

Trabajo fin de grado

Sistema DEVOPS para automatizar la creación de entornos y despliegues.



Pablo Sánchez Fernández del Pozo

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR



Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

**Sistema DEVOPS para automatizar la creación de
entornos y
despliegues.**

TFG

Autor: Pablo Sánchez Fernández del Pozo
Tutor: José Antonio Clavijo Blázquez

junio 2024

Todos los derechos reservados.

Queda prohibida, salvo excepción prevista en la Ley, cualquier forma de reproducción, distribución comunicación pública y transformación de esta obra sin contar con la autorización de los titulares de la propiedad intelectual.

La infracción de los derechos mencionados puede ser constitutiva de delito contra la propiedad intelectual (*arts. 270 y sgts. del Código Penal*).

DERECHOS RESERVADOS

© 6 marzo 2024 por UNIVERSIDAD AUTÓNOMA DE MADRID
Francisco Tomás y Valiente, nº 1
Madrid, 28049
Spain

Pablo Sánchez Fernández del Pozo
Sistema DEVOPS para automatizar la creación de entornos y despliegues.

Pablo Sánchez Fernández del Pozo
C\ Francisco Tomás y Valiente N° 11

*Dedicado a toda mi familia, por su apoyo incondicional durante todos estos años de estudios.
A mis amigos, por estar siempre presentes.
A Eva, por estar siempre pendiente y por todo su cariño.*

*Hace falta muy poco para tener una vida feliz;
está todo dentro de ti, en tu forma de pensar.*

Marco Aurelio

AGRADECIMIENTOS

En primer lugar dar gracias a todo el equipo de Delonia: Cesar, Ricardo, Stanis, More y Pepe, todos ellos unos profesionales y expertos en lo suyo, que me han apoyado y ayudado a resolver todas mis dudas siempre que lo he necesitado. Gracias a Jose Antonio, cofundador de Delonia, por brindarme esta oportunidad de realizar este proyecto de fin de grado en su empresa.

Más concretamente quiero agradecer a Sergio Martín por su paciencia al resolverme mis dudas de diseño y sus explicaciones magistrales de frontend que me han ahorrado más de un quebradero de cabeza.

Finalmente, querría agradecer a mi ponente Fco. Javier Gómez Arribas por su ayuda e implicación en la realización de la documentación de este proyecto.

RESUMEN

Este trabajo de fin de grado tiene como objetivo desarrollar una aplicación web de gestión empresarial que facilite el control y la administración de proyectos, hosts y despliegues en un entorno seguro y eficiente. Utilizando el sistema de autenticación, los usuarios acreditados podrán acceder a diversas funcionalidades protegidas mediante roles de usuario.

Entre los objetivos específicos se incluyen crear una interfaz intuitiva para automatizar el despliegue de aplicaciones, configurar y mantener entornos de desarrollo, pruebas y producción, y ejecutar comandos remotos mediante SSH. La aplicación busca mejorar la eficiencia operativa y garantizar una interfaz responsive y adaptable.

En resumen, esta herramienta web automatiza tareas de TI, facilita la gestión de entornos y despliegues, y optimiza significativamente la eficiencia operativa.

PALABRAS CLAVE

Ansible, entornos, infraestructura, servidores, proyectos, hosts, despliegues, SSH, backend, frontend, DEVOPS.

ABSTRACT

This final year project aims to develop a web-based business management application that facilitates the control and administration of projects, hosts, and deployments in a secure and efficient environment. Using the authentication system, authenticated users will be able to access various protected functionalities based on user roles.

Specific objectives include creating an intuitive interface to automate application deployment, configuring and maintaining development, testing, and staging environments, and executing remote commands via SSH. The application seeks to enhance operational efficiency and ensure a responsive and adaptable interface.

In summary, this web tool automates IT tasks, simplifies environment and deployment management, and significantly improves operational efficiency.

KEYWORDS

Ansible, environments, infrastructure, servers, projects, hosts, deployments, SSH, backend, frontend, DEVOPS.

ÍNDICE

1 Introducción	1
1.1 Motivación	1
1.2 Objetivos	2
1.3 Estructura de la memoria	3
2 Estado del arte	5
2.1 Arquitectura web	5
2.2 Frontend	6
2.3 Backend	6
2.4 Bases de Datos	7
2.5 APIs	7
2.6 Herramientas Influyentes	8
2.7 Protocolos de ejecución en remoto	8
2.8 Métodos de acceso y gestión de privilegios en máquinas en remoto	9
2.9 Proveedores de Identidad (IDPs)	9
2.10 Protocolos de Autenticación/Autorización	10
3 Análisis y diseño	11
3.1 Alcance	11
3.2 Requisitos del sistema	11
3.2.1 Acceso y navegación en la herramienta web	11
3.2.2 Vista de detalles del usuario	12
3.2.3 Gestión de Proyectos	12
3.2.4 Gestión de Hosts	13
3.2.5 Gestión de Despliegues	14
3.2.6 Otros requisitos	15
3.3 Diseño de la base de datos	16
3.4 Diagrama de la arquitectura	19
4 Implementación	21
4.1 Cronología del desarrollo	21
4.2 Capa del Frontend	22
4.3 Capa del Backend – Datos	25
4.3.1 Manipulación CRUD de colecciones	25

4.3.2	Validación de campos	26
4.4	Capa del Backend – Gestión del despliegue y configuración de las máquinas en remoto	28
4.4.1	Clase BashCommandRunner	28
4.4.2	RunAnsible.sh	30
4.4.3	SetupServer.sh	31
4.4.4	Ansible	33
5	Pruebas	35
5.1	Pruebas de casos de uso y caso real	35
5.2	Pruebas unitarias de la API	38
6	Conclusiones y trabajo futuro	39
Bibliografía		42
Apéndices		43
A	Casos de uso	45
A.1	Casos de uso de un usuario sin sesión y con sesión	45
A.2	Casos de uso del administrador de proyectos	47
A.3	Casos de uso del administrador de hosts	49
A.4	Casos de uso del administrador de despliegues	51
B	Herramienta Git	55
B.1	Introducción a Git	55
B.2	Importancia de Git en el proyecto	55

LISTAS

Lista de códigos

4.1	Enrutado dinamico de componentes	23
4.2	Validación de campos del DespliegueDto	26
4.3	Validación de campos del HostDto	27
4.4	Validación de campos del ProyectoDto	27

Lista de figuras

2.1	Aquitectura web	5
3.1	Diagrama de colecciones	16
3.2	Diagrama de la arquitectura	19
4.1	Esquema de componentes	24
4.2	Secuencia de startDeployment	29
4.3	SetupServer	31
5.1	Inicio de sesión en la herramienta	35
5.2	Creación de un proyecto	36
5.3	Creación de un host	36
5.4	Creación de un despliegue	36
5.5	Ejecución del despliegue	37
5.6	Resultado de ejecución del despliegue	37
5.7	Inspección de los logs del despliegue	37
5.8	Acceso a la aplicación	38
A.1	Casos de uso 1	45
A.2	Casos de uso 2	47
A.3	Casos de uso 2	49
A.4	Casos de uso 3	51

INTRODUCCIÓN

En la era de la digitalización, la automatización de procesos se ha convertido en un componente esencial para el desarrollo y mantenimiento de sistemas informáticos. La gestión eficiente de entornos de desarrollo, pruebas y preproducción es crucial para garantizar la calidad, seguridad y escalabilidad de las aplicaciones [1]. Esta necesidad requiere que administradores de sistemas y desarrolladores gestionen de manera eficiente múltiples servidores y configuraciones complejas, mantengan los equipos actualizados, creen infraestructura virtualizada y entornos en los que desplegar aplicaciones.

1.1. Motivación

La motivación detrás de este proyecto radica en la necesidad de simplificar y agilizar tanto el proceso de despliegue de aplicaciones y configuración de entornos, como de mantenimiento de equipos y creación de infraestructura. En muchas organizaciones, la configuración manual de estas tareas puede ser tediosa y propensa a errores humanos, lo cual no solo consume tiempo valioso, sino que también introduce riesgos que pueden afectar la estabilidad y seguridad de los sistemas. La gestión de entornos y el despliegue de aplicaciones son tareas críticas en el ciclo de vida del desarrollo de software. Las configuraciones incorrectas o inconsistentes pueden llevar a fallos en las aplicaciones, vulnerabilidades de seguridad y una disminución en la productividad del equipo.

Al desarrollar una herramienta web que aproveche las capacidades de Ansible [2] (detallado en 4.4.4), buscamos proporcionar una solución que elimine la complejidad y reduzca los errores humanos asociados con estas tareas. La automatización no solo optimiza el tiempo y los recursos, sino que también garantiza la coherencia, consistencia, repetibilidad y automatización de los procesos, aspectos fundamentales en la gestión de sistemas informáticos modernos. La necesidad de una herramienta que automatice estos procesos surge de varios desafíos comunes: la creciente complejidad de las aplicaciones y sus configuraciones, los limitados tiempos y recursos de los equipos de TI, la minimización de errores humanos y la capacidad de escalar rápidamente y de manera segura a medida que las organizaciones crecen. La automatización facilita el escalado de operaciones sin comprometer la calidad o la seguridad.

1.2. Objetivos

El objetivo principal de este proyecto es desarrollar una aplicación web de gestión empresarial que facilite el control y la administración de proyectos (apps, configuración de entornos, entre otros), hosts y procesos de despliegue en un entorno seguro y eficiente.

La aplicación permitirá a los usuarios acreditados mediante el sistema de autenticación Keycloak de Delonia acceder a diversas funcionalidades, garantizando una experiencia de usuario intuitiva y robusta.

Entre los objetivos específicos se incluyen:

- **Desarrollo de una herramienta web intuitiva:** Crear una interfaz de usuario que permita a los administradores y desarrolladores interactuar fácilmente con Ansible, facilitando la ejecución de comandos y la gestión de configuraciones.
- **Automatización de despliegue de proyectos:** Permitir el despliegue automático de aplicaciones “con un click” en diferentes entornos, asegurando que todas las configuraciones necesarias estén correctamente aplicadas.
- **Configuración y mantenimiento de entornos:** Proporcionar funcionalidades para la configuración automatizada de entornos de desarrollo, pruebas y preproducción, garantizando que todas las dependencias y configuraciones estén en su lugar y mantenerlas actualizadas.
- **Ejecución de comandos remotos:** Facilitar la ejecución de comandos y scripts en servidores remotos mediante el protocolo SSH, utilizando la flexibilidad y potencia de Ansible.
- **Implementación de un diseño Single Page Application (SPA):** Para una navegación fluida y rápida, mejorando la experiencia de usuario.
- **Integración de permisos y roles de usuario:** Para un acceso controlado a las funciones administrativas, asegurando que solo usuarios autorizados puedan realizar determinadas acciones.
- **Provisión de herramientas dinámicas:** Para la creación, edición y eliminación de proyectos, hosts y despliegues, ofreciendo flexibilidad y control a los usuarios.
- **Mejora de la eficiencia operativa:** Reducir el tiempo y el esfuerzo necesarios para la gestión de sistemas, permitiendo a los equipos de TI enfocarse en tareas de mayor valor añadido.
- **Responsividad y adaptabilidad de la interfaz:** Asegurar que todas las operaciones se realicen de manera responsive y que la interfaz sea adaptable a temas oscuros o claros según la configuración del navegador del usuario, mejorando así la accesibilidad y usabilidad del sistema.

En pocas palabras, esta herramienta web proporciona una solución integral para la automatización de tareas de TI, facilitando la gestión de entornos y el despliegue de aplicaciones, y mejorando significativamente la eficiencia operativa.

1.3. Estructura de la memoria

La memoria de este proyecto se organiza en los siguientes apartados:

- **Estado del arte:** Se examinará la situación actual de las diversas tecnologías existentes que buscan alcanzar objetivos similares, además de ofrecer una breve descripción de las características clave de cada una de ellas.
- **Análisis y Diseño:** Se definirá el alcance del proyecto y los requisitos funcionales y no funcionales que el sistema debe cumplir. Asimismo, se describirá la arquitectura del sistema y el enfoque de implementación utilizado tanto para las aplicaciones web como para la base de datos necesaria.
- **Desarrollo:** Se detallarán los principales patrones y metodologías usadas, centrándose en aquellos relacionados con el diseño de la aplicación web.
- **Pruebas y Resultados:** Se presentarán las pruebas realizadas y los resultados obtenidos según los casos de uso.
- **Conclusiones:** Se ofrecerá una breve reflexión sobre lo que ha significado este período de desarrollo del proyecto para el estudiante.
- **Trabajo futuro:** Se describirá el posible trabajo futuro que podría llevarse a cabo para mejorar el proyecto realizado.

ESTADO DEL ARTE

En el ámbito de la ingeniería informática y el desarrollo de herramientas web para el despliegue de proyectos y gestión de entornos, existe una amplia gama de tecnologías, herramientas y prácticas que influyen en el desarrollo del proyecto en cuestión.

A continuación se expone una revisión del estado del arte, abarcando aspectos relacionados con los entornos de desarrollo, bases de datos, APIs, herramientas influyentes, proveedores de identidad (IDPs) y protocolos de autenticación/autorización referentes al desarrollo del proyecto.

2.1. Arquitectura web

En el desarrollo de aplicaciones web, es común utilizar una **arquitectura desacoplada**, donde el **frontend** se encarga de la interfaz de usuario y el **backend** maneja la lógica de negocio, la interacción con la base de datos y las gestiones más críticas como puede ser conexiones con servidores y el resto de la infraestructura. [3]

En este proyecto se ha implementado de esta forma, haciendo uso de un servicio de **frontend** y otro de **backend**, cada uno de forma independiente pudiendo ser escalables ambos tanto vertical como horizontalmente.

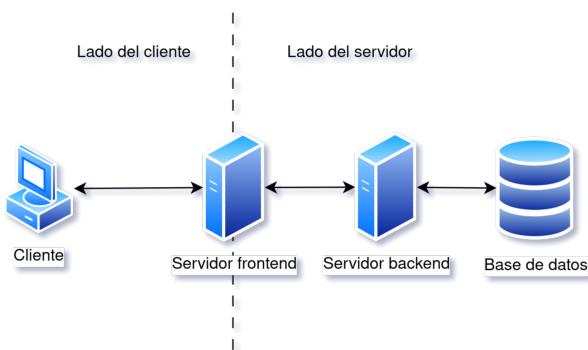


Figura 2.1: Diagrama de desacoplamiento de la arquitectura web.

