Universidad Politécnica
de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Implementación de un modelo Zero Trust en un entorno Cloud Híbrido Simulado

Autor: Mª de las Mercedes Jiménez Díaz-Varela

Tutor: Antonio Jesús Díaz Honrubia

Madrid, julio 2025

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería Informática
Título: Implementación de un modelo Zero Trust en un entorno Cloud Híbrido Simulado
Julio 2025

Autor: Mª de las Mercedes Jiménez Díaz-Varela

Tutor:
Antonio Jesús Díaz Honrubia
Lenguajes y Sistemas Informáticos e Ingeniería de Software
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

En un mundo tecnológico que se mueve hacia configuraciones híbridas y distribuidas, las empresas se enfrentan al reto de asegurar recursos que están fuera de su red tradicional. El auge del trabajo a distancia, el uso de servicios en la nube y la variedad de dispositivos han hecho que los sistemas de seguridad antiguos ya no sean útiles. Ante esto, el modelo Zero Trust ha surgido como una solución completa que asume que ningún acceso es confiable desde el principio, aplicando la idea de "nunca confiar, siempre verificar" como la base de cualquier plan de seguridad digital. Este modelo, impulsado por organizaciones como el NIST, se basa en la verificación continua, el acceso mínimo necesario, la división estricta y la visión completa del sistema.

Este proyecto busca diseñar, implementar y evaluar una estructura de red híbrida bajo el modelo Zero Trust, simulada en un ambiente de laboratorio cerrado. La implementación se realizó en un nodo Proxmox VE dentro de VMware, configurando varias máquinas virtuales y contenedores LXC que representan tanto servicios locales como servicios simulados en la nube. La estructura lógica incluye control de identidad, verificación unificada, división de la red, control del tráfico y reglas de acceso estrictas.

Entre los servicios implementados, destacan:

- Un servidor OpenLDAP para la administración central de identidades, organizado por departamentos (marketing, IT, recursos humanos, ventas, finanzas) y con usuarios definidos dentro del dominio corporativo zts.org.
- Un sistema de verificación unificada a través de Keycloak, configurado como proveedor OIDC, asignando atributos LDAP como employeeType para aplicar el control de acceso según el rol.
- Una aplicación web desarrollada en Python (Flask) con una interfaz en CSS y Bootstrap, que muestra contenido y funciones diferentes según el rol del usuario (empleado o administrador), incluyendo mensajería privada y canales para cada departamento.
- Una base de datos PostgreSQL que maneja tanto el soporte de la aplicación como el almacenamiento de usuarios internos y sesiones verificadas desde Keycloak.
- Un proxy HTTP Squid que filtra y limita el tráfico de salida, permitiendo solo el acceso a dominios corporativos y servicios autorizados.
- Un firewall pfSense, configurado con reglas de acceso mínimas que simulan una separación lógica entre LAN y WAN dentro de la misma red virtualizada, siguiendo los principios de Zero Trust.

El ambiente se diseñó con el objetivo de reducir la exposición, usando una sola interfaz de red puenteada (vmbr0) para todos los servicios, y controlando la comunicación entre nodos solo a través de reglas específicas. Se decidió no activar TLS en el servidor LDAP debido a que el ambiente es cerrado y controlado, dejando esta decisión documentada como una limitación aceptable para el laboratorio, y señalando los ajustes necesarios para una implementación real.

Durante las pruebas, se confirmó que el proceso de verificación y autorización funciona correctamente: el usuario se verifica en Keycloak, los atributos LDAP se asignan bien, y la aplicación web cambia su funcionamiento según el rol. Además, se comprobó la división del tráfico a través del firewall y el bloqueo efectivo de la navegación no autorizada a través del proxy.

Aunque funciones como la vigilancia en tiempo real (Wazuh, Suricata) o el uso de VPN (WireGuard) no se implementaron por completo, se documentaron como sugerencias para mejorar el ambiente en el futuro, asegurando que se pueda ampliar y adaptar a situaciones reales más complejas.

En conclusión, este proyecto prueba que es posible implementar un modelo Zero Trust en ambientes simulados, usando solo herramientas de código abierto y recursos accesibles. El resultado es una estructura que funciona, está dividida y es segura, que muestra los principios básicos del modelo y que se puede usar como base para aprender o como ambiente de prueba para futuras investigaciones.

Abstract

In a technological landscape increasingly moving toward hybrid and distributed configurations, organizations face the challenge of securing resources that lie beyond the traditional corporate network perimeter. The rise of remote work, the adoption of cloud services, and the diversity of devices have rendered legacy security models obsolete. In response to this shift, the Zero Trust model has emerged as a comprehensive solution that assumes no access is inherently trustworthy, applying the principle of "never trust, always verify" as the foundation of any modern cybersecurity strategy. This model, advocated by institutions such as the NIST [1], is based on continuous verification, least-privilege access, strict segmentation, and full visibility into system activity.

This project aims to design, implement, and evaluate a hybrid network architecture following the Zero Trust model, simulated within a closed laboratory environment. The implementation was carried out using a Proxmox VE node hosted on VMware, deploying multiple virtual machines and LXC containers that simulate both on-premise and cloud-like services. The logical architecture includes identity control, federated authentication, network segmentation, traffic control, and strictly enforced access policies.

The deployed services include:

- An OpenLDAP server for centralized identity management, structured by departments (marketing, IT, human resources, sales, and finance), with users defined under the corporate domain zts.org.
- A federated authentication system via Keycloak, configured as an OIDC provider, mapping LDAP attributes such as employeeType to enforce role-based access control.
- A Python Flask web application with a CSS/Bootstrap-based frontend, delivering role-specific content and functionalities, including private messaging and departmental communication channels.
- A PostgreSQL database supporting both the application backend and the storage of internal users and authentication sessions provided by Keycloak.
- An HTTP proxy (Squid) that filters and restricts outbound traffic, allowing access only to approved corporate domains and essential internal services.
- A pfSense firewall, configured with minimal access rules to simulate logical LAN/WAN separation within the same virtualized network, in alignment with Zero Trust principles.

The environment was designed to minimize exposure, using a single bridged network interface (vmbr0) for all services, while inter-node communication was strictly governed by explicitly defined rules. The decision was made not to enable TLS in the LDAP server, as the environment is entirely closed and controlled. This limitation is acknowledged and documented, with guidance provided for adapting the setup to a production scenario.

During testing, the authentication and authorization process was validated successfully: users authenticated through Keycloak, LDAP attributes were

correctly mapped, and the web application dynamically adjusted its behavior based on user roles. Network segmentation through the firewall and outbound traffic control via the proxy were also verified effectively.

While features such as real-time monitoring (Wazuh, Suricata) and VPN connectivity (WireGuard) were not fully implemented, they have been documented as future improvements to enhance scalability and applicability in more complex, real-world deployments.

In conclusion, this project demonstrates the feasibility of deploying a Zero Trust-based architecture in a simulated environment using only open-source tools and accessible resources. The result is a functioning, segmented, and secure infrastructure that illustrates the core principles of Zero Trust and serves as a valuable foundation for educational purposes or as a testbed for further research.

Tabla de contenidos

Índice de Figuras

1 Introducción

En este capítulo se introduce cuáles han sido las motivaciones de la realización del proyecto, su justificación, los objetivos que se han seguido, la estructura y alcance del mismo.

1.1 Contexto y motivación

Las arquitecturas híbridas, que entrelazan la infraestructura local con servicios en la nube, cada vez más son utilizadas por las empresas hoy en día. Por un lado, estas arquitecturas ofrecen flexibilidad, capacidad de ampliación y buen uso de los recursos, pero también se aumenta el riesgo de ataques. El movimiento de los empleados, el trabajo a distancia y el uso extenso de programas desde la nube (SaaS) han mostrado los problemas de las formas antiguas de seguridad, que confiaban en que todo dentro de la red de la empresa era seguro.

La transformación digital ha provocado que los datos y recursos importantes ya no tengan que estar dentro de un área segura. Además, las amenazas internas, los dispositivos que no se gestionan y los accesos a distancia hacen que no se pueda confiar en que alguien es legítimo solo porque está en la red de la empresa. Por eso, los sistemas de seguridad de ahora deben transformarse y usar ideas de nunca confiar, siempre verificar, para asegurar que siempre se revisen las identidades, los contextos y los permisos.

Ante esta necesidad, aparece el sistema Zero Trust, que el NIST define como un sistema donde no se confía en ningún usuario o dispositivo de forma automática, sin importar dónde esté [1]. Este sistema se basa en ideas importantes como la autenticación múltiple, el control de acceso detallado, la división de la red, la vigilancia constante y la limitación de los permisos. Usarlo ayuda a bajar el riesgo de que alguien se mueva sin permiso, acceda a donde no debe y cause problemas de seguridad [2].

Este proyecto propone una forma práctica y sencilla de implementar el modelo Zero Trust en un entorno híbrido simulado. Para ello, se ha creado un laboratorio virtual en Proxmox VE, que tiene servicios de identidades con OpenLDAP, autenticación unida con Keycloak, control del tráfico con Squid y división lógica con pfSense, entre otras cosas. Se ha completado con una aplicación web corporativa de mensajería que se ha diferenciado por roles y se ha conectado a una base de datos PostgreSQL, para simular diferentes clases de usuarios y casos de acceso controlado.

En este proyecto se ha mostrado de forma sencilla el comportamiento de las empresas que quieran incorporar el enfoque de Zero Trust en sistemas pequeños y con herramientas accesibles. En un tiempo donde la protección de la identidad y los datos es muy importante, esta idea ha ofrecido una base sólida para evaluar formas de dividir, autenticar y ampliar a sistemas más grandes. También, se ha mostrado los retos que aparecen al juntar varias tecnologías bajo una idea de seguridad unificada.

1.2 Justificación del modelo Zero Trust

El modelo de seguridad Zero Trust surge como una respuesta directa a las deficiencias de los sistemas tradicionales que confiaban en la confianza implícita dentro de las redes corporativas. Antes, la seguridad se basaba en dividir la red en una zona interna confiable y una externa no confiable. Este enfoque ya no funciona bien porque los usuarios son más móviles, acceden de forma remota, usan sus propios dispositivos (BYOD) y los servicios en la nube están distribuidos [3].

Zero Trust cambia esto al indicar que nadie debe ser considerado confiable por defecto, ya sea dentro o fuera de la red, y que cada acceso debe ser validado de forma clara. En lugar de depender de la ubicación en la red o de si se pertenece al dominio, la autorización se basa en la identidad del usuario, el contexto de la solicitud, el dispositivo que se usa y el nivel de riesgo involucrado [4].

Algunos de los principios principales que justifican el uso de del enfoque Zero Trust son:

- **Autenticación y autorización continuas:** no solo se debe validar la identidad del usuario al iniciar sesión, si no que para moverse y acceder a los diferentes recursos siempre debe validar su identidad.

- **Acceso mínimo necesario:** cada usuario o dispositivo solo debe tener los permisos necesarios para acceder a recursos. Se limita así el daño si hay una brecha.

- **Microsegmentación:** la red se divide en zonas lógicas donde se limita el acceso entre servicios, lo que reduce la posibilidad de que los atacantes se muevan libremente si logran entrar [5].

- **Visibilidad y monitoreo constante:** es importante registrar y analizar el tráfico, los accesos y los eventos de seguridad para encontrar problemas o intentos de acceso no autorizado en tiempo real.

El NIST formalizó estos principios en su Special Publication 800-207, donde da pautas para diseñar e implementar sistemas Zero Trust, sin depender de un proveedor o tecnología en particular [1]. Esta independencia ha hecho que Zero Trust se pueda aplicar tanto en grandes empresas como en entornos educativos, laboratorios y simulaciones.

La importancia de este modelo también se puede ver en su creciente uso por parte de organizaciones públicas y empresas privadas. De hecho, en 2021, el

gobierno de EE. UU. emitió la Executive Order 14028, que establece Zero Trust como el nuevo estándar de ciberseguridad federal [6].

Usar Zero Trust no solo mejora la resistencia ante ataques cibernéticos, sino que también facilita la gestión del acceso en entornos distribuidos. Su diseño modular permite una adopción gradual, integrando herramientas ya existentes como proxys, gestores de identidades, VPN o firewalls. Por lo tanto, su incorporación en este proyecto se debe tanto a razones técnicas como a la necesidad de alinear la seguridad con la situación actual de rápida digitalización y amenazas avanzadas persistentes (APT).

1.3 Objetivos del proyecto

El objetivo principal de este proyecto es crear, montar y probar un entorno de red virtual que imite una estructura empresarial híbrida, usando el modelo de seguridad Zero Trust como base. Se realiza como si se tratara de un laboratorio, usando herramientas accesibles y métodos de división, autenticación unificada y control del movimiento en la red para asegurar que se acceda a los recursos de manera segura y según el contexto.

