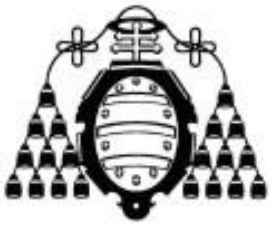
UNIVERSIDAD DE OVIEDO



ESCUELA DE INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

"DSL (Lenguaje de Dominio Específico) para configurar un servidor de forma segura"

DIRECTOR: Cristian González García

AUTOR: Álvaro García Infante

Resumen

La importancia de mantener un servidor seguro aumenta día a día, en parte gracias al uso de internet. Las vulnerabilidades que surgen en un sistema pueden ser conocidas por cualquier atacante y/o defensor en cualquier punto del globo en cuestión de horas desde que han sido descubiertas. Para el atacante es sencillo, solo tienen que usar la vulnerabilidad. Para el defensor no lo es tanto, ya que tienen que usar un parche para cubrir ese fallo.

Si bien existen páginas como el CVE *(Common Vulnerabilities and Exposures)* que se dedican íntegramente a detectar vulnerabilidades, informar y muchas veces sugerir como se pueden arreglar, estos parches no pueden ser aplicados por cualquier persona. Se necesita de un conocimiento técnico sobre sistemas informáticos para resolver estos *bugs.* Dichos *bugs* suelen ser corregidos en las actualizaciones de los sistemas operativos, pero esto no es inmediato.

Este proyecto, desarrollado en Python, permite a casi cualquier persona, con pocos conocimientos de seguridad informática, configurar un servidor Apache de una forma simple y segura. Todo esto a través de una interfaz de línea de comandos y un lenguaje de diseño específico, que se explicará más adelante. Los diferentes módulos y comandos que se aplican son extraídos de la guía de CIS Benchmarks de Apache llamada “CIS Apache HTTP Server 2.4 Benchmark”.

Además, gracias a la funcionalidad del sistema de crear comandos, el usuario puede mantener su sistema actualizado ante las nuevas vulnerabilidades que surjan. Incluso importar comandos de otro usuario para mantener su propio sistema actualizado.

El proyecto sirve para varios sistemas operativos, ya que se puede ejecutar tanto en Linux (Ubuntu y CentOS), como en Windows (Windows 10). Además, es similar a cualquier aplicación de interfaz de comandos y cuenta con opciones de ayuda, explicación de los comandos que se aplican e incluso de auto rellenado que ayudan al usuario a entender cómo usar el sistema en pocos minutos.

Existen tecnologías enfocadas con el mismo objetivo que esta como puede ser JShielder, que aplica una serie de medidas de seguridad en un script y que, en conjunto, mejoran lo robusto que es un servidor de Linux. El objetivo es similar, pero la ejecución es muy diferente, en este caso el usuario definirá un lenguaje. En la búsqueda de sistemas similares, no se encontró nada que pudiese cumplir una funcionalidad parecida a lo que se pretende desarrollar.

No se trata de una automatización a modo script que aplica todas las medidas a la vez (aunque también existe dicha opción), sino que el usuario indicará a través de un lenguaje de dominio específico que medidas se aplica, definirá nuevas medidas o incluso compartirá las mismas con otros usuarios a través de archivos de tipo JSON.

Palabras Clave

DSL, Apache, Seguridad, Servidores, Lenguaje de Dominio Específico, Vulnerabilidad.

Abstract

The importance of maintaining a server secure is increasing day by day, in part thanks to the use of the Internet. Vulnerabilities that arise in a system can be known to any attacker and/or defender anywhere in the world in a matter of hours after they have been discovered. For the attacker, it is simple, they just must use the vulnerability. For the defender, it is not so easy, as they must use a patch to cover the flaw.

Although there are pages such as CVE (Common Vulnerabilities and Exposures) thar are entirely dedicated to detecting vulnerabilities, reporting, and often suggesting how they can be fixed, these patches cannot be applied by just anyone. Technical knowledge of computer systems is required to solve these bugs. Such bugs are usually fixed in operating system updates, but this is not immediate.

This project, developed in Python, allows almost anyone, even with little knowledge of computer security, to configure an Apache server in a simple and secure way. All this through a command line interface and a specific design language, which will be explained later. The different modules and commands that are applied are extracted from the Apache CIS Benchmarks guide called “CIS Apache HTTP Server 2.4 Benchmark”.

In addition, thanks to the system’s command creation functionality, the user can keep his system up to date with new vulnerabilities as they arise. You can even import commands from another user to keep your own system up to date.

The project is suitable for various operating systems, as it can be run on both Linux (Ubuntu and CentOS) and Windows (Windows 10). In addition, it is like any command interface application and has help options, explanation of the commands that are applied and even autofill that help the user to understand how the system works in a few minutes.

There are technologies focused on the same objective as this one, such as JShielder, which applies a series of security measures in a script and that improve the robustness of a Linux server. The objective is similar, but the execution is very different, in this case the user will define a language. In the search for similar systems, nothing was found that could fulfill a functionality like what is intended to be developed.

It is not a script-like automation that applies all the commands at once (although this option also exists), but the user will indicate through a specific domain language which commands are applied, define new commands, or even share them with other users through JSON files.

Keywords

DSL, Apache, Security, Servers, Domain Specific Language, Vulnerability.

Índice General

Índice de Figuras

Capítulo 1. Memoria del Proyecto

El proyecto se trata de una aplicación que se ejecuta en la consola de comandos, al ejecutarla se abrirá un terminal en el que mediante una serie de términos pertenecientes al lenguaje de dominio específico que aquí se describe, se pueden ejecutar diversas órdenes que resulten en mejorar la seguridad del servidor. Dicha aplicación está enfocada tanto a sistemas Linux como a Windows, pero siempre teniendo en cuenta que cuenten con un servidor Apache desplegado.

1.1 Resumen de la Motivación, Objetivos y Alcance del Proyecto

La idea de desarrollar una aplicación de dichas características surge de la propia necesidad. Aunque existen sistemas que ejecutan una serie de comandos y en conjunto endurecen el sistema haciéndolo más robusto, no existen lenguajes de dominio específico para este tipo de tarea específica. Se comentará en secciones posteriores que existen lenguajes de dominio específico orientados a aumentar la seguridad de sistemas, pero no llegan a cumplir del todo con el propósito que tiene este trabajo.

El proyecto, que ahora puede ser usado en Ubuntu, CentOS y Windows 10, puede ser extendido para cualquier otro sistema operativo, suponiendo que se defina un nuevo JSON que siga la estructura propuesta en los demás y se modifique ligeramente el código de la aplicación.

El sistema permite la creación, ejecución, exportación e importación de comandos, dichos comandos están definidos en un JSON por sistema operativo. Las opciones de creación o importación de comandos se incluyen para cubrir la posibilidad de que surja una vulnerabilidad y se pueda añadir al sistema. Además, también permite cambiar el idioma de la aplicación a inglés e incluso correr varios comandos a la vez.

También se ha escrito un artículo científico sobre la aplicación y su utilidad tras analizar las alternativas existentes, de forma que se difundan los resultados de este trabajo final de grado (TFG).

1.2 Introducción

En esta sección explicaremos la justificación del proyecto, es decir, el motivo por el que se desarrolla el proyecto. Además, se enumerarán los objetivos del proyecto y se hará un estudio de la situación actual, de todos los sistemas similares al que se va a desarrollar.

1.3 Aspectos teóricos

Describiremos brevemente los conceptos, herramientas y tecnologías existentes que vamos a usar en nuestro proyecto.

1.4 Planificación del proyecto y presupuestos iniciales

Se trata de desarrollar la planificación del proyecto y presupuestos antes de iniciar el proyecto, por lo que estos pueden variar para la versión final. La descomposición de tareas en subtareas se aplica EDT.

1.5 Análisis

La sección de análisis se encarga de sentar una base sobre la cual se elaborará posteriormente el diseño del sistema. Se encarga de definir el sistema, determinar los requisitos de este, especificar los casos de uso y los actores, identificar los subsistemas y diseñar un plan de pruebas.

1.6 Diseño del Sistema

El diseño del sistema se trata de extender el análisis hacia un diseño más acabado y que es el que se aplica realmente en el sistema. Es por ello por lo que las subsecciones son similares a las de análisis del sistema.

1.7 Implementación del Sistema

Dentro de la implementación del sistema se pasará a explicar aquellos estándares, lenguajes y herramientas usadas para el desarrollo de la aplicación. Además, también se citarán algunos problemas encontrados durante el desarrollo de la aplicación.

1.8 Desarrollo de las pruebas

Se encarga esta sección de desarrollar las pruebas especificadas en el punto 6.7 de esta misma documentación, pruebas unitarias, de integración y del sistema, de usabilidad y de rendimiento.

1.9 Manuales del Sistema

Los manuales del sistema sirven para que un nuevo usuario sepa cómo usar el programa y un nuevo programador sepa como extender las funcionalidades del sistema. Los manuales son de instalación, de ejecución, de usuario y del programador.

1.10 Conclusiones y Ampliaciones

Esta sección se basa en extender las conclusiones sobre el sistema elaborado y mencionar cualquier labor de ampliación contemplada.

1.11 Planificación del Proyecto y Presupuesto finales

Se desarrolla en esta sección la versión definitiva de la planificación y presupuesto del proyecto, incluyendo todos los cambios respecto de la versión inicial.

1.12 Referencias

La sección de referencias bibliográficas tiene como propósito citar aquellos libros, artículos o referencias de Internet que han sido usados de alguna forma durante el desarrollo del proyecto o de esta documentación.

1.13 Apéndices

Esta sección se encarga de cubrir apartados necesarios de la documentación no mencionados anteriormente. Estos apartados serían: glosario y diccionario de datos, contenido entregado en el archivo adjunto, índice alfabético, código fuente, propiedad intelectual y licencias y actas de reuniones.

Capítulo 2. Introducción

Desarrollaremos a continuación el motivo por el que se desarrolla el proyecto, los objetivos que se quieren alcanzar y un estudio de los sistemas similares al que se va a desarrollar.

2.1 Justificación del Proyecto

El proyecto surge de la falta de aplicaciones que mediante un DSL permitan configurar un servidor de forma fácil, segura y rápida. La aplicación cubre diferentes aspectos de la seguridad de sistemas informáticos, divididos dentro de la propia aplicación como paquetes.

Dichos paquetes no serán más que conjuntos de comandos que se agrupan debido a tener la misma naturaleza en el aspecto de la seguridad, todos ellos incluidos en un único JSON.

Todos estos paquetes serán fácilmente aplicables a través de un sistema simple, será como cualquier terminal estándar, en el que usuario escribirá, dado el lenguaje de dominio específico, unas instrucciones que serán aplicadas. Dicho sistema tendrá formas de ayudar al usuario a entender cómo funcionan los comandos, cuáles son los paquetes que puede aplicar y su utilidad y sentido de ser. Además de un historial y predicción de comandos para aumentar la usabilidad del sistema.

El proyecto intenta, en la medida de lo posible, cubrir todas las necesidades básicas de seguridad de un servidor Apache, siguiendo como guía para estas necesidades la guía del CIS (Centro para la seguridad de Internet) “CIS Apache Http Server Benchmark”. Además, teniendo en cuenta la necesidad de aplicar parches a los sistemas y de mantenerlos actualizados, se incluyen maneras de crear nuevos comandos, importarlos y exportarlos.

Para los usuarios que tengan conocimientos más básicos, también se incluyen formas de correr una selección de comandos básicos, una especie de configuración base.

2.2 Objetivos del Proyecto

A forma de resumen, el proyecto tratará de cubrir los siguientes puntos:

1. Aumentar la seguridad del servidor: el objetivo principal de la aplicación será, aumentar la seguridad de los servidores donde se aplique. De esta manera una vez ejecutados los comandos que el usuario desee, se mejorará la seguridad general del servidor.
2. Sistema extensible: el sistema será muy fácil de extender, esto es, porque todos los módulos están escritos en un JSON muy simple de entender. En caso de querer mejorar la aplicación añadiendo más módulos, el programador sólo tendrá que copiar la estructura e indicar como quiere extenderlo.
3. Sistema multiplataforma: la aplicación podrá ser ejecutada tanto en Windows como en Linux. Dentro de Linux podrá ejecutarse tanto en Ubuntu como en CentOS.

4. Sistema adaptable: el sistema tendrá opciones para adaptarse a nuevos comandos que sean exigidos por nuevas vulnerabilidades surgidas en Apache.
5. Sistema en español e inglés: el lenguaje del sistema se extraerá del propio sistema que corre la aplicación, pero estará disponible en inglés y español, con opción de seleccionar el idioma al gusto del usuario.
6. Código simple: cada clase dentro de la aplicación cumple con una función específica siguiendo los patrones de diseño correspondientes y adecuados.
7. Alta usabilidad: el sistema pretende ser fácil de aprender para cualquier usuario que este familiarizado con los programas que funcionan por consola, copiando así las funcionalidades básicas que estos aportan.

2.3 Estudio de la Situación Actual

Se realizó una investigación en las principales páginas existentes de subida de trabajos y publicaciones. Se busco tanto en la página del IEEE, como en ScienceDirect y Google Scholar. Tras probar con todas las combinaciones posibles de los términos (*dsl*, *domain specific language*, *security*, *server*, *configuration*), se llegó a la conclusión de que no existen sistemas actuales que realicen lo que se describirá en este trabajo, aunque si hay algunas comparaciones que se harán a continuación.

Existen herramientas parecidas para aumentar la seguridad de los servidores, aunque no con un lenguaje específico. Por ejemplo, existe JShielder [5], que es un script de bash de código abierto que automáticamente endurece los servidores Linux.

El punto en común de nuestro sistema con JShielder es claro, mejorar la seguridad general del sistema. Aunque la ventaja que proporciona un lenguaje de dominio específico es que es el usuario el que decide que partes del sistema aplica que comandos, no es un script.

También se puede tener en cuenta en este ámbito el proyecto de OpenSCAP, que otorga una amplia variedad de guías, configuraciones y herramientas para buscar vulnerabilidades y problemas de configuración. Usaremos las guías de “CIS Benchmark” para los comandos que usa la aplicación y que ejecutan el código. Aunque son guías, no son ningún tipo de programa.

Haciendo una búsqueda extensiva encontramos tres sistemas que usan DSL en cuanto a la seguridad de los sistemas.

El primero, “Panoptis” [1] usa los registros de contabilidad producidos por todos los sistemas UNIX para detectar intrusos. El DSL se basa en configurar una serie de parámetros para detectar anomalías en el sistema y alertar al usuario.

Los dos siguientes “Lobster” [2] y “PPL” [3] se basan en definir políticas de usuario a través de un DSL. El primero centrado en el módulo de seguridad de Linux, SELinux. El segundo, PPL, se define de una forma general.

Los anteriores trabajos no están centrados en resolver un rango grande vulnerabilidades sino en resolver problemas específicos.

Después tenemos Egida [10], que permite desplegar configuraciones de seguridad en una infraestructura de máquinas. Utiliza CIS Benchmarks y Ansible. Sigue el enfoque estructurado que otorga el seguir la guía del CIS. Este sistema se centra en infraestructuras de máquinas, a diferencia con nuestra propuesta que se centra en servidores.

El propósito de nuestro lenguaje será que el usuario pueda realizar estas operaciones de endurecimiento del servidor de una forma más manual y configurable, es decir que el propio usuario sepa qué está aplicando y qué objetivo tendrá. Aparte, ninguno de los sistemas encontrados es multiplataforma.

Para el desarrollo de la aplicación se ha usado el lenguaje de programación Python, esto se debe a que estamos construyendo un DSL interno y estos se benefician de lenguajes más dinámicos. Además, para crear una aplicación ejecutable se usar auto-py-to-exe y para desarrollarla el IDE PyCharm.

2.3.1 Evaluación de Alternativas

A continuación, analizaremos las diferentes alternativas a nuestro sistema.

2.3.1.1 JShielder

JShielder [5] es un script de Bash de código abierto desarrollado para ayudar a los administradores de Sistema y desarrolladores a asegurar aquellos servidores Linux en los que despliegen cualquier aplicación web o servicio.

Las ventajas claras respecto a nuestro sistema es una mayor cantidad de comandos que se aplican de forma automática.

Las claras desventajas es que no es un DSL, solo se puede aplicar a sistemas Linux y no permite configurar que comandos aplicas y que comandos no.

2.3.1.2 Panoptis

Panoptis [1] es un DSL que se enfoca en la detección de intrusos dentro de un sistema, basado en los registros de contabilidad producidos por todas las versiones de UNIX. Dichos registros pueden ser usados para detectar situaciones anómalas, pero no pueden ser analizados manualmente. El lenguaje de dominio especifico se usará para especificar qué elementos van a ser comprobados.

Hablando del DSL escrito, el lenguaje permite comprobar diferentes partes del sistema y aplicar comandos específicos a cada uno para que comprueben una serie de parámetros. Por ejemplo, que compruebe el uso de memoria máximo de los usuarios. También permite mapear terminales y usuarios en grupos. Finalmente, también permite definir algunas constantes del sistema, como cuál será el tiempo exacto para avisar al usuario de que existe una intrusión.

La forma de escribir el archivo de configuración de una forma tan simple puede ser interesante a la hora de aplicarlo en nuestro sistema. Aunque este sistema no cubre toda la complejidad de la seguridad de un servidor, si cubre la parte de detección de intrusos de este. Aunque, como en los demás, solo sirve para Linux.

2.3.1.3 Lobster

Lobster [2] es un lenguaje de dominio específico para describir políticas de seguridad en los flujos de información. Los programas en Lobster son una secuencia de definiciones de clases y declaraciones del dominio. Describe una forma de refinar las políticas de flujo de información nativas de SELinux.

Este DSL solo cubre el módulo de seguridad SELinux, por lo que ni es multiplataforma, ni cubre todos los aspectos de seguridad necesarios que nuestro sistema pretende cubrir.

2.3.1.4 PPL

Policy Programming Language [3] es un lenguaje de dominio específico que se utiliza para especificar los requisitos de aprovisionamiento y autorización de las aplicaciones distribuidas. Sintácticamente hablando, está compuesto de cinco bloques esenciales: entidades, ámbitos, reglas, acciones y políticas para describir la operación adecuada. La especificación de una política se compondrá de un dominio, que definirá una serie de entidades, y unas reglas.

La forma de describir las políticas en interesante, pero una vez más no es multiplataforma.

2.3.1.5 Egida

Egida [10] permite desplegar configuraciones de seguridad en una infraestructura de máquinas. Utiliza CIS Benchmarks y Ansible. Utiliza un DSL llamado Aspida para definir qué secciones del CIS Benchmark se aplicarán al nodo maestro. Luego, utilizando Ansible aplica las mismas órdenes a los nodos esclavos. Este sistema no abarca servidores sino redes de ordenadores. Sin embargo, sigue el enfoque organizado que dan los Benchmarks del CIS.

Capítulo 3. Aspectos Teóricos

Desarrollaremos a continuación los conceptos, herramientas y tecnologías que usaremos en nuestro proyecto.

3.1 Hardening

El *hardening* o endurecimiento informático, es una medida de seguridad cuyo objetivo es reducir la superficie de vulnerabilidad, evitando así posibles ataques. El endurecimiento puede ser aplicado cambiando las contraseñas por defecto, eliminando software innecesario o deshabilitando servicios innecesarios.

La superficie de ataque es el número de formas posibles en las que el atacante puede acceder a un dispositivo o una red y extraer datos. Es decir, cuantas más funciones ejecute un sistema, más vulnerable es.

La división de la aplicación en paquetes responde a esta necesidad de reducir la superficie de vulnerabilidad al máximo. Dividir el problema general de la seguridad de los sistemas informáticos en partes permite abordar el problema de una forma más sencilla. Cada comando que se aplica en nuestra aplicación tiene como objetivo endurecer el sistema.

3.2 DSL

Un lenguaje de dominio específico (DSL) es un lenguaje informático especializado en un dominio concreto. Contrasta con un lenguaje de propósito general, que es aplicable a todos los dominios. Existen muchos tipos de DSL, desde los usados por las páginas web como HTML hasta algunos usados con un objetivo muy concreto en piezas de software. Los DSL más sencillos, sobre todo aquellos usados por una sola aplicación, se denominan a veces informalmente como mini lenguajes.

Establecer una línea divisoria entre los lenguajes de propósito general y los lenguajes de dominio específico no es siempre una tarea sencilla, ya que un lenguaje diseñado para un propósito concreto puede ser usado a veces de forma general o al revés. Por poner un ejemplo, Perl fue diseñado originalmente como una herramienta de procesamiento de texto, al igual que los scripts de Shell.

La diferencia principal entre los dos tipos de lenguajes es que el DSL se crea específicamente para resolver problemas de un dominio específico y no espera resolver problemas fuera del mismo.

Una ventaja clara de los lenguajes de dominio específico y el motivo de que existan es que, los expertos en el dominio del problema no deberían encontrar ningún problema entendiendo un DSL que esté bien diseñado.

Una clara desventaja de los lenguajes de dominio específico es la dificultad de diseñar uno correctamente, más luego enseñar a los usuarios a usarlo correctamente.

Dentro de los DSL, podemos ver diferentes clasificaciones. Pueden ser:

- Según la forma en la que modifican el lenguaje, pueden ser textuales o gráficos: si el usuario tiene que escribir como en SQL y HTML o tiene una herramienta gráfica que le puede ayudar como en BPMN con un drag and drop.
- Según su lenguaje padre, pueden ser externos o internos [4], [12]:
    - Los DSL externos no están ligados al lenguaje de propósito general (GPL) que se utiliza para trabajar con ellos. Pueden ser desarrolladas usando herramientas como Xtext para Java, DSL Tools o .Net, o usando archivos XML externos.
    - DSL interno: el DSL se escribe utilizando en GPL principal de una aplicación. Entonces, el DSL está incrustado en la GPL y comparte la infraestructura, las limitaciones y la sintaxis con la GPL.
- Según su desarrollo, pueden ser [4]:
    - Un DSL se considera fijo cuando ha sido implementado usando otro procesador separado. Entonces, necesitan un traductor y un procesador específicos. Por ejemplo, Latex y SQL.
    - Un DSL incrustado es cuando el DSL está incrustado en el GPL como una API y una Language Integrated Query (LINQ).
    - UN DSL modularmente compuesto es cuando se divide en módulos. Luego, se pueden combinar esos módulos para crear un DSL más grande.
    - UN DSL con soporte de meta programación son las plantillas como en C++ y los GPL reflexivos como Groovy.
- Según el punto de vista del dominio del problema pueden ser horizontales cuando tratan de modelar un problema genérico como Windows Forms, que no representa un dominio específico de unos pocos usuarios. Por otro lado, son verticales cuando modelan un dominio específico como la generación de videojuegos de estrategia en 2D.

3.3 Apache

El servidor Apache HTTP Server, coloquialmente llamado Apache, es un servidor web gratuito y de código abierto. Apache es desarrollado y mantenido por una comunidad de desarrolladores bajo el nombre de la *Apache Software Foundation* [6].

Actualmente y desde 1996 es el servidor web más usado en todo el mundo debido a su seguridad y estabilidad. La funcionalidad principal es servir a los usuarios los ficheros necesarios para visualizar la web.

Usaremos Apache en todos los sistemas operativos en los que probaremos la aplicación para asegurarnos de que funciona correctamente, concretamente probaremos la versión 2.4.

3.4 PyCharm

PyCharm es un entorno de desarrollo integrado para programación en Python creado por la compañía checa JetBrains [7]. Dentro de sus funcionalidades se pueden destacar análisis de código, un *debugger* gráfico, *unit testing*, integración con sistemas de control de versiones, soporte de desarrollo web con Django y ciencia de datos con Anaconda. Usaremos su versión Community Edition 2021.1.1.

Lo usaremos para el desarrollo de la aplicación en Python. Se escoge este IDE entre otros tras un pequeño estudio de las diferentes opciones que hay ya que es uno de los entornos más completos.

3.5 CIS Benchmark Apache Http 2.4

EL CIS o *Center for Internet Security* [8] es una organización sin ánimo de lucro creada en el año 2000. Su objetivo es hacer del mundo conectado un lugar más seguro para desarrollar, validar y promover soluciones prácticas que ayuden a la gente a protegerse ante las ciber amenazas.

Dentro del CIS se crean los CIS Benchmarks, que son guías creadas por expertos en el campo de la seguridad. En nuestro caso usaremos la CIS Benchmark Apache Http 2.4.

Se trata de una guía para la configuración segura de un servidor Apache. Se usarán para configurar los comandos que corren por debajo de la aplicación y que son resultado del empleo del lenguaje específico.

3.6 VirtualBox

VirtualBox [11] es un software de virtualización actualmente desarrollado por Oracle Corporation. Es usado para instalar sistemas operativos adicionales, a través de máquinas virtuales. Dentro de nuestro proyecto se usará para instalar máquinas en las que probar la aplicación.

La creación de máquinas virtuales y pruebas en ellas permite al desarrollador probar un sistema en pruebas en un entorno seguro, de esta forma no se corre el riesgo de estropear el sistema operativo original. Además, permite desde una sola máquina probar varios sistemas operativos al mismo tiempo, lo cual es totalmente necesario para desarrollar una aplicación multiplataforma como la nuestra.

3.7 Auto-py-to-exe

Auto-py-to-exe[9] es una aplicación creada por Brent Vollebregt. Convierte aplicaciones Python en ejecutables usando una interfaz gráfica simple.

Usaremos auto-py-to-exe para crear el ejecutable de Windows.

3.8 Métrica 3

La metodología métrica versión 3 es un instrumento para la sistematización de las actividades que dan soporte al ciclo de vida del software. Dicha metodología es promovida por el ministerio de Hacienda y Función Pública del Gobierno de España. La metodología está basada en el modelo de procesos del ciclo de vida de desarrollo ISO/IEC 12207, así como en la norma ISO/IEC 15504 SPICE.

Este documento está desarrollado usando la metodología Métrica 3, pero de forma adaptada, contemplando solo aquellos apartados que se han considerado más adecuados para un proyecto de fin de carrera.

Capítulo 4. Planificación del Proyecto y Presupuesto Iniciales

Se desarrolla a continuación la planificación del proyecto y el presupuesto inicial.

4.1 Planificación Inicial

La planificación del proyecto se realiza para tener control sobre el desarrollo de las diferentes partes de un proyecto. En este caso se aplica una estructura de descomposición del trabajo que permite dividir las actividades en subtareas, de forma que, resulte óptimo a la hora de estimar el tiempo que cada una de ellas llevarán.

Se incluyen a continuación dos capturas, de las dos principales partes del desarrollo del trabajo. La primera es la de desarrollo de la aplicación y la segunda la del desarrollo del trabajo.