El **desacoplamiento** claro entre **frontend** y **backend** no solo permite una mejor gestión de los recursos y una mayor flexibilidad a la hora de escalar la aplicación, sino que también facilita la implementación de mejoras y actualizaciones sin afectar significativamente al otro extremo del sistema.

2.2. Frontend

Para el frontend, se suelen utilizar tecnologías como HTML, CSS y JavaScript, junto con frameworks como **React.js**, **Angular.js** o **Vue.js** [4].

React.js es una biblioteca de JavaScript ampliamente utilizada en el desarrollo de interfaces de usuario (UI) para aplicaciones web. Fue desarrollada por Facebook y lanzada como código abierto en 2013. **React.js** se enfoca en la creación de componentes reutilizables que representan diferentes partes de la interfaz de usuario. Actualmente **React.js** es un estándar en el desarrollo de interfaces web por su gran ecosistema de herramientas y bibliotecas. [5]

En cuanto a frameworks, destacan **Angular.js**, desarrollado por Google, ofrece una solución integral con una estructura de aplicación bien definida y herramientas integradas [6]. Y **Vue.js**, que por su parte, es apreciado por su simplicidad y flexibilidad, lo que lo convierte en una excelente opción para proyectos de cualquier escala [7].

Para este proyecto se ha decantado por la utilización de **Vue.js**, por la facilidad de crear interfaces SPA. Este tipo de interfaces permite gestionar la navegación y las transiciones entre diferentes vistas sin recargar la página completa. Solo se recargan los componentes necesarios, mientras que los componentes comunes entre vistas, como las barras de navegación o los pies de página, se mantienen. Esto proporciona una experiencia de usuario fluida y continua. [7]

2.3. Backend

Para el backend, opciones populares hoy en día podemos destacar **Django** (en Python) como framework completo y robusto que ofrece una solución integral con una ORM integrada, administración automática de permisos de acceso y muchas características listas para usar. Es ideal para aplicaciones grandes y complejas [8].

Flask es otro framework en Python pero minimalista y flexible que permite más control sobre los componentes utilizados. Está más enfocado para aplicaciones pequeñas a medianas o cuando se necesita una arquitectura más personalizada [9].

También está **Spring Boot**, un framework robusto para Java que facilita la creación de aplicaciones empresariales. Ofrece un ecosistema completo con integración de seguridad, ORM, y soporte para microservicios. Es ideal para aplicaciones a gran escala y de alto rendimiento [10].

En este caso se ha implementado la lógica del backend en **Spring Boot** por resultar una tecnología novedosa (para mí pues no había desarrollado un back en otro lenguaje que no fuera Python (**Django** y **Flask**)) y por disponer de todos los requisitos necesarios en **Spring Boot** para la realización del proyecto.

2.4. Bases de Datos

Las bases de datos son componentes fundamentales en la mayoría de las aplicaciones web. Para el almacenamiento y gestión de datos, existen opciones tanto **relacionales** como **no relacionales**. Algunas bases de datos relationales populares son **MySQL**, **PostgreSQL**, **Oracle Database**, **DB2** y **SQL Server**, mientras que entre las no relacionales destacan **MongoDB**, **Redis** y **Elasticsearch**.

En este proyecto, se ha decidido utilizar una base de datos **MongoDB** debido a que no se requiere almacenar datos en relaciones complejas ni mantener una consistencia fuerte. MongoDB, con sus colecciones no estructuradas, es ideal para nuestras necesidades. Esta elección es beneficiosa ya que permite almacenar trazas de logs de despliegue, acciones de scripts en remoto, listas de configuraciones y flujos de ejecución de manera eficiente. [11]

Además, se ha decidido usar **MongoDB** ya que es una base de datos que no se ha usado de forma práctica en ninguna asignatura en la carrera y aporta más valor académico al trabajo final, brindando la oportunidad de aprender y aplicar esta nueva tecnología.

2.5. APIs

Las **APIs** (Application Programming Interface) son actualmente un estándar en la industria del desarrollo web son utilizadas para permitir la comunicación entre diferentes componentes de software. Las **APIs RESTful**, que utilizan HTTP para la transferencia de datos, son ampliamente adoptadas en el desarrollo de aplicaciones web debido a su simplicidad y flexibilidad. [12]

Anteriormente era popular el uso de protocolos como **SOAP** (Simple Object Access Protocol) o **CORBA** (Common Object Request Broker Architecture) o **sockets de red** (network sockets), ahora ya en desuso [13].

En el proyecto se ha implementado una **API RESTful** para toda la comunicación entre el frontend y backend mediante diferentes URLs que se componen de forma anidada.

2.6. Herramientas Influyentes

Existen varias herramientas que han tenido un impacto significativo en el desarrollo de proyectos de automatización y gestión de infraestructuras. Entre ellas se destacan:

- **Ansible:** Herramienta de automatización que permite el despliegue, configuración y gestión de sistemas de forma sencilla y eficiente [2].
- **Terraform:** Herramienta para la creación y gestión de infraestructura en múltiples plataformas de nube y servicios de terceros. Usa su propio lenguaje declarativo llamado HashiCorp Configuration Language (HCL) [14].
- **Jenkins:** Herramienta de integración continua que facilita la automatización de tareas relacionadas con la construcción, prueba y despliegue de software [15].

Estas herramientas nos han inspirado para usar lo mejor de cada una de estas tecnologías.

2.7. Protocolos de ejecución en remoto

Para el despliegue de aplicaciones en entornos remotos, estos son los protocolos de acceso remoto más comúnmente usados, incluyendo:

- **SSH (Secure Shell):** Protocolo de red que permite la ejecución de comandos de forma segura en servidores remotos, usa autenticación mediante pares de claves y encripta todos los datos de forma bidireccional mediante algoritmos de cifrado fuertes.

SSH también puede utilizarse para crear túneles seguros para otros protocolos de red. Esto puede ser útil para proteger los datos transmitidos por protocolos que no son seguros por sí mismos [16].

- **Telnet:** Antes de la aparición de SSH, el protocolo más comúnmente utilizado para acceso remoto era Telnet [16]. Telnet permite a los usuarios iniciar sesión en una máquina remota y ejecutar comandos como si estuvieran físicamente presentes.

Sin embargo, Telnet presenta un problema crítico de seguridad: transmite todos los datos, incluidas las contraseñas, en texto plano sin ninguna forma de cifrado.

Esto hace que sea extremadamente vulnerable a interceptaciones y ataques de intermediarios, donde cualquier persona con acceso a la red podría capturar y leer la información transmitida. [17]

SSH fue desarrollado para solucionar los problemas de seguridad asociados con **Telnet**. A diferencia de **Telnet**, **SSH cifra toda la información transmitida**, asegurando que los datos, incluidas las contraseñas, no puedan ser leídos por atacantes interceptando el tráfico si este se transmitiera por canales no seguros. Telnet a día de hoy se sigue utilizando para el acceso a sistemas heredados y comprobar conectividad a un puerto por TCP/IP [17].

Este cifrado bidireccional proporciona una capa de seguridad esencial para la administración de sistemas remotos, haciendo que **SSH** sea la opción preferida en entornos donde la seguridad es una preocupación crítica [16].

2.8. Métodos de acceso y gestión de privilegios en máquinas en remoto

Una cuestión muy importante en este proyecto es la gestión de acceso y los privilegios en las máquinas que figuren en el inventario de servidores en los que ejecutar los scripts.

Existen dos métodos de acceso a una máquina: por **contraseña** o por **clave**.

El uso de contraseñas es el método más común y simple de implementar. La problemática de las contraseñas es que deben ser almacenadas en texto claro o cifradas para poder enviarlas al servidor y autenticarse, lo cual es una mala práctica de seguridad. Si se cifran, necesitarás una forma segura de descifrarlas cuando las necesites. Además, pedir al usuario su contraseña cada vez puede ser molesto y llevar a errores.

El acceso por clave es más seguro (las claves de acceso son resistentes al phishing, a diferencia de las contraseñas) y conveniente para nuestro caso, puesto que solo se va a acceder a los hosts desde una única máquina. En lugar de una contraseña, utilizas un par de claves (una pública y una privada). La clave privada se mantiene segura en tu máquina local, y la clave pública se instala en el servidor remoto. Cuando intentas conectarte, el servidor utiliza la clave pública para verificar tu identidad. Este método elimina la necesidad de recordar o almacenar contraseñas. [18]

En cuanto a la gestión de privilegios, es crucial limitar el acceso a lo que cada usuario necesita para realizar su trabajo. Esto se conoce como el principio de mínimo privilegio. [19]

En la subsección 4.4.3 se detalla cómo se ha resuelto la cuestión.

2.9. Proveedores de Identidad (IDPs)

Los proveedores de identidad son servicios que gestionan la autenticación y autorización de usuarios en aplicaciones web. Algunos proveedores de identidad populares incluyen:

- 1.– **Okta**: Plataforma de gestión de identidad y acceso que proporciona funciones de autenticación y autorización centralizadas [20].
- 2.– **Auth0**: Servicio de autenticación y autorización como servicio (IDaaS) que facilita la implementación de funciones de autenticación en aplicaciones web y móviles [21].
- 3.– **Keycloak**: Software abierto de autenticación y autorización, que permite “Single Sign-On” en diversos servicios aparte de ser compatible con Samba AD [22].

Se ha implementado una autenticación con **Keycloak** ya que es el IDP que usa la empresa y existen módulos de autenticación para nuestro frontend en Vue.js

2.10. Protocolos de Autenticación/Autorización

En el ámbito de la seguridad de aplicaciones, se utilizan diversos protocolos para la autenticación y autorización de usuarios, incluyendo:

- 1.– **OAuth 2.0:** Protocolo de autorización ampliamente utilizado para permitir que aplicaciones de terceros accedan a recursos protegidos en nombre de un usuario [23].
- 2.– **JWT (JSON Web Tokens):** Estándar abierto que define un formato compacto y autocontenido para la transferencia segura de información entre partes como un objeto JSON [24].

Nuestra gestión de la autenticación en la aplicación web se ha implementado bajo el protocolo **OAuth 2.0**, ya que permite implementar roles, haciendo así mucho más sencillo el control de acceso a recursos protegidos o sensibles en la aplicación. También permite emitir **tokens** de acceso con una caducidad específica, lo que hace que el inicio de sesión sea más rápido si se requieren accesos recurrentes durante un corto periodo de tiempo. [23]

Como se puede ver, el estado del arte en el desarrollo de este proyecto abarca una amplia variedad de tecnologías y prácticas, desde la arquitectura de aplicaciones hasta la seguridad y la automatización de infraestructuras.

El proyecto en cuestión se sitúa en este contexto, aprovechando las mejores prácticas y tecnologías disponibles para ofrecer una solución eficiente y escalable.

ANÁLISIS Y DISEÑO

En este capítulo se especifican los diferentes objetivos de cada característica, con los requisitos que debe cumplir tanto funcionales como no funcionales que nuestro sistema debe cumplir, el alcance y arquitectura del proyecto y el diseño de la base de datos utilizada por el sistema.

3.1. Alcance

El sistema software surgido como solución al problema expuesto anteriormente en 1.2, permite a los profesionales técnicos de Delonia Software realizar la gestión de despliegues de proyectos, configuración y mantenimiento de entornos.

3.2. Requisitos del sistema

Esta sección de la documentación detalla los requisitos necesarios para el desarrollo y operación de nuestro proyecto. Incluye tanto los requisitos funcionales, que describen las características y funcionalidades específicas que el sistema debe ofrecer, como los requisitos no funcionales, que especifican los criterios de rendimiento, seguridad, y usabilidad.

3.2.1. Acceso y navegación en la herramienta web

Requisitos funcionales

- RF-1.-** La aplicación web se construirá con un diseño SPA.
- RF-2.-** La aplicación web solo debe dejar acceder a usuarios acreditados mediante el IDP empresarial Keycloak de Delonia pidiendo usuario y contraseña para un token que validará la sesión de forma temporal.
- RF-3.-** La aplicación web no permitirá el registro de nuevos usuarios
- RF-4.-** Siempre que no haya una sesión válida activa, el punto de entrada independientemente de la URL que se introduzca será la pantalla de login del Keycloak.
- RF-5.-** Al autenticarse, el usuario será redirigido a una pantalla de bienvenida con el logo de la empresa, una

descripción de la herramienta y dos accesos:

RF-5.1.- Accesos de redirección al panel de administración

RF-5.2.- Accesos de cerrar sesión

RF-6.- La aplicación web tendrá una barra de navegación en la que incluirá los siguientes contenidos:

RF-6.1.- Acceso a Home que redirige a la pantalla de bienvenida.

RF-6.2.- Acceso a Proyectos que redirige a la vista de gestión del inventario de proyectos.

RF-6.3.- Acceso a Hosts que redirige a la vista de gestión del inventario de hosts.

RF-6.4.- Acceso a Despliegues que redirige a la vista de gestión del inventario de despliegues.

RF-6.5.- Acceso a Logout que cierra la sesión actual y devuelve a la pantalla de login de Keycloak.

Requisitos no funcionales

RNF-1.- La interfaz de navegación contará con tema oscuro o claro que se adapte a la configuración del navegador del usuario.

RNF-2.- Uso del IDP empresarial Keycloak de Delonia para la autenticación de usuarios.

3.2.2. Vista de detalles del usuario

Requisitos funcionales

RF-1.- Se mostrará el nombre de usuario y los roles que tiene la sesión del usuario actual

Requisitos no funcionales

RNF-1.- Cada rol se mostrará en una etiqueta independiente

3.2.3. Gestión de Proyectos

Requisitos funcionales

RF-1.- Se mostrará un botón de “Crear nuevo proyecto” que solo estará activo si la sesión tiene el permiso de “AdministrarProyectos” y que lanzará un modal con los siguientes campos a llenar que valida de forma dinámica:

RF-1.1.- Nombre: una cadena de 30 caracteres máximo, se notificará si el campo está vacío con un mensaje.

