Facultad de Ciencias

DESARROLLO DE UN DASHBOARD PARA EL ACCESO A UNA PLATAFORMA DE ENTRENAMIENTO DE MODELOS DE INTELIGENCIA ARTIFICIAL

(Development of a dashboard for accessing an artificial intelligence model training platform)

Trabajo de Fin de Máster
para acceder al

MÁSTER UNIVERSITARIO EN INGENIERÍA INFORMÁTICA

Autora: Marta Obregón Ruiz

Director: Álvaro López García

Codirector: Pedro Andrés Heredia Canales

Julio - 2024

Resumen

Este Trabajo de Fin de Máster se ha desarrollado dentro de proyecto Artificial Intelligence for the European Open Science Cloud (AI4EOSC) de la Unión Europea, y tiene como objetivo facilitar el acceso a técnicas de inteligencia artificial a la comunidad investigadora. Cocretamente, se centra en el desarrollo de una aplicación web que permite visualizar, desplegar y gestionar diferentes módulos para posteriormente entrenarlos utilizando técnicas de machine learning y deep learning. A parte del propio desarrollo de la plataforma, también se engloban cuestiones de seguridad y control de accesos, control de versiones e integración y entrega continua (CI/CD).

Con el fin de lograr estos objetivos, se ha construido una aplicación web implementada sobre Angular y FastAPI, utilizando una metodología de desarrollo ágil basada en Kanban.

Palabras Clave: Aplicación Web, REST, Machine Learning, AI, Cloud, Aprendizaje Federado, Centro de Datos

Abstract

This Master's Thesis has been developed within the European Union project Artificial Intelligence for the European Open Science Cloud (AI4EOSC), aiming to facilitate access to artificial intelligence techniques for the research community. Specifically, it focuses on the development of a web application that allows visualization, deployment, and management of different modules for subsequent training using machine learning and deep learning techniques. Apart from the platform development itself, issues such as security and access control, version control, and continuous integration and delivery (CI/CD) are also encompassed.

In order to achieve these objectives, a web application has been built using Angular and FastAPI, employing an agile development methodology based on Kanban.

Keywords: Web Application, REST, Machine Learning, AI, Cloud, Federated Learning, Datacenter

Índice

1. Introducción **1**
- 1.1. Proyecto AI4EOSC . . . 1
- 1.2. Estado del arte . . . 2
- 1.3. Objetivos . . . 3
- 1.4. Alcance del TFM . . . 4
- 1.5. Metodología de desarrollo . . . 4
  - 1.5.1. Planificación externa . . . 5
  - 1.5.2. Planificación interna . . . 5

2. Background **6**

3. Requisitos **8**
- 3.1. Requisitos funcionales . . . 8
- 3.2. Requisitos no funcionales . . . 10

4. Arquitectura y Diseño **10**
- 4.1. Arquitectura de la aplicación . . . 10
- 4.2. Diseño de la API REST . . . 12
- 4.3. Diseño de la interfaz gráfica . . . 14
  - 4.3.1. Dashboard . . . 14
  - 4.3.2. Claves de acceso . . . 17

5. Implementación **18**
- 5.1. Frontend . . . 18
- 5.2. Backend . . . 23

6. Pruebas y despliegue **24**
- 6.1. Pruebas unitarias . . . 24
- 6.2. Pruebas end-to-end . . . 26
- 6.3. Pruebas de aceptación . . . 28
- 6.4. Integración continua y despliegue . . . 28

7. Conclusiones **33**

8. Trabajos futuros **34**

1. Introducción

Este trabajo de fin de máster se ha desarrollado dentro del proyecto europeo AI4EOSC [1], en el cual participo actualmente a través del Consejo Superior de Investigaciones Científicas (CSIC) [2]. Asimismo, todas las aportaciones y desarrollos presentados en este trabajo, también son utilizados en el proyecto iMagine [3], ya que hace uso de la plataforma desarrollada para AI4EOSC.

Este primer capítulo presenta el proyecto AI4EOSC y detalla su alcance y objetivos. El segundo apartado expone el estado del arte en el ámbito de la investigación. A continuación, se habla sobre los objetivos concretos de este trabajo de fin de máster y su alcance. Finalmente, el último apartado aborda la planificación y la metodología de desarrollo utilizadas para la realización del mismo.

1.1. Proyecto AI4EOSC

AI4EOSC forma parte de uno de los proyectos impulsados por la Unión Europea para implementar la European Open Science Cloud (EOSC) [4], un entorno diseñado para alojar y procesar datos de investigación dentro de la Unión Europea. Su principal propósito es proveer un entorno federado y multidisciplinar donde tanto investigadores europeos, como empresas y ciudadanos, puedan publicar, encontrar y reutilizar datos, herramientas y servicios, con fines de investigación, de innovación y educativos.

Concretamente, el proyecto AI4EOSC se centra en proveer servicios de inteligencia artificial (IA), machine learning (ML) y deep learning (DL) en el entorno EOSC. Esto se traduce en los siguiente objetivos específicos:

- Proveer servicios de ML que permitan construir y desplegar aplicaciones de ML, DL e IA personalizables, siguiendo un enfoque *serverless* con escalabilidad horizontal sobre el EOSC.
- Mejorar los servicios de cloud existentes para poder aplicar ML y DL sobre conjuntos de datos distribuidos, utilizando para ello el aprendizaje federado.
- Ampliar y potenciar la oferta de modelos de ML y DL disponibles en el portal EOSC.
- Coordinar las actividades de AI4EOSC con las de otros proyectos, tanto existentes como futuros, creando sinergias y estableciendo lazos de colaboración.

El proyecto se construyó sobre una iniciativa anterior llamada DEEP-Hybrid-DataCloud 2020 [5], la cual tenía como objetivo permitir a los investigadores explotar, de forma transparente, los recursos computacionales de e-infraestructuras panaeuropeas. Como resultado, se obtuvo la plataforma DEEP, un sistema utilizado hoy en día por investigadores de la UE para entrenar y desarrollar modelos de ML y DL, los cuales se ejecutan en Centros de Procesamiento de Datos (CPD) europeos. Tras el éxito de esta iniciativa, se ideó AI4EOSC, que comenzó en el mes de septiembre del 2022 y tiene como fecha de finalización agosto de 2025. Dentro de este periodo, existen una serie de hitos y puntos clave que marcan su desarrollo y evolución. Estos se pueden agrupar en dos etapas:

1. La primera etapa se centra en la transición de la plataforma DEEP al nuevo stack tecnológico, de forma que esta nueva plataforma ofrezca funciones de

provisión y entrenamiento de modelos. Asimismo, se desea integrar la plataforma con otros servicios EOSC existentes, como por ejemplo aquellos dedicados a la autenticación, autorización y monitorización de los usuarios.

2. La segunda fase consiste en refinar la integración llevada a cabo en la primera etapa e incorporar nuevos requisitos que hayan podido surgir durante el desarrollo. Esta fase también incluye la realización de actividades de divulgación, como workshops o hackathons, con el objetivo de dar a conocer el proyecto e integrar otros casos de uso que puedan verse beneficiados por la plataforma.

Actualmente, nos encontramos en la segunda etapa del proyecto, por lo que los objetivos definidos en la primera etapa ya se han completado y son accesibles a través de la aplicación web oficial de la plataforma [6].

Para poder estudiar su viabilidad y efectividad, se seleccionaron tres casos de uso, de entre varios casos reales diferentes, cuyos requisitos guían el desarrollo de la plataforma:

- Termografía automatizada: utilizará una solución basada en inteligencia artificial que aprovecha imágenes térmicas para poder identificar anomalías en la temperatura en entornos urbanos. La tecnología estará alojada en un servicio cloud automatizado y empleará técnicas de aprendizaje descentralizado como el aprendizaje federado.

- Agrometeorología: busca potenciar los modelos matemáticos actualmente utilizados en la detección de enfermedades vegetales junto con modelos de inteligencia artificial, de manera que puedan ser desarrollados y escalados en la plataforma AI4EOSC. Para ello también hará uso del aprendizaje federado y el ML.

- Protección integrada de plantas: combina el uso de la inteligencia artificial y los datos meteorológicos para proporcionar advertencias oportunas y precisas a los agricultores. Hará uso de la plataforma para desplegar y entrenar modelos que permitan comprender las características de estos procesos naturales.

La diversidad de los dominios científicos cubiertos por estos casos de uso evidencia la naturaleza interdisciplinaria del proyecto, que puede ser aplicado a problemas de distintos ámbitos. Asimismo, también existen otros casos de uso externos que se han ido sumando una vez empezado el proyecto.

1.2. Estado del arte

En este apartado, se mencionan las tecnologías más destacadas en el campo del desarrollo software de este tipo de aplicaciones web, así como las principales técnicas y enfoques utilizados en la actualidad. Además, se detalla el por qué de las distintas decisiones de diseño tomadas a la hora de elegir las tecnologías empleadas en este proyecto. Es importante destacar que estas elecciones se enfocan desde una perspectiva de investigación, que difiere significativamente del enfoque empresarial, con el objetivo de garantizar la idoneidad y efectividad del sistema propuesto en el contexto científico.

En el ámbito de la investigación se observa una constante evolución de tecnologías y metodologías en todos los campos de la informática. Por ejemplo, en la

orquestración y automatización de cargas de trabajo, tecnologías como Nomad y Kubernetes han ganado bastante popularidad. Nomad [7], desarrollado por HashiCorp [8], proporciona una plataforma flexible para la gestión de clústers de aplicaciones y servicios. Mientras tanto, Kubernetes [9] se ha convertido en el estándar de facto para la orquestación de contenedores, ofreciendo escalabilidad, resiliencia y portabilidad. En este proyecto, se decidió utilizar Nomad frente a Kubernetes debido a que este último únicamente permite orquestar contenedores, mientras que Nomad también permite gestionar trabajos más sencillos. Por otro lado, la instalación de Nomad es menos compleja, ya que se basa en un binario y un fichero de configuración, y está pensada para trabajos tipo batch y multicloud. Esto último es de gran importancia, ya que en este proyecto las cargas de trabajo a orquestar son de tipo batch, las cuales no requieren otros elementos que pueden añadir una mayor complejidad, como por ejemplo balanceadores de carga. Además se tiene como objetivo utilizar un clúster federado, para lo cual se necesitan ciertas prestaciones como Multi-Region Federation (permitir la administración de recursos y trabajos a través de distintas ubicaciones geográficas) o Consistent Global State (mantener un estado global consistente mediante la sincronización entre las regiones federadas). Aunque las dos tecnologías ofrecen estas funcionalidades, la implementación y el enfoque entre ambas difiere. Kubernetes tiende a tener una curva de aprendizaje más pronunciada y puede requerir herramientas y configuraciones adicionales para alcanzar la misma funcionalidad *out-of-the-box* que ofrece Nomad.

Por otro lado, también se necesitaba una herramienta para la gestión de secretos y protección de datos confidenciales dentro del entorno de AI4EOSC, para lo cual se eligió Vault [10]. Esta decisión fue consecuencia de que tanto Nomad como Vault pertenecen a la empresa HashiCorp, por lo que se consideró que utilizar ambas en conjunto facilitaría la integración y mantenibilidad del sistema.

En cuanto al desarrollo de interfaces de usuario, frameworks como Angular [11] y React [12] han transformado la forma en que se construyen aplicaciones web. Ambos utilizan un paradigma de programación reactiva, que se basa en sistemas que reaccionan de manera automática y eficiente a los cambios en los datos o en el estado del sistema. Por un lado, Angular ofrece una estructura robusta y modular para el desarrollo de aplicaciones escalables. Por otro lado, React destaca por la posibilidad de crear interfaces dinámicas y receptivas. En este caso, Angular fue el framework seleccionado debido a que, según la opinión del equipo de desarrollo, es más estructurado y consistente, mientras que como React es una librería, no tiene tantas normas ni reglas predefinidas [13].

Por último, respecto al backend, lenguajes de programación como Python [14] se han convertido en una de las tecnologías más utilizadas para su desarrollo, debido a su versatilidad, facilidad de uso y gran cantidad de bibliotecas disponibles. Además de Python, hay otras tecnologías comunes que se utilizan para el desarrollo de backend en proyectos de investigación, como por ejemplo, Node.js, PHP o Java. Sin embargo, Python se adecuaba mejor a las características de este proyecto, ya que tiene clientes para todas las herramientas que se iban a utilizar, como Nomad [15] y Oscar [16].

1.3. Objetivos

Tal y como se ha mencionado anteriormente, el proyecto AI4EOSC arrancó en 2022, y mi participación en el proyecto comenzó en noviembre del 2023, por lo que el desarrollo, y por tanto los objetivos de este TFM, se basan las aportaciones que

realicé a partir de ese momento. En base a esto, se tienen los siguientes propósitos:

- Realizar un seguimiento de los recursos computacionales de cada despliegue y mostrárselos al usuario en una nueva sección dedicada a estadísticas.
- Implementar claves de acceso en los despliegues de aprendizaje federado para permitir la colaboración controlada de distintos participantes.
- Aplicar mejoras en la interfaz gráfica de la plataforma, lo que incluye tareas como adaptar el diseño a todo tipo de pantallas (interfaz responsive).
- Implementar pruebas de las nuevas funcionalidades.
- Actualizar los procesos de integración continua y añadir controles de calidad de código.

1.4. Alcance del TFM

A continuación se especifica, fase por fase, la contribución de este trabajo a cada parte del ciclo de vida software. Cabe destacar que AI4EOSC se trata de un proyecto europeo a gran escala, y que ya estaba empezado cuando me incorporé en él. Sin embargo, lo reflejado en este trabajo son tareas realizadas exclusivamente por mí, para las cuales he contado la experiencia y el conocimiento de compañeros expertos en la materia, que me han ayudado y guiado durante su desarrollo.

En este caso, el proceso de obtención de los requisitos ya se había realizado al inicio del proyecto. Aun así, a partir de los comentarios y sugerencias de los casos de uso, surgieron nuevas funcionalidades y mejoras que se fueron incluyendo en la plataforma de manera incremental mediante el uso de una metodología de desarrollo ágil. Al igual que en el caso anterior, las decisiones referentes a la arquitectura fueron tomadas al comienzo del proyecto. No obstante, pude decidir sobre ciertos aspectos y detalles de las funcionalidades que implementé a posteriori. Entre ellas, este trabajo se centra en las destacadas en el apartado anterior. En cuanto a las pruebas, se han implementado tests unitarios y pruebas end-to-end de dichas funcionalidades y finalmente, se realizaron mejoras y ampliaciones en los procesos de integración continua y despliegue de la aplicación web.

1.5. Metodología de desarrollo

En este apartado se detalla cómo fue el proceso de incorporación al proyecto y cuál es la planificación que se ha llevado a cabo una vez pasó este periodo de adaptación. En primer lugar, cuando me incorporé al proyecto me explicaron en qué consistía la iniciativa AI4EOSC y qué papel iba a tener yo dentro del mismo. A continuación, accedí al dashboard de la plataforma para familiarizarme con el entorno y poder relacionar los objetivos del proyecto con las distintas funciones de la web. A su vez, realicé un análisis de la documentación del proyecto [17] que, a parte de explicar los aspectos de la parte frontend, detalla otros conceptos, tanto de la infraestructura como de IA, indispensables para entender el funcionamiento completo de la plataforma.

Las primeras semanas las dediqué a familiarizarme con el código de los repositorios solventando pequeños bugs, aplicando mejoras simples y formateando la interfaz con nuevos estilos. Después, empecé a realizar tareas más complejas, como la incorporación de nuevas funcionalidades o la mejora de otras ya existentes en

la plataforma. Finalmente, este proceso me ha llevado a asumir la responsabilidad de esta parte del proyecto. Las tareas concretas que detallan estos desarrollos se explican más adelante, en el apartado de Requisitos.

El proceso de desarrollo se llevó a cabo siguiendo las metodologías de desarrollo y la planificación descritas a continuación. En este caso, no se sigue una metodología de manera rígida sino que se han adaptado conceptos de distintas metodologías ágiles. Concretamente, se aplican ideas de:

- Kanban [18]: para ilustrar y organizar los flujos de trabajo de una manera visual, las tareas se representan con tarjetas, las cuales van avanzando por diversas etapas hasta su finalización.
- Scrum [19]: el desarrollo de los requisitos del proyecto es incremental y se da prioridad a aquellas tareas que son más importantes para los clientes, que en este caso son los casos de uso del proyecto. También se llevan a cabo reuniones periódicas en las que los participantes comentan sus avances y exponen los problemas que les han surgido durante el desarrollo.

Debido a la envergadura del proyecto, se distinguen dos planificaciones distintas: la planificación externa, que engloba la organización entre los distintos partners del proyecto, y planificación interna, que hace referencia a la organización que se ha llevado a cabo entre los participantes del IFCA.

1.5.1. Planificación externa

En el proyecto AI4EOSC participan 10 instituciones europeas, las cuales se organizan en 7 grupos de trabajo denominados Work Packages, cada uno de ellos centrado y especializado en una parte del proyecto. Cada Work Package tiene una reunión cada dos semanas, la cual sería el equivalente a la Daily Scrum, en la que cada grupo informa de los avances que ha realizado en su tarea y comenta si ha tenido algún problema. Además, se rellena una plantilla en Confluence [20] que documenta todos los aspectos importantes que se han comentado en la reunión. Por otro lado, al finalizar cada bloque de trabajo, es necesario rellenar un documento acerca de todos los avances que ha habido en el proyecto y entregarlo a la comisión para que este sea aprobado. Esto se podría asemejar a la Sprint Review, ya que se presentan y detallan los trabajos completados.

1.5.2. Planificación interna

Con el objetivo de llevar un seguimiento más detallado y personalizado del estado del proyecto, los participantes del IFCA utilizan la plataforma Wekan [21] para la gestión de tareas. Wekan es una plataforma de software libre que permite crear y gestionar tableros Kanban. La metodología Kanban, previamente mencionada, es un tipo de metodología ágil que consiste en organizar las tareas de un proyecto en un tablero compartido por todos los participantes. Dichas tareas se organizan en columnas en función de su estado. Esta organización tiene que distinguir entre qué tareas están por realizar, cuáles se están realizando y cuáles se han terminado, sin embargo, se pueden agregar otras categorías para adaptar el tablero al proyecto concreto. En este caso se tienen los siguientes grupos:

- Ideas: sugerencias o posibles funcionalidades que podrían incluirse en la hoja de ruta del proyecto.