Para lograr este objetivo general, se fijan los siguientes objetivos específicos, divididos en dos partes principales:

OE1: diseño y despliegue de la arquitectura del entorno Cloud híbrido con la implementación de un modelo Zero Trust.

> **OE1.1:** crear la estructura lógica del ambiente, agregando ideas de división de la red y reglas de acceso mínimo, para copiar una estructura empresarial dividida en áreas de confianza diferentes.
>
> **OE1.2:** arrancar los servicios clave necesarios para manejar identidades, autenticación y control del tráfico, como OpenLDAP, Keycloak, Squid, pfSense y, más adelante, herramientas de vigilancia.
>
> **OE1.3:** integrar contenedores que imiten servicios en la nube, como una aplicación web de la empresa y una base de datos, asegurando que se pueda acceder a ellos mediante reglas de seguridad que sigan el enfoque Zero Trust.

OE2: diseño e implementación de pruebas de seguridad y análisis del impacto del modelo.

> **OE2.1:** realizar pruebas prácticas para ver cómo actúan los sistemas de autenticación, control de acceso y división establecidos, revisando si son útiles en situaciones que se dan en el uso empresarial.

Este conjunto de objetivos permite elaborar una propuesta útil y educativa que muestra cómo se pueden poner en práctica las ideas del modelo Zero Trust en ambientes simulados, usando tecnologías accesibles y que se pueden ampliar. También sienta las bases para añadir en el futuro vigilancia activa, análisis del tráfico y respuesta automática a problemas.

1.4 Estructura y alcance de la memoria

El trabajo de Fin de Grado se divide en siete capítulos que permiten la comprensión del planteamiento, desarrollo y evaluación de una arquitectura de red híbrida basada en el modelo Zero Trust, desplegada en un entorno de laboratorio controlado. A continuación, se describe brevemente el contenido de cada capítulo:

- **Capítulo 2:** se examina las tecnologías consideradas y justifica las decisiones técnicas, explicando los criterios de selección usados.

- **Capítulo 3:** se especifica los requisitos de seguridad y funcionales del ambiente, incluyendo autenticación, control de acceso y segmentación de red.

- **Capítulo 4:** se describe el diseño lógico de la arquitectura y el proceso de implementación de los servicios, contenedores y máquinas virtuales.

- **Capítulo 5:** se presenta las pruebas llevadas a cabo para validar la implementación del modelo Zero Trust y examina los resultados conseguidos.

- **Capítulo 6:** se expone las conclusiones del proyecto, valorando los resultados, nombrando las dificultades encontradas y proponiendo mejoras.

- **Capítulo 7:** se expone el análisis de impacto del proyecto en la sociedad.

El alcance de este proyecto se limita a un ambiente de laboratorio ejecutado en un nodo Proxmox VE sobre VMware, sin conexión real a Internet. La red es aislada, y los servicios se comunican a través de una sola interfaz lógica (bridge). Esto simula un ambiente empresarial híbrido con recursos locales (LDAP, proxy, firewall, autenticación) y simulados en la nube (aplicación web y base de datos en contenedores LXC).

Esta configuración permite representar los principios del modelo Zero Trust propuestos por el NIST [1], aplicando segmentación, autenticación federada, control de tráfico y acceso mínimo. Debido a las limitaciones de tiempo y recursos de un proyecto académico, se priorizaron las funciones esenciales del modelo, dejando funciones avanzadas como la monitorización activa con Wazuh o Suricata para ampliaciones futuras.

Este trabajo es una propuesta didáctica que muestra la posibilidad de aplicar los principios Zero Trust usando herramientas abiertas, accesibles y replicables. La simulación de roles, flujos de acceso y servicios corporativos tiene un objetivo educativo, siguiendo las recomendaciones de autores como Ahmed [7] y Saini [8] en ambientes de virtualización y proxy seguro.

2 Tecnologías utilizadas

Una de las fases más importantes cuando se desarrolla cualquier sistema, es la es la selección de tecnologías que se van a utilizar. Por ello es importante buscar un modelo arquitectónico que contenga requerimientos específicos de seguridad. El modelo elegido en este desarrollo es el enfoque Zero Trust.

Esta fase define la evolución del proyecto pues está en función de la integración de los componentes y la viabilidad técnica del sistema que se va a simular.

Para asegurar estos objetivos, se ha realizado una evaluación técnica que identifica tecnologías compatibles con los sistemas de virtualización usados entre sí y que permiten una integración estable y segura.

2.1 Criterios de selección

La definición de los criterios de selección tecnológica ha permitido tomar decisiones mientras se diseñaban e implantaban cada componente. Estos criterios se centran las siguientes características:

- **Compatibilidad:** la totalidad de las herramientas que se van a utilizar deben poder ser instaladas y deben funcionar correctamente en máquinas virtuales y en contenedores gestionados por Proxmox VE, y ser implementadas desde una infraestructura anfitriona basada en VMware.

- **Alineación:** se ha dado prioridad a tecnologías que permitieran la segmentación de la red, la autenticación centralizada, el control de acceso basado en roles y la inspección del tráfico.

- **Disponibilidad:** se han elegido soluciones con fines académicos o de código abierto, asegurando así, la accesibilidad sin restricciones y la posibilidad de modificaciones.

- **Escalabilidad:** se han tenido en cuenta herramientas que, aunque no se hayan implementado por completo en esta fase, dieran la posibilidad de añadirse a corto o medio plazo para aumentar las funciones del sistema.

2.2 Análisis de tecnologías utilizadas

El desarrollo del sistema virtualizado orientado a la seguridad bajo el modelo Zero Trust, ha necesitado de una selección cuidados de diversas herramientas y plataformas. A continuación, se detalla el análisis de las tecnologías utilizadas:

- **Proxmox Ve y VMware:** para la capa de virtualización se ha elegido una combinación de VMware como base de ejecución y Proxmox Ve como

hipervisor principal de gestión. Proxmox Ve ofrecía una solución sólida, de código abierto y con una interfaz web fácil de usar para la administración de máquinas virtuales. (VMs) y contenedores LXC. Su compatibilidad con tecnologías como KVM y LXC ha permitido una implementación flexible de los servicios planificados. Esta elección se encuentra en línea con los principios de separación clara de roles, redes y niveles de privilegio dentro del sistema virtualizado [7].

- **OpenLDAP y phpLDAPadmin:** se ha seleccionado para gestionar las identidades OpenLDAPd como directorio principal, y como interfaz gráfica que la acompaña se ha seleccionado phpLDAPadmin, con objeto de hacer más sencilla la visualización y edición de objetos LDAP. La estructura jerárquica y la capacidad para definir permisos detallados que proporciona la LDAP, hacen que sea una opción adecuada para implementar políticas de acceso basado en el principio de mínimo privilegio y para delegar funciones dentro del sistema.

- **Keycloak:** se ha empleado Keycloak, como solución de autenticación centralizada porque destaca por ser un soporte nativo de protocolos como OpenID Connect (OIDC) y SAML 2.0. Además de unirse con LDAP, Keycloak ha permitido añadir autenticación multifactor, control de sesiones y asignación de roles personalizados, haciendo más asequible la aplicación de políticas de acceso adaptativas. La elección se ha basado en su integración con el backend de aplicación web y su capacidad para dar tokens seguros para sesiones autenticadas [2].

- **NGINX y Squid:** para el control del tráfico se han añadido dos herramientas diferentes: NGINX como proxy inverso, asegurando el acceso controlado a la aplicación web, y Squid como proxy directo para el filtrado de tráfico de salida. La configuración de Squid responde a uno de los pilares del modelo Zero Trust: la visibilidad completa sobre el tráfico de red y la restricción de accesos según políticas predefinidas [8].

- **PfSense:** para la segmentación de la red y el establecimiento de políticas firewell, se ha implementado pfSense, como dispositivo virtual. Esta herramienta permite definir reglas detalladas de filtrado y hacer tareas de NAT, siendo esencial en la separación de zonas dentro del sistema simulado.

- **WireGuard:** aunque la conexión VPN mediante WireGuard no se ha configurado por completo, su instalación ha quedado lista como función a desarrollar en fases futuras del sistema. Esta herramienta es conocida por su eficiencia, simplicidad de configuración y enfoque criptográfico moderno [9], estando en línea con los requerimientos de conectividad segura en sistemas distribuidos.

- **PostgreSQL y aplicación web en Python**: la aplicación web, desarrollada con el microframework Flask en Python y estilos CSS personalizados, se ha implementado como conector LXC. Su base de datos se ha gestionado mediante PostgreSQL, lo que ha permitido separar la lógica de autenticación (LDAP/Keycloak) del almacenamiento de mensajes y avisos internos. PostgreSQL también ha sido el sistema elegido para guardar la información de sesiones Keycloak, evitando así la pérdida de datos después de reiniciar el sistema.

2.3 Herramientas consideradas, pero no utilizadas

Durante el diseño del entorno simulado, se han valorado varias herramientas de seguridad que, aunque no se han utilizado en esta versión, son valiosas para un modelo Zero Trust. Estas herramientas son:

- **Wazuh:** es una plataforma de seguridad que une detección de intrusiones, análisis de logs y gestión de cumplimiento. Emplea un modelo agente-servidor para dar una vigilancia centralizada y adaptable de los sistemas protegidos. Su unión con ELK Stack (Elasticsearch, Logstash y Kibana) ayuda a ver y relacionar eventos de seguridad importantes [10].

- **Suricata:** es un motor de inspección profunda de paquetes (DPI) y sistema de detección y prevención de intrusiones (IDS/IPS), que analiza el tráfico en tiempo real. Es útil para encontrar comportamientos malos con reglas propias o de la comunidad y se puede añadir a arquitecturas que necesitan ver el tráfico de red de cerca [11].

- **ClamAV:** es una herramienta de código abierto para hallar software dañino. Su motor examina archivos en sistemas Linux y UNIX, utilizándose en servidores para proteger correos, webs y archivos. No es una protección completa en tiempo real, pero sirve como refuerzo en entornos cerrados o controlados [12].

- **OSSEC:** es un sistema de detección de intrusiones basado en el host (HIDS) para analizar logs, vigilar la integridad, encontrar rootkits y responder de forma activa. Su diseño ligero lo hace bueno para vigilar archivos importantes en sistemas operativos tipo UNIX [13].

Aunque estas herramientas son útiles, no se pusieron en marcha en esta fase por motivos de logística. Su configuración exigente, el consumo de recursos y el tiempo necesario para integrarlas bien, suponían una carga extra que no se podía asumir en los plazos del proyecto. Aun así, se ha guardado la idea de utilizarlas en el futuro para mejorar la capacidad de detectar amenazas y responder a posibles incidentes, acercando el laboratorio a los principios de Zero Trust.

3 Análisis de requisitos

Crear una arquitectura de red Zero Trust exige planificar bien las necesidades de seguridad y definir cómo debe funcionar. Zero Trust modifica la idea tradicional de confiar en todo dentro de la red. Ahora, todo acceso se debe justificar, vigilar y revisar, sin importar de dónde venga. Este capítulo revisa las necesidades que definieron la arquitectura, desde qué se espera que haga en el entorno de prueba hasta la división de la red, la identificación y el control de acceso.

El objetivo es tener una base firme y consistente con los principios de seguridad actuales para arquitecturas que mezclan recursos locales y en la nube. Este método sigue las recomendaciones del NIST en su publicación especial sobre arquitecturas Zero Trust [1].

3.1 Requisitos funcionales del entorno simulado

El entorno desarrollado tenía que cumplir con varias funciones clave para simular el comportamiento de una infraestructura empresarial distribuida. Primero, se requería un sistema centralizado de gestión de identidades, basado en LDAP, para almacenar información estructurada de los usuarios, sus atributos y roles. A partir de ahí, se ha incorporado un mecanismo de autenticación federada mediante Keycloak, capaz de conectarse al LDAP y actuar como punto de acceso a los distintos servicios [2].

El entorno también debía incluir una aplicación web accesible para usuarios autenticados, simulando el acceso a un recurso corporativo. Esta aplicación estaría respaldada por una base de datos PostgreSQL, gestionada desde un contenedor independiente. También, se ha considerado importante tener un sistema de proxy para controlar y filtrar el tráfico web (Squid) [8], así como un firewall (pfSense) para segmentar el tráfico entre nodos [14].