RF-1.2.- Descripción: una cadena de 50 caracteres máximo, se notificará si el campo está vacío con un mensaje.

RF-1.3.- Repositorio: una cadena de 50 caracteres máximo, se notificará si el campo está vacío con un mensaje. La url se construye a partir de un prefijo común entre ellos “<https://scm.delonia.com/scm-git/>”, se trata de el git empresarial.

RF-1.4.- Tipos de despliegues: Una lista de etiquetas de 20 caracteres como máximo, se notificará si la lista está vacía.

RF-2.- El modal de “Crear nuevo proyecto” cuenta con un botón de “Crear” que solo está activo si los campos del

modal cumplen los requisitos establecidos. Este botón añade el proyecto a la lista y cierra el modal.

RF-3.- Los proyectos se dispondrán en una tabla en la que se muestran los campos según se vayan añadiendo en la base de datos.

RF-4.- Se mostrará un botón de “Editar” por cada proyecto que solo estará activo si la sesión tiene el permiso de “AdministrarProyectos” y que lanzará un modal con los siguientes campos anteriormente mencionados con la posibilidad de editarlos: Nombre, Descripción, Repositorio y Tipos de despliegue. Estos son validados de forma dinámica al igual que al crear un nuevo proyecto.

RF-5.- Se mostrará un botón de “Eliminar” por cada proyecto que solo estará activo si la sesión tiene el permiso de “AdministrarProyectos” y que lanzará un modal en el que se confirma la decisión de eliminar el proyecto. Puede confirmar (eliminar el proyecto y cerrar el modal) o cancelar (solo cerrar el modal).

RF-6.- Se mostrará un botón de “Refrescar” que actualiza la información de los proyectos.

Requisitos no funcionales

RNF-1.- Los tipos de despliegue se muestran como etiquetas por cada elemento en la lista.

RNF-2.- El botón editar tendrá color gris y el de eliminar amarillo para que sea llamativo a la vista.

RNF-3.- El botón de “Refrescar” tendrá un símbolo de flechas en círculo para facilitar la comprensión del botón.

3.2.4. Gestión de Hosts

Requisitos funcionales

RF-1.- Se mostrará un botón de “Crear nuevo host” que solo estará activo si la sesión tiene el permiso de “AdministrarHosts” y que lanzará un modal con los siguientes campos a llenar que valida de forma dinámica:

RF-1.1.- Nombre: una cadena de 30 caracteres máximo, se notificará si el campo está vacío con un mensaje.

RF-1.2.- IP: una cadena de 16 caracteres máximo, se notificará si el campo está vacío con un mensaje.

RF-1.3.- Usuario: una cadena de 30 caracteres máximo, se notificará si el campo está vacío con un mensaje.

RF-1.4.- Password: Una lista de etiquetas de 30 caracteres como máximo, se notificará si la lista está vacía. Este campo no se almacenará en la base de datos.

RF-2.- El modal de “Crear nuevo host” cuenta con un botón de “Crear” que solo está activo si los campos del modal cumplen los requisitos establecidos. Este botón añade el host a la lista y cierra el modal.

RF-3.- El modal de “Crear nuevo host” cuenta con un botón de “Probar conexión SSH” que solo está activo si los campos de IP, Usuario y Password del modal cumplen los requisitos establecidos. Este botón hace una comprobación de las credenciales contra la máquina, y manda un mensaje por la pantalla si se ha podido realizar la conexión.

RF-4.- Los hosts se dispondrán en una tabla en la que se muestran los campos según se vayan añadiendo en la base de datos.

RF-5.- Se mostrará un botón de “Editar” por cada host que solo estará activo si la sesión tiene el permiso de “AdministrarHosts” y que lanzará un modal con los siguientes campos anteriormente mencionados con la posibilidad de editarlos: Nombre, IP y Usuario. Estos son validados de forma dinámica al igual que al crear un nuevo host.

RF-6.– Se mostrará un botón de “Eliminar” por cada host que solo estará activo si la sesión tiene el permiso de “AdministrarHosts” y que lanzará un modal en el que se confirma la decisión de eliminar el host. Puede confirmar (eliminar el host y cerrar el modal) o cancelar (solo cerrar el modal).

RF-7.– Se mostrará un botón de “Refrescar” que actualiza la información de los hosts.

RF-8.– En la tabla de los datos de los Hosts, hay una columna de “Estado”. Esta columna se actualiza de forma asíncrona cuando se carga la página y muestra si los hosts son accesibles a través del backend (mediante SSH). El valor se muestra en una etiqueta.

Requisitos no funcionales

RNF-1.– Los tipos de despliegue se muestran como etiquetas por cada elemento en la lista.

RNF-2.– El botón editar tendrá color gris y el de eliminar amarillo para que sea llamativo a la vista.

RNF-3.– El botón de “Refrescar” tendrá un símbolo de flechas en círculo para facilitar la comprensión del botón.

3.2.5. Gestión de Despliegues

Requisitos funcionales

RF-1.– Se mostrará un botón de “Crear nuevo despliegue” que solo estará activo si la sesión tiene el permiso de “AdministrarDespliegues” y que lanzará un modal con los siguientes campos a llenar que valida de forma dinámica:

RF-1.1.– Nombre: una cadena de 30 caracteres máximo, se notificará si el campo está vacío con un mensaje.

RF-1.2.– Grupo: una cadena de 30 caracteres máximo, se notificará si el campo está vacío con un mensaje.

RF-1.3.– Descripción: una cadena de 50 caracteres máximo, se notificará si el campo está vacío con un mensaje.

RF-1.4.– Host: Una lista desplegable del inventario de hosts del cual se selecciona uno.

RF-1.5.– Proyecto: Una lista desplegable del inventario de proyectos del cual se selecciona uno.

RF-1.6.– Tipo: Una lista desplegable de los tipos de despliegues para el proyecto seleccionado anteriormente. La lista se actualiza de forma dinámica al cambiar el proyecto.

RF-1.7.– Estado: una cadena de 30 caracteres máximo, se notificará si el campo está vacío con un mensaje.

RF-2.– El modal de “Crear nuevo despliegue” cuenta con un botón de “Crear” que solo está activo si los campos del modal cumplen los requisitos establecidos. Este botón añade el despliegue a la lista y cierra el modal.

RF-3.– Los despliegues se dispondrán en una tabla en la que se muestran los campos según se vayan añadiendo en la base de datos.

RF-4.– Se mostrará un botón de “Editar” por cada despliegue que solo estará activo si la sesión tiene el permiso de “AdministrarDespliegues” y que lanzará un modal con los siguientes campos anteriormente mencionados con la posibilidad de editarlos: Nombre, Descripción, Repositorio y Tipos de despliegue. Estos son validados de forma dinámica al igual que al crear un nuevo despliegue.

RF-5.– Se mostrará un botón de “Eliminar” por cada despliegue que solo estará activo si la sesión tiene el permiso de “AdministrarDespliegues” y que lanzará un modal en el que se confirma la decisión de eliminar el despliegue. Puede confirmar (eliminar el despliegue y cerrar el modal) o cancelar (solo cerrar el modal).

RF-6.– Se mostrará un botón de “Ver logs” por cada despliegue que solo estará activo si la sesión tiene el permiso de “AdministrarDespliegues” y el campo de logs no está vacío. Lanzará un modal en el que se visualizará la última traza de ejecución en con un scroll vertical.

RF-7.– Se mostrará un botón de “Desplegar” por cada despliegue que solo estará activo si la sesión tiene el permiso de “AdministrarDespliegues” y si los campos de proyecto y host no son nulos. Este botón comenzará el proceso de despliegue del proyecto, y hasta que no acabé estará desactivado.

RF-8.– Tras terminar el despliegue, los campos de “Fecha”, “Desplegado por” y “Estado” se actualizan al momento actual, usuario que inició el despliegue y “Éxito” o “Fallo” (según la ejecución) respectivamente.

RF-9.– El botón de “Desplegar” cambiará a “Reintentar” tras el primer intento de despliegue.

RF-10.– Se mostrará un botón de “Refrescar” que actualiza la información de los despliegues.

Requisitos no funcionales

RF-1.– Los tipos de despliegue se muestran como etiquetas por cada elemento en la lista.

RF-2.– El botón editar tendrá color gris y el de eliminar amarillo para que sea llamativo a la vista.

RF-3.– El botón de “Refrescar” tendrá un símbolo de flechas en círculo para facilitar la comprensión del botón.

RF-4.– El botón de “Ver logs” tendrá un símbolo de un libro abierto para facilitar la comprensión del botón.

RF-5.– El botón de “Desplegar” tendrá un símbolo de “play” para facilitar la comprensión del botón.

3.2.6. Otros requisitos

Requisitos funcionales

RF-1.– Siempre que haya una petición al backend, la respuesta se mostrará en forma de mensaje por pantalla con información de la ejecución (éxito o error y mensaje de error en su defecto)

Requisitos no funcionales

RNF-1.– La paleta de colores será de los del logo de la empresa: blanco, negro y azul celeste.

RNF-2.– El backend guardará las claves privadas en el directorio ./keys dentro del proyecto.

RNF-3.– El backend clonará los repositorios de proyectos para su despliegue en ./repos dentro del proyecto.

Casos de uso

Se han diseñado unos casos de uso que consisten en las operaciones de inicio de sesión y gestión de proyectos, hosts y despliegues. Se detallan el apéndice A.

3.3. Diseño de la base de datos

Como ya se comentó previamente, **MongoDB** es una base de datos NoSQL orientada a documentos, lo que significa que los datos se almacenan en documentos **JSON** o **BSON** (Binary JSON) [11].

A continuación se presenta el diseño de la base de datos para la aplicación web descrita, incluyendo las colecciones y un esquema de las colecciones y como se relacionan entre ellos (aunque sea una base de datos no-relacional, se han esquematizado como si las colecciones tuvieran relaciones de agragación para expresar la lógicas entre las colecciones, aunque estas referencias se hagan de forma manual).

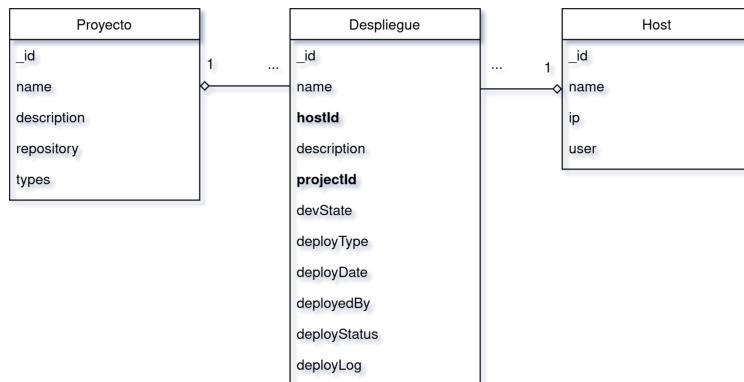


Figura 3.1: Diagrama de relaciones lógicas entre las colecciones de la base de datos.

Colecciones MongoDB:

- 1.- **Proyectos**
- 2.- **Hosts**
- 3.- **Despliegues**

Esquemas de las Colecciones:

Proyectos Collection

- - Nombre de la colección: *Proyectos*
 - Esquema:

```
{
  "_id": ObjectId,
  "name": String,
  "description": String,
  "repository": String,
  "types": [String]
}
```

Descripción de los campos:

- ◊ **name**: El nombre del proyecto. Este campo es una cadena de caracteres y debe ser único para cada proyecto.
- ◊ **description**: Una breve descripción del proyecto. Este campo es una cadena de caracteres que proporciona información adicional sobre el proyecto.
- ◊ **repository**: La URL del repositorio asociado con el proyecto. Este campo es una cadena de caracteres y almacena la dirección del repositorio de código fuente.
- ◊ **types**: Una lista de cadenas de caracteres que describe los diferentes tipos de despliegue asociados con el proyecto. Cada tipo de despliegue se usará para diferenciar flujos de ejecución en el despliegue.

• Hosts Collection

- Nombre de la colección: *Hosts*
- Esquema:

```
{
    "_id": ObjectId,
    "name": String,
    "ip": String,
    "user": String
}
```

Descripción de los campos:

- ◊ **name**: El nombre del host. Este campo es una cadena de caracteres y debe ser único para cada host.
- ◊ **ip**: La dirección IP del host. Este campo es una cadena de caracteres y almacena la dirección IP del host.
- ◊ **user**: El nombre de usuario utilizado para acceder al host. Este campo es una cadena de caracteres y se utiliza en las operaciones de autenticación y conexión.

• Despliegues Collection

- Nombre de la colección: *Despliegues*
- Esquema:

```
{
    "_id": ObjectId,
    "name": String,
    "hostId": ObjectId,
    "description": String,
    "projectId": ObjectId,
    "devState": String,
    "deployType": String,
```

```
        "deployDate": Date,  
        "deployedBy": String,  
        "deployStatus": String,  
        "deployLog": String  
    }
```

Descripción de los campos:

- ◊ **name**: El nombre del despliegue. Este campo es una cadena de caracteres y debe ser único para cada despliegue.
- ◊ **hostId**: Referencia al identificador único del host en la colección Hosts. Este campo es un ObjectId que establece una relación entre el despliegue y el host en el que se realiza.
- ◊ **description**: Una breve descripción del despliegue. Este campo es una cadena de caracteres que proporciona información adicional sobre el despliegue.
- ◊ **projectId**: Referencia al identificador único del proyecto en la colección Proyectos. Este campo es un ObjectId que establece una relación entre el despliegue y el proyecto al que pertenece.
- ◊ **devState**: El estado del desarrollo. Este campo es una cadena de caracteres que describe el estado actual del despliegue desde la perspectiva del desarrollo.
- ◊ **deployType**: El tipo de despliegue. Este campo es una cadena de caracteres y corresponde a uno de los tipos de despliegue definidos en la colección Proyectos.
- ◊ **deployDate**: La fecha en la que se realizó el despliegue. Este campo es de tipo Date y almacena la fecha y hora del despliegue.
- ◊ **deployedBy**: El nombre de usuario que realizó el último despliegue. Este campo es una cadena de caracteres que indica quién realizó el despliegue.
- ◊ **deployStatus**: El estado del despliegue. Este campo es una cadena de caracteres que describe si el despliegue fue exitoso, fallido, etc.
- ◊ **deployLog**: Los registros del despliegue. Este campo es una cadena de caracteres que almacena los logs generados durante el despliegue, proporcionando información detallada sobre el proceso y cualquier error ocurrido.