- Backlog: tareas pendientes que conforman la hoja de ruta del proyecto, que define cómo evolucionará a lo largo del tiempo. Contiene las nuevas funciones y características a implementar.
- Selected: tareas del backlog que han sido seleccionadas para ser implementadas. Suelen colocarse en este grupo las tareas con mayor prioridad, ya que son las primeras que se han de incluir.
- In progress: tareas que están en desarrollo.
- Done: tareas que han sido finalizadas y testeadas.

A parte de las reuniones mencionadas en la planificación externa, también se llevan a cabo reuniones quincenales en el grupo del IFCA para comentar los avances y mantener al tanto al equipo de en qué estado está cada tarea del proyecto. Además, estas reuniones sirven para comentar el progreso de cada Work Package y facilitar la integración entre cada uno de ellos, ya que no todos colaboramos en los mismos paquetes de trabajo. Por último, como el equipo de desarrollo encargado de la plataforma web es pequeño, los progresos diarios se comentaban de manera informal. Esto se debe a que somos tres personas a cargo de esta parte del desarrollo: inicialmente, un compañero estaba a cargo de la parte del frontend y otro de la parte del backend, mientras que yo colaboraba en ambas. Es por ello que se consideró más eficiente comentar los avances y los problemas de manera individual sin necesidad de establecer reuniones diarias.

2. Background

En este capítulo se describe a grandes rasgos cuál era el estado de la plataforma previo a las aportaciones de este trabajo, y se explican conceptos clave útiles para entender las nuevas funcionalidades implementadas.

La aplicación web de AI4EOSC permite a los usuarios interactuar con la plataforma de manera simple y gráfica. Una vez que el usuario ha iniciado sesión, la interfaz se divide en dos secciones principales: (1) Marketplace y (2) Deployments. La primera sección permite a los usuarios lanzar despliegues basados en un módulo genérico que se utiliza como base para crear nuevos módulos, o en uno de la amplia lista de módulos que ya están listos para entrenar (Modules). Por otro lado, las herramientas (Tools) están destinadas a proporcionar un entorno con un conjunto de utilidades software orientado a resolver un problema específico. Por esta razón, se tratan de manera diferente la aplicación, ya que necesitan entradas de configuración especiales adaptadas para cada uno de ellos. Esta primera sección está disponible tanto para usuarios registrados como para los no registrados, aunque con opciones limitadas para estos últimos. Todos los módulos y herramientas listados en esta sección tienen una vista detallada que muestra información adicional como: una breve descripción, las categorías a las que pertenece, enlaces al repositorio de código, la imagen de Docker e incluso el conjunto de datos (si está disponible). La Figura 1 muestra un ejemplo del Marketplace de AI4EOSC con los diferentes módulos preentrenados.

Respecto a las herramientas, la única disponible actualmente en la plataforma es la de Federated Learning Server, la cual permite crear un servidor de aprendizaje federado para poder entrenar un modelo con inteligencia artificial. Esta técnica de aprendizaje automático permite entrenar un algoritmo a través de una arquitectura descentralizada formada por un servidor y múltiples usuarios, los cuales mantienen

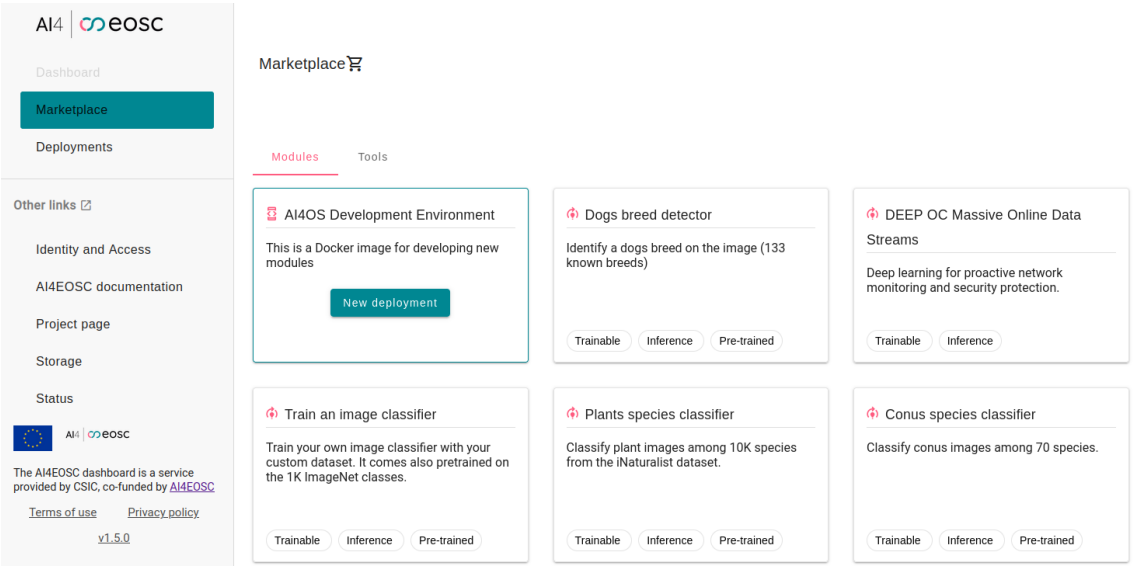

Figura 1: Versión inicial del Dashboard de AI4EOSC (Sección Marketplace)

sus propios datos de entrenamiento locales y privados. De esta manera, el servidor centralizado mantiene un modelo global y cada participante recibe una copia para entrenar con su propio conjunto de datos. Tras entrenar el modelo localmente durante una serie de iteraciones, los participantes envían su versión actualizada del modelo al servidor centralizado, manteniendo siempre su conjunto de datos dentro de su propia infraestructura. El principal objetivo del aprendizaje federado es poder conservar la integridad de la información que está siendo utilizada para el aprendizaje sin poner en peligro la privacidad y seguridad, ya que, en la mayoría de casos, los datos utilizados para entrenar este tipo de modelos son de carácter sensible.

La segunda sección, Deployments, muestra los despliegues actuales del usuario registrado, diferenciando entre módulos y herramientas, para que puedan monitorizar su estado en el clúster y realizar otras acciones como acceder a un despliegue o incluso eliminarlo por completo. La Figura 2 ilustra esta sección.

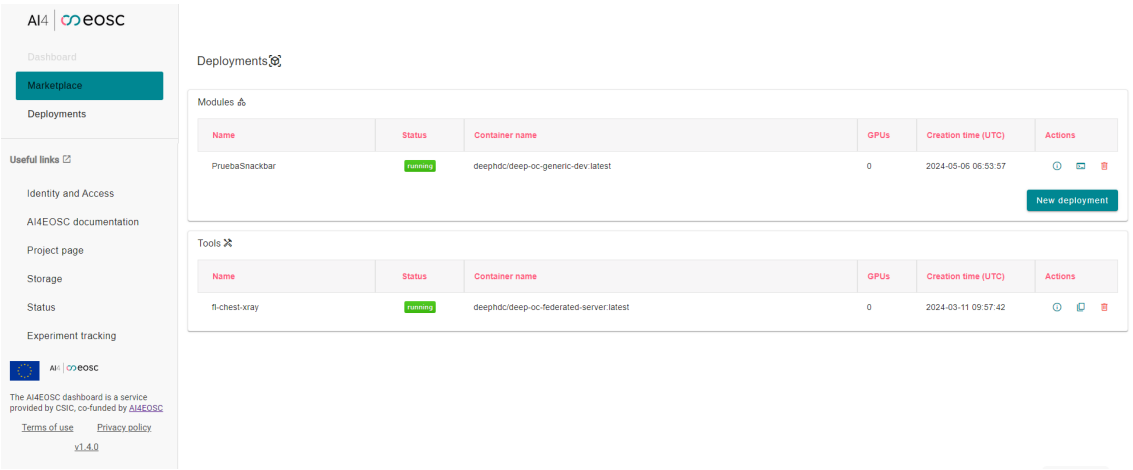

Figura 2: Versión inicial del Dashboard de AI4EOSC (Sección Deployments)

Para facilitar aún más el acceso de los usuarios a la plataforma, la aplicación web está integrada con la Infraestructura de Autenticación y Autorización (AAI) de EGI [22]. Esto significa que los usuarios pueden reutilizar sus credenciales actuales utilizadas para otros proyectos de EOSC y solo tienen que solicitar aprobación para

ser parte de la organización virtual de AI4EOSC para poder tener acceso a la funcionalidad completa. Una organización virtual (`vo`) es una manera de agrupar a los miembros de un proyecto, cada usuario puede pertenecer a uno o más grupos.

Todos los despliegues realizados sobre la plataforma se ejecutan sobre un Community Cloud, formado por centros de datos de los distintos participantes del proyecto. Actualmente la infraestructura soporta únicamente el centro de datos del IFCA, sin embargo, se están realizando tareas de ampliación para federar el clúster y expandirlo sobre el resto de centros del proyecto. Esta versión estable que solo contiene el CPD del IFCA, es la que utiliza la API del entorno de producción (prod). Sin embargo, el entorno de desarrollo (dev) sí que apunta al clúster federado.

3. Requisitos

El proceso formal de captura de requisitos ya se había realizado al comienzo del proyecto, por lo que este capítulo únicamente recoge los requisitos funcionales y no funcionales de este trabajo de fin de máster, los cuales se basan en los propósitos enumerados en el apartado de Objetivos.

3.1. Requisitos funcionales

Los requisitos funcionales se centran en cómo debe comportarse el sistema, es decir, definen qué funcionalidades específicas debe tener. En este caso, se consideran requisitos funcionales las dos nuevas características implementadas en la plataforma: la sección de estadísticas y la inclusión de claves de acceso en los despliegues de aprendizaje federado. En total hay 31 requisitos funcionales que se documentaron en base a historias de usuario. A continuación se muestran algunos ejemplos, las tablas completas se pueden encontrar en el Apéndice A.

En primer lugar, la sección de estadísticas tiene como objetivo informar a los usuarios sobre las características del clúster y su uso, de manera que puedan conocer datos acerca de cuántos recursos están disponibles, cuántos están siendo utilizados, y qué uso que han hecho ellos de dichos recursos, entre otras métricas (Tabla 1).

ID	Historia de Usuario
US-02	Visualizar el uso actual de recursos del usuario
Yo, como usuario, quiero ver cuántos recursos estoy utilizando actualmente con mis despliegues. Concretamente quiero tener información sobre el número de jobs, CPUs, frecuencia de las CPUs, memoria RAM, disco y GPUs. Esta información me permitiría tener un control de mi uso de recursos sobre la plataforma.	
US-05	Visualizar el uso agregado de los recursos utilizados por el clúster y el usuario
Yo, como usuario, quiero saber cuántos recursos he consumido en total respecto al agregado de lo que han utilizado todos los usuarios de la plataforma. Esta información me resultaría de interés para saber qué uso estoy haciendo yo comparado con lo que se ha utilizado en total en la plataforma.	

Tabla 1: Extracto de las Historias de Usuario (Estadísticas)

A partir de estas historias de usuario, se extrajeron los requisitos. La Tabla 2 muestra algunos de ellos, identificados por un ID y la historia de usuario (US) a la que pertenecen.

ID	US	Requisito
RF-01	US-01, US-03	Ver el número de CPUs libres y en uso del clúster.
RF-04	US-01	Ver el número de GPUs libres y en uso del clúster por tipo.
RF-23	US-05	Ver el uso agregado de memoria RAM del usuario frente al clúster.

Tabla 2: Extracto de los Requisitos Funcionales (Estadísticas)

En segundo lugar, se quería añadir una capa de seguridad a los servidores de aprendizaje federado. Cuando un usuario despliega un servidor de este tipo, automáticamente se genera un token o clave por defecto, el cual puede ser utilizado por cualquier participante para poder formar parte del aprendizaje. Sin embargo, no basta con tener una única clave, sino que es más conveniente asignar un token diferente a cada participante o grupo de participantes y poder gestionarlo de manera individual. Por ejemplo, si se utilizara únicamente la clave por defecto y se quisiera eliminar la participación de un usuario, habría que crear otra clave nueva y repartirla entre todos los participantes. Lo mismo ocurre si un usuario vulnera el token, todos los demás participantes se verían afectados por ello.

Para evitar estos problemas, el funcionamiento debe ser el siguiente: una vez creado el servidor, el usuario debe poder gestionar las claves secretas para comunicar a cada participante qué secreto o clave de acceso tiene que utilizar para poder participar en el entrenamiento. De esta manera, el usuario encargado del servidor puede añadir nuevos secretos para incluir a nuevos participantes, y revocar aquellos que no estén siendo utilizados o que se hayan visto vulnerados. A partir de esto surgen nuevas historias de usuario (Tabla 3) y requisitos (Tabla 4).

ID	Historia de Usuario
US-07	Listar las claves de acceso
Yo, como usuario, quiero ver cuáles son las claves de acceso asociadas a los servidores de aprendizaje federado que he creado. Deseo saber cuál es el identificador de cada secreto y cuál es su valor. De esta manera, puedo consultar y acceder fácilmente a las claves para para poder enviárselas a los distintos participantes.	
US-08	Crear una clave de acceso
Yo, como usuario, quiero poder crear nuevas claves de acceso a partir de un identificador asignado por mi. De esta manera, podré crear una clave diferente para cada participante. También podré crear secretos de prueba para experimentar y probar mi servidor.	

Tabla 3: Extracto de las Historias de Usuario (Claves de Acceso)

ID	US	Requisito
RF-28	US-08	Crear una nueva clave de acceso de un servidor de aprendizaje federado, creado por el usuario, a partir de un identificador.
RF-30	US-07, US-10	Ver el valor secreto de una clave de acceso perteneciente a un servidor de aprendizaje federado creado por el usuario.

Tabla 4: Requisitos Funcionales (Claves de Acceso)

3.2. Requisitos no funcionales

Los requisitos no funcionales describen cómo debe ser el sistema en términos de calidad y rendimiento. En este apartado se presenta la definición y clasificación de estos requisitos, las cuales se basan en el estándar ISO 25010 [23]. Este modelo se utiliza para evaluar la calidad del producto, es decir, el grado en que dicho producto satisface los requisitos de sus usuarios, que en este caso serán los investigadores que hagan uso de la plataforma AI4EOSC para desplegar y entrenar sus modelos de inteligencia artificial. Teniendo en cuenta este modelo y los objetivos del proyecto, la Tabla 5 plasma una parte de los requisitos no funcionales extraídos. La tabla completa se encuentra en el Apéndice A.

ID	Descripción
RN-02	Las pantallas deben tener un tiempo de carga medio inferior a 3 segundos
Las llamadas a la API deben ser lo más ágiles posibles, de manera que la experiencia del usuario al interactuar con la web sea agradable. Esto se corresponde con la característica de comportamiento temporal, dentro de la categoría de eficiencia del desempeño, la cual representa el comportamiento del producto en la realización de sus funciones dentro de unos parámetros de tiempo y rendimiento específicos.	
RN-03	La aplicación debe tener fácil navegabilidad
Con el objetivo de facilitar al usuario la utilización y navegabilidad por la aplicación web, se debe seguir un esquema de colores fijo y un posicionamiento de elementos consistente. Se pretende que el usuario pueda utilizar la plataforma con facilidad, incluyendo así las subcaracterísticas de aprendizabilidad y operabilidad de la clase capacidad de interacción.	

Tabla 5: Tabla de Requisitos No Funcionales

Cabe aclarar que los tiempos establecidos en el requisito temporal se basan en estudios realizados sobre los tiempos de carga de las páginas web en relación con la satisfacción de los usuarios, los cuales concluyen que el 53 % de los visitantes abandonan una página que tarda más de tres segundos en cargar [24].

4. Arquitectura y Diseño

La mayoría de aportaciones de este trabajo se centran en la arquitectura y diseño del frontend de la plataforma, no obstante, también se hicieron pequeñas contribuciones a la API. En este capítulo se detallan dichos aportes.

4.1. Arquitectura de la aplicación

La arquitectura de la aplicación está definida utilizando el modelo C4 [25], una técnica de notación gráfica que se fundamenta en UML (Unified Modeling Language) y ERD (Entity Relationship Diagram). Está basada en la descomposición estructural del software en cuatro niveles (de mayor a menor nivel de abstracción):

- Nivel 1: diagrama de contexto del sistema → ofrece una visión general del sistema completo que sirve como punto de partida para comprender su organización.

- Nivel 2: diagrama de contenedor → se corresponde con la vista en detalle de cada uno de los elementos que conforman el sistema.
- Nivel 3: diagrama de componente → muestra en detalle la información de un contenedor concreto, indicando los componentes que lo forman.
- Nivel 4: diagrama de código → indica cómo está implementado un componente. Cabe destacar que este tipo de diagramas se utilizan exclusivamente en componentes con un alto grado de importancia y/o complejidad.

De entre todos ellos, me encargué de hacer el diagrama de nivel 3 correspondiente al frontend de la aplicación, y los diagramas de nivel 4 que lo forman. En el diagrama de componente (Figura 3) se pueden observar los distintos componentes que conforman la web, los cuales se corresponden con el concepto de 'módulo' de Angular. Los módulos representan un conjunto de código dedicado a un ámbito o funcionalidad específica de la aplicación.

