UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

GCA

Agente para la Gestión Centralizada de la Configuración de Red en las aulas

Presentado por Rafael Martín Guerrero
en Universidad de Burgos — 9 de junio de 2025
Tutor: Pedro Renedo Fernández

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. Pedro Renedo Fernández, profesor del departamento de Ingeniería Informática, área de Lenguajes y Sistemas Informáticos.

Expone:

Que el alumno D. Rafael Martín Guerrero, con DNI 80087034E, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado GCA.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 9 de junio de 2025

Vº. Bº. del Tutor:

Pedro Renedo Fernández

Resumen

En plena era digital, los docentes y administradores de centros educativos se enfrentan a numerosos desafíos como consecuencia de la integración de la tecnología en las aulas. Resulta cada vez más complicado restringir el acceso a ciertos recursos y mantener un entorno de aprendizaje seguro y eficiente.

Este proyecto tiene como objetivo desarrollar e implementar una plataforma web y un agente que permitan la gestión centralizada de la configuración de red en los equipos de las aulas. La solución permite la segmentación de los dispositivos en grupos y la aplicación de restricciones personalizadas para el acceso a recursos internos y externos, incluyendo Internet. Se busca optimizar el control y administración de la red educativa, mejorando la seguridad y la eficiencia en la gestión de los permisos de acceso.

Toda la información del proyecto, incluyendo el código fuente, se encuentra disponible en el repositorio de GitHub accesible a través de la siguiente url: https://github.com/rmg1008/GCA.

Descriptores

Plataforma web, configuración de red, administración educativa, agente, control, segmentación, aplicación Java

Abstract

In the digital age, teachers and administrators in educational centers face numerous challenges as a result of the integration of technology in the classrooms. It is increasingly difficult to restrict access to certain resources and maintain a safe and efficient learning environment.

This project aims to develop and implement a web platform and an agent that allow centralized management of network configuration on classroom devices. The solution enables the segmentation of devices into groups and the application of customized restrictions for accessing both internal and external resources, including the Internet. The goal is to optimize control and administration of the educational network, improving security and efficiency in managing access permissions.

All project information, including the source code, is available in the GitHub repository accessible at the following URL: https://github.com/rmg1008/GCA.

Keywords

Web platform, network configuration, educational management, agent, control, segmentation, Java application

Índice general

Índice de figuras

Índice de tablas

1. Introducción

La Universidad de Burgos (UBU) cuenta con más de 1000 equipos informáticos distribuidos en las aulas de sus centros; cada una de ellas dispone de dispositivos destinados a satisfacer necesidades muy diversas (resolución de operaciones matemáticas complejas, redacción, edición multimedia... etc.). No obstante, dado que un mismo equipo no está enfocado a un único uso, el acceso ilimitado a todo tipo de herramientas de las que dispone puede derivar en un uso indebido por parte del usuario y suponer un riesgo para la seguridad de la información: en este contexto, por ejemplo, puede utilizarse fraudulentamente por los estudiantes a la hora de realizar una prueba de evaluación, como podría ser el acceso a información no autorizada o transferir archivos entre dispositivos de red.

En los últimos años, se ha observado un incremento de las peticiones entre los docentes para limitar los recursos a los que se pueden acceder desde los ordenadores. Por este motivo, surge la necesidad de implementar una gestión centralizada en las aulas, que permita de una manera sencilla controlar los accesos de los equipos, atendiendo a características diferentes, como nombres de equipo, aulas, centros...

Actualmente disponemos de herramientas comerciales capaces de gestionar el acceso a los recursos de manera eficiente; sin embargo, su elevado coste económico puede hacerlas poco rentables. Por otro lado, optar por un enfoque manual —donde un administrador configura y audita cada dispositivo individualmente— conlleva, inevitablemente, un notable aumento de la carga de trabajo y, en consecuencia, la posible necesidad de incorporar más personal, además de elevar el riesgo de errores humanos.

La solución que se propone en este proyecto es crear una plataforma para la Gestión Centralizada de las Aulas (GCA), con la que se pretende

optimizar costes y ofrecer una solución práctica, diseñada a medida para la UBU. El gran potencial de esta iniciativa es que se enfoca en resolver problemas reales sin añadir funciones innecesarias que acaban dificultando su uso.

GCA es capaz de:

1. **Registrar** cada dispositivo en un servidor centralizado utilizando una aplicación cliente.
2. **Aplicar** de forma automática políticas de control de acceso (IP, puertos, servicios...)
3. **Gestionar** remotamente las configuraciones individualmente o por grupos.

En los apartados siguientes del presente proyecto se expondrán de forma más amplia y pormenorizada los objetivos específicos de este, así como una guía del trabajo de programación llevado a cabo bajo una metodología definida.

Figura 1.1: Logo de GCA

2. Objetivos del proyecto

Existen diferentes objetivos que han motivado el nacimiento del proyecto:

2.1. Objetivos generales

- Crear una plataforma para el control de acceso a los recursos de los dispositivos del centro.
- Contribuir a la automatización de procesos de la Universidad.
- Solucionar las necesidades del personal docente.
- Guardar las distintas configuraciones y permitir una gestión eficiente.
- Permitir ahorrar en costes al desarrollar una aplicación a medida.

2.2. Objetivos técnicos

- Crear una aplicación web para permitir la gestión de los dispositivos de una manera sencilla e intuitiva.
- Desarrollar una aplicación Java para gestionar las configuraciones a implementar.
- Utilizar herramientas de control de versiones como Git y GitHub.
- Utilizar una arquitectura MVC (Modelo-Vista-Controlador).
- Aplicar una metodología Ágil (Scrum) en el desarrollo del proyecto.

- Utilizar Zube para gestión de proyectos.
- Aplicar TDD (Test Driven Development) durante todo el desarrollo, para garantizar calidad en el producto.
- Implementar un marco DevOps orientado a mejorar la colaboración entre desarrollo y operaciones.

2.3. Objetivos personales

- Aportar una herramienta práctica, útil y beneficiosa para la Universidad.
- Aplicar los conocimientos aprendidos durante el Grado.
- Ampliar conocimientos técnicos:
  - Gestión de dispositivos.
  - Python
  - Java.
  - Angular.
  - Herramientas de testeo.
- Profundizar en la gestión de políticas de control de acceso mediante comandos.
- Aprender a crear aplicaciones de escritorio.

3. Conceptos teóricos

El funcionamiento del proyecto se divide en varias fases.

1. **Registro de dispositivos:** cada dispositivo se registra usando la aplicación cliente y se mantiene a la espera de recibir configuraciones por parte de la API.
2. **Configuración inicial:** una vez registrado, el gestor asigna una localización específica al dispositivo.
3. **Asignación de configuraciones:** se asignan configuraciones a cada dispositivo o al grupo que pertenece.
4. **Aplicación de configuraciones:** mediante la aplicación cliente, se obtiene la configuración asignada para el dispositivo y se aplican los comandos correspondientes.

Para entender mejor cómo se desarrolla cada fase, es imprescindible explicar algunos conceptos teóricos relacionados con la gestión de configuraciones, protocolos de red y herramientas de administración.

