UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

Plataforma para la interoperabilidad de datos clínicos mediante Federated Learning

Trabajo de Fin de Máster

Autor:
Francisco Moreno García
Tutores:
Dra. Silvia Uribe Mayoral
Dr. Rafael Pastor Vargas
Dr. Antonio Robles Gómez

Máster en Ingeniería Informática

Septiembre de 2022

Agradecimientos

En primer Lugar quiero agradecer mis tutores Antonio, Rafael y Silvia por su ayuda, dedicación y contribución a que este TFM saliera adelante. En especial a Silvia que pasó por más problemas burocráticos para poder firmar el convenio que si tuviera que acceder a datos clínicos, agradecerla también todo el apoyo que he recibido de ella desde que entré en el grupo de investigación y el tiempo dedicado en este trabajo en particular.

Quiero agradecer igualmente a mi consejo de sabios sobre inteligencia artificial, Alberto y Borja, que me han ayudado en la parte del proyecto en la que estaba más verde. A los compañeros con los que comparto más tiempo, Fede, Javi S., Iago, Andrés y Rivers y también a los que aunque estén más lejos siempre puedo contar con ellos, Chus,José Manuel, Miguel, Jorge, Juanan, Álvaro, Juan, Vero, David, Javi G, Giusseppe y Anaida. Espero no olvidarme de ninguno.

A mis padres y hermanos por ayudarme a ser la persona que soy.

Y por último y más importante a Claudia , por el amor y la paciencia que me dedica.

A Carmela, Nacho y Nicolás, literal y figurativamente,
la razón por la que me levanto todos los días.

Resumen

En la mitología griega, Tántalo era un hijo de Zeus. Tántalo había sido acogido en el Olimpo e invitado a la mesa de los dioses. Luego robó la ambrosía, la comida de los inmortales, la llevo a los humanos y reveló los secretos de los dioses. Zeus se enojó y castigó a Tántalo a una tortura eterna. Su castigo consistió en permanecer en un lago cubierto de agua hasta la barbilla bajo un árbol con toda clase de manjares. Pero cuando tenía hambre o sed el agua y los manjares se alejaban de su alcance.

Podríamos hacer un símil con el mito de Tántalo y la situación actual de los datos clínicos. Existen extensas bases de datos con información clínica. Pero cuando necesitamos acceder a esta información para realizar una investigación, nos encontramos con trabas legales, de privacidad, burocráticas o tecnológicas y se alejan de nuestro alcance y podemos sentirnos como Tántalo cuando no era capaz de saciar sus necesidades.

En este trabajo se va a intentar facilitar el proceso del uso de estos datos para el entrenamiento de modelos de aprendizaje automático, evitando a los científicos de datos tener que lidiar con las dificultades que conlleva, mediante una plataforma con interfaz gráfica, que haciendo uso de fereated learning, convierta el proceso en algo transparente a sus ojos.

Palabras Clave

Federated Learning, Privacidad, Machine Learning, Interfaz Gráfica, Flower.

Índice general

Índice de figuras

Índice de tablas

Acrónimos

AI Inteligencia Artificial (Artificial Inteligence). 1, 13–15, 21

API Interfaz de Progrmación de Aplicaciones. 12, 15, 25, 26, 28, 41, 42, I, II, IV

CRUD Create, read, update and delete. 24, 27

CSS Cascading Style Sheet. 28

DP Privacidad Diferencial (Differncial Privaccy. 60

ELK Elasticsearch Kibana Logstash. 40, 41

EMR Electronic medical records. 9

FATE (Federated AI Technology Enabler). 11, 14, 15, 20, 21, IV

FHIR Fast Healthcare Interoperability Resources. 10

FL Aprendizaje federado (Federated Learning). 5, 6, 10–12, 14, 15, 20, 21, 23, 56, 59, 60