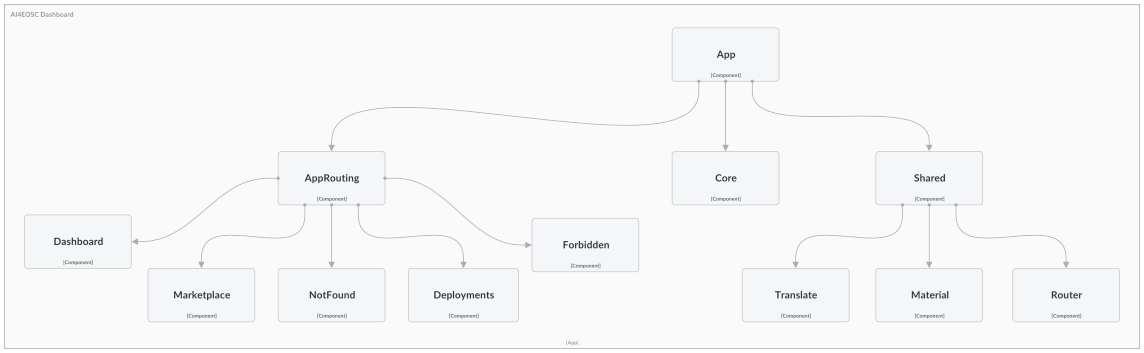

Figura 3: Modelo C4 (Nivel 3)

En primer lugar, `App` es el módulo raíz que describe cómo se relacionan todos los elementos que conforman la aplicación. Dentro de él encontramos el módulo `AppRouting`, que se encarga de definir y gestionar las rutas de la aplicación. Cabe destacar que este utiliza la técnica *lazy loading*, la cual permite cargar módulos solo cuando son necesarios, es decir, cuando el usuario navega a una ruta específica que requiere ese módulo. Esto mejora significativamente el rendimiento inicial de la aplicación, ya que no todos los módulos se cargan al principio, sino solo cuando el usuario los necesita. Dichas rutas determinan cómo se navega dentro de la aplicación, especificando qué componentes se deben mostrar cuando se accede a ciertas URLs. Los módulos `NotFound` y `Forbidden` son bastante sencillos: el primero de ellos se encarga de indicar que se está intentando acceder a una sección que no existe, y el segundo de informar que se está intentando acceder a un recurso protegido al que no se tiene acceso. Las interfaces que los forman son simples, por lo que en este caso no se consideró necesario llegar al diagrama de nivel 4. Por otro lado, `Dashboard`, `Marketplace` y `Deployments`, son más complejos, por lo que estos sí que tienen un diagrama de código asociado. La Figura 4 muestra un ejemplo de uno de ellos, que en este caso representa cómo está organizado el código dentro del componente `Dashboard`. En ella se pueden observar los componentes utilizados para las diferentes interfaces del `Dashboard`, los servicios que se utilizan para obtener los datos a mostrar, y las dependencias externas con otros módulos de la aplicación. En este caso esta funcionalidad se organiza en pestañas (tabs), y en cada una de ellas se muestra un tipo de información diferente. Para ello se hace uso de otros

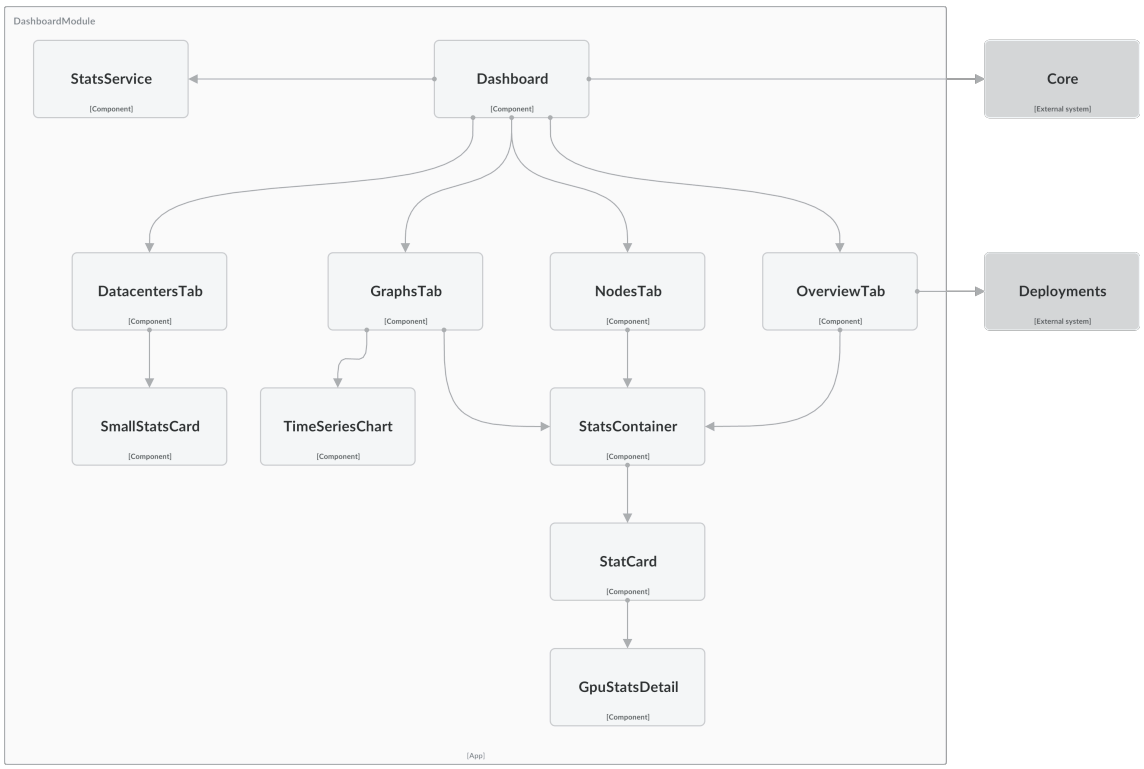

Figura 4: Modelo C4 Dashboard (Nivel 4)

componentes reutilizables que sirven para mostrar varios tipos de gráficas y mapas, entre otros.

Finalmente, el componente `Core` contiene los servicios de autorización de la aplicación, y el módulo `Shared`, incluye, como su propio nombre indica, todas aquellas utilidades cuyo uso es común y recurrente en distintas partes de la aplicación, para que se puedan utilizar en cualquiera de ellas. Esto incluye módulos para la traducción de texto (`Translate`), para navegar por las distintas rutas de la aplicación (`Router`) y para hacer uso de los componentes gráficos proporcionados por la biblioteca de Angular Material (`Material`). Los detalles de implementación y el uso de estos componentes se detallan más adelante, en el Apartado 5.

Este tipo de diagramas, además de tener fines de documentación, sirven para mejorar el diseño y la organización de la aplicación. Ayudan a tener una imagen visual de cómo está organizado el código, gracias a la cual se pueden percibir fallos o posibles mejoras a aplicar sobre él. Por ejemplo, durante la realización de estos modelos, pude ver que en el diagrama de nivel 4 del `Marketplace` se podía refactorizar parte del código para mejorar la distribución de los componentes. Este proceso junto con el resto de diagramas del proyecto se detallan en el Apéndice B.

4.2. Diseño de la API REST

La API está compuesta por varios recursos: secretos, módulos, herramientas y estadísticas. A continuación la Tabla 6 muestra el diseño de los endpoints utilizados en las nuevas funcionalidades, sus recursos y cómo pueden ser manipulados. De entre todos ellos, las aportaciones que se hicieron forman parte de `/stats`. Concretamente, se creó un nuevo endpoint que permite recoger datos e información relevantes sobre los recursos del clúster (`ClusterStats`). Únicamente dispone del método GET, y la respuesta va a ser 200 (Ok), siempre y cuando la `vo` introducida como parámetro

exista y el usuario pertenezca a ella, en caso contrario devuelve un código de error 401 (Unauthorized). En el caso de que no hubiera información disponible, la lista de centros de datos aparecería vacía y todos los recursos del clúster tendrían valor 0. El Apéndice C contiene una representación del objeto `ClusterStats`. Tal y como se puede observar, el servicio devuelve los datos en formato JSON, que es la práctica habitual hoy en día.

Recurso	URI	Métodos	Respuestas HTTP
UserStats	/stats/user	GET	200, 401
ClusterStats	/stats/cluster	GET	200, 401
Deployments (Modules)	/deployments/modules	GET	200, 401
Deployments (Tools)	/deployments/tools	GET	200, 401
Secrets	/secrets	GET	200, 401
		POST	200, 401
		DELETE	200, 401

Tabla 6: Diseño API REST

El resto de endpoints, que ya estaban desarrollados, funcionan de la siguiente manera:

- `/stats/user`: el método GET devuelve un objeto de tipo `UserStats` que contiene los datos referentes al uso agregado del usuario (CPUs, GPUs, frecuencia de CPUs, memoria RAM y disco, consumidos) y al uso agregado del clúster (ídem). Además, también se incluyen los consumos totales del clúster de los últimos 3 meses (los recursos mencionados anteriormente más el número de jobs ejecutados y el número de jobs en espera). Recibe como parámetro `vo`, que indica la organización virtual de la cual se quieren obtener las estadísticas. Este método devuelve un código 200 (Ok) siempre y cuando la `vo` introducida como parámetro exista y el usuario pertenezca a ella, en caso contrario devuelve un código de error 401 (Unauthorized).

- `/deployments/modules`: el método GET devuelve todos los deployments 'vivos' de tipo módulo que ha desplegado el usuario. Tiene dos parámetros opcionales: `vo` (organización virtual de la cual se quieren obtener los deployments del usuario) y `full_info` (booleano que determina si se quiere obtener la información completa de los deployments o solo un resumen, por defecto su valor es falso). Devuelve un código de respuesta 200 (Ok) con la lista de módulos. Si se envía como parámetro una `vo` a la que el usuario no pertenece, devuelve un código 401 (Unauthorized).

- `/deployments/tools`: funciona de la misma manera que los deployments de tipo módulo, pero en este caso se devuelven aquellos de tipo herramienta.

- `/secrets`: este endpoint es el encargado de gestionar los secretos de los despliegues de aprendizaje federado de los usuarios. El método GET permite obtener el listado de secretos, tanto de un deployment concreto, como de todos los del usuario. Recibe el parámetro obligatorio `vo`, que funciona de la misma manera

que en los casos anteriores, y el parámetro opcional `subpath`, que determina de qué deployment se quieren obtener las claves. Si se deja vacío, se devuelven todos los secretos del usuario. Este método devuelve un código 200 (Ok) siempre y cuando la `vo` introducida como parámetro exista y el usuario pertenezca a ella, en caso contrario devuelve un código de error 401 (Unauthorized). El método POST sirve para crear claves nuevas y, a parte del parámetro `vo`, tiene el parámetro obligatorio `subpath`, que indica dónde quiere almacenarse el secreto. Asimismo, se tiene que enviar en el cuerpo de la petición el valor del token correspondiente. Finalmente, el método DELETE tiene los mismos parámetros de entrada que el POST y devuelve un código 200 (Ok) si se ha eliminado el token correctamente, o un código 401 (Unauthorized) si la `vo` introducida no pertenece al usuario.

4.3. Diseño de la interfaz gráfica

Todas las decisiones sobre la interfaz y sus componentes se han tomado siguiendo las guías de estilo de los proyectos AI4EOSC e iMagine, en las cuales se especifica qué colores se tienen que utilizar, cómo se deben identificar los botones de acción, y cómo situar ciertos elementos gráficos, entre otros aspectos. Además, siguiendo el principio propuesto en los requisitos no funcionales del proyecto de facilitar la navegación del usuario, se ha seguido una organización consistente y se ha procurado evitar el uso de scroll en las pantallas de la aplicación siempre que fuera posible. A continuación, se especifican las interfaces implementadas y la navegación de la capa de presentación del sistema. Las figuras principales se muestran en este apartado y el resto se encuentran en el Apéndice D.

4.3.1. Dashboard

La sección del Dashboard, a la cual se puede acceder desde el menú lateral, se encuentra divida en 4 pestañas que agrupan la información de los recursos en distintos grupos. En primer lugar está la pestaña de 'Overview' (Figura 5) que, como su propio nombre indica, ofrece información general sobre los recursos del clúster (Cluster Usage Overview).

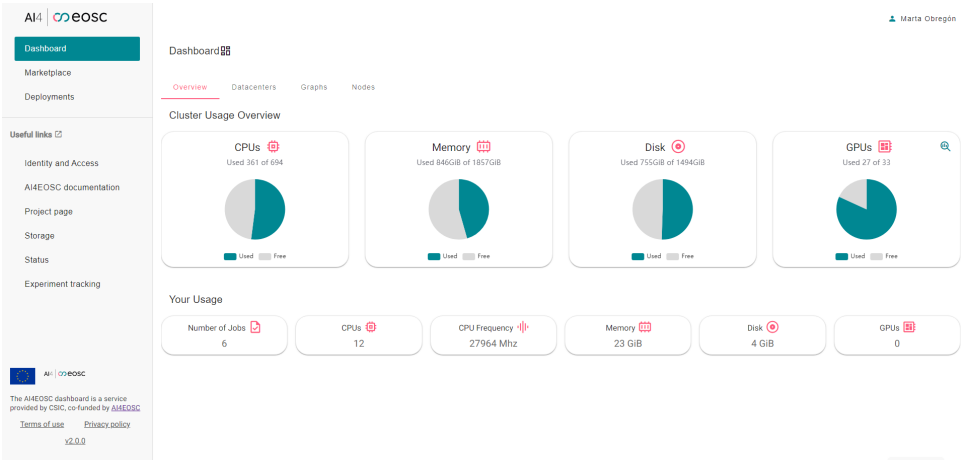

Figura 5: Sección Overview

Esta información se muestra mediante gráficos circulares que indican la cantidad de recursos que están siendo utilizados frente a cuántos están disponibles. Debajo

de esto, se muestra información detallada sobre los recursos que está utilizando actualmente el usuario (Your Usage). Además, la tarjeta que muestra el número de GPUs tiene un botón adicional que permite visualizar los datos de uso según el tipo de GPU (Figura 6). Esta información es bastante relevante, ya que en el ámbito de la inteligencia artificial y el entrenamiento de redes, las GPUs son el elemento más importante y demandado por los usuarios. Es por esto que no se muestra este desglose en el resto de recursos.

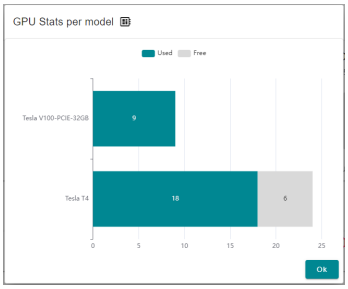

Figura 6: Modal de GPUs

Por simplicidad y para facilitar la visualización de las interfaces, a partir de aquí se obvia la barra lateral y el encabezado de la página, ya que no cambian. En la pestaña 'Datacenters' se encuentra un mapa interactivo (Figura 7).

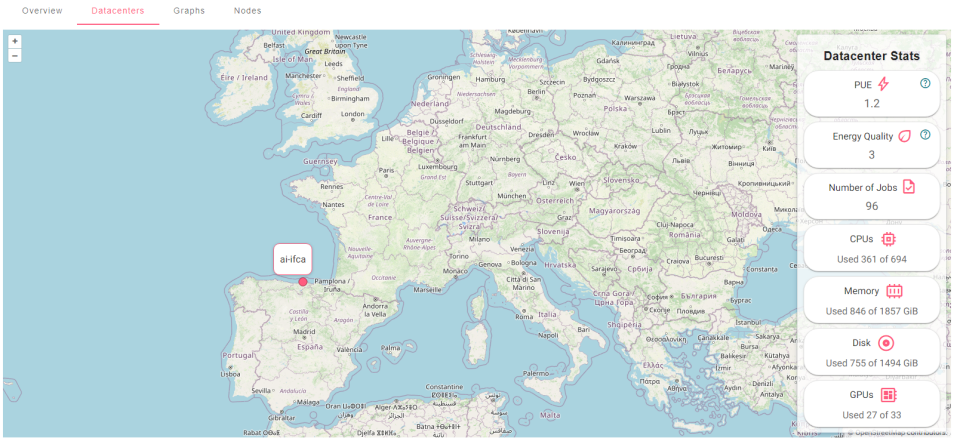

Figura 7: Sección Datacenters

En este mapa se coloca un punto por cada centro de datos perteneciente al clúster, en este caso solo se muestra uno (ai-ifca), porque, como se comentó en previos apartados, el clúster federado que reúne a varios datacenters europeos que participan en el proyecto aun está en desarrollo. Al pulsar sobre cualquiera de estos puntos, se abre una barra lateral que muestra información relevante sobre el centro. A parte de los recursos mencionados anteriormente, se observan dos métricas nuevas:

- PUE (Power Usage Effectiveness): métrica que determina la eficiencia de la energía de un centro de datos. Se calcula dividiendo el consumo de energía total del centro entre el consumo dedicado exclusivamente al equipamiento informático.
- Calidad de la Energía: hace referencia a cómo de "buena" es la energía empleada en el centro de datos. Representa cuántas emisiones de carbón emite, cuanto más bajo sea el valor, mejor es la calidad de la energía.

Con el objetivo de facilitar la comprensión de este tipo de información, en los casos en los que se muestran conceptos más técnicos o que requieren de una explicación más detallada, se ha colocado un icono de interrogación. De esta manera, cuando el usuario pasa el ratón sobre este icono, se muestra un cuadro de información emergente en el que se explica qué significa dicho concepto.

A continuación se encuentra la pestaña de gráficas, la cual está dividida en dos secciones (Figura 8). En la primera de ellas 'AI4EOSC Usage Over Time' se agrupan 7 gráficas que muestran la utilización de los recursos a lo largo del tiempo, concretamente en los últimos tres meses. En la parte inferior, se indica el uso agregado que el usuario ha hecho de los recursos de la plataforma frente al uso agregado total de la misma.

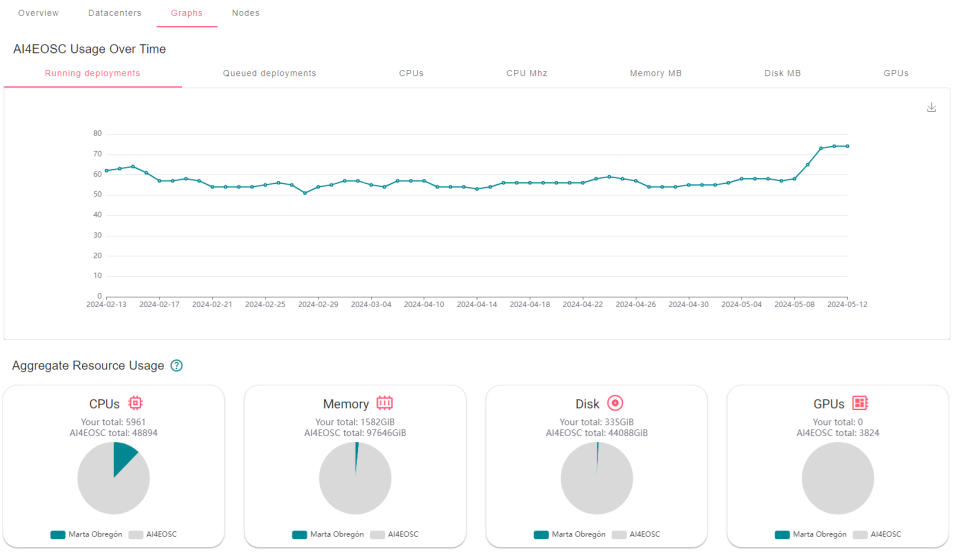

Figura 8: Sección de Gráficas

Finalmente, la pestaña de nodos muestra los recursos de todos los nodos pertenecientes al clúster (Figura 9).