3.1. Comunicación mediante API REST

Las comunicaciones entre el cliente y el servidor, así como entre el servidor y la interfaz web, se realizan mediante APIs REST.

Una API REST (Representational State Transfer) es un conjunto de definiciones y protocolos utilizados para integrar sistemas entre aplicaciones[**?**]: permite realizar comunicaciones entre consumidores (aquellos que realizan

las llamadas) y proveedores (aquellos que reciben la llamada y devuelven la respuesta). La información se transmite en formato JSON (JavaScript Object Notation), que es un formato de intercambio de datos ligero y fácil de leer. Esta comunicación hace uso del protocolo HTTP (Hypertext Transfer Protocol), que es el protocolo de transferencia de hipertexto utilizado en la World Wide Web. HTTP utiliza una serie de métodos para definir las diferentes acciones, los más comunes son:

- **GET**: Recupera información del servidor.
- **POST**: Envía datos al servidor para crear un nuevo recurso.
- **PUT**: Actualiza un recurso existente en el servidor.
- **DELETE**: Elimina un recurso del servidor.

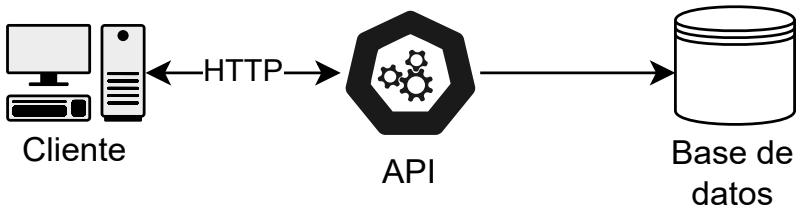

Figura 3.1: Funcionamiento de una API.

Autenticación

Para la autenticación de los usuarios, se utilizan tokens JWT (JSON Web Tokens)[**?**]. Estos tokens son objetos que contienen información codificada de datos del usuario, así como la fecha de expiración, y permiten transmitir información entre cliente y servidor de forma segura, ya que pueden ser firmados y verificados.

La estructura de un token JWT consta de tres partes:

Figura 3.2: Componentes de un JWT[**?**]

- **Header**: Contiene información sobre el tipo de token y el algoritmo utilizado para firmarlo.
- **Payload**: Contiene la información del usuario y los datos adicionales.
- **Signature**: Se utiliza para verificar que el emisor del token es quien dice ser y para asegurar que no ha sido modificado.

El token se envía en la cabecera de la solicitud HTTP, lo que permite al servidor verificar la autenticidad del usuario y otorgar acceso (o no) al recurso solicitado.

3.2. Gestión de configuraciones

Para la gestión de configuraciones utilizamos una herramienta de creación propia que se puede instalar en cualquier sistema operativo, ya sea Windows, Linux o MacOS. Una vez instalada en los equipos, esta herramienta se mantiene en segundo plano ejecutándose de manera ininterrumpida, encargándose de solicitar a la API las configuraciones que le correspondan.

Se gestiona a través de una interfaz web, mediante la cual se crean comandos que se agrupan en plantillas y se aplican a cada dispositivo de forma individual o grupos de dispositivos. Las plantillas incluyen un orden de ejecución de los comandos, lo que permite establecer prioridades. Por ejemplo, una plantilla se asigna a un grupo de dispositivos, y a su vez, otra plantilla se asigna a un dispositivo de ese mismo grupo: por el orden de prioridad establecido, esta segunda plantilla asignada a este último dispositivo tiene prioridad sobre la plantilla del grupo.

3.3. Comandos y protocolos para la gestión del acceso a la red

Para la gestión de acceso a la red en sistemas operativos Windows, se utilizan comandos y protocolos específicos.

Netsh

Netsh (Network Shell) es una herramienta de línea de comandos de Microsoft que permite la configuración y gestión de dispositivos de red en sistemas operativos Windows.

Su sintaxis es la siguiente:

```
netsh [-a <Aliasfile>]
   [-c <Context>]
   [-r <Remotecomputer>]
   [-u [<domainname>\<username>]
   [-p <Password> |
   [{<NetshCommand> | -f <scriptfile>}]
```

Una explicación mas detallada puede encontrarse en la documentación oficial de Microsoft[**?**].

Firewall

El firewall[**?**] es un sistema de seguridad de red que actúa como una barrera entre una red interna y una red externa. Su principal función es controlar el tráfico de red, permitiendo o bloqueando paquetes de datos según un conjunto de reglas de seguridad. Los firewalls pueden implementarse en hardware, software o de manera híbrida. En el caso de Windows, el firewall se implementa como un software integrado dentro del sistema operativo. En nuestro caso particular, administramos el firewall de Windows mediante la consola de comandos:

```
netsh advfirewall firewall
 [add | delete | set | show]
 [<RuleName>]
 [<RuleType>]
 [<LocalIP>]
 [<RemoteIP>]
 [<LocalPort>]
 [<RemotePort>]
 [<Protocol>]
 [<Action>]
 [<Direction>]
 [<Program>]
 [<Service>]
 [<InterfaceType>]
```

Mediante la ejecución de este comando, se pueden añadir, eliminar o modificar reglas del firewall: por ejemplo, habilitar o deshabilitar un puerto específico, bloquear un programa o servicio, e incluso deshabilitar el firewall [**?**].

PowerShell

PS (PowerShell)[**?**] es una solución para la automatización de tareas multiplataforma desarrollada por Microsoft. Combina un shell de línea de comandos, un lenguaje de scripting y un marco de administración de configuración.

La sintaxis de un comando de PowerShell es la siguiente:

```
Get-Command [[-ArgumentList] <Object[]>][-Verb <string[]>]
[-Noun <string[]>] [-Module <string[]>]
[-FullyQualifiedModule <ModuleSpecification[]>] [-TotalCount
<int>] [-Syntax] [-ShowCommandInfo] [-All] [-ListImported]
[-ParameterName <string[]>] [-ParameterType <PSTypeName[]>]
[<CommonParameters>]

Get-Command [[-Name] <string[]>] [[-ArgumentList] <Object[]>]
[-Module <string[]>] [-FullyQualifiedModule
<ModuleSpecification[]>] [-CommandType <CommandTypes>]
[-TotalCount <int>] [-Syntax] [-ShowCommandInfo]
```

```
[-All] [-ListImported] [-ParameterName <string[]>]
[-ParameterType <PSTypeName[]>] [-UseFuzzyMatching]
[-FuzzyMinimumDistance <uint>] [-UseAbbreviationExpansion]
[<CommonParameters>]
```

Puede ser ejecutado mediante la consola de Windows, o mediante un script. En concreto, para este proyecto, se utilizará para automatizar la configuración de sistemas y aplicaciones e implementar políticas de seguridad y configuración de red.