GDPR General Data Protection Regulation. 3, 4

gRPC Google Remote Procedure Calls. 21

HCS Healthcare Privacy and Security Classification System. 9, 10

HL7 Health Level 7. 9, 10

HTML Hyper Media Textual Language. 28

INRIA The National Institute for Research in Digital Science and Technology. 20

IoT Internet de las cosas (Internet of things). 10

JS Java Script. 28

LR Regresión Lineal (Linear Regresion). 49–51, 53, 55, 56, V

MAE Error absoluto medio (Mean absolute error). 49

ML Aprendizaje Automático (Machine LEarning). 1, 12, 14, 23, 56

MQTT MQ Telemetry Transport. 20

MRI Imágenes por resonancia magnética. 11

NN Red Neuronal (Neural Network). 48–50, 52–56, V

OWID Our World in data. 47, 48

ReLu Rectified Linear Unit. 48

REST representational state transfer. 23, 24

RPC Remote Procedure Calls. 15

Introducción

En la última década la Inteligencia Artificial (AI) y el aprendizaje automático (ML) han revolucionado la manera en la que afrontamos la innovación tecnológica, permitiendo el desarrollo de algoritmos muy diversos que pueden ir desde aquellos destinados a conseguir que los coches conduzcan solos hasta otros que ayuden a la detección temprana de enfermedades, entre otros. Este incremento en el uso de la AI ha venido de la mano de un crecimiento exponencial de datos, debido, entre otras razones, a la implantación del Internet de las Cosas (IoT), sin los cuales no sería posible dicho desarrollo. En este sentido, la existencia actual de grandes volúmenes de datos en casi cualquier ámbito es innegable, pero el problema con lo que se encuentran muchos investigadores es cuando no se puede acceder a ellos de manera sencilla, imposibilitando por lo tanto su uso y limitando con ello su impacto. Además, el inesperado y rápido impacto de la pandemia producido por el Covid-19 ha dejado al descubierto otras carencias ya existentes previamente en el ámbito de la salud como son la falta de interoperabilidad entre datos, soluciones y organizaciones [1] y la necesidad de respuestas colaborativas que permitan reducir el tiempo para probar con eficacia nuevas herramientas, aplicaciones, tecnologías y bases de conocimiento tanto en momentos de crisis como en situaciones normales.

En muchas ocasiones, los intentos de colaboración se abandonan incluso antes de empezar debido a diversas razones, ya sean burocráticas, éticas o legales, [2]haciendo muy difícil llegar a soluciones de alto impacto en cualquier ámbito, pero especialmente en el de la salud, donde los posibles beneficios de este tipo de soluciones son especialmente significativos. Por esta razón, el avance en soluciones que permitan el uso de datos masivos y la colaboración entre entidades debe situarse como prioridad, definiendo distintas estrategias para ello. En relación a esto, y como se comentará a continuación, este trabajo nace con la intención de aportar una solución al problema del acceso a datos masivos para el desarrollo de algoritmos avanzados principalmente en el ámbito sanitario pero no sólo circunscrito a él, permitiendo con ello obtener el máximo rendimiento derivado de su uso mediante el aseguramiento de un acercamiento respetuoso con cualquier limitación

existente relativa al uso de datos sensibles.

Privacidad de los datos médicos: ¿necesidad o limitación?

A día de hoy, la información es considerada como un bien en sí mismo puesto que permite la obtención de conocimiento pero, de acuerdo a la jerarquía o pirámide del conocimiento [3], ésta sólo puede definirse en términos de datos. Desde un punto de vista meramente formal, el término "dato"se puede entender como toda aquella recopilación en bruto o representación simbólica de un hecho o conocimiento, pero sin su agrupación, procesado y posterior interpretación es imposible derivar cualquier tipo de información. De esta realidad nace la necesidad de desplazar el foco, convirtiendo el dato en verdadero protagonista y pilar básico sobre el que descansa cualquier tipo de sabiduría que pueda desarrollarse a posteriori, y definiendo dos aspectos fundamentales sin los cuales es imposible la construcción de las capas superiores de la pirámide: la obtención y el acceso y uso de los datos.