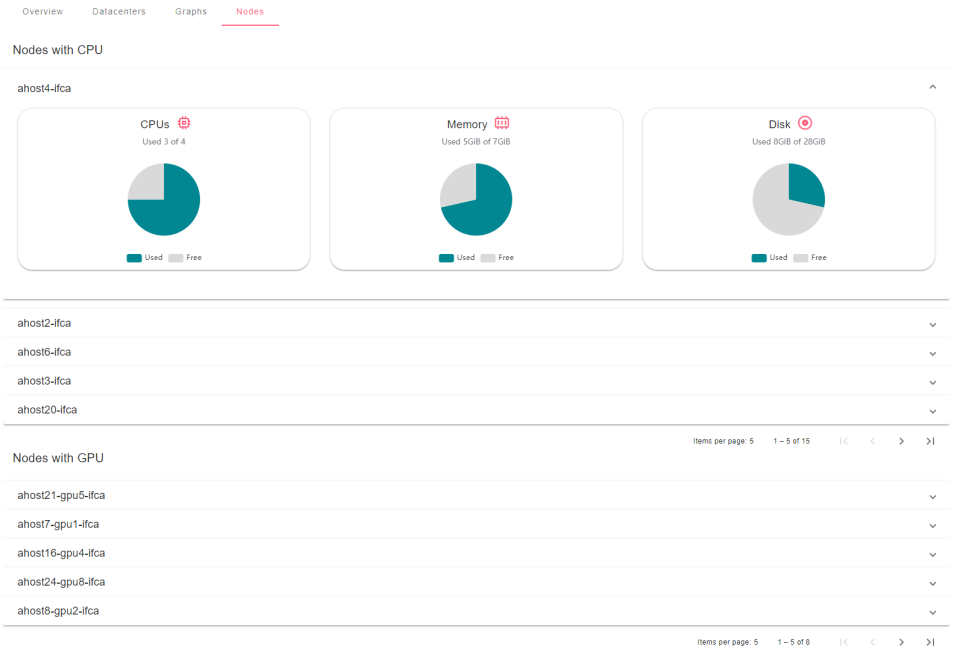

Figura 9: Sección de Nodos

Se encuentran divididos en aquellos que cuentan con CPU y aquellos que también disponen de GPU. En este caso, como la información a mostrar es extensa, no se ha podido reducir el tamaño de los datos a una sola pantalla, sin embargo, se han aplicado componentes desplegables y se han paginado los nodos para facilitar su visualización. De este modo, cuando el usuario pulsa sobre un nodo se abre un desplegable con su información, pudiendo navegar entre la lista de nodos utilizando las flechas de paginación.

4.3.2. Claves de acceso

Por otro lado, encontramos las interfaces pertenecientes a la gestión de claves de acceso de los servidores de aprendizaje federado. En este caso, una vez el usuario ha creado un servidor de este tipo, puede gestionarlo desde la pestaña 'Deployments' del menú de navegación fijo situado a la izquierda de la pantalla (Figura 10).

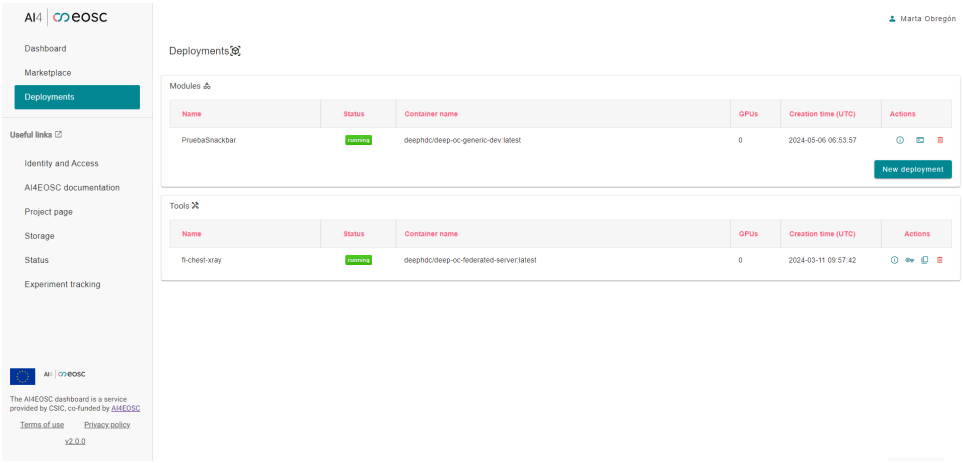

Figura 10: Lista de Despliegues

Dentro de las posibles acciones a realizar sobre las herramientas desplegadas, la opción de gestionar las claves secretas se encuentra en la segunda posición, representada con una llave. Cuando el usuario pulsa sobre ella, se abre un modal que muestra un listado paginado de todos los tokens del despliegue, con varias opciones para gestionarlos (Figura 11).

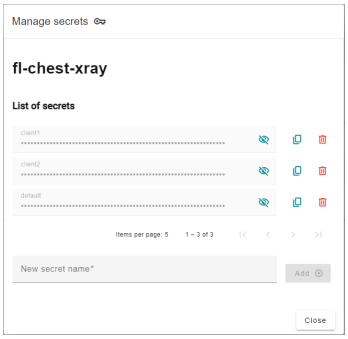

Figura 11: Modal para gestionar las claves secretas

En primer lugar, cada clave tiene un identificador, el contenido de la clave en sí, y tres botones que permiten visualizar/ocultar el secreto, copiarlo en el portapapeles o eliminarlo. A continuación, se muestra un cuadro de texto donde los usuarios pueden introducir un nuevo identificador para crear un secreto pulsando al botón 'Add'. En

este caso los usuarios únicamente tienen que escribir el identificador, ya que el secreto se genera automáticamente. Las únicas restricciones son que el identificador no debe contener únicamente espacios y que no puede repetirse. En caso de que se intente crear uno ya existente, el botón de añadir se bloquea y aparece un mensaje de error que informa al usuario del problema. Cuando un usuario intenta eliminar un token, se muestra un mensaje de confirmación para ratificar la acción (Figura 12).

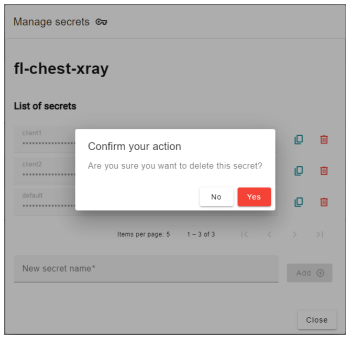

Figura 12: Eliminar una clave secreta

En el caso de que solo quede una clave y esta se intente eliminar, el mensaje de confirmación varía ligeramente, ya que indica al usuario que algunas herramientas necesitan como mínimo una clave para funcionar correctamente. Por último, cabe destacar, que siempre que se añade o elimina un token, aparece un mensaje emergente de color verde informando al usuario de que la acción se ha realizado correctamente. Si se produce algún error durante el proceso, también se muestra un mensaje indicando el problema, en este caso de color rojo.

5. Implementación

En este capítulo se detallan los aspectos más relevantes de la implementación de los distintos componentes de la aplicación. El código del frontend se puede encontrar en el repositorio `ai4-dashboard` [26], y el del backend en el respositorio `ai4-papi` [27].

5.1. Frontend

En este apartado se detallan varios ejemplos que representan cómo se han implementado las distintas partes que conforman la aplicación Angular del proyecto. Este tipo de aplicaciones se organizan en varios elementos clave: módulos, componentes y servicios, y otros como directivas, pipes, y rutas. Basándonos en la arquitectura y diseño presentados en el Apartado 4, a continuación se muestran distintos ejemplos de dichos elementos.

En primer lugar encontramos los módulos, los cuales se encargan de agrupar y organizar el código de la aplicación. Cada uno de ellos puede contener componentes, directivas, pipes y servicios. En el ejemplo detallado en el Código 1, el módulo `DashboardModule` engloba los distintos componentes encargados de la funcionalidad de las estadísticas, los cuales se definen con la etiqueta `declarations`. Por otro lado, define el resto de módulos que necesita para funcionar correctamente en la etiqueta `imports`.

```typescript
@NgModule({
    declarations: [
        TimeSeriesChartComponent,
        DashboardComponent,
        StatCardComponent,
        StatsContainerComponent,
        OverviewTabComponent,
        GraphsTabComponent,
        NodesTabComponent,
        GpuStatsDetailComponent,
        DatacentersTabComponent,
        SmallStatsCardComponent,
    ],
    imports: [
        CommonModule,
        DashboardRoutingModule,
        NgxEchartsModule.forChild(),
        SharedModule,
    ],
})
export class DashboardModule {}
```

Código 1: Contenido de `dashboard.module.ts`

Por su parte, los componentes representan una parte de la interfaz de usuario, es decir, encapsulan la lógica, los datos y la presentación de una parte específica de la aplicación.

```typescript
@Component({
    selector: 'app-stat-card',
    templateUrl: './stat-card.component.html',
    styleUrls: ['./stat-card.component.scss'],
})
export class StatCardComponent implements OnInit {
    @Input() title = '';
    @Input() usedValue = 0;
    @Input() totalValue = 0;
    @Input() iconName = '';
    @Input() memoryUnit?: string;
    @Input() gpuPerModelCluster?: GpuStats[];
    @Input() usedLabel? = 'Used';
    @Input() freeLabel? = 'Free';

    constructor(
        private changeDetectorRef: ChangeDetectorRef,
        private media: MediaMatcher,
        public dialog: MatDialog,
        public confirmationDialog: MatDialog
    ) {
        this.mobileQuery = this.media.matchMedia('(max-width: 650px)');
        this._mobileQueryListener = () => changeDetectorRef.detectChanges();
        this.mobileQuery.addEventListener('change', this._mobileQueryListener);
    }

    mobileQuery: MediaQueryList;
    private _mobileQueryListener: () => void;

    ngOnInit(): void {
        this.setColours();
    }
}
```

Código 2: Fragmento de `stat-card.component.ts`

Teniendo como ejemplo el Código 2, cada componente en Angular está compuesto por cuatro partes principales:

1. Clase del Componente: parte lógica del componente escrita en TypeScript. Contiene propiedades y métodos que definen su comportamiento. En este caso encontramos propiedades como `mobileQuery` y métodos como `setColours`.

2. Decorador `@Component`: define los metadatos del componente, su selector (nombre del elemento HTML que se utilizará para instanciar y renderizar el componente), la plantilla que utiliza y los estilos asociados a esta.

3. Plantilla (`templateUrl`): HTML que define la estructura de la vista del componente. En este caso, este componente muestra una tarjeta con un título, un icono y un botón. Este último se muestra de manera condicional, lo que significa que solo aparece en el caso de que la variable `gpuPerModelCluster` esté definida, para lo cual se utiliza la directiva de Angular `*ngIf`. En la parte inferior de la tarjeta se muestra una gráfica. El Código 3 muestra el ejemplo correspondiente a `StatCardComponent`.

```
<mat-card class="card">
    <div class="header">
        <p class="title-text">{{ title }}</p>
        <mat-icon class="icon">{{ iconName }}</mat-icon>
    </div>
    <button
        *ngIf="gpuPerModelCluster"
        id="detail-button"
        mat-icon-button
        matTooltip="{{ 'DASHBOARD.GPU-PER-MODEL' | translate }}"
        [matTooltipPosition]="'above'"
        (click)="openDetailGpuStats()">
        <mat-icon color="accent">
            <span class="material-symbols-outlined">
                <span class="material-symbols-outlined">
                    <span class="material-symbols-outlined"> search_insights </span>
                </span>
            </span>
        </mat-icon>
    </button>
    <div id="canvas" echarts [options]="chartOptionsCommon" class="pie-graph"></div>
</mat-card>
```

Código 3: Fragmento de `stat-card.component.html`

4. Estilos (`styleUrls`): hojas de estilo específicas del componente que definen su apariencia (Código 4).

```
.card {
    border-radius: 25px;
    display: flex;
    flex-direction: column;
    align-items: flex-start;
    height: 270px;
}
.header {
    display: flex;
    flex-direction: row;
    align-items: center;
    justify-content: center;
    width: 100%;
    padding-top: 0.5em;
}
```

Código 4: Fragmento de `stat-card.component.scss`

Los componentes en Angular pueden comunicarse entre sí de varias maneras, en este ejemplo se observa que el componente hijo `StatCardComponent` se comunica con su componente padre `StatsContainerComponent` (que contiene el selector `app-stat-card` dentro de su propia plantilla), a través de la etiqueta `@Input()`. Esta permite que el componente padre le envíe al hijo la información que tiene que

mostrar en la tarjeta, incluyendo entre otras cosas, la variable `gpuPerModelCluster` mencionada anteriormente.

Además de los módulos y componentes también están los servicios, los cuales encapsulan la lógica de negocio y la comunicación con APIs, entre otros. El Código 5 muestra como ejemplo una de las funciones del servicio `SecretsService`, encargado de comunicarse con el endpoint `/secrets` de la API.

```
@Injectable({
    providedIn: 'root',
})
export class SecretsService {
    constructor(
        private http: HttpClient,
        private appConfigService: AppConfigService
    ) {}

    getSecrets(subpath: string): Observable<Secret[]> {
        const url = `${base}${endpoints.secrets}`;
        const params = new HttpParams().set('vo', this.appConfigService.voName).set('subpath', subpath);
        return this.http.get<Array<Secret>>(url, {params: params,});
    }
}
```

Código 5: Fragmento de `secrets.service.ts`

Tomando como punto de partida el fragmento de código anterior, a continuación se muestran los aspectos generales más relevantes a la hora de implementar un servicio:

- El decorador `@Injectable` indica que este servicio puede ser inyectado en otros componentes o servicios. La propiedad `providedIn: 'root'` significa que el servicio está disponible en toda la aplicación.
- La clase `SecretsService` se declara incluyendo `export` para que pueda ser importada y utilizada en otros archivos del proyecto. Además, el constructor inyecta dos dependencias:
  - `HttpClient`: servicio de Angular que permite realizar solicitudes HTTP a servidores remotos.
  - `AppConfigService`: servicio que proporciona la configuración del usuario registrado. En este caso, se utiliza para obtener a qué organizaciones pertenece.
- El método `getSecrets` realiza una solicitud de tipo GET para obtener la lista de secretos, y recibe los parámetros `vo` y `subpath`, detallados en el Apartado 4.2. Este método retorna un observable [28] de un array de objetos `Secret`. Un observable es un objeto que se utiliza para gestionar flujos de datos asíncronos y emitir cambios a los componentes suscritos, y en este caso está representando una respuesta HTTP.

El Código 6 muestra un ejemplo de cómo un componente llama al método del servicio `SecretsService` para obtener un listado con las claves de un despliegue. En este caso, el componente tiene que subscribirse al observable, lo que significa que se queda 'escuchando' al valor que emita cuando termine su ejecución, para así poder reaccionar a él. La función `next` se invoca cada vez que el observable emite un valor y se utiliza para procesar los datos recibidos. Por el contrario, la función `error` se

invoca si el observable emite un error, y en este caso finaliza la animación de carga. Por otro lado, cabe destacar que el interceptor `HttpErrorInterceptor` es el que se encarga de mostrar mensajes informativos al usuario del estado de las operaciones. Este tipo de elementos permiten interceptar y manipular las solicitudes HTTP y sus respuestas antes de que lleguen a su destino final, en este caso el componente. Es por esto que las funciones `next` y `error` no muestran ningún mensaje por pantalla.

```
getSecrets() {
    this.isLoading = true;
    const subpath = '/deployments/' + this.data.uuid + '/federated/';
    this.secrets = [];

    this.secretsService.getSecrets(subpath).subscribe({
        next: (secrets) => {
            for (let i = 0; i < Object.values(secrets).length; i++) {
                const secret: SecretField = {
                    name: Object.keys(secrets)[i].substring(
                        Object.keys(secrets)[i].lastIndexOf('/') + 1
                    ),
                    value: Object.values(secrets)[i].token,
                    hide: true,
                };
                this.secrets.push(secret);
            }
            this.paginatedSecrets = this.secrets.slice(0, this.pageSize);
            this.length = this.secrets.length;
            this.isLoading = false;
        },
        error: () => {
            this.isLoading = false;
        },
    });
}
```

Código 6: Función de `getSecrets` de `secret-management-detail.component.ts`

Por otro lado, a parte de los componentes propios de Angular, también se utilizaron distintas librerías y bibliotecas externas para construir ciertos elementos de la aplicación. En este listado se detallan las más importantes:

- `Angular Material` [29]: biblioteca de componentes de interfaz de usuario. Proporciona una serie de componentes preconstruidos, como botones, formularios, menús, diálogos y tablas. Se utiliza en toda la aplicación, y ha permitido agilizar el desarrollo de nuevos componentes, aportando una apariencia uniforme y atractiva.

- `OpenLayers` [30]: biblioteca de mapas que permite mostrar y manipular datos geoespaciales sobre mapas interactivos dentro de aplicaciones web. Se ha utilizado para generar el mapa que muestra los distintos centros de datos del clúster.

- `ECharts` [31]: biblioteca de visualización de datos que permite crear gráficos interactivos y personalizables, como gráficos de barras, gráficos temporales y gráficos circulares, entre otros. Se ha utilizado principalmente en la sección del `Dashboard` para mostrar las estadísticas.

- `crypto-random-string` [32]: librería que genera cadenas aleatorias seguras criptográficamente. Es útil para generar identificadores únicos, contraseñas temporales y tokens. En este caso se ha utilizado para generar las claves de acceso de los servidores de aprendizaje federado.

- `RxJS` [33]: librería para la programación reactiva que utiliza el concepto de Observable para facilitar la creación y gestión de código asíncrono. Se utiliza a lo largo de toda la aplicación para manejar eventos y peticiones HTTP, entre otros.

5.2. Backend

El backend se desarrolló utilizando Python junto con FastAPI [34] y FastAPI Utilities [35]. FastAPI es un framework que permite construir APIs en Python de manera sencilla sin tener que escribir *boilerplate* [36]. Por otro lado, FastAPI Utils se trata de un paquete que incluye una colección de herramientas diseñadas para mejorar y simplificar el desarrollo de aplicaciones basadas en FastAPI. Estas utilidades están orientadas a reducir el código repetitivo y a facilitar la implementación de funcionalidades comunes.