Protocolos

Los protocolos de red son un conjunto de reglas y convenciones que permiten la comunicación entre dispositivos dentro de una red. A continuación, se describen aquellos más relevantes en el contexto del presente proyecto:

- **TCP/IP:** conjunto de protocolos fundamentales que permite la comunicación entre dispositivos en red. Incluye mecanismos para diferenciar aplicaciones mediante puertos. Cualquier configuración de red o del firewall afecta directamente a este protocolo[**?**].
- **HTTPS:** extensión segura del protocolo HTTP que utiliza SSL/TLS para cifrar la comunicación entre cliente y servidor. Es utilizado por la aplicación para el registro de dispositivos y la consulta de configuraciones a través de la API[**?**].
- **DNS:** sistema encargado de traducir nombres de dominio (más fáciles de recordar) a direcciones IP. Desde comandos como `netsh` es posible añadir o eliminar entradas relacionadas con el DNS[**?**].
- **DHCP:** protocolo que asigna direcciones IP dinámicas a dispositivos en una red. Aunque es utilizado por los dispositivos de la Universidad, no se contempla su modificación dentro del proyecto[**?**].
- **ICMP:** protocolo empleado para el envío de mensajes de diagnóstico y control, como ocurre con el comando `ping`. Su tráfico puede ser habilitado o bloqueado mediante reglas del firewall[**?**].
- **SMB:** protocolo de nivel de aplicación que permite compartir archivos, impresoras y otros recursos dentro de una red. Mediante SMB se puede restringir el acceso a determinados recursos por dispositivo[**?**].

- **ARP:** protocolo responsable de resolver direcciones IP en direcciones MAC dentro de una red local. Aunque es más avanzado, se puede utilizar para establecer restricciones a nivel de dispositivos concretos[**?**].

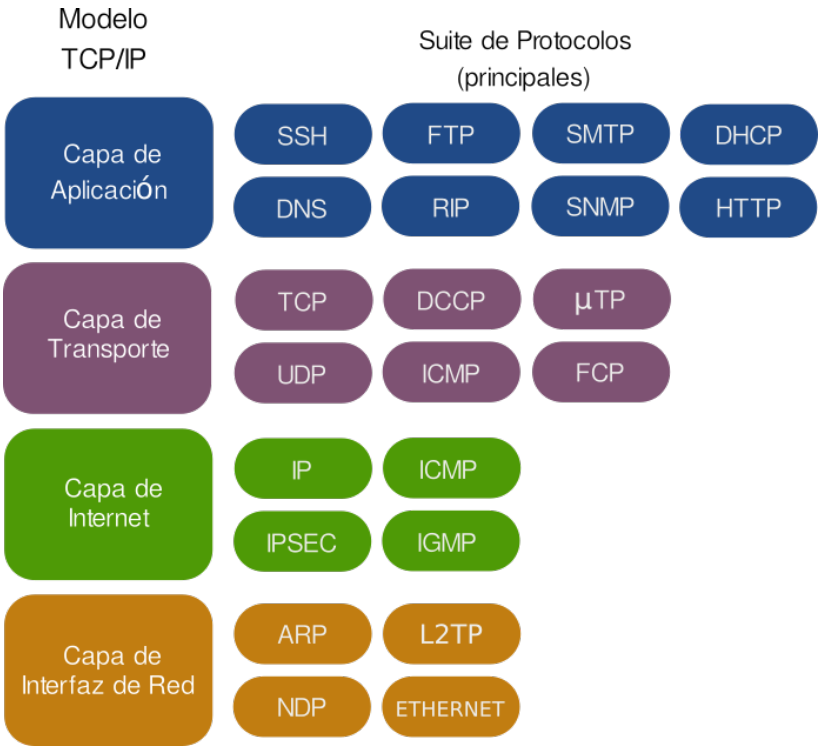

Figura 3.3: Familia de protocolos

4. Técnicas y herramientas

4.1. Metodologías

Scrum

La metodología ágil utilizada ha sido Scrum[**?**], mediante la cual se ha organizado el trabajo en interaciones cortas llamadas sprints. En cada uno de estos se establecen una serie de tareas, ordenadas por prioridad.

Este enfoque permite una mayor flexibilidad, ya que no se necesita tener todo definido desde el inicio del proyecto, y si surgen imprevistos, como cambios de requisitos, nuevas ideas, o cualquier tipo de problema, se introducen nuevas tareas en los siguientes sprints.

Este marco de trabajo encaja muy bien en equipos pequeños que trabajan de manera autónoma, favoreciendo una autogestión eficiente y una mejora continua del proyecto.

Test Driven Development (TDD)

En la misma línea de metodologías ágiles, se ha optado por seguir la práctica de TDD[**?**], en la cual se escriben los tests antes de implementar el código que se necesita probar.

TDD consta de tres fases repetitivas:

1. **Fase roja:** se escriben los test destinados a comprobar una determinada funcionalidad. Inicialmente, estos tests fallan.
2. **Fase verde:** se implementa el mínimo código necesario para que los tests anteriores pasen.

3. **Refactorización:** se optimiza el código a la vez que se asegura que todos los test siguen siendo correctos.

4.2. Gestión del proyecto

Control de versiones

Para la gestión de versiones del proyecto se ha escogido Git[**?**], una herramienta de control de versiones distribuido.

Permite llevar un seguimiento de todos los cambios realizados en el código, además de facilitar la creación de ramas independientes que pueden usarse para implementar nuevas funcionalidades o corregir errores.

Existen otras herramientas de gestión como Subversion (SVN)[**?**], la cual se había considerado también, aunque finalmente fue descartada tras un análisis comparativo con Git:

- **Modelo:** Git es un sistema distribuido, mientras que Subversion (SVN) es centralizado.
- **Repositorio local:** Git almacena todo el historial del proyecto localmente; SVN solo guarda una copia de trabajo.
- **Velocidad:** Git realiza operaciones locales mucho más rápido que SVN, que depende del servidor.
- **Gestión de ramas:** Git facilita la creación y fusión de ramas de forma ligera; en SVN, este proceso es más pesado y complejo.
- **Integración con plataformas:** Git se integra de forma nativa con plataformas como GitHub o GitLab, mientras que SVN tiene una integración más limitada.
- **Soporte y comunidad:** Git cuenta con una comunidad más amplia, con mayor soporte y documentación actualizada que SVN.

Repositorio

Las opciones contempladas fueron GitHub o Gitlab, debido a su integración nativa con Git. Finalmente se ha optado por utilizar GitHub[**?**], ya que se trata de una plataforma muy popular en proyectos open source porque

es muy intuitiva y está enfocada a desarrolladores individuales y equipos colaborativos.

Por su parte, Gitlab está más enfocada a entornos de empresas, es más completo pero más complejo de utilizar.

Además, GitHub incluye una serie de herramientas de integración CI / CD llamda GitHub Actions que se utilizarán en conjunto con otras herramientas que describiremos en otras secciones.

Plataforma de gestión

Existen infinidad de plataformas enfocadas a la gestión de proyectos; Trello, Jira, Asana, Zube, ZenHub ...

Hemos optado por utilizar Zube[**?**], ya que es una herramienta que se integra fácilmente con GitHub, permite crear sprints, épicas y tableros Kanban.

