parte se ejecuta directamente sobre el núcleo anfitrión y otra a nivel de usuario, de forma que el sistema anfitrión se puede destinar a diferentes usos. En la siguiente figura podemos observar las capas en las que se divide esta tecnología de virtualización:



# DESARROLLO DE LA PROPUESTA

## Introducción

Una vez expuestas las principales tecnologías que forman parte del proyecto, vamos a pasar a desarrollar cuáles se han elegido para la implementación, los procedimientos seguidos, así como detallar el entorno usado para los desarrollos y las pruebas.

## Procesamiento de la Información en Aplicaciones Telemáticas (PIAT)

### Introducción

Como hemos visto en el apartado 2.4, existen multitud de distribuciones GNU/Linux con una base común, pero con diferentes particularidades. A la hora de decidir cuál es la distribución más adecuada, se han tenido en cuenta los siguientes factores. En este caso, asumiendo que al ser distribuciones GNU/Linux todas cuentan con unos mínimos éticos en cuanto al software libre, estos criterios son principalmente técnicos y funcionales:

* Que esté fuertemente consolidada, con un número suficiente de usuarios, desarrolladores y documentación. De esta forma podemos tener garantía en cuanto a la resolución de posibles errores/problemas y que a medio-largo plazo la distribución no va a ser desatendida. En este sentido, entendemos como imprescindible que cuente con versiones LTS (soporte a largo plazo).
* Que históricamente no se hayan producido cambios bruscos en su modelo de desarrollo ni en sus características fundamentales.
* Que sea muy estable. Necesitamos una distribución que contenga el menor número de fallos y comportamientos inesperados posibles, de forma que no se vea afectada la comunidad educativa en su uso diario, ni cargue de trabajo extra al personal técnico.
* En relación con los dos puntos anteriores, la distribución a usar debe contar con actualizaciones puntuales y lo mejor programadas posible. Descartamos las rolling release porque son propensas a sufrir más fallos críticos y comportamientos inesperados, como la pérdida de configuraciones o drivers.
* Debe ser compatible con alguna de las herramientas de despliegue descritas en el apartado 2.6.
* Que sea de acceso gratuito.
* Que los esfuerzos principales en el desarrollo de la distribución sean para el uso en escritorio, ya que es el que nosotros pretendemos darle.
* Que cuente con la mayor cantidad de software disponible posible.

### Desarrollo de la comparativa

En la siguiente tabla realizamos un análisis básico en cuanto a dichas características, otorgando a cada distribución puntuación del 0 al 5 o de Verdadero/Falso según procede. Dichas puntuaciones se han asignado en base al análisis que se ha hecho de lo expuesto en el apartado

2.4 de esta memoria:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Debian** | **Ubuntu** | **Fedora** | **RHEL** | **CentOS** | **SLES** | **openSUSE** |
| **Consolidación y LTS** | 5 | 5 | 5 | 5 | 3 | 5 | 5 |
| **Sin cambios bruscos históricamente** | V | V | V | V | F | F | F |
| **Estabilidad** | 5 | 5 | 4 | 5 | 3 | 5 | 5 |
| **Actualizaciones** |  |  |  |  |  |  |  |
| **Compatibilidad con herramientas de despliegue** | 5 | 5 | 5 | 5 | 5 | 3 | 3 |
| **Gratuidad** | V | V | V | F | V | F | V |
| **Enfoque principal para escritorio** | 4 | 5 | 4 | 4 | 4 | 0 | 5 |
| **Software disponible** | 5 | 5 | 4 | 4 | 4 | 4 | 4 |

*Tabla 2.Comparativa de distribuciones GNU/Linux*

En los inicios del proyecto se barajó como principal opción la distribución CentOS, ya que venía siendo una de las más estables, fácilmente configurables, personalizables y automatizables y con mayor soporte por parte de las herramientas de despliegue. Sin embargo, el cambio del rumbo que sufrió, pasando a ser una distribución rolling release hizo que fuera finalmente descartada, en base a lo que hemos comentado anteriormente.

Dentro de la misma familia Red Hat, RHEL fue automáticamente descartado por ser una distribución de pago, enfocada exclusivamente al uso empresarial. En el caso de Fedora, el hecho de que sea el “campo de pruebas” de RHEL y que tenga un ciclo de actualizaciones no del todo claro, con un soporte máximo de 13 meses por cada versión, nos hizo descartarla por no ofrecer las garantías de estabilidad y mantenimiento mínimas que consideramos necesarias para nuestro proyecto.

Pasando a la familia SUSE, SLES es descartado principalmente por ser una versión de pago y orientada a servidores. Aunque existe SLED enfocada a escritorio, sigue siendo de pago y no es la apuesta principal de los desarrolladores. Dentro de esta familia existen las dos versiones gratuitas openSUSE. La variante Tumbleweed fue descartada automáticamente por ser una rolling release. En cuanto a la variante Leap, es suficientemente estable en el SW y un calendario de actualizaciones bien definido, sin embargo, decidimos descartarla porque no está enfocada principalmente al uso para escritorio de usuarios medios, el software disponible por defecto es menor que en otras familias y porque no cuenta con gran compatibilidad por parte de las herramientas de despliegue.

También se han analizado otras distribuciones populares hoy en día, como Manjaro Linux, el cual, aunque cuenta con interesantes características innovadoras y enfoque para escritorio, es una distribución rolling release y consideramos que su desarrollo y su comunidad no están lo suficientemente asentados.

Otras distribuciones como Linux Mint o ElementaryOS se han descartado, ya que están basadas en Ubuntu y consideramos que las características que las diferencian de este o no nos son útiles o no son lo suficientemente determinantes como para decantarse por ellas en lugar de la distribución de la que parten.

Esto nos deja con dos candidatos: Debian y Ubuntu. Ambas son potencialmente aptas para nuestros propósitos y comparten un gran número de características, en cuanto a tipo de paquetes, estabilidad, comunidad, continuidad y política de actualizaciones-soporte, teniendo en cuenta que Ubuntu toma a Debian como base, aunque a día de hoy los desarrollos sean independientes.

### Conclusiones

Llegados a este punto, sólo nos queda describir los principales hechos diferenciadores entre ambas distribuciones y concluir con la decisión tomada:

* Software disponible: Ubuntu cuenta con una mayor cantidad de SW disponible que Debian, ya que comparte con él toda la paquetería DEB, pero añade además otros tipos de repositorios propios como Snap [98] o PPA [141], esto convierte a Ubuntu en una de las distribuciones GNU/Linux con más SW disponible por defecto.
* En cuanto al SW instalado por defecto, Debian es más ligero que Ubuntu, ya que por defecto instala menos programas y herramientas. Esto es debido a que tienen diferentes enfoques, como ya contamos en la sección 2.4. Mientras que Ubuntu persigue que usuarios recién llegados a GNU/Linux tengan disponible todo lo que presumiblemente es necesario para un uso cotidiano, Debian está orientada a usuarios con mayor trayectoria, que son capaces y desean instalar y configurar manualmente todo lo necesario para cubrir sus necesidades.
* Enfoque escritorio/servidor: ambas distribuciones pueden funcionar tanto como sistemas de escritorio como servidores, pero mientras que Ubuntu diferencia de forma clara ambas versiones, con imágenes, instaladores y otras características diferentes, Debian parte de la misma imagen y es en tiempo de instalación dónde se elige si se instalan, o no, los paquetes extra para el entorno de escritorio. En este sentido, el proyecto principal de Ubuntu es su versión para escritorio, enfocada al usuario final medio, facilita los procedimientos y realiza un mayor trabajo previo de integración y personalización de los diferentes entornos de escritorio. En cambio, Debian, como hemos comentado en puntos anteriores, es más versátil y está pensado para que el usuario instale y configure manualmente, también, el entorno de escritorio.

Teniendo en cuenta estos factores, y el hecho de que el proyecto está destinado, aun hipotético uso académico diario por parte de alumnado y profesorado, y, por otro, a la instalación y mantenimiento por parte del personal técnico, llegamos a la conclusión de que la distribución GNU/Linux más adecuada para implementar con el proyecto es Ubuntu. Consideramos que ofrece una serie de ventajas por su enfoque al usuario final, tanto para quien lo instale como para quien lo use, además de ser, como ya hemos comentado, una de las distribuciones que cuenta por defecto con más cantidad de SW disponible, lo que para nosotros es una prioridad de cara a cubrir los requisitos de las asignaturas en cuanto a SW.

Las versiones más recientes de Ubuntu son las siguientes:

1. Versiones LTS:

* Ubuntu 20.04 “Focal Fossa”. Liberada en abril de 2020, es la versión estable con soporte de larga duración de 5 años más reciente.

1. Versiones intermedias:

* Ubuntu 20.10 “Groovy Gorilla”. Liberada en octubre de 2020.
* Ubuntu 21.04 “Hirsute Hippo”. Liberada en abril de 2021.

Como ya comentamos en el apartado 2.4.1.2, las versiones estables son las LTS, con soporte oficial de 5 años y con actualizaciones menos arriesgadas centradas en la estabilidad y la seguridad, añadiendo funcionalidades nuevas sólo cuando están completamente probadas y no afectan al funcionamiento del entorno. Por tanto, la versión que utilizaremos para la implementación será la 20.04, que en estos momentos se encuentra en su tercera versión de mantenimiento, la 20.04.03.

## Entorno de escritorio utilizado

Una vez hemos elegido justificadamente la distribución Ubuntu 20.04, pasamos a analizar y decidir qué entorno de escritorio es más adecuado para nuestros propósitos.

Ubuntu como tal viene con el escritorio GNOME 3.x incluido por defecto. Sin embargo, como ya comentamos en el apartado 2.4.1.2, existen los “sabores” de Ubuntu, ligeras variantes del sistema con otros entornos de escritorio y sus correspondientes herramientas. Los siguientes son los sabores oficiales de Ubuntu, cuyo mantenimiento está a cargo de Cannoincal y la comunidad:

* Kubuntu: con el escritorio KDE Plasma 5.
* Lubuntu: con el escritorio LXQt.
* Xubuntu: con el escritorio Xfce.
* Ubuntu MATE: con el escritorio MATE.
* Ubuntu Budgie: con el escritorio Budgie.

Para hacer una comparación entre dichos entornos de escritorio, vamos a tener en cuenta los siguientes factores técnicos y de usabilidad:

* Consolidación. Nos interesa que el entorno tenga una trayectoria, una implementación y una comunidad suficientemente aceptables.
* Consumo de recursos.
* Estabilidad. Cuanto más estable sea el entorno, menos repercutirá negativamente en la actividad diaria de la comunidad educativa y en el trabajo del personal técnico.
* Facilidad de uso para usuarios novatos o impacto. Volvemos a insistir en que este proyecto está enfocado al uso diario de estudiantes que presuponemos son usuarios novatos de GNU/Linux en su mayoría.
* Personalización. Facilidad para personalizar los diferentes elementos del entorno a nuestro gusto.
* Herramientas integradas. Cantidad y utilidad de las herramientas que incorpora cada entorno.

En la siguiente tabla comparamos los distintos sabores de Ubuntu en base a los factores comentados:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Ubuntu (GNOME)** | **Kubuntu (Plasma 5)** | **Lubuntu (LXQt)** | **Xubuntu (Xfce)** | **U. MATE** | **U. Budgie** |
| **Consolidación** | Alto | Alto | Alto | Alto | Medio | Bajo |
| **Consumo de recursos** | Alto | Medio | Bajo | Medio | Medio | Alto |
| **Estabilidad** | Alto | Alto | Alto | Alto | Medio | Medio |
| **Impacto** | Alto | Bajo | Bajo | Bajo | Medio | Alto |
| **Personalización** | Medio | Alto | Bajo | Alto | Bajo | Alto |
| **Utilidad herramientas** | Alto | Alto | Bajo | Medio | Medio | Alto |

*Tabla 3. Comparativa de sabores de Ubuntu*

Incluimos, además, una tabla con el consumo de memoria RAM en instalaciones limpias, con el sistema recién iniciado:

|  |  |
| --- | --- |
| **Sabor** | **RAM inicio** |
| Ubuntu | 1,3 GiB |
| Kubuntu | 0,311 GiB |
| Lubuntu | 0,3 GiB |
| Xubuntu | 0,408 GiB |
| U. MATE | 0,84 GiB |
| U. Budgie | 0,985 GiB |

*Tabla 4. Comparativa de consumo de memoria RAM*

Procedemos a desarrollar ahora la comparativa y las conclusiones:

En primer lugar, descartamos Ubuntu Budgie, ya que es el proyecto menos consolidado, además su consumo de recursos es alto, así como el impacto en el usuario procedentes de Windows.

En el caso de Ubuntu MATE, entendemos que su consumo de recursos es demasiado elevado para los efectos visuales que ofrece, ya que está basado en el antiguo GNOME 2.x. Cuenta con un aceptable y útil conjunto de aplicaciones propias y heredada de GNOME. Por otro lado, es de los menos personalizables, con una comunidad e implementación no demasiado amplia y, pese a todo, el impacto para usuarios nuevos es significativo. Sin lugar a dudas es una buena distribución para usuarios GNU/Linux con años de experiencia que buscan este tipo concreto de escritorio clásico, pero entendemos que no es la más adecuada para nuestro proyecto.

Por otro lado, Lubuntu, que hace uso del escritorio LXQt, es la más liviana y sencilla de todas. Está altamente consolidada, es la que menos recursos consume y el impacto para usuarios nuevos no es demasiado acentuado. Sin embargo, apenas es personalizable y la cantidad de herramientas de las que dispone, así como su utilidad, es muy reducida. Es una distribución enfocada a usarse en equipos con muy pocos recursos y/o antiguos, con funcionalidades muy pobres. Por ello, entendemos que este sabor tampoco es el más adecuado para nuestros objetivos, ya que los equipos disponibles en los laboratorios DTE pueden ejecutar sin problemas entornos más potentes.

En cuanto al entorno por defecto de Ubuntu, con GNOME 3.x, como hemos visto, el consumo de recursos es el más alto de todos con diferencia. Goza de alta estabilidad, consolidación, y sus herramientas son muy útiles. Sin embargo, además de alto consumo, en sus últimas versiones la capacidad de personalización se ha ido reduciendo, además su disposición y funcionamiento suponen un impacto considerable para usuarios acostumbrados a Windows. Para concluir, como vimos en el apartado 2.5.2.2, el proyecto GNOME suele tender hacia la innovación y los cambios bruscos entre versiones, lo que ha vuelto a ocurrir en la recién liberada GNOME 40. Teniendo en cuenta todos estos factores consideramos que tampoco es el entorno idóneo para un proyecto enfocado a nuevos usuarios, que persigue el menor impacto posible y que el conocimiento sobre el entorno que se vaya a manejar pueda seguir siendo útil a medio- largo plazo.

Respecto al entorno Xfce 4.14 que incorpora el sabor Xubuntu [142]: como dijimos en el apartado 2.5.2.3, este entorno se caracteriza por la estabilidad, la sencillez y un modesto consumo de recursos. La disposición de los elementos, el contenido de los menús y demás elementos son totalmente personalizable. A pesar de esto, da la sensación, de la misma forma que MATE y Cinnamon, de ser un entorno algo anticuado a nivel visual. Por otro lado, aunque no llega al nivel de simpleza de LXQt, el número de aplicaciones integradas de Xfce, así como sus capacidades, son significativamente más reducidas en comparación a GNOME o KDE Plasma.

Por último, pasamos a analizar Kubuntu, con el entorno de escritorio KDE Plasma 5.18 [143]. A diferencia de lo que ocurría con los escritorios KDE previos a Plasma, que solían ser muy pesados, Plasma 5 es muy liviano, equiparado a Xfce, como pudimos ver en la tabla sobre consumo de memoria RAM anterior. Pese a consumir tan pocos recursos, el apartado visual de este entorno es de los más atractivos, ofreciendo un aspecto moderno y cuidado. Por un lado, el aspecto y comportamiento recién instalado es, de entre todos los que hemos probado, el que más similitudes tiene respecto a la interfaz de Windows 10.

Además, podemos afirmar que es el más personalizable de todos los entornos analizados. No sólo es posible cambiar temas, disposición de los elementos, contenido de los menús, etc. si no que además incorpora una gran cantidad de extensiones y widgets que permiten adaptar el entorno y su comportamiento a las necesidades particulares del usuario. Por otro lado, cuenta con una de las suites de aplicaciones integradas más potente, por lo que hemos podido probar, respecto a los otros entornos.

Tras analizar los diferentes sabores de Ubuntu, nuestra conclusión es que Kubuntu es el más adecuado para este proyecto, ya KDE Plasma es el entorno que tiene un mejor equilibrio entre rendimiento, potencia y belleza. Además, el hecho de que su interfaz por defecto se asemeje al comportamiento de la GUI de Windows 10 es positivo para usuarios cllegados desde dicho sistema, a los que está enfocado el proyecto. Por último, sus aplicaciones integradas destacan por la cantidad de opciones y posibilidades que ofrecen, Existe en torno a 200 aplicaciones oficiales de KDE [85], algunas de ellas incluidas en la imagen de Kubuntu y otras disponibles gratuitamente a través del centro de software integrado. Algunos de los ejemplos más paradigmáticos son: El gestor de ficheros Dolphin [144] ofrece múltiples opciones para la visualización de los ficheros, así como la integración con diferentes protocolos para acceder a contenido remoto, lo cual es útil para nuestro proyecto, como veremos en el punto 3.6.1. También destaca Kate [145], el editor de textos integrado, que ofrece herramientas útiles para la escritura de código en prácticamente cualquier lenguaje de programación, además de complementos como por ejemplo emulador de terminal, depurador GDB o gestor SQL [146]. Esto convierte a Kate en un completo IDE multilenguaje, lo cual, como veremos en el apartado 3.5, es una funcionalidad que nos servirá de gran ayuda a la hora de proponer cambios de SW en la migración de los equipos de la propuesta.

## Herramienta de gestión y automatización utilizada

Una vez elegidos la distribución y el entorno de escritorio que se van a implementar en nuestra propuesta, sólo resta elegir la herramienta más adecuada para automatizar el despliegue de la infraestructura.

En el apartado 2.6.2 presentamos las características principales de las herramientas que se han evaluado. Procedemos a desarrollar el análisis y la decisión tomada en base a ello:

Para empezar, han sido descartadas automáticamente aquellas herramientas que actualmente están descontinuadas: Razor y Spacewalk.

Por otro lado, hemos decidido descartar xCat por contar con una licencia EPL (*Ecliplse Public License*), un tipo de licencia de código abierto, pero totalmente ajena a la filosofía del software libre. Consideramos que, habiendo más alternativas, debemos priorizar aquellas que se acercan más a esta filosofía.

En el caso de Cobbler, la hemos descartado porque no es compatible con Ubuntu.