Tal y como se mencionó en el Apartado 4.2, se creó un nuevo endpoint para poder obtener la información de los recursos del clúster. Obtener estos datos es una tarea costosa computacionalmente hablando, ya que para ello es necesario cargar todos los centros de datos y recorrer todos sus nodos y jobs existentes. Aunque de momento solo se trabaje con el centro de datos del IFCA, hay que tener en cuenta que en el futuro se desea añadir más centros, por lo que este proceso aumentará de manera exponencial cuantos más centros de datos haya y más jobs se estén ejecutando en ellos. Por esta razón, y teniendo en cuenta que uno de los requisitos no funcionales era que la media de tiempos de carga no superara los 3 segundos, se añadieron mejoras para reducir al máximo posible estas esperas. En primer lugar, se creó una tarea en segundo plano que se encarga de recalcular estos datos cada 30 segundos. Para ello se hizo uso de la directiva `@asynccontextmanager` de FastAPI [37], que permite definir la lógica que debe ejecutarse antes de que la aplicación se inicie (`on startup`) y antes de finalizar (`on shutdown`). En este caso, se añadió una función que calcula las estadísticas del clúster en el startup, lo que significa que se ejecutará una vez antes de que la aplicación comience a recibir peticiones (Código 7).

```python
@asynccontextmanager
async def lifespan(app: fastapi.FastAPI):
    # on startup
    await get_cluster_stats_thread()
    yield
    # on shutdown (nothing to do)
```

Código 7: Uso de la directiva `@asynccontextmanager`

A continuación, el Código 8 muestra la definición de la función, la etiqueta `@repeat_every` [38] indica que se llamará a esta función en segundo plano cada 30 segundos.

```python
@repeat_every(seconds=30)
async def get_cluster_stats_thread():
    get_cluster_stats_bg.cache_clear()
    get_cluster_stats_bg()
```

Código 8: Uso de la directiva `@repeat_every`

Dentro del cuerpo del método, `get_cluster_stats_bg` calcula las estadísticas de uso del clúster y almacena estos datos en una cache durante 30 minutos. Primero se limpia la cache, para eliminar posibles datos obsoletos de ejecuciones anteriores, y a continuación se recalculan los datos. El Código 9 representa el pseudocódigo de la función encargada de calcular dichos datos.

```
@cached(cache=TTLCache(maxsize=1024, ttl=30))
def get_cluster_stats_bg():
    carga un csv con la información de los datacenters disponibles (nombre, latitud, longitud, PUE, calidad de la energía)
    carga los nodos
    recorre todos los nodos:
        carga su nombre, número de jobs, cpu total, ram total, disco total, disco en uso
            recorre todos los dispositivos asociados al nodo:
                si el dispositivo es de tipo 'gpu':
                    carga la gpu total
                    carga los modelos de gpu y su gpu total
    carga los jobs que se estan ejecutando
        recorre todos los jobs:
            aumenta el número de jobs corriendo
            carga cpu, ram y gpu en uso
    calcula las estadisticas totales del cluster
    retorna las estadísticas del clúster
```

Código 9: Pseudocódigo de la función `get_cluster_stats_bg`

De esta manera, cada vez que se haga un GET sobre el endpoint que devuelve estas estadísticas (Código 10), este devolverá los datos almacenados en la caché, por lo que no tendrá que recalcularlos y la respuesta será prácticamente inmediata.

```
@router.get("/cluster")
def get_cluster_stats():
   global cluster_stats
   return cluster_stats
```

Código 10: Función `get_cluster_stats`

6. Pruebas y despliegue

Este capítulo describe las diferentes pruebas realizadas durante el desarrollo de la aplicación junto con las mejoras aplicadas sobre el proceso de despliegue. Cabe mencionar que no se desarrollaron pruebas de integración, ya que las pruebas unitarias y end-to-end abarcaban la mayoría de escenarios de prueba. Se consideró que los beneficios que se conseguirían con estas pruebas, no compensaban el esfuerzo requerido para su desarrollo. Además, las interacciones entre los componentes de la aplicación no son complejas, y se decidió dar mayor importancia a las pruebas end-to-end, que están más orientadas al usuario final.

6.1. Pruebas unitarias

En el frontend, las pruebas unitarias se realizaron sobre los componentes utilizando el framework de Jest [39]. En total se realizaron tests sobre los 11 componentes implementados, en las cuales se comprueba que se renderizan correctamente los elementos de cada uno de ellos, y que el funcionamiento al utilizar los botones y cuadros de texto, entre otros, es el esperado. Para la funcionalidad de las estadísticas se desarrollaron tests sobre sus 10 componentes y su servicio, y para las claves de acceso

a los servidores de aprendizaje federado se realizaron pruebas sobre su componente y su respectivo servicio.

- DashboardComponent
- GpuStatsDetailComponent
- StatsCardHorizontalComponent
- StatCardComponent
- StatsContainerComponent
- DatacentersTabComponent
- GraphsTabComponent
- NodesTabComponent
- OverviewTabComponent
- TimeSeriesChartComponent
- StatsService
- SecretManagementDetailComponent
- SecretsService

El Código 11 muestra, a modo de ejemplo, uno de los tests desarrollados sobre el componente `SecretManagementDetail`.

```
const mockedSecrets = { '/deployments/1234/federated/default': { token: '1234' } };

describe('SecretManagementDetailComponent', () => {
    let component: SecretManagementDetailComponent;
    let fixture: ComponentFixture<SecretManagementDetailComponent>;

    beforeEach(async () => {
        await TestBed.configureTestingModule({
            declarations: [SecretManagementDetailComponent],
            imports: [NoopAnimationsModule, HttpClientTestingModule, SharedModule, TranslateModule.forRoot()],
            providers: [
                { provide: AppConfigService, useValue: mockedConfigService },
                { provide: MAT_DIALOG_DATA, useValue: {} },
                { provide: MediaMatcher, useValue: mockedMediaMatcher },
                { provide: SecretsService, useValue: mockedSecretsService },
            ],
        }).compileComponents();

        fixture = TestBed.createComponent(SecretManagementDetailComponent);
        component = fixture.componentInstance;
        component.data = { uuid: '1', name: 'Test' };
        fixture.detectChanges();
    });

    it('should delete a secret if user confirms it', () => {
        const spyConfirmationDialog = jest.spyOn(component.confirmationDialog, 'open')
                                          .mockReturnValue({ afterClosed: () => of(true) } as MatDialogRef<typeof component>);
        const spyDeleteSecret = jest.spyOn(mockedSecretsService, 'deleteSecret');
        const spySnackBar = jest.spyOn(component['_snackBar'], 'open');

        const button = fixture.debugElement.query(By.css('#delete-button'));
        button.nativeElement.click();

        expect(spyConfirmationDialog).toHaveBeenCalledTimes(1);
        expect(spyDeleteSecret).toHaveBeenCalledTimes(1);
        expect(spySnackBar).toHaveBeenCalledWith('Successfully deleted secret with name: default', 'X',
                                                 expect.objectContaining({duration: 3000, panelClass: ['success-snackbar']}));
        expect(fixture.componentInstance.secrets).toEqual([]);
    });
});
```

Código 11: Fragmento de `secret-management-detail.component.spec.ts`

En la función `beforeEach`, que se ejecuta antes de cada test, el método `configure TestingModule` de `TestBed`, se encarga de construir dinámicamente un módulo Angular test que emula la clase `@NgModule`. A esta función se le pasan como parámetros los siguientes metadatos:

- Declarations: el propio componente que se está probando.

- Imports: módulos que utiliza este componente.
- Providers: proveedores de los servicios que utiliza el componente. En este caso, como se trata de pruebas unitarias, los servicios deben ser mockeados.

Después de configurar el componente, se utiliza la función `createComponent` para crearlo, se le asigna un deployment ficticio con uuid 1 y nombre 'Test', y finalmente se llama a `detectChanges`, que inicia un ciclo de detección de cambios para el componente y lo inicializa.

Seguidamente, se define el código perteneciente al test que comprueba que un secreto se elimina correctamente si el usuario confirma la acción. En primer lugar, `spyOn` crea una función mock que monitoriza el diálogo de confirmación donde el usuario declara si quiere eliminar la clave o no. En este caso, como se quiere emular que el usuario valida esta operación, se determina que esta devuelve el valor true. También se monitoriza la función `deleteSecret` perteneciente al servicio `SecretsService`, para posteriormente comprobar que se ha llamado, y el snackbar que muestra un mensaje informativo del estado de la operación. Después de esto, se busca el botón de borrado, a través de su id, y se pulsa sobre él. Finalmente, se comprueba que: el diálogo de confirmación y la función de borrado se han llamado una vez, el snackbar se ha mostrado con el mensaje y el color correctos, y el listado de secretos está vacío, ya que se ha eliminado el único que había.

El resto de pruebas unitarias se encuentran en el repositorio del proyecto y están estructuradas siguiendo la organización de Angular: los componentes y servicios tienen su propia carpeta en la cual se encuentra el archivo con extensión `.spec.ts` con las pruebas realizadas. En cuanto al backend, las pruebas sobre los nuevos endpoints fueron desarrolladas por el encargado de la API de la aplicación.

6.2. Pruebas end-to-end

Se realizaron pruebas end-to-end en las cuales se comprueban los distintos flujos de trabajo de la aplicación y se verifica que la interacción y el intercambio de datos entre sus componentes es correcto. Estas pruebas verifican el funcionamiento de la aplicación completa (integración del frontend con el backend), centrándose en las dos nuevas funcionalidades de la plataforma. Para ello se utilizó Cypress [40], una herramienta que permite escribir, configurar y depurar todo tipo de pruebas sobre aplicaciones web. En total, se comprobaron dos secciones de la plataforma:

- Dashboard.
- Deployments: centrándose en la parte referente a los secretos de los servidores de aprendizaje federados.

Para utilizar esta herramienta, fueron necesarias una serie de tareas de configuración, las cuales se distribuyen en distintos archivos:

- `cypress.config.ts` almacena toda la configuracion específica de Cypress (Código 12). En este caso se define el tamaño del navegador donde se van a ejecutar los tests (`viewportWidth` y `viewportHeight`), y se añaden dos configuraciones específicas de los tests end-to-end: la variable `experimentalStudio` habilita una nueva funcionalidad experimental de Cypress que genera código a partir de las interacciones del usuario, y la variable `testIsolation` evita que se borre la sesión (cookies y almacenamiento local) entre los tests. Esto

permite que, una vez que el 'usuario' test se ha identificado en la web, todos los tests puedan ejecutarse sin tener que volver a iniciar sesión. Por último, la línea 2 indica que se tienen que utilizar las variables de entorno definidas en `env`, que incluyen el usuario y contraseña de la cuenta de test. Se profundizará sobre la ubicación y el contenido de estas variables en el Apartado 6.4.

```
1  import { defineConfig } from 'cypress';
2  require('dotenv').config();  // Populate process.env with values from .env file
3
4  export default defineConfig({
5      viewportWidth: 1380,
6      viewportHeight: 800,
7      e2e: {
8          experimentalStudio: true,
9          testIsolation: false,
10     },
11 });
```

Código 12: Contenido de `cypress.config.ts`

- `global.d.ts` recoge los comandos personalizados creados por el desarrollador (Código 13). Estos actuan como 'métodos' que engloban cierta funcionalidad que se repite o que es de utilidad en varios tests. En este caso se definió un nuevo comando, `login`, que engloba todos los pasos necesarios para iniciar sesión en la web.

```
1  declare namespace Cypress {
2      interface Chainable {
3          login(username: string, password: string): Chainable<any>;
4      }
5  }
```

Código 13: Contenido de `global.d.ts`

- `commands.ts` contiene la implementación de los comandos definidos en el archivo `global.d.ts`. Por motivos de brevedad, el código del comando `login` se omite en este apartado, encontrándose disponible en el Apéndice E.

Por último, el Código 14 muestra un ejemplo de una de las pruebas realizadas sobre la sección del Dashboard. En este caso, la semántica de Cypress es bastante sencilla, por lo que es fácil entender el comportamiento de cada función. La primera de ellas, `before`, que se ejecuta una única vez antes de los tests, inicia sesión en la página y accede a la sección del Dashboard. A continuación, se muestra un fragmento del test `show graphs tab`, en el cual se comprueba que la pestaña de gráficas renderiza todos los elementos correctamente. Se accede a la pestaña correspondiente y se verifica que los títulos son correctos y que las gráficas de la parte superior se actualizan cuando el usuario cambia de tab. También se comprueba que las tarjetas de estadísticas situadas debajo de las gráficas se muestran correctamente. Cabe destacar que se han establecido tiempos de espera (`timeouts`) elevados porque la ejecución en las Github Actions es más lenta que en el navegador. Al igual que en las pruebas unitarias, el resto de tests se pueden encontrar en el repositorio, concretamente bajo la carpeta `/cypress/e2e`.

```
describe('dashboard section', function () {
    before(function () {
        cy.login(Cypress.env('EMAIL'), Cypress.env('PASSWORD'));
        cy.contains('Dashboard', { timeout: 10000 }).click();
    });

    it('shows graphs tab', function () {
        cy.contains('Graphs', { timeout: 10000 }).click();
        cy.contains('AI4EOSC Usage Over Time').should('be.visible');

        cy.contains('Running deployments').should('be.visible');
        cy.get('canvas').should('be.visible');

        cy.contains('Queued deployments').should('be.visible');
        cy.contains('Queued deployments').click();
        cy.get('canvas').should('be.visible');

        cy.contains('CPUs').should('be.visible');
        cy.contains('CPUs').click();
        cy.get('canvas').should('be.visible');

        (...)

        cy.get('#title-aggregate').contains('Aggregate Resource Usage');
        cy.get('app-stat-card').find('.header p:contains("CPUs")').should('have.length', 1);
        cy.get('app-stat-card').find('.header p:contains("Memory")').should('have.length', 1);
        cy.get('app-stat-card').find('.header p:contains("Disk")').should('have.length', 1);
        cy.get('app-stat-card').find('.header p:contains("GPUs")').should('have.length', 1);
    });
});
```

Código 14: Fragmento de `dashboard.component.cy.ts`

6.3. Pruebas de aceptación

Finalmente, las pruebas de aceptación siguen la misma dinámica que las pruebas alfa-beta [41]. Las pruebas alfa se realizan por el equipo de desarrollo en un entorno controlado (dev), antes de que el software sea actualizado para el resto de usuarios (prod). Su objetivo es evaluar el software para detectar y corregir errores antes de lanzar nuevas funcionalidades. En este caso, este tipo de pruebas son realizadas por mis directores Álvaro y Andrés.

Por otro lado, las pruebas beta, las realizan los usuarios finales en el entorno de producción. Su objetivo es obtener feedback sobre la aplicación, identificar posibles problemas y errores que no fueron detectados en la prueba alfa, y hacer mejoras en el software para la siguiente versión. Estos comentarios llegan al equipo de desarrollo en las reuniones de los Work Packages, y en los eventos y workshops que se realizan a lo largo del año con los casos de uso.

6.4. Integración continua y despliegue

Las operaciones de CI/CD (Continuous Integration and Continuous Delivery) se realizan utilizando la herramienta GitHub Actions [42], la cual permite automatizar flujos de trabajo personalizados directamente desde el propio repositorio de GitHub. Estos flujos de trabajo pueden incluir tareas como la ejecución de pruebas de código y el despliegue de aplicaciones. Se escriben utilizando sintaxis YAML y se almacenan en la sección `.github/workflows` del repositorio. En este proyecto existen varios ficheros de este tipo:

- CI: establece el flujo de integración continua.
- Build: determina qué acciones realizar para la creación del ejecutable. Hay varios ficheros de build, uno por cada entorno: desarrollo (`dev`) y producción (`prod`).

- Release-Please: automatiza la generación del `CHANGELOG`, la creación de releases y la actualización de la versión del proyecto.

La figura del Apéndice F muestra, de manera esquemática, el flujo de trabajo que se sigue durante la integración y entrega continua. Dentro de este flujo, tuve que actualizar y adaptar los workflows correspondientes a la integración continua y el build. A continuación se detalla el comportamiento de cada uno de ellos.

En primer lugar, el Código 15 muestra el flujo de trabajo de CI. Al comienzo se coloca el nombre del flujo y a continuación se especifica con la etiqueta `on` cuándo se debe ejecutar el workflow, que en este caso es en todas las pull request que se realicen sobre la rama `main`. Seguidamente se listan los 'jobs' o tareas que se han de realizar, en este caso se especifica que para ello se utilice una máquina con la última versión de Ubuntu junto con un entorno de ejecución Node.js en su versión 18 con su patch más reciente.

```yaml
name: CI AI4EOSC

on:
    pull_request:
        branches: ['main']

jobs:
    build:
        runs-on: ubuntu-latest
        strategy:
            matrix:
                node-version: [18.x]
        steps:
            - name: Checkout
              uses: actions/checkout@v3
              with:
                  fetch-depth: 0
            - name: Use Node.js ${{ matrix.node-version }}
              uses: actions/setup-node@v3
              with:
                  node-version: ${{ matrix.node-version }}
                  registry-url: 'https://npm.pkg.github.com'
                  cache: 'npm'
              env:
                  NODE_AUTH_TOKEN: ${{ secrets.AI4_DASHBOARD_TOKEN }}
            - run: npm ci
            - run: npm run build-ai4eosc-prod
            - run: npm run lint
            - name: Cypress
              uses: cypress-io/github-action@v6
              with:
                  start: npm start
                  wait-on: 'http://localhost:8080'
                  wait-on-timeout: 300
              env:
                  CYPRESS_EMAIL: ${{secrets.CYPRESS_EMAIL}}
                  CYPRESS_PASSWORD: ${{secrets.CYPRESS_PASSWORD}}
            - run: npm run test
            - name: SonarQube
              uses: sonarsource/sonarqube-scan-action@master
              env:
                  SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}
                  SONAR_HOST_URL: ${{ secrets.SONAR_HOST_URL }}
```

Código 15: Contenido de `ci-ai4eosc-dashboard.yml`

La etiqueta `steps` enumera los pasos que se han de realizar durante el proceso, en este caso se realizan las siguientes tareas:

1. `Checkout` clona el repositorio completo.
2. `Use Node.js` configura Node.js utilizando la versión especificada (18.x). Además, se configura el registro de `npm` para usar el token de autenticación necesario para poder descargar ciertas dependencias.