Cada componente debía desplegarse en un entorno virtualizado en Proxmox VE, con una distinción clara entre máquinas virtuales (servicios on-premise) y contenedores (simulando el entorno en la nube). Esta diferenciación buscaba reflejar la estructura de una arquitectura cloud híbrida, manteniendo la gestión centralizada y segura de identidades, accesos y datos [7]. Por último, aunque se consideraron herramientas de monitorización como Wazuh o Suricata [10][15], se ha decidido posponer su implementación para el futuro, dando prioridad a la estabilidad y coherencia del entorno.

3.2 Consideraciones de seguridad bajo un modelo Zero Trust

Se ha utilizado un enfoque Zero Trust, la seguridad no depende de dónde esté el usuario o el recurso, sino de una verificación continua basada en varios factores y en políticas dinámicas que se adaptan al contexto. En este entorno, se han aplicado los siguientes principios:

- **Verificación continua de identidad y acceso:** todo acceso a la aplicación web, a la base de datos y a los servicios internos debe hacerse después de autenticar al usuario a través de un proveedor centralizado como Keycloak, que a su vez valida las credenciales con el directorio LDAP. Esta doble capa permite separar la gestión de identidades de los servicios finales y facilita el seguimiento del acceso [2].

- **Acceso con los privilegios mínimos:** cada usuario solo puede acceder a los recursos que necesita para su trabajo, según su departamento y rol, tal como se especifica en los atributos y mapeos del sistema de autenticación y la aplicación. La división del acceso se refleja tanto en las políticas de Keycloak como en el código de la aplicación web.

- **Supervisión del tráfico y control del flujo de datos:** para regular el tráfico y el flujo de datos, se ha implementado un proxy Squid que limita el acceso web a servicios internos autorizados, como la aplicación web de la empresa y la interfaz de administración LDAP (phpLDAPadmin). Esto impide que los usuarios exploren libremente la web o dirijan el tráfico a recursos externos, simulando así un entorno controlado donde las comunicaciones salientes se supervisan de cerca.

  Simultáneamente, se ha instalado un firewall pfSense con reglas básicas de entrada y salida entre zonas, asegurando que cada componente (como LDAP, la base de datos o la aplicación web) se comunique solo con los nodos necesarios. Este método ayuda a mantener un control preciso del flujo de datos, siguiendo el principio de confianza mínima del modelo Zero Trust [8][14].

- **Independencia del plano de confianza:** el diseño del entorno asegura que ningún componente se considere automáticamente confiable solo por estar en la red local. Esto se ve en la necesidad de autenticarse incluso desde dentro del entorno virtual, y en la configuración específica de cada flujo de datos entre máquinas y contenedores, en lugar de permitir el tráfico implícito o predeterminado [1].

Estas consideraciones aseguran que la arquitectura cumpla con los principios del modelo Zero Trust, siguiendo las recomendaciones del NIST y las prácticas que se describen en los manuales técnicos de herramientas como Keycloak, Squid y pfSense [2][8][14].

3.3 Criterios de autenticación, control de acceso y visibilidad

Un aspecto central al implementar arquitecturas Zero Trust es crear buenos sistemas de autenticación, control de acceso y visibilidad. Estas herramientas ayudan a confirmar la identidad de usuarios y aparatos, limitar sus permisos y observar cómo actúan dentro del sistema. En el laboratorio que se ha creado, se han utilizado varias estrategias para lograr estos tres objetivos.

- **Autenticación centralizada y federada:** se ha empleado un sistema doble donde OpenLDAP guarda las identidades principales y Keycloak ofrece una forma de autenticación basada en OpenID Connect (OIDC). Esta combinación hace que la autenticación sea fuerte y adaptable, usando detalles como el departamento o el tipo de empleado para cambiar sus permisos de acceso según sea necesario [2], [7].

- **Control de acceso basado en atributos y roles:** los permisos se dan según los detalles guardados en LDAP y mostrados en Keycloak, como el tipo de empleado. La aplicación web usa esta información para decidir qué mostrar a cada usuario, separando claramente a los administradores de los empleados. Este método sigue la idea de dar el mínimo permiso necesario, asegurando que cada usuario solo vea lo que necesita para su trabajo [2], [8].

- **Visibilidad y registro del acceso:** Keycloak registra los inicios de sesión, los errores de autenticación y la asignación de puestos. Esta información, que se puede ver desde su panel de control, ayuda a revisar completamente el ciclo de vida de los accesos. Además, herramientas como Squid y pfSense ayudan a ver cómo se mueve el tráfico y a controlar qué partes del sistema acceden a qué recursos, mejorando la visibilidad en todos los niveles [8], [14].

Estos sistemas, unidos en una arquitectura modular desplegada sobre Proxmox, aseguran que cada acceso sea verificado, limitado y registrado correctamente. Esta forma de trabajar está de acuerdo con lo que recomienda el NIST, que presta atención a la importancia de agregar telemetría, reglas dinámicas y separación lógica para seguir las ideas de Zero Trust [1].

4 Diseño y Despliegue de la Arquitectura Zero Trust

Implementar un modelo de seguridad Zero Trust exige una arquitectura planificada desde el inicio para asegurar el control de los flujos de información, la administración de identidades y el aislamiento de servicios. A diferencia de los enfoques comunes donde la red interna se adopta como segura, una arquitectura Zero Trust asume que cada solicitud de acceso debe ser revisada, autorizada y registrada, sin importar de dónde viene. Esto significa dividir los servicios, autenticar a los usuarios de forma constante y restringir los accesos con políticas detalladas.

Este capítulo presenta el diseño técnico y la puesta en marcha del ambiente creado, centrando la atención en la división de servicios, la virtualización de recursos y la unión de herramientas de seguridad. Se detalla cómo se ha organizado la infraestructura en máquinas virtuales y contenedores LXC dentro de Proxmox VE, y los criterios para dar funciones específicas a cada componente. Esta división permite copiar una situación real de arquitectura cloud híbrida, donde servicios locales (on-premise) y en la nube están bajo una misma administración de seguridad.

El despliegue se ha hecho sobre una instancia de Proxmox VE instalada en VMware, lo que permite tener un laboratorio autónomo y adaptable. Cada parte del sistema se encuentra aislada en su propia VM o contenedor, lo que ayuda a separar las funciones y reduce el riesgo de movimientos laterales. También, se han considerado las buenas prácticas de publicaciones especializadas, como la NIST SP 800-207 [1], que destaca el control de acceso constante, la telemetría contextual y la capacidad de respuesta ante incidentes.

El diseño resultante busca dar una base sobre la que se puedan añadir ampliaciones de seguridad, como herramientas SIEM o sistemas IDS/IPS, sin cambiar la arquitectura central. Se ha dado importancia a la compatibilidad y escalabilidad de los servicios, haciendo fácil su reproducción en otros ambientes corporativos o de formación.

4.1 Diseño lógico de la arquitectura Zero Trust

La arquitectura lógica definida sigue los principios de mínimo privilegio, segmentación estricta y control de los accesos. Los componentes se reparten en nodos virtuales independientes para reforzar el aislamiento y hacer más fácil el control del tráfico. La infraestructura está desplegada sobre Proxmox VE, organizada en una subred (192.168.1.0/24) para hacer más sencillo la configuración y concentrar el control en el firewall de borde (pfSense). En la Figura 1.1 se refleja el diagrama de la arquitectura del proyecto.

- Distribución de servicios:
    - Máquinas Virtuales (VM):
        - On-Premise (192.168.1.201): OpenLDAP.
        - Administration (192.168.1.202): WireGuard, Squid, NGINX.
        - Auth (192.168.1.205): Keycloak.
        - Firewall (192.168.1.206): pfSense.
    - Contenedores LXC:
        - Web Server (192.168.1.203): aplicación Flask.
        - Storage (192.168.1.204): PostgreSQL.

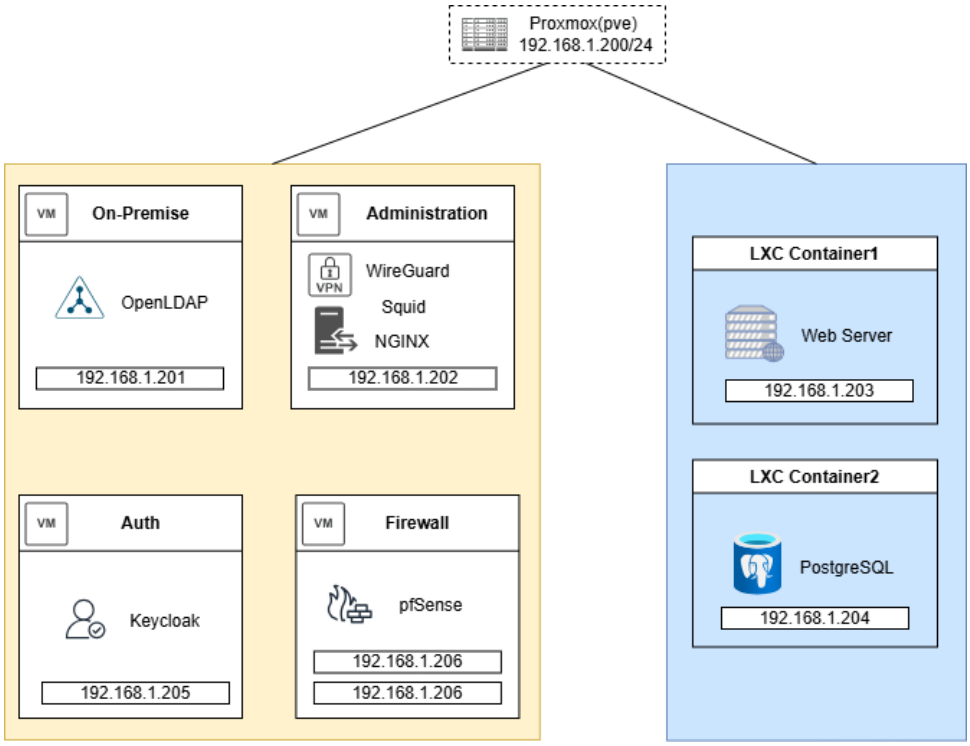


Figura 1.1: Diagrama de la arquitectura

El esquema muestra esta distribución lógica, separando los recursos on-premise (zona amarilla) de los servicios cloud (zona azul), simulando una infraestructura híbrida con segmentación de servicios, funciones diferenciadas y control de flujos.

Esta arquitectura hace más fácil la aplicación de políticas de seguridad Zero Trust, al fijar flujos de acceso definidos, reducir conexiones implícitas y permitir una gestión centralizada de políticas de acceso. Cada relación entre componentes está limitada por reglas del firewall y vigilada por el proxy, lo que permite un control de, qué servicios se comunican y en qué condiciones.

4.2 Máquinas virtuales y contenedores: implementación

El ambiente se ha implementado por completo en un solo nodo físico con Proxmox VE como hipervisor. Desde esta plataforma se han gestionado las instancias virtuales y contenedores que conforman la estructura Zero Trust. Aunque Proxmox VE incluye firewall, snapshots, backups y autenticación, en este proyecto se han limitado estas funciones a la creación, asignación de recursos y gestión de red de las VM y contenedores. El sistema de seguridad, control de tráfico, autenticación y monitoreo se ha hecho con servicios independientes, instalados dentro de las instancias virtualizadas.

Dentro de ese nodo, desplegamos máquinas virtuales para los componentes on-premise y por otra parte, desplegamos los contenedores LXC para simular los recursos en la nube. En la Figura 1.2 se refleja toda la arquitectura funcionando dentro de Proxmox VE.

Las ISO de las VM que se han utilizado han sido Ubuntu Server 24.04.2 (ubuntu-24.04.2-live-server-amd64.iso) para on-premise, administration y auth, y para el firewall se ha usado la imagen oficial de pfSense Community Edition (pfsense-CE-2.7.2-RELEASE-amd64.iso) de Netgate [14].

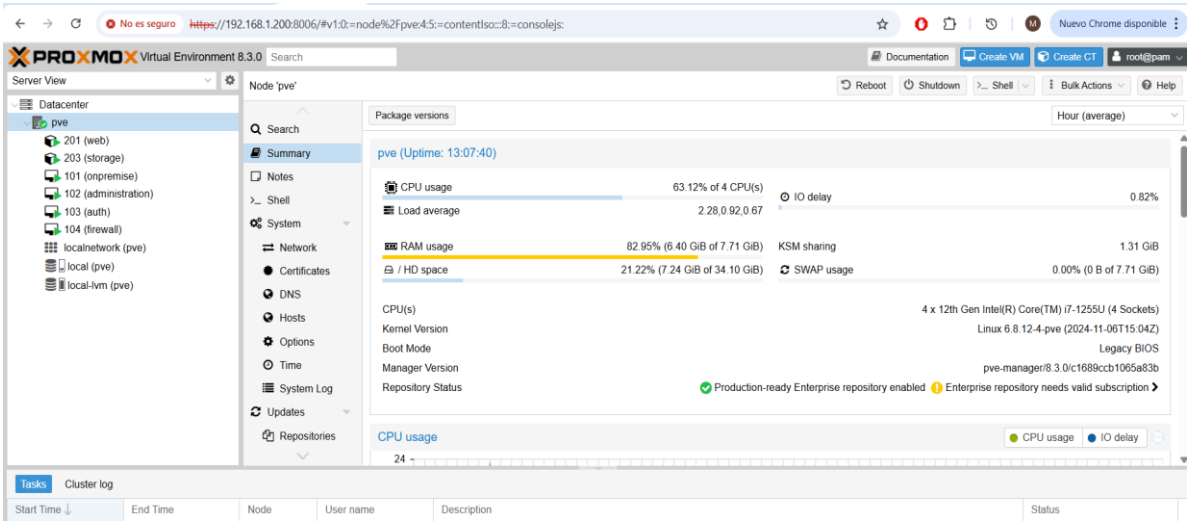


Figura 1.2: Imagen de la Arquitectura en Proxmox funcionando.