Generación de _id:

Todos los _id son generados de forma auto-incrementada en el backend.

3.4. Diagrama de la arquitectura

Arquitectura ideal del servicio: servicios frontend, backend y base de datos en máquinas dedicadas dentro de la red empresarial con un único punto de acceso en el servidor del frontend.

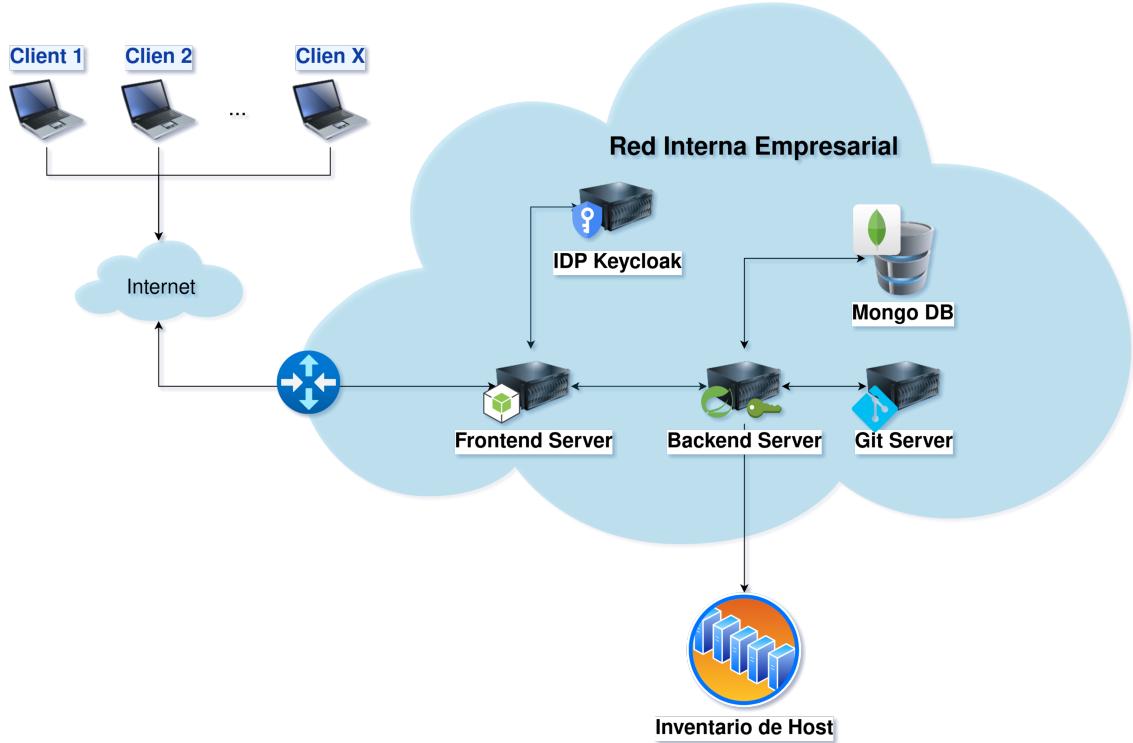


Figura 3.2: Diagrama de la arquitectura propuesta.

IMPLEMENTACIÓN

El proyecto se ha ido implementado de forma “capa por capa”, lo que ha permitido abordar cada aspecto de manera organizada. Este enfoque ha facilitado la identificación y resolución de problemas en etapas tempranas, asegurando que cada capa funcione correctamente antes de avanzar a la siguiente.

4.1. Cronología del desarrollo

Inicialmente, se construyó la **capa de presentación** (frontend), incluyendo la integración de seguridad, utilizando el sistema de autenticación Keycloak de Delonia para gestionar permisos y roles de usuario, garantizando un acceso controlado y seguro a las funciones administrativas.

Una vez ya estaba probada la seguridad, se fue implementando un diseño SPA para garantizar una experiencia de usuario fluida y responsiva. En esta etapa, se prestó especial atención a la usabilidad y accesibilidad, haciendo uso de librerías de componentes agradables para la vista del usuario final y que resulten intuitivas. Se recurrió a componentes de la librería opensource de Naive UI [25], la cual está específicamente diseñada para ser usada en Vue.js y tener un aspecto elegante y sobrio. Para los iconos de botones y actuadores, se han usado iconos de FontAwesome [26], la cual es otra librería open source que cuenta con una amplia gama de iconos de uso gratuito.

De forma paralela, se comenzó el desarrollo de la **capa de lógica de negocio** (backend), integrando las funcionalidades clave y asegurando que los procesos esenciales se ejecuten sin inconvenientes y las operaciones contra la base de datos eran consistentes.

Este enfoque por capas no solo ha mejorado la eficiencia del desarrollo, sino que también ha permitido una mejor gestión del proyecto, facilitando pruebas y modificaciones en cada etapa de manera independiente.

En esta sección se detalla el desarrollo de los componentes de nuestra aplicación web, incluyendo el diseño de la arquitectura del sistema, el uso de patrones de diseño y modelos, así como los desafíos enfrentados y las soluciones adoptadas.

4.2. Capa del Frontend

El frontend ha sido implementado con el framework **Vue.js**, a continuación se desarrollan las prácticas y patrones de programación que he seguido durante el desarrollo del frontend:

Patrón MVVM

Vue.js implementa el patrón **modelo–vista–Vista-modelo** (MVVM), un patrón de arquitectura de software que trata de desacoplar la interfaz de usuario de la parte lógica de la aplicación. [27]

Este patrón se compone de los siguientes elementos:

- **El modelo**, que representa la capa de datos y/o la lógica de negocio. El modelo contiene la información, pero nunca las acciones o servicios que la manipulan, no teniendo dependencia alguna con la vista.
- **La vista**, que representa la información a través de los elementos visuales que la componen.
- **Modelo de vista**, este elemento hace de intermediario entre el modelo y la vista. Contiene toda la lógica de presentación y se comporta como una abstracción de la interfaz.

Con este patrón de programación hemos diseñado los siguientes componentes:

- **Home.vue**: pantalla “Home” de la aplicación.
- **Dashboard.vue**: marco principal de la interfaz, cuenta con un componente tipo Navegación y un “router-view” el cual cambia de componente según se vaya cambiando la URL.
- **Navegación.vue**: el componente siempre está presente en la navegación de la interfaz. Cuenta con el logo de la empresa y botones de redirección para cambiar de componente del “router-view”.
- **Componentes que se montan en “router-view”:**
 - **Profile.vue**: componente de información referente al usuario de la sesión actual.
 - **Proyectos.vue**: componente para la visualización y gestión de los proyectos.
 - **Hosts.vue**: componente para la visualización y gestión de los hosts.
 - **Despliegues.vue**: componente para la visualización y gestión de los despliegues.

Modelos SPA

Una **Single Page Application** es una aplicación web que carga una sola página HTML y actualiza dinámicamente el contenido a medida que el usuario interactúa con la aplicación, sin necesidad de recargar toda la página [7].

Esto proporciona una **experiencia más rápida y fluida**, similar a la de una aplicación de escritorio. En Vue una aplicación web SPA puede implementar navegabilidad en páginas lógicas dentro de la aplicación. En este caso contamos con URLs lógicas como /proyectos, /hosts... que cambian según varía el componente <router-view> [28].

Código 4.1: En esta figura se muestra el enrutado de URLs mediante un mapeo de URL-componente para el children (<router-view>) en Dashboard..

```
1 const router = createRouter({
2   history: createWebHistory(import.meta.env.BASE_URL),
3   routes: [
4     {
5       path: '/dashboard',
6       name: 'dashboard',
7       component: Dashboard,
8       children: [
9         {
10          path: '/profile',
11          name: 'profile',
12          component: Profile
13        },
14        {
15          path: '/projects',
16          name: 'projects',
17          component: ProyectosList
18        },
19        {
20          path: '/hosts',
21          name: 'hosts',
22          component: HostsList
23        },
24        {
25          path: '/despliegues',
26          name: 'despliegues',
27          component: Despliegues
28        }
29      ],
30    },
31    {
32      path: '/',
33      name: 'home',
34      component: Home
35    }
36  ]
37})
```

Para la autenticación y autorización del usuario, se ha configurado un fichero TypeScript para hacer llamadas a la API de Keycloak y unas funciones específicas para poder obtener un token de sesión temporal para acceder a la aplicación web, otra función para obtener los roles y datos asociados al usuario que responde al token de sesión del navegador y otra para hacer logout de la aplicación. [29]

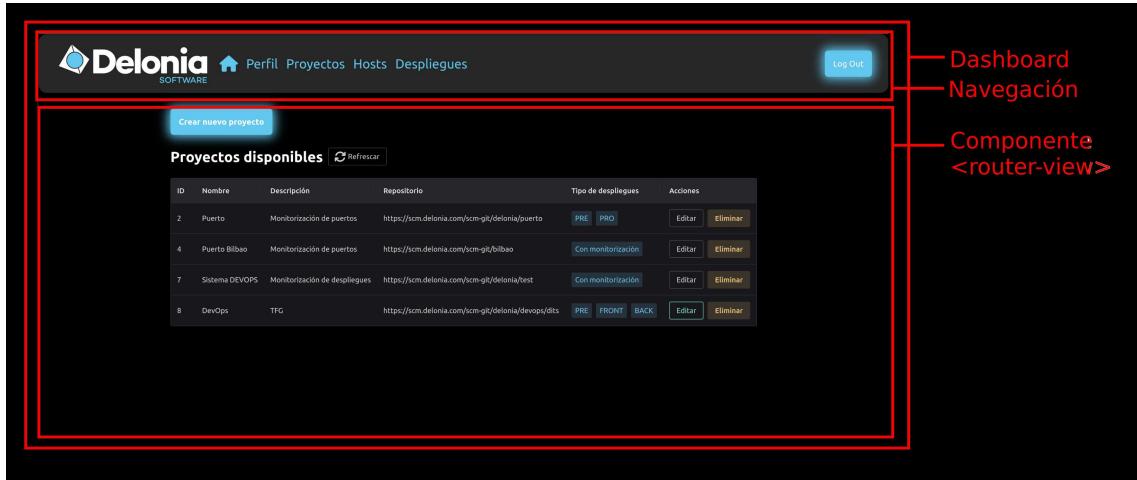


Figura 4.1: Esquema de componentes de la vista en el navegador.

4.3. Capa del Backend – Datos

Para el backend se ha usado **Spring Boot** junto con una base de datos **MongoDB** para el almacenamiento de datos. La elección de **Spring Boot** se ha debido a que es un framework que facilita la creación de aplicaciones Java, permitiendo la creación de aplicaciones de forma rápida y sencilla. Además, **Spring Boot** es un framework muy popular en la industria y cuenta con una amplia documentación y una gran comunidad de desarrolladores.

En cuanto a la base de datos, se ha decidido usar **MongoDB** debido a que no se requiere almacenar datos en relaciones complejas ni mantener una consistencia fuerte. **MongoDB**, con sus colecciones no estructuradas, es ideal para nuestras necesidades. Esta elección es beneficiosa como ya comentamos en 2.4 por permitir almacenar trazas de logs de despliegue, acciones de scripts en remoto, listas de configuraciones y flujos de ejecución de manera eficiente.

4.3.1. Manipulación CRUD de colecciones

La manipulación de los datos se ha realizado usando **DTOs** (Data Transfer Objects) para cada una de las colecciones de nuestra base de datos, utilizando la librería de MongoDB adaptada a Spring Boot. Para cada DTO se ha creado además un **controller**, un **entity**, un **repository** y un **service**. [30] A continuación se presenta la justificación de cada una de estas partes:

Controller:

- **Propósito:** Maneja las solicitudes HTTP y actúa como intermediario entre la vista (interfaz de usuario) y la capa de servicio.
- **Responsabilidad:** Recibir las solicitudes de los clientes, procesarlas (validarlas) y devolver las respuestas adecuadas. Utiliza los servicios para realizar la lógica de negocio necesaria.
- **Por qué:** Separa la lógica de presentación de la lógica de negocio, facilitando la gestión y mantenimiento de las rutas y puntos de acceso de la aplicación.

Entity:

- **Propósito:** Representa una tabla en la base de datos y es utilizada por JPA (Java Persistence API) para mapear los datos a objetos Java.
- **Responsabilidad:** Definir la estructura de los datos que se almacenarán en la base de datos.
- **Por qué:** Separa la representación de datos y facilita la persistencia y recuperación de datos a través de JPA.

Repository:

- **Propósito:** Proveer una abstracción sobre las operaciones de acceso a la base de datos (CRUD: Create, Read, Update, Delete).
- **Responsabilidad:** Interactuar con la base de datos usando las entidades definidas y realizar operaciones de persistencia.

- **Por qué:** Encapsula la lógica de acceso a los datos y separa esta lógica de las capas superiores (servicios y controladores), facilitando el uso de principios como la inyección de dependencias.

Service:

- **Propósito:** Contiene la lógica de negocio y actúa como una capa intermedia entre el controlador y el repositorio.
- **Responsabilidad:** Ejecutar operaciones de negocio, interactuar con los repositorios para gestionar los datos y procesar la lógica más compleja.
- **Por qué:** Separa la lógica de negocio de las capas de presentación y persistencia, permitiendo una mayor modularidad y facilitando las pruebas unitarias.