En relación a la obtención, y tal y como se ha señalado anteriormente, la aparición e implantación de la IoT durante los últimos años ha facilitado no sólo la generación sino también la recopilación de datos de muy diversa índole, garantizándose con ello un flujo continuo con una alta previsión de crecimiento (algunos estudios indican una previsión de 79.4 zettabytes de volumen generado en 2025 frente a los 13.6 zettabytes que fueron generados en 2019, [4]).

El segundo aspecto, esto es, el del acceso a los datos generados, conlleva una problemática mayor, en especial cuando nos referimos a información personal y sensible derivada del análisis de datos similares de especial importancia. Dicho esto, y con el fin de garantizar una correcta comprensión de la situación en el ámbito que nos ocupa, vamos a dedicar una líneas a intentar distinguir entre diversos términos que en general son usados como sinónimos pero que no lo son: datos personales, sensibles y referidos a salud.

El primer nivel de clasificación se concentra en lo que se conocen como datos personales, esto es, toda aquella información relevante concerniente a una persona que la hace identificable, independientemente del medio usado para recogerla, guardarla, tratarla, etc.

Dentro de los datos personales podemos encontrar aquellos que por su importancia son considerados sensibles o especialmente protegidos, y que hacen refe-

rencia a cualidades de la persona relacionadas con su dignidad, personalidad o comportamiento y que, por lo tanto, precisan unas garantías especiales de recogida, tratamiento y uso.En relación a esto, y aunque no es la única clasificación existente en el estado del arte, en [5] se propone una división en tres bloques para los cuales identifica, a su vez, los procedimientos a aplicar en cada caso:

- Datos relativos a ideología, afiliación sindical, religión y creencias. Su recogida solo será posible mediante un consentimiento expreso y por escrito del afectado.
- Datos relativos al origen racial, la salud y la vida sexual, que solo pueden ser recogidos, tratados y cedidos bajo tres supuestos: una finalidad de interés general, una disposición de una ley o un consentimiento de forma expresa por parte del usuario.
- Datos relativos a las infracciones penales o administrativas, que solo pueden ser incluidos en ficheros que posean las Administraciones Públicas en determinados supuestos.

Finalmente, dentro de los datos sensibles, nos encontramos con los datos de salud, que contienen información relativa a la salud presente, pasada y futura, así como datos genéticos y biométricos, de personas sanas o enfermas [6]. Estos datos forman parte de su esfera más íntima puesto que incluyen información sobre su cuerpo, sexualidad, raza, código genético, hábitos de vida, de alimentación y de consumo e, incluso, antecedentes familiares.

De acuerdo a estas definiciones, la protección de los datos sensibles, y en concreto de los datos de salud, es especialmente necesaria y se realiza mediante el desarrollo de un marco legal adecuado. En este sentido, el 25 de mayo de 2018 entró en vigor el Reglamento General de Protección de Datos (GDPR) [7] o reglamento europeo relativo a la protección de las personas físicas en lo que respecta al tratamiento de sus datos personales y a la libre circulación de estos datos, lo que supuso una revolución en cuanto a la protección y tratamiento de los datos, devolviendo el control a los individuos sobre sus datos. Este nuevo paradigma garantiza, por tanto, la seguridad y confidencialidad de los datos, aspectos esenciales para los usuarios, pero, a la vez puede limitar su obtención, tratamiento y uso, ralentizando con ello la definición de soluciones basadas en el análisis masivo de datos. En este sentido, y como ejemplo especialmente relevante en el ámbito sanitario, este reglamento convierte a los encargados del tratamiento de datos en responsables de garantizar un nivel de seguridad adecuado al riesgo y la aplicación de técnicas que

aseguren la privacidad (como la seudonimización y el cifrado, por ejemplo). Esto hace que los hospitales u otros organismos poseedores de datos sean especialmente reticentes a permitir su acceso o compartición.