La estructura implementada es la siguiente:

- **VM On-Premise (192.168.1.201):** en esta máquina virtual se encuentra el servidor OpenLDAP, la fuente principal de identidades del proyecto, y es donde se guardan los usuarios, atributos y unidades organizativas. También se ha instalado phpLDAPadmin para la gestión gráfica del directorio [7].

- **VM Auth (192.168.1.205):** se encarga de la autenticación federada con Keycloak. Esta herramienta se conecta al directorio LDAP para validar

credenciales y dar tokens de acceso mediante OpenID Connect. Esta máquina sirve como interfaz de autenticación centralizada para los usuarios [2].

- **VM Administration (192.168.1.202):** aquí se han instalado servicios de apoyo y control, como el proxy Squid, el servidor NGINX para gestionar peticiones a la aplicación web y el servicio WireGuard (preparado para desarrollo futuro, pero no configurado). Esta VM concentra los servicios de administración del tráfico y reverse proxy [8]. La Figura 1.3 refleja el resumen de la creación de esta máquina virtual.

- **VM Firewall (192.168.1.206):** se ejecuta pfSense como firewall principal. Este componente permite definir reglas entre zonas y segmentar la red entre las VMs internas y los contenedores, simulando un ambiente segmentado y controlado como se espera en un modelo Zero Trust. La instalación se ha hecho desde ISO con el asistente gráfico de Netgate, y luego se ha configurado la interfaz LAN con la IP estática [14].

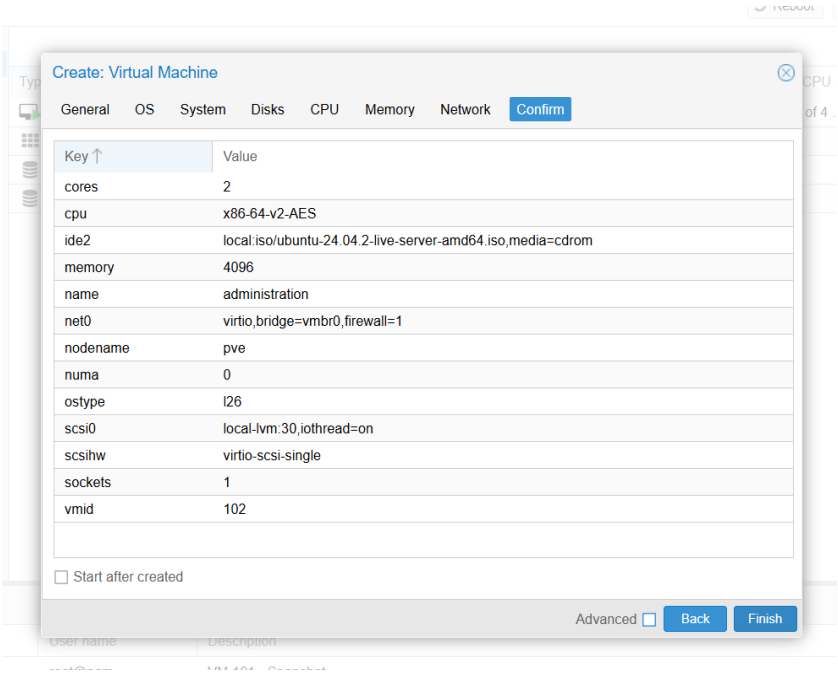


Figura 1.3: Imagen del resumen de la creación de una máquina virtual en Proxmox

- **Contenedor LXC1 (192.168.1.203):** servidor web. Aquí se ha implementado la aplicación web corporativa, desarrollada en Python (Flask) y con estilos CSS. Este componente es el servicio de acceso para los usuarios y está protegido por autenticación delegada en Keycloak.

- **Contenedor LXC2 (192.168.1.204):** base de datos PostgreSQL. Este contenedor tiene el sistema gestor de base de datos, usado para guardar la información estructural de la aplicación web y la base de datos de Keycloak. La conexión con la aplicación se hace con credenciales seguras

y direccionamiento interno dentro de Proxmox [16]. En la Figura 1.4 se refleja un resumen de la creación de este contenedor.

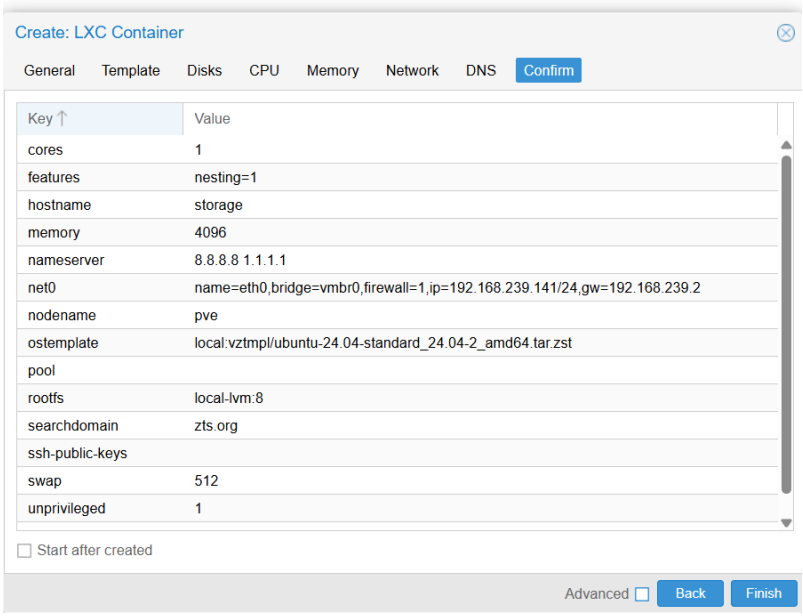


Figura 1.4: Imagen del resumen de la creación de un contenedor LXC en Proxmox

Este diseño, junto con el uso de estas tecnologías en un ambiente virtualizado, da la flexibilidad necesaria para simular políticas de control de acceso, monitoreo y segmentación, que son clave en el modelo Zero Trust propuesto por el NIST [1].

4.3 Implementación de servicios

En esta sección, se explicará cada uno de los servicios que se ha desarrollado junto con sus características. En primer lugar, se comenzará por OpenLDAP, Keycloak y el desarrollo de la web junto con el almacenamiento en PostgreSQL.

4.3.1 OpenLDAP

Se ha utilizado OpenLDAP para construir el sistema de administración de identidades, y se ha instalado en una máquina virtual con Ubuntu Server 24.04.2. OpenLDAP permite el almacenamiento, estructuración y consulta de información sobre usuarios de manera jerárquica y escalable. En un modelo de seguridad Zero Trust, donde la identidad del usuario es fundamental, OpenLDAP juega un rol importante en los controles de acceso.

La estructura del directorio se ha configurado bajo el dominio dc=zts,dc=org, simulando una empresa con cinco departamentos: Tecnología (IT),

Comunicaciones (COMM), Recursos Humanos (RRHH), Marketing (MRK) y Finanzas (FS). Se ha creado una unidad organizativa (ou) para cada uno, con al menos cinco usuarios y atributos a la medida. Los nombres de usuario siguen el formato nombre.apellido@departamento.zts.org, que sirve como identificador único y atributo de autenticación. En la Figura 1.5 se muestra la consulta de datos de un usuario siendo el usuario admin el que accede a la interfaz.

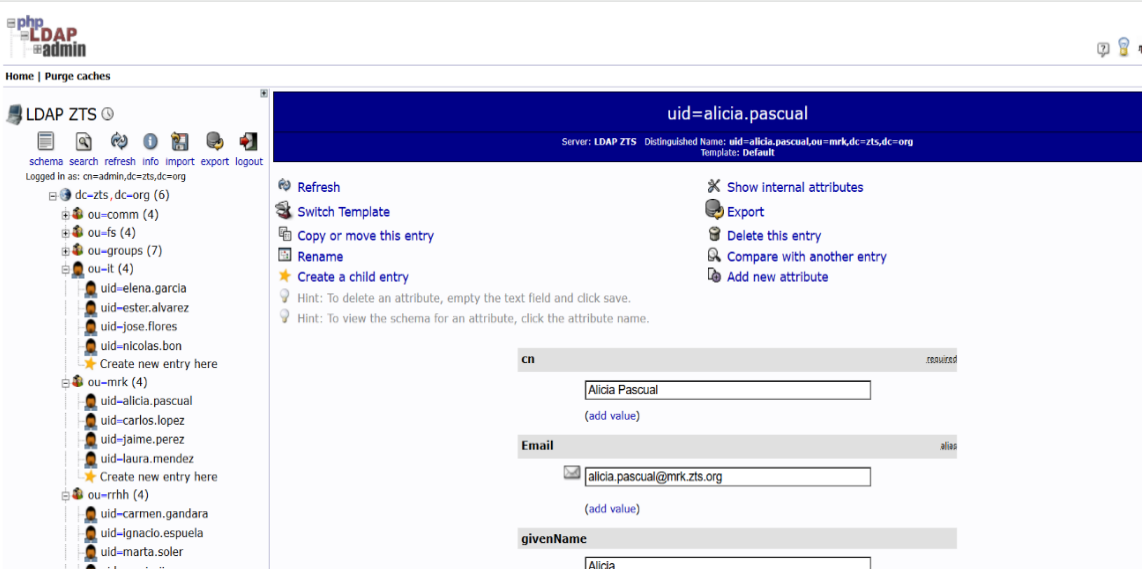


Figura 1.5: Imagen de la interfaz phpLDAPAdmin cuyo login es de un usuario admin.

Se ha añadido un atributo employeeType para diferenciar empleados y administradores, aparte del atributo mail. Este dato es importante en la aplicación web y el sistema de autenticación federada, pues posibilita la creación de políticas de acceso específicas según el rol del usuario. Por ejemplo, un usuario con employeeType=admin en ou=IT puede acceder a funciones administrativas, mientras que un usuario con employeeType=employee solo puede ver contenido de su departamento.

La estructura de grupos se ha diseñado bajo una lógica de control delegada. En ou=groups, se crearon entradas LDAP del tipo groupOfNames, incluyendo cn=admins, que agrupa a los administradores delegados de IT, COMM y RRHH. Esta agrupación se usó para definir Listas de Control de Acceso (ACLs) que permiten, por ejemplo, que los administradores solo puedan modificar objetos de su propio departamento, siguiendo el principio de mínimo privilegio. Estas reglas se han puesto en marcha a través de archivos LDIF y aplicadas con ldapmodify sobre el backend de configuración dinámica (cn=config), evitando reiniciar el servicio slapd. En la Figura 1.6 se muestra el resultado de una acción de un usuario sin permisos suficientes.

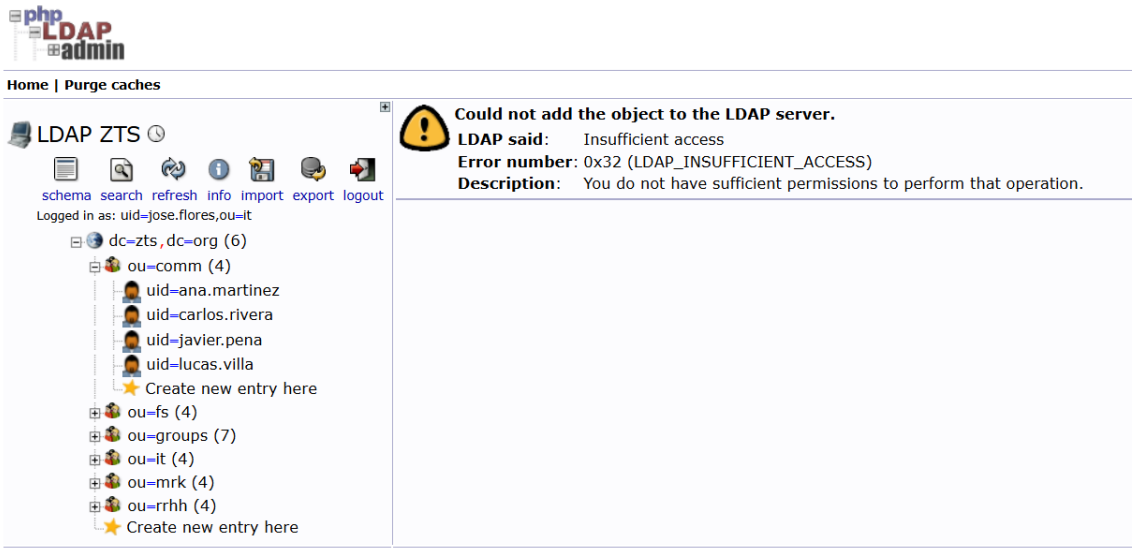


Figura 1.6: Imagen de la interfaz phpLDAPAdmin de un usuario que no tiene suficientes permisos para realizar dicha acción.