Nos quedan, por tanto: FAI, Foreman, m23 y MaaS. Vamos a compararlas en detalle, en base a los criterios que consideramos más determinantes:

* Compatibilidad con nuestra distribución y entorno de escritorio.
* Interfaz.
* Curva de aprendizaje.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **FAI** | **Foreman** | **m23** | **MaaS** |
| **Compatibilidad** | Total | Total | Total | Parcial |
| **Interfaz** | CLI | CLI/GUI | GUI | CLI/GUI |
| **Curva de aprendizaje** | Media | Alta | Baja | Media-Alta |

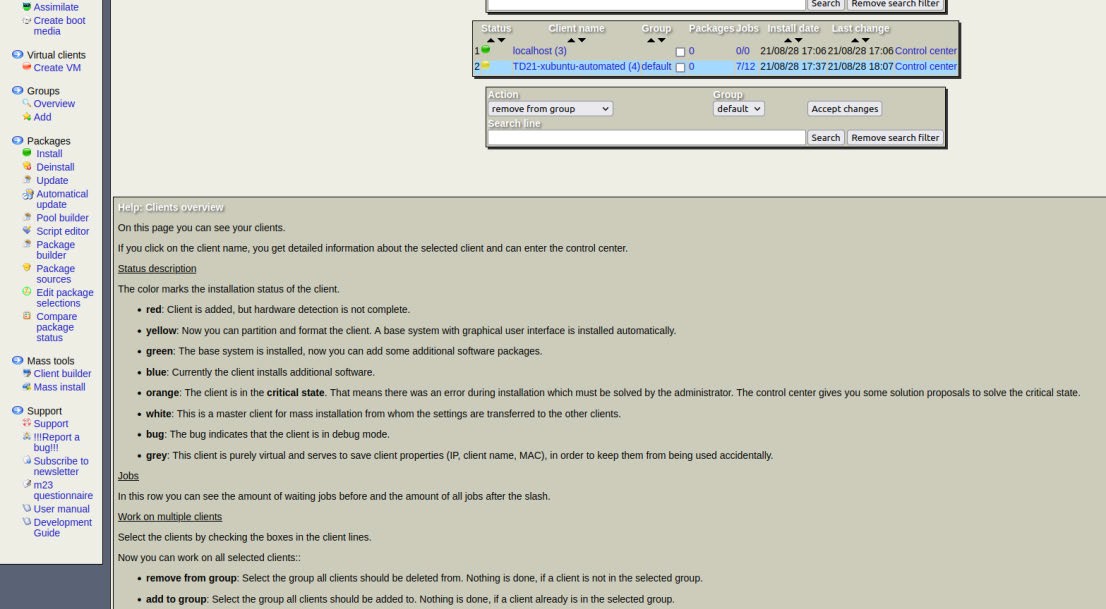
*Tabla 5. Comparativas herramientas de automatización*

Entrando en detalles, en primer lugar, decidimos descartar **MaaS**. Aunque es una herramienta de Cannonical, desarrollador principal de Ubuntu, está pensada para la gestión de servidores y crear arquitecturas que emulen el cloud computing con equipos físicos, de ahí su nombre *Metal as a Service* (Equipo físico como servicio). No permite instalar por defecto entorno de escritorio, solamente imágenes de Ubuntu Server. El resto de opciones que ofrece tampoco están orientadas al control de entornos de las características de los laboratorios DTE.

En cuanto a **Foreman**: es la herramienta más potente de todas las que hemos probado, sus posibilidades de aprovisionamiento tanto de equipos físicos, como de máquinas virtuales y equipos cloud son enormes. Es compatible con prácticamente todas las distribuciones GNU/Linux (aunque está patrocinado y orientado especialmente a la familia Red Hat) y hace uso de herramientas avanzadas de configuración como Puppet y Ansible. Además, ofrece una interfaz gráfica amigable. La contraparte de todas estas características es que su curva de aprendizaje es extremadamente alta y la documentación oficial es extensa y poco clara. Tras analizar la herramienta y realizar algunas pruebas, llegamos a la conclusión de que el nivel dificultad para su puesta a punto y la realización de las tareas que necesitamos es demasiado alta y cargaría de trabajo extra al personal técnico.

En el caso de **FAI** (*Fully Automatic Installation*), es la herramienta más antigua de entre las analizadas, su desarrollo comenzó en 1999. Tiene, por tanto, una importante consolidación. Además, es versátil y adaptable a las necesidades, contando con compatibilidad total para Ubuntu en cualquiera de sus sabores. La puesta a punto es sencilla, sin embargo, carece de interfaz gráfica. En este sentido, toda la configuración acerca de cómo queremos desplegar los entornos se debe hacer a través de comandos y de un sistema de clases con una sintaxis propia. Esto hace que su curva de aprendizaje crezca cuando queremos usar sus características más allá de instalarla y desplegar equipos de prueba con las demostraciones que nos ofrece. Entendemos que para la operativa cotidiana del personal técnico supondría un esfuerzo extra e innecesario tener que trabajar con una herramienta de estas características.

Por último, hemos analizado la herramienta **m23**. En este caso, la herramienta ofrece control únicamente a través de su interfaz web. La puesta a punto, así como su operativa posterior es extremadamente sencilla. Además, ofrece compatibilidad directa con todos los sabores de Ubuntu, así como una serie de herramientas que encontramos de suma utilidad para nuestros objetivos: instalación masiva de equipos con el uso de plantillas, posibilidad de añadir cualquier repositorio de paquetes y seleccionar paquetes o grupos de estos para incluir en tiempo de instalación del sistema, asignación de direcciones IP, integración directa con un servidor LDAP existente, y otras tantas. Todo esto, como hemos comentado, se hace a través de una interfaz web intuitiva y sencilla. La documentación es concisa y, por cada acción que pretendemos realizar podemos encontrar un fragmento resumido de la documentación relacionada incrustado en la página.



*Figura 33. Ejemplo de la interfaz web m23 con información incrustad*

Por todo esto, hemos llegado a la conclusión de que m23 es la herramienta que se adapta a nuestras necesidades concretas de forma más clara, agilizando y simplificando las labores del personal técnico respecto a las otras analizadas y a los procedimientos que tienen establecidos en estos momentos, con la creación de imágenes mediante FOG [147], que requieren una instalación manual por cada equipo HW diferente, y que aportan pocas soluciones a la hora de hacer cambios a mitad de un cuatrimestre, como añadir nuevo SW.

Las cuestiones concretas sobre el uso de la interfaz de m23 y sus diferentes opciones se desarrollan capítulos posteriores.

## Software actualmente instalado en los laboratorios DTE

Como hemos ido comentando a lo largo de la memoria, este proyecto se ha desarrollado bajo la premisa de que pueda ser realmente implementado, por lo que uno de nuestros objetivos principales es estudiar el impacto que tendría el cambio de Microsoft Windows a GNU/Linux respecto al software que hay actualmente instalado en los equipos, necesario para las diferentes asignaturas.

En este sentido, el personal técnico nos ha facilitado la lista de SW que se ha instalado en los equipos de cara al curso 2021/22. Debemos tener en cuenta que esta lista puede sufrir alteraciones en momentos posteriores a la escritura de esta memoria y/o durante el desarrollo del curso. Sin embargo, nuestro objetivo aquí es dar una solución general y aproximada en base a la información de que disponemos.

Con la lista de SW que se nos ha proporcionado, hemos analizado uno por uno los programas, y, a continuación, desarrollamos los resultados, indicando cuáles se encuentran de forma nativa para GNU/Linux (concretamente para Ubuntu 20.04) y cuáles no, en cuyo caso hemos planteado posibles alternativas de forma razonada, siempre dentro de los límites del proyecto.

### Software totalmente disponible

En la siguiente tabla presentamos el SW para el curso 2021/22 que se encuentra disponible para Ubuntu de forma nativa, así como una breve descripción de cada programa:

Estos programas se encuentran disponibles para instalar en Ubuntu con paquetes DEB a través de repositorios oficiales.

### Software parcialmente disponible

A continuación, enumeramos los programas que no cuentan con paquetes nativos para Ubuntu, pero ofrecen alguna solución intermedia, o cuya funcionalidad es totalmente cubierta por otro programa alternativo.

*abla 7. SW de los laboratorios DTE parcialmente disponible para Ubuntu*

### Software no disponible

Por último, pasamos a enumerar los programas que se instalarán en los laboratorios DTE que no están disponibles para GNU/Linux ni cuentan, a priori, con alternativas que puedan cubrir su funcionalidad totalmente. Después, explicaremos las soluciones que proponemos.

*Tabla 8. SW de los laboratorios DTE no disponible para Ubuntu*

El programa XCAL-W, utilizado en la asignatura Redes de Comunicaciones Móviles en Ingeniería Telemática para las prácticas sobre redes 2G, era desarrollado por la empresa Couei, la cual fue adquirida por Huawei hace años. Desde entonces la aplicación dejó de tener soporte, apenas existe información sobre ella en internet. Tras hablar con los profesores de la asignatura, nos dicen que las prácticas sobre este tipo de redes quedan excluidas del plan de estudios a partir del próximo curso, por lo que entendemos que la no disponibilidad del programa para GNU/Linux en estas condiciones no afecta al proyecto.

Las otras dos herramientas que aparecen en la tabla son entornos de desarrollo para microprocesadores, cada una con sus particularidades:

4D Workshop 4 IDE es un software propietario de Intel para la programación de sus propios microprocesadores. Desde la página oficial nos dicen que para ejecutarlo en GNU/Linux o Mac, lo podemos hacer mediante una máquina virtual con sistema huésped Windows [174], cuya instalación y configuración corre a cargo del usuario. A falta de tener información respecto al uso concreto que se hace de este programa en las asignaturas, no podemos afirmar si sería posible hacer uso de otras aplicaciones genéricas para la programación de dichos procesadores. En este sentido, proponemos dos posibles soluciones: el uso de una máquina virtual como dice el fabricante, o, nuestra preferida, el uso de Wine [175] y PlayOnLinux [176].

En el caso de Keil µVision4, es un IDE genérico para programar microprocesadores basados en ARM Cortex. Existen multitud de herramientas y entornos nativos para Linux que permiten programar este tipo de HW, sin embargo, no hemos encontrado ninguna tan genérica, completa e intuitiva como Keil µVision4. Como alternativas concretas planteamos openocd [177] y Eclipse IDE [178] con el plugin Eclipse Embedded CDT [179]. Sin embargo, el uso de alguna de estas alternativas implicaría cambiar de forma significativa los enunciados y manuales de las prácticas. En este sentido, podríamos ejecutar Keil µVision4 en Ubuntu con el uso de Wine y PlayOnLinux, o una máquina virtual con sistema Windows como huésped.

Pasamos a explicar brevemente qué son las herramientas Wine y PlayOnLinux:

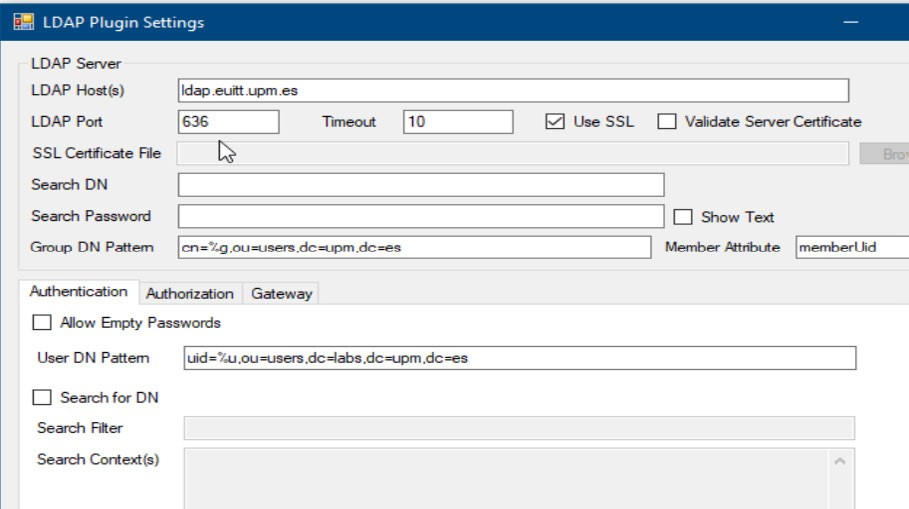
* Wine: se trata de un entorno que permite ejecutar aplicaciones de Windows en sistemas GNU/Linux, proporcionando bibliotecas y demás dependencias que estas necesitan para su ejecución.
* PlayOnLinux: interfaz gráfica que facilita de forma significativa la configuración y ejecución de aplicaciones Windows en GNU/Linux haciendo uso de Wine.

## Análisis de archivos XML. Práctica 3

Para la autenticación del alumnado, profesorado o personal técnico en los equipos de los laboratorios DTE se hace uso del protocolo LDAP [180], que es el protocolo de autenticación centralizada más extendido en el mundo para todo tipo de organizaciones, ya sean empresas, universidades, organismos públicos, etc. En concreto, se usa el cliente LDAP para Windows pGina [181].

Además, la implementación más extendida del protocolo es OpenLDAP, que está desarrollada bajo criterios de software Open source comunitariamente [182].

No es objeto del presente proyecto entrar en detalles sobre las cuestiones técnicas del protocolo. Además, por cuestiones de seguridad, como ya comentamos en el apartado 2.3, no hemos podido acceder directamente a la configuración de los servidores LDAP de la UPM. Contando con la información del personal técnico, que además nos permitieron acceder temporalmente de forma remota a una máquina virtual con Ubuntu 20.04 alojada en un laboratorio DTE, y con la información que es pública por parte de la UPM [183] hemos configurado un servidor LDAP local en una máquina virtual Debian 10 de la forma más aproximada posible a la realidad.



*Figura 34. Configuración actual del cliente LDAP pGina en los laboratorios DTE*

De esta forma, hemos podido instalar el cliente LDAP en los equipos desplegados y probar a autenticarnos contra dicho servidor. Esto no garantiza al 100% que la solución funcione contra los servidores de la UPM en todos los casos, aunque es muy aproximada, y pudimos hacer pruebas simples contra los servidores reales mientras tuvimos acceso a la máquina virtual anteriormente comentada. Luego, la ETSIST cuenta con un proxy LDAP que toma la información del principal, haciendo algunas modificaciones necesarias en los parámetros de los usuarios. Es a este servidor LDAP intermedio al que están conectados los equipos de los laboratorios DTE.

En cualquier caso, una de las características de la herramienta de despliegue que hemos escogido, m23 [122], es la integración con servidores LDAP existentes para configurar automáticamente los equipos cliente que se instalan a través de su interfaz de forma que se puedan autenticar de forma directa contra dichos servidores, por lo que nos ha facilitado mucho el trabajo. Configurando en su interfaz nuestro servidor LDAP de prueba nos hemos podido autenticar sin mayor problema en los clientes instalados a través de la herramienta.

### Acceso al disco de red del alumnado

Uno de los servicios de red a los que tiene acceso el alumnado de la ETSIST es el disco de almacenamiento en red asociado a su cuenta de usuario, de 1 GB de capacidad, alojado en los servidores de la ETSIST/UPM. A esta unidad se puede acceder o bien por SFTP, o bien montándola en el sistema de ficheros, como es el caso de los laboratorios DTE [184].

De nuevo, no hemos podido acceder directamente a la configuración de los servidores para ver cómo está implementado con exactitud este servicio, pero gracias a la información proporcionada por el personal técnico, sabemos que se utiliza el protocolo SMB [185]. Teniendo en cuenta esto, podemos acceder al disco de red desde los clientes que instalemos con Ubuntu haciendo uso de Samba [186], ya que, según la información del personal técnico, el servidor LDAP de la escuela añade a los usuarios todos los parámetros necesarios para acceder mediante esta herramienta, tanto al disco en red como al sistema de impresoras.

## Análisis de archivos JSON. Práctica 4

Por la limitación de acceso a recursos HW, para la realización del proyecto se ha requerido del uso de software de virtualización con el fin de crear máquinas virtuales, tanto en la parte del servidor como en la del cliente, como veremos en el apartado 3.8.

En base a lo analizado en el apartado 2.7, se ha buscado la mejor alternativa para instalar y gestionar las MV necesarias. En este sentido, hemos descartado directamente los hipervisores de tipo 1, ya que es necesario que el HW disponible se utilice para más propósitos además de la virtualización.

Respecto a la virtualización con QEMU/KVM, en los inicios del proyecto fue la opción prioritaria, ya que es la tecnología nativa en los sistemas GNU/Linux y por su forma de funcionar ofrece mejor rendimiento que los hipervisores de tipo 2 estándar.

Sin embargo, volviendo al terreno de la realidad, se ha priorizado que la propuesta sea lo más versátil posible y ahorre esfuerzos al personal técnico en el caso de una implementación real. Por ello, de cara a la elaboración de los manuales del apartado 6, se ha decidido usar VirtualBox, en pos de ser adaptable a la mayor parte de situaciones como, por ejemplo, que para la instalación de los sistemas clientes se disponga únicamente de un equipo con Mirosoft Windows.

De esta forma se garantiza que el despliegue de la propuesta siguiendo los manuales desarrollados, se pueda llevar a cabo indistintamente del tipo de recursos de los que se disponga, ya que VirtualBox se ejecuta indistintamente en sistemas GNU/Linux y Windows, y su operativa es prácticamente idéntica en ambos.

## Configuración del servidor

Para la implementación del servidor m23 hemos utilizado una MV con sistema operativo Debian 10. Esta decisión se ha tomado principalmente porque Debian es la distribución oficialmente soportada por m23, según su documentación oficial [122]. Además, como se explicó en los apartados 2.4.1 y 3.2.2, Debian está orientada a usuarios con algo de experiencia que desean configurar la distribución para sus objetivos particulares, ya que por defecto instala una menor cantidad de SW que, por ejemplo, Ubuntu. En este sentido, hemos instalado Debian 10 solamente con los paquetes básicos del sistema, sin entorno gráfico, lo cual supone una gran ventaja de rendimiento si no es necesario, especialmente cuando el sistema se está ejecutando en una MV.

Tras la instalación del sistema base, hemos realizado las configuraciones oportunas y la instalación del SW m23 mediante su repositorio oficial de paquetes, con la ayuda de una herramienta propia que se ha desarrollado para el proyecto, como veremos en el siguiente apartado.

Si bien el SW m23 se ofrece en otros formatos previamente instalados y configurados [187], como una imagen ISO o una OVA para VirtualBox, hemos preferido hacerlo manualmente en un sistema Debian original, ya que de esta forma se tiene un mayor control sobre el SW instalado y las configuraciones realizadas. Además, pensando en una implementación real, basándonos en esta opción de cara a los manuales de usuario supondrán una mayor versatilidad, ya que el personal técnico podría instalar m23 en un sistema Debian ya instalado para otros propósitos, en lugar de tener una distribución particularizada exclusivamente para m23.

En los manuales de usuario se detallará el proceso de instalación del SW.

## Herramientas propias desarrolladas

Además del uso de distribuciones GNU/Linux y herramientas de despliegue para poner en práctica la propuesta, se han desarrollado algunas herramientas que pretenden, por un lado, facilitar las labores del personal técnico y, por otro, mostrar algunas de los beneficios que GNU/Linux aporta de cara a la administración de sistemas.