3. `npm ci` instala las dependencias de los paquetes del proyecto. Una de las mejoras que se aplicó fue sustituir el comando `npm install` por este, ya que `npm ci` está pensado para ser utilizado en entornos automatizados como plataformas de pruebas, integración continua y despliegue, como es el caso. Por otro lado, `npm install` instala todas las dependencias e intenta actualizarlas. En el proceso genera un fichero `package-lock.json` en el que fija exactamente la versión de cada paquete, lo cual puede generar conflictos causados por paquetes que se hayan actualizado y puedan romper el código. Además como necesita calcular y comprobar las versiones de todas las dependencias, es mucho más lento. Como `npm ci` utiliza el fichero `package-lock.json` existente para instalar las dependencias exactas, es mucho más ligero y evita que se produzcan conflictos en el código.

4. `npm run build-ai4eosc-prod` script definido en el archivo `package.json` del repositorio, que se corresponde con el comando `ng build --configuration production-ai4eosc`, el cual se encarga de compilar la aplicación utilizando la configuración especificada en el archivo `environment.production.ts`.

5. `npm run lint` herramienta de análisis estático del código que comprueba la aplicación y revisa que no haya inconsistencias que se puedan convertir en bugs en un futuro. Además, comprueba que el código sea consistente en estilo y que siga las directrices de buenas prácticas relativas al lenguaje utilizado, en este caso TypeScript.

6. `Cypress` acción predefinida que permite configurar y ejecutar las pruebas Cypress del proyecto. Para ello se utilizan las variables de entorno (mencionadas en el fichero `cypress.config.ts` del apartado Pruebas end-to-end), encargadas de hacer el inicio de sesión del usuario de testing. Estos valores, no se pueden almacenar como texto plano, ya que estarían expuestos a todo el mundo. Es por eso, que no se almacenaron en los ficheros de configuración de Cypress, sino que sus valores se obtienen de los Action Secrets de GitHub.

7. `npm run test` script definido en el archivo `package.json` del repositorio, que se corresponde con el comando `jest --verbose --coverage`. Se encarga de ejecutar las pruebas de Jest detallando el estado de cada test de manera individual (`--verbose`) e indicando su información de cobertura, es decir, qué porcentaje de líneas de código están siendo probadas (`--coverage`).

8. `SonarQube` acción predefinida que permite ejecutar la herramienta SonarQube [43], la cual analiza el código de la aplicación y ofrece métricas sobre la seguridad, mantenibilidad y fiabilidad del código, entre otros. Las claves de acceso para lanzar el análisis también se guardan como GitHub Secrets.

Estos cuatro últimos puntos, junto con el cambio del comando `npm ci`, forman parte de las actualizaciones que se aplicaron sobre el proceso de integración continua para mejorarlo. Todas ellas se ejecutan sobre entornos de ejecución virtuales de GitHub a excepción de SonarQube, la cual se optó por alojar en un servidor propio del IFCA. Esto fue debido a que el resto de herramientas no ofrecían la posibilidad de utilizar *self-hosting*, por lo que cuestiones como la privacidad y el data ownership podrían verse vulneradas. Por tanto, para poder utilizar esta herramienta fue necesario un proceso previo de configuración e instalación del entorno, en el cual se creó una máquina virtual que ejecuta un contenedor con la imagen Docker de SonarQube

y que cuenta con su propia base de datos PostgreSQL. Una vez realizada esta configuración, para poder acceder a la interfaz web de Sonar hubo que utilizar un proxy, ya que el número de direcciones públicas disponibles en el IFCA (y en el resto del mundo en general) es limitado. Finalmente, teniendo la web accesible y tras iniciar sesión en ella, se puede observar un resumen con algunas de las métricas calculadas por Sonar (Figura 13).

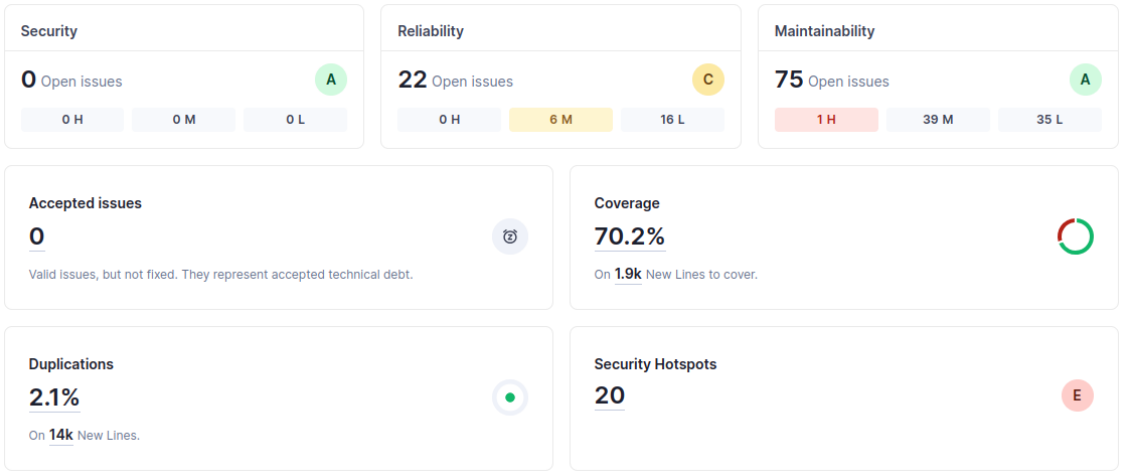

Figura 13: Análisis SonarQube

Este tablero sirve de ayuda para mejorar la calidad del código, y gracias a sus gráficas e histogramas, se puede ver el efecto que tienen las modificaciones del código sobre estas métricas. Por ejemplo, la Figura 14 muestra la comparativa de la métrica Coverage antes y después de incluir los tests unitarios y end-to-end. Por otro lado, la Figura 15 muestra la diferencia del porcentaje de código duplicado antes y después de realizar tareas de refactorización sobre el código.

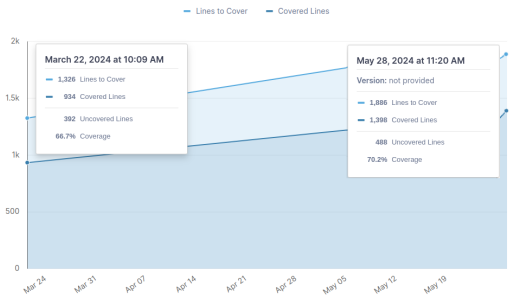

Figura 14: Análisis Coverage

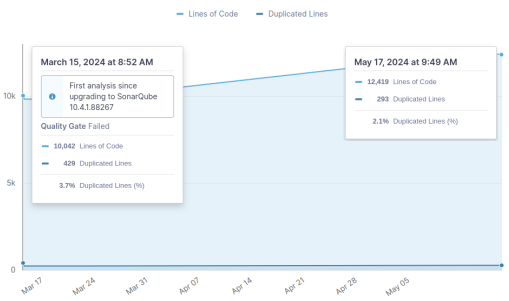

Figura 15: Análisis Duplicaciones

En cuanto al build, el Código 16 muestra, a modo de ejemplo, el flujo de ejecución en producción. Una de las mejoras aplicadas fue subir la imagen a Docker Hub. Anteriormente solo se subía a Harbor, pero se optó por redundarla almacenándola en dos sitios como medida de seguridad para evitar la pérdida de las imágenes. También se modificó el input del flujo para que el usuario pudiera seleccionar entre el entorno de producción y el de staging (un entorno intermedio entre dev y prod). Además, también se crearon webhooks [44] para automatizar el proceso de actualización de las imágenes. En el caso de `prod`, la ejecución de estas acciones se tiene que activar de manera manual. Sin embargo, en `dev` se ejecutan automáticamente siempre que se hace un push a la rama `main`.

```yaml
name: Build AI4EOSC:prod docker image
on:
  workflow_dispatch:
    inputs:
      select-env:
        type: choice
        description: 'Environment'
        required: true
        default: 'production'
        options:
            - production
            - staging

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout
        uses: actions/checkout@v3
        with:
          fetch-depth: 0
      - name: Set tag based on selected option
          run: |
              if [[ "${{ github.event.inputs.select-env }}" == "production" ]]; then
                 echo "TAG=${{ github.ref_name }}-prod" >> $GITHUB_ENV
              elif [[ "${{ github.event.inputs.select-env }}" == "staging" ]]; then
                 echo "TAG=${{ github.ref_name }}-stage" >> $GITHUB_ENV
              fi
      - name: Login to Harbor
        uses: docker/login-action@v2
        with:
          registry: ${{ secrets.HARBOR_URL  }}
          username: ${{ secrets.HARBOR_USERNAME  }}
          password: ${{ secrets.HARBOR_TOKEN }}
      - name: Login to Docker Hub
        uses: docker/login-action@v2
        with:
          username: ${{ secrets.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}
      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v2
      - name: "Create .npmrc"
        run: |
            echo "//npm.pkg.github.com/:_authToken=${{ secrets.AI4_DASHBOARD_TOKEN }}" > $HOME/.npmrc
      - name: Build and push
        uses: docker/build-push-action@v3
        with:
          context: .
          push: true
          tags: ${{ secrets.HARBOR_URL }}/ai4os/ai4eosc-dashboard:${{ env.TAG }},
                ${{ secrets.DOCKERHUB_USERNAME }}/ai4eosc-dashboard:${{ env.TAG }}
          file: docker/ai4eosc/prod/Dockerfile
          secret-files: |
            npmrc=/home/runner/.npmrc
      - name: Send Webhook to Deploy Dashboard
          env:
              SECRET: ${{ secrets.WEBHOOK_SECRET }}
          run: |
               HASH=$(echo -n "$PAYLOAD" | openssl dgst -sha1 -hmac "$SECRET" | sed 's/^.* //')
               curl -X POST https://webhooks.cloud.ai4eosc.eu/hooks/deploy-dashboard-prod \
               -H 'Content-Type: application/json' \
               -H "X-Hub-Signature: sha1=$HASH" \
               -d '{"ref": "refs/heads/master"}'
```

Código 16: Código de `build-ai4eosc-prod-dashboard.yml`

A continuación, se detallan los distintos pasos del proceso:

1. `Checkout` clona el repositorio completo.
2. `Set tag based on selected option` crea una variable con la tag de la imagen correspondiente, dependiendo de si se trata de `prod` o de `stage`.
3. `Login to Harbor` inicia sesión en el registro de Harbor utilizando las credenciales almacenadas en los secretos de GitHub.

4. `Login to Docker Hub` inicia sesión en Docker Hub utilizando las credenciales almacenadas en los secretos de GitHub.

5. `Set up Docker Buildx` configura Docker Buildx, una herramienta de construcción avanzada para imágenes de Docker.

6. `Create .npmrc` crea un archivo `.npmrc` en el directorio `home` del runner, configurando un token de autenticación para acceder al registro de paquetes npm de GitHub.

7. `Build and push` construye y sube la imagen de Docker.

8. `Send Webhook to Deploy Dashboard` realiza una llamada POST al endpoint configurado con los webhooks. Cuando se realiza una llamada al webhook, este ejecuta un script en el cual actualiza el archivo `docker-compose.yml` para que utilice la imagen más reciente disponible y la actualiza en el entorno de producción. Con el fin de añadir una capa de seguridad a esta llamada, y para evitar que personas externas realicen llamadas sobre este endpoint, se utiliza una clave encriptada utilizando el algoritmo SHA-1 [45].

7. Conclusiones

El desarrollo de este trabajo se ha centrado en el proceso de incorporación de nuevas funcionalidades en la plataforma web AI4EOSC y en su respectiva API. Concretamente, se ha añadido una nueva sección de estadísticas en la cual se muestra información sobre los recursos del clúster, se han implantado claves de acceso para poder gestionar los despliegues de aprendizaje federado, se han realizado tareas de diseño y pruebas, y se ha mejorado y ampliado el proceso de integración continua y despliegue. Estas implementaciones han quedado reflejadas en las estadísticas de los repositorios del frontend y el backend, en los cuales se pueden ver las contribuciones realizadas a la rama main durante el desarrollo del proyecto (Figura 16 y Figura 17).

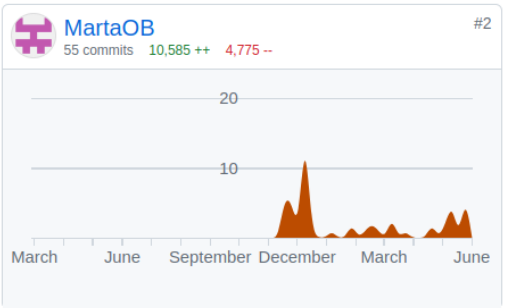

Figura 16: Contribuciones Frontend

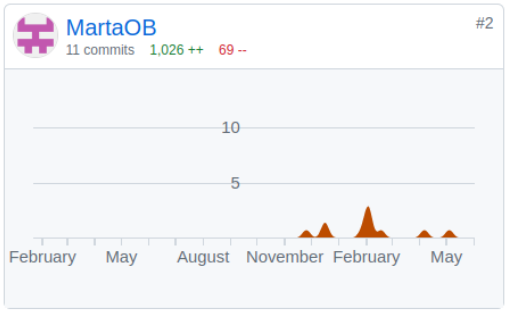

Figura 17: Contribuciones Backend

Esta memoria detalla mi experiencia al integrarme en un proyecto ya en marcha y describe las principales aportaciones realizadas sobre el mismo. Durante este proceso he ido asumiendo progresivamente la responsabilidad de su desarrollo, hasta convertirme en la encargada del frontend del proyecto. Este proceso no solo me ha permitido aplicar los conocimientos adquiridos durante mi formación, sino que también me ha dado a conocer una visión del desarrollo de software desde el punto de

vista de la investigación. He tenido la oportunidad de explorar tecnologías emergentes y enriquecer mi experiencia profesional, además de aportar nuevas herramientas a la comunidad científica, ya que se trata de código abierto.

Por otro lado, a parte de lo mencionado anteriormente, durante el desarrollo de este trabajo también he podido colaborar en mi primer paper [46], y he tenido la oportunidad de asistir a un workshop del proyecto iMagine [47]. En este evento pude conocer a los participantes de los distintos casos de uso del proyecto en persona, y obtener feedback sobre la plataforma. Estos comentarios fueron muy valiosos de cara a mejorar y guiar el rumbo de las nuevas funcionalidades de la web.

Finalmente, me gustaría destacar que el resultado de las implementaciones desarrolladas durante este trabajo de fin de máster, son utilizadas por una gran cantidad de investigadores, tanto del proyecto AI4EOSC como de iMagine, en su día a día. A continuación, las Figuras 18 y 19 muestran las estadísticas de uso de ambas plataformas.

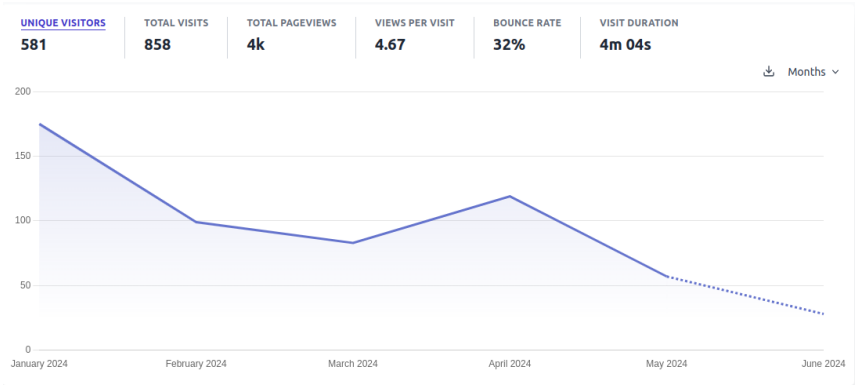

Figura 18: Estadísticas Usuarios AI4EOSC

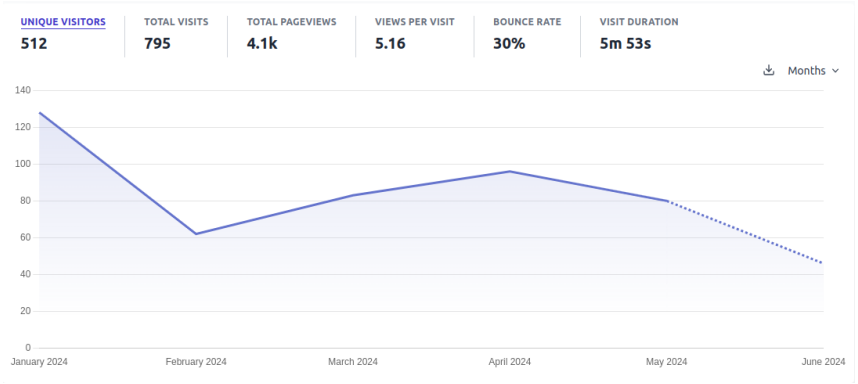

Figura 19: Estadísticas Usuarios iMagine

Con esto me gustaría resaltar que no se trata simplemente de una propuesta teórica o un ejercicio académico, sino de una herramienta práctica que ha sido adoptada en varias organizaciones. Esta aceptación y uso generalizado subrayan su relevancia y utilidad, confirmando que el trabajo realizado cumple con las necesidades y expectativas de los casos de uso.

8. Trabajos futuros

Durante el desarrollo de este trabajo de fin de máster, se han alcanzado varios objetivos importantes que han contribuido al desarrollo y mejora de la plataforma

AI4EOSC. Sin embargo, al tratarse de un proyecto en curso, ya que finaliza en el año 2025, existen algunas tareas y mejoras adicionales que no han podido ser realizadas dentro del alcance de este trabajo, pero que se irán añadiendo durante los próximos meses. A continuación, se detallan algunas de las posibles líneas de trabajo futuras que podrían abordar estas tareas pendientes y ampliar los resultados obtenidos. Por un lado estarían aquellas relacionadas con la metodología:

- Respecto a las herramientas utilizadas para realizar el seguimiento del proyecto, el uso de Wekan podría sustituirse por GitHub Issues [48]. Esta herramienta permite crear un panel compartido en el cual se pueden agregar vistas como un tablero Kanban o un roadmap. Esto facilitaría el seguimiento de las tareas, ya que estaría sincronizado con los repositorios del proyecto y se podría asociar cada *issue* a una de las distintas ramas de los repositorios. Asimismo, tiene opciones de personalización que permiten clasificar las tareas según su importancia y categoría, lo cual mejoraría el flujo de trabajo y la priorización de actividades.