Este recelo a la hora de compartir datos, ya sea por precaución o por imposibilidad, se traduce en muchas ocasiones en una falta de disponibilidad de grandes volúmenes de datos de alta calidad para el entrenamiento de modelos de aprendizaje automático. Además de barreras legales, éticas y económicas, existen otras limitaciones como la falta de confianza o la complejidad técnicas, como por ejemplo la falta de soluciones de puesta en común ampliamente adoptadas [8] que impiden la puesta en común de datos y con ello en avance de este tipo de soluciones. Además, y como problema añadido, estas barreras se agravan en el contexto de la investigación de enfermedades raras o nuevas, como el Covid-19, donde los conjuntos de datos suelen ser pequeños y dispersos y su puesta en común es más complicada puesto que los datos contenidos suelen ser más fácilmente identificables, y a menudo requieren la integración de múltiples modalidades de datos (genómicos, fenotípicos, demográficos, historia clínica, imágenes radiológicas y otros) para obtener resultados relevantes. En resumen, aun cuando los datos médicos existen, tienden a distribuirse mediante silos incomunicados, imposibilitando o haciendo muy complicado su uso en entornos de big data, por lo que el desarrollo de soluciones que permitan salvar esta problemática se hace especialmente necesario.

Federated Learning como solución para el uso de datos masivos sensibles

Una de las principales características del aprendizaje automático es la necesidad de disponer de un elevado volumen de datos y de una gran potencia computacional para obtener los modelos deseados. En general, esto suele conseguirse mediante nodos únicos de procesado con altísimas capacidades de computación y almacenamiento donde se realizan los entrenamientos necesarios. Pero, en algunos ámbitos, y en especial en el entorno de la salud, esta solución no suele ser la más óptima, tal y como se comentó en el apartado anterior. De acuerdo a la anteriormente citada GDPR, varios principios deben regir la gestión de datos, tales como los siguientes:

- Minimización de datos: la recopilación y uso de los datos debe ser el mínimo necesario para el objetivo buscado. Estos datos serán distintos para cada estudio, pero deben mantenerse al tratarse de datos pertenecientes al historial médico de los pacientes.

- Limitación de almacenamiento: los datos solo deberían mantenerse el tiempo estrictamente necesario para el proceso para el que se requieran, lo que entra en conflicto con la necesidad de mantener y mejorar los datos para hacer lo mismo con el modelo generado.
- Integridad y confidencialidad: debe garantizarse la seguridad de los datos en estos dos aspectos, pero la forma en la que se haga queda a criterio del proveedor de datos.

Teniendo en mente estos aspectos, a día de hoy no siempre es posible contar con servidores suficientemente potentes o garantizar la seguridad de los datos frente a todas las posibles amenazas existentes mediante las soluciones más comúnmente utilizadas, esto es, soluciones centralizadas, por lo que se hace necesario avanzar en nuevas propuestas que permitan el uso de datos bajo las premisas indicadas.

Recientemente, el paradigma del Aprendizaje Federado (FL) ha ganado popularidad como medio para superar estas limitaciones, proporcionando un enfoque escalable y que preserva la privacidad para el entrenamiento conjunto de modelos de aprendizaje automático a través de repositorios de datos federados [9]. La idea central de FL es que, en lugar de compartir los datos, los distintos centros sólo tienen que compartir los parámetros del modelo de aprendizaje automático que se está entrenando, suprimiendo con ello las principales desventajas asociadas a las aproximaciones centralizadas.