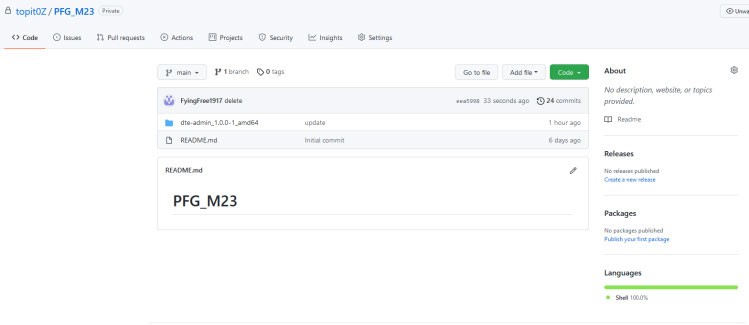
Para facilitar la instalación y configuración del servidor m23, se ha desarrollado un paquete en formato DEB, que a través de scripts de bash permite realizar una serie de tareas de forma sencilla. El nombre de la aplicación es **DTE Admin**.

Se ha decidido usar scripts de bash en lugar de otros lenguajes de programación como Python, Java o C, en primer lugar, para mostrar cómo esta herramienta, que nació con el proyecto GNU en 1988, ofrece una gran cantidad de posibilidades y un desarrollo sencillo, a la par que su ejecución es más rápida que el código compilado de cualquier otro lenguaje. Sin embargo, cabe mencionar que es idóneo para una aplicación como la nuestra, encargada de realizar una serie de operaciones básicas; si se necesitara un programa más complejo y que manejara una gran cantidad de datos, probablemente sería mejor opción un lenguaje de programación como los citados anteriormente, o una combinación de ambos.

Por otro lado, se ha decidido desarrollar el programa con la API Whiptail [188], que permite crear diálogos en la terminal de Linux sin necesidad de servidor gráfico. Hemos tomado esta decisión por diferentes motivos:

* La portabilidad. Con este diseño el programa funcionará haya o GUI y, en el caso de que la haya, funcionará igualmente sea cual sea el entorno. Si hubiéramos decidido desarrollarlo usando diálogos gráficos, como Zenity [189] o kdialog [190], estaríamos atados a las diferentes bibliotecas en las que se basan: GTK+, Qt, etc. lo cual, para el tipo de operaciones que realiza nuestro programa, nos parece innecesario y contraproducente.
* En relación con esto, la implementación de la parte servidora del proyecto se hará con una distribución sin GUI, como veremos más adelante.
* Ejecuciones remotas. A pesar de que existen muchas formas de ejecutar aplicaciones de GNU/Linux de forma remota, suelen requerir de configuración extra y tienden a tener más problemas. Con el diseño del DTE Admin basado en whiptail, se puede ejecutar de forma remota a través de SSH de forma totalmente directa.

El desarrollo del programa se ha hecho con el uso de un repositorio en GitHub [191] mediante el cliente Git [151]:



*Figura 36. Repositorio GitHub utilizado para el desarrollo de DTE Admin*

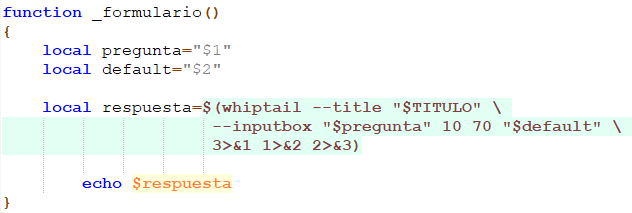
El aspecto de la interfaz de DTE Admin desarrollada con whiptail se puede observar en la siguiente figura:

### Estructura de DTE Admin

La aplicación se ha distribuido en diferentes scripts:

* **DTE-Admin**: implementa el menú principal de la figura anterior, además de realizar algunas operaciones de inicialización, como asegurarse de que se está ejecutando con privilegios de administrador.
* **DTE-Admin-funciones**: implementa las funciones necesarias para realizar las diferentes operaciones que ofrece el programa, así como las funciones auxiliares para obtener parámetros o generar los diálogos whiptail.
* **DTE-Admin-first-launcher**: este script es lanzado de forma automática cuando se instala el paquete. Permite realizar la configuración básica de red de forma directa, además se encarga de hacer las copias de seguridad necesarias de los archivos susceptibles de modificación posterior.
* **DTE-Admin-uninstaller**: script que se ejecuta al desinstalar el programa. Básicamente se encarga de restaurar las copias de seguridad y, con ello, las configuraciones iniciales de la máquina.
* **DTE-Admin-globals**: este script implementa todas las variables globales que son usadas por los otros.

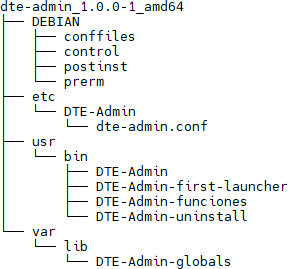
El código de estos scripts estará incluido en el CDROM del proyecto y en el repositorio GitHub creado para el control del desarrollo, disponible en [192]. En la siguiente figura se muestra un fragmento del script DTE-Admin-funciones:



*Figura 38. Fragmento de código de DTE-Admin-funciones.*

*Función para dibujar un formulario con whiptail*

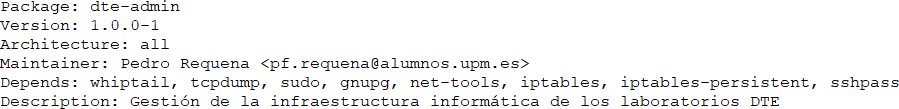
Por otra parte, el paquete a través del cual se instala la aplicación se llama “dte-admin\_1.0.0- 1\_amd64.deb” (por el estándar *nombre\_versión-arquitectura.deb*) [193], ideado para instalarse en Debian (se podría instalar en derivados como Ubuntu sin problema, pero algunas funcionalidades provocarían fallos) y presenta la siguiente estructura previa a su construcción:



*Figura 39. Estructura del paquete dte-admin*

Respecto a los subdirectorios que podemos observar:

* **DEBIAN**: este es un directorio especial que debe existir de forma obligatoria en cualquier paquete DEB, su contenido es el siguiente:
  + **control**: es el fichero de descripción donde se indican diferentes parámetros del paquete, tales como su nombre, versión, arquitectura o dependencias. El contenido se puede ver en la siguiente figura:



*Figura 40. Contenido del fichero control*

* + **conffiles**: en este fichero se listan los archivos de configuración del paquete, que se tratarán de forma especial en una hipotética actualización. En nuestro caso hemos declarado un fichero de configuración, aunque finalmente no le estamos dando uso. El contenido de conffiles tiene la siguiente forma:



*Figura 41. Contenido de conffiles*

* + **postinstall**: se trata de un script que el gestor de paquetes ejecutará de forma automática tras la instalación del paquete. En nuestro caso lo utilizamos para llamar a DTE-Admin-first-launcher, mencionado anteriormente. El contenido del script postinstall es el siguiente:
  + : en este caso, este script se ejecuta de forma automática por el gestor de paquetes en el momento de la desinstalación, previo al borrado de los ficheros. En nuestro caso lo utilizamos para apuntar a DTE-Admin-uninstall, como podemos ver en la siguiente figura:



*Figura 43. Contenido del script prerm*

* En cuanto al resto de directorios que aparecen: **usr/bin**, **var/lib** y **etc** hacen referencia a los directorios del sistema donde se desplegarán los diferentes ficheros de nuestro paquete cuando sea instalado.
* El directorio **/etc/DTE-Admin** albergará el fichero de configuración descrito anteriormente, así como otros archivos que el programa creará en tiempo de ejecución, generalmente para guardar estados asociados a diferentes configuraciones.

En cuanto a la generación del paquete, primero nos situamos por encima del directorio dte- admin\_1.0.1-1\_amd64 que podemos ver en la figura 36:



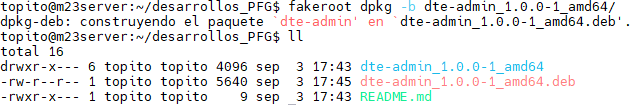
*Figura 44. Directorio superior a la estructura del paquete dte-admin*

Ahora, para construir el paquete, ejecutamos el comando:

$ fakeroot dpkg –b dte-admin\_1.0.0-1\_amd64/

La información sobre los comandos fakeroot y dpkg –b está disponible en [194] y [193], respectivamente.

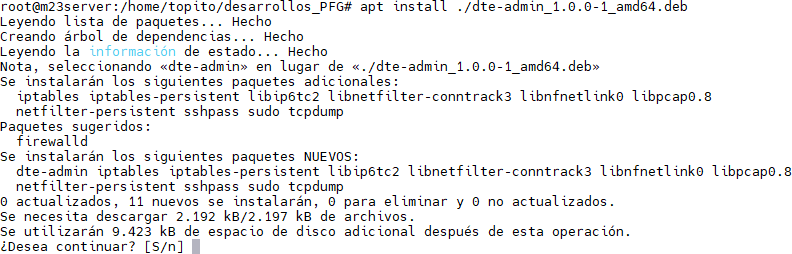
La ejecución del comando anterior nos generará el paquete:



*Figura 45. Directorio con el paquete dte-admin generado*

Dicho paquete se puede instalar con la instrucción:

# apt install ./dte-admin\_1.0.0-1\_amd64.deb



### Funcionalidades de DTE Admin

Una vez desarrollada la estructura del programa, pasamos a explicar sus diferentes opciones, reflejadas en el menú principal de la figura [34].

1. **Instalar m23**: esta opción habilita los repositorios e instala los paquetes necesarios para poner en marcha el servidor m23.
2. **Consultar nombrado de las interfaces de red**: actualmente en Linux existen dos criterios de nombrado para las interfaces de red, uno clásico y otro nuevo incorporado hace pocos años [195]. El primero es totalmente predecible, mientras que el segundo no. Es por esto que para realizar configuraciones de red de forma automática es preferible usar el primer criterio. Esta opción del programa nos muestra qué criterio se está usando:



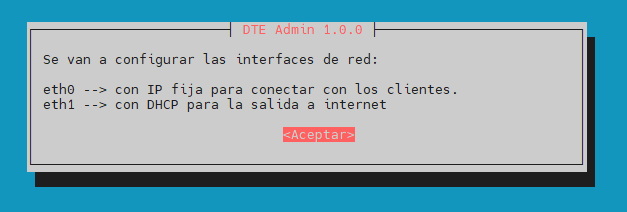
*Figura 47. Opción 2 del DTE Admin*

1. **Cambiar nombrado de las interfaces de red**: en relación con lo expuesto en el punto anterior, esta opción permite cambiar el criterio de nombrado:



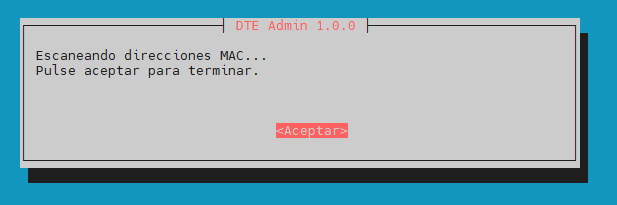
*Figura 48. Opción 3 del DTE Admin*

1. **Configurar red**: esta opción configura de forma automática las interfaces de red en base a nuestras necesidades. En los manuales de usuario se desarrolla eso último.



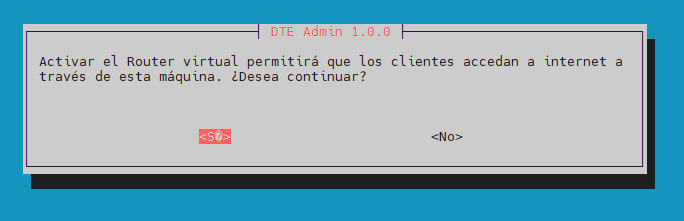
*Figura 49. Opción 4 del DTE Admin*

1. **Escanear direcciones MAC**: esta opción nos permite detectar clientes que traten de arrancar por red en nuestra LAN y guarda sus direcciones MAC en un fichero, lo cual es útil para la operativa con m23, como veremos más adelante. Las direcciones MAC se guardarán por orden de detección.



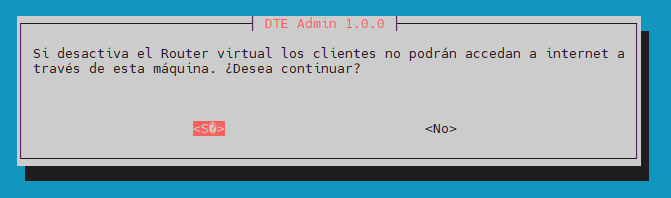
*Figura 50. Opción 5 del DTE Admin*

1. **Activar Router virtual**: nos permite activar el reenvío de paquetes IP [196], de forma que el sistema pueda actuar como un router/gateway. Esto puede ser útil en el caso de que tengamos que dar conectividad a internet a un segmento de red aislado de la del router físico.



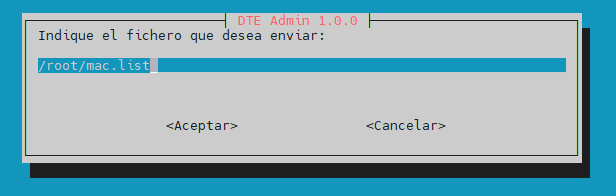
*Figura 51. Opción 6 del DTE Admin*

1. **Desactivar Router virtual**: permite desactivar el reenvío de paquetes comentado en la opción anterior.



*Figura 52. Opción 7 del DTE Admin*

1. **Enviar fichero por scp**: esta opción nos permite enviar ficheros a través de la red a otros equipos que se encuentren en nuestra LAN, mediante el protocolo SCP [197]. Para ello, el programa nos irá pidiendo mediante formularios los parámetros necesarios para el envío. Esta opción puede ser de utilidad para la operativa con m23, como veremos más adelante.



*Figura 53. Opción 8 del DTE Admin*

1. **Activar/Desactivar acceso root por SSH**: para facilitar el manejo del servidor de forma remota, nos permite autenticarnos como root a través de SSH.
2. **Salir**: salir del programa.

## Entorno de desarrollo y pruebas

Tras exponer el conjunto de SW, ajeno y propio, utilizado en el desarrollo del proyecto, pasamos a explicar el entorno sobre el que se ha llevado a cabo dicho desarrollo, así como las pruebas de despliegue y la toma de resultados.

Dividiremos los diferentes elementos del entorno según sean físicos o virtuale:

1. Elementos físicos:

### Ordenador portátil personal Acer E5:

* + - Procesador: Intel i5-7200U de 4 núcleos a 2,50 GHz.
    - Memoria RAM: 16 GB.
    - Almacenamiento: SSD 240 GB y HDD 1 TB.
    - Interfaces de red: 1 Ethernet y 1 WLAN.
    - Sistema operativo: Ubuntu 20.04 con escritorio GNOME 3,36.
    - Funciones: host para virtualización, entorno de desarrollo.

### Ordenador sobremesa ClienteDTE-4

* + - Procesador: Intel Core 2 Quad de 4 núcleos a 2,33 GHz.
    - Memoria RAM: 4 GB.
    - Almacenamiento: HDD de 160 GB.
    - Interfaces de red: 1 Ethernet.
    - Sistema operativo: ninguno.
    - Funciones: Cliente físico para pruebas de despliegue.

### Ordenador sobremesa ClienteDTE-5

* + - Procesador: Intel i5-3470 de 4 núcleos a 3.20 GHz.
    - Memoria RAM: 8 GB.
    - Almacenamiento: HDD de 500 GB.
    - Interfaces de red: 1 Ethernet.
    - Sistema operativo: ninguno.
    - Funciones: Cliente físico para pruebas de despliegue.
* **Switch Ethernet** TP-Link TL-SF1008D de 8 puertos.
  + Función: interconectar los diferentes elementos dentro de la LAN.
* **Conmuntador KVM VGA/USB** TCNEWL de 4 puertos.
  + Función: facilitar el manejo de los diferentes ordenadores físicos.

1. Elementos virtuales:

### Máquina virtual m23-server

* + - Procesador: 1 núcleo.
    - Memoria RAM: 4 GB.
    - Almacenamiento: HDD virtual de 20 GB.
    - Interfaces de red: 1 Ethernet en modo NAT y 1 Ethernet en modo Bridge.
    - Sistema operativo: Debian 10.
    - Funciones: Servidor HTTP, TFTP, PXE, LDAP y m23.

### Máquina virtual ClienteDTE-1

* + - Procesador: 1 núcleo.
    - Memoria RAM: 2 GB.
    - Almacenamiento: HDD virtual de 20 GB.
    - Interfaces de red: 1 Ethernet en modo Bridge.
    - Sistema operativo: ninguno.
    - Funciones: Cliente virtual para pruebas de despliegue.

### Máquina virtual ClienteDTE-2

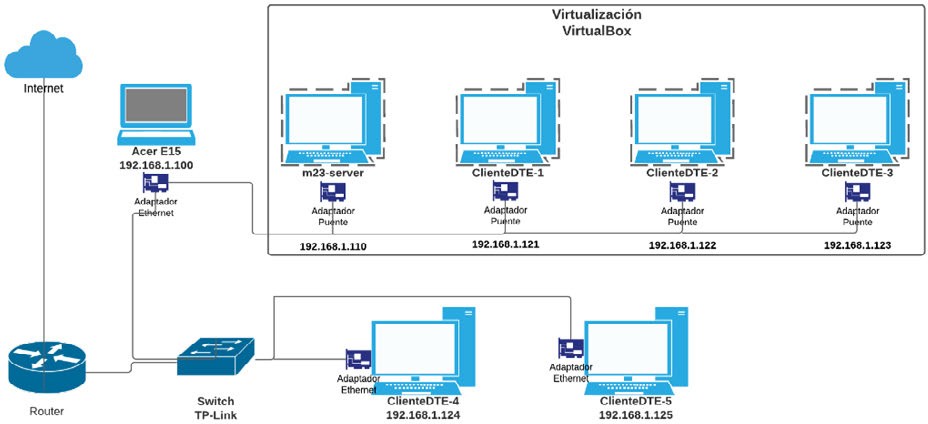
* + - Procesador: 1 núcleo.
    - Memoria RAM: 2 GB.
    - Almacenamiento: HDD virtual de 20 GB.
    - Interfaces de red: 1 Ethernet en modo Bridge.
    - Sistema operativo: ninguno.
    - Funciones: Cliente virtual para pruebas de despliegue.

### Máquina virtual ClienteDTE-3

* + - Procesador: 1 núcleo.
    - Memoria RAM: 2 GB.
    - Almacenamiento: HDD virtual de 20 GB.
    - Interfaces de red: 1 Ethernet en modo Bridge.
    - Sistema operativo: ninguno.
    - Funciones: Cliente virtual para pruebas de despliegue.

Sobre la topología de red utilizada, se ha tratado de asemejarse lo máximo posible a la que existe en los laboratorios DTE. Según nos ha explicado el personal técnico, de forma general, todos los equipos de los laboratorios pertenecen a un mismo segmento de red privado, y acceden a internet a través de NAT [198] de forma transparente.