Durante las pruebas realizadas, se decidió no activar el cifrado TLS en el servicio LDAP, ya que el entorno se ejecuta en una red local simulada, sin salida a internet, con un punto de entrada controlado por Keycloak y políticas de filtrado mediante proxy y firewall. Esta configuración resulta aceptable cuando se ejecuta en un laboratorio cerrado, pero no sucede así en producción. Si el sistema se adaptara para accesos remotos, por ejemplo, con una VPN WireGuard, se debería activar ldaps:// en el puerto 636, generar certificados válidos y configurar la autenticación mutua para proteger las credenciales.

Finalmente, la interfaz phpLDAPadmin se ha usado para revisar visualmente la organización de usuarios y grupos, y para verificar los permisos de edición de los administradores delegados. Esta herramienta se instaló en el mismo servidor LDAP para facilitar la validación y supervisión del directorio, llevando a una administración más sencilla en la fase de desarrollo.

4.3.2 Keycloak

Keycloak implementó el servicio de autenticación federada. Esta plataforma gestiona identidades y controla el acceso basándose en estándares como OpenID Connect (OIDC), OAuth 2.0 y SAML 2.0. Se instaló en una máquina virtual Ubuntu Server 24.04.2 en modo start-dev, lo que permite un inicio rápido sin necesidad de configurar una base de datos externa o certificados. Esta modalidad fue adecuada para un entorno de laboratorio cerrado, aunque en implementaciones de producción, se sugiere configurar el sistema en modo persistente con TLS habilitado y respaldo en una base de datos externa.

Keycloak actúa como puerta de entrada (ZTNA) al entorno Zero Trust, ya que todo acceso a la aplicación web requiere su autenticación centralizada. Para esto, se ha creado un Realm específico llamado zts, donde se han configurado usuarios, roles, clientes y mapeos personalizados.

La integración con OpenLDAP se ha hecho a través de la función de User Federation, conectando Keycloak a ldap://192.168.1.203 como servidor externo. En esta configuración, Keycloak sincroniza usuarios y atributos desde el árbol LDAP, respetando el dominio zts.org y accediendo con un bind DN con permisos de lectura sobre las entradas de usuario. Se han definido mapeos para obtener atributos como cn, mail y employeeType, este último esencial para distinguir roles en la aplicación web. La sincronización se ha programado manualmente desde la interfaz de administración con el fin de asegurar consistencia y seguimiento durante las pruebas. En la figura 1.7 se muestra la sincronización de usuarios de LDAP con Keycloak.

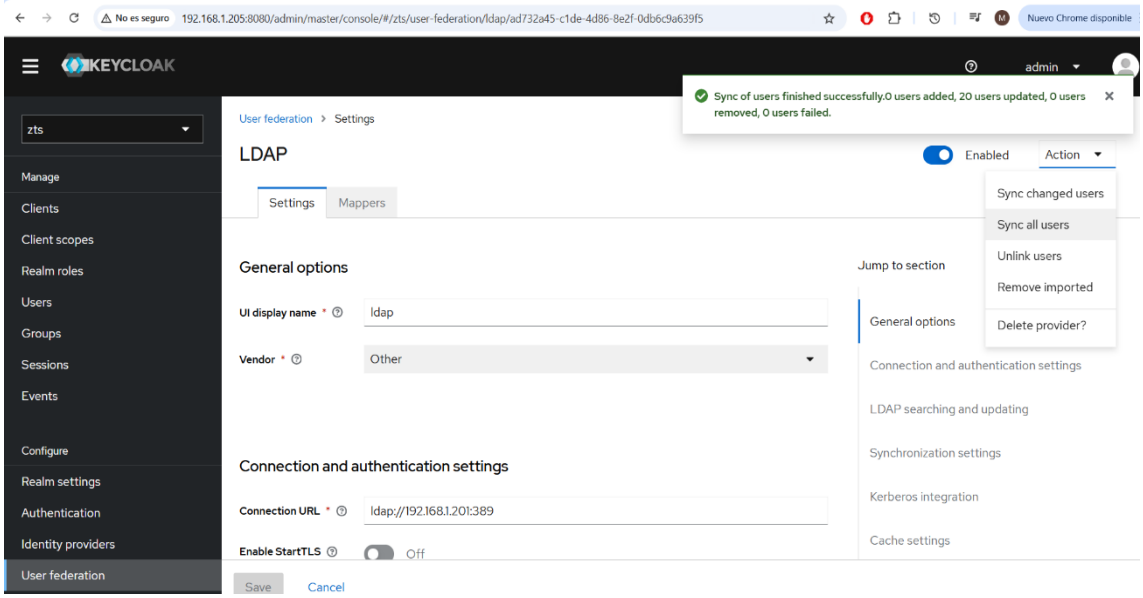


Figura 1.7: Imagen de la interfaz de Keycloak después de sincronizar los usuarios de LDAP.

En cuanto a la configuración de roles, se han creado dos principales: admin y employee, mapeados automáticamente desde el atributo employeeType del directorio LDAP. Esta asignación dinámica permite a Keycloak emitir tokens OIDC con información contextual sobre el usuario autenticado. La aplicación Flask interpreta esta información para controlar la interfaz y los permisos de cada sesión.

Adicionalmente, se ha definido un cliente llamado zts-web-dedicated, se muestra en la Figura 1.8, con protocolo OIDC, configurado como aplicación pública. El redirect_uri es http://192.168.1.200:8000/callback, el punto de entrada de la aplicación después de la autenticación. Este cliente ha sido

configurado con Access Type: public, Standard Flow Enabled y User Info Signature activada. La comunicación entre Keycloak y la aplicación se hace mediante intercambio de tokens y consulta al endpoint userinfo para obtener los datos del usuario en cada sesión.

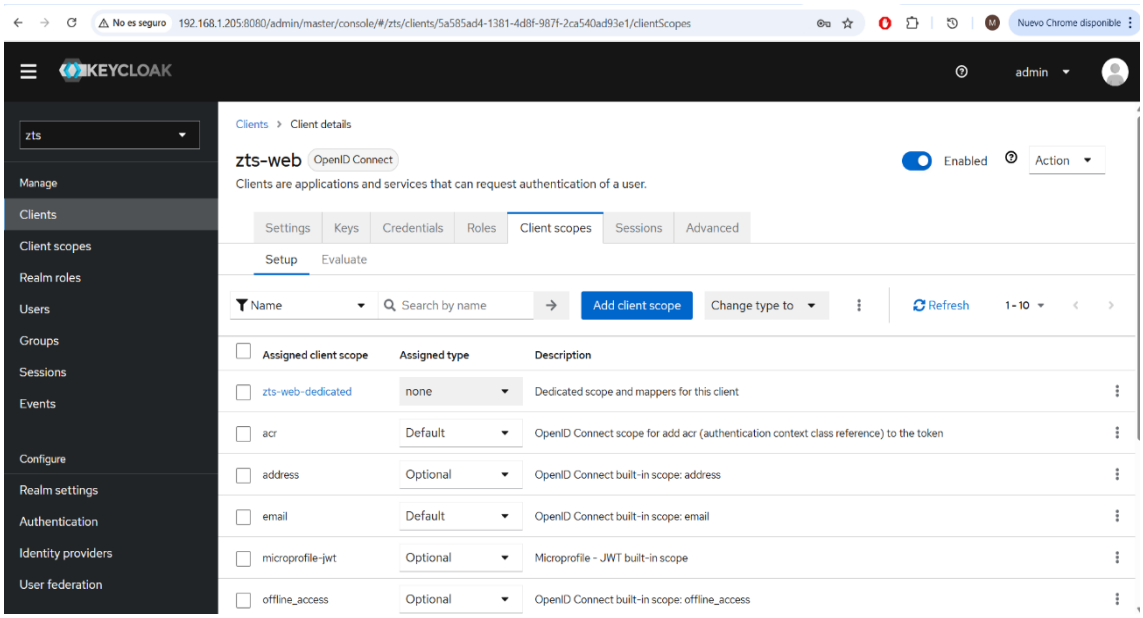


Figura 1.8: Imagen de la interfaz de Keycloak donde se ha definido el cliente.

Para mejorar el seguimiento y la visibilidad del sistema, Keycloak registra eventos detallados, como accesos exitosos, errores de autenticación, asignación de roles y sincronizaciones LDAP. Esta información ha sido clave para verificar el funcionamiento correcto de la federación y el mapeo de permisos, así como para documentar el comportamiento esperado en una arquitectura Zero Trust.

Es importante notar que, aunque el despliegue se ha realizado sin TLS ni persistencia en una base de datos externa, esta decisión se tomó debido al enfoque de laboratorio: todos los servicios están en una red local simulada y el acceso está dentro del entorno virtualizado. En un entorno de producción, sería necesario conexiones seguras (HTTPS) con certificados válidos, configurar el modo de producción de Keycloak y guardar la información en un backend PostgreSQL protegido, lo cual Keycloak soporta desde sus versiones recientes.

4.3.3 Aplicación Web y Base de datos PostgreSQL

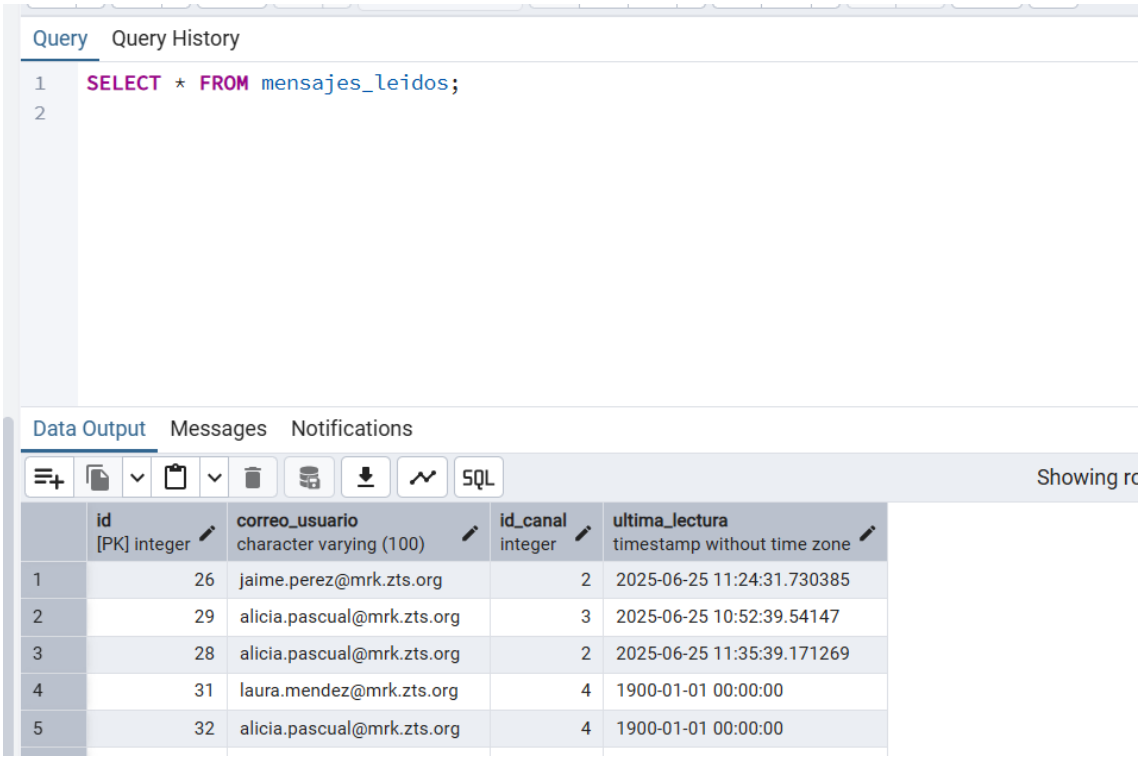
La aplicación web corporativa, un componente central, se ha desarrollado en Python, usando Flask. Esta aplicación da acceso autenticado a los usuarios. Los contenidos se ven de manera diferente según el puesto y sección del usuario. Adicionalmente, se pueden manejar conversaciones internas, simulando un sistema básico de comunicación interna a través de chats privados y canales.