4.3.2. Validación de campos

Se ha implementado una validación de campos completa usando la librería `jakarta.validation` [31] en todos los campos necesarios de los DTOs con sus correspondientes errores de validación. Estas son las etiquetas de validación de cada atributo:

Código 4.2: Validación de campos del DespliegueDto.

```

1 public class DespliegueDto {
2
3     @NotBlank(message = "Group_is_mandatory")
4     private String group;
5
6     @NotBlank(message = "Name_is_mandatory")
7     private String name;
8
9     @NotNull(message = "Host_is_mandatory")
10    private int hostId;
11
12    private String description;
13
14    @NotNull(message = "Proyecto_is_mandatory")
15    private int projectId;
16
17    @NotBlank(message = "DevState_is_mandatory")
18    private String devState;
19
20    ...
21 }
```

Código 4.3: Validación de campos del HostDto.

```

1  public class HostDto {
2      @NotBlank(message = "Name_is_mandatory")
3      @NotNull(message = "Name_is_mandatory")
4      private String name;
5
6      @NotBlank(message = "IP_is_mandatory")
7      @NotNull(message = "IP_is_mandatory")
8      private String ip;
9
10     @NotBlank(message = "User_is_mandatory")
11     @NotNull(message = "User_is_mandatory")
12     private String user;
13
14     @NotBlank(message = "Password_is_mandatory")
15     @NotNull(message = "Password_is_mandatory")
16     private String password;
17
18     ...
19 }
```

Código 4.4: Validación de campos del ProyectoDto.

```

1  public class ProyectoDto {
2
3      @NotBlank(message = "Name_is_mandatory")
4      @NotNull(message = "Name_is_mandatory")
5      private String name;
6
7      private String description;
8
9      @NotBlank(message = "Repository_is_mandatory")
10     @NotNull(message = "Repository_is_mandatory")
11     private String repository;
12
13     @NotNull(message = "At_least_one_type_is_mandatory")
14     @Size(min = 1, message = "At_least_one_type_is_mandatory")
15     private List< @NotBlank(message = "Type_can't_be_empty") String> types;
16
17     ...
18 }
```

4.4. Capa del Backend – Gestión del despliegue y configuración de las máquinas en remoto

Se han implementado la clase **BashCommandRunner** que cuenta con funciones críticas dentro de la operativa de configuración de las máquinas y ejecución de despliegue de proyectos.

Además se ha hecho uso de scripts de **Bash** para la ejecución de comandos, los cuales los ejecuta la clase **BashCommandRunner**.

4.4.1. Clase BashCommandRunner

Tiene la responsabilidad de gestionar la ejecución de comandos bash y procesos de despliegue de aplicaciones a través de scripts y herramientas como **Git** (detallado en el apéndice B) y **Ansible** (detallado en la sección 4.4.4).

Atributos:

- **keypath:** Ruta a la ubicación de las claves SSH necesarias para la autenticación.

Métodos:

- **runCommand(String command, String ip, String username, String password)**

Este método ejecuta una función bash concreta del fichero `SetupServer.sh` en un proceso separado. El comando se construye concatenando varios parámetros y se ejecuta usando `ProcessBuilder`. Se capturan y muestran las salidas del comando y se retorna el estado de salida del proceso.

- **Parámetros:**

- **command:** Comando a ejecutar.
- **ip:** Dirección IP de la máquina.
- **username:** Usuario de la máquina.
- **password:** Contraseña de la máquina.

- **Retorno:** Estado de salida del proceso (int).

- **startDeployment(String ip, String user, String repo, String mode)**

Este método y función de entrada que gestiona el inicio del proceso de despliegue.

Sigue los siguientes pasos y comprobaciones:

- 1.– Verifica el acceso a la máquina: Llama a `runCommand` con el comando `check_key` para asegurar que se puede acceder a la máquina destino.
- 2.– Verifica el acceso al repositorio: Llama a `isRepoAccessible` para comprobar si el repositorio es accesible con las credenciales de Git proporcionadas (especificadas en un fichero de variables de entorno).
- 3.– Clona el repositorio: Si el acceso es exitoso, clona el repositorio en un directorio local.
- 4.– Inicia Ansible: Llama a `startAnsible` para iniciar el proceso de despliegue usando Ansible.

- **Parámetros:**

- ◊ **ip:** Dirección IP de la máquina destino.
- ◊ **user:** Usuario para la conexión.
- ◊ **repo:** Nombre del repositorio a clonar.
- ◊ **mode:** Modo de despliegue.

- **Retorno:** Una tupla (Tuple<Integer, String>) con el estado de salida y un mensaje detallado del proceso (este “mensaje detallado” posteriormente se inserta en el campo de “logs” del despliegue).

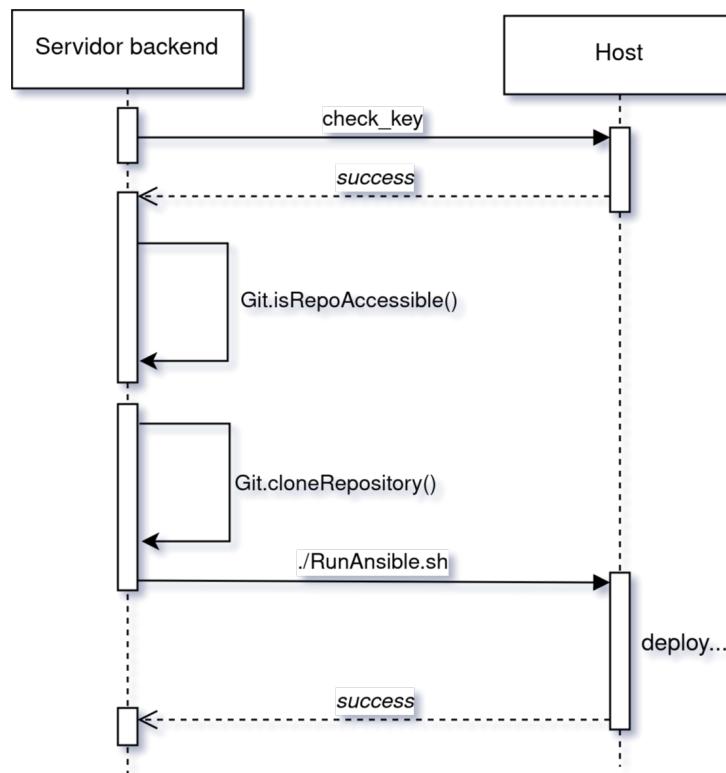


Figura 4.2: En este diagrama de secuencia se esquematiza la función startDeployment

- `isRepoAccessible(String repoUrl)`

Verifica si el repositorio remoto es accesible utilizando las credenciales predeterminadas. Usa la librería JGit para realizar esta comprobación.

- **Parámetros:**

- ◊ **repoUrl:** URL del repositorio a comprobar.

- **Retorno:** boolean indicando si el repositorio es accesible.

- `startAnsible(StringBuilder message, String repo, File cloneDir, String mode, String ip, String user)`

Este método construye y ejecuta el comando para iniciar un playbook de Ansible, pasando los parámetros necesarios como la IP de la máquina, la ruta de la clave SSH, el usuario, el modo de despliegue, la URL del repositorio y la ruta del archivo Setup.yaml.

- **Parámetros:**

- **message:** StringBuilder para acumular mensajes del proceso.
- **repo:** Nombre del repositorio.
- **cloneDir:** Directorio donde se ha clonado el repositorio.
- **mode:** Modo de despliegue.
- **ip:** Dirección IP de la máquina destino.
- **user:** Usuario para la conexión.

- **Retorno:** Una tupla (`Tuple<Integer, String>`) con el estado de salida y un mensaje detallado del proceso.

4.4.2. RunAnsible.sh

Este script se encarga de ejecutar el comando de **Ansible** formado por el **BashCommandRunner**.

En 4.4.4 se detalla la invocación del comando de Ansible, su funcionalidad y su uso en el proyecto.

4.4.3. SetupServer.sh

Este script bash tiene la funcionalidad de **configurar un servidor objetivo**: verificar la conexión usando autenticación por contraseña, hacer la copia de clave y verificar la conexión usando autenticación por clave.

Este script implementa y configura el acceso con clave, la cual solo requiere de un usuario y contraseña de acceso una primera vez, antes de copiar la clave pública, permitiendo posteriores conexiones sin necesidad de una contraseña.

También en esta primera conexión se genera un fichero de configuración en /etc/sudoers.d/ para permitir que el usuario pueda ejecutar tareas con privilegios sin necesidad de introducir una contraseña (**passwordless sudo**).

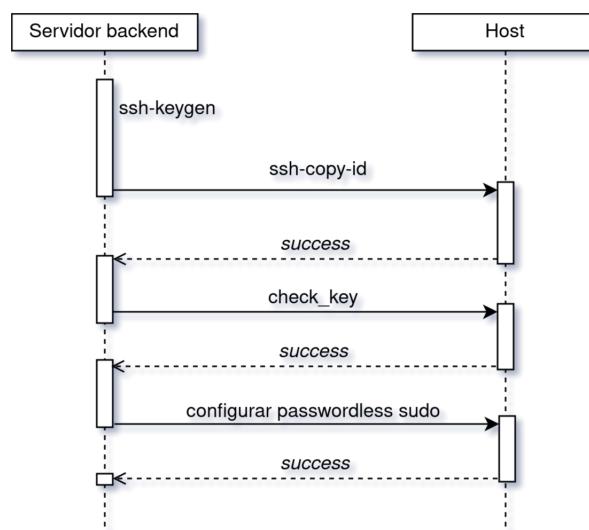


Figura 4.3: Diagrama de secuencia de la configuración de un host para futuros accesos por clave.

A continuación se detalla cada una de sus funciones:

Métodos:

- `setup_server`

Configura el servidor objetivo para permitir la autenticación sin contraseña mediante claves SSH.

- **Parámetros:**

- **username:** Nombre de usuario para la conexión SSH.
- **password:** Contraseña del usuario.
- **host:** Dirección IP o nombre de host del servidor remoto.

- **Operaciones:**

- 1.– Crea el directorio `keys` si no existía ya.
- 2.– Genera un par de claves RSA si no existen para el `username` que recibe.
- 3.– Copia la clave pública al servidor remoto usando `ssh-copy-id`.

4.– Configura el servidor remoto para permitir la autenticación sin contraseña mediante sudo (passwordless sudo).

- **check_password**

Verifica la conexión al servidor remoto usando autenticación por contraseña.

- **Parámetros:**

- **username:** Nombre de usuario para la conexión SSH.
 - **password:** Contraseña del usuario.
 - **host:** Dirección IP o nombre de host del servidor remoto.

- **Operaciones:**

- 1.– Usa `sshpass` para intentar una conexión SSH con contraseña.
 - 2.– Verifica el éxito de la conexión y muestra un mensaje adecuado.

- **check_key**

Verifica la conexión al servidor remoto usando autenticación por clave SSH.

- **Parámetros:**

- **username:** Nombre de usuario para la conexión SSH.
 - **host:** Dirección IP o nombre de host del servidor remoto.

- **Operaciones:**

- 1.– Usa una clave SSH para intentar una conexión sin contraseña.
 - 2.– Verifica el éxito de la conexión y muestra un mensaje adecuado.

- Función de entrada (`main script`)

Verifica que se hayan proporcionado cuatro argumentos, signa los argumentos a variables y llama a la función correspondiente según el primer argumento.

- **Parámetros:**

- **function:** Función a ejecutar (`setup_server`, `check_password`, `check_key`).
 - **host:** Dirección IP o nombre de host del servidor remoto.
 - **username:** Nombre de usuario para la conexión SSH.
 - **password:** Contraseña del usuario.

- **Operaciones:**

- 1.– Verifica que se hayan proporcionado cuatro argumentos.
 - 2.– Asigna los argumentos a variables.
 - 3.– Llama a la función correspondiente según el primer argumento.

Otros endpoints

También se ha configurado una clase específica “OtherAPIs” que maneja las solicitudes HTTP referentes a comprobaciones de estados de máquinas en remoto:

- `/testssh/password`

Este endpoint comprueba la conexión a una IP con las credenciales proporcionadas mediante **CommandRunner** (`check_password`).

- `/testssh/key`

Este otro endpoint comprueba la conexión a una IP con un usuario proporcionado mediante **CommandRunner** (`check_key`).

Todos los datos (IPs, usuarios y contraseñas) se envían en el **cuerpo de la petición, nunca en la cabecera**.

4.4.4. Ansible

Ansible ha sido la herramienta elegida para realizar la ejecución de los scripts de los despliegues contra las máquinas de nuestro inventario de hosts. Es una herramienta diseñada para la configuración de sistemas, despliegue de aplicaciones y administración de redes. [13]

¿Cómo funciona?

Ansible utiliza archivos de configuración en formato **YAML** llamados **playbooks**. En estos playbooks se declaran las tareas y procesos que van a ser ejecutados.

No requiere la instalación de software adicional en los nodos gestionados; **se comunica a través de SSH**, lo cual le proporciona **robustez** y **confiabilidad**.

Ansible usa inventarios en los que se definen los hosts y grupos de hosts que serán gestionados. En este caso, solo hemos contemplado la ejecución en un único host, por lo que hemos decidido pasar el host como argumento en la ejecución y no gestionar inventarios de hosts.

Ansible cuenta con módulos: Ejecuta tareas específicas en los nodos gestionados, como instalar paquetes, copiar archivos o reiniciar servicios.

Existen muchos más artefactos de Ansible, pero estos son los principales y más comúnmente usados.

Roles

Los **roles** permiten cargar automáticamente variables, archivos, tareas, handlers y otros artefactos Ansible basados en una estructura de archivos.

Después de agrupar tu contenido en roles, puedes reutilizarlos fácilmente y compartirlos con otros usuarios.

Un rol de Ansible tiene una **estructura de directorio** definida con siete directorios estándar principales. Debes incluir al menos uno de estos directorios en cada rol. Puedes omitir cualquier directorio que el rol no utilice. [32] Por ejemplo:

```

roles/
└ common/
    ├── tasks/
    │   └── main.yml
    ├── handlers/
    │   └── main.yml
    ├── templates/
    │   └── ntp.conf.j2
    ├── files/
    │   ├── bar.txt
    │   └── foo.sh
    ├── vars/
    │   └── main.yml
    ├── defaults/
    │   └── main.yml
    ├── meta/
    │   └── main.yml
    ├── library/
    ├── module_utils/
    └── lookup_plugins/

```

En este proyecto **se han usado los roles para definir diferentes flujos de ejecución durante el despliegue**, definiendo si es necesario operativas que dependiendo del contexto pueden ser contraproducentes o críticas en su defecto.