Para el desarrollo del proyecto, se ha dispuesto de una red doméstica con acceso a internet a través de un router de fibra óptica convencional, por medio de NAT. Dicho router se ha conectado por cable ethernet al switch, a través del cual hemos conectado el ordenador portátil y los dos equipos de sobremesa. A su vez, las máquinas virtuales han sido conectadas a esta misma LAN a través del adaptador puente de VirtualBox [199]. En la siguiente figura se puede observar un diagrama de la arquitectura de red explicada:



# PRUEBAS DE DESPLIEGUE

Pasamos ahora a explicar los diferentes modos que nos proporciona la herramienta m23 para desplegar los sistemas en el entorno anteriormente descrito, los principales problemas que nos hemos encontrado y cómo les hemos ido dando solución hasta obtener la metodología que mejor se ha adaptado a nuestras necesidades y que está resumida en el apartado 6 de esta memoria.

En un principio, la línea de trabajo pensada era la de probar primero exclusivamente en entornos virtuales y, tras ser exitoso, pasar a los equipos físicos. Finalmente, debido a la imposibilidad de mantener todas las máquinas virtuales encendidas a la vez, se ha optado por un modelo mixto, trabajando con clientes virtuales y físicos de forma simultánea.

Cabe mencionar, antes de nada, que disponíamos originalmente de tres equipos prestados por el personal de los laboratorios, pero que uno de ellos producía un fallo en la ejecución del script de análisis de hardware de m23. Se probó a reinstalar el servidor y a usar una tarjeta Ethernet PCI para este cliente, pero en ningún caso conseguimos que funcionara. Sin embargo, pudimos instalar sin problemas un sistema Debian 10 de forma manual. Dado que buscando en los foros de soporte de m23 no encontramos ningún bug ni nada relacionado, y que es un problema que sólo ocurría en este hardware, decidimos extraerlo del entorno de pruebas.

Lo primero a tener en cuenta es que m23, según su manual de usuario [200], tiene dos modos generales de operativa: la instalación de clientes individuales y la instalación de clientes de forma masiva.

Para las primeras pruebas se ha utilizado la opción individual con las configuraciones por defecto, de cara a poder familiarizarnos con el manejo del entorno y observar su comportamiento, utilizando tanto las máquinas virtuales como los equipos físicos.

La instalación individual de clientes consta de dos pasos:

1. Se da de alta el cliente en la interfaz web, rellenando todos los campos del formulario que se puede observar en la figura 55. Para poder acceder a él, el servidor necesita que indiquemos la dirección MAC del cliente de forma manual. Una vez dado de alta, el cliente arranca por red y el servidor realiza un reconocimiento de hardware. El cliente pasa a estar en estado preparado.
2. Ahora indicamos que queremos instalar el cliente, seleccionando la distribución GNU/Linux, el entorno de escritorio y los grupos de paquetes, que dejamos por defecto.

Esta es una lista con las principales ventajas que observamos en m23 en las primeras pruebas:

* La interfaz es sencilla y las opciones muy concretas. Con unos conocimientos técnicos básicos se puede comenzar a instalar clientes de forma individual en muy poco tiempo.
* Toda la gestión de las herramientas auxiliares como DHCP, TFTP y PXE se realiza de forma automática y transparente para el usuario.
* Dispone de múltiples opciones que a priori encontramos útiles, como la agrupación de clientes en grupos, la creación de fuentes y listas de paquetes personalizadas, la elección de forma nativa de diferentes versiones de Debian y Ubuntu con todos los entornos de escritorio disponibles o la integración directa con servidores LDAP existentes.

Sin embargo, cuando pasamos al modo de instalación masiva, que será el realmente útil para la propuesta, comienzan a surgir los primeros problemas que hay que resolver.

En resumen, el proceso de instalación masiva de clientes se basa en dos pasos:

1. Se crea una plantilla genérica de cliente, similar a la del modo individual, pero omitiendo algunos campos. A continuación, creamos directamente la plantilla para el particionado del disco duro y elegimos la distribución y la selección de paquetes.
2. Después, partiendo de dicha plantilla creamos los clientes concretos. Para ello, debemos completar sus parámetros, y m23 nos da la opción de mantener lo introducido en la plantilla, introducirlos a mano para cada cliente o generarlos.

Tras hacer unas pruebas y leer detenidamente el manual [200] nos encontramos con el principal problema, que no habíamos contemplado hasta ahora:

M23 no proporciona ningún método de auto descubrimiento de clientes, es decir, nos exige introducir sus direcciones MAC manualmente. Esto es asumible en nuestro entorno de pruebas, con un total de 5 clientes, pero no lo es para el caso real de los laboratorios DTE, donde hay más de 100 equipos.

En este sentido, entendemos que es fundamental ofrecer una solución propia a este problema. A pesar de que m23 no implementa el auto descubrimiento, sí permite subir ficheros de texto plano con los parámetros de todos los clientes que necesitamos. Es por ello que decidimos desarrollar un script de BASH que, haciendo uso de tcpdump [164], hace una escucha de nuestra red interna en busca de equipos que hagan peticiones DHCP en tiempo de arranque BIOS, y que de los paquetes capturados extraiga las direcciones MAC para almacenarlas en un fichero de texto. Los comandos de este script inicial se pueden observar en la siguiente figura:

Tras probar el script con todos los clientes y con diferentes tipos de casuísticas vemos que los resultados son satisfactorios ya que suele ser preciso. Sin embargo, observamos que, en ciertas ocasiones, si arrancamos dos clientes simultáneamente, sólo se captura la petición DHCP de uno de ellos. Es por ello que, de cara a la metodología desarrollada en el capítulo 6, deberemos dejar un tiempo de margen de entre 5 y 10 segundos entre el arranque de cada cliente. Cumpliendo este margen, los resultados son exitosos en el 100% de los casos.

Otra cuestión asociada a esto, aunque de menor gravedad, era la de que el acceso a la web de m23 se hacía con el host, mientras que el servidor se alojaba en una máquina virtual sin entorno gráfico. Esto conllevaba tener que mandar el fichero de resultados con las MAC continuamente por SCP al host. Para facilitar esta tarea, de nuevo decidimos implementar un script de BASH que hiciera el envío de forma automática. Es de aquí de donde surge la idea del DTE-Admin, como un programa unificado que permitiera hacer diferentes tareas repetitivas de una forma rápida y semi automatizada.

Siguiendo esta metodología, la instalación de clientes de forma masiva ha sido, por norma general, exitosa. Sin embargo, mientras que vamos usando la herramienta m23 nos percatamos de algunos otros problemas que debemos solventar:

* No todo el SW planteado en el apartado 3.5 dispone de paquetes deb en repositorios apt, si no que debemos hacer uso de paquetes Snap, y m23 no ofrece soporte para ellos.
* A pesar de que es sencillo crear nuestras propias listas de repositorios de paquetes, en la documentación oficial no aparece ningún método para comprobar las claves públicas de estos.

Estos dos problemas nos llevan a profundizar en el funcionamiento interno de la herramienta que, resumiendo, se basa en scripts PHP que se descargan en los clientes y ejecutan los scripts de BASH que llevan insertos, a través de diferentes funciones. A pesar de que existe una API en la web de m23 [201], no queda muy claro cómo utilizar estos scripts para configuraciones particulares que no ofrece la aplicación.

Por suerte, en el manual de usuario [200] encontramos que existe una sección que permite agregar scripts PHP personalizados y cargarlos en los clientes, tratándolos como paquetes especiales. Teniendo en cuenta esto, y ayudándonos de los scripts existentes para comprender mejor la lógica, desarrollamos dos scripts PHP para ejecutar de forma automatizada en los clientes la instalación de paquetes Snap y las claves públicas de los repositorios. Dichos scripts se muestran en las dos siguientes figuras:

Una vez subidos a la aplicación, los añadimos a la lista de paquetes junto al resto de SW seleccionado en base a lo expuesto en el apartado 2.5. Tras esto, hubo que ajustar sus prioridades respecto al resto de paquetes, de forma que la firma de los repositorios fuera previa al resto, y la instalación de los paquetes Snap, posterior a la instalación del propio Snap. A pesar de que conseguimos ajustarlo y conseguir resultados exitosos en la mayoría de los casos, en aproximadamente una de cada 6 instalaciones, los scripts no se ejecutan en el orden correcto y la instalación de paquetes no acaba con los resultados esperados.

Por último, otro problema al que nos enfrentamos era el de poder realizar pruebas de autenticación contra los servidores LDAP de la ETSIST. Durante unas semanas, gracias al personal técnico de los laboratorios, tuvimos acceso remoto a una máquina virtual con Kubuntu situada en la red de los laboratorios. Aunque por entonces el proyecto aún no estaba maduro, pudimos configurar manualmente un cliente LDAP en la máquina virtual que se autenticaba correctamente contra el servidor. Tomando la configuración de dicho cliente, creamos un servidor LDAP que simulara al de la escuela, dando de alta un usuario con mis propias credenciales, y otro inventado.

Por un lado, la integración de este servidor con los clientes a través de m23 fue inmediata, permitiendo crear en los clientes un estado similar al que tienen en los laboratorios DTE actualmente, con un usuario local administrador y con el resto de usuarios autenticándose a través de LDAP. Sin embargo, debido a la falta de información por cuestiones de seguridad institucional, no hemos podido simular la configuración de la unidad de almacenamiento compartida en red del alumnado, quedando esto para las líneas futuras del proyecto.

Teniendo todo esto en cuenta, se ha podido estandarizar un proceso para el despliegue masivo de clientes cubriendo la mayoría de necesidades previstas.

En el próximo capítulo analizaremos algunos datos tomados en todo el proceso de pruebas, así como una comparativa de rendimiento de los equipos con Windows y Kubuntu.

# RESULTADOS

Pasamos en este capítulo a mostrar algunos datos significativos de las pruebas realizadas, así como su interpretación. En primer lugar, detallaremos los tiempos de instalación de los clientes en base a diferentes criterios. Después, pasaremos a comparar el rendimiento de uno de los clientes con el SO original Windows 10 y el propuesto en el proyecto, Kubuntu 20.04.

## Tiempos de instalación

En este apartado se detallan los datos de tiempo obtenidos en los diferentes casos de despliegue que se han realizad. Dichos datos se muestran en forma de gráficos, y van acompañados de su correspondiente interpretación.

Para todas las medidas se han utilizado los equipos descritos en el capítulo 3.10 con el modo de instalación masiva descrito en el capítulo 4.

Además, los resultados de tiempo de instalación se muestran en función de la utilización, o no, de AptCatcherNg [202]. Esta herramienta permite establecer un proxy de paquetes, que los descargará y almacenará de forma local en la primera petición, y posteriormente los servirá a los clientes que se los soliciten. Como hemos visto, m23 permite configurar esta opción de forma directa cuando damos de alta nuevos clientes, convirtiéndose el propio servidor m23 en el proxy apt. Sin embargo, no ofrece un procedimiento estándar para eliminar la configuración de los clientes una vez instalados, por lo que, si tratáramos de acceder a los repositorios de paquetes posteriormente con el servidor desconectado, se produciría un error. En este sentido, hemos tratado de analizar cuantitativamente la ventaja en tiempo con la utilización del proxy, para tomar una decisión acerca de si merece la pena, o no usarlo, aunque esto dependerá en última instancia del caso de uso.



*Figura 59. Formulario para la configuración del proxy de paquetes en m23*

En adelante, se utilizará la siguiente nomenclatura cuando proceda:

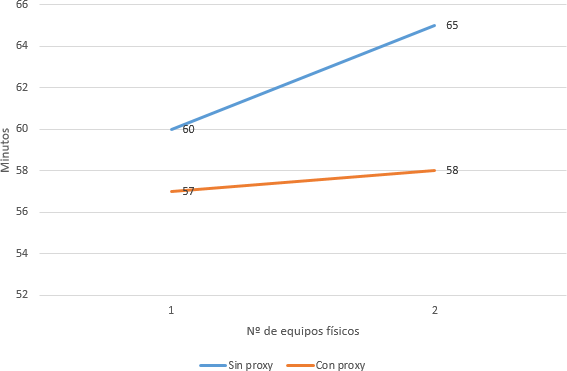
* Caso 1: instalación de 1 máquina virtual.
* Caso 2: instalación de 2 máquinas virtuales.
* Caso 3: instalación de 3 máquinas virtuales.
* Caso 4: instalación de 1 máquina física.
* Caso 5: instalación de 2 máquinas físicas.
* Caso 6: instalación de 2 máquinas físicas y 1 máquina virtual
* Caso 7: instalación de 2 máquinas físicas y 2 máquina virtual
* Caso 8: instalación de 2 máquinas físicas y 3 máquina virtual

En primer lugar, mostramos los tiempos de instalación obtenidos únicamente en máquinas virtuales:

Como se puede observar, la tendencia en el incremento de tiempo de instalación, según se añaden más clientes virtuales, es cercana a la exponencial.

* Este resultado es lógico, ya que el servidor debe atender en paralelo cada vez más peticiones y procesar más instrucciones. Por otro lado, recordemos que tanto los clientes como el servidor se ejecutan sobre un mismo ordenador anfitrión, de forma que la limitación total de recursos HW aumenta con el número de clientes.
* En cuanto a la ventaja del tiempo con el uso del proxy, se obtiene un valor de 6,3 minutos de diferencia. El aumento de tiempo con el número de clientes es prácticamente el mismo con y sin proxy.

Ahora, mostramos los resultados de instalación únicamente en equipos físicos. Recordemos que únicamente disponemos de dos equipos físicos funcionales, por lo que las muestras de datos están muy limitadas.



*Figura 61.Tiempos medios de instalación en equipos físicos*

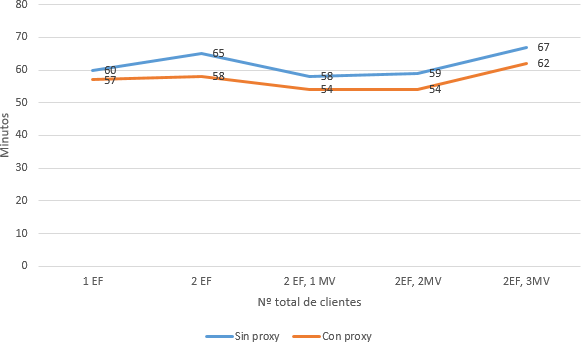
Observando este gráfico, podemos llegar a las siguientes conclusiones:

* El tiempo de instalación en equipos físicos es significativamente superior al registrado en máquinas virtuales.
* La diferencia de tiempos entre uno y dos equipos es ligeramente inferior al caso anterior, aunque al carecer de una tercera muestra no podemos confirmar esta tendencia.
* En este caso, la ventaja con el uso de proxy es, de media, 5,5 minutos, ligeramente menor que en el caso anterior, aunque, como se puede observar en el gráfico, la diferencia aumenta de forma más pronunciada que en el caso de máquinas virtuales. Una vez más no podemos confirmar esta tendencia al carecer de una tercera muestra.

Las explicaciones que damos a estas diferencias entre ambos casos son las siguientes:

* El hecho de que las máquinas virtuales se ejecuten sobre el mismo equipo anfitrión reduce los tiempos en el envío y recepción de paquetes, además de provocar menor congestión a nivel físico del switch ethernet. Todo esto provoca, por un lado, que el tiempo de instalación sea menor en máquinas virtuales. Por otro, que la ventaja con el uso del proxy sea mayor también máquinas virtuales, ya que los paquetes que necesitan descargar para la instalación se encuentran de forma física en el mismo equipo, reduciéndose los tiempos de petición y descarga al mínimo posible.
* Las características HW del ordenador portátil anfitrión son significativamente superiores a las de los ordenadores de sobremesa. Esto hace que las operaciones en las máquinas virtuales se realicen más rápidamente, influyendo en el tiempo de instalación.

Por últimos, mostramos los datos generales, partiendo de la instalación de un único equipo físico, al que después se le añade el segundo y, a continuación, cada una de las máquinas virtuales, llegando al total de cinco equipos instalándose a la vez.



*Figura 62. Tiempos medios de instalación totales*

Explicamos los valores obtenidos en este gráfico ya que, a priori, pueden parecer incoherentes:

El hecho de que al sumar un tercer equipo (virtual) se reduzca el tiempo medio de instalación se debe a que, como hemos visto anteriormente, el tiempo general de instalación es inferior en máquinas virtuales que en los ordenadores físicos y esto hace que baje la media. Sin embargo, si tomamos las muestras del caso 8 y las dividimos entre máquinas reales y virtuales, llegamos a unos resultados similares a los obtenidos en los casos 3 y 5:

En este sentido, las conclusiones finales de los datos obtenidos son las siguientes:

* No se puede obtener una conclusión exacta a partir de los datos, ya que provienen de equipos con diferentes características HW y de diferente naturaleza (virtual y física).
* En las máquinas virtuales la tendencia del incremento de tiempo al introducir nuevos equipos es más acentuada. Esto se debe a que los recursos HW son los mismos a repartir entre cada vez más máquinas.
* Observamos, en general, una media de 60 minutos de instalación en los equipos físicos. Sería necesario estudiar este punto añadiendo más equipos y ver si el tiempo crece significativamente o se estabiliza, y de esta forma establecer los criterios de instalaciones simultáneas más ventajosos desde el punto de vista del tiempo medio necesario.
* El uso del proxy de paquetes aporta una ventaja media de 5 minutos de tiempo. La recomendación concreta sobre su uso o no, dependerá de las características del equipo que juegue el rol de servidor, así como de la capacidad de la red, tanto a nivel interno como en el enlace a internet.

## Comparativa Windows y Kubuntu

Pasamos ahora a comparar el rendimiento de Windows y Kubuntu. Para ello, hemos utilizado el HW que más se asemeja al que existe actualmente en los laboratorios DTE. Se trata del ordenador de sobremesa ClienteDTE-5 descrito en el apartado 3.10, cuyas características principales recordamos:

* Procesador: Intel i5-3470 de 4 núcleos a 3.20 GHz
* Memoria RAM: 8 GB
* Almacenamiento: HDD de 500 GB
* Interfaces de red: 1 Ethernet

Para la comparativa se han usado las siguientes versiones de sistema operativo:

* Windows 10 Pro 20H2
* Kubuntu 20.04.3

En primer lugar, hemos tomado algunos datos simples en cuanto al consumo de memoria RAM y almacenamiento en disco con los sistemas operativos vírgenes, sin el SW propio del DTE añadido, tras el primer arranque después de la instalación. Para las dos primeras medidas, se han realizado la media de valores de consumo tras 5 minutos con la sesión de usuario iniciada en reposo:

|  |  |  |
| --- | --- | --- |
| *Sistema virgen* | **Windows 10** | **Kubuntu 20.04** |
| **Espacio libre en disco** | 402 GiB | 427 GiB |
| **Consumo medio de RAM** | 1,9 GiB | 492 MiB |

*Tabla 9. Comparativa básica Windows y Kubuntu en sistema virgen*

En segundo lugar, hemos repetido las pruebas añadiendo el SW extra del DTE. Tras reiniciar el equipo estos son los resultados:

|  |  |  |
| --- | --- | --- |
| *Sistema final* | **Windows 10** | **Kubuntu 20.04** |
| **Espacio libre en disco** | 392 GiB | 419 GiB |
| **Consumo medio de RAM** | 1,96 GiB | 517 MiB |

*Tabla 10. Comparativa básica Windows y Kubuntu en sistema final*

Además de esto, hemos querido comprobar algo de lo que nos había informado el personal técnico del DTE y que aparece en el documento [183], que es la limitación de 20 caracteres en el nombre de usuario que existe en Windows (contando sólo la parte de la ID previa a @alumnos.upm.es), lo que obliga a, como se vio en el apartado 3.6, configurar el servidor LDAP de la ETSIST y el cliente de los equipos para que genere nombres recortados compatibles en los casos en que se exceden los 20 caracteres.