Desde el punto de vista teórico, existen dos aproximaciones posibles a las estructuras federadas que pueden ser tenidas en cuenta a la hora de definir un flujo de trabajo concreto:

- Aprendizaje federado horizontal: es el que se lleva a cabo cuando la estructura de datos en todos los nodos es similar, permitiendo con ello que los nodos secundarios descarguen del nodo central el modelo actualizado para así poder entrenar en local y devuelvan los cambios de los coeficientes que se hayan obtenido al servidor central que combinará todos los recibidos, sin que medie ningún intercambio de datos a lo largo de todo el proceso.
- Aprendizaje federado vertical: en esta caso, cada nodo de la red federada no cuenta con toda la información necesaria para desarrollar el modelo, sino con un subconjunto de las características del mismo, por lo que es necesario combinar todos para disponer de la información necesaria para ejecutar el modelo. Se trata de una tipología muy característica de los entornos de

salud, donde es típico que unos nodos alberguen un tipo de datos mientras otros almacenan otros y, para poder realizar una predicción, es necesario que los datos relativos al mismo paciente se encuentren identificados de alguna manera en las distintas bases de datos. Es difícil de llevar a cabo sobre todo cuando los conjuntos de datos no están gestionados por la misma compañía (lo que se traduciría en identificadores incoherentes o en mayor posibilidad de identificación de los individuos entre los distintos conjuntos de datos), pero en la literatura se han presentado distintos métodos que intentan hacer frente a esta situación [10]

En cualquier caso, y a pesar de ofrecer un acercamiento plausible y ofrecer múltiples beneficios, el FL aún no puede considerarse como una tecnología madura ya que se enfrenta a algunos retos a los que se debe dar respuesta [11] como son el alto coste de las comunicaciones dentro de las redes federadas, la disparidad en cuanto a las capacidades de los distintos notos y a la distribución de los datos a lo largo de estos y, finalmente, aspectos avanzados relativos a la privacidad de los datos. Además, junto a su reciente aparición y a la complejidad de su puesta en marcha hace que, tal y como se señalará en el Capítulo 2, las iniciativas internacionales basadas en este enfoque estén empezando a desarrollarse actualmente, por lo que podría decirse que a día de hoy se trata de una solución que se encuentra en pleno proceso de implementación, desarrollo y despliegue.

Objetivos del trabajo

De acuerdo a la situación presentada, el objetivo de este trabajo consiste en crear una plataforma que permita utilizar FL de manera sencilla, con el fin de que los científicos de datos puedan centrarse en trabajar con los datos clínicos y los algoritmos de aprendizaje automático sin necesidad de preocuparse de la privacidad, de la arquitectura y del despliegue de los servidores y, de esta forma, colaborar en su adaptación sabiendo las ventajas que esto conlleva en el entorno clínico. Además, se ha identificado una serie de objetivos unitarios o parciales que debe cumplir la plataforma para garantizar la consecución del objetivo general:

- La plataforma deberá proporcionarse en forma de software de fácil instalación y configuración.
- La plataforma deberá presentar una interfaz intuitiva y accesible.
- El sistema deberá permitir el entrenamiento de algoritmos federados, con la posibilidad de usar datos de terceros, permitiendo salvar las limitaciones de

privacidad gracias al entrenamiento en local de estos algoritmos sin necesidad de acceder a los datos de forma centralizada y permaneciendo estos en los servidores de los propios centros clínicos.

- La plataforma deberá permitir la gestión de todos los recursos de la red FL, estos son: nodos,bases de datos, usuarios,grupos, entrenamientos, etc.
- Permitirá consultar los resultados de los entrenamientos
- Deberá permitir el almacenamiento de datos sanitarios, fuentes de datos públicas y privadas, datos clínicos y otros datos del mundo real, garantizando la privacidad de los datos y que estos no pueden salir de las instalaciones de salud donde fueron recogidos.
- Permitirá la visualización de los datos, presentados estos de manera que puedan ser comprendidos por el ser humano con herramientas de visualización semiautomáticas, y solo a las personas autorizadas.