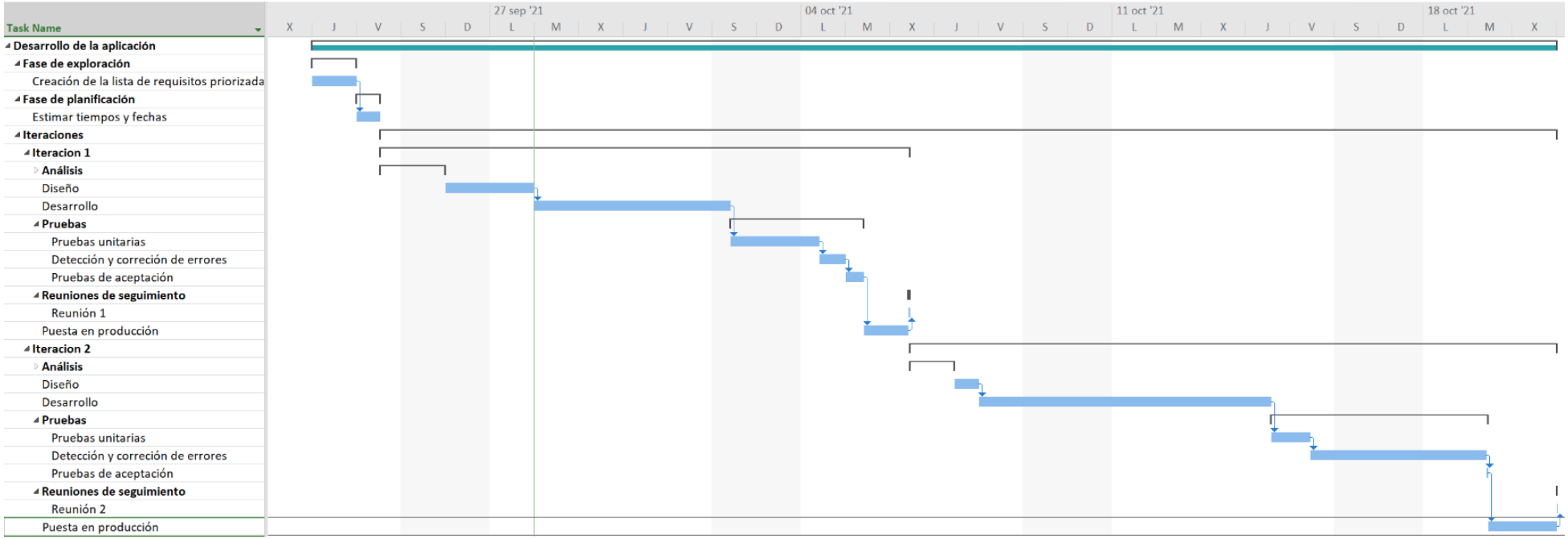


Figura 4. 1. Diagrama Gantt del Desarrollo de la Aplicación

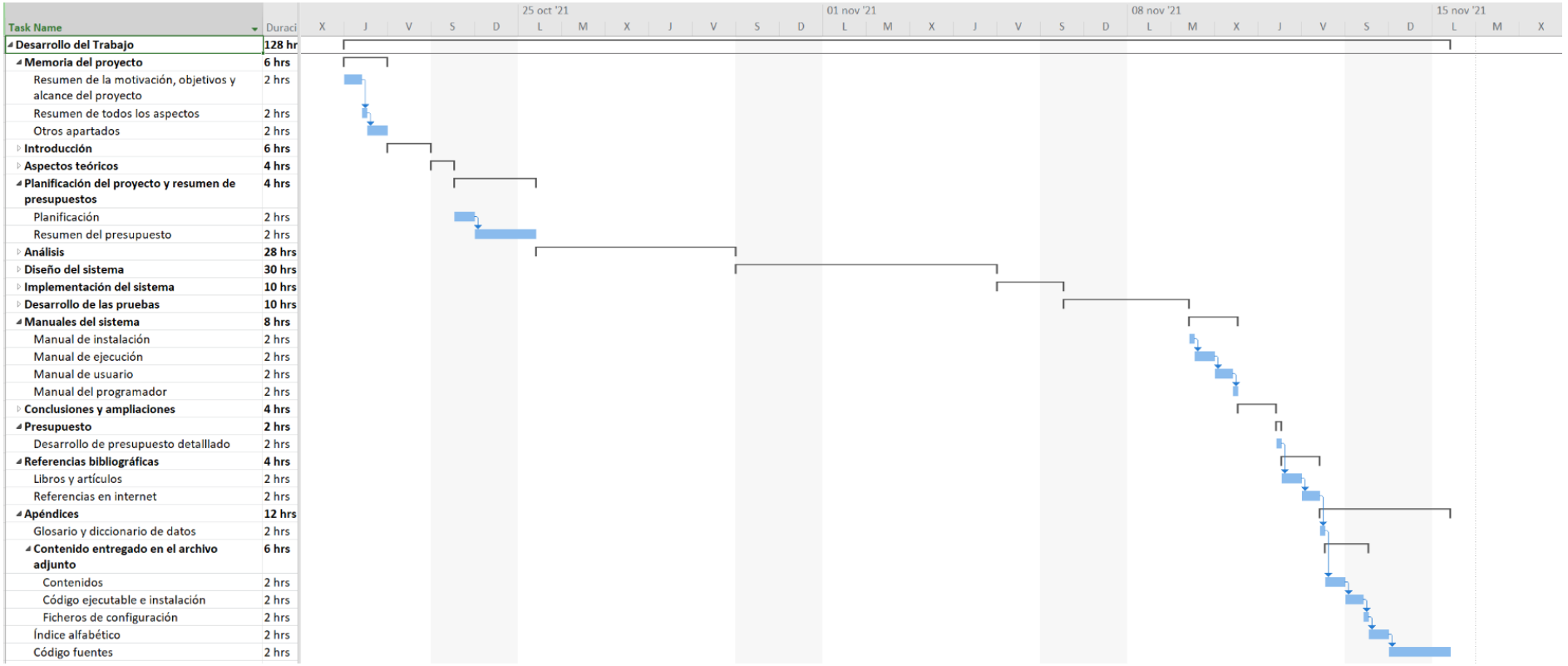


Figura 4. 2. Diagrama Gantt del Desarrollo del Trabajo

La forma de enfocar la planificación es con una metodología ágil, basada en Scrum, para el desarrollo de la aplicación.

Para el desarrollo de la aplicación primero se crea una lista de requisitos priorizados, a esto lo llamo la fase de exploración. Después en la fase de planificación se estiman los tiempos para esta lista de requisitos. Más adelante es donde empiezan las iteraciones, donde se desarrollará el proyecto. La primera de las iteraciones tiene menos tiempo de ejecución, ya que, al ser iteraciones de dos semanas y tener que incluir las fases previas de exploración y planificación, le roba un poco de tiempo. En la planificación original solo se tienen en cuenta dos iteraciones para el desarrollo de la aplicación, en caso de variar, se modificará en la planificación final.

La metodología para la parte de desarrollo de trabajo se basará en iteraciones de dos semanas que terminan en los jueves que es cuando son las reuniones de seguimiento con el tutor de este trabajo para enseñarle el progreso. Cada iteración se dividirá en: análisis, diseño, pruebas, puesta en producción y la propia reunión. Las iteraciones tienen como objetivo obtener un producto que funcione y que implemente los requisitos desarrollados en la fase de exploración. Al iniciar una iteración se crearán las historias de usuario a partir de los requisitos.

El desarrollo de trabajo se inicia inmediatamente después del desarrollo de la aplicación, aunque esto no es totalmente cierto, ya que determinadas secciones se irán haciendo al mismo tiempo sobre la aplicación como sobre el trabajo, pero a la hora de crear la estructura de descomposición del trabajo, resulta más sencillo.

Las secciones del desarrollo del trabajo no son más que las diferentes subsecciones que incluye esta plantilla.

4.2 Presupuesto Inicial

El presupuesto se divide en 3 secciones simples; el desarrollo de la aplicación, el desarrollo del trabajo y los recursos materiales.

El desarrollo de la aplicación se corresponde con programar la aplicación y el desarrollo del trabajo con desarrollar este documento. Los recursos materiales son aquellos que necesitamos para desarrollar el proyecto.

A continuación, se ve un resumen del presupuesto teniendo en cuenta las 3 secciones, el beneficio de un 25%, el IVA del 21% y posibles sobrecostes

Resumen del presupuesto		
Código Partida	**Partida**	**Total**
1	Desarrollo de la Aplicación	7.469,15 €
2	Desarrollo del Trabajo	4.361,97 €
3	Recursos materiales	1.460,50 €
Total		**13.291,62 €**

Figura 4. 3. Resumen del presupuesto

4.2.1 Desarrollo de Presupuesto Detallado (Empresa)

A continuación, pasaremos a desarrollar el presupuesto de costes de forma detallada. Empezaremos por el desarrollo de la aplicación, que se compone de 4 subsecciones que son, la fase de exploración, la fase de planificación y las dos iteraciones originalmente planeadas.

Ha de comentarse que el presupuesto ha sido realizado por un solo desarrollador, que es el propio alumno, su sueldo se ve reflejado en todas las tablas del desarrollo de la aplicación y desarrollo del trabajo en precio, que es el precio hora. Este cálculo se extrae de la siguiente tabla:

Persona	Coste real a la empresa	Sueldo bruto	Sueldo neto	Coste/hora para la empresa	Horas totales	Coste TOTAL del trabajador por este proyecto
Álvaro G. Infante	39000	30000	23157	18,75	345	6468,75
TOTAL						**6468,75**

Figura 4. 4. Cálculo del sueldo

El sueldo bruto de 30.000 euros es extraído de diferentes medias de sueldo de ingeniero de software en España. El sueldo neto es lo que percibiría el trabajador al año realmente.

Luego, el coste real a la empresa sale del sueldo bruto más un 30% que costaría a la empresa por las cuotas a la seguridad social y diferentes retenciones del IRPF. Todo esto resulta en un total de 39.000 euros al año, que dividido entre 52 semanas que tiene el año y 40 horas de trabajo semanal daría lugar a un sueldo/hora de 18,75.

Ahora se pasa a desarrollar las diferentes fases que componen el presupuesto detallado. La siguiente tabla muestra la fase de exploración de la planificación:

Fase de exploración				
Indice	**Descripción**	**Horas**	**Precio**	**Total**
1	Investigar sobre DSL	12	18,75	225
2	Investigar el dominio del problema	12	18,75	225
3	Buscar sistemas similares	12	18,75	225
4	Determinar el software a utilizar	12	18,75	225
5	Creación de lista de requisitos priorizada	12	18,75	225
Total				**1125**

Figura 4. 5. Fase de exploración

A continuación, se ve la fase de planificación:

Fase de planificación				
Indice	**Descripción**	**Horas**	**Precio**	**Total**
1	Estimar tiempos y fechas	10	18,75	187,5
2	Plasmar las estimaciones en el presupuesto y planificación	6	18,75	112,5
Total				**300**

Figura 4. 6. Fase de planificación

Las dos siguientes tablas muestran las dos iteraciones planificadas. Se decide dos iteraciones porque equivalen a un mes de desarrollo y parece suficiente tiempo para desarrollar la aplicación:

Iteración 1				
Indice	**Descripción**	**Horas**	**Precio**	**Total**
1	Análisis	8	18,75	150
2	Diseño	6	18,75	112,5
3	Desarrollo	28	18,75	525
4	Pruebas	22	18,75	412,5
5	Puesta en producción	6	18,75	112,5
6	Reunión 1	1	18,75	18,75
Total				**1331,25**

Figura 4. 7. Iteración I

Iteración 2				
Indice	**Descripción**	**Horas**	**Precio**	**Total**
1	Análisis	8	18,75	150
2	Diseño	6	18,75	112,5
3	Desarrollo	28	18,75	525
4	Pruebas	22	18,75	412,5
5	Puesta en producción	6	18,75	112,5
6	Reunión 2	1	18,75	18,75
Total				**1331,25**

Figura 4. 8. Iteración II

Luego tenemos el desarrollo del trabajo, cuyas subsecciones coinciden con las secciones de este mismo trabajo, aquí se ven las diferentes tablas que lo componen:

Memoria del proyecto				
Indice	**Descripción**	**Horas**	**Precio**	**Total**
1	Resumen de la motivación, objetivos y alcance del proyecto	2	18,75	37,5
2	Resumen de todos los aspectos	2	18,75	37,5
3	Otros apartados	2	18,75	37,5
Total				**112,5**

Figura 4. 9. Memoria del proyecto

Introducción				
Indice	**Descripción**	**Horas**	**Precio**	**Total**
1	Justificación del proyecto	2	18,75	37,5
2	Objetivos del proyecto	2	18,75	37,5
3	Estudio de la situación actual	1	18,75	18,75
Total				**93,75**

Figura 4. 10. Introducción

Aspectos teóricos				
Indice	**Descripción**	**Horas**	**Precio**	**Total**
1	Concepto 1	2	18,75	37,5
2	Concepto 2	2	18,75	37,5
3	Concepto 3	2	18,75	37,5
Total				**112,5**

Figura 4. 11. Aspectos teóricos

Planificación del proyecto y resumen del presupuesto				
Indice	**Descripción**	**Horas**	**Precio**	**Total**
1	Planificación	2	18,75	37,5
2	Resumen del presupuesto	2	18,75	37,5
Total				**75**

Figura 4. 12. Planificación del proyecto y resumen del presupuesto

Análisis				
Indice	**Descripción**	**Horas**	**Precio**	**Total**
1	Definición del sistema	2	18,75	37,5
2	Requisitos del sistema	6	18,75	112,5
3	Identificación de los subsistemas en la fase de análisis	4	18,75	75
4	Diagrama de clases preliminar del análisis	4	18,75	75
5	Análisis de casos de uso y escenarios	4	18,75	75
6	Análisis de interfaces de usuarios	6	18,75	112,5
7	Especificación del plan de pruebas	2	18,75	37,5
Total				**525**

Figura 4. 13. Análisis

Diseño del sistema				
Indice	**Descripción**	**Horas**	**Precio**	**Total**
1	Arquitectura del sistema	2	18,75	37,5
2	Diseño de clases	6	18,75	112,5
3	Diagramas de interacción y estados	4	18,75	75
4	Diagrama de actividades	4	18,75	75
5	Diseño de la base de datos	4	18,75	75
6	Diseño de la interfaz	6	18,75	112,5
7	Especificación técnica del plan de pruebas	2	18,75	37,5
Total				**525**

Figura 4. 14. Diseño del sistema

Implementación del sistena				
Indice	**Descripción**	**Horas**	**Precio**	**Total**
1	Estándares y normas seguidos	2	18,75	37,5
2	Lenguajes del programación	2	18,75	37,5
3	Herramientas y programas usados para el desarrollo	2	18,75	37,5
4	Creación del sistema	4	18,75	75
Total				**187,5**

Figura 4. 15. Implementación del sistema

Desarrollo de las pruebas				
Indice	**Descripción**	**Horas**	**Precio**	**Total**
1	Pruebas unitarias	2	18,75	37,5
2	Pruebas de integración y del sistema	2	18,75	37,5
3	Pruebas de usabilidad y accesibilidad	4	18,75	75
4	Pruebas de rendimiento	2	18,75	37,5
Total				**187,5**

Figura 4. 16. Desarrollo de las pruebas

Manuales del sistema				
Indice	**Descripción**	**Horas**	**Precio**	**Total**
1	Manual de instalación	2	18,75	37,5
2	Manual de ejecución	2	18,75	37,5
3	Manual de usuario	2	18,75	37,5
4	Manual del programador	2	18,75	37,5
Total				**150**

Figura 4. 17. Manuales del sistema

Conclusiones y ampliaciones				
Indice	**Descripción**	**Horas**	**Precio**	**Total**
1	Conclusiones	2	18,75	37,5
2	Ampliaciones	2	18,75	37,5
Total				**75**

Figura 4. 18. Conclusiones y ampliaciones

Presupuesto				
Indice	**Descripción**	**Horas**	**Precio**	**Total**
1	Desarrollo de presupuesto detallado	2	18,75	37,5
Total				**37,5**

Figura 4. 19. Presupuesto

Referencias bibliográficas				
Indice	**Descripción**	**Horas**	**Precio**	**Total**
1	Libros y artículos	2	18,75	37,5
2	Referencias en internet	2	18,75	37,5
Total				**75**

Figura 4. 20. Referencias bibliográficas

Apéndices				
Indice	**Descripción**	**Horas**	**Precio**	**Total**
1	Glosario y diccionario de datos	2	18,75	37,5
2	Contenido entregado en el archivo adjunto	6	18,75	112,5
3	índice alfabético	2	18,75	37,5
4	Código fuentes	2	18,75	37,5
Total				**225**

Figura 4. 21. Apéndices

Como última sección del presupuesto de costes están los costes materiales, que son aquellos gastos que se deben incluir en el presupuesto debido a que son consecuencia directa o indirecta del desarrollo de dicho proyecto. Las unidades de esta sección exigen una explicación, para los materiales físicos queda claro que una unidad equivale a una unidad física del producto, 100 unidades de folios A4 equivalen a 100 folios A4. Pero, por ejemplo, las licencias de software se refieren a una licencia software de ese producto, como sería Microsoft Office 365 y el IDE PyCharm. En cuanto a los costes materiales no tangentes como serían la electricidad, el agua, la limpieza o el internet las unidades se refieren a meses, siendo el coste asignado a ellos el coste mensual:

Objeto	Coste	Unidades	Subtotal
Microsoft Office 365	69	1	69,00 €
Lenovo YOGA 530	150	1	150,00 €
Folios A4	0,0241	100	2,41 €
Disco duro TOSHIBA 1TB	64,99	1	64,99 €
IDE PyCharm	199	1	199,00 €
Electricidad	36	3	108,00 €
Internet	29,9	3	89,70 €
Agua	33,1	3	99,30 €
Limpieza	5	3	15,00 €
Bolígrafo BIC	0,29	6	1,74 €
TOTAL			**799,14 €**

Figura 4. 22. Costes materiales

El precio del portátil esta amortizado, es decir, el dispositivo no costó la cantidad que ahí se muestra, pero debido a que solo se va a usar el portátil por un tiempo determinado y calculando lo que costó, obtenemos el precio que cuesta para el tramo de tiempo que se va a usar en el proyecto. Se calculó un quinto del precio original, a pesar de que el tiempo de desarrollo de la aplicación y de este trabajo no corresponde con un quinto de la vida útil del ordenador, sino que se tiene en cuenta otros factores como que el ordenador se estropee.

En el caso de electricidad, agua e internet se ve que se multiplican los precios mensuales por 3, suponiendo según la estimación inicial del proyecto que la realización del trabajo lleve cerca de 3 meses en realizar.

Finalmente, el resumen de este presupuesto de costes sería la siguiente tabla:

Presupuesto de costes				
Código Partida	**Item**	**Partida**	**Importe**	**Total**
1		Desarrollo de la Aplicación		4.087,50 €
	1	Fase de Exploración	1.125,00 €	
	2	Fase de Planificación	300,00 €	
	3	Iteración 1	1.331,25 €	
	4	Iteración 2	1.331,25 €	
2		Desarrollo del Trabajo		2.381,25 €
	1	Memoria del proyecto	112,50 €	
	2	Introducción	93,75 €	
	3	Aspectos teóriticos	112,50 €	
	4	Planificación del proyecto y resumen del presupuesto	75,00 €	
	5	Análisis	525,00 €	
	6	Diseño del sistema	525,00 €	
	7	Implementación del sistema	187,50 €	
	8	Desarrollo de las pruebas	187,50 €	
	9	Manuales del sistema	150,00 €	
	10	Conclusiones y ampliaciones	75,00 €	
	11	Presupuesto	37,50 €	
	12	Referencias bibliográficos	75,00 €	
	13	Apéndices	225,00 €	
3		Recursos materiales		799,14 €
Subtotal				**7.267,89 €**

Figura 4. 23. Presupuesto de costes

Esto no es el presupuesto final, ya que se necesitarán aplicar diversas tasas al presupuesto, como posibles sobrecostes, el beneficio y el IVA, como se detalla en la siguiente tabla:

Resumen del presupuesto	
Partida	**Total**
Presupuesto de costes	7.267,89 €
Beneficio(25%)	1.816,97 €
IVA(21%)	1.526,26 €
Posibles sobrecostes	2.662,50 €
Total	**13.273,62 €**

Figura 4. 24. Resumen del presupuesto

4.2.2 Desarrollo de Presupuesto Simplificado (Cliente)

La siguiente versión del presupuesto sería la versión simplificada o el presupuesto del cliente:

Presupuesto del cliente				
Código Partida	**Item**	**Partida**	**Importe**	**Total**
1		Desarrollo de la Aplicación		7.469,15 €
	1	Fase de Exploración	2.055,63 €	
	2	Fase de Planificación	548,90 €	
	3	Iteración 1	2.432,31 €	
	4	Iteración 2	2.432,31 €	
2		Desarrollo del Trabajo		4.361,97 €
	1	Memoria del proyecto	206,46 €	
	2	Introducción	172,22 €	
	3	Aspectos teóricos	206,46 €	
	4	Planificación del proyecto y resumen del presupuesto	137,98 €	
	5	Análisis	959,83 €	
	6	Diseño del sistema	959,83 €	
	7	Implementación del sistema	343,44 €	
	8	Desarrollo de las pruebas	343,44 €	
	9	Manuales del sistema	274,95 €	
	10	Conclusiones y ampliaciones	137,98 €	
	11	Presupuesto	69,49 €	
	12	Referencias bibliográficos	137,98 €	
	13	Apéndices	411,93 €	
3		Recursos materiales		1.460,50 €
Total				**13.291,62 €**

Figura 4. 25. Presupuesto del cliente

También podría estar más resumido aún, como se muestra en la siguiente tabla:

Resumen del presupuesto		
Código Partida	**Partida**	**Total**
1	Desarrollo de la Aplicación	7.469,15 €
2	Desarrollo del Trabajo	4.361,97 €
3	Recursos materiales	1.460,50 €
Total		**13.291,62 €**

Figura 4. 26. Resumen del presupuesto

Capítulo 5. Análisis

A continuación, detallaremos la especificación de requisitos, la documentación del análisis de la aplicación, a partir de la cual se elaborará posteriormente el diseño.

5.1 Definición del Sistema

Definiremos ahora los límites del sistema, es decir, hasta donde vamos a llegar en su construcción.

5.1.1 Determinación del Alcance del Sistema

El sistema será una aplicación funcional que permita mejorar la seguridad de servidores Apache en tres sistemas operativos, Windows, Ubuntu y CentOS. Los comandos añadidos para los tres sistemas se basan en la guía CIS Benchmark de Apache. El usuario podría crear o importar más comandos a la aplicación o incluso modificar los archivos JSON para añadir más funcionalidades

5.2 Requisitos del Sistema

5.2.1 Obtención de los Requisitos del Sistema

Estos son los requisitos del sistema. Primero los requisitos funcionales:

Código	**Nombre Requisito**	**Descripción del Requisito**
RF1	Interpretar lenguaje específico	El sistema debe permitir leer la entrada de la consola de comandos en el lenguaje específico
RF2	Determinar el OS	La aplicación debe determinar el sistema en el que está siendo ejecutado
RF2.1		La aplicación debe determinar si el sistema operativo es Windows
RF2.2		La aplicación debe determinar si el sistema operativo es Ubuntu
RF2.3		La aplicación debe determinar si el sistema operativo es CentOS
RF3	Uso de archivos JSON	El sistema interacciona con archivos JSON para usar los comandos
RF3.1		El sistema debe usar diferentes archivos dependiendo del sistema operativo
RF4	Comprobar gramática	El sistema debe comprobar la gramática del lenguaje específico
RF4.1		El sistema debe lanzar excepciones en caso de encontrar errores
RF4.2		El sistema cogerá las excepciones, mostrando al usuario la forma de arreglarlo
RF5	Historial de comandos	El sistema ha de contar con un historial de comandos
RF5.1		El sistema contará con un archivo físico que guardará el historial de comandos
RF5.2		El usuario podrá acceder al historial a través de la consola de comandos con las flechas del teclado

RF6	Ayuda en la gramática	El sistema debe permitir que los comandos sean previsibles
RF6.1		El sistema consigue la previsibilidad a través de varios medios
RF6.1.1		El sistema cuenta con autocompletar en los comandos a través de la tecla TAB
RF6.1.2		El sistema muestra en los errores de gramática el término que más se le parece de los comandos
RF6.1.3		El sistema no distingue mayúsculas de minúsculas
RF7.1	Importar comandos	El sistema debe permitir importar comandos entre diferentes archivos JSON
RF7.1		El sistema importa de un archivo JSON externo al propio del sistema
RF8	Exportar comandos	El sistema debe permitir exportar comandos entre diferentes archivos JSON
RF8.1		El sistema exporta desde los comandos del propio sistema al archivo externo
RF9	Crear comandos	EL sistema debe permitir crear nuevos comandos que se inserten en el archivo JSON de comandos
RF10	Configuración base	El sistema debe tener una opción de configuración básica para correr varios comandos al mismo tiempo
RF11	Internacionalización	El sistema debe estar internacionalizado
RF11.1		El sistema debe permitir el idioma inglés
RF11.2		El sistema debe permitir el idioma español
RF12	Establecer el idioma	El sistema debe permitir establecer el idioma que se desea en la aplicación
RF12.1		El sistema debe permitir establecer el idioma español
RF12.2		El sistema debe permitir establecer el idioma inglés

Figura 5. 1. Requisitos funcionales

Luego los requisitos de usuario:

Código	**Nombre Requisito**	**Descripción del Requisito**
RU1	Configuración de Apache	El usuario debe tener un servidor apache desplegado
RU2	Archivos de configuración	El usuario debe modificar el archivo de configuración, existe uno por sistema operativo
RU2.1		El usuario debe modificar la variable APACHE para todos los sistemas operativos
RU2.2		El usuario debe modificar la variable PASSWORD para todos los sistemas operativos

Figura 5. 2. Requisitos de usuario

Finalmente, los requisitos tecnológicos:

Código	**Nombre Requisito**	**Descripción del Requisito**
RT1	Multiplataforma	El sistema debe ser multiplataforma
RT1.1		Uno de los sistemas operativos debe ser Windows
RT1.2		Uno de los sistemas operativos debe ser Ubuntu
RT1.3		Uno de los sistemas operativos debe ser CentOS

Figura 5. 3. Requisitos

5.2.2 Identificación de Actores del Sistema

En nuestro sistema podemos identificar varios actores. El primero y más obvio sería el usuario base, el usuario que corre la aplicación, sea en el sistema operativo que sea. Ha de comentarse que para la ejecución de algunos comandos se necesita ser administrador, por lo que puede ser que el sistema pregunte por claves de administrador, aunque esto no lo consideramos un rol diferente, ya que es parte del sistema externo.