Si bien es cierto que la herramienta por excelencia en entornos empresariales es Jira, considero que tiene demasiadas funcionalidades que no van a ser aprovechadas en este proyecto, por lo que es más conveniente utilizar algo más ligero.

Comunicación

Para cualquier tipo de comunicación, se utilizarán email y Teams.

4.3. Lengujes de programación

Cliente

Para el desarrollo de la aplicación cliente, se debía escoger un lenguaje de scripts o cercano a estos, ya que el objetivo era construir una herramienta capaz de obtener configuraciones y aplicarlas a los dispositivos de forma sencilla.

Entre los candidatos estaban PowerShell, Java, y Python. Todos son lenguajes que permiten satisfacer los requisitos del proyecto, con ligeras diferencias:

- **PowerShell:** aunque es potente, limita su uso a dispositivos con sistema operativo Windows, restringiendo así la portabilidad.

- **Java:** gracias a su arquitectura multiplataforma, permite ejecutar programas en cualquier entorno. No obstante, generar un archivo ejecutable (.exe) que funcione universalmente resulta complejo, y en caso de distribuir un archivo .jar, se requeriría que los dispositivos clientes tuvieran Java instalado previamente.
- **Python:** en cambio, ofrece una solución más práctica: mediante sus librerías, es posible generar ejecutables de forma sencilla, compatibles con distintos sistemas operativos, y sin necesidad de instalar software adicional en los dispositivos cliente.

Entre estas tres alternativas, se decide utilizar Python[**?**] para desarrollar la aplicación de escritorio, ya que sus características se ajustan mejor al objetivo del proyecto.

API

En cuanto al desarrollo de la API, los dos candidatos por excelencia son Spring Boot y Node.js.

Node.js ha adquirido gran relevancia en los últimos años porque permite crear APIs de una manera muy sencilla, siendo especialmente útil en aplicaciones que precisan de mucha velocidad y eficiencia en tiempo real. Por otro lado, Spring Boot ofrece un amplio conjunto de módulos que permiten configurar la aplicación atendiendo a las necesidades de cada proyecto, a cambio de una mayor complejidad de desarrollo.

Se ha escogido Spring Boot[**?**]: aunque el proyecto inicial es relativamente pequeño, ofrece la posibilidad de escalar la aplicación y gestionar un mayor número de usuarios en un futuro. Además, integra módulos (beans, core, security...) que facilitan la implementación de mecanismos avanzados de desarrollo, como el control de accesos de usuarios.

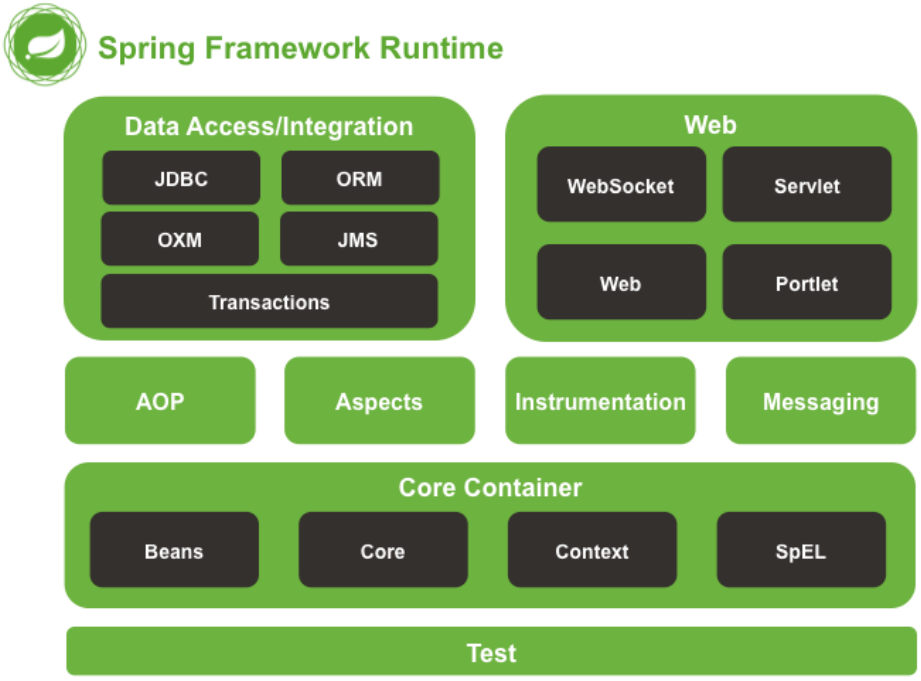

Figura 4.1: Spring Framework

Frontal

Finalmente, para el frontend se evaluaron dos alternativas: Angular o React. React proporciona gran flexibilidad y es ideal para proyectos en los que se quiere tener control total de la arquitectura y librerias usadas, aunque tiene la desventaja de que depende en gran medida de librerias de terceros para su desarrollo. Angular, en cambio, ofrece un framework completo con una gran cantidad de funcionalidades predefinidas, lo que facilita el mantenimiento y la optimización del código, especialmente en proyectos grandes.

Por esto, se ha optado por Angular[**?**], priorizando su robustez, escalabilidad y la gran cantidad de herramientas que proporciona.

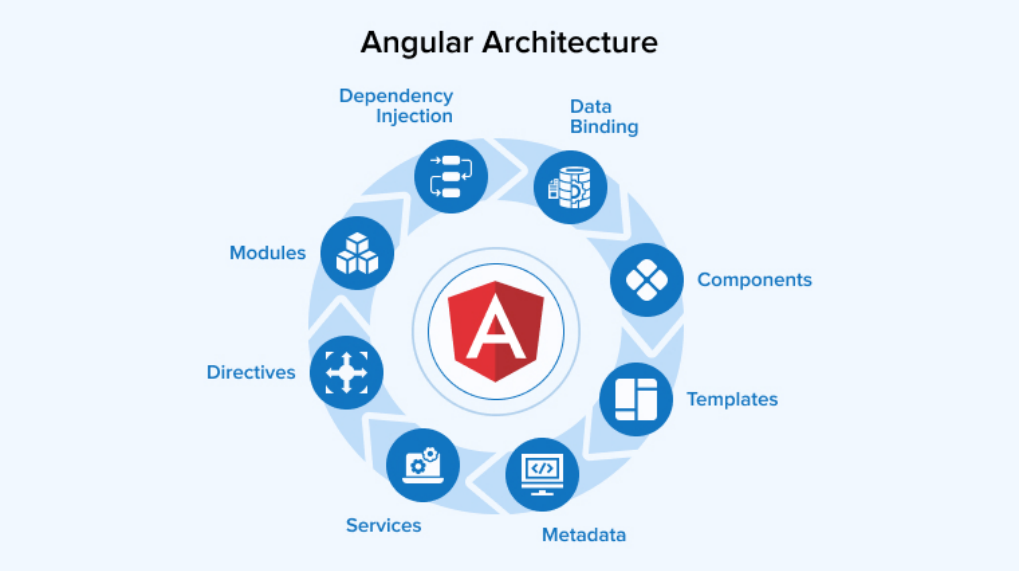

Figura 4.2: Angular Framework

4.4. Arquitectura

Patrón de diseños

En este proyecto se han aplicado distintos patrones de diseño adaptados a las diferentes necesidades.