Para ello, hemos dado de alta en nuestro servidor LDAP un usuario con el ID [nombrealumno.apellidoalumno@alumnos.upm.es,](mailto:nombrealumno.apellidoalumno@alumnos.upm.es) que excede los 20 caracteres, y haciendo una comprobación rápida con el ClienteDTE-5 instalado, hemos podido autenticarnos sin problema:

Finalmente, para obtener datos más concretos acerca del rendimiento de los sistemas, hemos utilizado la herramienta *Phoronix Test Suite* [203], que permite ejecutar una gran cantidad de pruebas y realizar comparaciones a través de la web gratuita *openbenchmarking* con nuestros resultados y con los del resto de la comunidad [204].

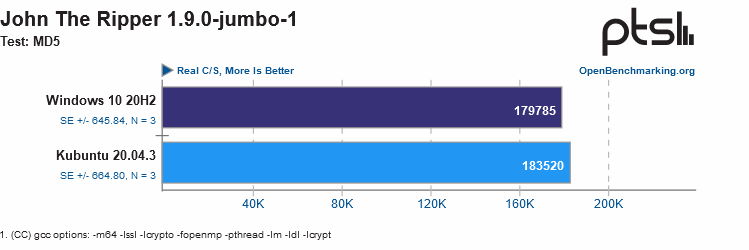
Aunque a día de hoy la herramienta está disponible para los principales sistemas operativos, muchos de sus tests sólo son compatibles con GNU/Linux. Esto nos ha limitado a la hora de realizar las pruebas, concretamente las relacionadas con el modelado CAD [205], la simulación de circuitos electrónicos [206] y la simulación de redes LTE [207], ya que dichas pruebas no son compatibles con sistemas Windows.

Por lo tanto, hemos escogido algunas pruebas básicas asegurándonos que era compatibles para ambos sistemas. Pasamos a describirlas y mostrar los resultados:

1. Prueba *DaCapo Benchmark* [208]: este test analiza el rendimiento de la CPU mediante la ejecución de la máquina virtual de Java.

Los resultados nos muestran la cantidad de milisegundos que necesita el sistema para realizar una serie de instrucciones sobre la máquina virtual de Java. Como vemos, Kubuntu es notablemente más rápido en este aspecto.

1. Prueba *John The Ripper* [209]: Analiza el rendimiento del sistema utilizando dicha herramienta, que es un conocido motor de *cracking* de contraseñas escrito en C, por lo que también nos sirve para comprobar el comportamiento del sistema ejecutando programas escritos en dicho lenguaje.



*Figura 65. Resultados prueba John The Ripper*

En este caso, el resultado que muestra es en *cracking per second*, es decir, cálculos de cifrado o *hash* de contraseñas por segundo [210]. De nuevo, aunque de forma algo más ajustada que en el caso anterior, Kubuntu vuelve a obtener un mejor rendimiento en la prueba.

1. Prueba *7-Zip Compression* [211]: como su nombre indica, se encarga de medir el rendimiento en base a la compresión de ficheros con el método 7-Zip.

Nos muestra los resultados en el formato millones de instrucciones por segundo. Como vemos, aunque muy ajustado, el rendimiento de Kubuntu es superior al de Windows 10.

1. Prueba *LAME MP3 Encoding* [212]: este test analiza el tiempo necesario para convertir y codificuar un fichero de sonido WAV en MP3.

Como podemos observar, mientras que en Windows esta tarea conlleva unos 13 segundos, en Kubuntu se lleva a cabo en 11.

1. Prueba *Git* [213]: En este caso para comprobar el rendimiento del sistema se calcula el tiempo medio necesario para hacer un conjunto de operaciones básicas de Git.

Como se puede observar en la gráfica, mientras que Windows necesita un tiempo medio de 91,84 segundos para levar a cabo la prueba, a Kubuntu le basta con 67,47. En este sentido, el sistema obtiene un rendimiento significativamente mejor con Kubuntu.

### Conclusiones de la comparativa

Cabe mencionar, antes de nada, que estas pruebas presentan ciertas limitaciones, por lo que hemos mencionado anteriormente sobre la compatibilidad de los tests Phoronix, además de porque el SW instalado en ambos sistemas operativos no es idéntico, en referencia al apartado

3.5 de la memoria.

Por otro lado, se han descartado pruebas de otra naturaleza, como el procesado de gráficos en 3D, ya que no es un tipo de operación que se realice en los laboratorios DTE. En ese punto en concreto, tras revisar diferentes análisis, Windows suele obtener mejores resultados que GNU/Linux, ya que, como hemos comentado en capítulos anteriores, los controladores de las tarjetas gráficas que realizan los fabricantes suelen estar mejor optimizados para el primero.

En cualquier caso, analizando los datos tanto de la comparativa simple como de la avanzada podemos afirmar que con el SO Kubuntu se obtiene un mayor rendimiento generalizado del sistema que con Windows, además de un mejor aprovechamiento de los recursos HW.

En este sentido, se confirma una de las hipótesis principales del proyecto.

# MANUALES

En este apartado vamos a desarrollar los manuales necesarios para la instalación del servidor, el despliegue y posterior mantenimiento de los clientes.

## Instalación de JavaCC

Se puede utilizar JavaCC directamente desde la línea de comandos, o puede optar por utilizar un IDE, que permite desarrollar proyectos con rapidez [1] e integra varias herramientas como para desarrollar proyectos de forma más eficiente y productiva.

### Descarga e Instalación de JavaCC desde línea de comandos

**Descarga**

Descargue la última versión estable (al menos el código fuente y los binarios). Actualmente la versión estable más reciente es la 7.0.13, por lo que la descarga del binario se accedería a través de:

<https://repo1.maven.org/maven2/net/java/dev/javacc/javacc/7.0.13/javacc-7.0.13.jar>

La descarga del código fuente se realiza a través del repositorio oficial de Java CC:

[https://github.com/javacc/javacc/releases](https://github.com/javacc/javacc/releases%20)

**Instalación**

Una vez que haya descargado los archivos, navegue hasta el directorio de descarga y descomprima el archivo fuente, creando así el llamado directorio de instalación de JavaCC:

$ unzip javacc-7.0.13.zip

o

$ tar xvf javacc-7.0.13.tar.gz

A continuación, mueva el archivo binario bajo el directorio de descarga en un nuevo directorio bajo el directorio de instalación y cámbiele el nombre a

javacc-7.0.13.jar target/javacc.jar

Posteriormente, añada el directorio del directorio de instalación de JavaCC a su archivo . Los scripts/ejecutables de invocación de JavaCC, JJTree y JJDoc residen en este directorio scripts/PATH

En los sistemas basados en UNIX, es posible que los scripts no se puedan ejecutar inmediatamente. Esto se puede resolver usando el comando del directorio: javacc-7.0.13/

chmod +x scripts/javacc

### Descarga e Instalación de JavaCC desde un IDE

Para poder usar JavaCC en un IDE se necesita como mínimo que el IDE tenga soporte para Java, y soporte para Maven con Java. IDEs como IntelliJ o Eclipse son compatibles con JavaCC mediante la instalación de un complemento para su desarrollo.

* Descarga de IntelliJ: <https://www.jetbrains.com/idea/>
* Plugin IntelliJ JavaCC: <https://plugins.jetbrains.com/plugin/11431-javacc/>
* Descarga de Eclipse: <https://www.eclipse.org/ide/>
* Plugin Eclipse JavaCC: <https://marketplace.eclipse.org/content/javacc-eclipse-plug>

Para Maven, hay que añadir la siguiente dependencia al archivo pom.xml:

<dependency>

<groupId>net.java.dev.javacc</groupId>

<artifactId>javacc</artifactId>

<version>7.0.13</version>

</dependency>

En el caso de utilizar un IDE, deberá descargarse el [binario](#DescargaBinarioJavaCC), al igual que si desease desarrollar JavaCC desde la línea de comandos.

### Configuración de JavaCC en Eclipse

Una vez instalado JavaCC, para poder desarrollar proyectos utilizando la JavaCC en Eclipse hay que seguir los siguientes pasos:

1. Crear un proyecto nuevo o elegir uno existente.
2. Abrir las propiedades del proyecto (Alt+Enter) (⌘+I)
3. Ir a *JavaCC* > *Global Options*, y en “*Set the default JavaCC jar file*”, poner la ruta al binario descargado anteriormente (javacc.jar)

IMPORTANTE: Si mueve el binario descargado a la ruta por defecto, no es necesario realizar el paso 3. Es recomendable poner el binario en la ruta por defecto ya que, de lo contrario, para cada proyecto que quiera desarrollar va a tener que realizar este procedimiento. La ruta por defecto se indica debajo de “*Set the default JavaCC jar file*” del paso 3, *(default: plugin’s jar)*.

Si la configuración se ha realizado correctamente, a la hora de guardar los archivos JavaCC, se generarán la compilación de las clases Java correspondientes.

Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza media

Figura 3. Compilación de Archivos JavaCC en Eclipse

l servidor m23 se instalará sobre la distribución Debian 10.10 ya que, en el momento de escribir este manual, m23 no está soportado para la versión 11, que es la más reciente.



*Figura 69. Aviso en la página de m23 sobre las versiones de Debian soportadas [214]*

La descarga de la imagen de Debian 10.10 está disponible en su página oficial: <https://www.debian.org/releases/buster/debian-installer/> [215]

Para la instalación del sistema operativo se pueden seguir los manuales oficiales, disponibles en: [216]

En el caso de querer instalar el servidor directamente en un equipo físico, habrá que seguir el procedimiento indicado en los puntos 2.4 y 5 del manual oficial para crear el medio de instalación y arrancar el instalador.

En nuestro caso, vamos a realizar la instalación sobre una máquina virtual de VirtualBox. Si la aplicación no se encuentra instalada en el sistema anfitrión, se puede obtener en: <https://www.virtualbox.org/wiki/Downloads> [217] para diferentes sistemas operativos, con la documentación relacionada para su instalación en cada caso.

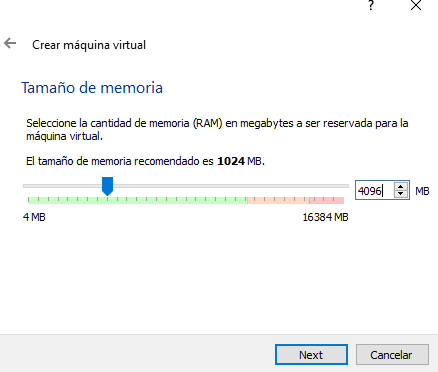
Una vez instalado VirtualBox, crearemos una máquina virtual de la siguiente forma:

**IMPORTANTE**: La instalación normal de Debian requiere una conexión a internet.

1. Abrir la aplicación VirtualBox.
2. En el menú superior pulsamos en: **Máquina ** **Nueva**. Rellenamos el formulario con los siguientes parámetros. (La carpeta de la máquina puede ser cambiada a otra que deseemos).

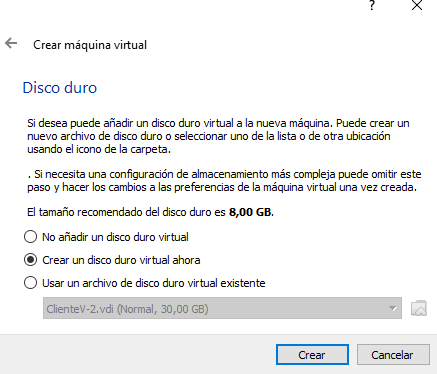
Pulsamos **Next**.

1. En tamaño de memoria, recomendamos un mínimo de 4096 MB para una memoria en el host de 8GB. Pulsamos **Next**.



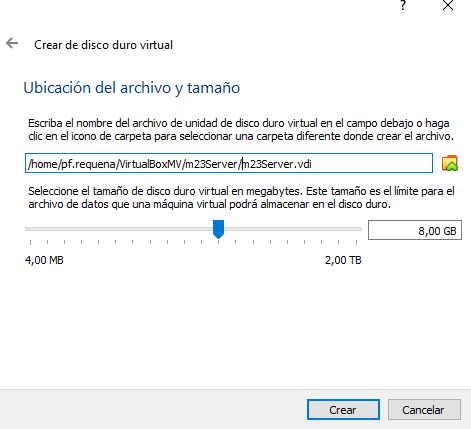
*Figura 71. Creación de MV con VirtualBox (2)*

1. Seleccionamos **Crear un disco virtual ahora** y pulsamos **Crear**.



*Figura 72. Creación de MV con VirtualBox (3)*

1. Elegimos formato **VDI** y pulsamos **Next**.
2. Elegimos **Reservado dinámicamente** y pulsamos **Next**.
3. Elegimos un tamaño aproximado de 20 GB y pulsamos **Crear**.



*Figura 73. Creación de MV con VirtualBox (4)*

Ahora nuestra máquina virtual se habrá creado, aparecerá en la columna de la izquierda.

1. Hacemos Click derecho sobre ella y elegimos **Configuación**.
2. En la nueva ventana, elegimos **Sistema** y la pestaña **Procesador**. Asignamos 1 núcleo en el caso de que nuestro host tenga 2, 2 núcleos en el caso de 4 en el host, o 4 núcleos si el host tuviera 8. Pulsamos “Aceptar”
3. Con la máquina creada, seguimos el manual oficial de instalación para completar el proceso según las necesidades. Para este ejemplo recomendamos lo siguiente en cuento al software a instalar:
   * No instalar ningún entorno de escritorio.
   * Marcar solamente **utilidades estándar del sistema** y **Servidor SSH**

### Instalación del dte-admin

Una vez instalado el sistema operativo, procedemos a instalar y configurar el SW necesario. Para ello, en primer lugar, instalaremos nuestro paquete dte-admin.

1. Accedemos a la máquina virtual y accedemos con el usuario “root”
2. Una vez dentro, descargamos el paquete dte-admin con el comando:

# wget [https://github.com/topit0Z/PFG\_M23/raw/main/dte-admin\_1.0.0-](https://github.com/topit0Z/PFG_M23/raw/main/dte-admin_1.0.0-1_amd64.deb) [1\_amd64.deb](https://github.com/topit0Z/PFG_M23/raw/main/dte-admin_1.0.0-1_amd64.deb)

1. Actualizamos la base de datos y los paquetes del sistema con: # apt update && apt dist-upgrade
2. Cuando termine, instalamos el dte-admin con el comando # apt install ./dte- admin\_1.0.1-1\_amd64.deb –y
3. De esta forma se nos instalará el paquete y todas sus dependencias. Nos aparecerán dos diálogos del paquete iptables. Pulsamos Intro en ambos con la opción por defecto.
4. Cuando la instalación esté llegando a su fin, aparecerán los diálogos de configuración del DTE Admin.
   1. En primer lugar, se nos mostrará un diálogo que nos permite cambiar el criterio de nombrado de las interfaces de red, de Normal (enpXsY, enoX, etc.) a Clásico (ethX). Este cambio es necesario para posteriores configuraciones, por lo que elegimos Sí.

Nos saldrá un segundo diálogo de confirmación. Elegimos Sí.

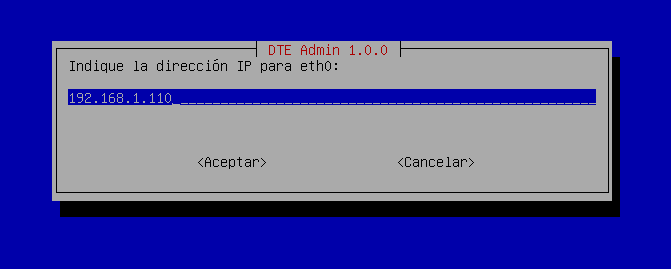


*Figura 76. Diálogo DTE-Admin. Nombrado de interfaces 1*

1. Una vez se apliquen los cambios, pulsamos Aceptar. Ahora nos preguntará si queremos configurar la red. Respondemos que Sí.
2. Después, nos preguntará si queremos aplicar la configuración por defecto.

Respondemos que Sí.

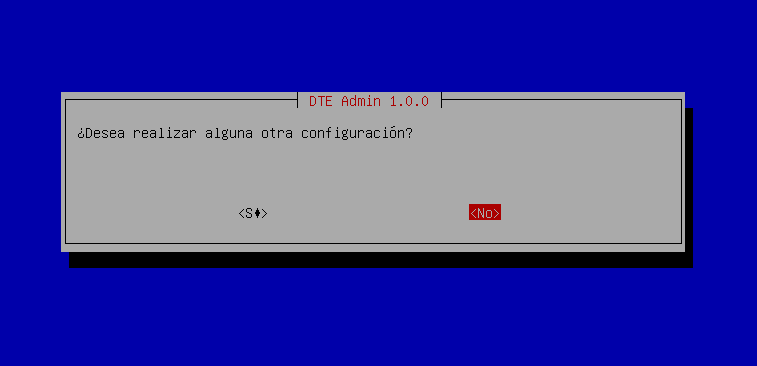
Nos aparecerán tres diálogos preguntándonos por la dirección IP, la máscara de subred y el Gateway. Debemos asegurarnos de que estos valores sean correctos en base a la configuración de nuestra LAN. Para este ejemplo, se dejarán los valores por defecto.