El usuario interactuará con el sistema a través de una interfaz de comandos, en ella escribirá las órdenes definidas en el lenguaje específico.

Otros actores del sistema podrían ser los subsistemas que corren los comandos, tenemos dos diferentes. El primero sería el PowerShell de Windows que usaría los comandos en caso de estar ejecutando la aplicación en Windows. El otro sería el Shell de Linux que haría lo mismo que el anterior, pero para Linux.

5.2.3 Especificación de Casos de Uso

A continuación, se especifican todos los casos de uso, con su correspondiente diagrama.

Nombre del Caso de Uso
Ayuda sobre el sistema
Descripción
Un usuario que sea nuevo en el sistema necesitará algún tipo de guía. Mediante el comando *help* se mostrarán las diferentes opciones que el usuario puede ejecutar en el sistema. Existe una alternativa a este comando que sería usando el comando *help comando*, siendo comando el nombre del comando del que se quiere información. Como alternativa de fallo en el comando de ayuda solo existe que se escriba incorrectamente el comando de ayuda o el nombre del comando, en ese caso se informará al usuario del error y de la correcta gramática.

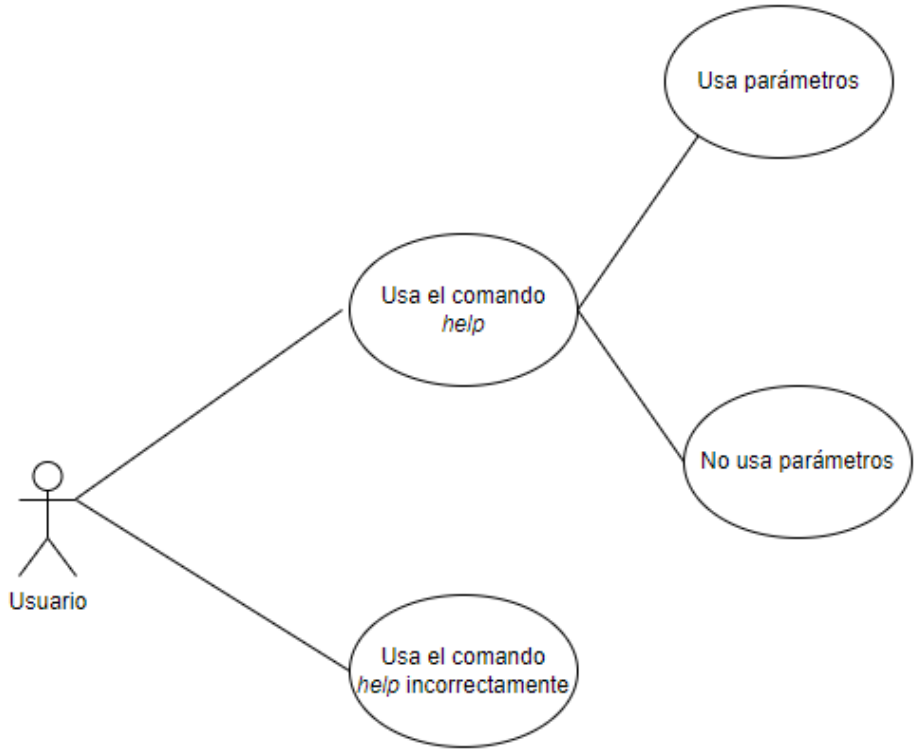


Figura 5. 4. Ayuda sobre el sistema

Nombre del Caso de Uso

Usar un comando del sistema
Descripción
El usuario escribirá el nombre del paquete y el comando, en caso de estar correctamente escrito, se mostrará al usuario cual es la salida que se espera del comando (en caso de haberla) y se ejecutará el comando. Como fallo, en caso de que se escriba incorrectamente, se informará al usuario del error y de la correcta gramática.

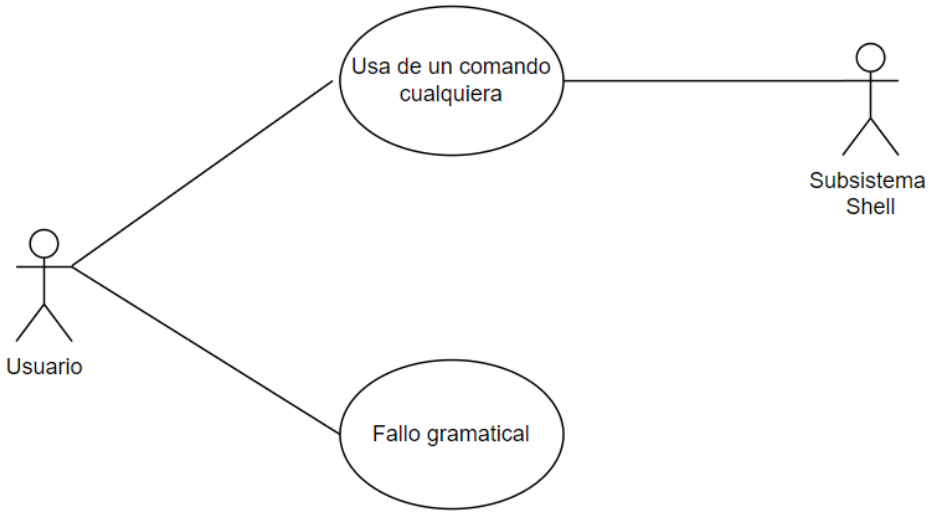


Figura 5. 5. Usar un comando del sistema

Nombre del Caso de Uso
Crear un nuevo comando para el sistema

Descripción
El usuario escribirá el comando *create nombre*, siendo nombre el nombre del comando. Después de esto se mostrará al usuario los distintos campos que tiene que rellenar del comando para que sea válido. Al igual que en los demás comandos en caso de fallo gramatical se informará al usuario del error y de cuál es la correcta gramática.

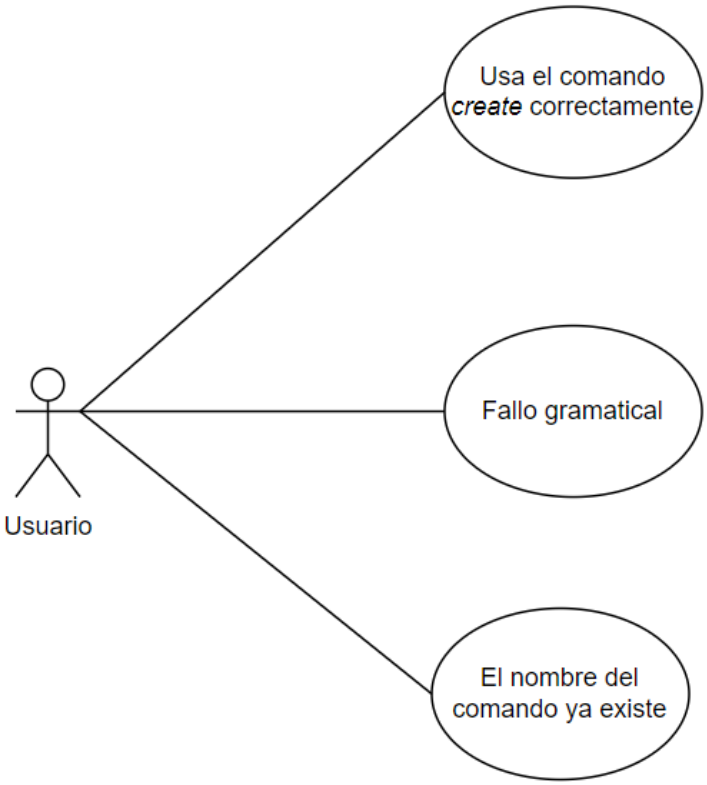


Figura 5. 6. Crear un nuevo comando del sistema

Nombre del Caso de Uso
Importar un comando del sistema
Descripción
El usuario escribirá el comando *import from archivo [comandos]*. Siendo comandos los comandos que quiere importar en nuestro archivo, o dejarlo vacío si quiere que se importen todos los comandos. Y siendo archivo el nombre del archivo JSON. Al igual que en los demás comandos en caso de fallo gramatical se informará al usuario del error y de cuál es la correcta gramática. Si el archivo del que se importan los comandos no existe se informará al usuario de ello.

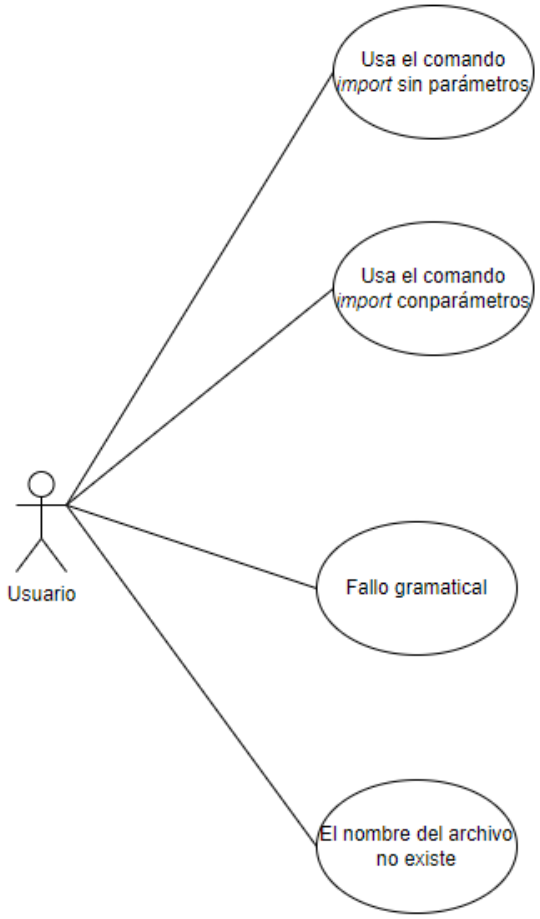


Figura 5. 7. Importar un comando del sistema

Nombre del Caso de Uso
Exportar un comando al sistema

Descripción
El usuario escribirá el comando *export [comandos] into archivo*. Siendo comandos los comandos que quiere exportar en el archivo, o dejarlo vacío si quiere que se exporten todos los comandos. Y siendo archivo el nombre del archivo JSON. Al igual que en los demás comandos en caso de fallo gramatical se informará al usuario del error y de cuál es la correcta gramática.

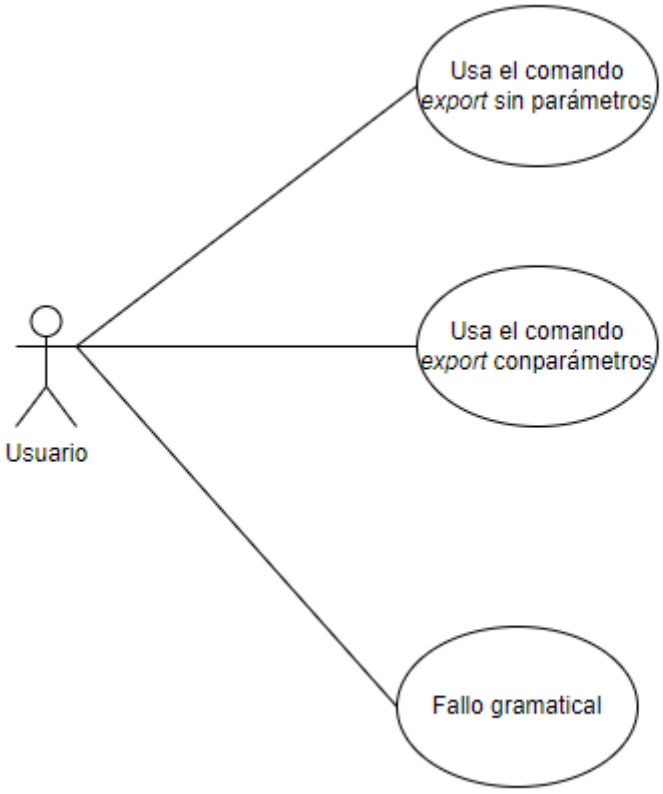


Figura 5. 8. Exportar un comando al sistema

Nombre del Caso de Uso
Correr varios comandos al mismo tiempo en una configuración básica
Descripción
El usuario podrá correr una configuración básica de comandos basada en la guía CIS Apache que define dos niveles. El usuario usará el comando run y dos opciones: - Level1 - Level2 Al igual que en los demás comandos en caso de fallo gramatical se informará al usuario del error y de cuál es la correcta gramática.

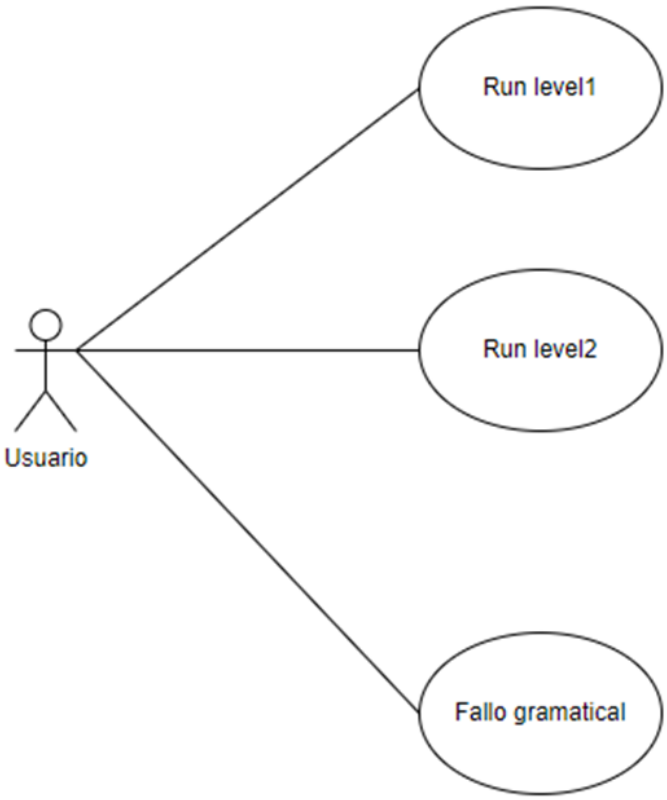


Figura 5. 9 Correr varios comandos al mismo tiempo en una configuración básica

Nombre del Caso de Uso
Establecer el idioma

Descripción
El usuario podrá cambiar el idioma a español o inglés con el comando *set language IDIOMA*, siendo IDIOMA el idioma. En caso de especificar otro idioma que no sea inglés o español, se pondrá inglés. Al igual que en los demás comandos en caso de fallo gramatical se informará al usuario del error y de cuál es la correcta gramática.

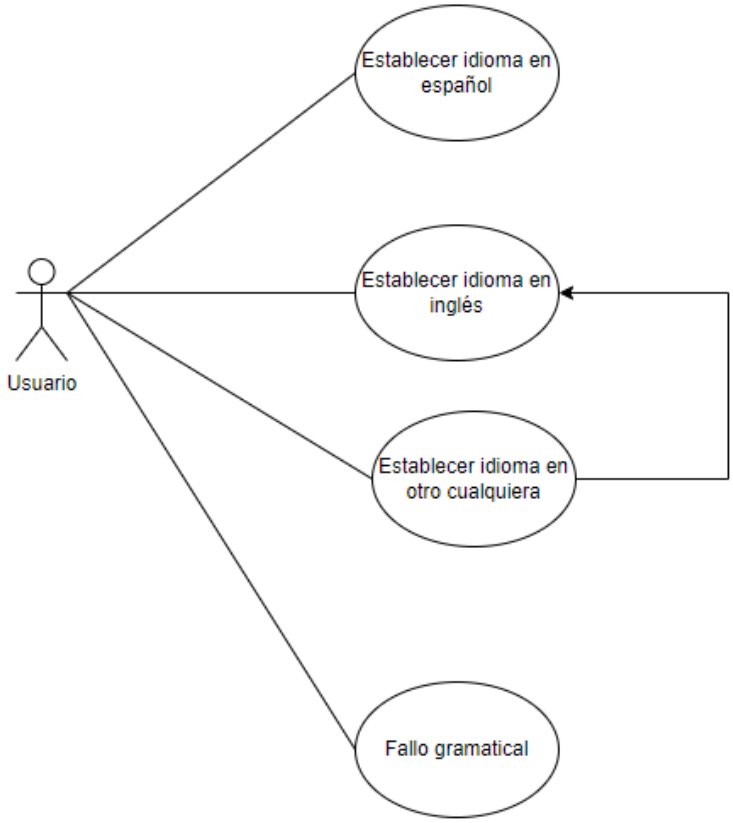


Figura 5. 10. Establecer el idioma

5.3 Identificación de los Subsistemas en la Fase de Análisis

Aquí pasaremos a analizar el sistema y dividirlo en las distintas subsecciones que lo componen.

5.3.1 Descripción de los Subsistemas

Dentro de nuestro sistema encontramos distintos subsistemas, que pasaremos a comentar ahora individualmente:

- Interpretación del lenguaje: mediante el lenguaje específico definido, este subsistema se encargará de interpretar las órdenes escritas por el usuario. Además, también comprobará que la gramática es correcta, antes de ser ejecutado. En el caso de que la gramática no sea correcta, sugerirá al usuario lo más parecido a la orden que haya escrito. Además, en este campo de sugerencia, el subsistema se encarga de auto completar un comando, si el usuario usa la letra del teclado TAB, esto está inspirado en muchos sistemas de interfaz de comandos.
- Ejecución del lenguaje: una vez se comprueba que el lenguaje no tiene errores gramaticales, se ejecutará. Este subsistema se encarga de comprobar cuál es el sistema operativo que está corriendo la aplicación y donde debe ejecutar el comando. También se encarga de mostrar la salida del comando y todo lo necesario para que el usuario comprenda lo que está pasando. Además de los comandos, existen otras opciones dentro del sistema, como explicamos en el punto 5.2.3, que también están cubiertas por este subsistema.
- Interacción con archivos tipo JSON: el sistema guarda todos los comandos de los diferentes sistemas operativos en archivos de tipo JSON. Es por eso por lo que para que las partes de interpretación y ejecución del lenguaje funcionen, deben comunicarse con estos archivos. Este subsistema se encarga de la lectura, escritura y modificación de estos archivos en relación con los comandos.
- Establecer idioma de la aplicación: este subsistema es pequeño, pero debe ser mencionado. Es el que determina cual es el idioma de la aplicación dado el idioma que usa el sistema operativo que corre la aplicación. Esta internacionalizado para español e inglés y permite a través de un comando que se cambie a uno u a otro.

Este es el diagrama de subsistemas:

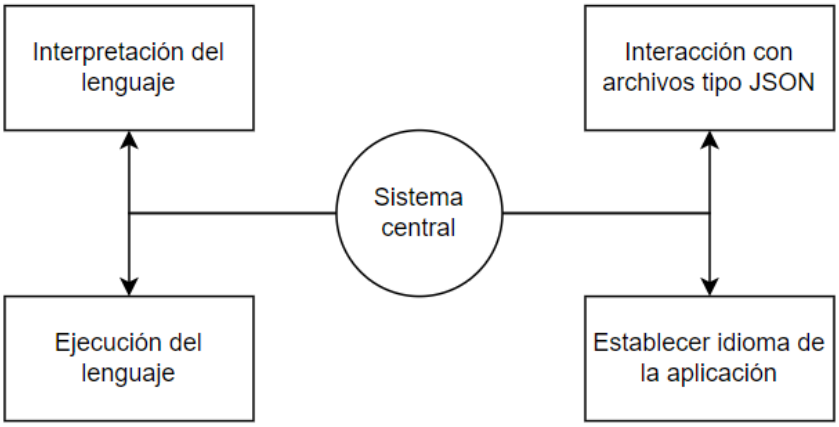


Figura 5. 11. Diagrama de subsistemas

5.4 Diagrama de Clases Preliminar del Análisis

5.4.1 Diagrama de Clases

El diagrama de clases de la aplicación será el siguiente:

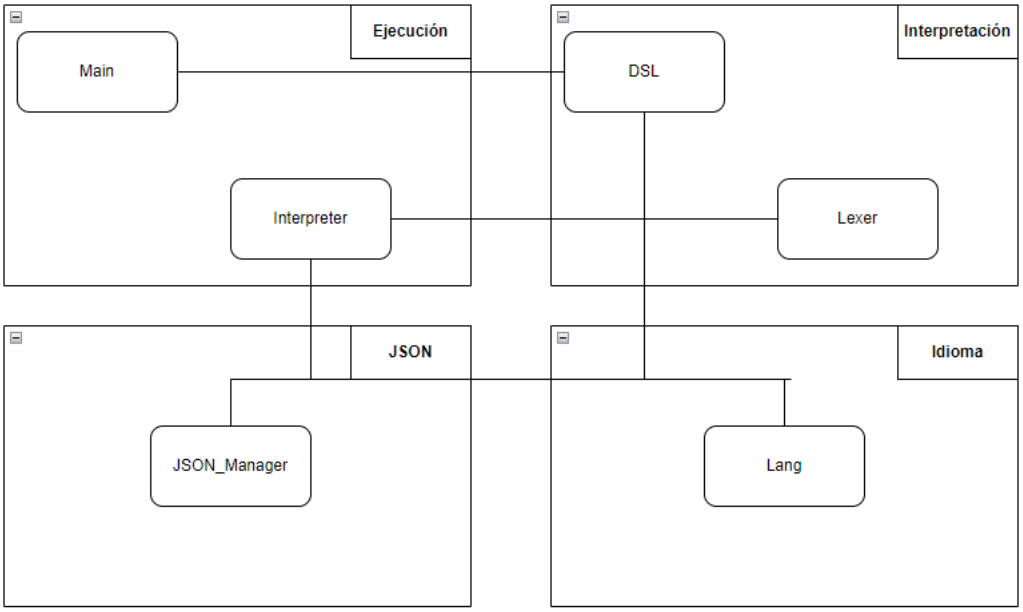


Figura 5. 12. Diagrama de clases

5.4.2 Descripción de las Clases

Las clases se organizan por los diferentes subsistemas identificados en la anterior sección siendo estos: interpretación del lenguaje, ejecución del lenguaje, cambio de idioma y la interacción con archivos de tipo JSON. A continuación, analizamos todas las clases dentro de su subsistema.

5.4.2.1 Interpretación del lenguaje

Nombre de la Clase
DSL
Descripción
Esta clase se usa para conectar la aplicación y hacer llamadas a todas las diferentes partes que se usan en el proyecto.
Responsabilidades
Controla la entrada del usuario y llama a la clase *Lexer* para que compruebe la gramática
Atributos Propuestos
Ninguno
Métodos Propuestos
Main_loop: Es el método principal, llamara a read_input() y a métodos de la clase *Lexer* Read_input: Lee la entrada del usuario por líneas y llama a get_input() Get_input: Es una versión del método nativo de Python input(), la diferencia principal con el método input es que este no para la ejecución de la máquina. Necesitamos que no se pare la ejecución de la máquina para la predicción de texto con la letra TAB.

Nombre de la Clase
Lexer
Descripción
Este módulo se usa para comprobar que la gramática es correcta e informar al usuario en caso contrario.
Responsabilidades
Comprobar la gramática y los argumentos que el usuario introduzca.
Atributos Propuestos
History: Historial de comandos de la aplicación en un array First_Methods_List: Array de String que son contiene los nombres de los métodos
Métodos Propuestos
Check_grammar: Comprobará todos los términos introducidos por el usuario, añadirá los comandos al historial de comandos y llamará al método check_parts y a la clase *interpreter* en caso de que no haya errores gramaticales. Check_parts: Comprobará que todas las ordenes son escritas correctamente, sino lanzará una excepción de tipo *assertion* con el mensaje que se debe mostrar al usuario. Llamará a diferentes comprobaciones individuales del sistema. Predict: Predecirá el termino al que se refiere el usuario

5.4.2.2 Ejecución del lenguaje

Nombre de la Clase
Main
Descripción
Clase Main de la aplicación
Responsabilidades
Llamar a DSL para ejecutar la aplicación
Atributos Propuestos
Ninguno
Métodos Propuestos
Main: Llama a la clase DSL para iniciar el bucle

Nombre de la Clase
Interpreter
Descripción
Esta clase se usa para ejecutar las órdenes
Responsabilidades
Una vez la gramática ha sido comprobada, esta clase ejecutará las órdenes.
Atributos Propuestos
Ninguno
Métodos Propuestos
Use_module: Este es el principal método para la clase interprete, su función es ver cuál es comando que se le pasa como parámetro y ejecutarlo correctamente

5.4.2.3 Cambio de idioma

Nombre de la Clase
Lang
Descripción
Es una clase de utilidades varias
Responsabilidades
Guardar la internacionalización de los mensajes del sistema y detectar el idioma
Atributos Propuestos
Os: Sistema operativo que corre la aplicación
Métodos Propuestos
Detect_Language: Determina cual es el idioma del sistema operativo Get_message Devuelve los mensajes solicitados en el idioma correcto Check_os: Comprueba el sistema operativo Check_linux_os: Comprueba la distribución de Linux que se va a usar

5.4.2.4 Interacción con archivos tipo JSON

Nombre de la Clase
JSON manager
Descripción
Este módulo se usa para trabajar con los archivos de tipo JSON
Responsabilidades
Hacer de puente entre los archivos y la aplicación para leer y modificar archivos
Atributos Propuestos
File: nombre del archivo JSON que almacena los comandos
Métodos Propuestos
Get_modules: Extraer los comandos del archivo JSON. Conf: Se usa para conseguir del archivo de configuración los enlaces específicos a la carpeta Apache. Import_into: Para importar comandos del comando *import*. Export_into: Para exportar comandos del comando *export*. Write_new_command: Se usa para introducir comandos del comando *create*.