Para la presentación visual, se ha utilizado HTML5 y CSS3, con Bootstrap para una interfaz moderna y adaptable.

Al principio del proyecto, la autenticación se hacía conectando directamente al servidor LDAP. Después, se integró Keycloak como proveedor OIDC. Esto permitirá a los usuarios autenticarse mediante un sistema que usa el directorio LDAP y produce un token de acceso. Este token contiene la información necesaria para configurar la sesión del usuario, como nombre, correo electrónico y puesto (sacado del atributo employeeType del directorio) [2].

Para guardar los datos (mensajes, canales, avisos), se ha usado un contenedor LXC con PostgreSQL, gestionado desde la máquina storage. La base de datos se divide en varias tablas que relacionan usuarios, mensajes y secciones, Esto permite guardar un historial de conversaciones y ver la actividad por canal. Se eligió PostgreSQL por su fiabilidad, fácil uso con Python y capacidad para operaciones relacionales complejas [16]. En la Figura 1.9 se muestra una consulta de mensajes leídos de un canal.

Los puestos se dividen en dos niveles: por sección (tomando el correo del usuario con formato usuario@departamento.zts.org) y por tipo de empleado (con el atributo employeeType, que distingue entre admin y empleado). Keycloak usa estos datos para controlar el acceso, y la aplicación Flask los usa para definir la presentación, mostrando, por ejemplo, solo el canal admin a los administradores, o limitando los mensajes visibles a los de la misma sección en el caso de los empleados.



	id [PK] integer	correo_usuario character varying (100)	id_canal integer	ultima_lectura timestamp without time zone
1	26	jaime.perez@mrk.zts.org	2	2025-06-25 11:24:31.730385
2	29	alicia.pascual@mrk.zts.org	3	2025-06-25 10:52:39.54147
3	28	alicia.pascual@mrk.zts.org	2	2025-06-25 11:35:39.171269
4	31	laura.mendez@mrk.zts.org	4	1900-01-01 00:00:00
5	32	alicia.pascual@mrk.zts.org	4	1900-01-01 00:00:00

Figura 1.9: Imagen de una consulta en pgAdmin4.

La aplicación web se conecta con PostgreSQL usando el controlador psycopg2, y las claves se guardan en variables de entorno protegidas. La base de datos solo muestra su puerto dentro del entorno virtualizado, cumpliendo con la idea de reducir al mínimo la exposición de riesgos en arquitecturas Zero Trust [1].

4.4 Implementación de servicios

Como parte central de una arquitectura Zero Trust, se han establecido controles de tráfico de red para asegurar que solo las comunicaciones requeridas y autorizadas pudieran realizarse. Se han incluido dos herramientas principales: Squid, como proxy transparente para manejar el tráfico HTTP/HTTPS de salida, y pfSense, como firewall de red para definir las reglas de comunicación entre los nodos en el entorno simulado.

4.4.1 Proxy Squid: Filtrado de Salida Según Políticas de Confianza

El servidor proxy Squid ha sido configurado en la máquina de administración para actuar como punto de control obligatorio para el tráfico web. Bajo el enfoque Zero Trust, esto permite limitar los destinos accesibles desde la red interna solo a los dominios necesarios para que los servicios funcionen y para simular el uso corporativo real.

- La configuración del proxy incluye:
  - Control de acceso por red (ACL de origen)
    - `acl localnet src 192.168.1.0/24`
  - Control de acceso por destino (ACL de destinos permitidos)
    - `acl allowed_sites dstdomain .zts.org .ubuntu.com`
    - `acl pgadmin dstdip 192.168.1.204`
    - `acl phpldap dstdip 192.168.1.201`
  - Permisos explícitos
    - `http_access allow localnet`
    - `http_access allowed_sites`
    - `http_access allow pgadmin`
    - `http_access allow phpldap`

El acceso se permite solo si se cumplen ambas condiciones (origen autorizado y destino permitido). Todo el tráfico no cubierto por las reglas anteriores es bloqueado, siguiendo el principio denegar por defecto de Zero Trust [8].

En la Figura 1.10 se muestra la imagen de la configuración aplicada en Squid.

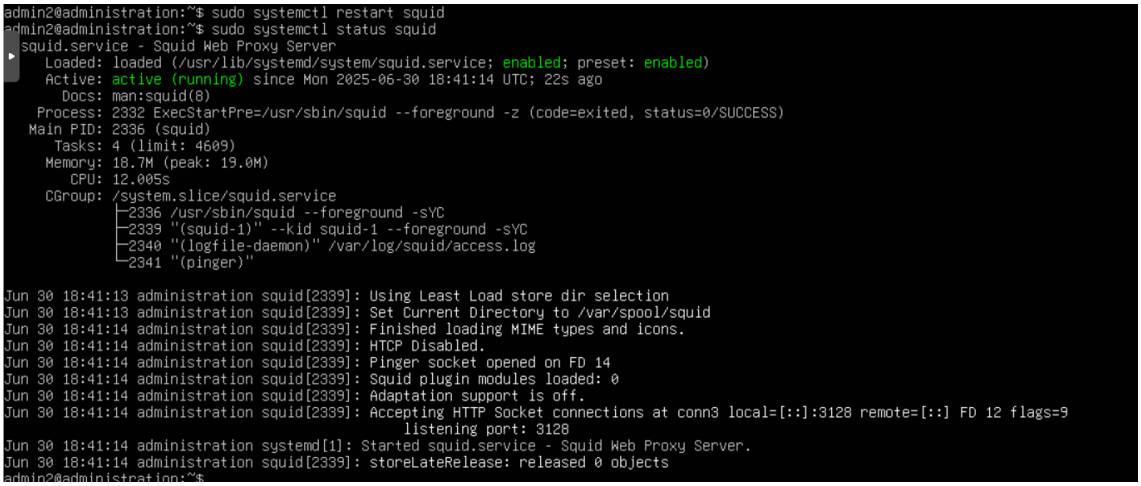


Figura 1.10: Imagen donde se muestra la configuración de Squid.

Esta configuración asegura que los usuarios autenticados no puedan acceder libremente a Internet o realizar peticiones externas no autorizadas, reforzando el perímetro de seguridad desde el interior.

4.4.2 Firewall pfSense: Segmentación Lógica y Control Entre Zonas

Para reforzar la separación entre componentes del entorno, se ha implementado un firewall virtual basado en pfSense. Aunque Proxmox incluye funciones nativas de firewall, se optó por pfSense para tener más visibilidad, personalización y documentación de las reglas aplicadas [14]. En la Figura 1.11 se muestra el menú principal de pfSense.

Este firewall, desplegado como una VM conectada a la interfaz bridge vmbr0, simula la separación entre redes (LAN, servicios, autenticación) a través de reglas definidas:

- Permitir servicios esenciales: se permite el tráfico desde LAN net hacia puertos como 80 (HTTP), 389 (LDAP), 5432 (PostgreSQL) y 3128 (proxy).

- Permitir autenticación OIDC: se permite el acceso al puerto 8080 del nodo de Keycloak (192.168.1.205) para el flujo de inicio de sesión OIDC.

- Acceso controlado a la aplicación web: se permite solo el tráfico necesario hacia la IP del contenedor web (192.168.1.203) en el puerto 8000.

- DNS interno: se permite el puerto 53 (UDP) hacia la IP del propio pfSense si se usa como resolver local.

- Regla final de bloqueo: se bloquea cualquier otro tráfico no autorizado de forma explícita.

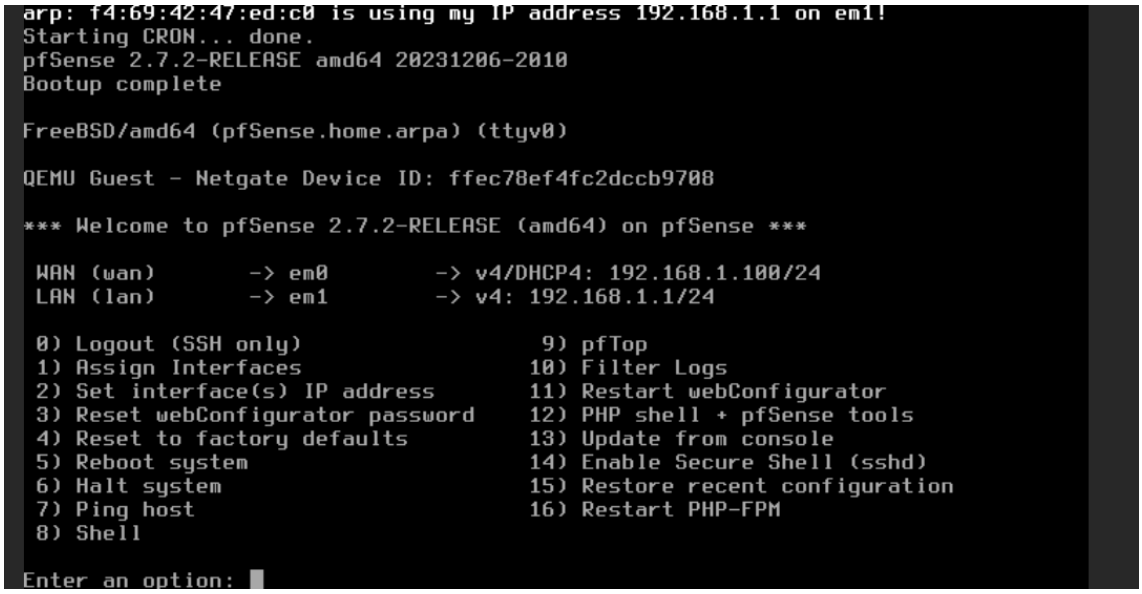


Figura 1.11: Imagen donde se muestra el menú principal de PfSense.

Con esta estrategia, se asegura que cada VM o contenedor solo puede comunicarse con lo necesario para su función, restringiendo el movimiento lateral y evitando riesgos innecesarios. Esto refuerza el aislamiento lógico de la infraestructura y aplica el modelo de microsegmentación [1].

Dado que este entorno está virtualizado por completo en un único host físico, sin salida real a Internet, se decidió reutilizar una sola interfaz de red (vmbr0) para todas las VMs, incluido el firewall. El filtrado y la segmentación se aplican de forma lógica, con pfSense como punto central de control. Esta elección permite simular una arquitectura Zero Trust sin comprometer la estabilidad de los servicios ya implementados.

5 Evaluación del modelo Zero Trust

Después de implementar la arquitectura, se ha verificado que los principios del modelo Zero Trust se han aplicado de forma correcta. Esta evaluación se ha centrado en revisar los controles de acceso, la segmentación de la red y la autenticación. También, se ha confirmado la respuesta del sistema a varios roles y usos. La evaluación tiene tres partes:

- Validación de las políticas de segmentación
- Pruebas de autenticación federada
- Análisis del comportamiento de la aplicación web según el rol del usuario

El objetivo que se ha establecido confirmar que el entorno funciona y mostrar cómo las tecnologías (LDAP, Keycloak, Squid, pfSense y PostgreSQL) se unen para dar una seguridad que se adapte al perfil y al objetivo de cada usuario. Como dice el NIST en su modelo Zero Trust, la confianza debe depender de políticas dinámicas y una visibilidad constante, que va más allá de la red [1].

5.1 Validación de las políticas de segmentación

La segmentación lógica de servicios se ha considerado muy importante en el diseño de la arquitectura. Con el firewall pfSense, se han creado reglas que permiten solo las comunicaciones necesarias entre los componentes. En las pruebas, se ha revisado que los servicios importantes (como la autenticación LDAP, la consulta a la base de datos o el acceso al proxy) solo podían accederse desde las máquinas que lo requerían.

Estas reglas aseguran que cada componente ha funcionado sin ir más allá de lo necesario, lo que concuerda con el principio de mínimo privilegio y microsegmentación en entornos Zero Trust [1][14].

Por otro lado, el proxy Squid limitó bien el tráfico saliente a los dominios corporativos (.zts.org) y herramientas específicas (.ubuntu.com, IPs de pgAdmin y phpLDAPadmin). Se ha negado cualquier intento de conectarse a Internet fuera de esos dominios.