*Figura 78. Diálogo DTE-Admin. Configuración de red 2*

1. Tras esto, nos preguntará si deseamos hacer alguna configuración de red adicional.

Respondemos que No



*Figura 79. Diálogo DTE-Admin. Configuración de red 3*

1. Nos preguntará ahora si deseamos activar el acceso root por SSH. En nuestro caso, como estamos haciendo la instalación en una máquina virtual, vamos a responder que sí para una mayor comodidad posterior. No es aconsejable si la instalación se está realizando en un equipo real. Nos aparecerá una ventana de confirmación advirtiendo de los riesgos. Volvemos a decir que Sí.
2. Cuando se apliquen los cambios, pulsamos Aceptar.
3. Un diálogo nos informará de que se está finalizando con la instalación y que el equipo se reiniciará de forma automática:

### Instalación de m23

Una vez se ha reiniciado la máquina, si hemos configurado la red, podremos acceder por SSH, que es más cómodo que la consola nativa de VirtualBox. Para ello seguimos los siguientes pasos:

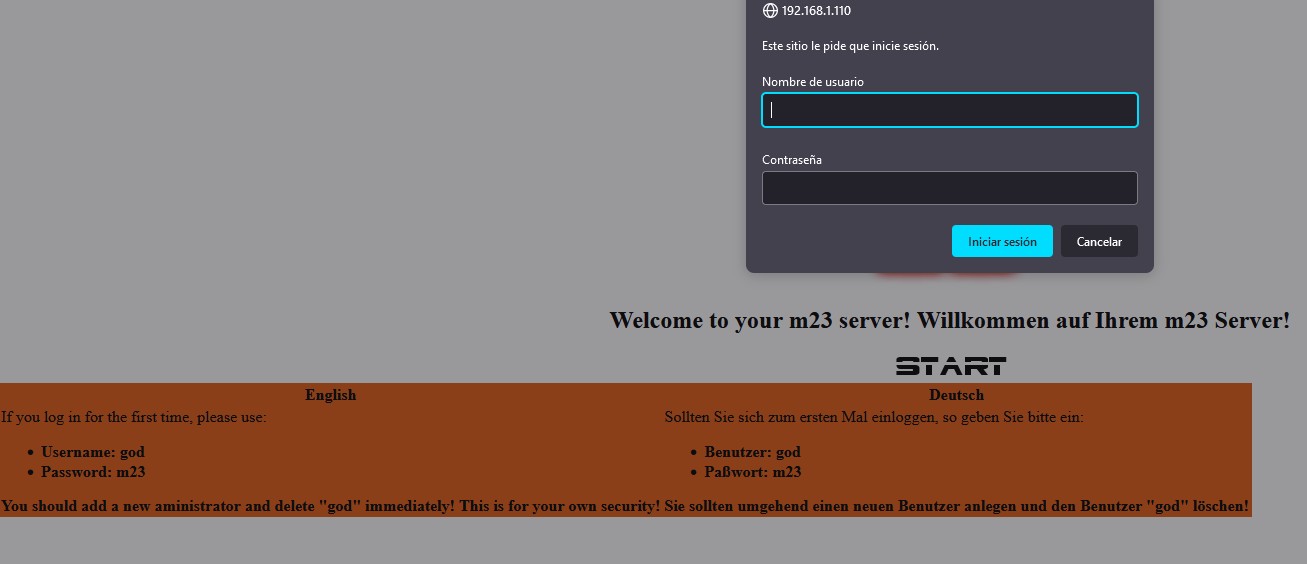
1. Modificamos en VirtualBox la interfaz de red. Vamos al menú superior “Máquina”  “Configuración”  “Red”. En el adaptador 1, cambiamos el campo “Conectado a” de “NAT” a “Adaptador puente”. En “nombre” elegimos el adaptador de red del host conectado a la subred donde se encuentran los clientes. Desplegamos el menú “Avanzadas” y en modo promiscuo seleccionamos “Permitir todo” y pulsamos “Aceptar”. La configuración debería quedar similar a como se muestra en la siguiente figura:
2. Ahora, para acceder por SSH elegimos el método que prefiramos. Para hosts con Windows se recomienda el programa MobaXterm. Para GNU/Linux se puede hacer a través de la CLI con el uso del comando ssh o con los gestores de ficheros de GNOME o KDE.
3. Nos conectamos a la máquina por SSH con el usuario **root** y ejecutamos el comando

**DTE-Admin**. Se nos mostrará el siguiente menú:

1. Con la opción “1) Instalar M23” seleccionada, pulsamos el cursor derecho del teclado para posicionarnos en aceptar y pulsamos Intro. El proceso de instalación de m23 comenzará. Si nos mostrara fallo, volvemos ejecutarlo una segunda vez. Comenzará el proceso de instalación de m23 y sus dependencias.
2. En los diferentes diálogos que nos irán apareciendo, pulsamos Intro en cada uno para introducir la opción por defecto. La instalación continuará durante unos minutos.
3. Cuando finalice la instalación, el dte-admin nos avisará y volveremos a su menú principal:
4. Con el cursor del teclado bajamos hasta la opción “9) Salir” y pulsamos Aceptar

## Instalación de los clientes

Lo primero que debemos hacer es acceder a la interfaz web de m23. Si nuestro servidor cuenta con entorno de escritorio, podemos hacerlo directamente a través de él. Si no, como es el caso de este ejemplo, accedemos desde el host sobre el que se está ejecutando VirtualBox. Para acceder debemos abrir el navegador web y escribir la URL **“https://<ip\_servidor\_m23>**. Nos aparecerá la ventana de bienvenida y nos pedirá las credenciales:



*Figura 88. Página de bienvenida de m23*

Como podemos ver, el usuario por defecto es “god” y la contraseña es “m23”. Introducimos la información en el formulario e iniciamos sesión.

Lo primero que haremos será cambiar las credenciales por defecto por cuestiones de seguridad:

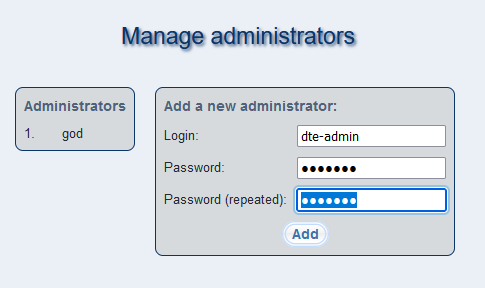
1. Pulsamos *Server settings* en el menú de la izquierda.
2. Pulsamos ahora en **Manage administrators.**



*Figura 90. Página "Server settings" de m23*

* 1. Ahora debemos añadir un nuevo usuario administrador que sustituya al genérico. Para este ejemplo se creará el administrador **dte-admin** y la contraseña será **12dte34**.

Pulsamos **Add**.



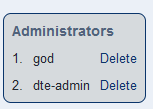
*Figura 91. Gestión de administradores de m23*

* 1. Hecho esto, debemos cerrar sesión mediante el botón **Log out** del menú superior.



*Figura 92. Botón de cerrar sesión de m23*

* 1. Iniciamos sesión de nuevo, con el usuario **dte-admin**.
  2. Ahora volvemos a la configuración de administradores y borramos el administrador genérico **god**. Pulsando **delete** junto a su nombre. En el mensaje de confirmación decimos **YES**.



*Figura 93. Lista de administradores m23*

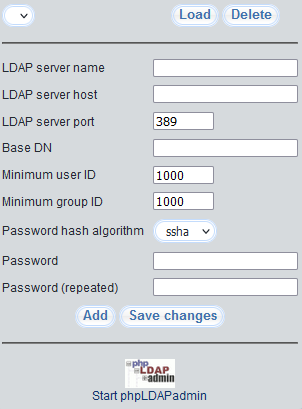
El siguiente paso es asociar el servidor LDAP de nuestra red con m23. Para ello, dentro de

**Server settings ** **Components**, pulsamos el botón LDAP.



*Figura 94. Apartado "Components" de m23*

Se nos abrirá la siguiente ventana:



*Figura 95. Formulario para añadir servidor LDAP en m23*

Debemos rellenar el formulario en base a los parámetros del servidor LDAP de nuestra red. En este ejemplo, como se comentó en el apartado 3.6, se ha usado un servidor OpenLDAP virtualizado con una configuración similar al de la UPM. Si no tuviéramos acceso completo al servidor, dejamos en blanco los campos de contraseña para añadirlo en modo sólo lectura. Tras introducir los campos pulsamos **Add**. Si todo ha ido bien, nos mostrará lo siguiente:

**NOTA**: Como se puede observar, m23 incorpora el *framework* phpLDAPadmin para la gestión de servidores LDAP, lo que puede ser muy útil, aunque escapa al alcance de este proyecto. Para más información consultar el manual de usuario de m23 [200].

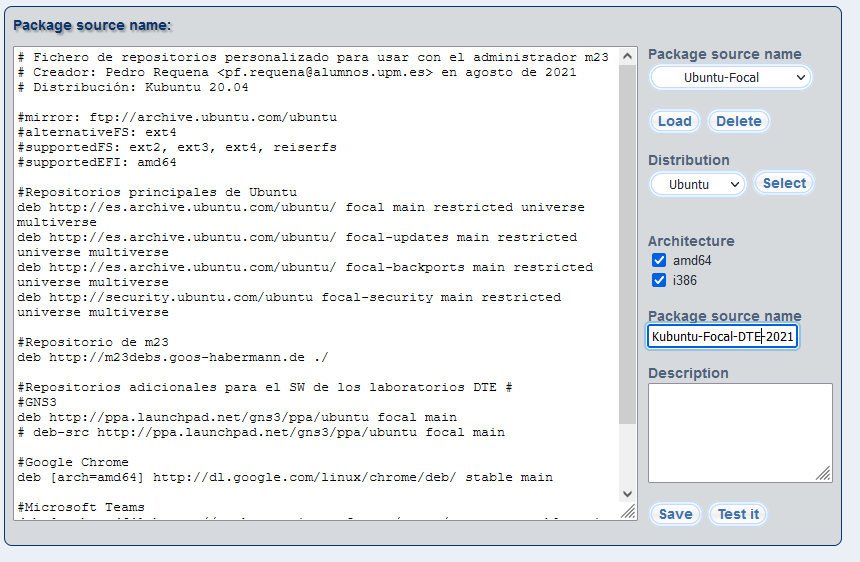
### Configuración de las fuentes de paquetes

Ahora, procederemos a configurar las fuentes del SW que queremos instalar en los clientes:

1- En primer lugar, vamos a copiar el contenido de la lista personalizada con los repositorios de paquetes. El fichero con la lista de repositorios se encuentra disponible en el CDROM y en:

<https://raw.githubusercontent.com/topit0Z/PFG_M23/main/recursos_m23/sources_m23.txt> 2- En el menú de la izquierda de m23, pulsamos en **Package sources***.*

*S*e nos abrirá una ventana donde debemos copiar el contenido de dicho fichero. En **Distribution** indicamos Ubuntu, y en **Package source name** escribimos **Kubuntu- Focal-DTE-2021**. Y pulsamos **Save***.*



*Figura 98. Editor de fuentes de paquetes de m23*

**NOTA**: Esta lista de repositorios se ha hecho en base a la información descrita en el apartado

3.5. En siguientes cursos académicos, este fichero debería revisarse para adaptarse a posibles nuevas listas de SW.

1. Una vez pulsado **Save** se nos desplegarán las opciones **Supported user interfaces**. Debemos marcar **UbuntuKubuntu2004** siguiendo nuestra propuesta. Si se quisiera usar otro entorno, habría que seleccionar el correspondiente. Una vez marcada, pulsamos de nuevo **Save**.
2. Una vez configuradas las fuentes, debemos añadir sus claves GPG públicas para comprobar la validez de los repositorios externos en tiempo de instalación, de lo contrario no podremos instalar algunos paquetes. Como en la documentación oficial de m23 no nos indica cómo realizar este procedimiento, lo haremos a través de un script de PHP que nos permite importar en el administrador y asignar como un paquete más.

NOTA: Los repositorios ofrecen sus claves públicas por diferentes medios, pero, para facilitar el trabajo del personal técnico, se han recopilado los enlaces necesarios y se ha creado el script anteriormente comentado, que instalará las firmas en todos los clientes. Dicho script está disponible en el CDROM y en:

[https://raw.githubusercontent.com/topit0Z/PFG\_M23/main/recursos\_m23/ImportarGPGRepo](https://raw.githubusercontent.com/topit0Z/PFG_M23/main/recursos_m23/ImportarGPGRepos.php) [s.php](https://raw.githubusercontent.com/topit0Z/PFG_M23/main/recursos_m23/ImportarGPGRepos.php)

En el menú izquierdo, en la sección **Packages***,* pulsamos en **Script editor**. Se nos abrirá un editor de texto donde debemos copiar el contenido del script **ImportarGPGREpos.php**. En el formulario **Save as** indicamos su nombre y pulsamos **Save***.*

*Figura 100. Editor de scripts PHP de m23 1*

**NOTA**: Al igual que con el fichero de repositorios, las claves GPG deben actualizarse adecuadamente cuando cambien las fuentes de software. El script PHP simplemente se encarga de traducir a BASH lo que está dentro de la función *echo*, por lo que, si queremos añadir nuevas claves, simplemente copiamos y pegamos el comando que empieza con **curl** y sustituimos la URL con el archivo de clave pública correspondiente.

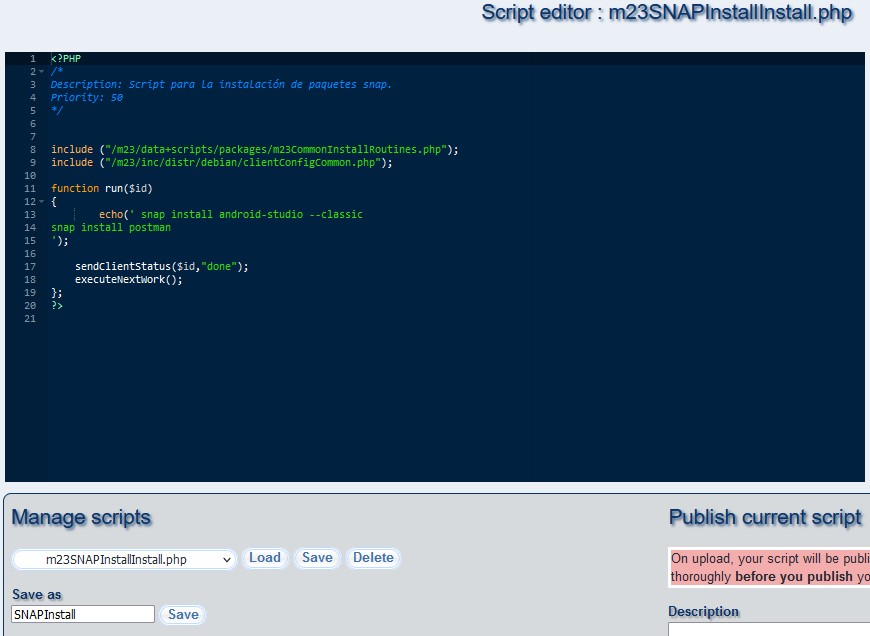
### Software en paquetes Snap

Aunque vamos a poder cubrir la mayoría de SW del apartado 3.5 haciendo uso del método estándar de paquetes deb, en algunos casos no es posible.

Estos casos se subdividen en dos: los que están disponibles por paquete Snap [140] y los que no.

Los primeros son, concretamente: Android Studio y Postman. Ya que m23 no incorpora de forma nativa la gestión de estos paquetes, hemos creado el script PHP SNAPInstall.php que se puede añadir al servidor para que lo ejecute en los clientes y, de esta forma, instale dichos paquetes mediante Snap.

Dicho script está disponible en el CDROM y en: <https://raw.githubusercontent.com/topit0Z/PFG_M23/main/recursos_m23/SNAPInstall.sh> El procedimiento para subirlo es el mismo que el descrito en el punto anterior.



*Figura 101. Editor de scripts PHP de m23 2*

De igual forma que en el caso anterior, este script simplemente ejecuta por línea de comandos BASH lo que se encuentra dentro de la función *echo*. Si necesitáramos instalar más paquetes Snap en el futuro, tendríamos que añadir más líneas con **snap install <paquete>**. Una vez guardado el script, vamos al siguiente paso.

### Seleccionar la lista de paquetes para instalar

De todo el SW que se debe instalar para el próximo curso, al margen de los paquetes necesarios que sabemos con certeza que están incluidos por defecto en nuestra distribución, debemos indicar manualmente su instalación. La forma de realizar esta operación es la siguiente:

1. En el menú de la izquierda, en el apartado **Packages**, pulsamos **Edit package selections**.
2. Se nos abrirá una nueva ventana. Seleccionamos como distribución Ubuntu y como fuente de paquetes la que hemos creado. Pulsamos **Select**.



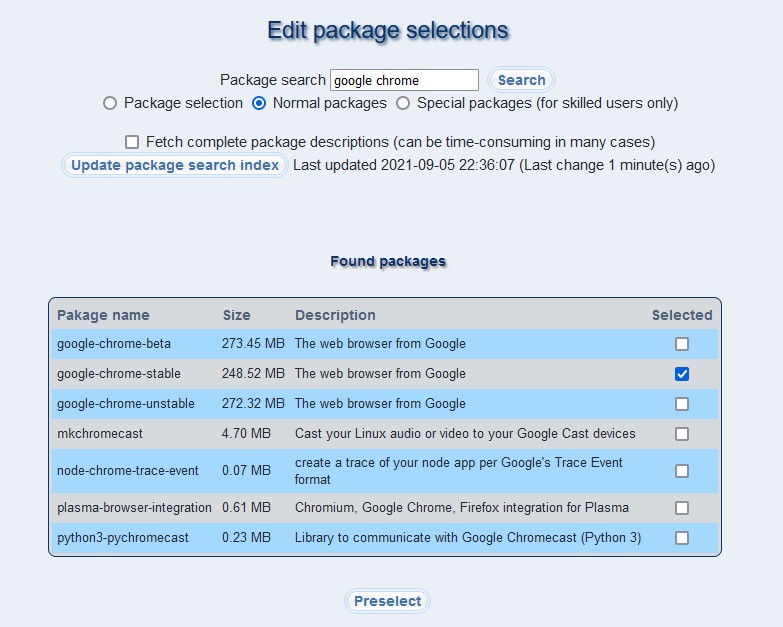
*Figura 103. Elección de la distribución y la fuente de paquetes para construir la nueva lista*

1. Pulsamos ahora en **Update package search index** para actualizar la base de datos.



*Figura 104. Botón para actualizar el índice de paquetes de m23*

1. Ahora debemos introducir en el buscador, uno a uno, los paquetes que deseamos añadir a la lista para su posterior instalación. Cuando se cargue la lista de candidatos, marcamos el, o los, paquetes necesarios, y pulsamos el botón **preselect**, como se puede ver en el siguiente ejemplo:



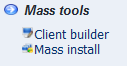
*Figura 105. Resultado de la búsqueda de paquetes*

1. Repetimos este proceso hasta tener en la lista de preseleccionados todos los paquetes SW que queremos añadir.
2. Ahora es el momento de importar los scripts PHP que creamos previamente. Para ello, seleccionamos **Special packages** en el buscador, y buscamos los scripts por su nombre. Como con los paquetes, los seleccionamos y pulsamos **Selected** para añadirlos a la lista.
3. En este momento tenemos la lista completa de SW y scripts:
4. Antes de finalizar, debemos ajustar la prioridad de los scripts. En m23 la prioridad se asigna con números naturales, siendo el más prioritario el 0. Seleccionamos el script m23ImportarGPGRepos y en el cuadro de abajo indicamos prioridad 24, de forma que se ejecute antes que la instalación de los paquetes, para que los repositorios estén firmados. Pulsamos **Change priority**
5. Ahora seleccionamos el script m23SNAPInstall y le cambiamos la prioridad a 26, para asegurarnos de que se ejecute una vez que el programa Snap esté instalado. Pulsamos **Change priority** de nuevo.
6. Por último, en el diálogo inferior guardamos la selección de paquetes, asignándole un nombre, en nuestro caso **Paquetes-DTE-Kubuntu-2021**, y pulsamos **Save**.
7. Como vemos, nos da la posibilidad de **exportar la lista de paquetes como un fichero de texto**, así como importar una liste desde un fichero. La lista que hemos creado para este ejemplo la hemos exportado con el nombre “m23PackageList” y está disponible en el CDROM y en [https://raw.githubusercontent.com/topit0Z/PFG\_M23/main/recursos\_m23/m23Packag](https://raw.githubusercontent.com/topit0Z/PFG_M23/main/recursos_m23/m23PackageList.txt) [eList.txt](https://raw.githubusercontent.com/topit0Z/PFG_M23/main/recursos_m23/m23PackageList.txt) de forma que el personal técnico puede utilizarla como base para crear futuras listas, importándola.