5.5 Análisis de Casos de Uso y Escenarios

Se analizan a continuación los casos de uso del sistema:

5.5.1 Ayuda sobre el sistema

Ayuda sobre el Sistema	
Precondiciones	No
Postcondiciones	Se muestra un mensaje de ayuda
Actores	Usuario
Descripción	El usuario: 1. Escribirá la orden *help* 2. Pulsará ENTER dos veces 3. Se imprimirá en pantalla la descripción general de ayuda
Variaciones (escenarios secundarios)	• **Escenario alternativo 1:** Se comete algún error gramatical, lo que hará mostrar un mensaje de error y una predicción de lo que el usuario quizá quisiera decir • **Escenario alternativo 2:** Se introduce como parámetro en el paso 1 cualquier nombre de comando. En ese caso se vuelve al paso 2 y en el paso 3 se imprimirá la descripción de ayuda de dicho comando.
Excepciones	• **Error gramatical:** Lanza una excepción de tipo Assertion que luego se refleja en un mensaje al usuario informando del error.
Notas	Ninguna

5.5.2 Usar un comando del sistema

Usar un comando del sistema	
Precondiciones	No
Postcondiciones	Se muestra la salida esperada y la salida del comando ejecutado
Actores	Usuario, Shell del sistema operativo
Descripción	El usuario: 1. Escribirá la orden del modo *paquete comando* (siendo comando el nombre del comando y paquete el paquete en el que está metido el comando) 2. Pulsará ENTER dos veces 3. Se imprimirá en pantalla el comando a ejecutar, la salida del comando si tiene y la salida esperada
Variaciones (escenarios secundarios)	• **Escenario alternativo 1:** Se comete algún error gramatical, lo que hará mostrar un mensaje de error y una predicción de lo que el usuario quizá quisiera decir
Excepciones	• **Error gramatical:** Lanza una excepción de tipo Assertion que luego se refleja en un mensaje al usuario informando del error.
Notas	Ninguna

5.5.3 Crear un nuevo comando para el sistema

Crear un nuevo comando del sistema	
Precondiciones	No
Postcondiciones	Se añade el comando al archivo JSON en un nuevo paquete *Others*
Actores	Usuario
Descripción	El usuario: 1. Escribirá la orden del modo *create comando* (siendo comando el nombre del comando) 2. Pulsará ENTER dos veces 3. El usuario introducirá la descripción del comando 4. El usuario introducirá el comando en si 5. Se imprimirá en pantalla la confirmación de que se ha añadido
Variaciones (escenarios secundarios)	• **Escenario alternativo 1:** Se comete algún error gramatical, lo que hará mostrar un mensaje de error y una predicción de lo que el usuario quizá quisiera decir
Excepciones	• **Error gramatical:** Lanza una excepción de tipo Assertion que luego se refleja en un mensaje al usuario informando del error.
Notas	Ninguna

5.5.4 Importar un comando del sistema

Importar un comando del sistema	
Precondiciones	Debe existir el archivo del que importar los comandos
Postcondiciones	Se importa un comando de otro sistema al nuestro
Actores	Usuario
Descripción	El usuario: 1. Escribirá la orden del modo *import from file comandos* (siendo comando el nombre de los comandos que quiere importar y file el archivo del que importar) 2. Pulsará ENTER dos veces 3. Se imprimirá en pantalla una confirmación y se añadirán los comandos a nuestro archivo JSON en la sección imported
Variaciones (escenarios secundarios)	• **Escenario alternativo 1:** Se comete algún error gramatical, lo que hará mostrar un mensaje de error y una predicción de lo que el usuario quizá quisiera decir • **Escenario alternativo 2:** No se añade ningún comando, solo se pone *import from file*, de este modo se añaden todos los comandos que haya en ese archivo a la aplicación, en la sección imported.
Excepciones	• **Error gramatical:** Lanza una excepción de tipo Assertion que luego se refleja en un mensaje al usuario informando del error.
Notas	Ninguna

5.5.5 Exportar un comando al sistema

Exportar un comando al sistema	
Precondiciones	No
Postcondiciones	Se exporta un comando de nuestro sistema a otro
Actores	Usuario
Descripción	El usuario: 1. Escribirá la orden del modo *export comandos into file* (siendo comando el nombre de los comandos que quiere importar y file el archivo del que importar) 2. Pulsará ENTER dos veces 3. Se imprimirá en pantalla una confirmación y se añadirán los comandos a el archivo JSON en la sección imported
Variaciones (escenarios secundarios)	• **Escenario alternativo 1:** Se comete algún error gramatical, lo que hará mostrar un mensaje de error y una predicción de lo que el usuario quizá quisiera decir • **Escenario alternativo 2:** No se añade ningún comando, solo se pone *export into file*, de este modo se añaden todos los comandos que haya a ese archivo, en la sección imported.
Excepciones	• **Error gramatical:** Lanza una excepción de tipo Assertion que luego se refleja en un mensaje al usuario informando del error.
Notas	Ninguna

5.5.6 Correr varios comandos al mismo tiempo en una configuración básica

Correr varios comandos al mismo tiempo en una configuración básica	
Precondiciones	No
Postcondiciones	Se muestra la ejecución comando a comando
Actores	Usuario, Shell del sistema operativo
Descripción	El usuario: 1. Escribirá la orden del modo *run level* (siendo *level* level1 o level2) 2. Pulsará ENTER dos veces 3. Se imprimirá en pantalla los comando a ejecutar, la salida de los comandos si tiene y la salida esperada de cada comando
Variaciones (escenarios secundarios)	• **Escenario alternativo 1:** Se comete algún error gramatical, lo que hará mostrar un mensaje de error y una predicción de lo que el usuario quizá quisiera decir
Excepciones	• **Error gramatical:** Lanza una excepción de tipo Assertion que luego se refleja en un mensaje al usuario informando del error.
Notas	Ninguna

5.5.7 Establecer el idioma

Establecer el idioma	
Precondiciones	No
Postcondiciones	Se cambia el idioma de la aplicación
Actores	Usuario
Descripción	El usuario: 1. Escribirá la orden del modo *set language idioma* (siendo idioma el lenguaje al que cambiar) 2. Pulsará ENTER dos veces 3. Se cambiará el idioma de la aplicación
Variaciones (escenarios secundarios)	• **Escenario alternativo 1:** Se comete algún error gramatical, lo que hará mostrar un mensaje de error y una predicción de lo que el usuario quizá quisiera decir • **Escenario alternativo 2:** Se pone algún idioma que no sea inglés o español, lo que hace que se ponga inglés y vuelta al paso 3.
Excepciones	• **Error gramatical:** Lanza una excepción de tipo Assertion que luego se refleja en un mensaje al usuario informando del error.
Notas	Ninguna

5.6 Relación Escenarios – Casos de Uso – Requisitos

En la siguiente tabla se muestra de una forma resumida, la relación entre los Escenarios y los Casos de uso.

Casos de uso	**1.1**	**1.2**	**1.3**	**1.4**	**1.5**	**1.6**	**1.7**
Escenario 1.1	X	X	X	X	X	X	X
Escenario 1.1.2	X						
Escenario 1.4.2				X			
Escenario 1.5.2					X		
Escenario 1.7.2							X

Figura 5. 13. Relación escenarios - casos de uso

5.7 Análisis de Interfaces de Usuario

A continuación, se analiza la interfaz del usuario usada en el sistema.

5.7.1 Descripción de la Interfaz

La interfaz usada en el sistema es de tipo CLI o interfaz de línea de comandos. Se elige este tipo de interfaz por ser más adecuada al tipo de aplicación que queremos desarrollar, además la interfaz de línea de comandos es la mejor opción cuando se trabaja en un servidor, ya que las interfaces gráficas consumen muchos más recursos.

5.7.2 Descripción del Comportamiento de la Interfaz

Uno de los objetivos principales de cualquier sistema que use una interfaz de comandos es que sea similar al resto de interfaces de comandos. Es por eso por lo que nuestro sistema tiene las siguientes características:

- Predicción de texto mediante la tecla TAB: Como en muchos sistemas de este tipo la aplicación predice lo que el usuario pretende escribir cuando se pulsa la tecla TAB. En nuestro caso si hay más de una coincidencia se escribe la parte común. Por ejemplo, si las posibilidades son Endure_1 y Endure_2, y está escrito en el momento de pulsar TAB "end" el sistema rellenará hasta formar Endure_.
- Historial de comandos accesible mediante las flechas del teclado: El usuario podrá en cualquier momento de entrada de texto acceder al historial de comandos usando la flecha para arriba del teclado. Pulsando una vez accederá al último registro del historial y de esa manera, podrá subir más en el historial o volver para abajo con la flecha hacia abajo.
- Características de escritura estándar: Todo lo demás del sistema sería como cualquier otro CLI.
- Comando de ayuda help: Para dar ayuda al usuario sobre el uso del sistema como en otros sistemas, podrá usar el sistema *help* de esta forma se muestra una introducción de todas las opciones del sistema. En caso de que el usuario quiera ayuda sobre un comando podría también usar *help nombre*, siendo nombre el nombre del comando. Esta funcionalidad es muy similar a man de Linux.
- Mensajes de error: Los mensajes de error muestran al usuario cual es el error que ha cometido si es que comete alguno.
- Sugerencia en los errores: Aparte de los mensajes de que error has cometido el sistema cuenta con una sugerencia de "Quizás quisiste decir…" que mediante un algoritmo busca la opción más cercana al error y la muestra.

5.7.3 Diagrama de Navegabilidad

Se muestra a continuación el diagrama de navegabilidad de la aplicación:

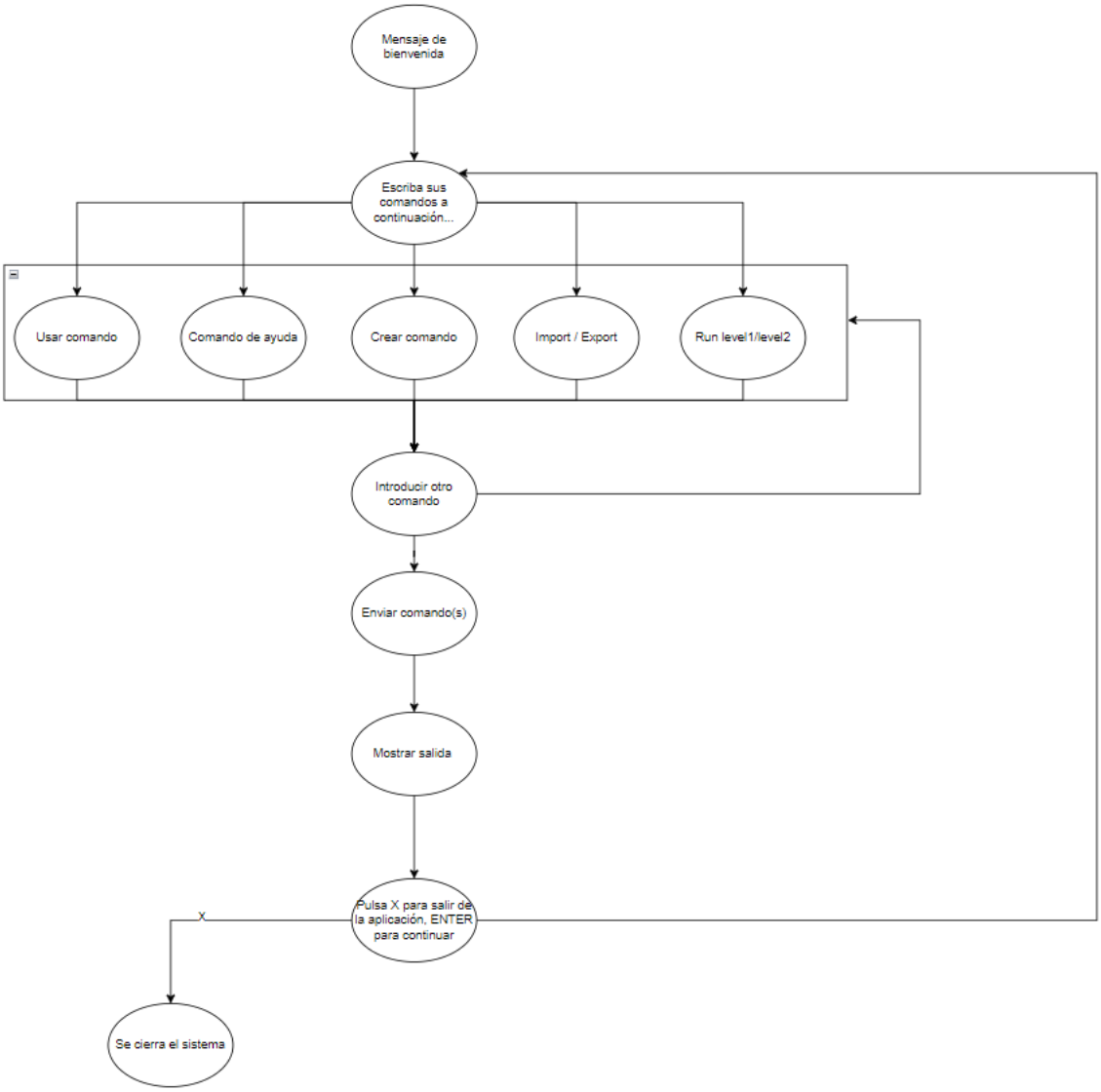


Figura 5. 14. Diagrama de navegabilidad

5.8 Especificación del Plan de Pruebas

En esta sección se crea y diseña un plan de pruebas de la aplicación y sus funciones, así como todos los mecanismos que utilizaremos para detectar errores y corregirlos.

Caso de Uso 1: Ayuda sobre el sistema	
Prueba realizada	**Resultado obtenido**
Usar el comando help solo	El sistema muestra el mensaje de ayuda general
Prueba realizada	**Resultado Esperado**
Usar el comando help nombre, siendo nombre el nombre correcto de un comando	El sistema muestra el mensaje de ayuda de dicho comando
Prueba realizada	**Resultado obtenido**
Escribir mal el comando help, de la forma hepl	El sistema informa de que no reconoce ese comando y que hay muy pocos términos
Prueba realizada	**Resultado obtenido**
Escribir mal el comando help comando, siendo comando el nombre incorrecto de un comando	El sistema informa de que el segundo termino no corresponde con ninguno del lenguaje

Caso de Uso 2: Usar un comando del sistema	
Prueba realizada	**Resultado obtenido**
Usar un comando cualquiera con el nombre del paquete y el comando.	El sistema muestra el comando que se ejecuta, la salida esperada y la salida actual.
Prueba realizada	**Resultado obtenido**
Usar un comando cualquiera con el nombre del paquete y el comando, pero el nombre del paquete no existe	El sistema informa de que no reconoce el comando, además de mostrar cual podría ser el paquete al que se refiere el usuario.
Prueba realizada	**Resultado obtenido**
Usar un comando cualquiera con el nombre del paquete y el comando, pero el nombre del comando no existe	El sistema informa de que no reconoce ese comando y que quizás quisiste decir otro
Prueba realizada	**Resultado obtenido**
Escribir solo el nombre del paquete	El sistema informa de que solo el comando help puede tener tan pocos términos

Caso de Uso 3: Crear un nuevo comando del sistema	
Prueba realizada	**Resultado obtenido**
Usar el comando create nombre, siendo nombre el nombre del comando a crear	El sistema pide que introduzcas la descripción y las ordenes exactas. El sistema añadirá las órdenes al archivo JSON.
Prueba realizada	**Resultado obtenido**
Usar el comando create nombre, siendo nombre el nombre del comando a crear que ya existe	El sistema muestra el mensaje de error informando que el nombre de ese comando ya existe.
Prueba realizada	**Resultado obtenido**
Escribir mal el comando create, de la forma creat	El sistema informa de que no reconoce ese comando y que hay muy pocos términos

Caso de Uso 4: Importar un comando del sistema	
Prueba realizada	**Resultado obtenido**
Usar el comando import comandos from file, siendo comandos los comandos y file el nombre del archivo.	El sistema añade a nuestro sistema los comandos especificados en la sección imported
Prueba realizada	**Resultado obtenido**
Usar el comando import from file, siendo file el nombre del archivo.	El sistema añade a nuestro sistema todos los comandos del archivo en la sección imported
Prueba realizada	**Resultado obtenido**
Escribir mal el comando import	El sistema informa de que no reconoce ese comando.
Prueba realizada	**Resultado obtenido**
Escribir el comando import bien, pero el archivo no existe	El sistema informa de que el archivo no existe

Caso de Uso 5: Exportar un comando del sistema	
Prueba realizada	**Resultado obtenido**
Usar el comando export comandos into file, siendo comandos los comandos y file el nombre del archivo.	El sistema añade a el archivo los comandos especificados
Prueba realizada	**Resultado obtenido**
Usar el comando export into file, siendo file el nombre del archivo.	El sistema añade a el archivo todos los comandos de nuestro archivo
Prueba realizada	**Resultado obtenido**
Escribir mal el comando export	El sistema informa de que no reconoce ese comando.
Prueba realizada	**Resultado obtenido**
Escribir el comando export bien, pero el archivo no existe	El sistema crea un archivo con ese nombre y añade los comandos especificados a la zona imported

Caso de Uso 6: Correr varios comandos al mismo tiempo en una configuración básica	
Prueba realizada	**Resultado obtenido**
Usar el comando run level, siendo level, level1 o level2	El sistema corre uno a uno todos los comandos del nivel especificado
Prueba realizada	**Resultado obtenido**
Escribir mal el comando run	El sistema informa de que no reconoce ese comando

Caso de Uso 7: Establecer el idioma	
Prueba realizada	**Resultado obtenido**
Usar el comando set language idioma, siendo idioma el idioma al que lo queremos cambiar.	El sistema cambia al idioma establecido, si es inglés o español. Si no es ninguno de estos se pondrá en inglés.
Prueba realizada	**Resultado obtenido**
Escribir mal el comando set language	El sistema informa de que no reconoce ese comando

Capítulo 6. Diseño del Sistema

El diseño del sistema se basa en analizar la arquitectura, el diseño de las clases, los diagramas de interacción y estados y el plan de pruebas de nuestra aplicación.

6.1 Arquitectura del Sistema

6.1.1 Diagramas de Paquetes

A continuación, se muestran los diagramas de paquetes:

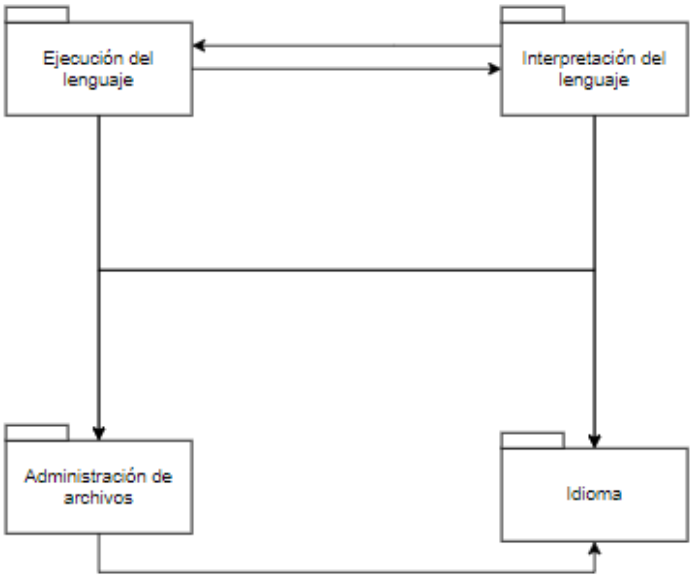


Figura 6. 1. Diagrama de paquetes

6.1.1.1 Ejecución del lenguaje

Este paquete contiene las siguientes clases:

- Main: Esta clase se encarga de ejecutar la aplicación.
- Interpreter: Esta clase se encarga de ejecutar el código escrito por el usuario que ya ha sido comprobado gramaticalmente.

6.1.1.2 Interpretación del lenguaje

Este paquete contiene las siguientes clases:

- DSL: Se encarga de obtener la entrada del usuario e invocar a las diferentes partes que lo comprueban y lo interpretan.
- Lexer: Se encarga de comprobar que la gramática del usuario es correcta.

6.1.1.3 Administración de archivos

En este paquete se incluye solo una clase:

- Json_manager: Se encarga de modificar y leer todos los archivos JSON que se usan en la aplicación.

6.1.1.4 Idioma

En este paquete se incluye solo una clase:

- Lang: Se encarga de almacenar la internacionalización de los mensajes y de detectar el idioma que tiene el sistema operativo.

6.1.2 Diagramas de Despliegue

A continuación, se desarrollan los diagramas de despliegue:

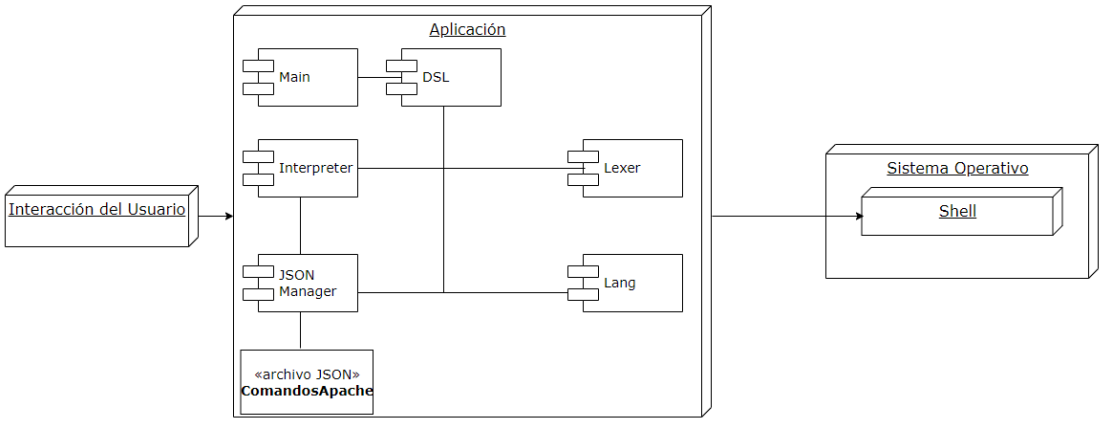


Figura 6. 2. Diagrama de despliegue

El usuario interactúa con la aplicación a través de una interfaz de comandos, después la aplicación interactúa con el sistema operativo a través de la clase *Interpreter*. Dicha clase interactúa con el sistema operativo que esté corriendo la aplicación. Es decir, si estuviera ejecutándose en Windows, usaría el PowerShell del propio sistema operativo para correr la aplicación.

6.1.2.1 Interacción del Usuario

Es la entrada que introduce el usuario al sistema.

6.1.2.2 Aplicación

La aplicación y todas las clases y archivos que la forman. Se comunicará con el sistema operativo, mandándole órdenes a ejecutar.

6.1.2.3 Sistema operativo

El sistema operativo se encargará de recibir órdenes y ejecutarlas.

6.2 Diseño de Clases

Se muestra los diagramas, la información y las relaciones de las clases que componen el sistema.

6.2.1 Diagrama de Clases

El diagrama de clases del sistema es el siguiente:

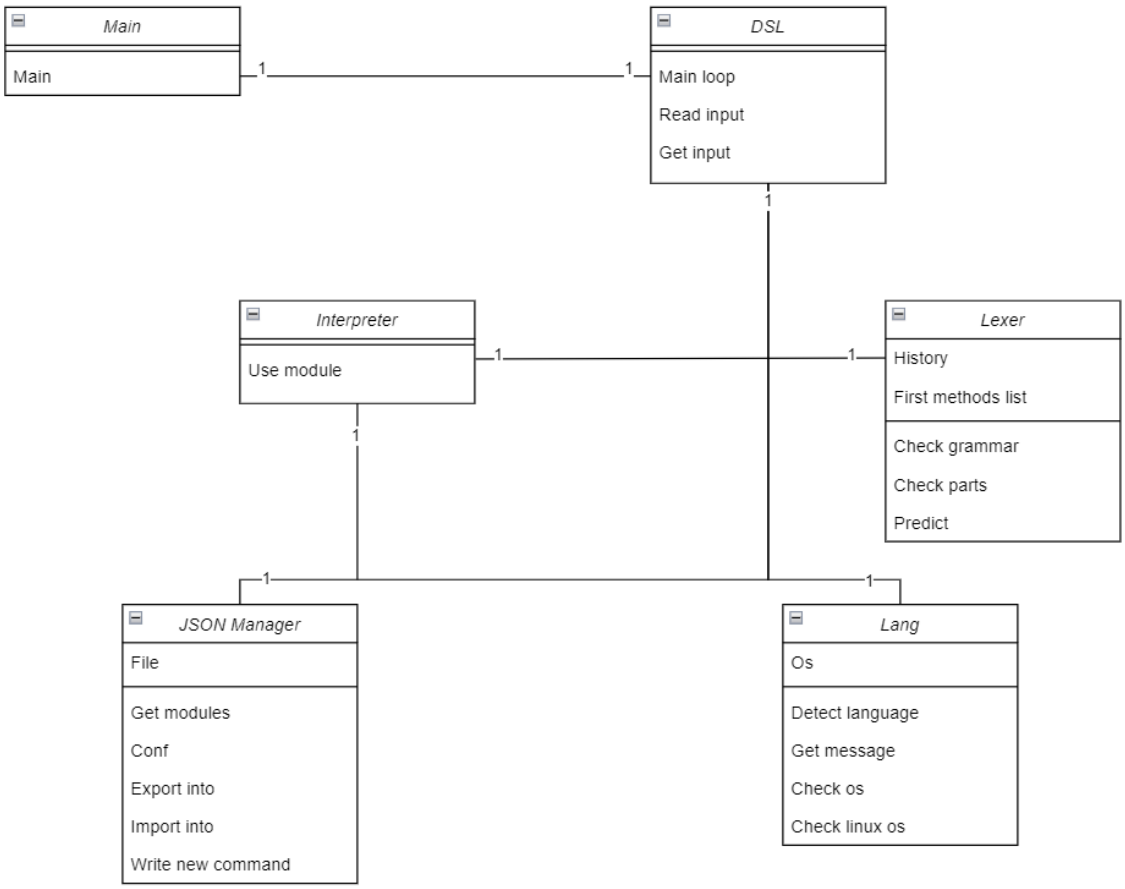


Figura 6. 3. Diagrama de clases

6.3 Diagramas de Interacción

Los diagramas de interacción se usan para evolucionar y detallar los diagramas desarrollados en la sección de análisis.