- En relación con el punto anterior, se podrían mejorar las prácticas de desarrollo utilizando una nomenclatura estándar para nombrar las ramas según el *issue* que tengan asociado [49].

Por último, se listan aquellas relativas a nuevas funcionalidades:

- En un futuro es posible que haya más tipos de herramientas, a parte del servidor de aprendizaje federado, por tanto, será necesario adaptar y ampliar la gestión de secretos a las futuras herramientas de la plataforma.

- Otro enfoque que se podría incluir en la plataforma, sería el *experiment centric*, es decir, en vez de trabajar con despliegues de módulos directamente, se podrían crear experimentos que agruparan varios módulos y datasets relacionados.

- Tras la implantación de SonarQube, el siguiente paso sería reducir la deuda técnica a partir del análisis proporcionado por la herramienta. En este trabajo se realizaron tareas para aumentar la cobertura de las pruebas y reducir las duplicaciones de código, pero se podrían realizar más acciones para mejorar la calidad del código.

- Actualmente, los usuarios necesitan descargar sus propios dataset para poder entrenar los distintos módulos de la plataforma. Una mejora que se podría aplicar sería la integración con plataformas como Zenodo [50] o HuggingFace [51], las cuales podrían proveer a la plataforma de infinidad de datos, y además evitarían que el usuario tuviera que descargarlos él mismo, ya que se podría gestionar su vinculación con los distintos despliegues a través de la propia plataforma AI4EOSC.

- Se podrían realizar integraciones con otro tipo de herramientas como por ejemplo Gradio [52], un paquete de Python de código abierto que permite crear rápidamente una demostración o aplicación web para modelos de aprendizaje automático y APIs de Python. También sería interesante una integración con Swagger [53], para que los usuarios tuvieran acceso a la documentación de los servicios web RESTful de los módulos alojados en la plataforma. A través de ellos podrían consultar datos como por ejemplo los argumentos necesarios para realizar una predicción o los metadatos de la propia aplicación.

- Finalmente, se podrían mejorar los códigos de respuesta de los endpoints de la plataforma, de manera que fueran más descriptivos y concisos a la hora de describir el error. Por ejemplo: en vez de devolver un error 401 (Unauthorized) cuando un usuario no pertenece a una `vo`, o esta no existe, enviar un código 401 en el primer caso, y un código 404 (Not Found) en el segundo.

Apéndice A

ID	Historia de Usuario
US-01	Visualizar el estado del clúster
Yo, como usuario, quiero ver cuál es la cantidad total de recursos del clúster: cuántos de ellos están siendo utilizados y cuántos están disponibles. Concretamente quiero tener información sobre las CPUs, la memoria RAM, el disco y las GPUs. Esta información me permitiría conocer el estado de uso de la plataforma y saber qué recursos están disponibles para poder utilizarlos en mis próximos despliegues.	
US-02	Visualizar el uso actual de recursos del usuario
Yo, como usuario, quiero ver cuántos recursos estoy utilizando actualmente con mis despliegues. Concretamente quiero tener información sobre el número de jobs, CPUs, frecuencia de las CPUs, memoria RAM, disco y GPUs. Esta información me permitiría tener un control sobre mi uso de recursos sobre la plataforma.	
US-03	Conocer información básica de los datacenters que forman el clúster
Yo, como usuario, estoy interesado en conocer qué centros de datos conforman el clúster, junto con información básica acerca de ellos, específicamente: dónde están situados, su nombre, su PUE, la calidad de la energía que utilizan, el número de jobs que están ejecutando, y el número de recursos totales y utilizados del centro (CPUs, memoria RAM, disco y GPUs). Conocer estos datos me resultaría interesante, ya que deseo saber cuál es el impacto medioambiental y de consumo energérico de los centros, y me ofrecería una mayor confianza, ya que quiero saber dónde se están ejecutando mis despliegues.	
US-04	Visualizar un histórico del uso del clúster
Yo, como usuario, quiero conocer, de manera cronológica, cuáles han sido los niveles de utilización del clúster en los últimos 3 meses. Especialmente me interesaría conocer esta información sobre: los despliegues que están siendo ejecutados, los que se han encolado, las CPUs, la frecuencia de las CPUs, la memoria RAM, el disco y las GPUs. Esto me ayudaría a conocer cuál ha sido el estado del clúster en distintas fechas, y poder determinar ciertas tendencias o patrones que pudieran estar relacionadas con estas.	
US-05	Visualizar el uso agregado de los recursos utilizados por el clúster y el usuario
Yo, como usuario, quiero saber cuántos recursos he consumido en total respecto al agregado de lo que han utilizado todos los usuarios de la plataforma. Esta información me resultaría de interés para saber qué uso estoy haciendo yo comparado con lo que se ha utilizado en total en la plataforma.	
US-06	Visualizar información detallada de los recursos de cada nodo
Yo, como usuario, quiero conocer en detalle las características de los nodos del clúster. Concretamente, quiero saber qué recursos tiene cada uno de ellos (CPU, GPU, memoria RAM y disco) clasificándolos entre los que tienen GPU y los que no. Conocer esta información me permitiría saber cuáles son las características de los nodos que tengo disponibles a la hora de lanzar mis despliegues.	

Tabla 7: Historias de Usuario (Estadísticas)

ID	US	Requisito
RF-01	US-01, US-03	Ver el número de CPUs libres y en uso del clúster.
RF-02	US-01, US-03	Ver la cantidad de memoria libre y en uso del clúster.
RF-03	US-01, US-03	Ver la cantidad de disco libre y en uso del clúster.
RF-04	US-01, US-03	Ver el número de GPUs libres y en uso del clúster.
RF-05	US-01	Ver el número de GPUs libres y en uso del clúster por tipo.
RF-06	US-02	Ver el número de jobs en ejecución del usuario.
RF-07	US-02	Ver el número de CPUs en uso del usuario.
RF-08	US-02	Ver la frecuencia de las CPUs en uso del usuario.
RF-09	US-02	Ver la cantidad de memoria RAM en uso del usuario.
RF-10	US-02	Ver la cantidad de disco en uso del usuario.
RF-11	US-02	Ver el número de GPUs en uso del usuario.
RF-12	US-03	Ver la localización de los distintos centros de datos.
RF-13	US-03	Ver el número de jobs en ejecución de cada centro de datos.
RF-14	US-03	Ver el PUE de cada centro de datos.
RF-15	US-03	Ver la calidad de la energía utilizada en cada centro de datos.
RF-16	US-04	Ver de manera cronológica la cantidad de deployments ejecutados en el clúster los últimos 3 meses.
RF-17	US-04	Ver de manera cronológica la cantidad de deployments encolados en el clúster los últimos 3 meses.
RF-18	US-04	Ver de manera cronológica el número de CPUs utilizadas en el clúster los últimos 3 meses.
RF-19	US-04	Ver de manera cronológica la frecuencia de las CPUs utilizadas los últimos 3 meses.
RF-20	US-04	Ver de manera cronológica la cantidad de memoria RAM utilizada en el clúster los últimos 3 meses.
RF-21	US-04	Ver de manera cronológica la cantidad de disco utilizado en el clúster los últimos 3 meses.
RF-22	US-04	Ver de manera cronológica el número de GPUs utilizadas en el clúster los últimos 3 meses.
RF-23	US-05	Ver el uso agregado de CPUs del usuario frente al clúster.
RF-24	US-05	Ver el uso agregado de memoria RAM del usuario frente al clúster.
RF-25	US-05	Ver el uso agregado de disco del usuario frente al clúster.
RF-26	US-05	Ver el uso agregado de GPUs del usuario frente al clúster.
RF-27	US-06	Ver las características (CPUs, GPUs, memoria RAM y disco) de los nodos del clúster.

Tabla 8: Requisitos Funcionales (Estadísticas)

ID	Historia de Usuario
US-07	Listar las claves de acceso
Yo, como usuario, quiero ver cuáles son las claves de acceso asociadas a los servidores de aprendizaje federado que he creado. Deseo saber cuál es el identificador de cada secreto y cuál es su valor. De esta manera, puedo consultar y acceder fácilmente a las claves para para poder enviárselas a los distintos participantes.	
US-08	Crear una clave de acceso
Yo, como usuario, quiero poder crear nuevas claves de acceso a partir de un identificador asignado por mi. De esta manera, podré crear una clave diferente para cada participante. También podré crear secretos de prueba para experimentar y probar mi servidor.	
US-09	Eliminar una clave de acceso
Yo, como usuario, quiero poder eliminar las claves de acceso de mi servidor de aprendizaje federado, tanto las que se han generado por defecto, como las que haya creado posteriormente. Esto me permitiría revocar las claves que se hayan visto vulneradas y eliminar aquellas pertenecientes a los usuarios u organizaciones que ya no colaboran en el servidor.	
US-10	Visualizar una clave de acceso
Yo, como usuario, quiero conocer el valor secreto de una clave de acceso perteneciente a mi servidor de aprendizaje federado. Esto me permitirá repartir y gestionar las claves entre los participantes.	

Tabla 9: Historias de Usuario (Claves de Acceso)

ID	US	Requisito
RF-28	US-07	Ver el listado de claves de acceso de un servidor de aprendizaje federado creado por el usuario.
RF-29	US-08	Crear una nueva clave de acceso de un servidor de aprendizaje federado, creado por el usuario, a partir de un identificador.
RF-30	US-09	Eliminar una clave de acceso de un servidor de aprendizaje federado creado por el usuario.
RF-31	US-07, US-10	Ver el valor secreto de una clave de acceso perteneciente a un servidor de aprendizaje federado creado por el usuario.

Tabla 10: Requisitos Funcionales (Claves de Acceso)

ID	Descripción
RN-01	La interfaz web debe ser adaptable a ordenadores y dispositivos móviles
La interfaz debe ser capaz de ajustarse dinámicamente según el dispositivo y el tamaño de pantalla en el que se visualiza. Esto se recoge en la característica de flexibilidad, que es la capacidad del producto para adaptarse a cambios en sus requisitos, contextos de uso o entorno del sistema. Concretamente pertenece a la subcaracterística de adaptabilidad, que consiste en que el producto pueda ser adaptado de forma efectiva y eficiente a diferentes entornos determinados de hardware, software, operacionales o de uso.	
RN-02	Las pantallas deben tener un tiempo de carga medio inferior a 3 segundos
Las llamadas a la API deben ser lo más ágiles posibles, de manera que la experiencia del usuario al interactuar con la web sea agradable. Esto se corresponde con la característica de comportamiento temporal, dentro de la categoría de eficiencia del desempeño, la cual representa el comportamiento del producto en la realización de sus funciones dentro de unos parámetros de tiempo y rendimiento específicos.	
RN-03	La aplicación debe tener fácil navegabilidad
Con el objetivo de facilitar al usuario la utilización y navegabilidad por la aplicación web, se debe seguir un esquema de colores fijo y un posicionamiento de elementos consistente. Se pretende que el usuario pueda utilizar la plataforma con facilidad, incluyendo así las subcaracterísticas de aprendizabilidad y operabilidad de la clase capacidad de interacción.	
RN-04	La aplicación debe proteger al usuario de posibles errores
Siempre que el usuario tenga que enviar información o tomar acciones desde la aplicación, la interfaz de usuario del sistema deberá ser capaz de detectar la mayor cantidad de errores posibles en los datos de entrada y alertar adecuadamente al usuario de ello, informándole además de cómo solucionar los problemas detectados. Esto contribuye a la subcaracterística de protección contra errores de usuario, la cual evalúa la capacidad del sistema para prevenir errores en su operación.	

Tabla 11: Tabla de Requisitos No Funcionales

Apéndice B

En este apartado se detallan el resto de modelos C4 de la aplicación. El modelo que representa la organización de los Deployments (Figura 20) es bastante sencillo, ya que consiste en una lista de deployments organizada en dos tablas (módulos y herramientas), donde cada fila tiene un botón que permite mostrar el detalle de cada deployment y otro que permite manejar los secretos (en el caso de las herramientas).

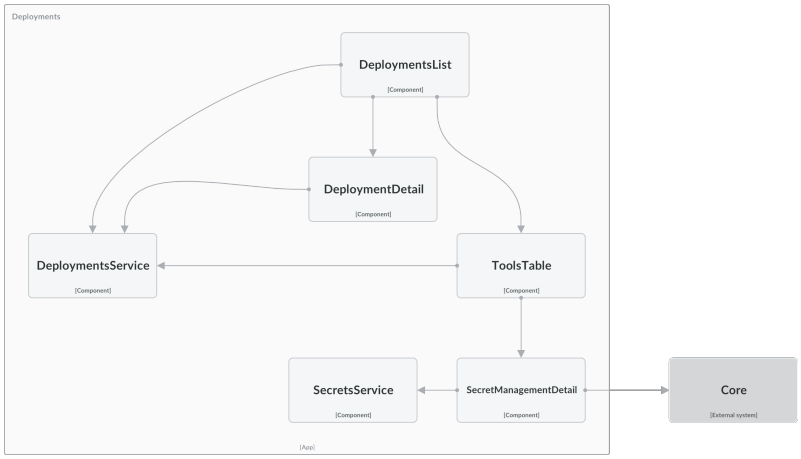

Figura 20: Modelo C4 Deployments (Nivel 4)

Por otra parte, el modelo del Core (Figura 21) tiene menos componentes, ya que solo contiene los servicios de autorización para que los usuarios puedan iniciar sesión en la plataforma y utilizar sus distintas funcionalidades.

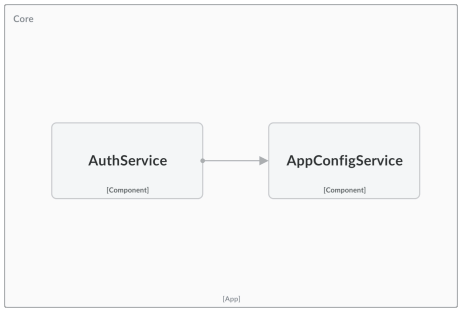

Figura 21: Modelo C4 Core (Nivel 4)

Finalmente, tal y como se mencionó en la Sección 4.1, el diagrama inicial del Marketplace, que representa la organización del catálogo de módulos y herramientas junto con sus formularios de creación, presentaba cierta inconsistencia (Figura 22). Concretamente, la organización de los formularios (componentes *Form), era distinta dependiendo de si se trataba de la creación de un módulo o de una herramienta. Para mejorar la reusabilidad y favorecer la modularidad del código, se utilizó como base para todos los formularios el componente StepperForm, el cual sirve de plantilla para crear un formulario personalizable con 3 pasos (Figura 23). En el caso de los módulos estos tres pasos son: GeneralConfForm, HardwareConfForm y StorageConfForm. Por tanto, el componente ModuleTrain que se encarga de mostrar el formulario de creación de los módulos, tendrá un componente StepperForm al cual le pasará como parámetros las plantillas que necesita. De esta manera, los formularios para la creación de herramientas (de momento solo existe una, FederatedServer), podrán actuar de la misma forma, cambiando únicamente qué plantillas envían.

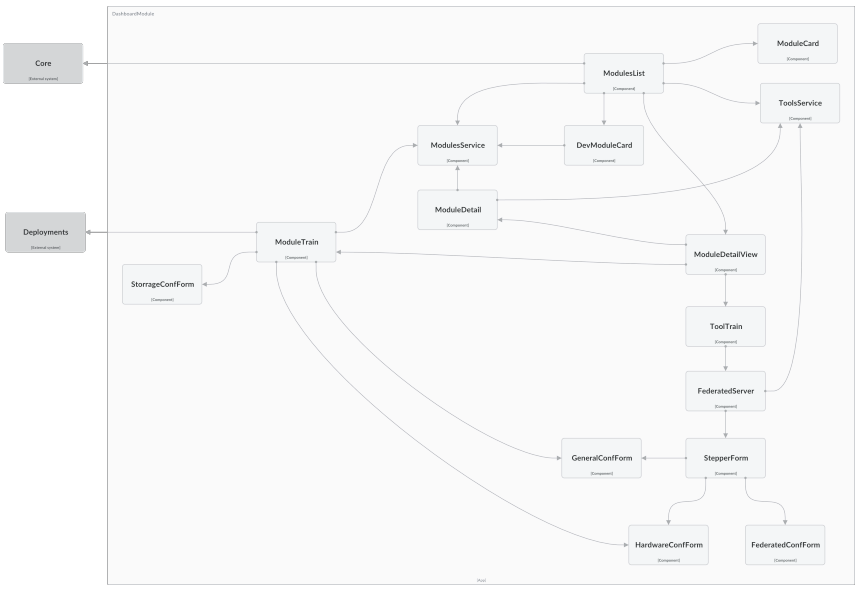

Figura 22: Modelo C4 Marketplace Versión Inicial (Nivel 4)

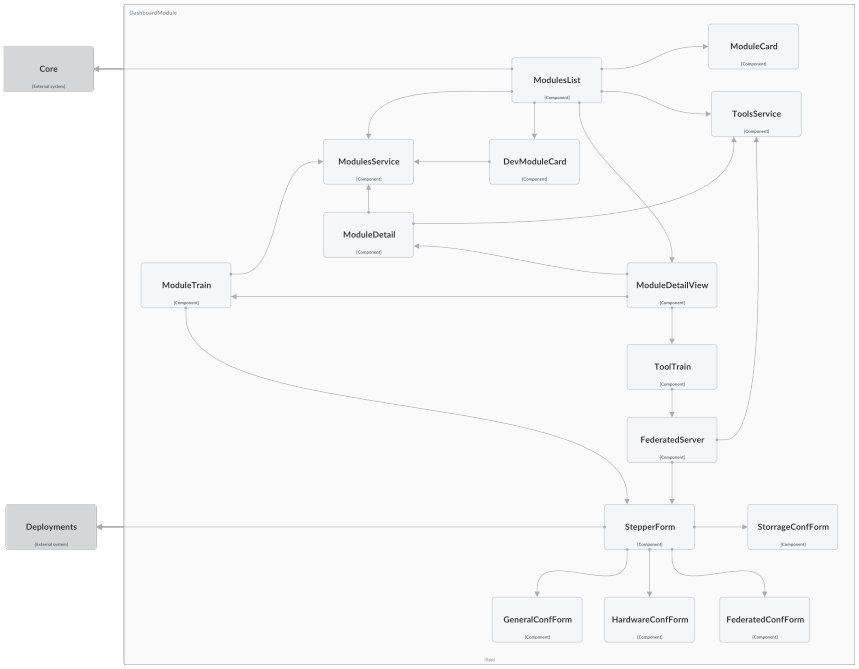

Figura 23: Modelo C4 Marketplace Versión Actualizada (Nivel 4)

Apéndice C


```
{
  "datacenters": {
    "ai-ifca": {
      "lat": 43.471629,
      "lon": -3.802222,
      "PUE": 1.2,
      "energy_quality": 3,
      "nodes": {
        "15843b50-187c-ee3f-374d-d22f93ae2a78": {
          "cpu_total": 86,
          "cpu_used": 45,
          "gpu_total": 8,
          "gpu_used": 8,
          "ram_total": 354054.359375,
          "ram_used": 104000,
          "disk_total": 201487.828125,
          "disk_used": 62201.88671875,
          "name": "ahost21-gpu5-ifca",
          "jobs_num": 11,
          "gpu_models": {
            "Tesla T4": {
              "gpu_total": 8,
              "gpu_used": 8
            }
          }
        },
        "16c8be83-db39-48dc-7bd9-fd495d3ac2e6": {
          "cpu_total": 4,
          "cpu_used": 3,
          "gpu_total": 0,
          "gpu_used": 0,
          "ram_total": 7263.41015625,
          "ram_used": 6000,
          "disk_total": 29587.0625,
          "disk_used": 6899.0859375,
          "name": "ahost6-ifca",
          "jobs_num": 3,
          "gpu_models": {}
        },
      }
    }
  },
  "cluster": {
    "cpu_total": 90,
    "cpu_used": 48,
    "gpu_total": 8,
    "gpu_used": 8,
    "ram_total": 361317.76953125,
    "ram_used": 110000,
    "disk_total": 231074.890625,
    "disk_used": 69100.97265625,
    "gpu_models": {
      "Tesla T4": {
        "gpu_total": 8,
        "gpu_used": 8
      }
    }
  }
}
```


Código 17: Representación de ClusterStats

Apéndice D

A continuación se muestran las ilustraciones del resto de interfaces gráficas.