Esta segmentación, aunque usa una sola interfaz bridge (vmbr0) compartida en Proxmox, ha sido suficiente para simular entornos LAN y WAN mediante filtrado lógico. Esto reproduce condiciones realistas de aislamiento entre zonas [8].

5.2 Pruebas funcionales de autenticación y acceso

Se han hecho pruebas funcionales para juzgar el sistema de autenticación. Estas pruebas imitaban a usuarios reales que han entrado a la aplicación web a través de Keycloak, que está enlazado con OpenLDAP. En la Figura 1.12 se

muestra la pantalla de Keycloak sobre el cliente. El propósito ha sido revisar si el sistema de control de acceso funcionaba bien y si los usuarios han sido verificados según sus características, siguiendo el modelo Zero Trust.

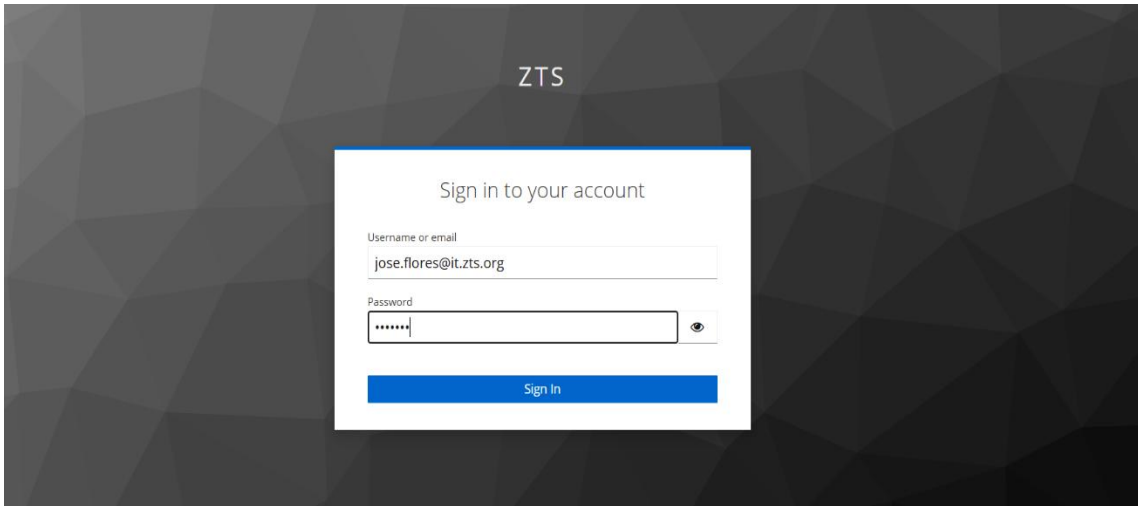


Figura 1.12: Imagen del login de Keycloak con el cliente zts.

Cuando un usuario ha iniciado sesión con Keycloak, el sistema guardaba datos del inicio de sesión como la hora, el usuario, la dirección IP y si la autenticación fue exitosa. En la Figura 1.13 se refleja el resultado de registro de un usuario. Esto ayudaba a revisar lo que pasaba y encontrar intentos fallidos o accesos sin permiso. Se han podido ver las sesiones activas y su historial en la consola de administración de Keycloak, lo que apoyaba la idea de visibilidad continua del NIST [1].

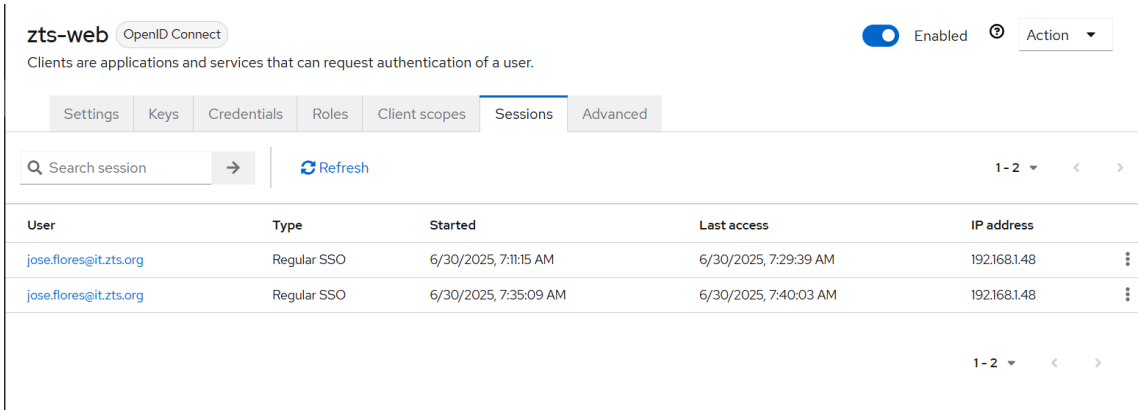


Figura 1.13: Imagen del registro de sesión de Keycloak de un usuario.

La aplicación web, se ha desplegado en el contenedor web, cambiaba su forma de trabajar según la característica employeeType que daba el directorio LDAP y que Keycloak ha interpretado bien. Las pruebas mostraron que:

- Los usuarios con el rol de empleado, se refleja en la Figura 1.14, solo podían entrar al canal de su departamento (como finanzas o marketing), donde podían ver avisos y hablar con sus compañeros.

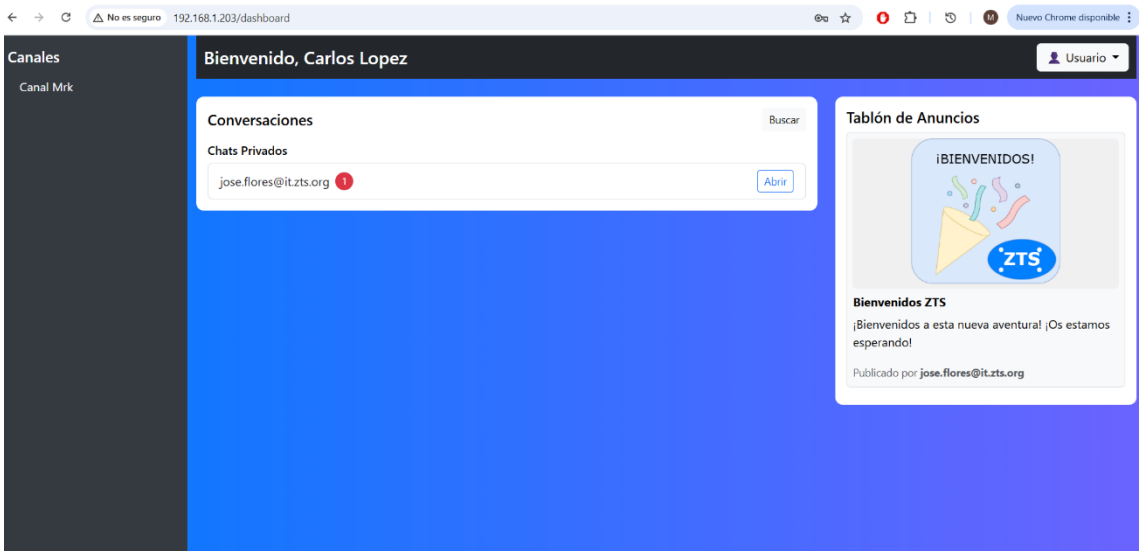


Figura 1.14: Imagen de la pantalla principal de la web corporativa como si fuese un usuario que no es admin.

- Los usuarios con el rol de administrador, se muestra en la Figura 1.15, también podían entrar al canal general de administración, con más opciones para supervisar. En la Figura 1.16 se muestra un ejemplo del chat de canal.

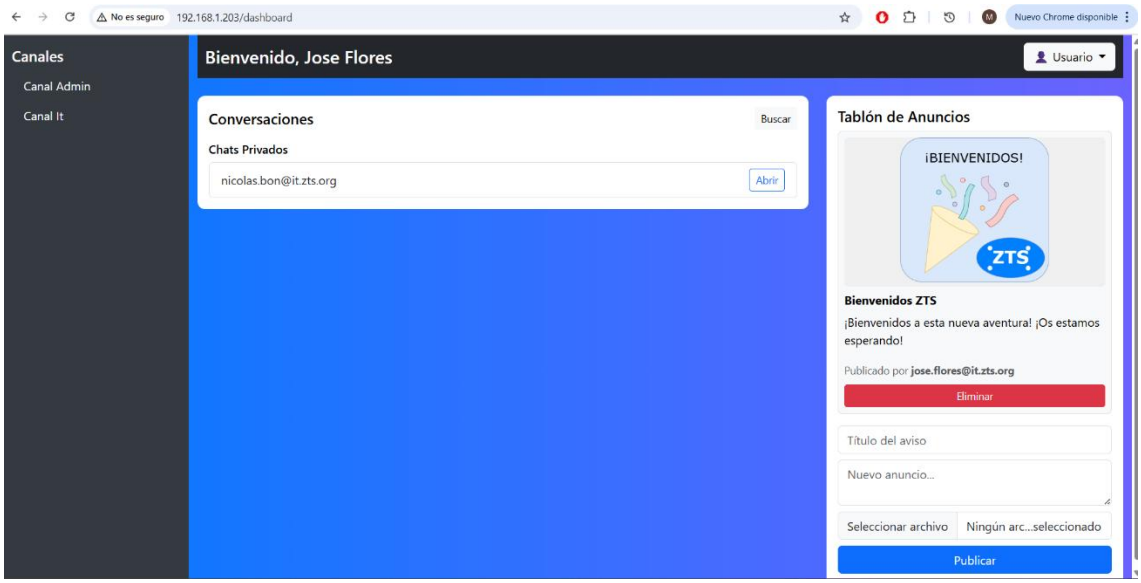


Figura 1.15: Imagen de la pantalla principal de la web corporativa como si fuese un usuario admin.

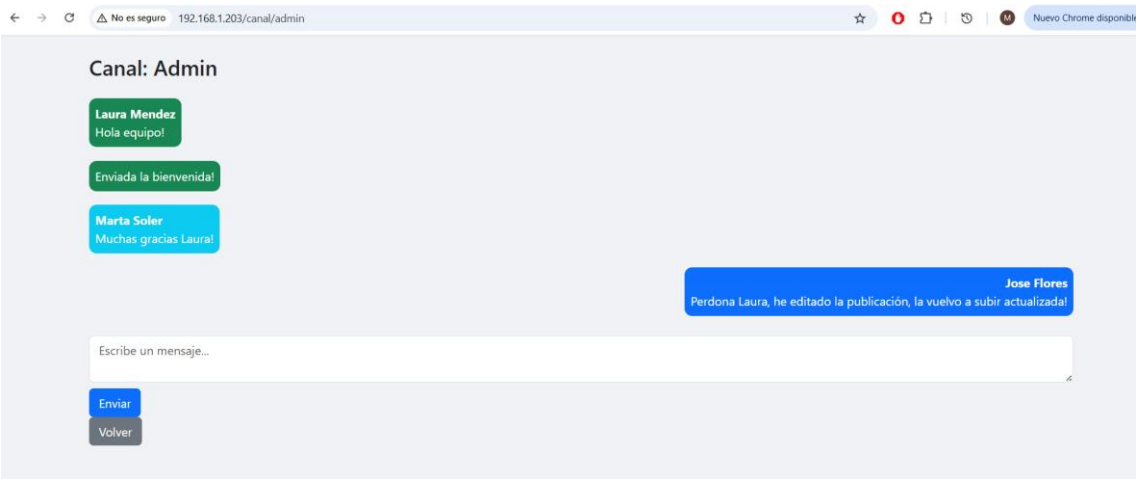


Figura 1.16: Imagen de ejemplo de un canal de mensajes.

- Si alguien intentaba entrar directamente a partes protegidas sin haber iniciado sesión, el sistema lo mandaba al inicio de sesión de Keycloak, cumpliendo con la regla de acceso autenticado siempre.

El sistema ha funcionado bien cuando usuarios sin permisos o con características mal configuradas han intentado entrar, ya que no se les ha permitido ver el contenido. Estas pruebas dicen que los sistemas de autenticación, autorización y control de visibilidad trabajan como se esperaba, haciendo más fuerte el diseño modular y seguro.

5.3 Análisis de resultados

La implementación del modelo Zero Trust en el entorno simulado ha validado los principios clave definidos por el NIST [1], los cuales han sido aplicados a un laboratorio híbrido con recursos segmentados, autenticación centralizada y control de acceso granular. El análisis de los resultados se ha organizado en cuatro ejes:

- **Apego al principio de mínimo privilegio:** los usuarios solo han podido tener acceso a los recursos definidos por su rol (employeeType) y departamento, esto gracias a una mezcla de autenticación con Keycloak y lógica de visibilidad en la aplicación. Esta división lógica ha permitido que empleados y administradores vieran distintos contenidos, evitando accesos cruzados no autorizados. Este método afirma la idea de acceso just-in-time y just-enough, elementos clave en diseños Zero Trust [1].