6.3.1 Ayuda sobre el sistema

Diagrama de interacción de la ayuda del sistema:

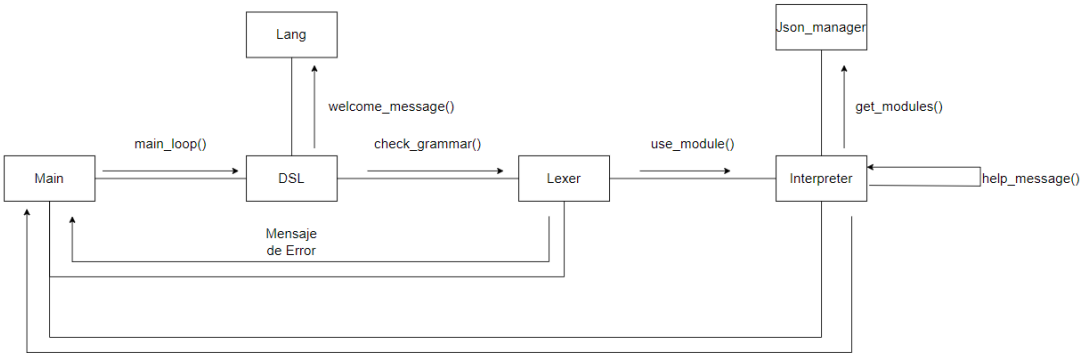


Figura 6. 4. Diagramas de Interacción - Ayuda sobre el sistema

6.3.2 Usar un comando del sistema

Diagrama de interacción del uso de un comando del sistema:

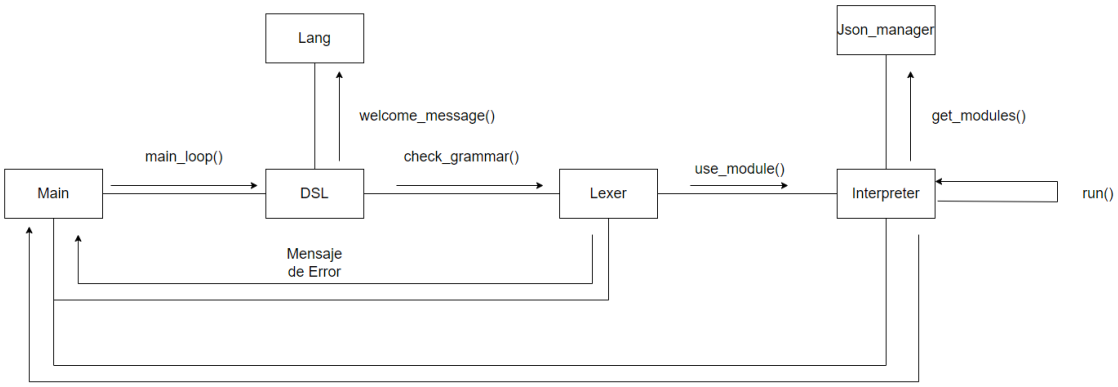


Figura 6. 5. Diagramas de Interacción - Usar un comando del sistema

6.3.3 Crear un nuevo comando para el sistema

Diagrama de interacción de la creación de un nuevo comando para el sistema:

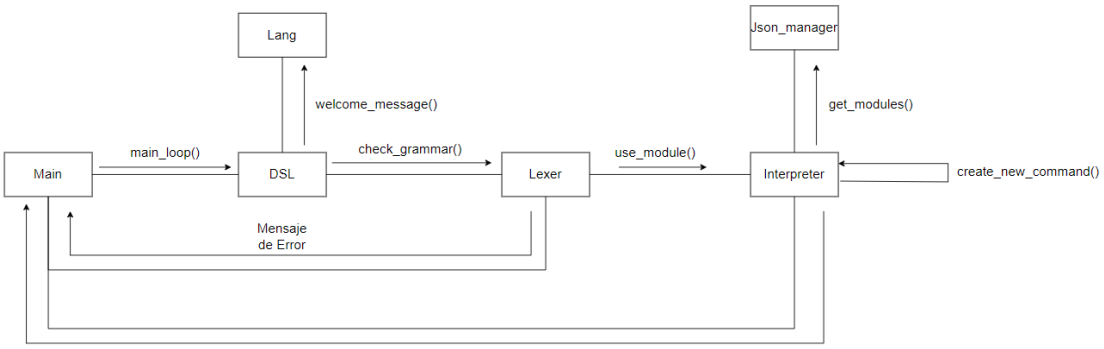


Figura 6. 6. Diagramas de Interacción - Crear un nuevo comando para el sistema

6.3.4 Importar un comando del sistema

Diagrama de interacción de la importación de un comando del sistema:

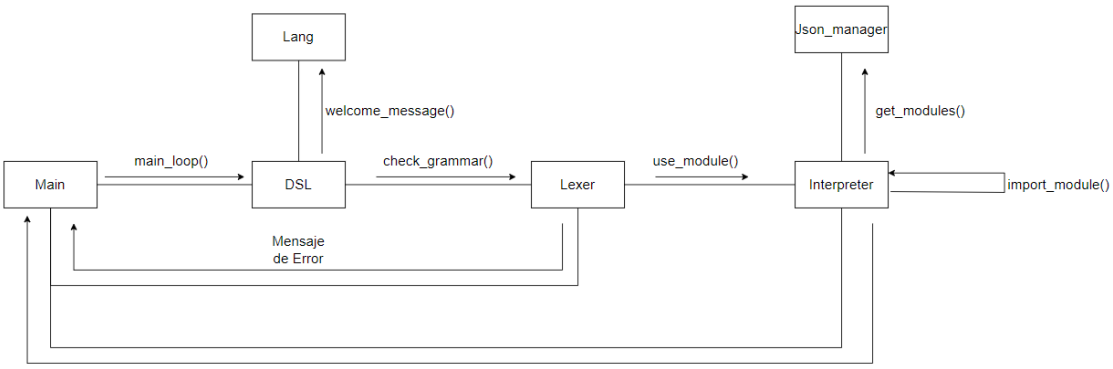


Figura 6. 7. Diagramas de Interacción - Importar un comando del sistema

6.3.5 Exportar un comando del sistema

Diagrama de interacción de la exportación de un comando del sistema:

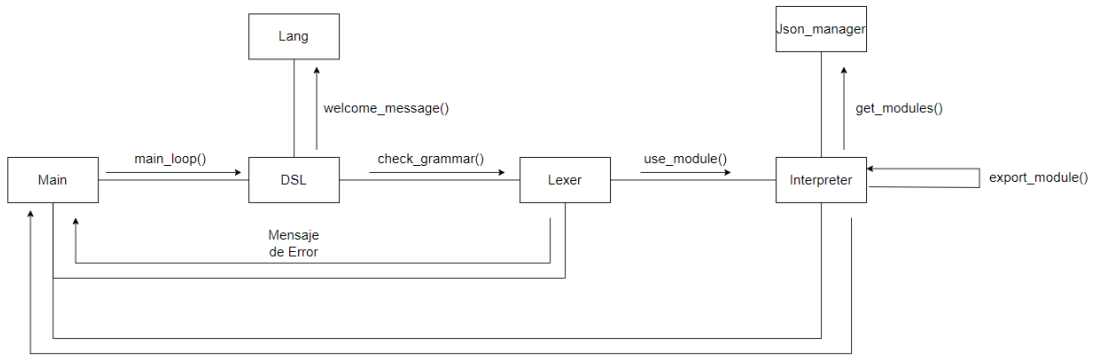


Figura 6. 8. Diagramas de Interacción - Exportar un comando del sistema

6.3.6 Correr varios comandos al mismo tiempo en una configuración básica

Diagrama de interacción sobre ejecutar varios comandos simultáneamente:

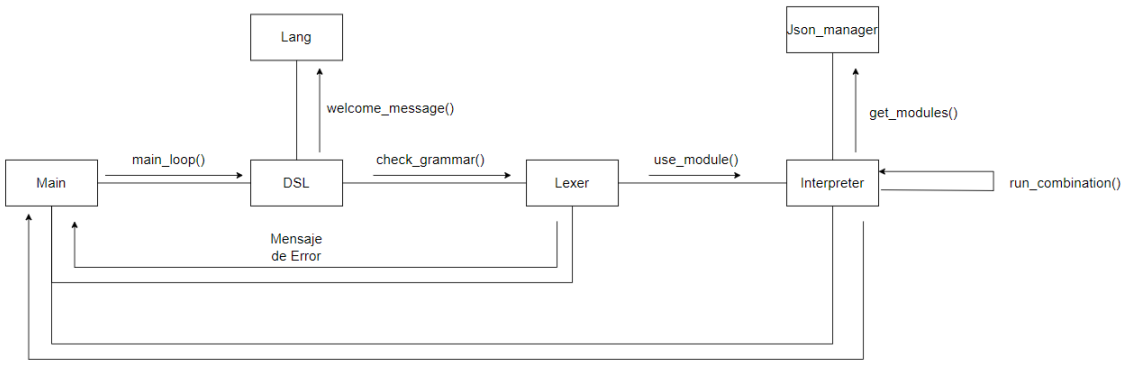


Figura 6. 9. Diagramas de Interacción - Correr varios comandos al mismo tiempo en una configuración básica

6.3.7 Establecer el idioma

Diagrama de interacción sobre cómo se establece el idioma:

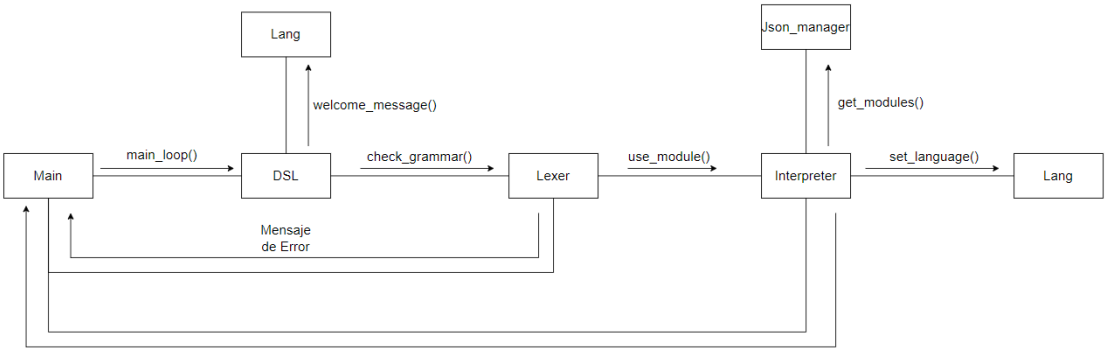


Figura 6. 10. Diagramas de Interacción - Establecer el idioma

6.4 Especificación Técnica del Plan de Pruebas

Se especifica el plan de pruebas que se ejecutan en el sistema; las pruebas unitarias, de integración, del sistema, de usabilidad y de rendimiento

6.4.1 Pruebas Unitarias

Nuestra aplicación cubre este tipo de pruebas con las aserciones. Para comprobar que no se producen errores gramaticales se comprueba cada línea de código que escribe el usuario, en caso de producirse un error se lanza una excepción de tipo *AssertionError*.

Esta excepción después es capturada y ayuda al usuario a comprender cual es el fallo que cometió.

6.4.2 Pruebas de Integración y del Sistema

Las pruebas fueron diseñadas en la sección 5.8. En esta sección pasaremos a comentar las pruebas y la salida exactas que tiene. Todas estas pruebas se hacen en una máquina Windows 11.

Caso de Uso 1: Ayuda sobre el sistema	
Prueba	**Resultado obtenido**
Introducir "help"	El sistema muestra el siguiente mensaje: "En caso de necesitar ayuda sobre un comando cualquiera X, escribir help X En caso de querer definir una actualización sobre un comando Y, escribir create Y En caso de querer usar un comando, escribir primero el nombre del grupo de comando, por ejemplo: AppArmor, después el comando concreto, por ejemplo: Ensure_AppArmor_Is_Enabled En caso de desconocer el sistema, existe una opción de correr un conjunto de comandos con la orden: run level1 o run level2 En caso de querer exportar un comando o conjunto de comandos usar la orden: export nombre_del_comando_1, nombre_del_comando2, ... into nombre_del_archivo En caso de querer importar un comando usar la orden: import from nombre_del_archivo_1 nombre_del_comando_1, nombre_del_comando2, ... Si no se especifica que comandos importar, se importarán todos Para ejecutar todos los comandos, pulsar la tecla ENTER dos veces, con una vez se cambia de línea Si quieres cambiar el idioma escribir set language LANGUAGE. Siendo LANGUAGE el mensaje al que cambiar (Solo están disponibles Español e Inglés)"
Prueba	**Resultado obtenido**
Introducir "help check_server_is_not_multi_use_system"	El sistema muestra el mensaje "Asegurarse de que el servidor no es de multiuso. Devolverá la lista de servicios activos"
Prueba	**Resultado obtenido**
Introducir "hepl"	El sistema muestra el mensaje "Muy pocos términos, solo el mensaje general de ayuda debe estar definido por un único comando help"
Prueba	**Resultado obtenido**
Introducir "help check_servers_is_not_multi_use_system"	El sistema muestra el mensaje "El segundo término no corresponde con ninguno del lenguaje. Quizás quisiste decir: Check_Server_Is_Not_Multi_Use_System"

Caso de Uso 2: Usar un comando del sistema	
Prueba	**Resultado obtenido**
Introducir "installation check_server_is_not_multi_system"	Se muestra: "La orden ejecutada es la siguiente: Get-Service | Where-Object {$_. Status -eq 'Running'}" luego "Devolverá la lista de servicios activos" y se imprime la lista de servicios activos.
Prueba	**Resultado obtenido**
Introducir "error no funciona"	Se muestra: "El primer término no corresponde con ninguno del lenguaje"
Prueba	**Resultado obtenido**
Introducir "installation error"	Se muestra: "El segundo término no corresponde con ninguno del lenguaje. Quizás quisiste decir: Ensure_Apache_Is_Installed_Appropriately"
Prueba	**Resultado obtenido**
Introducir "installation"	Se muestra: "Muy pocos términos, solo el mensaje general de ayuda debe estar definido por un único comando help"

Caso de Uso 3: Crear un nuevo comando del sistema	
Prueba	**Resultado obtenido**
Introducir "create prueba1" Introducir "desc1" Introducir "ls"	Se muestra: ""
Prueba	**Resultado obtenido**
Introducir "create prueba1"	Se muestra: ""
Prueba	**Resultado obtenido**
Introducir "creat"	Se muestra: "Muy pocos términos, solo el mensaje general de ayuda debe estar definido por un único comando help"

Caso de Uso 4: Importar un comando del sistema	
Prueba	**Resultado obtenido**
Introducir "import CVE-2021-41773 from CVE-updateUbuntu-1"	No se muestra nada Se añade al archivo la orden indicada.
Prueba	**Resultado obtenido**
Introducir "import from CVE-updateUbuntu-1"	No se muestra nada se añaden todas las ordenes al archivo
Prueba	**Resultado obtenido**
Introducir "imporrrt CVE-2021-41773 from CVE-updateUbuntu-1"	Se muestra: nada
Prueba	**Resultado obtenido**
Introducir "import CVE-2021-41773 from file10"	Se muestra: "No existe ningún archivo file10.json"

Caso de Uso 5: Exportar un comando del sistema	
Prueba	**Resultado obtenido**
Introducir "export Check_Server_Is_Not_Multi_Use_System into prueba1.json"	No se muestra nada Se añade al archivo la orden indicada.
Prueba	**Resultado obtenido**
Introducir "export into prueba1.json"	No muestra nada
Prueba	**Resultado obtenido**
Introducir "expot into prueba1.json"	Se muestra: "El primer término no corresponde con ninguno del lenguaje"
Prueba	**Resultado obtenido**
Introducir "export Check_Server_Is_Not_Multi_Use_System into prueba111.json"	Se muestra: "No existe ningún archivo: prueba111.json"

Caso de Uso 6: Correr varios comandos al mismo tiempo en una configuración básica	
Prueba	**Resultado obtenido**
Introducir "run level1"	Se muestra cómo se ejecutan uno a uno todos los paquetes pertenecientes al nivel 1.
Prueba	**Resultado obtenido**
Introducir "run level43"	Se muestra: "Las combinaciones de paquete solo pueden ser o level1 o level2"

Caso de Uso 7: Establecer el idioma	
Prueba	**Resultado obtenido**
Introducir "set language English"	Se muestra: "Type letter x to close the app or ENTER to continue:", ya en inglés
Prueba	**Resultado obtenido**
Introducir "set languaje English"	Se muestra: "If you want to change the language write: set language. LANGUAGE being the desired language (Only English and Spanish are available)" Está en inglés porque se corre inmediatamente después de la anterior prueba

6.4.3 Pruebas de Usabilidad

Debido a la complejidad de tener montado un sistema con un servidor Apache corriendo, incluso tener que instalar una máquina virtual para probar la aplicación, se optará por una alternativa. La alternativa se basa en mostrar a los usuarios un vídeo de las funcionalidades de la aplicación y que sean ellos los que opinen de la aplicación.

El proceso será el siguiente, primero se enviará a los usuarios el vídeo de las funcionalidades de la aplicación, acompañado de una explicación verbal de que sucede en el vídeo. Junto al vídeo acompañará un cuestionario, que se describirá en la siguiente sección.

6.4.3.1 Diseño de Cuestionarios

Detallaremos a continuación algunas pautas a seguir en el diseño de los cuestionarios.

6.4.3.1.1 Cuestionario de Evaluación

Distinguiremos dentro del cuestionario 4 secciones:

- **1º**: **Preguntas de carácter general** a través de las cuales se determine el tipo de usuario y su nivel de conocimiento informático.
- **2º**: **Batería de preguntas cortas** con los distintos aspectos de la aplicación que se pretendan evaluar.
- **3º**: **Observaciones**, para que el usuario aporte todo lo que considere oportuno de nuestra aplicación.

6.4.3.2 Actividades de las Pruebas de Usabilidad

6.4.3.2.1 Preguntas de carácter general

¿Usa un ordenador frecuentemente?
1. Todos los días 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de actividades realiza con el ordenador?
1. Es parte de mi trabajo o profesión 2. Lo uso básicamente para ocio 3. Solo empleo aplicaciones estilo Office 4. Únicamente leo el correo y navego ocasionalmente
¿Ha usado alguna vez software como el de esta prueba?
1. Sí, he empleado software similar 2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares 3. No, nunca
¿Alguna vez ha usado Apache?
1. Si 2. No

Figura 6. 11. Preguntas de carácter general

6.4.3.2.2 Preguntas Cortas sobre la Aplicación y Observaciones

Para este cuestionario se usa la escala Likert, que define como respuestas 5 posibilidades: totalmente en desacuerdo, en desacuerdo, neutral, de acuerdo y totalmente de acuerdo. El cuestionario es el siguiente:

Pregunta	Totalmente en desacuerdo	En desacuerdo	Neutral	De acuerdo	Totalmente de acuerdo
Esta herramienta ofrece una asistencia útil para la configuración segura de servidores web Apache					
El uso del DSL disminuye la complejidad de configuración de este tipo de servidores					
El DSL ofrece herramientas suficientes para crear cualquier tipo de configuración para los servidores Apache					
Basado en su experiencia previa, este DSL ofrece una mayor facilidad para la configuración de servidores de este tipo					
La seguridad y configuración de servidores podrían beneficiarse de esta herramienta al contribuir con una gran facilidad para configurarlos					
Este DSL proporciona una forma fácil e intuitiva para configurar servidores					
El usuario comete menos errores mientras trabaja de una manera más rápida y eficaz mediante el uso de este DSL					
Este DSL puede contribuir a facilitar la desaparición de errores de seguridad en los servidores					
Este DSL puede considerarse útil y usable					
Los elementos del lenguaje están bien definidos y son entendibles por el usuario					
Prefieres está metodología a hacerlo todo manualmente					

Figura 6. 12. Preguntas cortas sobre la aplicación

6.4.4 Pruebas de Rendimiento

Para hacer estas pruebas se va a usar programas nativos al sistema operativo Windows, de ellos obtendremos los gráficos para comparar el rendimiento de nuestra aplicación.

Tendremos cuatro gráficos diferentes:

- Gráfico de memoria de la sección memoria en el apartado rendimiento del administrador de tareas
- Gráfico de CPU en la sección de CPU en el apartado rendimiento del administrador de tareas
- Gráfico de memoria en el monitor de recursos en la sección memoria
- Gráfico de CPU en el monitor de recursos en la sección de CPU

Se compararán entonces los gráficos de una ejecución del sistema normal, sin el uso de la aplicación y otra con el uso de la aplicación.

Las mediciones para obtener los gráficos se harán en intervalos de 60 segundos y de gráfico en gráfico, así se obtiene una comparación más precisa.

Capítulo 7. Implementación del Sistema

Esta sección cubre todos los estándares, normas, lenguajes de programación, herramientas y programas usados y todos los problemas encontrados en el desarrollo de esta aplicación.

7.1 Estándares y Normas Seguidos

En cuanto al desarrollo de la aplicación, se usa la guía creada por el CIS, Apache HTTP Server 2.4. Esta guía es usada para desarrollar los archivos JSON que utiliza el sistema para ejecutar los comandos. La aplicación se desarrolla en Python, por lo que también seguiremos los estándares y normas que aplica Python.

También es importante mencionar que esta propia plantilla usa una variación de la métrica v3.

7.2 Lenguajes de Programación

Para el desarrollo del proyecto se usa Python, en su versión 3.7.1. Unido a esto, se usan varios módulos de Python para el desarrollo de la aplicación, que se explicarán a continuación:

- Ctypes: Usamos este módulo para identificar el idioma que usa el sistema y así mostrar nuestra aplicación en el mismo. En esta aplicación solo está internacionalizado en español e inglés, si el sistema se encontrase en otro idioma, se pondrá como base inglés.
- Datetime: Usamos el módulo datetime para determinar cuál es la fecha actual para reflejarlo en el historial de comandos.
- Distro: Usamos el módulo distro para determinar cuál es la distribución de Linux (en caso de que la aplicación corra en un sistema Linux) que se está usando y así determinar que comandos ejecutar.
- Google Translator from Deep Translator: Usamos el módulo Google Translator para traducir al inglés las descripciones de los comandos que se ejecuten o que se pida información sobre ellos, esto siempre y cuando el sistema que corra la aplicación esté en inglés o se haya especificado que el idioma de la aplicación sea inglés.
- Io: Usamos el módulo io para leer y modificar archivos.
- JSON: Usamos el módulo JSON para trabajar con los archivos de este tipo.
- Keyboard: Usamos el módulo keyboard para crear eventos de entrada por teclado como se haría con el método input. No usamos el método input, porque paraliza la ejecución hasta que se pulsa Enter y no valía para nuestra aplicación. Esto se debe a que necesitamos que la máquina no se paralice para implementar un historial y la previsión de texto al pulsar Tab.
- Locale: Usamos el módulo en combinación con Ctypes para el mismo propósito que explicamos en su descripción.

- Os: Usamos el módulo *os* para interactuar con el sistema operativo que corre nuestra aplicación, más concretamente para correr los comandos en el sistema operativo Linux.
- Sleep: Usamos el módulo time y de él sleep para hacer que la máquina espere unos milisegundos y de esta forma se actualice el buffer de la consola.
- Subprocess: Usamos el módulo subprocess para interactuar con el sistema Windows y usar Powershell para ejecutar los comandos que se usen.
- Sys: Usamos este módulo para escribir en consola, ya que no usamos el comando input () como ya explicamos en la aclaración del módulo keyboard.

7.3 Herramientas y Programas Usados para el Desarrollo

Se explican a continuación las diferentes herramientas y programas usados para el desarrollo de la aplicación.

7.3.1 PyCharm

PyCharm es un entorno de desarrollo integrado para programación en Python creado por la compañía checa JetBrains. Dentro de sus funcionalidades se puede destacar análisis de código, un debugger gráfico, unit testing, integración con sistemas de control de versiones y soporta desarrollo web con Django y ciencia de datos con Anaconda. Usaremos su versión Community Edition 2021.1.1.

Lo usaremos para el desarrollo de la aplicación en Python. Se escoge este IDE entre otros tras un pequeño estudio de las diferentes opciones que hay ya que es uno de los entornos más completos.

7.3.2 VirtualBox

VirtualBox es un software de virtualización actualmente desarrollado por Oracle Corporation. Es usado para instalar sistemas operativos adicionales, a través de máquinas virtuales. Dentro de nuestro proyecto se usará para instalar máquinas en las que probar la aplicación.

La creación de máquinas virtuales y pruebas en ellas permite al desarrollador probar un sistema en pruebas en un entorno seguro, de esta forma no se corre el riesgo de estropear el sistema operativo original. Además, permite desde una sola máquina probar varios sistemas operativos al mismo tiempo, lo cual es totalmente necesario para desarrollar una aplicación multiplataforma como la nuestra.

7.3.3 Auto-py-to-exe

Auto-py-to-exe[9] es una aplicación creada por Brent Vollebregt. Convierte aplicaciones Python en ejecutables usando una interfaz gráfica simple.

Usaremos auto-py-to-exe para crear el ejecutable de Windows.

7.4 Creación del Sistema

Se explican a continuación los problemas que surgieron durante la creación de la aplicación y la descripción detallada de las clases del sistema.

7.4.1 Problemas Encontrados

Durante el desarrollo de la aplicación se encontraron algunos problemas que merece la pena comentar puesto que aumentaron la dificultad del desarrollo o hicieron el desarrollo más tedioso.

7.4.1.1 Problema 1: Inexistencia de la guía HTTP Apache 2.4 para Windows

No existe una guía CIS HTTP Apache 2.4 para Windows. La guía creada por el *Center for Internet Security, está* creada para sistemas Linux, más concretamente, para las distribuciones de RedHat. Por consecuencia de esto, habrá que buscar equivalencias en comandos y probarlos uno a uno.

Además, no todos los comandos podrán ser traducidos a PowerShell, ya que algunos de los comandos que describe la guía son para módulos específicos de Linux como SELinux o AppArmor.

La solución a esto no es otra que echarle mucho tiempo e investigar los suficiente.

7.4.1.2 Problema 2: Inexistencia de la guía HTTP Apache 2.4 para Ubuntu

No existe una guía CIS HTTP Apache 2.4 para Ubuntu. Como explicamos en el comando anterior, la guía es creada para sistemas de la distribución RedHat. Por consecuencia de esto, habrá que buscar equivalencias en comandos y probarlos uno a uno.

La traducción de estos comandos a Ubuntu es mucho más sencilla y rápida, debido a que ambos son sistemas Linux, pero consume mucho tiempo.

La solución es simple, buscar las equivalencias y crear el documento.

7.4.1.3 Problema 3: Dificultad en probar la aplicación debido a ser multiplataforma

Probar la aplicación consume mucho tiempo. A la hora de realizar pruebas al insertar una nueva funcionalidad se consume mucho tiempo en probarlo correctamente.

Primero hay que crear y configurar una máquina virtual que tenga Apache corriendo sin problemas. Después, hay que probar en cada plataforma si funciona esa nueva funcionalidad en ese sistema. En caso de encontrar algún fallo y tener que realizar correcciones, se tendría que empezar de nuevo el proceso de pruebas.

7.4.1.4 Problema 4: No se puede usar el método input nativo de Python

No se puede hacer uso del método de Python llamado input. Esto es un problema bastante grande ya que es una aplicación cuyo mayor objetivo es interactuar con la entrada del usuario.

No se puede hacer uso del método ya que, debido a la implementación del método nativo, paraliza el flujo de ejecución de la máquina hasta que recibe una entrada y se pulsa *Enter*. Esto se convierte en un problema para nosotros ya que necesitamos acceder al flujo de la máquina en caso de querer hacer uso del historial o de la predicción de texto.