### Construcción de la plantilla del cliente

Una vez tenemos listo el SW, pasamos a crear las especificaciones que servirán para la instalación automática de los clientes.

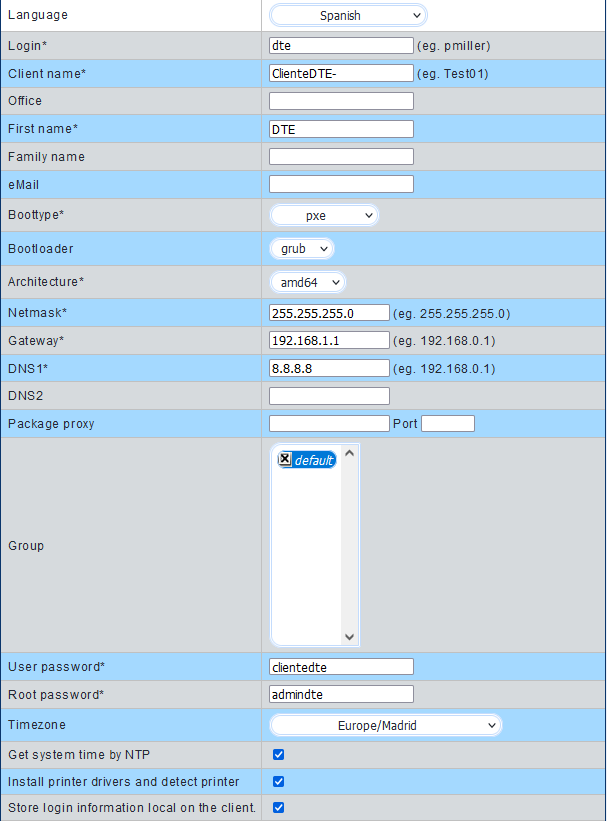
1. En el menú de la izquierda, en el apartado **Mass tools**, pulsamos en **Client builder**.



*Figura 110. Menú principal m23. Sección Mass tools*

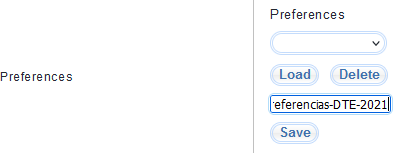
1. Se nos abrirá una ventana para rellenar una serie de datos sobre el cliente. En la siguiente figura se muestra la plantilla realizada para este ejemplo. Adaptar los datos según las necesidades del entorno, basándose en ella:

**NOTA:** Para este ejemplo, no hemos activado el proxy de paquetes. Si se desea activar, introducir la dirección IP del servidor M23 y el puerto 2323.



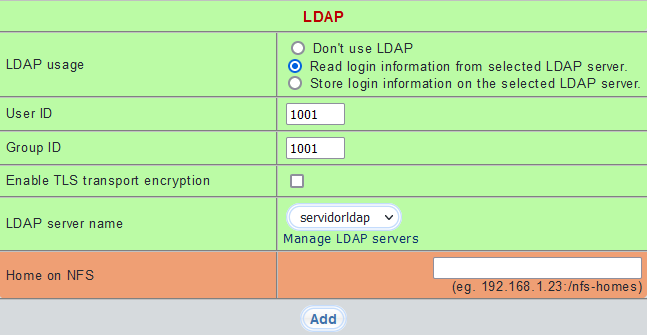
*Figura 111. Plantilla básica para nuevos clientes*

1. En la parte superior se nos permite guardar estas opciones para importarlas en otro momento. Es recomendable hacerlo. Indicamos un nombre de referencia y pulsamos en **Save**.



*Figura 112. Opción para guardar preferencias*

En la parte de abajo veremos un cajón relacionado con LDAP. Lo primero que debemos hacer es pulsar en **Manage LDAP servers**. Lo dejaremos de la siguiente forma, en base a nuestro ejemplo. Ajustar para cada entorno concreto:



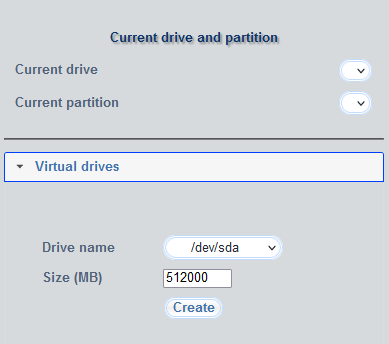
*Figura 113. Configuración LDAP en los clientes*

De esta forma, lo que estamos haciendo es crear un usuario local en los clientes con el nombre **dte**, y estamos indicando que el resto de usuarios se autenticarán mediante el servidor LDAP. Es importante desmarcar la casilla **Enable TLS transport encryptation**, a no ser que tengamos la certeza de que nuestro servidor LDAP funciona a través de **ldaps** [218].

1. Por último, pulsamos **Define**.

Se nos abrirá ahora una nueva ventana para crear las plantillas de particionado de disco duro. Como podemos ver en [200], hay múltiples opciones para el particionado. Para nuestro ejemplo hemos optado por el caso más sencillo: un único disco duro con dos particiones, una para todo el sistema y la otra para *swap*. El proceso a seguir es el siguiente:

1. Creamos una nueva unidad /dev/sda. Para el tamaño hemos seleccionado 51200 MB (500 GB), que el tamaño que tiene nuestro equipo físico de referencia, aunque tras las diferentes pruebas realizadas hemos visto que el instalador se adaptará al tamaño real de disco de cada uno de los clientes. Introducimos los datos y pulsamos **Create**.



*Figura 114. Configurador de particiones de m23 1*

1. Ahora seleccionamos la opción **Partition scheme** y elegimos la opción comentada anteriormente. Pulsamos en **Execute scheme**.
2. Por último, en la parte inferior pulsamos en **Finalise the partitioning and formating and choosen distribution**.

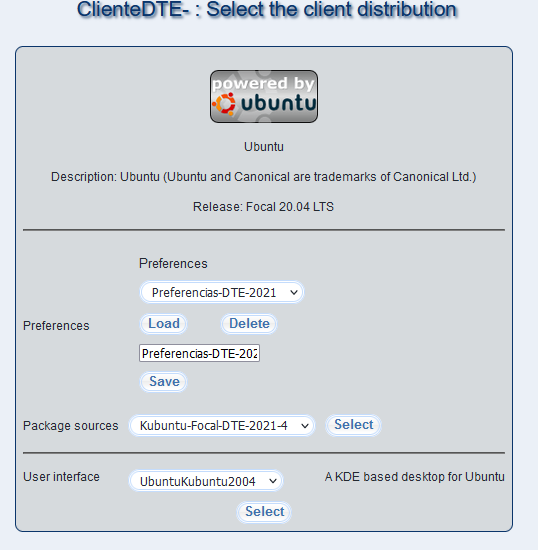


*Figura 116. Botón de finalizar particionado*

Nos indicará que los datos se han guardado. Pulsamos **Next**. Ahora debemos seleccionar la fuente y la lista de paquetes.

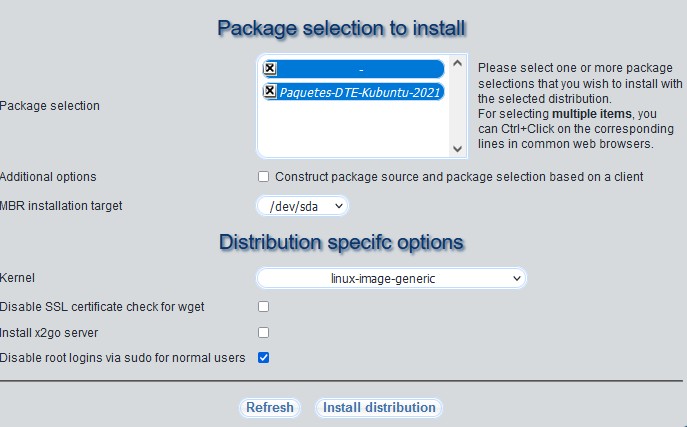
1. Elegimos la que hemos creado y pulsamos **Select**. Dejamos la interfaz

**UbuntuKubuntu2004** y pulsamos **Select** en la parte inferior.



*Figura 117. Selección de la distribución para los clientes*

1. En la nueva ventana que se desplegará, seleccionamos nuestra lista de paquetes personalizada con Ctrl izquierdo y pulsando en ella, de forma que la que está seleccionada por defecto siga estándolo. Dejamos el resto de opciones como están, salvo la de **Disable root logins via sudo for normal users**. De esta forma nos aseguraremos que el alumnado no tenga privilegios de administrador de cara al buen funcionamiento de los equipos.



*Figura 118. Selección de las listas de paquetes para los clientes*

1. Pulsamos en **Install distribution**.

Hemos finalizado la creación de la plantilla de clientes.

### Obtención de las direcciones MAC de los equipos cliente

Antes de pasar a la instalación de los clientes, necesitamos obtener sus direcciones MAC/Ethernet para que el servidor pueda conectarse a ellos mediante arranque PXE. Para ello, nos vamos a ayudar del DTE-Admin.

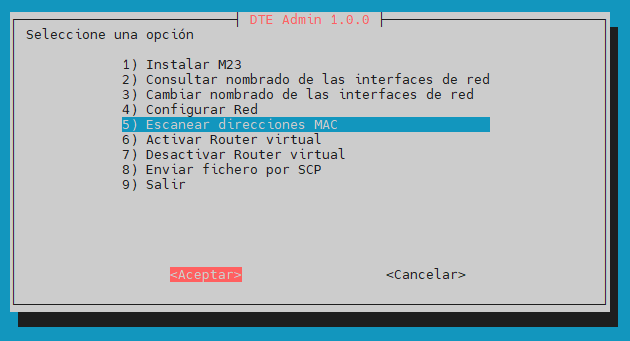
**NOTA**: antes de proceder, es importante asegurarse de que los diferentes equipos cliente se encuentran apagados, conectados a la subred del m23server y tienen activado el arranque por red como primera opción.

**NOTA**: Hay que tener en cuenta que, si existe otro servidor DHCP en nuestra subred, se producirán colisiones. En este sentido, la documentación de m23 [200] nos da varias opciones. Nosotros optaremos por desconectar servicio DHCP de nuestro router doméstico. No se volverá a activar hasta completar la instalación de los clientes.

**NOTA**: la interfaz del m23 nos avisará si detecta otro servidor DHCP en la red:

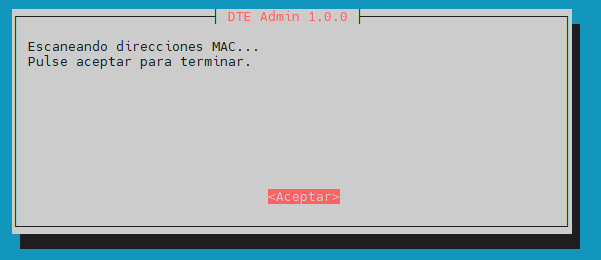
Abrimos un terminal del m23server (local o SSH) y ejecutamos el DTE-Admin.

Elegimos la opción 4: Escanear direcciones MAC.



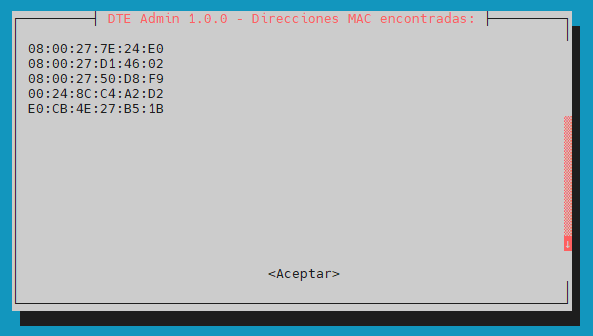
*Figura 120. Opción 5 del DTE-Admin*

1. Nos pedirá el nombre del fichero donde guardar los resultados. En nuestro caso lo dejamos por defecto. El escaneo de la red comenzará.
2. Ahora debemos ir encendiendo, uno a uno y con un margen aproximado de entre 5 y 10 segundos, todos los clientes de nuestra red.
3. Cuando estén todos encendidos y haya pasado un tiempo prudencial (en torno a 2 minutos) para que les dé tiempo a mandar sus peticiones de arranque al servidor, pulsamos aceptar para detener la captura.



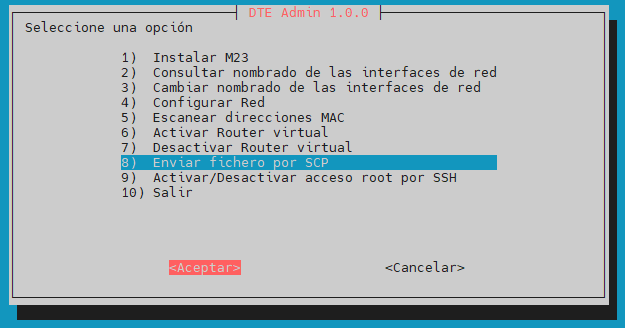
*Figura 121. Escaneo de direcciones MAC del DTE-Admin*

1. Se nos mostrarán por pantalla las direcciones MAC obtenidas. En nuestro caso son 5, como esperábamos. Deberían aparecer en el orden de arranque, por esto es importante dejar algo de margen entre el encendido de un equipo y otro. Para este ejemplo, lo hemos cotejado y efectivamente aparecen en orden.



*Figura 122. Direcciones MAC encontradas por el DTE-Admin*

1. Ahora necesitamos tener el fichero con los resultados en el mismo equipo desde el que accedemos a la interfaz web de m23. Si es desde el propio servidor, puede obviarse este y el siguiente paso. Si no, debemos enviar el fichero de la forma que elijamos.
2. Una de las formas más sencillas es a través de SCP. Para ello debemos tener implementado un servidor SSH en nuestro equipo anfitrión. Usaremos la opción 8 del DTE-Admin para ello:



*Figura 123. Opción 8 del DTE-Admin*

### Dando de alta a los clientes

Ya solo queda dar de alta a los clientes y comenzar el proceso de instalación. Para ello, seguimos los siguientes pasos:

1. En el menú de la izquierda, dentro del apartado **Mass tools**, pulsamos **Mass install**.
2. Seleccionamos la plantilla que habíamos creado y pulsamos el botón **Mass install** junto a ella:



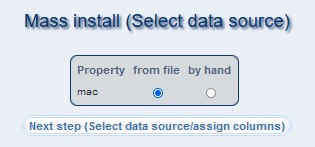
*Figura 124. Listado de plantillas de clientes de m23*

1. Ahora elegimos el criterio para completar la información particular de cada cliente. Dentro de las opciones que nos permite elegir, marcamos como **Keep** todas, **a excepción de Client name e IP Address**, que las marcaremos con **Generate**. Pulsamos **Save**.



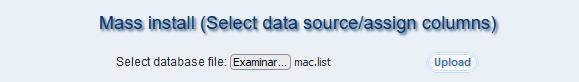
*Figura 125. Método de adición de parámetros de clientes m23*

1. En la siguiente ventana nos preguntará el método para introducir las direcciones MAC. Seleccionamos **from file** y pulsamos **Next step**.



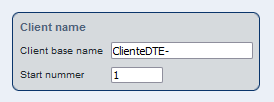
*Figura 126. Selección del formato de datos a introducir*

1. Ahora, mediante el botón Examinar, seleccionamos el archivo generado por el DTE- Admin y pulsamos **Upload**.



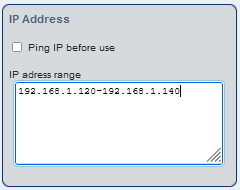
*Figura 127. Diálogo para subir archivo con los datos de clientes*

1. Introducimos un espacio en el campo **Separator** y pulsamos **Refresh**. Después, pulsamos de nuevo **Save**.
2. Ahora debemos configurar el criterio para generar los nombres de cliente. En nuestro caso ponemos como nombre base **ClienteDTE-** y como número de inicio el **1**:



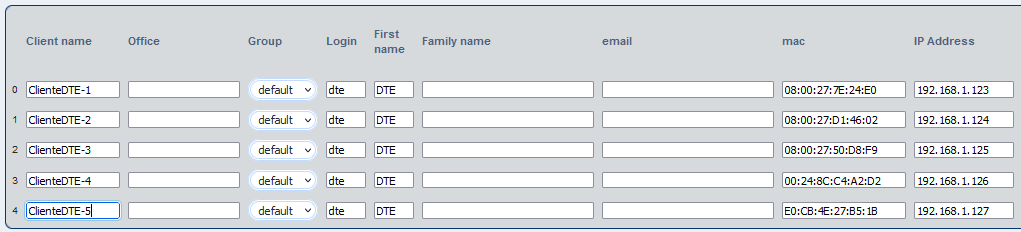
*Figura 128. Diálogo para la generación de nombres de cliente*

1. Indicamos ahora el rango de direcciones IP que deseamos usar para asignar a los clientes. También nos da la opción de comprobar con ping previamente si dichas direcciones están libres. En nuestro caso no usaremos esta opción.



*Figura 129. Diálogo para la generación de direcciones IP de cliente*

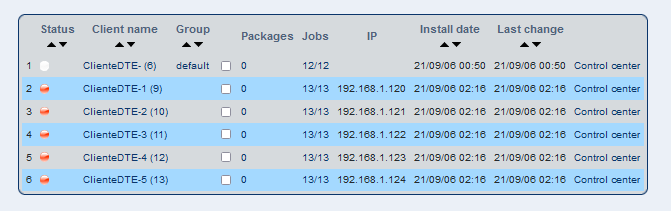
1. Pulsamos **Save**.
2. Nos aparecerá ahora una última ventana con el repaso de la información de los clientes. Podemos modificar algo a mano si es necesario. En nuestro caso pulsamos directamente **Save**.



*Figura 130. Resumen de los parámetros de los clientes dados de alta*

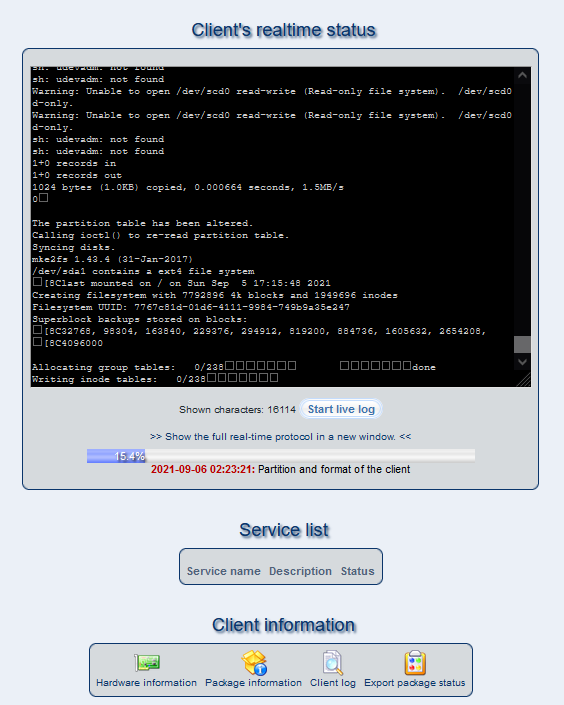
Una vez finalizado este paso, no tenemos más que encender, o reiniciar, los clientes. Arrancarán por red y el servidor m23 los configurará e instalará de forma automática.