La aplicación de escritorio del cliente, desarrollada en Python, emplea una estructura basada en la gestión dinámica de una única ventana, ocultando y mostrando distintos elementos. Esto es similar al concepto de SPA (Single Page Application) en el ámbito web.

La API REST desarrollada en Spring Boot, sigue una arquitectura MVC (Modelo-Vista-Controlador)[**?**].

El frontend en Angular adopta un patrón similar a MVC, donde los componentes representan la vista, y el controlador son los servicios que encapsulan el modelo de datos y la comunicación con la API.

4.5. Herramientas de desarrollo

Cliente

Para el desarrollo de la aplicación cliente se ha utilizado el Entorno de Desarrollo Integrado (IDE) PyCharm[**?**], ya que se encuentra plenamente enfocado a Python, ofrece multitud de herramientas y permite realizar configuraciones rápidas.

Otra opción era Visual Studio Code, pero para este lenguaje en concreto se necesita descargar multitud de extensiones para poder desarrollar cómodamente.

Para la instalación de librerias externas, se utiliza PIP[**?**], un gestor de paquetes que permite instalar, actualizar y gestionar librerías.

API

Para nuestra API con Spring Boot, se ha optado por el IDE IntelliJ IDEA[**?**] frente a Eclipse, por la sencilla razón de que IntelliJ es mucho más potente, intuitivo y dispone de todas las herramientas necesarias, sin tener que instalar nada.

En conjunto con este IDE, se ha añadido una herramienta de automatización de compilación moderna y potente, como Gradle[**?**], que permite compilar, construir, testear y desplegar aplicaciones de software.

En combinación con Spring Boot, destaca su uso frente al de Maven, ya que es más ligero, flexible y permite tener un control de los despliegues.

Frontend

El frontend se ha desarrollado utilizando Visual Studio Code[**?**]; también es posible utilizar IntelliJ y tener ambos proyectos (API y Frontend) en el mismo IDE, sin embargo, no está tan optimizado para la parte front y la sensación de programar el frontal dista mucho de ser óptima.

Además, utilizamos NPM (Node Package Manager)[**?**] como gestor de paquetes para instalar librerías externas de una manera simple y segura.

4.6. Integración contínua

Integración y despliegue continuos (IC/DC)

Para automatizar compilaciones, pruebas y despliegue, se ha utilizado GitHub Actions[**?**]; al escoger GitHub como repositorio, viene nativamente integrado.

Cobertura y calidad de código

SonarQube Cloud[**?**] se ha seleccionado como plataforma para analizar la calidad y cobertura del código fuente. La razón de su uso se debe a que no es necesario instalar ningún servidor, basta con conectarlo al repositorio de GitHub para auditar el código.

Despliegue

Docker[**?**] es una plataforma que permite empaquetar una aplicación y todas sus dependencias en un contenedor portable y ejecutarse en cualquier entorno, siempre y cuando Docker esté instalado en él.

4.7. Documentación

La documentación se ha desarrollado utilizando LaTeX[**?**] en lugar de otras herramientas de procesamiento de texto, como Word. Word funciona muy bien para documentos simples y, además, es fácil e intuitivo desde el principio. En cambio, LaTeX supone un manejo más complicado y necesita un periodo de adaptación para aprender a usarlo, pero permite un control de todo el formato, genera documentos de una calidad tipográfica excelente y es ideal para documentos técnicos.

Como editor LaTeX se utiliza IntelliJ Idea junto con el plugin TeXiFy-IDEA y el compilador pdfLaTeX.

Visualizamos los documentos PDF generados con SumatraPDF por integrarse perfectamente con las herramientas mencionadas.

Diagramas

Para la creación de diagramas y esquemas, se han utilizado dos herramientas:

- **Draw.io:** es una herramienta en línea que permite crear diagramas de flujo, organigramas, diagramas UML... Es fácil de usar y permite exportar los diagramas en diferentes formatos.
- **PlantUML:** es una herramienta que permite crear diagramas a partir de texto, lo que facilita su integración en el código fuente y su mantenimiento.

4.8. Librerias

En este apartado mencionamos las librerias más importantes, sin contar aquellas que se encuentran preinstaladas en los frameworks utilizados.

Cliente

PyInstaller

PyInstaller[**?**] es una herramienta que permite convertir programas escritos en Python en ejecutables independientes para Windows, Linux y MacOS.

Inno Setup Compiler

Inno Setup[**?**] es una herramienta gratuita para crear instaladores para aplicaciones de Windows. Permite empaquetar archivos, configuraciones y recursos necesarios, generando un asistente de instalación (.exe) que facilita la distribución y el despliegue de programas.

API

Jjwt-api

Jjwt-api forma parte de la biblioteca JJWT (Java JWT)[**?**], una de las librerías más populares para trabajar con JSON Web Tokens (JWT) en aplicaciones Java.

JUnit

JUnit[**?**] es uno de los frameworks de testing más utilizados para aplicaciones Java. Permite crear y ejecutar de manera estructurada tests unitarios, ayudando a garantizar que cada componente del programa funcione correctamente de forma aislada.

Jacoco

Jacoco[**?**] es una herramienta de análisis de cobertura de código para aplicaciones Java.

Frontend

Angular CLI

Angular CLI (Command Line Interface)[**?**] es una herramienta oficial para crear, desarrollar, construir y mantener aplicaciones Angular desde la línea de comandos.

Tailwind CSS

Tailwind CSS[**?**] es un framework de CSS que permite diseñar interfaces de usuario rápidamente mediante la combinación de pequeñas clases directamente en el HTML.

DaisyUI

DaisyUI[**?**] es un complemento de componentes UI construido sobre Tailwind CSS. Ofrece componentes predefinidos como botones, modales, menús, tabs, alertas, etc., todos diseñados usando clases de Tailwind.

5. Aspectos relevantes del desarrollo del proyecto

En este punto se detallan los aspectos más relevantes en el desarrollo del proyecto, desde la primera reunión con el tutor para la elección del tema, hasta la entrega del mismo.

5.1. Elección del proyecto

En la primera reunión con el tutor, se enumeraron los posibles temas para el Trabajo de Fin de Grado.

El motivo de la elección de este tema viene motivado por experiencias de desarrollo pasadas, donde realicé una aplicación web para la gestión de las escuelas deportivas del centro donde cursé el CFGS de Desarrollo de Aplicaciones Web (DAW).

5.2. Métodos de trabajo

A pesar de no contar con un equipo de trabajo, se decidió seguir una metodología ágil, concretamente Scrum; al no contar, evidentemente, con un equipo de trabajo, no se ha aplicado de una manera estricta, pero se han seguido los principios básicos de la misma:

- **Sprint**: se han realizado sprints de dos semanas, donde se planifica el trabajo a realizar según un orden de prioridades.

- **Reuniones semanales**: cada semana, los lunes concretamente, se mantuvieron reuniones con el tutor para revisar el avance del proyecto (a modo de *demo*) y resolver dudas.
- **Backlog**: en esta lista se han ido añadiendo, durante el curso del desarrollo del proyecto, todo tipo de tareas a realizar (de análisis, documentación, diseño, desarrollo, pruebas...).