Debido a esto, se tiene que crear un nuevo método que cubra esta necesidad.

7.4.2 Descripción Detallada de las Clases

La documentación se puede encontrar dentro de la carpeta entregada en la sección doc.

Capítulo 8. Desarrollo de las Pruebas

A continuación, se desarrolla las diferentes pruebas que se hacen al sistema, pruebas unitarias, de integración y del sistema, de usabilidad y de rendimiento.

8.1 Pruebas Unitarias

Las pruebas unitarias del sistema, como se ha mencionado anteriormente, son llevadas a cabo por las aserciones en el código. De esta forma se comprueba que no se ejecute nada de forma errónea y se avise al usuario.

Las pruebas unitarias se usan en la clase *Lexer* para comprobar la gramática del usuario y ayudarle a corregir sus errores, en caso de haberlos, se enseñará como en un ejemplo.

Suponemos que el usuario quiere información sobre el comando "Ensure_Log_Config_Module_Is_Enabled", pero al escribirlo comete un error y escribe "Ensure_Log_Confij_Module_Is_Enabled".

Al pasar esto, la clase *Lexer* lanzaría una excepción, puesto que no pasaría la prueba unitaria del método check_help_terms(), con el mensaje indicando cual sería el método más parecido al escrito, de la siguiente manera:

```
    Bienvenido al DSL version Windows
    Escriba a continuación las órdenes del lenguaje (help para ayuda)
help Ensure_Log_Confij_Module_Is_Enabled

El segundo término no corresponde con ninguno del lenguaje. Quizás quisiste decir: Ensure_Log_Config_Module_Is_Enabled
Pulsa la letra x para cerrar la aplicación o ENTER para continuar escribiendo:
```

Figura 8. 1. Ejemplo de pruebas unitarias

Esto es solo un ejemplo, puesto que toda la clase *Lexer* se basa en pruebas unitarias para comprobar los errores gramaticales que puede producir el usuario y para indicarle cual sería la correcta gramática.

8.2 Pruebas de Integración y del Sistema

Se ejecutan a continuación las pruebas funcionales ya diseñadas anteriormente y anotamos el resultado obtenido.

Caso de Uso 1: Ayuda sobre el sistema	
Prueba	**Resultado obtenido**
Introducir “help”	El sistema muestra el siguiente mensaje: “En caso de necesitar ayuda sobre un comando cualquiera X, escribir help X En caso de querer definir una actualización sobre un comando Y, escribir create Y En caso de querer usar un comando, escribir primero el nombre del grupo de comando, por ejemplo: AppArmor, después el comando concreto, por ejemplo: Ensure_AppArmor_Is_Enabled En caso de desconocer el sistema, existe una opción de correr un conjunto de comandos con la orden: run level1 o run level2 En caso de querer exportar un comando o conjunto de comandos usar la orden: export nombre_del_comando_1, nombre_del_comando2, ... into nombre_del_archivo En caso de querer importar un comando usar la orden: import from nombre_del_archivo_1 nombre_del_comando_1, nombre_del_comando2, ... Si no se especifica que comandos importar, se importarán todos Para ejecutar todos los comandos, pulsar la tecla ENTER dos veces, con una vez se cambia de línea Si quieres cambiar el idioma escribir set language LANGUAGE. Siendo LANGUAGE el mensaje al que cambiar (Solo están disponibles Español e Inglés)”
Prueba	**Resultado obtenido**
Introducir “help check_server_is_not_multi_use_system”	El sistema muestra el mensaje “Asegurarse de que el servidor no es de multiuso. Devolverá la lista de servicios activos”
Prueba	**Resultado obtenido**

Introducir “hepl”	El sistema muestra el mensaje “Muy pocos términos, solo el mensaje general de ayuda debe estar definido por un único comando help”
Prueba	**Resultado obtenido**
Introducir “help check_servers_is_not_multi_use_system”	El sistema muestra el mensaje “El segundo término no corresponde con ninguno del lenguaje. Quizás quisiste decir: Check_Server_Is_Not_Multi_Use_System”

Caso de Uso 2: Usar un comando del sistema	
Prueba	**Resultado obtenido**
Introducir “installation check_server_is_not_multi_system”	Se muestra: “La orden ejecutada es la siguiente: Get-Service | Where-Object {$_. Status -eq 'Running'}” luego “Devolverá la lista de servicios activos” y se imprime la lista de servicios activos.
Prueba	**Resultado obtenido**
Introducir “error no_funciona”	Se muestra: “El primer término no corresponde con ninguno del lenguaje”
Prueba	**Resultado obtenido**
Introducir “installation error”	Se muestra: “El segundo término no corresponde con ninguno del lenguaje. Quizás quisiste decir: Ensure_Apache_Is_Installed_Appropriately”
Prueba	**Resultado obtenido**
Introducir “installation”	Se muestra: “Muy pocos términos, solo el mensaje general de ayuda debe estar definido por un único comando help”

Caso de Uso 3: Crear un nuevo comando del sistema	
Prueba	**Resultado obtenido**
Introducir “create prueba1” Introducir “desc1” Introducir “ls”	Se muestra: “”
Prueba	**Resultado obtenido**
Introducir “create prueba1”	Se muestra: “”
Prueba	**Resultado obtenido**
Introducir “creat”	Se muestra: “Muy pocos términos, solo el mensaje general de ayuda debe estar definido por un único comando help”

Caso de Uso 4: Importar un comando del sistema	
Prueba	**Resultado obtenido**
Introducir “import from CVE-updateUbuntu-1 CVE-2021-41773”	No se muestra nada Se añade al archivo la orden indicada.
Prueba	**Resultado obtenido**
Introducir “import from CVE-updateUbuntu-1”	No se muestra nada se añaden todas las ordenes al archivo
Prueba	**Resultado obtenido**
Introducir “imporrrt from CVE-updateUbuntu-1 CVE-2021-41773”	Se muestra: nada
Prueba	**Resultado obtenido**
Introducir “import from file10 CVE-2021-41773”	Se muestra: “No existe ningún archivo file10.json”

Caso de Uso 5: Exportar un comando del sistema	
Prueba	**Resultado obtenido**
Introducir “export Check_Server_Is_Not_Multi_Use_System into prueba1.json”	No se muestra nada Se añade al archivo la orden indicada.
Prueba	**Resultado obtenido**
Introducir “export into prueba1.json”	No muestra nada
Prueba	**Resultado obtenido**
Introducir “expot into prueba1.json”	Se muestra: “El primer término no corresponde con ninguno del lenguaje”
Prueba	**Resultado obtenido**
Introducir “export Check_Server_Is_Not_Multi_Use_System into prueba111.json”	Se muestra: “No existe ningún archivo: prueba111.json”

Caso de Uso 6: Correr varios comandos al mismo tiempo en una configuración básica	
Prueba	**Resultado obtenido**
Introducir “run level1”	Se muestra cómo se ejecutan uno a uno todos los paquetes pertenecientes al nivel 1.
Prueba	**Resultado obtenido**
Introducir “run level43”	Se muestra: “Las combinaciones de paquete solo pueden ser o level1 o level2”

Caso de Uso 7: Establecer el idioma	
Prueba	**Resultado obtenido**
Introducir “set language english”	Se muestra: “Type letter x to close the app or ENTER to continue:”, ya en inglés
Prueba	**Resultado obtenido**
Introducir “set languaje English”	Se muestra: “If you want to change the language write: set language. LANGUAGE being the desired language (Only English and Spanish are available)” Está en inglés porque se corre inmediatamente después de la anterior prueba

8.3 Pruebas de Usabilidad

Aquí se muestra el resultado de los cuestionarios desarrollados anteriormente en el 6.4.3.

Los usuarios que han respondido al cuestionario son perfiles de informática, en su mayoría ingenieros informáticos o estudiantes de la carrera. En total se han obtenido X respuestas.

A continuación, se muestran las tablas con las respuestas, primero las de control del usuario:

Pregunta	Todos los días	Varias veces a la semana	Ocasionalmente	Nunca o casi nunca
¿Usa un ordenador frecuentemente?	22	3	0	0

Pregunta	Es parte de mi trabajo o profesión	Lo uso básicamente para ocio	Solo empleo aplicaciones estilo Office	Únicamente leo el correo y navego ocasionalmente
¿Qué tipo de actividades realiza con el ordenador?	25	0	0	0

Pregunta	Si, he empleado software similar	No, aunque si empleo otros programas que me ayudan a realizar tareas similares	No, nunca
¿Ha usado alguna vez software como el de esta prueba?	18	3	4

Pregunta	Si	No
¿Alguna vez ha usado Apache?	22	3

Y a continuación veremos las preguntas y respuestas sobre la aplicación. Las respuestas usan la escala Likert [13] que tiene las opciones: 1 totalmente en desacuerdo, 2 en desacuerdo, 3 neutral, 4 de acuerdo y 5 totalmente de acuerdo.

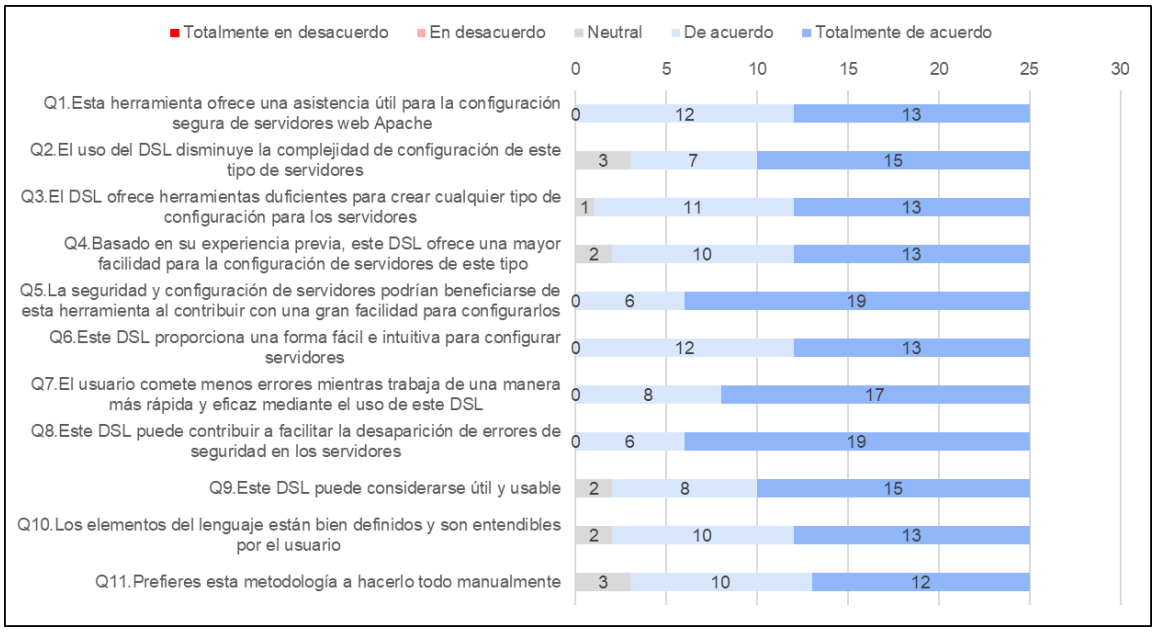


Figura 8. 2 Resultados del cuestionario usando la escala Likert

Como vemos en la Figura 8. 2, los resultados del cuestionario han sido bastante positivos. En la siguiente figura veríamos las respuestas de cada participante a cada pregunta, los valores vuelven a corresponder con la escala Likert. Las columnas representan a cada persona que ha respondido al cuestionario y las columnas representan cada pregunta.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11
P1	5	5	5	4	5	4	5	5	5	5	4
P2	5	3	4	5	5	4	4	5	5	4	4
P3	5	5	5	5	5	5	5	5	5	5	5
P4	4	5	4	5	4	5	5	4	5	5	5
P5	4	5	4	4	5	4	5	5	4	4	5
P6	5	4	4	4	5	5	5	4	4	5	4
P7	4	4	4	3	5	5	5	5	4	3	3
P8	4	5	4	5	4	4	4	5	5	4	4
P9	5	4	3	3	4	4	4	4	3	4	3
P10	5	4	4	4	4	4	4	4	4	4	4
P11	5	4	4	4	5	4	4	4	4	4	4
P12	5	3	5	4	5	4	5	5	4	4	5
P13	5	5	5	4	5	4	5	5	5	4	5
P14	4	5	5	5	5	5	5	5	5	5	4
P15	4	5	5	5	5	5	5	5	5	5	4
P16	4	5	5	5	5	5	5	5	5	5	5
P17	4	5	5	5	5	5	5	5	5	5	5
P18	4	5	5	5	5	5	5	5	5	5	5
P19	5	5	5	5	5	4	5	5	5	4	4
P20	5	4	5	5	5	4	5	4	5	5	4
P21	5	5	5	5	5	4	4	5	5	5	5
P22	5	5	5	5	4	5	5	5	4	5	5
P23	4	4	4	4	5	5	4	5	5	5	5
P24	4	3	4	4	4	5	5	5	4	4	5
P25	4	5	4	4	5	5	4	5	3	3	3

Figura 8. 3 Respuestas de los participantes a cada pregunta

Si analizamos los resultados de las figuras Figura 8. 2 y Figura 8. 3 vemos lo siguiente:

- La opción más escogida en todas las respuestas es “Totalmente de acuerdo”
- La segunda opción más escogida es “De acuerdo”
- Solo en las preguntas Q2, Q3, Q4, Q9, Q10, Q11 hay respuestas en la opción neutral.
- En ninguna de las preguntas hay respuestas negativas.

8.4 Pruebas de Rendimiento

Como definimos en la sección 6.7.4, pasaremos a comparar los gráficos mencionados de dos en dos. Siendo estos, uno de la ejecución estándar y otro de la ejecución con la aplicación.

Las pruebas de rendimiento se realizan en un ordenador Lenovo Yoga 720, con un procesador Intel Core i5-8250U de 1.6GHz de 4 núcleos, además, tiene 8GB de memoria RAM DDR4 y un disco duro SSD de 256 GB. Para las pruebas se usa el Sistema operativo Microsoft Windows 11 Home.

Compararemos primero los gráficos de memoria y luego los de CPU.

- Gráfico de memoria de la sección memoria en el apartado rendimiento del administrador de tareas:

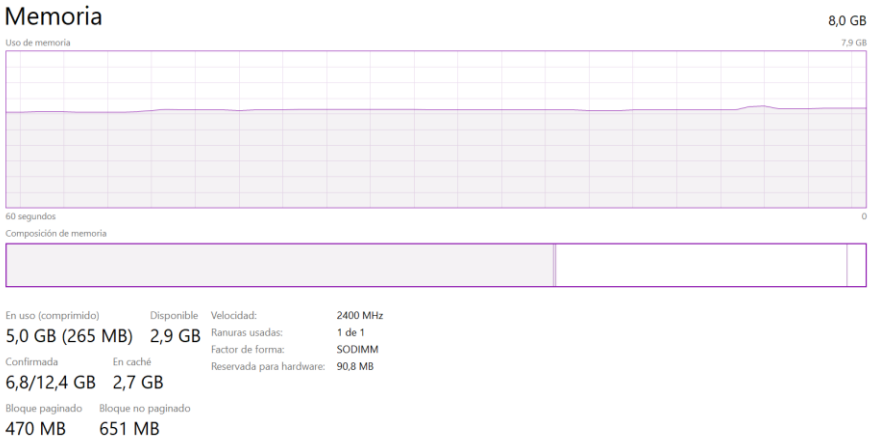


Figura 8. 4. Memoria sin la aplicación en el administrador de tareas

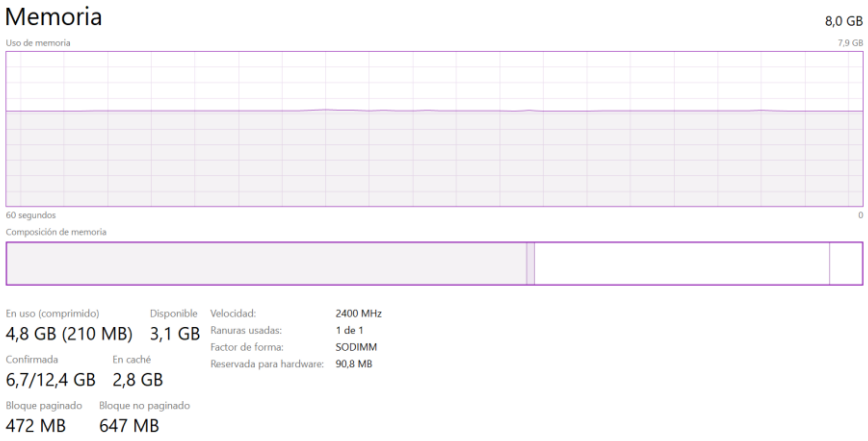


Figura 8. 5. Memoria con la aplicación en el administrador de tareas

- Gráfico de memoria en el monitor de recursos en la sección memoria

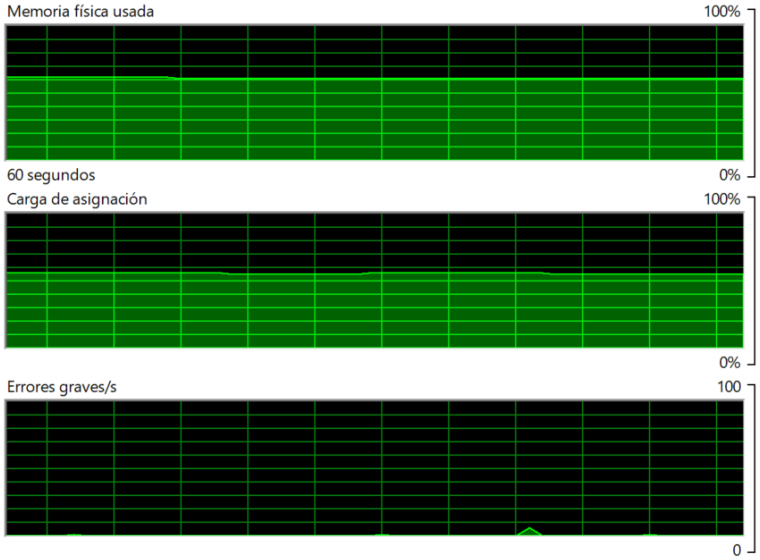


Figura 8. 6. Memoria sin la aplicación en el monitor de recursos

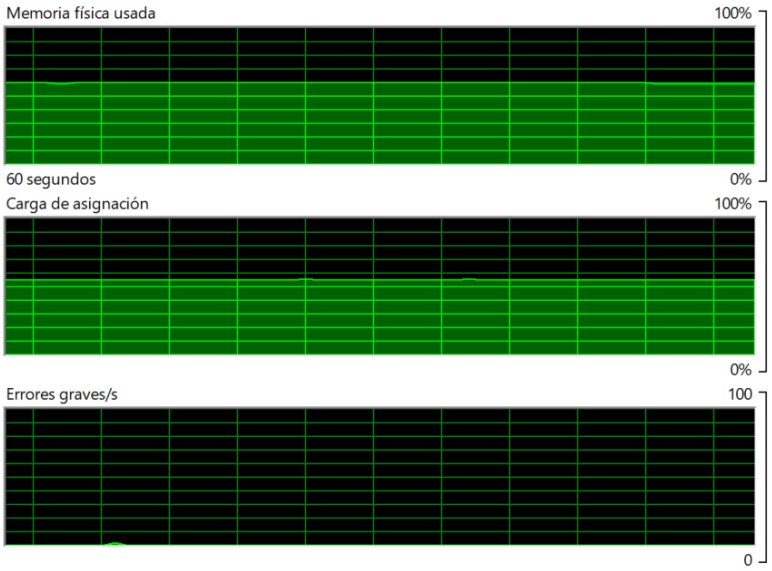


Figura 8. 7. Memoria con la aplicación en el monitor de recursos

Como podemos observar el uso de la aplicación no hace apenas cambio en la memoria del sistema. Pasaremos a ver ahora en la CPU.

- Gráfico de CPU en la sección de CPU en el apartado rendimiento del administrador de tareas

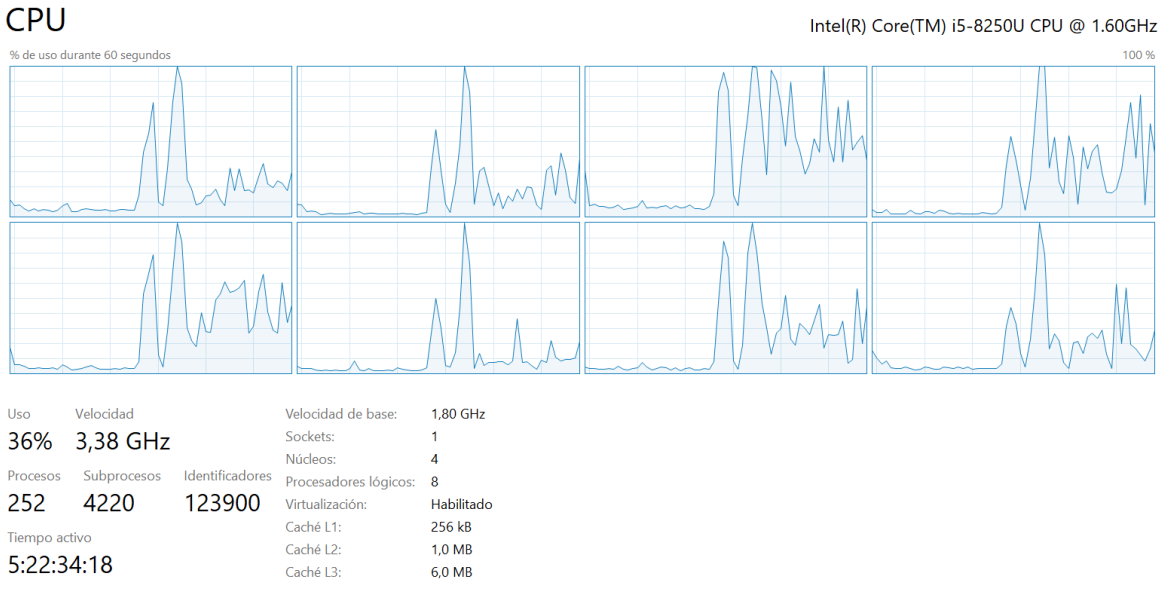


Figura 8. 8. CPU sin la aplicación en el administrador de tareas

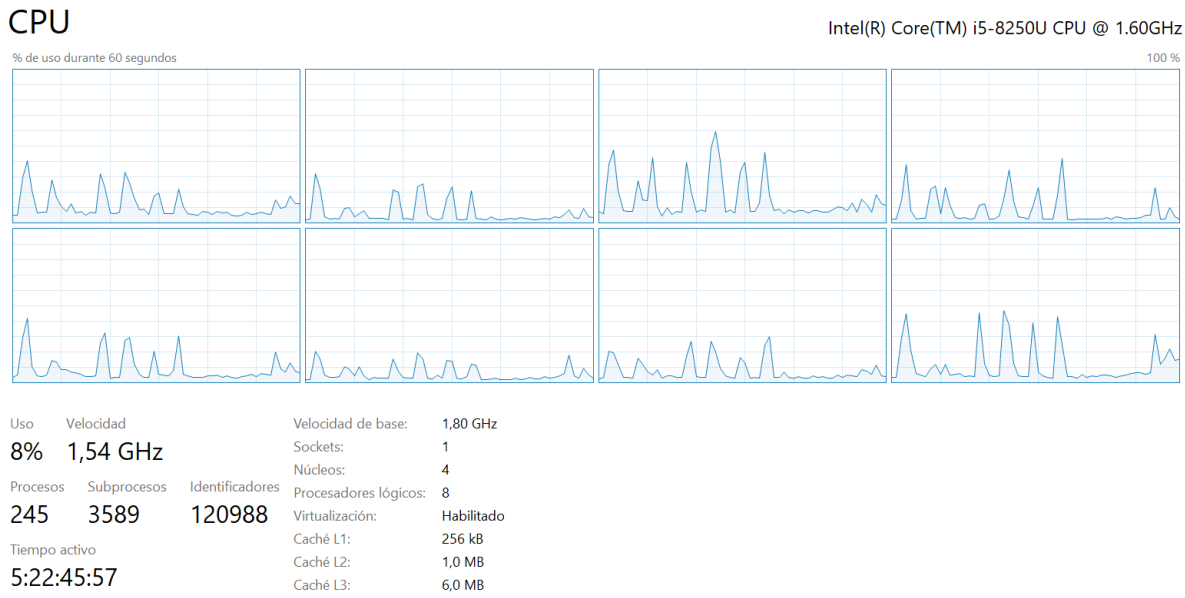


Figura 8. 9. CPU con la aplicación en el administrador de tareas

- Gráfico de CPU en el monitor de recursos en la sección de CPU

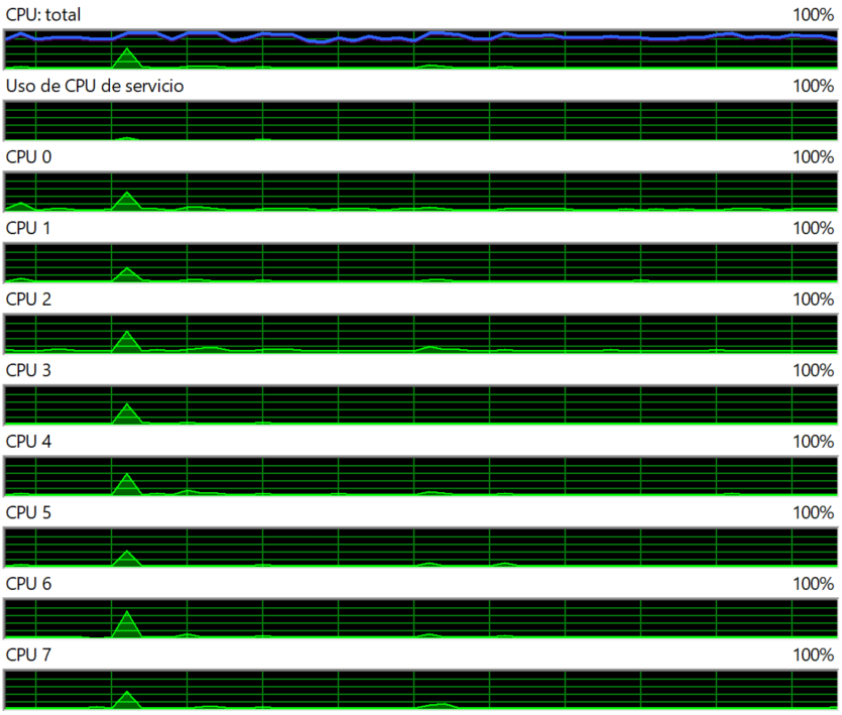


Figura 8. 10. CPU sin la aplicación en el monitor de recursos

Figura 8. 11. CPU con la aplicación en el monitor de recursos

Como vemos en la CPU si se producen cambios cuando está en ejecución la aplicación. Los picos que se ven en los gráficos cuando se usa la aplicación vienen de la ejecución de comandos que interactúan con el sistema operativo.