Podemos hacer el seguimiento desde las propias pantallas de los clientes, o desde la interfaz del m23 en el menú de la izquierda, pulsando en **Clients** ** **Overview**.



*Figura 131. Clientes dados de alta en m23*

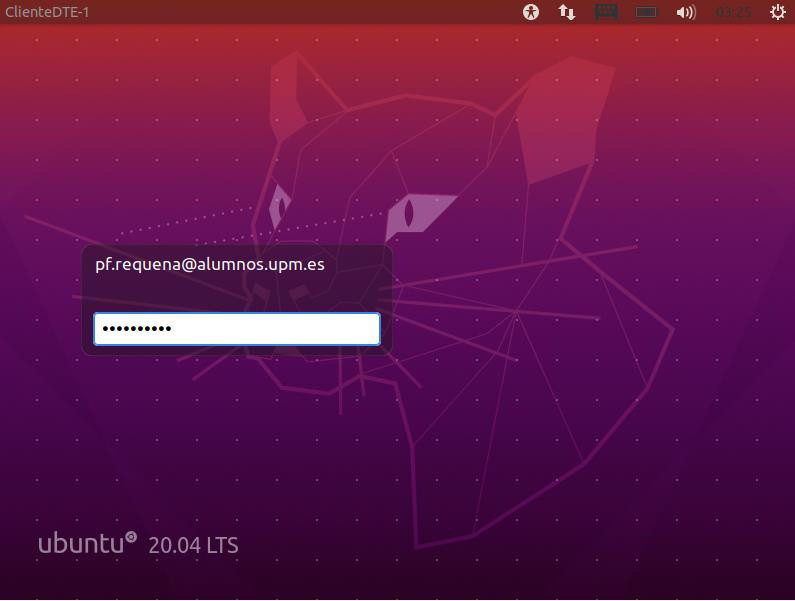
Pulsando en **Control center** en cada uno de ellos, podemos acceder a una terminal por red en tiempo real, a los logs y muchas otras opciones descritas en el manual de usuario [200]



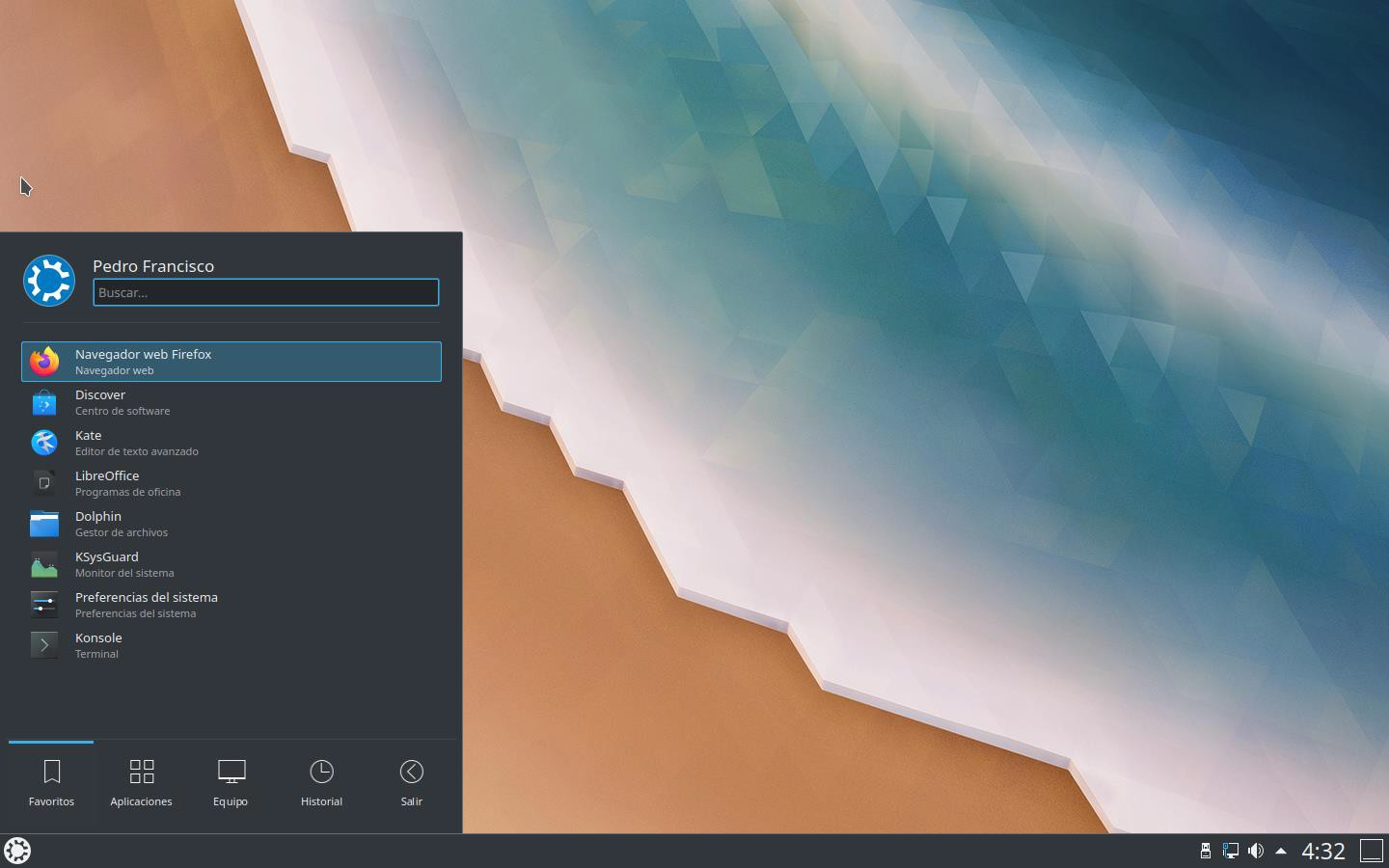
*Figura 132. Panel de control de cliente m23*

Esperamos en torno a 40 minutos y los clientes se habrán instalado y reiniciado de forma automática.

A continuación se muestran capturas de pantalla del ClienteDTE-2 y ClienteDTE-4 ya instalados donde se aprecia el funcionamiento de la autenticación por LDAP y el aspecto inicial del escritorio KDE Plasma 5 en Kubuntu:



*Figura 133. Gestor de sesiones Kubuntu con credenciales LDAP*



*Figura 134. Escritorio KDE Plasma 5 en Kubuntu recién instalado con acceso por LDAP*

## Mantenimiento de los clientes

Una vez los clientes están instalados, es posible que haya que realizar operaciones de mantenimiento, tales como instalar nuevo SW, actualizarlo, liberar espacio en disco, etc.

A través de la interfaz de m23 se pueden hacer la mayoría de estas tareas, además de otras como la gestión de copias de seguridad del servidor, la creación de imágenes a partir de clientes instalados, asimilación de clientes instalados previamente, etc. Además, si alguna tarea precisa no se pudiera realizar directamente, se puede recurrir a los scripts PHP que hemos mostrado, que cuentan con una API oficial detallada [201].

En cualquier caso, entendemos que la planificación concreta del mantenimiento de los equipos queda fuera de los límites del proyecto. Creemos que, con la información proporcionada aquí, unida al manual de usuario de la aplicación [200], no supondría ningún problema para el personal técnico realizar con este nuevo entorno las labores de mantenimiento que vienen desempeñando hasta ahora con sistemas Windows y el gestor de imágenes FOG.

# PRESUPUESTO

En este capítulo haremos una estimación presupuestaria del coste del proyecto, que ha sido sufragado por la ETSIST y por el alumno.

|  |  |
| --- | --- |
| **Concepto** | **Precio estimado (€)** |
| **Ordenador portátil i5 7ª Generación** | 400 |
| **Sobremesa i5 3ª generación** | 200 |
| **2 x Sobremesa Core 2 Quad** | 200 |
| **Switch Ethernet** | 12 |
| **Cables Ethernet** | 10 |
| **KVM VGA/USB** | 30 |
| **2 x Teclado USB** | 20 |
| **Ratón USB** | 5 |
| **Ratón Inalámbrico** | 9 |
| **Monitor VGA 19”** | 50 |
| **Monitor HDMI 27”** | 150 |

*Tabla 11. Desglose presupuestario*

En cuanto a la mano de obra, se han dedicado en torno a 300 horas de trabajo. Si tomamos el sueldo medio de un ingeniero recién titulado, obtenemos:

|  |  |
| --- | --- |
| **Horas trabajadas** | **Coste (11€/hora)** |
| **300** | 3.300€ |

*Tabla 12. Coste de la mano de obra*

Si sumamos las cantidades, obtenemos un **total presupuestario estimado de 4.386 euros** para la realización del proyecto.

# CONCLUSIONES

Como conclusiones finales tras la realización del proyecto, pueden resumirse en:

Para empezar, ha quedado constancia de que el ámbito del software libre es gigantesco e implica a millones de personas alrededor del mundo, en base a toda la información que se ha recopilado. Esto implica, por un lado, una serie de ventajas importantes tanto a nivel ético como técnico respecto al software y metodologías cerradas-privativas, como ya comentamos en los primeros capítulos de la memoria. Sin embargo, también se ha podido ver cómo entraña dificultades, en el sentido de que existe un sinfín de alternativas con particularidades, por lo que uno se puede llegar a sentir algo perdido en la toma de decisiones. La filosofía del software libre y abierto requiere, sin duda, un mayor esfuerzo de análisis y estudio tecnológico que el privativo, pero, a su vez, es una filosofía socialmente liberadora, sirviendo como equilibrio frente a los monopolios de la información y defendiéndonos de la cada vez mayor vulneración de la intimidad a través la tecnología comercial.

También me gustaría afirmar, haciendo una síntesis entre todo lo aprendido en la realización del proyecto y la experiencia de mis años en la universidad, que el desarrollo de software libre y colaborativo es la metodología que va más en consonancia con la naturaleza humana, es decir, con una naturaleza que necesita de la colaboración con los demás para progresar. En los tiempos que vivimos, marcados por el individualismo y el beneficio económico privado por encima de cualquier consideración humana, el hecho de que exista un mundo tan complejo como el del software libre que, desde la participación conjunta y el conocimiento colectivo sin ánimo de lucro pueden competir cada a cara con grandes corporaciones multinacionales.

Por otro lado, y como ya se ha comentado en varias ocasiones a lo largo de la memoria, aunque estemos convencidos de las ventajas del software libre y deseemos su implementación, debemos ser realistas y buscar siempre un equilibrio real entre lo libre y lo manejable. Por ello se ha tratado de garantizar en todo momento que, de aplicarse esta propuesta, el impacto fuera mínimo para todas las personas involucradas, aunque esto requiera instalar software privativo o evitar soluciones 100% libres, pero poco compatibles a nivel de hardware. Para cambiar la realidad debemos ser conscientes de las circunstancias que la conforman y adaptarnos a ella.

Yendo a lo concreto, he disfrutado mucho conociendo más a fondo el ecosistema GNU/Linux. Por mi experiencia laboral, estaba acostumbrado únicamente a la administración de sistemas de la familia Red Hat y, gracias a la realización de este proyecto, he podido adentrarme en la familia Debian – Ubuntu, experiencia que considero muy enriquecedora desde el punto de vista personal, académico y laboral.

Por último, sobre la experiencia con el uso de la herramienta m23: creo que es una herramienta interesante y totalmente apta para entornos como el de los laboratorios DTE por el tipo de operativa que ofrece, además, como decíamos, su sencillez ofrece una rápida puesta en marcha sin necesidad de configuraciones manuales, ni manejo de ficheros complejos. Sin embargo, y a pesar de que son muchas las opciones con las que cuenta, el hecho de que su manejo sea a tan alto nivel y su documentación algo escueta, nos ha llevado a tener que tomar puertas traseras y no del todo controladas para la resolución de problemas concretos que no ofrecen soporte desde su interfaz, que, aunque versátil, tiene unos límites muy marcados.

# LÍNEAS FUTURAS

Por último, en cuanto a posibles líneas futuras del proyecto, se pueden resumir en las siguientes:

* La primera y más importante, que sea realmente implementado. Un despliegue sencillo en alguno de los laboratorios DTE, o solamente en algunos de los equipos, permitiría, por un lado, el uso de las herramientas de despliegue por parte del personal técnico y, por otro, de los sistemas por parte de alumnado y profesorado. Con esta experiencia de uso y su consiguiente retroalimentación, se podrían saber a ciencia cierta los puntos fuertes y débiles de la propuesta, de forma que se desarrollen mejoras a nivel de procedimientos, uso de otras herramientas, distribuciones, etc. con el fin de que todos los grupos implicados en la infraestructura informática de los laboratorios se sintieran cada vez más cómodos.
* Dentro de esa implementación real, quedaría pendiente configurar adecuadamente el acceso a la unidad de disco compartido en red, así como estudiar una posible metodología para instalar el SW que requiere de intervención manual.
* Integración con otros proyectos con finalidad similar. Por ejemplo, existe un proyecto llamado OpenGnSys desarrollado por diferentes universidades españolas [219], cuyo objetivo es instalar y gestionar infraestructuras informáticas en entornos universitarios. Aunque el fin es similar al nuestro, la forma de llevarlo a cabo es diferente, ya que trabaja con imágenes de sistema previamente instalados a mano, para después copiarlas al resto de equipos, de la misma forma que lo hace FOG, la herramienta usada actualmente por el personal técnico del DTE [147]. Esta metodología conlleva importantes restricciones a nivel de modificaciones, distinción de HW, grado de intervención manual, etc. Sin embargo, OpenGnSys incorpora una serie de funcionalidades de utilidad especialmente pensadas para entornos como el nuestro, como la gestión de aulas, distinción de usuarios según sean PAS, profesorado o alumnado, etc. Por ello, una línea de trabajo futura podría ser integrar este tipo de funcionalidades con despliegues automatizados GNU/Linux, en lugar de imágenes de disco.
* Aumentar el nivel de personalización implicando al alumnado. Como hemos ido viendo a lo largo de la memoria, una de las características principales de GNU/Linux es su versatilidad y capacidad para la configuración y la personalización. En este sentido, cabría la posibilidad de desarrollar paquetes, o, incluso, distribuciones propias, contando con la participación de toda la comunidad educativa. De esta forma se conseguiría tener entornos totalmente adaptados a las necesidades de los laboratorios y la escuela, a la par que puede ser útil como contenido práctico en determinadas asignaturas, adquiriendo los alumnos un tipo de conocimiento útil sobre el funcionamiento de los sistemas operativos, tanto para su futuro laboral como académico.
* En relación a todo lo anterior, la escalabilidad es un factor clave. Aunque esta propuesta está pensada para el caso particular de los laboratorios DTE de la ETSIST, podría ser adaptada sin mayor problema para otros entornos de la escuela, de la UPM, o, en general de cualquier entorno educativo. Cuanto más escalable sea, mayor será la comunidad a su alrededor y más y mejor crecerá un proyecto de estas características, tal y como el software libre y de código abierto nos han enseñado en las últimas décadas.

# BIBLIOGRAFÍA

Aho, A., Ullman, J., Sethi, R., & Lam, M. (2006). *Compilers: Principles, Techniques, and Tools* (2a ed.). Boston: Addison Wesley.

*JavaCC. The most popular parser generator for use with Java applications.* (s.f.). Recuperado el 10 de 10 de 2023, de https://javacc.github.io/javacc/

*Java™ Platform, Standard Edition 8. API Specification*. (s.f.). Recuperado el 26 de 09 de 2023, de https://docs.oracle.com/javase/8/docs/api/index.html

# REFERENCIAS

[1] «¿Qué es y para qué sirve un IDE?» Accedido: 19 de noviembre de 2023. [En línea]. Disponible en: https://www.redhat.com/es/topics/middleware/what-is-ide

# 

# ANEXO

# SÍMBOLOS DE EXPRESIONES REGULARES EN JAVACC

+: El símbolo + se usa para indicar que un elemento puede aparecer una o más veces. Por ejemplo, si tienes A+, significa que se espera que haya al menos una instancia de A, pero puede haber más.

\*: El símbolo \* se usa para indicar que un elemento puede aparecer cero o más veces. Por ejemplo, si tienes B\*, significa que B es opcional y puede aparecer cero o más veces.

?: El símbolo ? se usa para indicar que un elemento puede aparecer cero o una vez. Por ejemplo, si tienes C?, significa que C es opcional y puede aparecer cero o una vez.

~: El símbolo ~ se utiliza para excluir ciertos caracteres o elementos. Por ejemplo, ~A significa "cualquier cosa excepto A". En una expresión regular, ~ se usa para negar un conjunto de caracteres. Por ejemplo, [~0-9] significa "cualquier carácter que no sea un dígito del 0 al 9".

# Preguntas Frecuentes

##### ¿Qué pasa si tengo dos tokens que coinciden con la misma cadena de caracteres?

Todas las expresiones regulares en el estado léxico actual se consideran posibles candidatas a coincidir. El administrador de tokens consume la cantidad máxima posible de caracteres del flujo de entrada que coincidan con una de estas expresiones regulares. Es decir, el administrador del token prefiere la coincidencia más larga posible. Si hay varias coincidencias más largas de la misma longitud, la expresión regular que coincide es la que tiene el primer orden de aparición en el archivo de gramática.

##### ¿Cómo hago para leer de un fichero?

Dependiendo de si pones la opcion modern

options

{

JAVA\_TEMPLATE\_TYPE = "modern";

}

, se podrá leer de dos diferentes maneras:

sin JAVA\_TEMPLATE\_TYPE = "modern" →

XMLParser parser = new XMLParser(new XMLParserTokenManager(new SimpleCharStream(new InputStreamReader(new FileInputStream(args[0]), "UTF-8"))));

con JAVA\_TEMPLATE\_TYPE = "modern" →

XMLParser parser = new XMLParser(new XMLParserTokenManager(new SimpleCharStream(new StreamProvider(new FileInputStream(args[0]), "UTF-8"))));

##### ¿Cuál es la diferencia entre InputStream y InputStreamReader?

They represent somewhat different things.

The InputStream is the ancestor class of all possible streams of bytes, it is not useful by itself but all the subclasses (like the FileInputStream that you are using) are great to deal with binary data.

On the other hand, the InputStreamReader (and its father Reader) are used specifically to deal with characters (so strings) so they handle charset encodings (utf8, iso-8859-1, and so on) gracefully.

The simple answer is: if you need binary data you can use an InputStream (also a specific one like a DataInputStream), if you need to work with text use an InputStreamReader..

##### ¿Cuáles son las opciones que tiene el parser?

Ignore\_Case= true, para no hacer distinción entre mayúsculas y minúsculas.

Static = false, para que los métodos que genere la compilación no sean estáticos.