- **Verificación continua y trazabilidad:** El sistema ha mostrado un monitoreo correcto de los eventos de inicio de sesión y manejo de sesiones activas. Esta capacidad de trazabilidad, hasta en un ambiente local, permite cumplir con los estándares de verificación continua [2].

- **Segmentación y aislamiento de componentes:** el uso de pfSense como firewall y Squid como proxy ha permitido aplicar políticas claras de comunicación entre componentes. El acceso a internet se ha restringido a dominios corporativos (.zts.org), repositorios del sistema (.ubuntu.com), y servicios internos (phpLDAPadmin y pgAdmin). Se ha negado por defecto, toda conexión fuera de este margen, bajando la superficie de ataque y dando más fuerza al perímetro interno, siguiendo las sugerencias de seguridad modernas [8][14].

- **Consistencia entre la arquitectura lógica y el comportamiento real:** se ha validado que la estructura definida en el diseño lógico se ajusta al comportamiento visto: los servicios se comunican entre sí por medio de reglas claras, la app web trabaja sobre una base de datos aislada, y la identidad se ha manejado de forma centralizada. Las pruebas funcionales han confirmado que la autenticación es un requisito para tener acceso a cualquier recurso, y que la red no permite flujos no controlados.

En conclusión, los resultados logrados han mostrado que el entorno simulado reproduce con exactitud una arquitectura Zero Trust funcional, basada en estándares y herramientas libres. Se ha dado prioridad a la funcionalidad básica sobre la integración de herramientas avanzadas de monitoreo (como Wazuh o Suricata), pero la estructura creada es una base buena para expansiones futuras.

6 Conclusiones

6.1 Evaluación del entorno y del modelo Zero Trust

El proceso de desarrollo ha validado la viabilidad de establecer una arquitectura Zero Trust operativa y consistente mediante recursos de código abierto, virtualización y servicios bien definidos. La integración de tecnologías como OpenLDAP, Keycloak, Squid y pfSense ha hecho posible la aplicación de los principios centrales del modelo Zero Trust: confirmación continua, acceso con los privilegios mínimos requeridos, segmentación lógica y gestión explícita del tráfico.

La estructura ejecutada ha replicado un escenario híbrido, en el que los servicios corporativos se dividen entre máquinas virtuales (locales) y contenedores (simulando la nube), con una administración centralizada de los accesos. Esto ha permitido una simulación realista del comportamiento de una organización con múltiples departamentos, roles diferenciados y medidas de control y protección activas. La aplicación web ha funcionado como punto de convergencia, mostrando cómo los usuarios obtienen acceso a los recursos según sus atributos y permisos, lo que ha confirmado la validez del modelo de confianza cero.

6.2 Retos confrontados

En el curso del proyecto, se han definido varios retos que han tenido influencia en los tiempos de ejecución y en la dificultad de la configuración:

- **Integración de TLS en OpenLDAP:** la activación del módulo TLS en slapd ha presentado varios inconvenientes vinculados con dependencias, falta de bibliotecas (tls.la) y errores en la carga de módulos. Dada la naturaleza contenida del entorno, se ha optado por desactivar TLS de forma provisional, justificándose por la segmentación interna y la ausencia de tráfico externo.

- **Configuración de Keycloak con OIDC y asignación de roles:** la federación con LDAP ha tenido éxito, pero la propagación de atributos personalizados (employeeType) al token OIDC ha demandado asignaciones manuales que no fueron detectadas de inmediato en la app, lo que ha complicado el control de la visibilidad. Se ha establecido una solución alterna que combinaba un doble inicio de sesión (Keycloak + autenticación LDAP) para asegurar el funcionamiento de la aplicación.

- **Limitaciones de red debido a la doble virtualización:** el uso de VMware + Proxmox en modo NAT ha complicado el establecimiento de túneles VPN con WireGuard y la salida hacia ciertos recursos. Esto ha requerido

de un nuevo diseño de la red en modo bridge y una reconsideración provisional de los escenarios de acceso remoto.

- **Carga operativa y limitaciones de tiempo:** el tiempo disponible ha impedido incorporar servicios avanzados como Wazuh, Suricata o sistemas de alerta en tiempo real. Sin embargo, su análisis y documentación se han incluido como una propuesta de desarrollo.

6.3 Sugerencias para mejorar

Con el objetivo de lograr una versión más sólida y cercana a un entorno de producción, se han propuesto las siguientes mejoras:

- **Activación de TLS para LDAP:** la incorporación de certificados válidos y conexiones LDAPS promueven la confidencialidad de las credenciales, incluso en redes mixtas o con acceso VPN.

- **Revisión total de OIDC y app web:** unificar la autenticación a través de Keycloak, removiendo la lógica de inicio de sesión paralela y asegurando que los atributos (departamento, rol) deben estar presentes en los tokens de acceso.

- **Integración de instrumentos SIEM y detección de amenazas:** la inclusión de Wazuh, Suricata y un motor antivirus como ClamAV u OSSEC hace posible el cumplimiento de la visibilidad extendida, la detección de irregularidades y la correlación de eventos, que son elementos básicos en una arquitectura Zero Trust desarrollada.

- **Simulación de escenarios de ataque:** para evaluar de manera más precisa la eficiencia del control de acceso, sería útil llevar a cabo pruebas de penetración controladas o simulaciones con instrumentos de ataque.

- **Implementación realista del acceso remoto:** concluir la configuración de WireGuard facilita una mejor representación del acceso desde fuera del perímetro, validando el modelo Zero Trust en situaciones más demandantes.

7 Análisis de Impacto

La finalidad de este trabajo de Fin de Grado se encuentra en proveer una solución de ciberseguridad avanzada para empresas emergentes o pequeñas, con un diseño de tecnologías accesibles, especialmente para aquellas que dispongan de recursos limitados. Se ha diseñado escalable con el fin de permitir la integración gradual y económica de una tecnología compatible.

Este modelo impacta positivamente en las empresas al reforzar su seguridad y disminuir el riesgo de ciberataques. Esta mayor seguridad se traduce en una continuidad operativa, incrementando la eficiencia puesto que minimiza interrupciones, costes y daños a la reputación.

Dada la complejidad y el coste de otras soluciones Zero Trust, se ha creado una alternativa accesible para organizaciones con menos recursos, utilizando tecnologías comunes como Proxmox VE, OpenLDAP, psSense, WireGuard, NGINX, Wazuh y Suricata.

El modelo está pensado para una implementación gradual y económica. Su escalabilidad y flexibilidad facilitan la integración y compatibilidad, dando una oportunidad de crecimiento y desarrollo a la arquitectura de la organización, permitiendo una mejora continua en su sistema de ciberseguridad.

En el aspecto social, este modelo ayuda a la formación y capacitación de personas. El uso de tecnologías accesibles mejora las habilidades laborales, el desarrollo profesional y la empleabilidad.

Desde un punto de vista cultural, el modelo incentiva la innovación, al ser fácil de integrar y ser escalable, lo que permite a las organizaciones desarrollar su arquitectura y evolucionar al mismo tiempo adaptándose a los cambios del entorno empresarial.

En cuanto al medio ambiente, el modelo promueve prácticas empresariales sostenibles. Al estar construido con VPN (WireGuard) y NGINX (proxy), herramientas esenciales para conexiones seguras y controlables, facilita el teletrabajo y reduce los desplazamientos de los empleados.

La implementación de este modelo Zero Trust en un entorno híbrido simulado mediante Proxmox VE, contribuye al Objetivo de Desarrollo Sostenible 8: Trabajo Decente y crecimiento Económico

Esta investigación se alinea con los apartados **8.2 y 8.3 del ODS** (enfocados en el aumento de la productividad económica mediante la modernización tecnológica y la innovación, alentando actividades productivas, emprendimiento, formalización, crecimiento de empresas pequeñas y la creación de empleo.

El modelo diseñado representa una modernización tecnológica que implica innovación, aumento de la productividad económica y reducción de costos. Al ser una solución avanzada y económica, reduce la barrera par adoptar tecnologías seguras, y ofrece oportunidades de crecimiento promoviendo la creación de empleos decentes.

De igual forma, incentiva el desarrollo personal y profesional, pues el modelo fomenta habilidades en ciberseguridad e impulsa el emprendimiento y la innovación.

Por tanto, al hacer la ciberseguridad avanzada más accesible, escalable y desarrollar la formación de los trabajadores, este proyecto contribuye directamente a la modernización del sistema económico, al crecimiento de las pequeñas organizaciones y a la creación de empleos, elementos centrales del **objetivo 8 de los ODS.**

Bibliografía

[1] J. Rose et al., “Zero Trust Architecture,” *NIST Special Publication 800-207*, 2020.

[2] S. Thorgersen and P. I. Silva, *Keycloak: Identity and Access Management for Modern Applications*, Packt Publishing Ltd., 2021.

[3] S. Arora, "Modern Cybersecurity: Zero Trust Networks and Beyond," *Journal of Information Security*, vol. 13, no. 2, pp. 85–101, 2022.

[4] J. Kindervag, "Build Security Into Your Network’s DNA: The Zero Trust Network Architecture," Forrester Research, 2010.

[5] S. Lal, "Microsegmentation in Zero Trust Architectures," *IEEE Security & Privacy*, vol. 19, no. 1, pp. 52–59, 2021.

[6] Executive Order 14028, "Improving the Nation’s Cybersecurity," *Federal Register*, vol. 86, no. 93, pp. 26633–26647, May 2021.

[7] W. Ahmed, *Mastering Proxmox: Build Virtualized Environments Using the Proxmox VE Hypervisor*, Packt Publishing Ltd., 2017.

[8] K. Saini, *Squid Proxy Server 3.1: Beginner’s Guide*, Packt Publishing Ltd., 2011.

[9]Donenfeld, J. A. (2017, February). WireGuard: Next Generation Kernel Network Tunnel. In NDSS (pp. 1-12).

[10] Wazuh, *Wazuh Documentation*. Wazuh Inc., 2023.

[11] J. Stoffer, *Suricata User Guide*. Open Information Security Foundation (OISF), 2018.

[12] ClamAV Team, *ClamAV User Manual*, Cisco Talos, 2022.

[13] B. Hay, *OSSEC Host-Based Intrusion Detection Guide*. Syngress, 2016.

[14] Netgate, "pfSense Documentation,".

[15] OISF, "Suricata - Open Source IDS/IPS/NSM engine,".

[16] PostgreSQL Global Development Group, *PostgreSQL 15 Documentation*.

Anexo

Fecha de entrega
2 jul 2025, 7:55 p.m. GMT+2

Fecha de descarga
2 jul 2025, 8:07 p.m. GMT+2

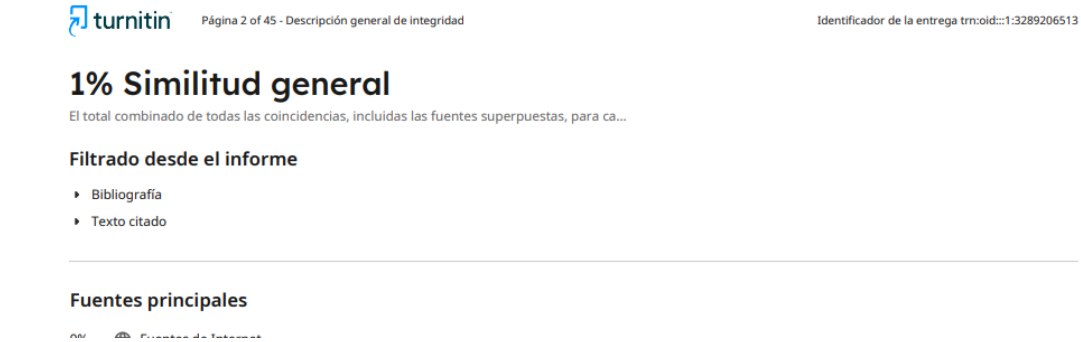
Nombre de archivo
9364_MERCEDES_JIMENEZ_DIAZ-VARELA_TFG_z170411_83762_1264417326.pdf

Tamaño de archivo
1.5 MB

11.377 Palabras

63.267 Caracteres

turnitin Página 1 of 45 - Portada Identificador de la entrega trn:oid:::1:3289206513



turnitin Página 2 of 45 - Descripción general de integridad Identificador de la entrega trn:oid:::1:3289206513

1% Similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para ca...

Filtrado desde el informe

- Bibliografía
- Texto citado

Fuentes principales

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Wed Jul 02 20:24:16 CEST 2025
	Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
	Numero de Serie	561
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)