Para cumplir estos objetivos el trabajo se ha estructurado según las siguientes fases básicas:

1. Diseño del sistema
   a) Análisis del problema
   Estudio detallado del problema, estado del arte y las diferentes tecnologías disponibles para desarrollar la solución
   b) Definición de requisitos
   Estudio y documentación de los requisitos, características y funcionalidades del sistema.
   c) Definición de la arquitectura
   Definición de las tecnologías a usar y la estructura de la plataforma.
2. Implementación de la plataforma
   a) Desarrollo del nodo central
   Desarrollo de una herramienta central que permita la gestión de usuarios, del flujo de los modelos a ejecutar mediante FL, de las colas de ejecución de los entrenamientos y la actualización de los modelos.
   b) Desarrollo nodos locales
   Desarrollo de una herramienta que pueda ser contenerizada y desplegada

de forma independiente en los servidores de terceros y que les permita la ingesta, almacenamiento, y consulta y visualización de datos y que les permita entrenar sus modelos de inteligencia artificial utilizando FL.

3. Validación mediante un caso de uso

   a) Definición del caso de uso
   Creación de un modelo de inteligencia artificial para probar el sistema basado en un modelo de datos de una fuente pública.

   b) Despliegue del modelo
   Validación del sistema desarrollado usando el modelo seleccionado en un entorno real compuesto por tres nodos desplegados.

Organización de la memoria

La memoria se ha organizado en 5 Capítulos:

1. Introducción, donde se explica brevemente el problema a resolver, las motivaciones y objetivos.
2. **Estado del arte**. En este capítulo se explica desde dónde se parte, cuál es el estado actual de la tecnología, se realiza una comparativa entre diferentes soluciones ya existentes y se explica cuál se ha decido utilizar como base de la presente plataforma y por qué.
3. **Diseño**, donde se desglosan los requisitos funcionales y no funcionales de la plataforma, la arquitectura global y de los diferentes nodos, la tecnología utilizada y los desarrollo propios de la plataforma.
4. **Validación**, o capítulo encargado de explicar la metodología usada para validar la plataforma, los resultados obtenidos y la comparativa entre los diferentes experimentos.
5. **Conclusiones**. Por último, en este capitulo se exponen las principales conclusiones del proyecto, se fijan las lineas de trabajo futuro y se incluye la planificación temporal y los costes del proyecto.

Estado del Arte

¿Cómo se comparten los datos clínicos ahora?

Cuando se trata de datos clínicos, su uso se pude dividir en dos tipos: primario y secundario. El uso primario de los datos es el que se da durante el propio tratamiento del paciente, cuando el personal sanitario necesita acceder a las historias clínicas electrónicas o EMR (Electronic medical records) para evaluar el estado del paciente, acceder a resultados de análisis clínicos de laboratorio, o actualizar el propio EMR. El uso secundario de los datos es el que se da en la investigación, ya sea con el objetivo de mejorar los tratamientos futuros o para la toma de decisiones clínicas. Estos datos se suelen añadir a lagos de datos que recopilan información de diferentes fuentes de datos médicas. Para proteger estos lagos de datos, el grupo de trabajo HL7 (Health Level 7), ha creado el standar HCS (Healthcare Privacy and Security Classification System) [12]una arquitectura para el etiquetado y la segmentación automatizada de la privacidad y la seguridad de la información sanitaria. HCS se centra en las etiquetas de seguridad, que se utilizan para etiquetar recursos. Las políticas de acceso de seguridad se referirán a dichas etiquetas para determinar el acceso a los recursos etiquetados. Las etiquetas se pueden dividir en cuatro grupos:

1. La confidencialidad, que se utiliza para indicar el grado de clasificación de un hecho clínico. Los valores van desde "no restringido.ª "muy restringido".
2. La categoría, que indica por que ley esta protegido un hecho clínico.
3. La integridad, que se utiliza para expresar la fiabilidad de los datos insertados.
4. El control, que indica las advertencias de manejo cuando los documentos clínicos se intercambian entre dos o más sistemas.