Por ejemplo: podemos definir roles que añadan tareas como una instalación limpia de bases de datos, instalar software de monitoreo de recursos, copias de seguridad, clonado de bases de datos...

Y son estos roles los que están estrechamente relacionados con el campo “**deployType**” de los despliegues, ya que un “deployType” puede incluir uno o varios roles según se configure el **playbook** principal del proyecto.

En el script `RunAnsible.sh` Ansible se ejecuta construyendo comandos parametrizados de esta forma:

```

ansible-playbook -i ${ip}, --private-key=${keypath} -e
\"ansible_user=${user} role=${mode} gitRepo=${repoUrl}\"
${setupYamlPath}

```

Donde las variables son los siguientes:

- **ip:** IP destino de la máquina.
- **keypath:** Path donde está la clave privada para acceder a la máquina.
- **user:** Usuario de la máquina.
- **mode:** El deployType para incluir los roles necesarios.
- **repoUrl:** Repositorio de código de donde sacar los ficheros del proyecto. No es necesario en todos los casos clonar el repositorio dentro de la máquina, pero en muchos casos es necesario.
- **setupYamlPath:** Path al Setup.yaml, el cual es el punto de entrada del script.

PRUEBAS

Para comprobar el correcto funcionamiento del sistema que se ha implementado, se han realizado dos tipos de pruebas que se presentan a continuación:

- **Pruebas de casos de uso y caso real:** Se han definido unos casos de uso específicos para comprobar el correcto funcionamiento de la herramienta y verificar que su funcionalidad es correcta, además de un caso de uso real de la herramienta web.
- **Pruebas unitarias de la API:** Se ha creado una batería de pruebas para testear la API del backend, ya que se trata del punto crítico del sistema, además de comprobar que la validación de campos en los DTOs es correcta.

5.1. Pruebas de casos de uso y caso real

Los casos de uso que se han diseñado y probado consisten en las operaciones de inicio de sesión y gestión de proyectos, hosts y despliegues. Se detallan el apéndice A.

A continuación, se presenta un **caso real** práctico del uso de la herramienta web, en el que **se despliega este mismo proyecto en un servidor local**.

Despliegue de la herramienta web en un servidor local

- 1.– **Inicio de sesión:** El usuario accede a la URL de la herramienta web y se autentica con sus credenciales de Keycloak.

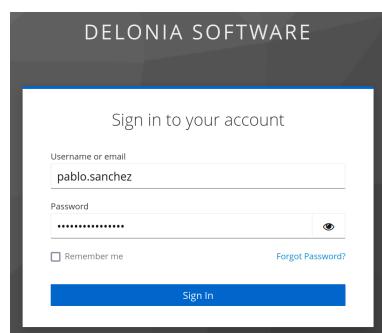


Figura 5.1: Inicio de sesión en la herramienta web.

2.- Creación de un Proyecto: El usuario crea un nuevo proyecto llamado "TFG" con la URL al repositorio del proyecto.

The screenshot shows a dark-themed user interface for managing projects. At the top, there are two notifications: 'Proyecto creado' (Project created) and 'Proyectos recargados' (Projects reloaded). Below this is a button labeled 'Crear nuevo proyecto' (Create new project). The main area is titled 'Proyectos disponibles' (Available Projects) and includes a 'Refrescar' (Refresh) button. A table lists the project details:

ID	Nombre	Descripción	Repository	Tipo de despliegues	Acciones
1	TFG	Herramienta de despliegue de proyectos y entornos	https://scm.delonia.com/scm-git/delonia/devops/dits	FRONT BACK FULLSTACK NOBBDD	Editar Eliminar

Figura 5.2: Listado de proyectos creados.

3.- Creación de un Host: El usuario añade un nuevo host con la dirección IP del servidor.

The screenshot shows a browser window for the Delonia software. The address bar indicates the URL is 'localhost:1177/hosts'. The page title is 'Hosts disponibles'. It features a 'Crear nuevo host' (Create new host) button and a 'Refrescar' (Refresh) button. A table lists the host details:

ID	Name	Ip	Usuario	Estado	Acciones
18	TFG	192.168.222.131	delonia	Cargando...	Editar Eliminar

Figura 5.3: Listado de hosts.

4.- Creación de un Despliegue: El usuario crea un nuevo despliegue con el nombre "Despliegue-TFG" y selecciona el proyecto "TFG" y el host creado anteriormente.

The screenshot shows a dark-themed user interface for managing deployments. At the top, there are two notifications: 'Proyecto creado' (Project created) and 'Proyectos recargados' (Projects reloaded). Below this is a button labeled 'Crear nuevo host' (Create new host) and a 'Refrescar' (Refresh) button. The main area is titled 'Despliegues disponibles' (Available Deployments) and includes a 'Refrescar' (Refresh) button. A table lists the deployment details:

ID	Nombre	Grupo	Descripción	Host	Proyecto	Estado desarrollo	Tipo	Fecha	Desplegado por	Estado	Log	Acciones
1	Despliegue TFG	TFG ing Inf	Prueba despliegue TFG	TFG - 192.168.222.131	TFG	Prueba despliegue	FULLSTACK	-	-	N/A	No logs	▶ Desplegar Editar Eliminar

Figura 5.4: Listado de despliegues.

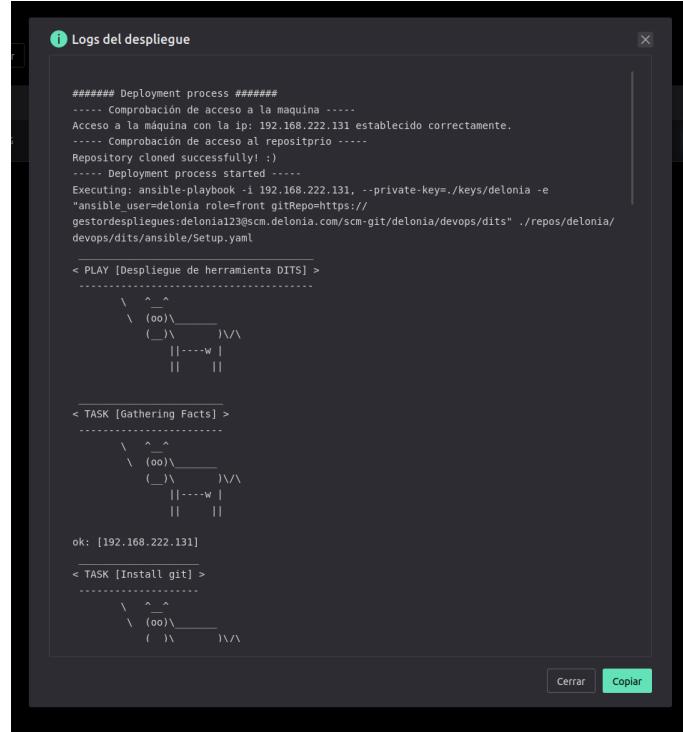
5.- Ejecución del Despliegue: El usuario ejecuta el despliegue y observa cómo se realiza la configuración del servidor.

Despliegues disponibles												
ID	Nombre	Grupo	Descripción	Host	Proyecto	Estado desarrollo	Tipo	Fecha	Desplegado por	Estado	Log	Acciones
1	Despliegue TFG	TFG Ing Inf	Prueba despliegue TFG	TFG - 192.168.222.131	TFG	Prueba despliegue	FULLSTACK	-	pablo	Desplegando...	No logs	Deployar Editar Eliminar

Figura 5.5: Ejecución del despliegue en la herramienta web.

Despliegues disponibles												
ID	Nombre	Grupo	Descripción	Host	Proyecto	Estado desarrollo	Tipo	Fecha	Desplegado por	Estado	Log	Acciones
1	Despliegue TFG	TFG Ing Inf	Prueba despliegue TFG	TFG - 192.168.222.131	TFG	Prueba despliegue	FULLSTACK	31/05/2024 - 12:17:28	pablo	Éxito Leer logs Reintentar Editar Eliminar		

Figura 5.6: Resultado de la ejecución del despliegue. Se observa como ha cambiado el botón de estado a "Éxito", y el de "Desplegar" a "Reintentar".



```

Logs del despliegue

#####
Deployment process #####
----- Comprobación de acceso a la máquina -----
Acceso a la máquina con la ip: 192.168.222.131 establecido correctamente.
----- Comprobación de acceso al repositorio -----
Repository cloned successfully! ;)
----- Deployment process started -----
Executing: ansible-playbook -i 192.168.222.131, --private-key=./keys/delonia -e
"ansible_user=delonia role=front gitRepo=https://gestordespliegues:delonial23@cm.delonia.com/scm-git/delonia/devops/dits" ./repos/delonia/
devops/dits/ansible/setup.yaml

< PLAY [Despliegue de herramienta DITS] >
-----
\   ^__^
 \  (oo)\_____
   (__)\       )\/\
    ||----w |
    ||     ||

< TASK [Gathering Facts] >
-----
\   ^__^
 \  (oo)\_____
   (__)\       )\/\
    ||----w |
    ||     ||

ok: [192.168.222.131]

< TASK [Install git] >
-----
\   ^__^
 \  (oo)\_____
   (__)\       )\/\
    ||----w |
    ||     ||

ok: [192.168.222.131]

```

Cerrar Copiar

Figura 5.7: Inspección de los logs del despliegue.

6.- **Acceso a la aplicación:** El usuario accede a la IP del host y puerto en el que se desplegó la aplicación y comprueba que funciona correctamente.

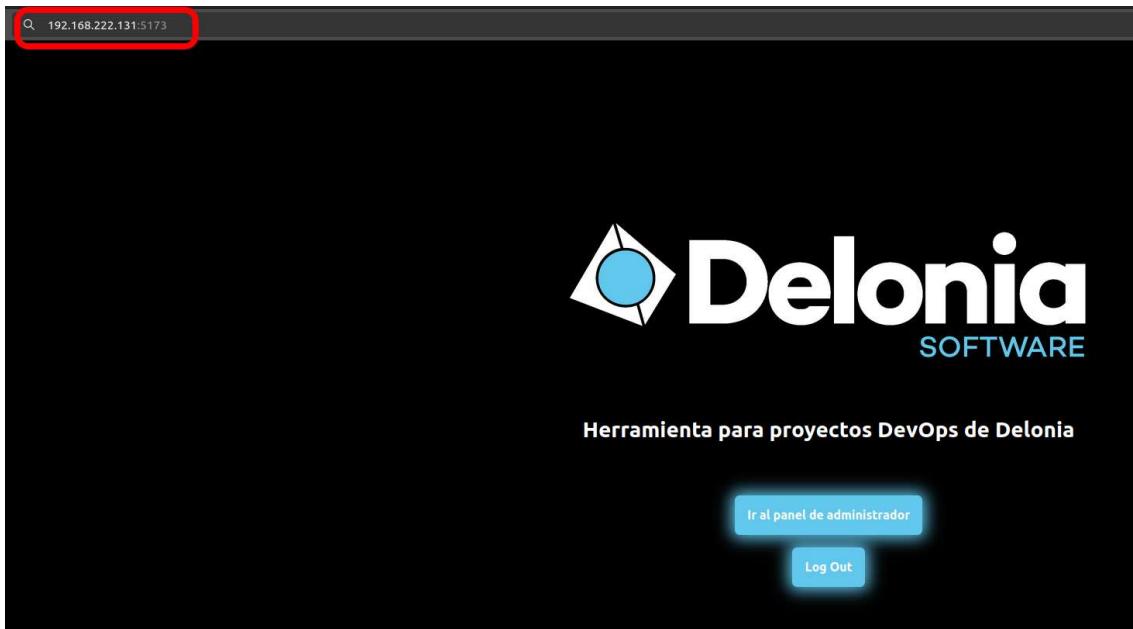


Figura 5.8: Acceso a la aplicación desplegada.

Este caso de uso muestra cómo la herramienta web facilita la gestión de proyectos, hosts y despliegues, permitiendo a los usuarios automatizar tareas de TI de forma eficiente y segura.

5.2. Pruebas unitarias de la API

Se ha utilizado **Postman** [33] para realizar la prueba de la API, un software libre que permite generar peticiones personalizadas para verificar las respuestas del backend.

Se ha elaborado una amplia batería de pruebas que abarca todos los endpoints de la API, contemplando todas las posibles peticiones. Además, se han creado una serie de casos que involucran la validación de campos, detectando y notificando errores en las peticiones. Se han considerado también situaciones de excepción y se ha probado la **resistencia de la API frente a la polución HTTP de parámetros con caracteres especiales**, con el objetivo de evaluar su robustez [34].

CONCLUSIONES Y TRABAJO FUTURO

La constante evolución de la tecnología ha obligado a los técnicos a adaptarse a los nuevos medios y herramientas disponibles para la gestión de las máquinas y servidores.

En este trabajo se ha logrado el objetivo principal de desarrollar una herramienta intuitiva que facilite el control y la administración de proyectos, hosts y despliegues en un entorno seguro y eficiente.

Este proyecto ha sido desarrollado aplicando una amplia gama de conocimientos adquiridos a lo largo de la carrera de Ingeniería Informática, tales como la gestión de sistemas informáticos, desarrollo web, protocolos de comunicación, programación orientada a objetos, gestión de bases de datos... entre otros.

Sin embargo, también ha sido necesario aprender y dominar nuevas tecnologías y herramientas para cumplir con los criterios establecidos por la empresa Delonia Software. Este proceso ha requerido un esfuerzo adicional significativo, pero ha sido gratificante ver cómo este esfuerzo se ha traducido en el resultado final: una aplicación web que es sólida, funcional y fácil de usar.