En cuanto al desarrollo con Scrum, no se define un conjunto de fases específicas, sino que se trabaja de una forma iterativa e incremental: en cada sprint se van añadiendo nuevas funcionalidades y se van corrigiendo errores.

5.3. Fase de análisis

Durante este periodo, se dedicaron varias sesiones a la recopilación de información sobre el tema a desarrollar, las distintas necesidades de los usuarios y la manera de satisfacerlas.

La idea principal era crear una herramienta que permitiese gestionar, de forma centralizada, el acceso a la red de los dispositivos que dispone la Universidad. Para ello, se estudió la forma de gestionar este acceso desde la consola de comandos y se descubrieron herramientas que permitían esta gestión, como el firewall de Windows y Netsh.

Uno de los principales problemas que surgieron fue la manera de identificar a cada dispositivo de forma única. La primera idea fue utilizar el GUID del dispositivo, pero se descubrió que los dispositivos del centro son clonados, y, por lo tanto, comparten el mismo, por lo que se tuvo que descartar irremediablemente esta vía.

Ante esta dificultad, se planteó la posibilidad de combinar este identificador GUID con el nombre del dispositivo para crear una huella única.

Estos dispositivos se guardarían en una base de datos junto con las configuraciones que se pretendan aplicar. Un detalle importante que se especificó en esta fase, es que contaríamos con grupos o localizaciones de dispositivos que formarían una jerarquía y, así, se podrían aplicar configuraciones de carácter general a todos los dispositivos que pertenecen a cada grupo.

5.4. Fase de diseño

Barajamos la posibilidad de utilizar un script que se mantuviese en ejecución en segundo plano, pero se descartó por ser poco eficiente y seguro. En su lugar, decidimos utilizar un ejecutable y mantenerlo activo de forma continua, mediante el programador de tareas de Windows; de esta manera, se restringe la posibilidad de que un usuario pueda modificar el script y ejecutar comandos que puedan comprometer la seguridad del sistema. Las tecnologías propuestas para el ejecutable fueron Java y Python, pero la creación de un ejecutable (.exe) en Java causó muchos problemas y no se pudo realizar. Con un archivo JAR sí se pudo crear, pero generaba la necesidad de tener instalado Java en cada dispositivo. Por eso se optó por utilizar Python, ya que permite crear un ejecutable independiente del sistema operativo.

En cuanto a la base de datos (BD) se decidió utilizar un modelo relacional, porque los datos a almacenar son de carácter estructurado y van a tener relación entre ellos.

Para la API se decidió utilizar un modelo REST, ya que es el más utilizado en la actualidad y permite una fácil interacción entre el cliente y el servidor. La herramienta elegida fue Spring Boot y Java, debido a que está enfocada a la creación de servicios REST y cuenta con una gran cantidad de herramientas y librerías ya instaladas.

Por último, se decidió crear una interfaz web para realizar toda la gestión de manera centralizada: por compatibilidad y por ser uno de los frameworks actuales más utilizados, se decidió utilizar Angular.

Con toda esta estructura, aseguramos que la aplicación sea escalable y fácil de mantener, e incluso si se quisiese crear una aplicación para smartphone en un futuro, se podría reutilizar la parte backend, agilizando el proceso de desarrollo de la app.

5.5. Fase de desarrollo

La construcción del código comienza con el ejecutable en Python: una primera versión con una ventana y un formulario para iniciar sesión, y, tras la autenticación, un botón para registrar el dispositivo en la base de datos.

El principal desafío consistió en crear un instalador. Para ello, se utilizó Inno Setup combinado con el ejecutable (.exe) generado mediante la librería PyInstaller; esto permitió no solo instalar la aplicación en el dispositivo, sino

también mantenerla en ejecución en segundo plano, mostrando un icono en la barra de tareas.

El siguiente paso fue la creación de la API con la base de datos, que se realizó de forma paralela al ejecutable. La configuración inicial de la API con Spring Boot fue lo más tedioso, porque fue necesario instalar todas aquellas dependencias que se van a utilizar, además de establecer la estructura de carpetas y archivos. La forma de autenticación elegida fue mediante JWT, una de las más utilizadas en la actualidad, lo que hace posible una fácil interacción entre el cliente y el servidor. Se implementaron, de manera gradual, los distintos casos de uso de la API, generando diferentes endpoints para cada uno de estos. Añadimos SonarCloud para el análisis de la calidad del código con GitHub Actions; sin embargo, inicialmente, se desactivaron los tests porque la BD no es en memoria ni se contaba con Docker, además tuvimos que utilizar una versión de Java inferior a la 21, ya que SonarCloud soporta hasta la 17.

Para el desarrollo de la interfaz web, fue necesario descargar Node.js y Angular CLI con tal de generar el proyecto y sus dependencias. A partir de este paso, cualquier librería externa se añade mediante el gestor de paquetes NPM. Utilizamos Tailwind CSS + DaisyUI para el diseño de la interfaz, ya que incluye estilos predefinidos para maquetar la aplicación.

La interfaz web se desarrolla de forma paralela a la API: primero se escriben los tests de la API, y a continuación, se escribe el código necesario para que pasen, aplicando el desarrollo dirigido por pruebas (TDD); por último, se implementa la interfaz web. Una de las funcionalidades más difíciles de implementar fue la estructura arbórea de los grupos situada en el menú izquierdo de la interfaz; tenía que cargar de forma dinámica el listado de dispositivos que pertenecen a cada uno en la parte derecha, además de mantener la concordancia con los datos. Esto es clave, porque facilita al usuario la gestión de los dispositivos de una manera intuitiva y rápida.

La gestión de plantillas y comandos se implementa de una manera en la que se crean los comandos “fijos“, y estos son añadidos a las distintas plantillas. No obstante, se descubrió que esta lógica no era muy eficaz. Si queríamos, por ejemplo, añadir un valor a un comando de una dirección IP, teníamos que crear un comando para cada dirección, lo que resultaría tremendamente tedioso y se perdería la agilidad que pretendemos conseguir de forma general con GCA. Para solventar este problema, se estableció una estructura básica para añadir valores tal que el valor de un comando que contenga `{{variable}}` sirve para cambiar el valor de esa variable al asignarse a una plantilla.

Cuando terminamos con la construcción de la API (o, al menos, el desarrollo de aquellas estructuras que se consideraron necesarias para su funcionamiento), se decidió mejorar la aplicación Python para el cliente. Inicialmente, había demasiada lógica en cada archivo .py, por lo que creamos clases más pequeñas con una lógica bien definida. El funcionamiento de mostrar / ocultar botones limita el desarrollo y añade lógica innecesaria, por lo que se opta por crear frames o páginas y añadir en estas los elementos necesarios. En nuestro caso, creamos una página para iniciar sesión y otra para mostrar el panel de usuario una vez iniciada. Otro punto a destacar aquí es que, cuando intentamos realizar el registro de un equipo y aplicar la configuración, no se sabía qué estaba ocurriendo, más allá de alguna alerta programada. Introdujimos un recuadro para ir mostrando todos los logs de la aplicación mediante diferentes colores: los de información en azul, advertencias en naranja y errores en rojo.