Figura 24: Cuadro de Ayuda Emergente

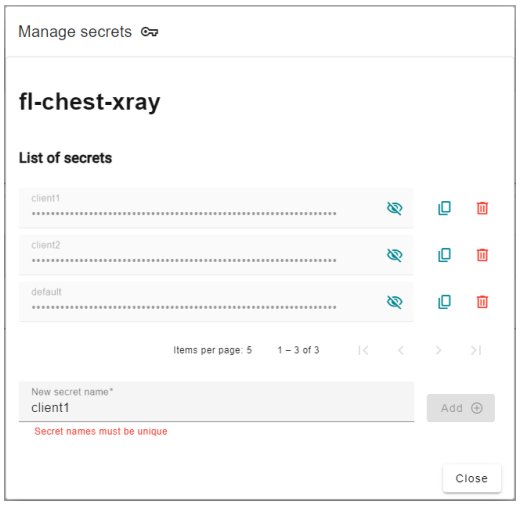

Figura 25: Añadir una clave secreta con un identificador repetido

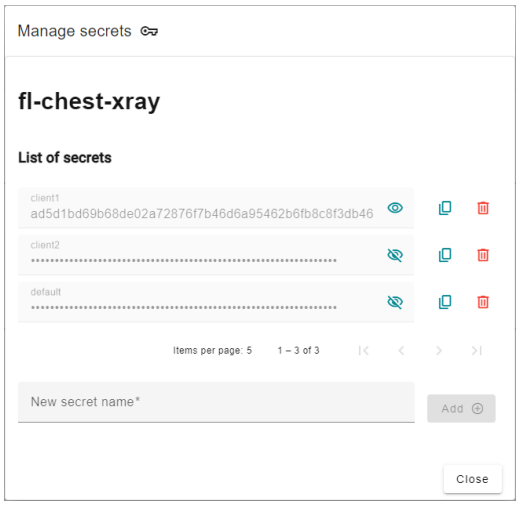

Figura 26: Ocultar/mostrar una clave secreta

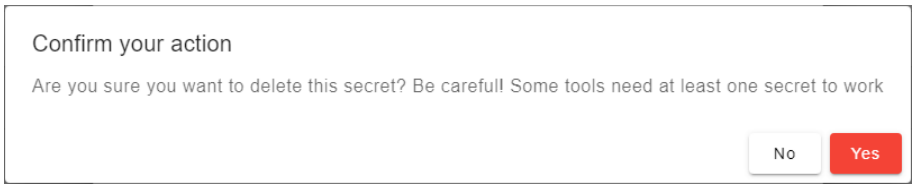

Figura 27: Eliminar la última clave secreta

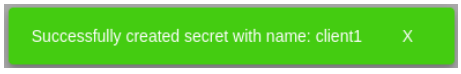

Figura 28: Notificación emergente al crear una clave secreta

Figura 29: Notificación emergente de error al eliminar una clave secreta

Apéndice E

En este apartado se detalla el funcionamiento del código del comando `login` que se utiliza para automatizar el proceso de autenticación del usuario test en la aplicación (Código 18). En primer lugar se añade el comando personalizado, indicando que acepta dos argumentos: `username` y `password`, que representan las credenciales del usuario. Después, se crea un registro (log) para poder depurar su flujo de esta operación. Finalmente, se utiliza `cy.session` para gestionar la sesión del usuario. Esto permite reutilizar la sesión autenticada en múltiples pruebas, evitando la necesidad de iniciar sesión repetidamente. Este proceso se incluye en los siguientes pasos:

1. Se navega a la página principal de la aplicación y se pulsa sobre el botón 'Login'.

2. Dentro de la página de inicio de sesión de EGI, se hace click sobre el botón para iniciar sesión utilizando Bitbucket.

3. En la página de Bitbucket, se introducen las credenciales del usuario y se hace click en los botón de inicio de sesión.

4. Se verifica que la URL actual es correcta después del proceso de autenticación. También se realiza una validación adicional para asegurar que la página contiene el texto 'Test AI4EOSC', es decir, el usuario ha iniciado sesión correctamente. La variable `cacheAcrossSpecs` permite que la sesión se mantenga entre diferentes archivos de prueba.

```
Cypress.Commands.add('login', (username: string, password: string) => {
    const log = Cypress.log({
        displayName: 'AUTHO LOGIN',
        message: [`Authenticating | ${username}`],
        autoEnd: false,
    });
    log.snapshot('before');
    const args = { username, password };
    cy.session(
        `auth-${username}`,
        () => {
            cy.visit('http://localhost:8080/');
            cy.contains('Login').click();
            cy.origin('https://aai.egi.eu', () => {
                cy.get('.ssp-btn.bitbucket').click();
            });
            cy.origin(
                'https://id.atlassian.com/l',
                { args },
                ({ username, password }) => {
                    cy.get('#username').type(username, { log: false });
                    cy.get('#login-submit').click();
                    cy.get('#password').type(password, { log: false });
                    cy.get('#login-submit').click();
                }
            );
            cy.url().should('equal', 'http://localhost:8080/');
        },
        {
            validate() {
                cy.contains(`Test AI4EOSC`);
            },
            cacheAcrossSpecs: true,
        }
    );
    log.snapshot('after');
});
```

Código 18: Contenido de `commands.ts`

Apéndice F

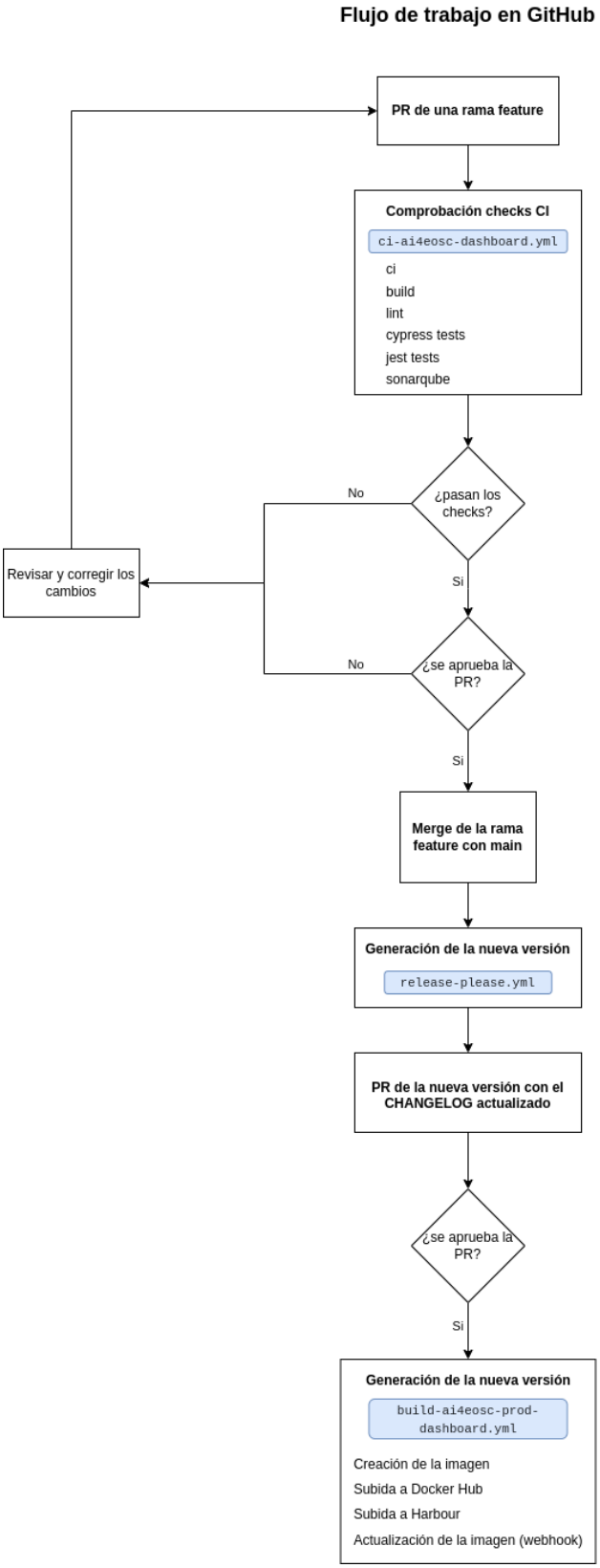

Figura 30: Diagrama de Despliegue

Referencias

[1] A. project, "AI4EOSC," 2024. [Online]. Available: https://ai4eosc.eu/

[2] I. y. U. Ministerio de Ciencia, "CSIC," 2024. [Online]. Available: https://www.csic.es/es

[3] iMagine, "iMagine Proyect," 2024. [Online]. Available: https://www.imagine-ai.eu/

[4] EOSC, "EOSC," 2024. [Online]. Available: https://eosc.eu/

[5] DEEP-HYBRID, "DEEP-HYBRID," 2024. [Online]. Available: https://deep-hybrid-datacloud.eu/

[6] AI4EOSC, "AI4EOSC Dashboard," 2024. [Online]. Available: https://dashboard.cloud.ai4eosc.eu/marketplace

[7] HashiCorp, "Nomad," 2024. [Online]. Available: https://www.nomadproject.io/

[8] ——, "HashiCorp," 2024. [Online]. Available: https://www.hashicorp.com/

[9] Google, "Kubernetes," 2024. [Online]. Available: https://kubernetes.io/es/

[10] HashiCorp, "Vault," 2024. [Online]. Available: https://www.vaultproject.io/

[11] Google, "Angular," 2024. [Online]. Available: https://angular.io/

[12] Facebook, "React," 2024. [Online]. Available: https://es.react.dev/

[13] CKENDRICK, "Fallacies of React: Why so many React projects fail," 2023. [Online]. Available: https://blog.smartclient.com/fallacies-of-react-why-so-many-react-projects-fail/

[14] P. S. Foundation, "Python," 2024. [Online]. Available: https://www.python.org/

[15] J. Cross, "Python Nomad," 2024. [Online]. Available: https://python-nomad.readthedocs.io/en/latest/

[16] UPV, "Python Oscar," 2024. [Online]. Available: https://pypi.org/project/oscar-python/

[17] AI4EOSC, "AI4EOSC Docs," 2024. [Online]. Available: https://docs.ai4eosc.eu/en/latest/user/index.html

[18] K. Tool, "Kanban," 2024. [Online]. Available: https://kanbantool.com/es/metodologia-kanban

[19] Wikipedia, "Scrum," 2024. [Online]. Available: https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software)

[20] Atlassian, "Confluence," 2024. [Online]. Available: https://www.atlassian.com/es/software/confluence

[21] Wekan, "Wekan," 2024. [Online]. Available: https://wekan.github.io/

[22] EGI, "EGI: Advanced Computing for a Data-Driven Future," 2024. [Online]. Available: https://www.egi.eu/

[23] ISO, "ISO 25010," 2024. [Online]. Available: https://iso25000.com/index.php/normas-iso-25000/iso-25010

[24] A. C. BEHNAM POURGHASSEMI, ARDALAN AMIRI SANI, "What-If Analysis of Page Load Time in Web Browsers Using Causal Profiling," 2019. [Online]. Available: https://sci-hub.se/https://doi.org/10.1145/3341617.3326142

[25] S. Brown, "Modelo C4," 2024. [Online]. Available: https://c4model.com/

[26] AI4EOSC, "AI4EOSC - Dashboard," 2024. [Online]. Available: https://github.com/ai4os/ai4-dashboard

[27] ——, "AI4EOSC - Platform API," 2024. [Online]. Available: https://github.com/ai4os/ai4-papi

[28] Angular, "Observable," 2024. [Online]. Available: https://docs.angular.lat/guide/observables-in-angular

[29] Google, "Angular Material," 2024. [Online]. Available: https://material.angular.io/

[30] O. Layers, "Open Layers," 2024. [Online]. Available: https://openlayers.org/

[31] Echarts, "Echarts," 2024. [Online]. Available: https://echarts.apache.org/en/index.html

[32] sindresorhus, "Crypto Random String," 2024. [Online]. Available: https://www.npmjs.com/package/crypto-random-string

[33] Angular, "RxJS," 2024. [Online]. Available: https://angular.io/guide/rx-library

[34] FastAPI, "FastAPI," 2024. [Online]. Available: https://fastapi.tiangolo.com/

[35] F. Utilities, "FastAPI Utilities," 2024. [Online]. Available: https://fastapi-utils.davidmontague.xyz/

[36] Wikipedia, "Boilerplate," 2024. [Online]. Available: https://en.wikipedia.org/wiki/Boilerplate_code

[37] FastAPI, "Async Context Manager," 2024. [Online]. Available: https://fastapi.tiangolo.com/advanced/events/?h=

[38] F. Utilities, "Repeat Every," 2024. [Online]. Available: https://fastapi-utils.davidmontague.xyz/user-guide/repeated-tasks/#the-repeat_every-decorator

[39] Facebook, "Jest," 2024. [Online]. Available: https://jestjs.io/es-ES/

[40] Cypress, "Cypress," 2024. [Online]. Available: https://www.cypress.io/

[41] T. Hamilton, "Pruebas Alfa Beta," 2024. [Online]. Available: https://www.guru99.com/es/alpha-beta-testing-demystified.html

[42] GitHub, "GitHub Actions," 2024. [Online]. Available: https://github.com/features/actions

[43] Sonar, "SonarQube," 2024. [Online]. Available: https://www.sonarsource.com/products/sonarqube/

[44] A. Hajdarević, "Webhooks," 2024. [Online]. Available: https://github.com/adnanh/webhook

[45] Wikipedia, "SHA-1," 2024. [Online]. Available: https://en.wikipedia.org/wiki/SHA-1

[46] J. Sáinz-Pardo Díaz, A. Heredia Canales, I. Heredia Cachá, V. Tran, G. Nguyen, K. Alibabaei, M. Obregón Ruiz, S. Rebolledo Ruiz, and A. López García, "Making federated learning accessible to scientists: The ai4eosc approach," in *Proceedings of the 2024 ACM Workshop on Information Hiding and Multimedia Security*, ser. IHMMSec '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 253–264. [Online]. Available: https://doi.org/10.1145/3658664.3659642

[47] EGI, "iMagine Competence Centre Workshop," 2024. [Online]. Available: https://indico.egi.eu/event/6371/

[48] GitHub, "GitHub Issues," 2024. [Online]. Available: https://github.com/features/issues

[49] T. University, "Best Practices for Naming Git Branches," 2024. [Online]. Available: https://tilburgsciencehub.com/topics/automation/version-control/advanced-git/naming-git-branches/

[50] CERN, "Zenodo," 2024. [Online]. Available: https://zenodo.org/

[51] I. Hugging Face, "Hugging Face," 2024. [Online]. Available: https://huggingface.co/

[52] Gradio, "Gradio," 2024. [Online]. Available: https://www.gradio.app/

[53] S. Software, "Swagger," 2024. [Online]. Available: https://swagger.io/

Lista de Figuras

1. Versión inicial del Dashboard de AI4EOSC (Sección Marketplace) . . 7
2. Versión inicial del Dashboard de AI4EOSC (Sección Deployments) . . 7
3. Modelo C4 (Nivel 3) 11
4. Modelo C4 Dashboard (Nivel 4) 12
5. Sección Overview 14
6. Modal de GPUs 15
7. Sección Datacenters 15
8. Sección de Gráficas 16
9. Sección de Nodos 16
10. Lista de Despliegues 17
11. Modal para gestionar las claves secretas 17
12. Eliminar una clave secreta 18
13. Análisis SonarQube 31
14. Análisis Coverage 31
15. Análisis Duplicaciones 31
16. Contribuciones Frontend 33
17. Contribuciones Backend 33
18. Estadísticas Usuarios AI4EOSC 34
19. Estadísticas Usuarios iMagine 34
20. Modelo C4 Deployments (Nivel 4) 41
21. Modelo C4 Core (Nivel 4) 41
22. Modelo C4 Marketplace Versión Inicial (Nivel 4) 42
23. Modelo C4 Marketplace Versión Actualizada (Nivel 4) 42
24. Cuadro de Ayuda Emergente 44
25. Añadir una clave secreta con un identificador repetido 44
26. Ocultar/mostrar una clave secreta 44
27. Eliminar la última clave secreta 44
28. Notificación emergente al crear una clave secreta 44
29. Notificación emergente de error al eliminar una clave secreta 44
30. Diagrama de Despliegue 46