Es importante señalar que el desarrollo de la herramienta no ha concluido. Las capacidades de la herramienta están limitadas únicamente por las necesidades y deseos de los técnicos que la utilizarán. Siempre surgen nuevas ideas y funciones que podrían añadirse para mejorar la capacidad de los usuarios de controlar y gestionar los recursos de la empresa de manera más eficiente y efectiva. La evolución continua de la herramienta es fundamental para adaptarse a las cambiantes necesidades de los usuarios y a los avances tecnológicos, asegurando que siga siendo una solución relevante y útil.

Una adición al proyecto podría incluir el registro de versiones de paquetes/proyectos y la programación de actualizaciones, permitiendo a los administradores tener una visión completa y actualizada del estado de la infraestructura. También sería interesante la opción de realizar despliegues de proyectos en múltiples hosts de forma simultánea (en clúster), lo que optimizaría el tiempo y los recursos, asegurando que las actualizaciones y nuevas implementaciones se realicen de manera uniforme y eficiente en todos los nodos del sistema.

Además de las nuevas adiciones, también se debe gestionar la mantenibilidad, abstrayendo todo el código posible para su posterior reutilización. Esto facilita la implementación de nuevos elementos en

el proyecto o la corrección de errores, aislando el código afectado lo máximo posible. Un claro ejemplo de mantenibilidad es la característica implementada para la configuración de máquinas mediante un fichero bash y funciones.

Este planteamiento de modularizar el código hace muy fácil la incorporación de nuevas funcionalidades y la modificación de las existentes sin afectar otras partes del sistema. Por ejemplo, si se necesita agregar una nueva función o actualizar una existente, se puede hacer de manera aislada y controlada, reduciendo así el riesgo de introducir errores y mejorando la eficiencia en el desarrollo y mantenimiento del software.

BIBLIOGRAFÍA

- [1] "El futuro del mantenimiento informático." <https://wikin.es/blog/el-futuro-del-mantenimiento-informatico>. Accessed: 2024-06-14.
- [2] "Ansible." <https://www.ansible.com/>. Accessed: 2024-06-14.
- [3] F. García de Zúñiga, "Todo sobre la arquitectura cliente-servidor." <https://www.arsys.es/blog/todo-sobre-la-arquitectura-cliente-servidor>. Accessed: 2024-06-14.
- [4] C. Rodríguez, "10 principales tecnologías frontend para usar en 2022." <https://apliint.com/2022/03/15/10-principales-tecnologias-frontend-para-usar-en-2022/>. Accessed: 2024-06-14.
- [5] "React." <https://es.wikipedia.org/wiki/React>. Accessed: 2024-06-14.
- [6] "Angular (framework)." [https://es.wikipedia.org/wiki/Angular_\(framework\)](https://es.wikipedia.org/wiki/Angular_(framework)). Accessed: 2024-06-14.
- [7] "Ways of using vue." <https://vuejs.org/guide/extras/ways-of-using-vue>. Accessed: 2024-06-14.
- [8] "Create network apps in python django." <https://webwizard.com.pl/es/blog/2021/11/12/create-network-apps-in-python-django/>. Accessed: 2024-06-14.
- [9] "Qué es flask: beneficios y usos." <https://www.mytaskpanel.com/que-es-flask-beneficios-y-usos/>. Accessed: 2024-06-14.
- [10] M. García, "Qué es spring boot." <https://platzi.com/blog/que-es-spring-boot/>. Accessed: 2024-06-14.
- [11] "Qué es mongodb y para qué sirve." <https://imagineinformacion.com/tutoriales/que-es-mongodb-y-para-que-sirve>. Accessed: 2024-06-14.
- [12] "What are application programming interfaces." <https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces>. Accessed: 2024-06-14.
- [13] "Antecedentes rest." https://gsyc.urjc.es/~mortuno/st/antecedentes_rest. Accessed: 2024-06-14.
- [14] "Terraform." https://developer.hashicorp.com/terraform?product_intent=terraform. Accessed: 2024-06-14.
- [15] "Jenkins." <https://www.jenkins.io/>. Accessed: 2024-06-14.
- [16] "Secure shell." https://en.wikipedia.org/wiki/Secure_Shell. Accessed: 2024-06-14.
- [17] "Telnet." <https://en.wikipedia.org/wiki/Telnet>. Accessed: 2024-06-14.
- [18] A. Trevino, "Passkey vs. password: What's the difference?" <https://www.keepersecurity.com/blog/es/2023/10/17/passkey-vs-password-whats-the-difference/>. Accessed: 2024-06-14.

- [19] "Least privilege." <https://www.cyberark.com/es/what-is/least-privilege/>. Accessed: 2024-06-14.
- [20] "Okta." <https://www.okta.com/>. Accessed: 2024-06-14.
- [21] "Auth0." <https://auth0.com/>. Accessed: 2024-06-14.
- [22] "Keycloak." <https://www.keycloak.org/>. Accessed: 2024-06-14.
- [23] "What is oauth 2." <https://auth0.com/es/intro-to-iam/what-is-oauth-2>. Accessed: 2024-06-14.
- [24] "Introduction." <https://jwt.io/introduction>. Accessed: 2024-06-14.
- [25] "Naive ui." <https://www.naiveui.com/en-US/os-theme>. Accessed: 2024-06-14.
- [26] "Fontawesome." <https://fontawesome.com/>. Accessed: 2024-06-14.
- [27] "Mvvm." <https://learn.microsoft.com/es-es/dotnet/architecture/maui/mvvm>, 2024-05-30. Accessed: 2024-06-14.
- [28] "Router view slot." <https://router.vuejs.org/guide/advanced/router-view-slot.html>. Accessed: 2024-06-14.
- [29] "Keycloak-js." <https://www.npmjs.com/package/keycloak-js>. Accessed: 2024-06-14.
- [30] "Forma correcta de realizar la persistencia de varias entidades con el mínimo número de consultas." <https://es.stackoverflow.com/questions/419529/forma-correcta-de-realizar-la-persistencia-de-varias-entidades-con-el-minimo-de-consultas>. Accessed: 2024-06-14.
- [31] "Bean validation." <https://beanvalidation.org/>. Accessed: 2024-06-14.
- [32] "Playbooks reuse roles." https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_reuse_roles.html, 2024-06-13. Accessed: 2024-06-14.
- [33] "Postman." <https://www.postman.com/>. Accessed: 2024-06-14.
- [34] "Http parameter pollution." <https://www.imperva.com/learn/application-security/http-parameter-pollution/>. Accessed: 2024-06-14.

APÉNDICES

CASOS DE USO

A.1. Casos de uso de un usuario sin sesión y con sesión

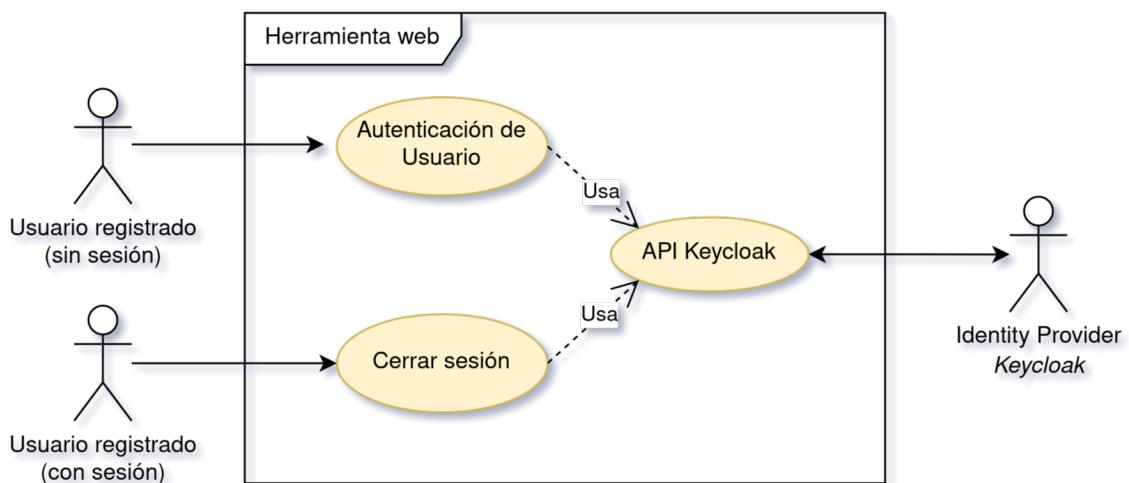


Figura A.1: Diagrama de casos de uso de un usuario sin sesión y con sesión.

Caso de Uso 1: Autenticación de Usuario

- **Actor:** Usuario registrado en el dominio de la empresa
- **Precondiciones:** El usuario debe tener un ID y una contraseña válidos proporcionados por el sistema de autenticación Keycloak de Delonia.
- **Postcondiciones:** El usuario accede a la pantalla de bienvenida de la aplicación web.
- **Flujo Principal:**
 - 1.- El usuario accede a la URL de la aplicación web.
 - 2.- La aplicación redirige al usuario a la pantalla de login de Keycloak.
 - 3.- El usuario ingresa su ID y contraseña.
 - 4.- Keycloak valida las credenciales y genera un token de sesión temporal.
 - 5.- La aplicación redirige al usuario a la pantalla de bienvenida con el logo de la empresa y opciones de navegación.

Caso de Uso 2: Cerrar Sesión

- **Actor:** Usuario autenticado en la aplicación web
- **Precondiciones:** El usuario debe estar autenticado en la aplicación.
- **Postcondiciones:** El usuario es redirigido a la pantalla de login de Keycloak.
- **Flujo Principal:**

- 1.– El usuario hace clic en el botón de cerrar sesión en la barra de navegación.
- 2.– La aplicación elimina el token de sesión y redirige al usuario a la pantalla de login de Keycloak.

A.2. Casos de uso del administrador de proyectos

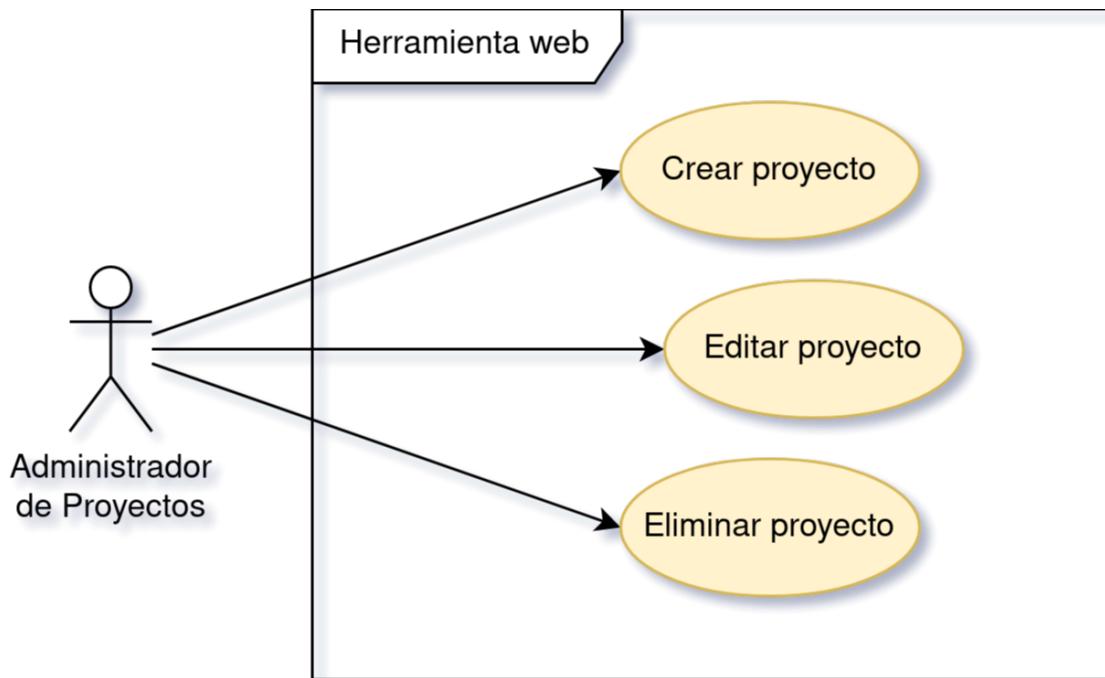


Figura A.2: Diagrama de casos de uso del administrador de proyectos.

Caso de Uso 3: Crear Nuevo Proyecto

- **Actor Principal:** Administrador de Proyectos
- **Precondiciones:** El usuario debe estar autenticado y tener el permiso de "AdministrarProyectos".
- **Postcondiciones:** Un nuevo proyecto es creado y añadido a la lista de proyectos.
- **Flujo Principal:**
 - 1.- El administrador navega a la vista de gestión de proyectos.
 - 2.- El administrador hace clic en el botón "Crear nuevo proyecto".
 - 3.- La aplicación muestra un modal con los campos a llenar: Nombre, Descripción, Repositorio y Tipos de despliegues.
 - 4.- El administrador rellena los campos y hace clic en el botón "Crear".
 - 5.- La aplicación valida los campos y, si son correctos, añade el proyecto a la base de datos y cierra el modal.
 - 6.- El nuevo proyecto aparece en la tabla de proyectos.

Caso de Uso 4: Editar Proyecto

- **Actor Principal:** Administrador de Proyectos
- **Precondiciones:** El usuario debe estar autenticado y tener el permiso de "AdministrarProyectos".
- **Postcondiciones:** Los detalles del proyecto seleccionado son actualizados en la base de datos.
- **Flujo Principal:**

- 1.– El administrador navega a la vista de gestión de proyectos.
- 2.– El administrador selecciona el proyecto que desea editar y hace clic en el botón "Editar".
- 3.– La aplicación muestra un formulario prellenado con los detalles actuales del proyecto: Nombre, Descripción, Repositorio y Tipos de despliegues.
- 4.– El administrador modifica los campos necesarios y hace clic en el botón "Guardar".
- 5.– La aplicación valida los campos y, si son correctos, actualiza los detalles del proyecto en la base de datos y cierra el formulario de edición.
- 6.– Los cambios se reflejan automáticamente en la tabla de proyectos.