Tras conseguir una primera aplicación de la configuración asignada a un dispositivo, comenzaron los problemas:

- Al deshabilitar la red para el dispositivo, se perdía la conectividad a la API.
- Tras aplicar la configuración de plantilla, quitar la asignación y añadir otra diferente, no se aplicaba correctamente.
- Al bloquear toda conexión saliente de la interfaz, no podíamos conectar de nuevo a la API.

Tuvimos que trabajar en el algoritmo para aplicar la configuración; así, el funcionamiento queda definido de la siguiente manera:

1. Se restaura la copia de seguridad.
2. Se eliminan los registros de plantillas anteriores.
3. A partir de aquí, se comprueba de manera periódica la existencia de configuración a aplicar.
4. Se obtiene el valor de la huella digital almacenada en el dispositivo.
5. Si existe una huella digital, se solicita al servidor la configuración a aplicar.
6. Si se reciben datos de configuración:

- Se comprueba si la configuración obtenida es más actual que la que tiene el equipo.
- Si hay cambios, se restaura la copia de seguridad y se aplican los comandos.
- Se actualizan los valores en el registro del dispositivo.

7. Si no hay configuración o está vacía, se revisa si se ha eliminado la asignación del equipo y restaura.
8. Se espera el tiempo definido antes de reiniciar el ciclo.

La copia de seguridad se realiza una única vez en el momento de registrar el equipo, de forma que nos aseguramos de que el dispositivo pueda volver a su estado original ante cualquier imprevisto. Esta copia incluye la configuración del firewall y la configuración de red. En el proceso de restauración también se habilitan de nuevo las interfaces de red y se configuran las reglas de firewall para mantener la conexión a las URL's configuradas en el json como permitidas.

5.6. Fase de pruebas

Las pruebas se realizan de manera continua a lo largo de todo el ciclo del desarrollo: para el backend utilizamos las librerías JUnit para pruebas unitarias, y Spring Boot Test para pruebas de integración.

En primer lugar, escogimos la misma BD con la que inicia la API para ejecutar los tests, pero esto generaba conflictos con los datos, por lo que se decidió utilizar una BD en memoria, H2, para realizar las pruebas. Para la configuración de la BD en memoria, creamos un nuevo archivo de application.properties dentro del directorio de test/resources. Este archivo contiene la configuración de la BD en memoria, y se carga automáticamente al ejecutar los tests. Aquí surgió un problema: los scripts se ejecutaban antes de que Hibernate crease la BD dando lugar a errores de inserción. Para que funcionase correctamente, según lo propuesto en StackOverflow[**?**], fue necesario añadir la siguiente propiedad spring.jpa.defer-datasource-initialization=true para invertir el orden, y que primero se generase la estructura y después se insertase los datos.

Una vez configurada la BD en memoria, se vuelven a activar los tests en SonarQube Cloud, pero se producen una serie de errores relacionados con el contexto de Spring. Aquellos test que hacen uso de la anotación

@SpringBootTest levantan el contexto, permiten inyectar beans y utilizar todos los componentes, y como resultado, se ejecutaban de nuevo los scripts de la base de datos. Existía una confusión entre las tablas generadas por JPA y los scripts que se ejecutaban para la BD en memoria, por lo que se ajustó poniendo todos los nombres de tablas en minúsculas, y para los tests de spring, estableciendo el perfil activo a la hora de ejecutar los tests de "tests". De esta manera, solamente se ejecutan aquellas propiedades dentro de la carpeta test/.

Mediante este enfoque conseguimos un entorno de pruebas aislado, reproducible y compatible con SonarQube.

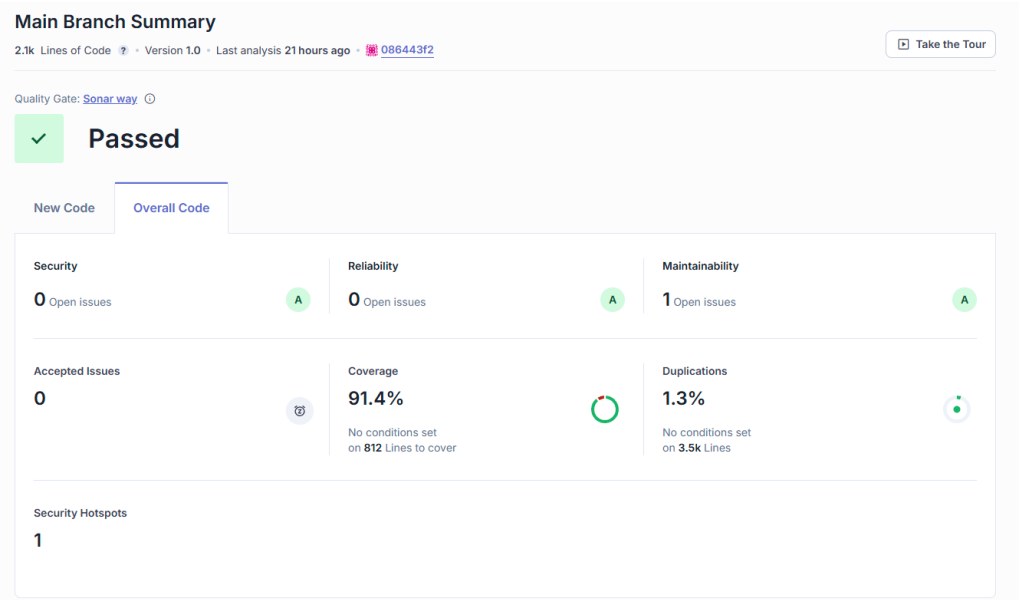

Figura 5.1: Resultado de SonarQube en el proyecto

Las pruebas de la aplicación cliente se hicieron de manera manual, tratando de cubrir todos aquellos casos que se puedan dar, como por ejemplo:

- Asignar una plantilla de configuración vacía.
- Asignar una plantilla a un dispositivo y al grupo directo al que pertenece y ver qué plantilla se obtiene.
- Eliminar la plantilla asignada y ver qué sucede con la configuración.

- Reiniciar / Apagar el dispositivo y comprobar que se aplique la configuración correcta.

De esta manera, aseguramos que la funcionalidad más importante esté cubierta.

Por último, para la aplicación web, al alojar un número reducido de funcionalidades, simplemente se comprueba que toda la gestión de dispositivos, grupos, comandos y plantillas sea correcta, es decir, que se pueda añadir, eliminar, modificar y listar.

6. Trabajos relacionados

La gestión centralizada de dispositivos es una práctica habitual en la administración de sistemas y, casi desde sus inicios, se han desarrollado herramientas para la gestión de dispositivos de red. Esta gestión ha evolucionado (y sigue evolucionando) hacia la automatización de tareas, la orquestación de servicios y la gestión de configuraciones.