Aunque use recursos del ordenador, son muy pocos, debido en gran parte a ser una aplicación de consola.

Capítulo 9. Manuales del Sistema

Se desarrollan los diferentes manuales que explican al usuario como instalar, ejecutar y usar el programa. Además, existe un cuarto manual para el programador.

9.1 Manual de Instalación

Se tienen que distinguir dos versiones de este manual de instalación, la primera será para sistemas Windows y la segunda para Linux.

Para Windows, la instalación será así:

Dentro de la carpeta DSL se encuentra la carpeta dist, al abrirla se observan muchos archivos, pero el único que interesa para ejecutar la aplicación es ejecutable con el nombre main.exe.

Figura 9. 1. Archivo main.exe

No es necesario más instalación que ejecutar el archivo mencionado. Si se puede comentar que, lógicamente, el usuario deberá tener un servidor Apache para que el uso de la aplicación tenga sentido.

Para Linux, la instalación será de la siguiente manera:

Primero de todo se necesita tener Python en su versión 3.8, que se puede encontrar aquí: https://www.python.org/downloads/release/python-380/.

Se descarga la carpeta, se abre la carpeta dist y se abre el terminal. Se deberían instalar los siguientes módulos de Python, con la orden sudo pip install *module*, siendo *module* los siguientes módulos:

- deep-translator
- keyboard
- distro
- datetime

Después se ejecutaría con la orden sudo main.py

Esta sección del manual de instalación será común a los dos sistemas operativos:

En cuanto a la configuración previa que se debe hacer, existe tres archivos que se deben mencionar:

- configurationCentos
- configurationUbuntu
- configurationWindows

Estos tres archivos se pueden encontrar en la carpeta entregada al usuario.

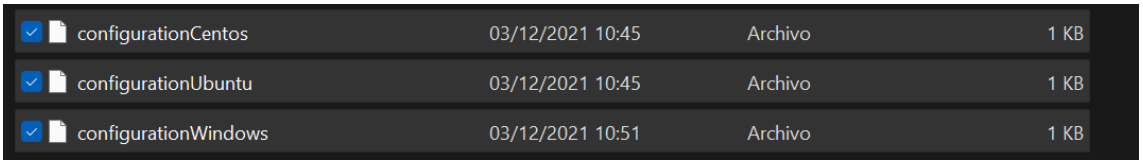


Figura 9. 2. Archivos de configuración

Dependiendo del sistema operativo en el que el usuario esté ejecutando esta aplicación, el usuario tendrá que editar el archivo y cambiar las dos variables que puede contener:

- variable APACHE: Indica la ruta completa del sistema a la carpeta de Apache. Esta variable se encuentra en los 3 sistemas operativos. Para facilitar al usuario la configuración, se incluye dentro del archivo una ruta de ejemplo.
- Variable PASSWORD: Indica la ruta completa al archivo de configuración de contraseñas. Esta variable es solo para las distribuciones de Linux. Para facilitar al usuario la configuración, se incluye dentro del archivo una ruta de ejemplo.

Así es como se le entregan al usuario los tres archivos de los tres sistemas operativos:

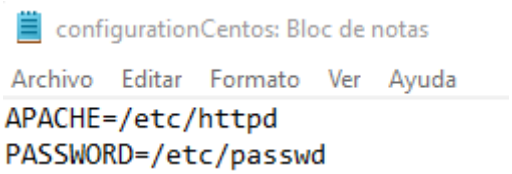


Figura 9. 3. Archivo configurationCentos.txt

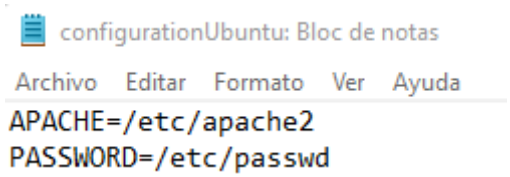


Figura 9. 4. Archivo configurationUbuntu.txt

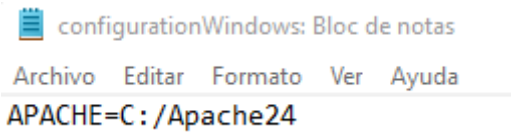


Figura 9. 5. Archivo configurationWindows.txt

9.2 Manual de Ejecución

En el caso de Windows, la ejecución es de lo más simple, doble clic en el archivo main.exe y ya se inicia la aplicación.

En el caso de las dos distribuciones Linux, no pueden ejecutar archivos .exe, por lo que tendremos que contemplar otras posibilidades. Se puede ejecutar la aplicación con sudo python main.py o sudo python3 main.py

9.3 Manual de Usuario

Se debe aclarar que las capturas y ejemplos que se dan en este manual son para la versión de la aplicación en Windows, no obstante, la ejecución es la misma para los tres sistemas operativos. El usuario solo notará de cambio el mensaje de bienvenida y en los comandos.

Al iniciar la aplicación se muestra el mensaje:

```
Bienvenido al DSL version Windows
Escriba a continuación las órdenes del lenguaje (help para ayuda)
```

Figura 9. 6. Mensaje de bienvenida

En este momento el usuario tiene varias posibilidades que se explorarán una a una:

- Help: este comando se usa para saber sobre todas las posibilidades que tiene el usuario dentro de la aplicación, muestra por consola un resumen de todas las funcionalidades disponibles dentro de la aplicación, como se ve a continuación

```
help

En caso de necesitar ayuda sobre un comando cualquiera X, escribir help X
En caso de querer definir una actualización sobre un comando Y, escribir create Y
En caso de querer usar un comando, escribir primero el nombre del grupo de comando por ejemplo: AppArmor, después el comando concreto por ejemplo
En caso de desconocer el sistema, existe una opción de correr un conjunto de comandos con la orden: run level1 o run level2
En caso de querer exportar un comando o conjunto de comandos usar la orden: export nombre_del_comando_1, nombre_del_comando2, ... into nombre_del
En caso de querer importar un comando usar la orden: import from nombre_del_archivo_1 nombre_del_comando_1, nombre_del_comando2, ... . Si no se e
Para ejecutar todos los comandos, pulsar la tecla ENTER dos veces, con una vez se cambia de línea
Si quieres cambiar el idioma escribir set language LANGUAGE. Siendo LANGUAGE el mensaje al que cambiar (Solo están disponibles Español e Inglés)
```

Figura 9. 7. Mensaje de ayuda, comando help

- Help X: este comando se usa para obtener información sobre un comando X. Para el ejemplo, suponemos que el comando del que queremos obtener la ayuda es “Ensure_the_Error_Log_Filename_and_Severity_Level_are_Configured_Correct”. La salida obtenida será la descripción del propio comando:

```
help Ensure_the_Error_Log_Filename_and_Severity_Level_are_Configured_Correctly

Asegurarse que el log de errores y el nivel de severidad están configurados correctamente. Comprobar la linea LogLevel y ErrorLog
```

Figura 9. 8. Descripción del comando "Ensure the error log filename and severity level are configured correct"

- Create Y: este comando es usado para crear comandos nuevos, estos comandos se añadirán al archivo JSON que almacena el resto de los comandos. Para ejecutarlo se le debe dar un nombre Y, este nombre no puede coincidir con ningún otro nombre de los registrados en el archivo o avisará con el error. Para poner el ejemplo ejecutaremos create ejemplo:

```
create ejemplo

Introduce una descripción del nuevo comando:
Esta es una descripcion de ejemplo

Introduce las ordenes exactas que debe ejecutar el sistema Windows(si tiene algún parámetro variable el comando, recuerda escribir PARAM en el sitio
ls
```

Figura 9. 9. Comando create

Esto genera en el documento JSON, sección *Others* un nuevo comando.

- Correr un comando: este es el centro de la aplicación, todos los comandos se almacenan en un archivo JSON llamado comandosApache.json, dependiendo del OS, el archivo tendrá como nombre comandosApacheWindows.json, comandosApacheUbuntu.json o comandosApacheCentOS.json. Aquí se localizan los paquetes de comandos y los comandos en sí. Para ejecutar un comando hay que escribir su paquete de comandos y su nombre. Por ejemplo, en la siguiente captura vemos el paquete de comandos *Installation* y el comando *Check_Server_Is_Not_Multi_Use_System:*

```
"Installation": [
  {
    "nombre": "Check_Server_Is_Not_Multi_Use_System",
    "desc": "Asegurarse de que el servidor no es de multiuso. Devolver\u00e1 la lista de servicios activos",
    "comando": "Get-Service | Where-Object {$_.Status -eq 'Running'}"
  },
```

Figura 9. 10. Ejemplo del archivo JSON

Entonces, para ejecutar este comando haríamos la orden *installation check_server_is_not_multi_use_system* que resultaría en la siguiente salida:

```
Installation Check_Server_Is_Not_Multi_Use_System

La orden ejecutada es la siguiente: Get-Service | Where-Object {$_.Status -eq 'Running'}
Devolverá la lista de servicios activos

Status   Name               DisplayName
------   ----               -----------
Running  AarSvc_32c2f36d    Agent Activation Runtime_32c2f36d
Running  ApacheHTTPServer   Apache HTTP Server
Running  Appinfo            Información de la aplicación
Running  AppXSvc            Servicio de implementación de AppX ...
Running  AtherosSvc         AtherosSvc
Running  AudioEndpointBu... Compilador de extremo de audio de W...
Running  Audiosrv           Audio de Windows
```

Figura 9. 11. Ejecución del comando installation check_server_is_not_multi_use_system

- Run level1 o level2: se usa este comando para correr un conjunto de comandos al mismo tiempo. El comando puede ser run level1 o run level2, es entonces cuando corre todos los comandos del nivel 1 o 2, respectivamente. Estos niveles surgen de los definidos en la guía CIS Benchmark Apache Http 2.4. Lo que hace el comando es correr uno a uno los comandos que estén en el nivel mencionado.
- Export nombre_del_comando_1, nombre_del_comando2, ... into nombre_del_archivo: este comando es usado para exportar uno o varios comandos a un archivo. Los comandos que se exportan son aquellos que estén incluidos en el archivo JSON. Para poner un ejemplo:

```
export Remediate_Server_Is_Not_Multi_Use_System into prueba1
```

Figura 9. 12. Comando export

Esto añadiría dentro del archivo JSON prueba1.json, en la sección *Imported,* el comando *remediate_server_is_not_multi_use_system*. Si se quisiese añadir varios comandos al mismo tiempo simplemente habría que escribirlos separados por comas.

- Import from nombre_del_archivo_1 nombre_del_comando_1, nombre_del_comando2, ...: Este comando es parecido al anterior, pero al revés, sirve para importar comandos de otros archivos JSON al nuestro. Al correr el comando, se añadiría en la sección *Imported* de nuestro JSON todas las órdenes definidas.
- Set language LANGUAGE: Este comando se usa para cambiar el idioma de la aplicación, como lenguajes solo está disponible el español y el inglés. Aquí se ve un ejemplo cambiando el idioma a inglés:

```
set language english



Type letter x to close the app or ENTER to continue:

    Write here the orders for the language (type help for help)
help Check_Server_Is_Not_Multi_Use_System

Make sure the server is not multipurpose. It will return the list of active services
```

Figura 9. 13. Comando set language

- Historial de comandos: Es importante comentar que existe un historial de comandos, este es accesible de forma física en el archivo history.txt o a través de las flechas de teclado cuando se está usando la aplicación, esto permite al usuario acceder a comandos previos.
- Auto completar: La aplicación permite al usuario autocompletar texto mediante la tecla TAB.

9.4 Manual del Programador

Se verá a continuación dos secciones diferentes. La primera es para realizar cualquier ampliación o modificación de la aplicación y la segunda sobre cómo crear el ejecutable.

9.4.1 Ampliación o modificación de la aplicación

Se pueden comentar en este aspecto tres posibles formas en las que otros programadores podrían ampliar o modificar la aplicación.

1- La primera sería usando las funcionalidades nativas de la aplicación, de esta forma se añade en el archivo JSON del sistema operativo, llamado comandosApache más el nombre del sistema operativo, para Windows sería comandosApacheWindows.json, este archivo se encuentra en el directorio raíz del proyecto.
   a. Comando *create*: Se siguen los puntos detallados en el 9.3 y se generaría en el archivo JSON, en la sección *Others* el nuevo comando.
   b. Comando *import*: Se siguen los puntos detallados en el 9.3 y se generaría en el archivo JSON, en la sección *Imported* el comando importado.

2- La segunda sería editando el propio archivo JSON que contiene los comandos de la aplicación y que se llama comandosApache, más el sistema operativo, depende en dónde estemos ejecutando la aplicación (para Windows sería comandosApacheWindows.json). El archivo comandosApache se encuentra en el directorio raíz del proyecto que contiene el código fuente.

3- La tercera sería modificar la propia funcionalidad de la aplicación, añadiendo más sistemas operativos en los que correr esta aplicación. Para hacer eso se tendría que copiar la estructura del JSON de comandos y después modificar el código, la estructura sería la siguiente:

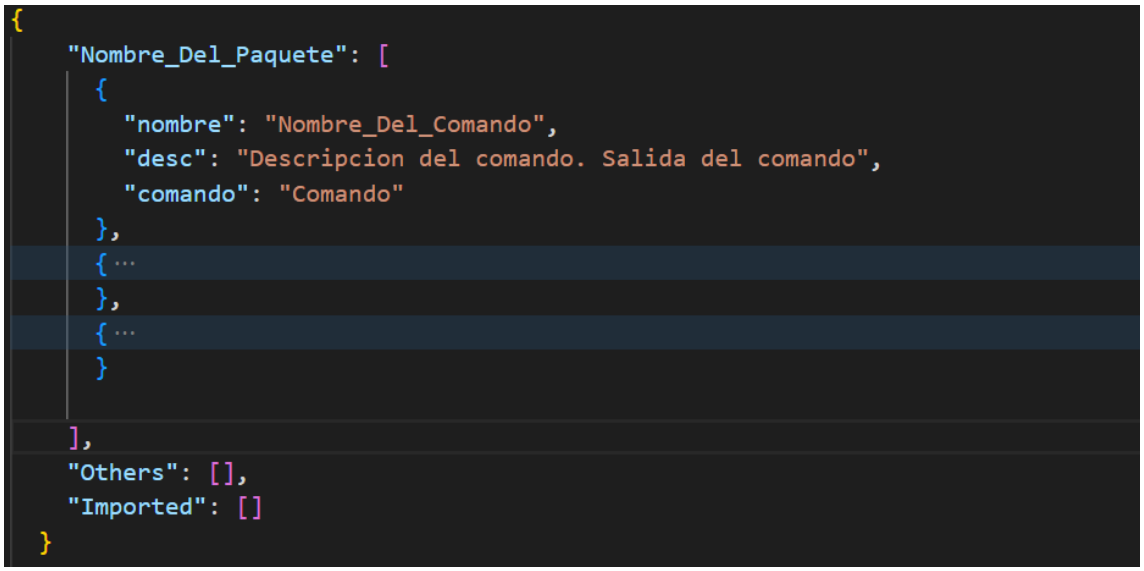


Figura 9. 14. Estructura ejemplo del JSON

El nombre del paquete se usa para organizar los comandos en clases con funcionalidades similares. Dentro de cada paquete de comandos tenemos los comandos, que están formados por:

- Nombre: El nombre del comando
- Desc: La descripción del comando, después del punto, la salida del comando
- Comando: El comando en sí

Además, se ha de comentar dos paquetes de comandos especiales:

- Others: Es dónde se guardan los comandos que crean los usuarios
- Imported: Es dónde se guardan los comandos importados

A continuación, podemos ver un ejemplo de cómo sería una entrada real en el sistema:

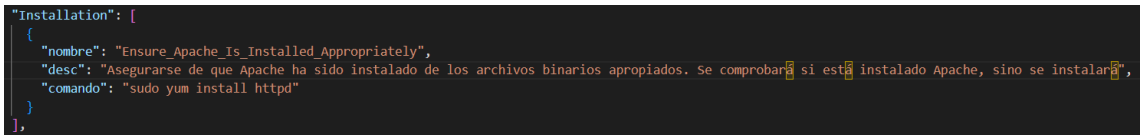


Figura 9. 15 Ejemplo real de una entrada JSON

Después, se debería modificar el método de la clase Lang.py llamado check_os (), que es el que reconoce los sistemas operativos del sistema.

9.4.2 Creación del ejecutable

Un punto importante a la hora de realizar cambios en el código es cómo crear el ejecutable de Windows de nuevo. Para hacer esto seguiremos los siguientes pasos:

1. Instalar con el comando: pip install auto-py-to-exe
2. Una vez instalado ejecutarlo con el comando: auto-py-to-exe
3. Se muestra la siguiente pantalla:

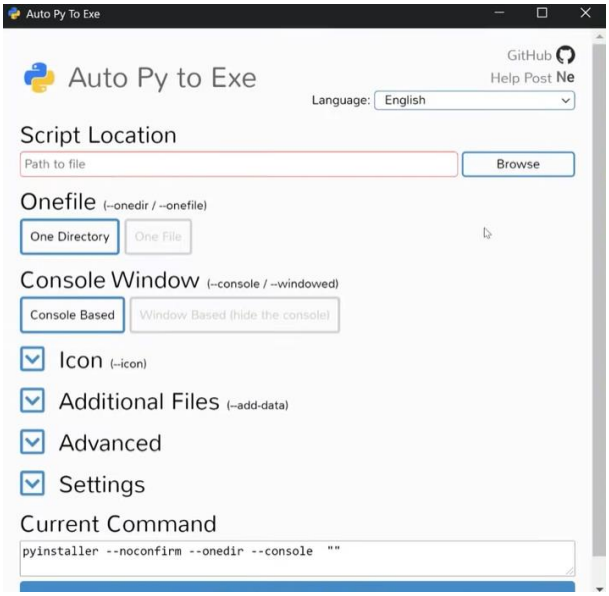


Figura 9. 16. Pantalla principal auto-py-to-exe

4. En *browse* seleccionamos el archivo main.py
5. En *additional files* seleccionamos los siguientes archivos:
    a. comandosApacheCentos.json
    b. comandosApacheWindows.json
    c. comandosApacheUbuntu.json
    d. history.txt
    e. conf/ configurationCentos.txt
    f. conf/ configurationWindows.txt
    g. conf/ configurationUbuntu.txt
6. En *Settings* vamos *Output Directory* y seleccionamos la carpeta dist de nuestro proyecto
7. Pulsamos el botón inferior que muestra "*convert py to exe"*
8. En la carpeta dist se genera el archivo .exe además de lo necesario para que funcione.

Capítulo 10. Conclusiones y Ampliaciones

Conclusiones del sistema que hemos elaborado y cualquier labor de ampliación que tengamos contemplada en el sistema.

10.1 Conclusiones

En la fase de investigación de este trabajo de fin de grado no se encontró ninguna aplicación que cumpliese con las funcionalidades que se desarrollan aquí, por lo que se puede afirmar que es algo innovador.

La aplicación definitivamente consigue aumentar la seguridad de los servidores Apache, tiene herramientas para mantener los comandos actualizados o añadir unos nuevos. Estas funcionalidades están pensadas, además de para personalizar comandos, para poder mantener el sistema actualizado frente a nuevas vulnerabilidades que surjan.

La aplicación es sencilla de usar y permite que un usuario nuevo controles las funciones básicas viendo las respuestas a las pruebas de usabilidad en el punto 8.3. Esto viene de que el lenguaje específico diseñado es sencillo y de todas las herramientas que aumentan la usabilidad del sistema (autocompletar, mensajes de ayuda y mensajes sugerencia en los errores).

De los resultados del cuestionario vemos una respuesta muy positiva por parte de los encuestados. Ya que en todas las preguntas la opción más respondida es “Totalmente de acuerdo”. Sin embargo, los resultados de las preguntas Q2 y Q11 demuestran que el DSL puede ser mejorado. La pregunta Q2 hacía referencia a si el uso de la aplicación reducía la complejidad de configurar este tipo de servidores y la Q11 preguntaba si el usuario prefería esta metodología a hacerlo todo a mano.

El no tener referencias sobre proyectos similares requiere un mayor esfuerzo por parte del desarrollador, lo que influye positivamente para el progreso académico. Además del desarrollo en Python, el estudio de lenguajes de dominio específico y diferentes sistemas operativos.

En mi opinión, la aplicación es realmente útil, simple y fácil de usar, por lo que considero que cumple con los objetivos planeados.

Desarrollando esta aplicación he aprendido y profundizado sobre diversos campos como seguridad informática, lenguajes de dominio específico y sistemas. La aplicación y el trabajo me fueron resultando más interesantes mientras iba avanzando en el desarrollo, entendiendo que la aplicación podía convertirse en algo realmente útil.

Antes de empezar el proyecto, pensaba que iba a ser más sencillo y llevar menos tiempo, teniendo en cuenta que me llevo el primer semestre entero desarrollar la aplicación y hacer este

trabajo y una pequeña parte del segundo el realizar el artículo. No sabría si considerarlo difícil o fácil, pero si muy laborioso.

10.2 Ampliaciones

Entre las ampliaciones posibles que tenemos consideradas para el sistema encontramos:

- Aumentar la cantidad de comandos: a mayor cantidad de comandos, mayor es la posible seguridad y la configurabilidad que el usuario puede aplicar en su servidor. Estos comandos se podrían extraer consultando los errores de Apache que no estén cubiertos por la versión 2.4 y arreglándolos mediante comandos.
- Aumentar los sistemas operativos compatibles: se modificaría el código como se explica en la sección 3 de la parte 9.4 de esta documentación. Ahora mismo son compatibles para Windows, Ubuntu y CentOS, sería interesante tener una versión para MacOS, Solaris o Fedora. Aumentar los servidores con los que la aplicación sería compatible: ahora mismo la aplicación solo es compatible con servidor Apache. Por ejemplo, sería interesante que la aplicación fuese compatible con otros populares como Nginx, Microsoft ISS o Google Web Server.
- Gracias a los resultados del cuestionario sabemos que la aplicación podría ser mejorada aumentado su usabilidad.

Capítulo 11. Planificación del Proyecto y Presupuesto finales

Se desarrolla en esta sección la versión final de la planificación y presupuestos.

11.1 Planificación Final

La planificación final resulta de la siguiente manera:

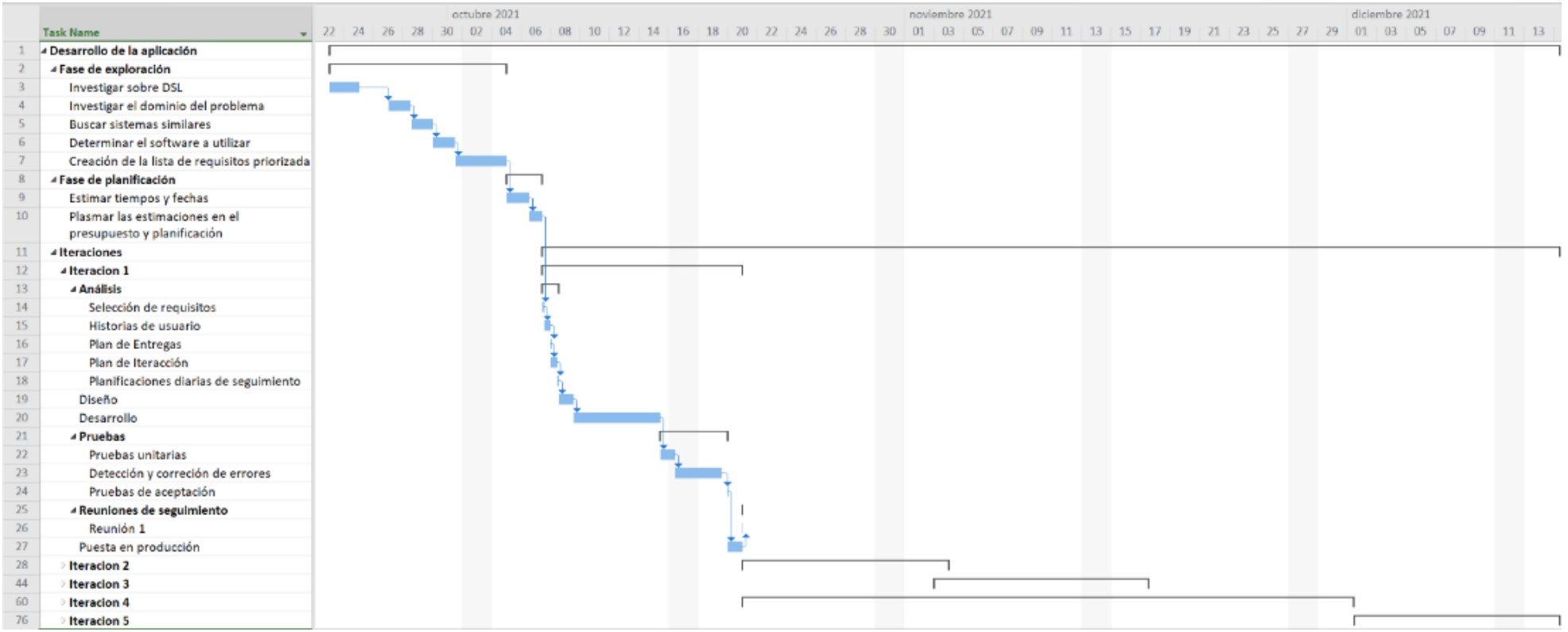


Figura 11. 1. Diagrama de Gantt parte 1

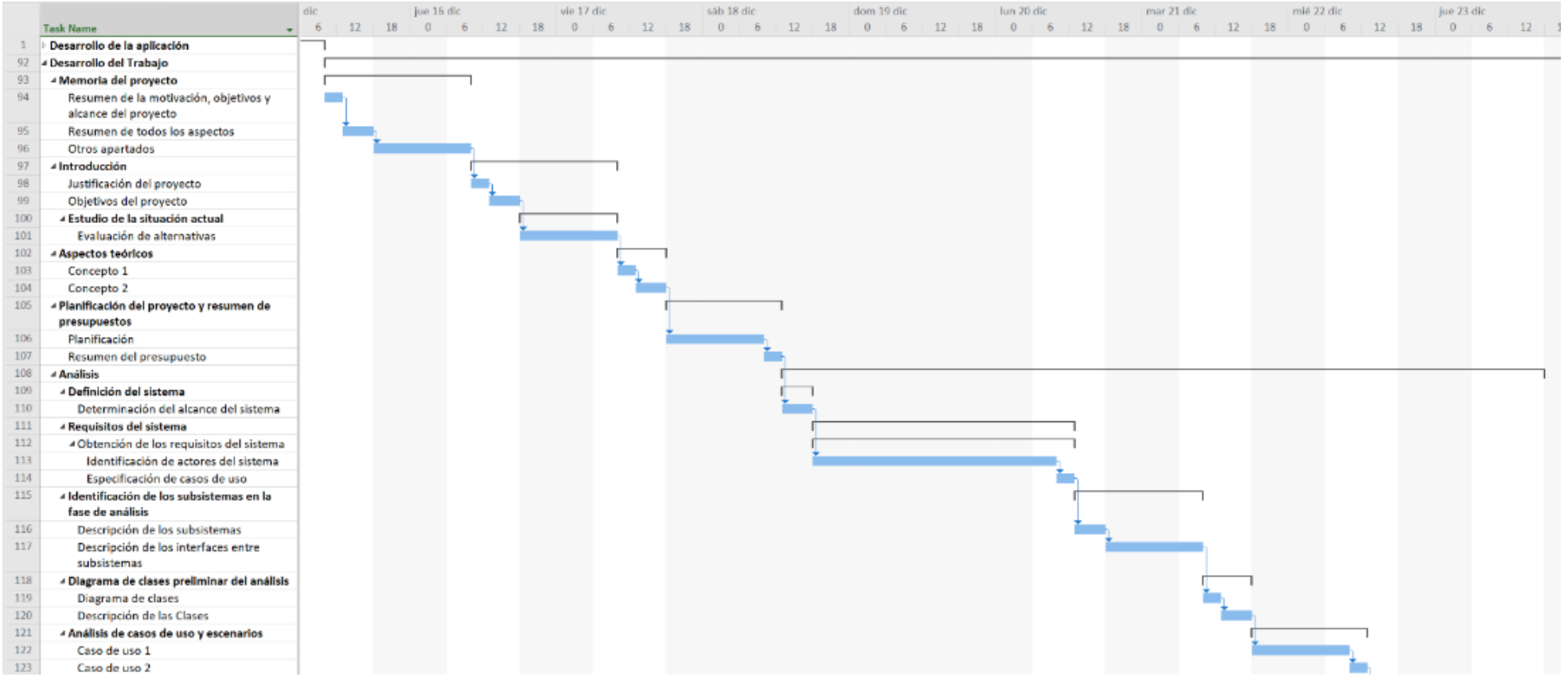


Figura 11. 2. Diagrama de Gantt parte 2

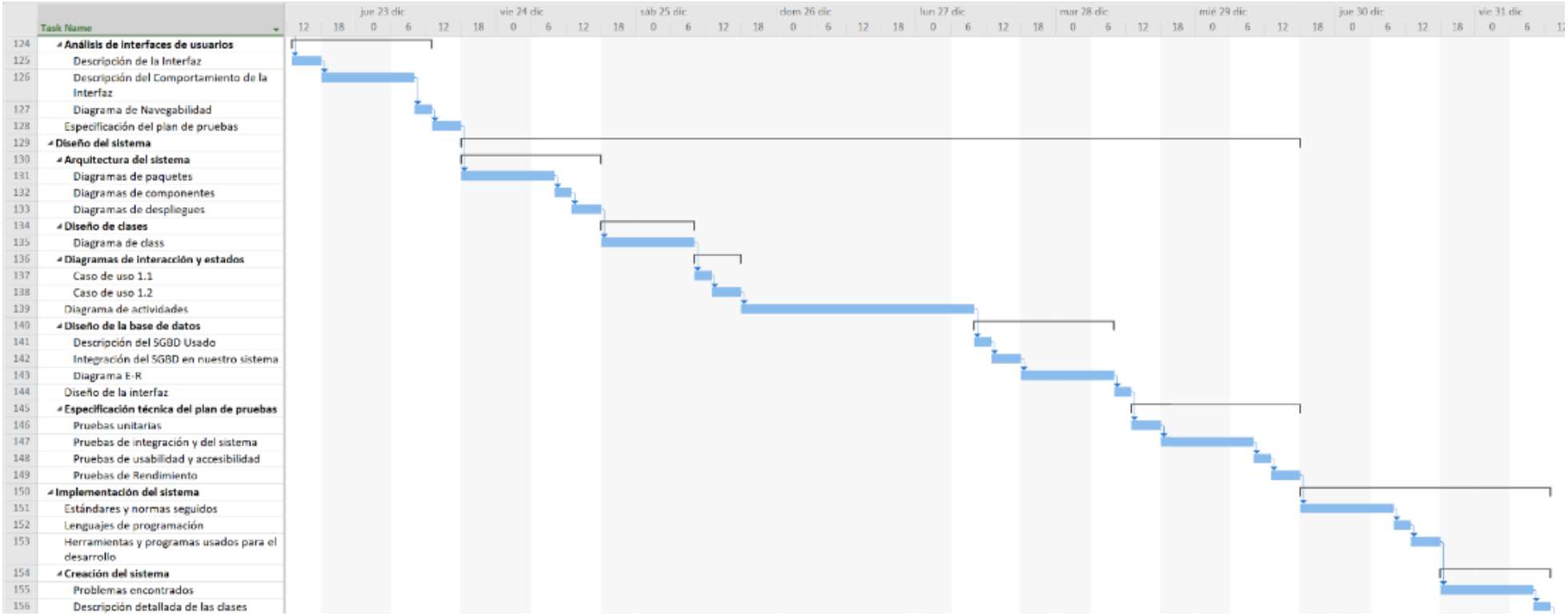


Figura 11. 3. Diagrama de Gantt parte 3

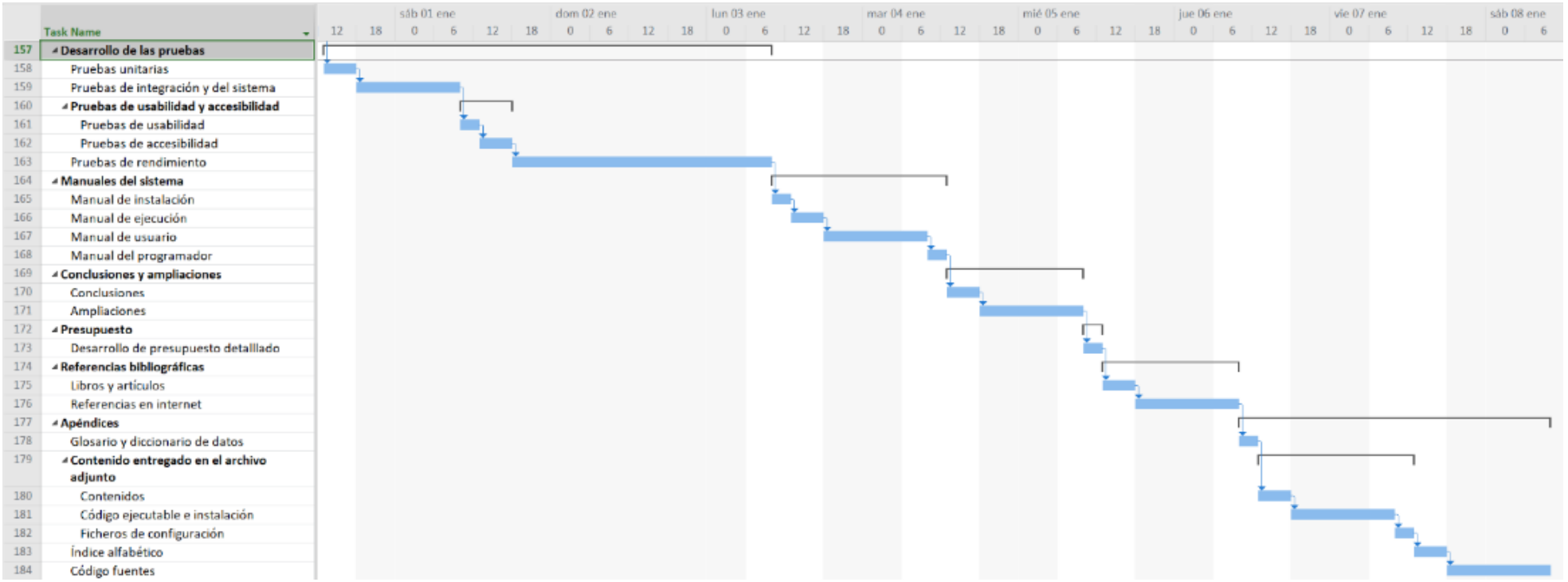


Figura 11. 4. Diagrama de Gantt parte 4

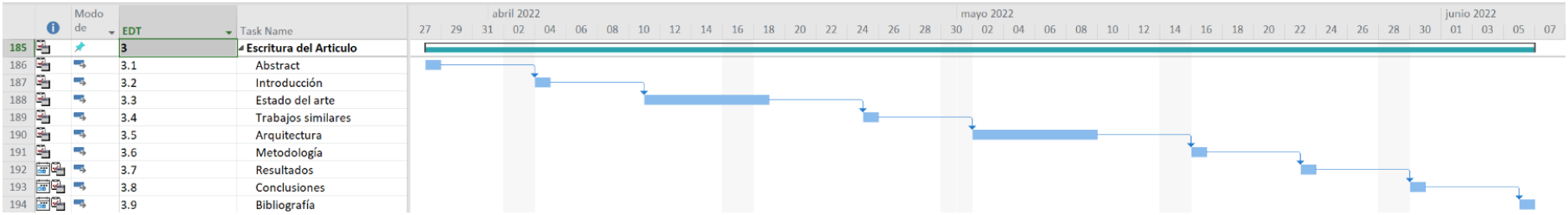


Figura 11. 5 Diagrama de Gantt parte 5

La diferencia con la planificación inicial es el tiempo de desarrollo. En primera instancia se había planeado dos iteraciones de desarrollo de la aplicación, esto fue un error de planificación debido al desconocimiento de cuanto llevaría el proyecto.

Como se ve en la planificación final en vez de dos iteraciones hay cinco, siendo cada iteración dos semanas, resulta en 6 semanas de diferencia. Estas 6 semanas de diferencia se pueden traducir en 216 horas de trabajo.

Además, también se incluye el tiempo que llevó escribir el artículo, que fueron otras 66 horas.

11.2 Presupuesto Final

En el presupuesto original de proyecto se había presupuestado la cantidad de 13291,62 euros, en este cálculo entraba el beneficio, el IVA y los posibles sobrecostes.

Se añade a continuación una captura del presupuesto final teniendo en cuenta este cambio y sin contar en este caso beneficios ni posibles sobrecostes, ya que es la versión final:

Presupuesto del cliente				
Código Partida	**Item**	**Partida**	**Importe**	**Total**
1		Desarrollo de la Aplicación		9.785,31 €
	1	Fase de Exploración	1.362,25 €	
	2	Fase de Planificación	364,00 €	
	3	Iteración 1	1.611,81 €	
	4	Iteración 2	1.611,81 €	
	5	Iteración 3	1.611,81 €	
	6	Iteración 4	1.611,81 €	
	7	Iteración 5	1.611,81 €	
2		Desarrollo del Trabajo		2.894,31 €
	1	Memoria del proyecto	137,13 €	
	2	Introducción	114,44 €	
	3	Aspectos teóricos	137,13 €	
	4	Planificación del proyecto y resumen del presupuesto	91,75 €	
	5	Análisis	636,25 €	
	6	Diseño del sistema	636,25 €	
	7	Implementación del sistema	227,88 €	
	8	Desarrollo de las pruebas	227,88 €	
	9	Manuales del sistema	182,50 €	
	10	Conclusiones y ampliaciones	91,75 €	
	11	Presupuesto	46,38 €	
	12	Referencias bibliográficos	91,75 €	
	13	Apéndices	273,25 €	
3		Escritura del artículo		1.498,38 €
4		Recursos materiales		967,96 €
Total				**15.145,96 €**

Figura 11. 6. Presupuesto del cliente final

Como vemos hay un desajuste de 1.854,34 euros en nuestra contra, todo esto sin contar los beneficios. Si tuviésemos en cuenta los beneficios se produciría un desajuste de 4.949,12 euros.

Esto se debe en parte a la escritura del artículo, que no se había previsto en la planificación original y en parte a las 3 iteraciones extra que no habíamos planeado originalmente.

Capítulo 12. Referencias Bibliográficas

Libros, artículos y referencias de internet usadas para el desarrollo del trabajo.

12.1 Libros y Artículos

[1] Spinellis, Diomidis; Gritzalis Dimitris.” A Domain-specific Language for Intrusion Detection”. 2000.

[2] Hurd, Joe; Carlsson, Magnus; Letner, Brett; White, Peter.” Lobster: A Domain Specific Language for SELinux Policies”. 2008.

[3] Hamdi, Hédi; Mosbah, Mohamed.” A DSL Framework for Policy-based Security of Distributed Systems”. University de Bordeaux, 20077.

[4] K. Czarnecki, “Generative programming: Methods, techniques, and applications: Tutorial abstract,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2319, pp. 351–352, 2002, doi: 10.1007/3-540-46020-9_38.

12.2 Referencias en Internet

[5] Soto, Jason. "JShielder"**.** https://jsitech1.gitbooks.io/jshielder-linux-server-hardening-script/content/jshielder1.html

[6] "Apache Foundation". https://www.apache.org/

[7] "JetBrains". https://www.jetbrains.com/

[8] "Center for Internet Security"**.** https://www.cisecurity.org/

[9] "Auto-py-to-exe https://github.com/brentvollebregt/auto-py-to-exe

[10] Paya, A., Cotarelo, A., & Redondo, J. (2022). Egida: Automated security configuration deployment systems with early error detection. *Elsevier*. https://doi.org/10.1016/j.cose.2022.102638

[11] "VirtualBox"**.** https://www.virtualbox.org/

[12] N. Ford, "No Fluff, Just Stuff Anthology (Pragmatic Programmers)," 2006, Accessed: Mar. 26, 2022. [Online]. Available: https://dl.acm.org/doi/abs/10.5555/1199389

[13] R. L.-A. of psychology and undefined 1932, "A technique for the measurement of attitudes.," *psycnet.apa.org*, Accessed: Apr. 24, 2022. [Online]. Available: https://psycnet.apa.org/record/1933-01885-001

Capítulo 13. Apéndices

13.1 Glosario y Diccionario de Datos

- **Apache**: Es un servidor HTTP de código abierto para plataformas Unix, Microsoft Windows. Macintosh y otras.
- **Bash**: Es una interfaz de usuario de línea de comandos de Unix.
- **Benchmark**: Una prueba de rendimiento o comparativa utilizada para medir el rendimiento de un sistema o uno de sus componentes.
- **CentOS**: Distribución Linux que consiste en una bifurcación a nivel binario de la distribución RHEL.
- **CIS**: *Center for Internet Security*, es una organización sin ánimo de lucro dedicada a combatir las ciber amenazas.
- **CLI**: La interfaz de línea de comandos es un tipo de interfaz de usuario que permite al usuario dar órdenes al sistema operativo por medio de una línea de texto.
- **CVE:** *Common Vulnerabilities and Exposures* es una lista de información registrada sobre vulnerabilidad de seguridad conocidas.
- **DSL**: *Domain-Specific Language* , lenguaje de dominio específico es un lenguaje de programación o especificación dedicado a resolver un problema en particular.
- **EDT**: Estructura de desglose de trabajo, es una herramienta fundamental que consiste en la descomposición jerárquica.
- **GUI**: La interfaz gráfica de usuario es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles.
- **Hardening**: EL endurecimiento es el proceso de asegurar un sistema reduciendo sus vulnerabilidades.
- **HTTP**: Protocolo de transferencia de hipertexto, es el protocolo de comunicación que permite las transferencias de información a través de archivos en la World Wide Web.
- **IDE**: Entorno de desarrollo integrado, es una aplicación informática que proporciona servicios integrales para facilitarle al programador el desarrollo software.
- **IEEE**: Instituto de Ingenieros Eléctricos y Electrónicos, es una asociación mundial de ingenieros dedicada a la normalización y el desarrollo en áreas técnicas.
- **JSON**: Es un formato de texto sencillo para el intercambio de datos.
- **Linux**: Denominación técnica que reciben una serie de sistemas operativos de tipo Unix.
- **OpenSCAP:** Guías de endurecimiento y configuración para sistemas informáticos.
- **PyCharm**: Un IDE de Python.
- **Python**: Es un lenguaje de programación.
- **Seguridad**: Refiriéndose a la ciberseguridad, es el área relacionada con la protección de la infraestructura computacional y todo lo vinculado en la misma.
- **Servidor**: Conjunto de computadores capaz de atender las peticiones de un cliente y devolverá una respuesta en concordancia.

- **Sistema Operativo**: Es el conjunto de programas de un sistema informático que gestiona los recursos de hardware y provee servicios a los programas de aplicación de software.
- **Ubuntu**: Distribución de Linux basada en Debian.
- **UNIX**: Sistema operativo portable, multitarea y multiusuario.
- **Usabilidad**: Facilidad con que las personas puede utilizar una herramienta particular.
- **VirtualBox**: Software de virtualización para arquitecturas x86/amd64.
- **Vulnerabilidad**: Debilidad en el software o hardware que permite a un atacante comprometer la integridad, disponibilidad o confidencialidad del sistema o sus datos.
- **Windows**: Nombre de una familia de distribuidores de software para PC y servidores, sistemas empotrados.

13.2 Contenido Entregado en el Archivo adjunto

Se especifica en esta sección la estructura del contenido que se encuentra en la carpeta entregada.

13.2.1 Contenidos

La estructura del archivo adjunto es la siguiente:

Directorio	Contenido
./ Directorio raíz	Contiene un fichero leeme.txt explicando toda esta estructura y la licencia Apache en LICENSE.txt
./DSL	Contiene toda la estructura de directorios del proyecto para desarrollo. **Ver la tabla de estructura de directorios de desarrollo**.
./documentacion	Contiene toda la documentación asociada al proyecto.
./documentacion/img	Directorio que contiene las imágenes utilizadas en la documentación. Estas imágenes tendrán formato *.png* si son capturas de pantalla, *.wmf* si son diagramas o esquemas y *.jpg* sólo si son fotografías.
/articulo	Incluirá una copia del artículo que se ha hecho sobre la aplicación

Figura 13. 1. Estructura del archivo adjunto

Directorio	Contenido
./ Directorio raíz de "DSL"	Contiene los ficheros de proyecto del IDE utilizado. Los módulos de Python y los archivos JSON.
./idea.	Directorio para ejecutar el proyecto con PyCharm (No es relevante)
./__pycache__	Directorio que surge al ejecutar el programa en Python (No es relevante)

./build	Contiene el build de PyCharm (No es relevante)
./conf	Contiene los diferentes ficheros de configuración del proyecto. En este caso tres: • configurationCentos • configurationUbuntu • configurationWindows
./dist	Directorio donde se sitúan los ficheros para la distribución del proyecto.
./doc	Contiene toda la documentación relativa a las clases. Se encontraría dentro de la carpeta build/html
./venv	Contiene diferentes archivos para que PyCharm funcione adecuadamente (No es relevante)

Figura 13. 2. Estructura de directorios de desarrollo

13.2.2 Código Ejecutable e Instalación

Como explicamos en más profundidad en el manual de instalación, habría dos versiones. Para el sistema operativo Windows, simplemente hay que ir a la carpeta "dist" y hacer doble clic sobre el archivo main.exe. Para las distribuciones Linux se abre la carpeta dist y el terminal. Se deberían instalar los siguientes módulos de Python, con la orden sudo pip install *module*, siendo *module* los siguientes módulos:

- deep_translator
- keyboard
- distro
- datetime

Después se ejecutaría con la orden sudo main.py

13.2.3 Ficheros de Configuración

Los ficheros de configuración se encuentran, como se explica en el punto 9.1 y 13.2.1, son tres dependiendo del sistema operativo en el que se ejecute la aplicación. Para la version de Windows se debe indicar la ruta a la carpeta Apache. En la versión Linux se debe indicar ,además, dónde se encuentra la carpeta PASSWD.

13.3 Índice Alfabético

13.4 Código Fuente

El código fuente se encuentra, como se indica en la sección 13.2, en la carpeta "DSL (Lenguaje de Dominio Específico) para configurar un servidor de forma segura".

A la vez dentro de esta carpeta encontramos los siguientes archivos:

- comandosApacheCentos.json
- comandosApacheUbuntu.json
- comandosApacheWindows.json
- configurationCentos
- configurationUbuntu
- configurationWindows
- CVE-updateUbuntu-1.json
- dsl.py
- history.txt
- interpreter.py
- json_manager.py
- lang.py
- lexer.py
- main.py
- prueba1.json

Y encontramos también las siguientes carpetas:

- build
- dist
- docs
- output
- venv

13.5 Propiedad intelectual y licencias

Se mencionan a continuación las licencias utilizadas para completar el proyecto:

- JetBrains licencia educativa: se usa esta para poder usar el programa en el que se desarrolla el programa, PyCharm.
- Office365 licencia Universidad de Oviedo: se usa esta para el sistema operativo Windows, para el editor de texto Word, para Excel y para Project.

En cuanto a la licencia del propio proyecto, se usa la licencia de Apache, Apache License Version 2.0, se incluye la licencia en el texto LICENSE.txt en la carpeta distribuida.

13.6 Actas de reuniones

Reunión 23 de Septiembre:

- Resumen: Hablar de la metodología de trabajo, de correcciones y mejoras tanto en el presupuesto como en la planificación. Hablar de la necesidad de hacer una gestión de riesgos.
- Duración:25 minutos
- Medio de comunicación: Microsoft Teams
- Propuestas de la siguiente reunión: Hacer las correcciones en el presupuesto y la planificación y revisar la nueva plantilla. Además, investigar sobre el dominio y los lenguajes específicos de dominio.

Reunión 7 de Octubre:

- Resumen: Enseñar las correcciones del presupuesto y la planificación. Comentar la idea que se tiene para el desarrollo de la aplicación y el lenguaje del DSL.
- Duración: 20 minutos
- Medio de comunicación: Microsoft Teams
- Propuestas de la siguiente reunión: Hacer la investigación inicial en busca de sistemas que tengan similitudes con lo que se quiere desarrollar.

Reunión 21 de Octubre:

- Resumen: Comentar los resultados de la investigación y la intención que se tiene para la aplicación dados los resultados de la búsqueda de trabajos similares
- Duración: 20 minutos
- Medio de comunicación: Microsoft Teams
- Propuestas de la siguiente reunión: Tener hecha una primera versión de la aplicación que se ejecute en Linux

Reunión 4 de Noviembre:

- Resumen: Enseñar una primera versión de la aplicación y comentar ideas a aplicar para el siguiente sprint.
- Duración: 20 minutos
- Medio de comunicación: Microsoft Teams
- Propuestas de la siguiente reunión: Añadir nuevas funcionalidades a la aplicación.

Reunión 18 de Noviembre:

- Resumen: Enseñar las nuevas funcionalidades de la aplicación
- Duración: 17 minutos
- Medio de comunicación: Microsoft Teams
- Propuestas de la siguiente reunión: Añadir nuevas funcionalidades, permitir que la aplicación se ejecute en Windows

Reunión 2 de Diciembre:

- Resumen: Enseñar las nuevas funcionalidades de la aplicación añadidas, como la posibilidad de ejecución en sistemas Windows
- Duración: 15 minutos
- Medio de comunicación: Microsoft Teams
- Propuestas de la siguiente reunión: Corregir y añadir nuevas funcionalidades. Empezar a trabajar en la documentación

Reunión 23 de Diciembre:

- Resumen: Enseñar las nuevas funcionalidades de la aplicación añadidas. Comentar el avance de esta documentación.
- Duración: 10 minutos
- Medio de comunicación: Microsoft Teams
- Propuestas de la siguiente reunión: Acabar una primera versión de este trabajo y crear cuestionarios para que otros compañeros prueben la aplicación.

Reunión 13 de Enero:

- Resumen: Comentar la revisión de la primera versión del TFG.
- Duración: 10 minutos.
- Medio de comunicación: Microsoft Teams
- Propuestas de la siguiente reunión: Corregir los errores, finalizar las secciones que no estaban acabadas, mandar los cuestionarios a usuarios para que prueben la aplicación.

Reunión 27 de Enero:

- Resumen: Comentar la revisión de la segunda versión del TFG.
- Duración: 5 minutos
- Medio de comunicación: Microsoft Teams
- Propuestas de la siguiente reunión: Finalizar la tercera versión

Reunión 24 de Febrero:

- Resumen: Comentar la revisión de la tercera versión del TFG
- Duración: 16 minutos
- Medio de comunicación: Microsoft Teams
- Propuestas de la siguiente reunión: Arreglar los últimos errores. Empezar el artículo.

Reunión 10 de Marzo:

- Resumen: Comentar la primera versión del artículo.
- Duración: 10 minutos
- Medio de comunicación: Microsoft Teams
- Propuestas de la siguiente reunión: Hacer el *State of Art* y el diagrama de capas.

Reunión 24 de Marzo:

- Resumen: Comentar la segunda versión del artículo.
- Duración: 12 minutos
- Medio de comunicación: Microsoft Teams
- Propuestas de la siguiente reunión: Arreglar lo comentado en la reunión.

Reunión 7 de Abril:

- Resumen: Comentar la tercera versión del artículo.
- Duración: 17 minutos
- Medio de comunicación: Microsoft Teams
- Propuestas de la siguiente reunión: Arreglar lo comentado en la reunión y hacer secciones de los casos de uso y trabajos similares.

Reunión 21 de Abril:

- Resumen: Comentar la cuarta versión del artículo.
- Duración: 9 minutos
- Medio de comunicación: Microsoft Teams
- Propuestas de la siguiente reunión: Arreglar lo comentado en la reunión, hacer secciones *Used Hardware and Software* y *Evaluation and Discussion.*

Reunión 5 de Mayo:

- Resumen: Comentar la quinta versión del artículo.
- Duración: 4 minutos
- Medio de comunicación: Microsoft Teams
- Propuestas de la siguiente reunión: Finalizar la introducción y arreglar pequeñas partes.

Reunión 19 de Mayo:

- Resumen: Comentar la sexta versión del artículo. Revisar la documentación del TFG para ver que todo este correcto.
- Duración: 9 minutos
- Medio de comunicación: Microsoft Teams
- Propuestas de la siguiente reunión: Finalizar el artículo y revisar la documentación.

Reunión 2 de Junio:

- Resumen: Finalizar detalles tanto del trabajo como del artículo.
- Duración: 12 minutos
- Medio de comunicación: Microsoft Teams
- Propuestas de la siguiente reunión: Finalizar ambos documentos y revisarlo todo.