Caso de Uso 5: Eliminar Proyecto

- **Actor Principal:** Administrador de Proyectos
- **Precondiciones:** El usuario debe estar autenticado y tener el permiso de "AdministrarProyectos".
- **Postcondiciones:** El proyecto seleccionado es eliminado de la base de datos.
- **Flujo Principal:**

- 1.– El administrador navega a la vista de gestión de proyectos.
- 2.– El administrador selecciona el proyecto que desea eliminar y hace clic en el botón "Eliminar".
- 3.– La aplicación muestra una confirmación de eliminación.
- 4.– El administrador confirma la eliminación.
- 5.– La aplicación elimina el proyecto de la base de datos.
- 6.– El proyecto eliminado desaparece de la tabla de proyectos.

A.3. Casos de uso del administrador de hosts

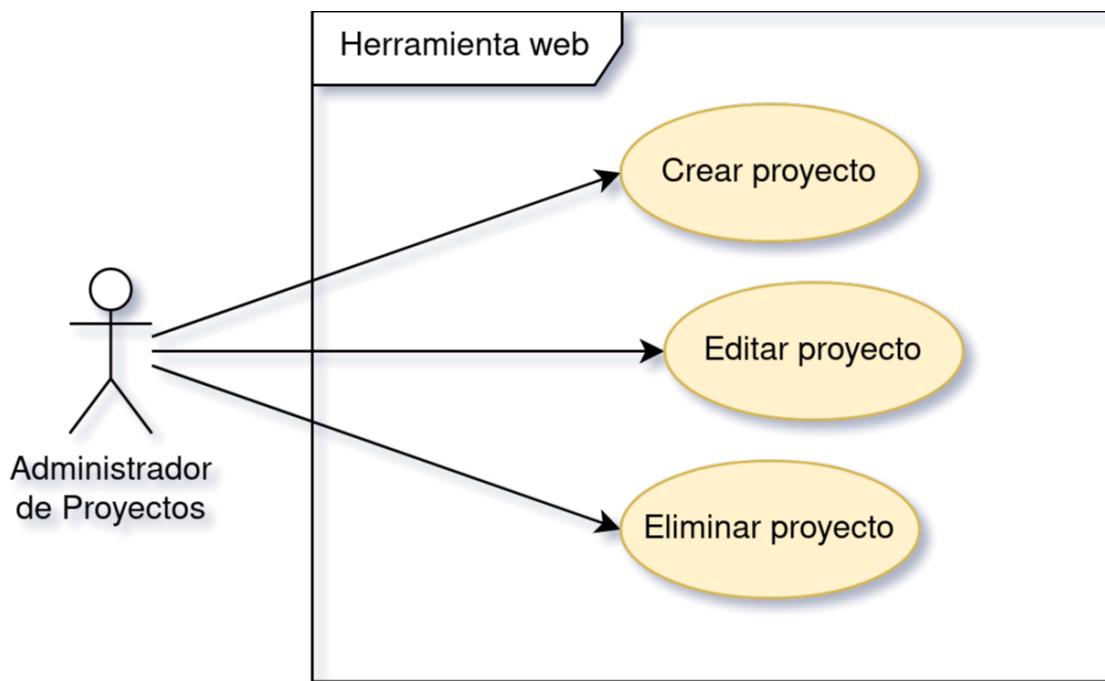


Figura A.3: Diagrama de casos de uso del administrador de hosts.

Caso de Uso 6: Crear Nuevo Host

- **Actor Principal:** Administrador de Hosts
- **Precondiciones:** El usuario debe estar autenticado y tener el permiso de "AdministrarHosts".
- **Postcondiciones:** Un nuevo host es creado y añadido a la lista de hosts.
- **Flujo Principal:**
 - 1.– El administrador navega a la vista de gestión de hosts.
 - 2.– El administrador hace clic en el botón "Crear nuevo host".
 - 3.– La aplicación muestra un formulario con los campos a llenar: Nombre, IP, Usuario.
 - 4.– El administrador rellena los campos y hace clic en el botón "Crear".
 - 5.– La aplicación valida los campos y, si son correctos, añade el nuevo host a la base de datos y cierra el formulario de creación.
 - 6.– El nuevo host aparece en la lista de hosts de la vista de gestión de hosts.

Caso de Uso 7: Editar Host

- **Actor Principal:** Administrador de Hosts
- **Precondiciones:** El usuario debe estar autenticado y tener el permiso de "AdministrarHosts".
- **Postcondiciones:** Los detalles del host seleccionado son actualizados en la base de datos.
- **Flujo Principal:**
 - 1.– El administrador navega a la vista de gestión de hosts.

- 2.- El administrador selecciona el host a editar y hace clic en el botón "Editar".
- 3.- La aplicación muestra un modal con los campos actuales del host: Nombre, IP, Usuario.
- 4.- El administrador modifica los campos necesarios y hace clic en el botón "Guardar".
- 5.- La aplicación valida los campos y, si son correctos, actualiza los detalles del host en la base de datos y cierra el modal.
- 6.- Los cambios se reflejan en la tabla de hosts.

Caso de Uso 8: Eliminar Host

- **Actor Principal:** Administrador de Hosts
- **Precondiciones:** El usuario debe estar autenticado y tener el permiso de "AdministrarHosts".
- **Postcondiciones:** El host seleccionado es eliminado de la base de datos.
- **Flujo Principal:**
 - 1.- El administrador navega a la vista de gestión de hosts.
 - 2.- El administrador selecciona el host que desea eliminar y hace clic en el botón "Eliminar".
 - 3.- La aplicación muestra una confirmación de eliminación.
 - 4.- El administrador confirma la eliminación.
 - 5.- La aplicación elimina el host de la base de datos.
 - 6.- El host eliminado desaparece de la tabla de hosts.

A.4. Casos de uso del administrador de despliegues

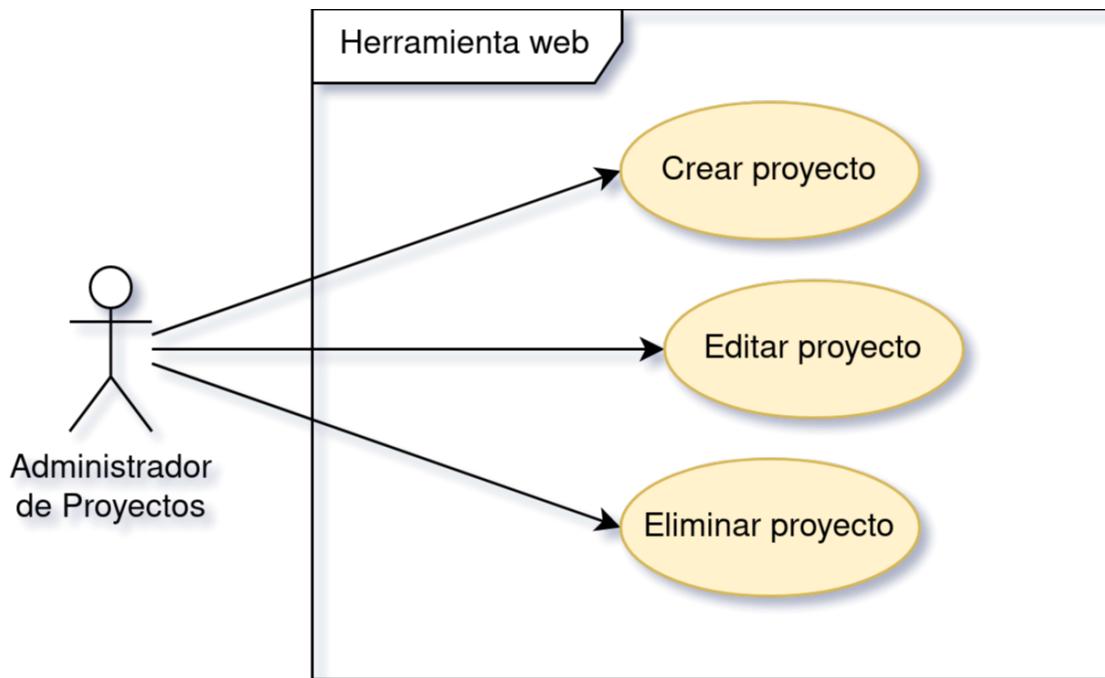


Figura A.4: Diagrama de casos de uso del administrador de despliegues.

Caso de Uso 9: Crear Nuevo Despliegue

- **Actor Principal:** Administrador de Proyectos
- **Precondiciones:** El usuario debe estar autenticado y tener el permiso de "AdministrarDespliegues".
- **Postcondiciones:** Un nuevo despliegue es creado y añadido a la lista de tipos de despliegue del proyecto.
- **Flujo Principal:**
 - 1.– El administrador navega a la vista de gestión de proyectos.
 - 2.– El administrador selecciona el proyecto al cual desea añadir un nuevo despliegue.
 - 3.– En la sección de tipos de despliegue del proyecto, el administrador hace clic en el botón "Crear nuevo despliegue".
 - 4.– La aplicación muestra un formulario con los campos a llenar: Nombre, Descripción, URL.
 - 5.– El administrador rellena los campos y hace clic en el botón "Crear".
 - 6.– La aplicación valida los campos y, si son correctos, añade el nuevo despliegue a la base de datos y cierra el formulario de creación.
 - 7.– El nuevo despliegue aparece en la lista de tipos de despliegue del proyecto.

Caso de Uso 10: Editar Despliegue

- **Actor Principal:** Administrador de Proyectos
- **Precondiciones:** El usuario debe estar autenticado y tener el permiso de "AdministrarDespliegues".
- **Postcondiciones:** Los detalles del despliegue seleccionado son actualizados en la base de datos.

- **Flujo Principal:**

- 1.– El administrador navega a la vista de gestión de proyectos.
- 2.– El administrador selecciona el proyecto que contiene el despliegue a editar.
- 3.– En la sección de tipos de despliegue del proyecto, el administrador selecciona el despliegue a editar y hace clic en el botón "Editar".
- 4.– La aplicación muestra un formulario con los campos actuales del despliegue: Nombre, Descripción, URL.
- 5.– El administrador modifica los campos necesarios y hace clic en el botón "Guardar".
- 6.– La aplicación valida los campos y, si son correctos, actualiza los detalles del despliegue en la base de datos y cierra el formulario de edición.
- 7.– Los cambios se reflejan en la lista de tipos de despliegue del proyecto.

Caso de Uso 11: Eliminar Despliegue

- **Actor Principal:** Administrador de Proyectos

- **Precondiciones:** El usuario debe estar autenticado y tener el permiso de "AdministrarDespliegues".

- **Postcondiciones:** El despliegue seleccionado es eliminado de la base de datos.

- **Flujo Principal:**

- 1.– El administrador navega a la vista de gestión de proyectos.
- 2.– El administrador selecciona el proyecto que contiene el despliegue a eliminar.
- 3.– En la sección de tipos de despliegue del proyecto, el administrador selecciona el despliegue a eliminar y hace clic en el botón "Eliminar".
- 4.– La aplicación muestra una confirmación de eliminación.
- 5.– El administrador confirma la eliminación.
- 6.– La aplicación elimina el despliegue de la base de datos.
- 7.– El despliegue eliminado desaparece de la lista de tipos de despliegue del proyecto.

Caso de Uso 12: Desplegar Proyecto

- **Actor Principal:** Administrador de Despliegues

- **Precondiciones:** El usuario debe estar autenticado y tener el permiso de "AdministrarDespliegues".

- **Postcondiciones:** El proyecto es desplegado y el estado del despliegue se actualiza.

- **Flujo Principal:**

- 1.– El administrador navega a la vista de gestión de despliegues.
- 2.– El administrador selecciona el despliegue a realizar y hace clic en el botón "Desplegar".
- 3.– La aplicación verifica que los campos de proyecto y host no sean nulos.
- 4.– La aplicación inicia el proceso de despliegue.
- 5.– Durante el despliegue, el botón "Desplegar" está desactivado.
- 6.– Tras completar el despliegue, los campos de "Fecha", "Desplegado por" y "Estado" se actualizan.
- 7.– Si el despliegue falla, el botón "Desplegar" cambia a "Reintentar".

Caso de Uso 13: Ver Logs de Despliegue

- **Actor Principal:** Administrador de Despliegues
- **Precondiciones:** El usuario debe estar autenticado, tener el permiso de "AdministrarDespliegues" y el campo de logs no debe estar vacío.
- **Postcondiciones:** Los logs del despliegue seleccionado son mostrados en pantalla.
- **Flujo Principal:**
 - 1.– El administrador navega a la vista de gestión de despliegues.
 - 2.– El administrador selecciona el despliegue cuyos logs desea ver y hace clic en el botón "Ver logs".
 - 3.– La aplicación muestra un modal con la última traza de ejecución con un scroll vertical.
 - 4.– El administrador puede revisar los logs y cerrar el modal al finalizar.

HERRAMIENTA GIT

B.1. Introducción a Git

Git es un sistema de control de versiones distribuido ampliamente utilizado en proyectos de desarrollo de software. Proporciona un entorno colaborativo y seguro para gestionar el código fuente de un proyecto.

En Git, cada proyecto tiene un repositorio que almacena todas las versiones del código y su historial de cambios. Los desarrolladores pueden clonar este repositorio en sus propias máquinas y trabajar en él de forma independiente. Los cambios realizados por cada desarrollador se registran en su propio repositorio local.

B.2. Importancia de Git en el proyecto

La importancia de Git en el proyecto radica en varios aspectos cruciales para la gestión y desarrollo efectivo del código fuente. En primer lugar, Git facilita la obtención de la versión más actualizada del código de un proyecto. Esto es fundamental porque garantiza que todos los desarrolladores trabajen con la misma base de código y tengan acceso a las últimas modificaciones realizadas por el equipo.

Una vez obtenido el código fuente del proyecto desde Git, se utiliza el **BashCommandRunner** para ejecutar los scripts de **Ansible** necesarios. Esta integración es crucial para automatizar tareas como la configuración de máquinas y el despliegue de aplicaciones, asegurando que los scripts y configuraciones utilizados sean consistentes y actualizados. Esto garantiza la estabilidad y el correcto funcionamiento de operaciones automatizadas, incluyendo la configuración de entornos, despliegue de aplicaciones y ejecución de procesos críticos dentro del proyecto.



Universidad Autónoma
de Madrid