En este apartado se exponen aquellos artículos científicos y herramientas existentes que guardan similitud con el presente trabajo.

6.1. Articulos cientificos

An Application to Automate the Configuration Management and Assessment of Practically Based Learning Outcomes in Computer Networking [?]

Este estudio describe una aplicación desarrollada para automatizar la configuración y evaluación de escenarios prácticos en laboratorios de redes. La herramienta permite aplicar configuraciones a dispositivos reales y simulados, facilitando la evaluación de resultados de aprendizaje prácticos. Aunque se centra en el ámbito educativo, su enfoque en la automatización y gestión de configuraciones es aplicable a entornos universitarios más amplios.

Configuration Management and Security [?]

Este artículo analiza los desafíos de la gestión de configuración en sistemas de gran escala, destacando la importancia de políticas centralizadas para garantizar la seguridad y el cumplimiento normativo. Aunque no se enfoca específicamente en la educación, se describe la necesidad de herramientas que permitan la gestión de configuraciones de manera eficiente y segura.

6.2. Herramientas

System Center Configuration Manager (SCCM)

SCCM[**?**] es una solución de Microsoft para la gestión centralizada de configuraciones y actualizaciones. Permite aplicar configuraciones a grupos de dispositivos, gestionar actualizaciones de software ... Aunque es una herramienta comercial, su enfoque en la gestión de configuraciones es similar al propuesto en este trabajo. Para su uso es necesario tener instalado Microsoft AD para permitir gestionar dispositivos y usuarios dentro de la red.

Active Directory (AD) o Entra ID

AD[**?**] y Entra ID[**?**] (anteriormente conocido como Azure AD) son soluciones de Microsoft para la gestión centralizada mediante servicios que controlan el acceso a recursos y la configuración de dispositivos en entornos empresariales. La diferencia principal entre ambos es que Entra ID está diseñado para entornos en la nube, mientras que AD se utiliza principalmente en entornos locales.

Netsh

Netsh[**?**] es una herramienta de comandos de Microsoft que permite la configuración y gestión de dispositivos de red en sistemas operativos Windows, ya sea de forma local o remota.

Características	**GCA**	**SCCM**	**AD**	**Netsh**
Automatización de configuraciones	Sí	Sí	Sí	No
Gestión centralizada	Sí	Sí	Sí	No
Priorización por dispositivos	Si	Limitada	Sí	No
Interfaz de usuario	Sí	Sí	Sí	No (terminal)
Curva de aprendizaje	Baja	Alta	Alta	Media
Dependencia con otras herramientas	No	Sí	Sí	No
Multiplataforma	Sí*	Limitada	Limitada	No

Tabla 6.1: Comparativa de las herramientas de los proyectos con codificación de colores.

Esta comparativa revela que, aunque soluciones como SCCM y Active Directory ofrecen capacidades avanzadas de gestión de configuraciones y escalabilidad, no resultan especialmente rentables para el fin que pretendemos porque requieren de hardware específico, son económicamente costosas y

están diseñadas para entornos empresariales. Por el contrario, la propuesta de GCA se centra en la simplicidad y accesibilidad, permitiendo su uso en entornos educativos y de investigación sin la necesidad de infraestructura costosa y compleja, pudiendo ser implementada en cualquier tipo entorno, si así se desea.

No obstante, la elección de la herramienta adecuada dependerá de las necesidades específicas de cada entorno y de los recursos disponibles.

6.3. Fortalezas y debilidades del proyecto

Fortalezas	**Debilidades**
Automatización de tareas repetitivas	Dependencia de la conectividad
Gestión centralizada y escalable	Complejidad en la implementación
Priorización de configuraciones	Riesgo de obsolescencia tecnológica
Integración con herramientas existentes	Requiere primera instalación manual
Mejora en la seguridad	Posibles problemas de compatibilidad
Compatible con cualquier sistema operativo	Necesidad de mantenimiento continuo

Tabla 6.2: Fortalezas y Debilidades del Proyecto

7. Conclusiones y Líneas de trabajo futuras

Una vez finalizado y revisado el contenido del proyecto, se pueden extraer diversas conclusiones.

En lo que al ámbito personal respecta, me ha resultado útil para aprender a gestionar mejor la carga de trabajo; he sido consciente de que no es eficiente tratar de tener todo hecho de la manera más rápida posible, sino que, si nos paramos a realizar un análisis previo de todo aquello que se necesita hacer y lo organizamos en prioridades, podremos dedicar tiempo a aquellas cosas que sean más necesarias y nos permitan abrirnos paso a completar otras tareas que, a priori, antes de este análisis parecían imposibles. El trabajo diario y la estructuración previa de tareas resulta mucho más eficiente que dedicar un sinfín de horas al proyecto sin un orden previamente establecido, ya que la falta de organización puede conllevar a un uso improductivo del tiempo dedicado. Al tener bien distribuidas las tareas, cuando me encontraba atascado con alguna de ellas, traté de continuar con otra y volver a la primera en otro momento; así conseguía "resetear" y regresar con un punto de vista diferente que, generalmente, me permitía resolver el problema.

El resultado del proyecto, si bien no cuenta con un número extenso de funcionalidades, incluye todos los elementos necesarios para realizar una gestión de los dispositivos en las aulas de una manera sencilla, pero eficaz. La interfaz web se caracteriza por su facilidad de uso; es intuitiva, no precisa de un manual extenso para su utilización porque los textos, iconos e imágenes permiten diferenciar las distintas funcionalidades. La aplicación cliente, siguiendo la misma línea, implementa lo básico para posibilitar la gestión de los dispositivos y mostrar en todo momento qué está sucediendo. La API

es más compleja, ya que requiere conocimientos en Java y Spring Boot para poder entender toda su funcionalidad.

Teniendo en cuenta todo esto, este proyecto es una primera versión de lo que puede llegar a ser. Soy consciente de que se trata de un proyecto con una gran capacidad de mejora: al contar con permisos de administrador en la ejecución de la aplicación, abre un abanico de infinitas posibilidades a nuevas funcionalidades. Además, la propia API ha sido construida para ser escalable y poder ejecutarse en cualquier tipo de plataforma y de sistema operativo.

Dentro de la columna de "Backlog" están aquellas tareas que, debido al tiempo disponible para la realización del TFG, no han podido ser completadas, las cuales son ligeras mejoras del proyecto de cara al futuro.

Por último, considero importante plantear una serie de ideas que pueden ser realizadas en versiones más avanzadas de GCA:

- La creación de una aplicación cliente para sistemas operativos Linux y MacOS.
- Almacenar en la nube las copias de seguridad realizadas en los dispositivos y llevar un histórico.
- Implementar un panel *Administrador* para realizar la gestión de otros usuarios.
- Generar informes para estudiar qué comandos se utilizan más y en qué orden.
- Actualizar el "Look and Feel" de la aplicación cliente para Windows.
- Utilizar SignTool en Windows para firmar la app cliente y que sea categorizada como segura.
- Desarrollar un diseñador visual para implementar workflows con plantillas y comandos.
- Machine Learning para analizar tendencias de tráfico, detectar patrones inusuales y recomendar ajustes.