HCS integra un mecanismo de control de acceso en su diseño para determinar el acceso a los datos sanitarios del lago de datos, mediante un proceso de asignación de etiquetas previo. Además, para acceder a estos datos debe realizarse una solicitud. El sistema compara las políticas de privacidad y las reglas del recurso con los atributos de la solicitud para decidir si se concede la autorización para acceder a el.[12]. EL estándar HL7 FHIR [13] es la continuación del HL7 v2,esta basado en HCS y es está siendo ampliamente adoptado por diferentes organizaciones sanitarias para lograr la interoperabilidad [14] y el na nueva versión que viena substituir a HL7 v2 [15],que copaba el 95 % del mercado en los Estados Unidos [16]. Aunque existen estos estándares normalmente son habilitados para el uso primario y secundario dentro de la misma organización, pero estas mismas organizaciones son bastante reacias a compartir estos a organismos externos, lo que dificulta el desarrollo de algoritmos avanzados basados en el tratamiento masivo de datos compartidos.

¿Qué es Federated Learning?

Como se comentó en el capítulo anterior, FL es un nuevo enfoque que se utiliza para entrenar modelos de aprendizaje automático descentralizados a través de múltiples dispositivos satélites o nodos secundarios, desde teléfonos inteligentes a wearables médicos, pasando por vehículos y en general cualquier dispositivo IoT. El objetivo, por tanto, es que distintos nodos entrenen de forma colaborativa un modelo de aprendizaje automático compartido mientras mantienen los datos de entrenamiento localmente sin intercambiarlos entre ellos y sin necesidad de que residan en una ubicación central. En términos sencillos, con el aprendizaje federado no son los datos los que se mueven hacia el modelo, sino que es el modelo el que se mueve hacia los datos, evitando con ello posibles problemas derivados de la compartición de datos sensibles y dando respuesta a los recelos que se presentan en algunos casos.

De acuerdo a su propia definición, esta nueva aproximación supone una alternativa al enfoque centralizado tradicional para construir modelos de ML en el que los datos de diferentes fuentes se recogen y almacenan en un servidor, y luego el modelo se entrena también en un único servidor. Además, debido a las ventajas que su aplicación puede aportar, a día de hoy se está empezando a desarrollar en todos los campos de investigación pero en especial en el ámbito de la salud donde FL puede cobrar más importancia debido a la particularidad y sensibilidad de sus datos. En este sentido, el campo del análisis de imágenes médicas puede considerarse como uno de los más maduros donde se ha hecho patente el poder del FL durante los últimos años. De hecho, las revisiones recientes han puesto de manifiesto la gran

cantidad y diversidad de proyectos de FL que se están llevando a cabo en los últimos años en el ámbito sanitario [17, 18, 19]. Como ejemplo tenemos el desarrollo de proyectos a gran escala con notables aplicaciones en el análisis de la segmentación de todo el cerebro en imágenes por resonancia magnética (MRI) [20], la segmentación de tumores cerebrales [21, 22], las radiografías de tórax [23] y las imágenes del fondo de ojo de la retina [24]]. También se están desarrollando proyectos comerciales y gubernamentales a nivel nacional e internacional con el objetivo de construir una infraestructura de aprendizaje federada y multicéntrica que vincule los repositorios de datos sanitarios. Algunos ejemplos notables son la Plataforma Conjunta de Imágenes desarrollada por el Consorcio Alemán del Cáncer [25] y la American College of Radiology AI Lab [26]. En Francia, el consorcio Healthchain incluye varios hospitales nacionales [27], mientras que el framework de aprendizaje federado Fed-BioMed [28] ha sido adoptado por el consorcio UniCancer. A nivel europeo, se ha creado la iniciativa Gaia-X para construir un espacio de datos sanitarios descentralizado [29]. En Estados Unidos, las iniciativas conjuntas de los sectores público y privado también están estableciendo las primeras pruebas a gran escala para una infraestructura sanitaria FL, encabezadas por empresas como Intel [30] y Nvidia [31].

Plataformas existentes de FL

Dado que el objetivo del presente trabajo no es el desarrollo en sí de un nuevo marco, sino la obtención de una plataforma completa que permita la generación y ejecución de soluciones basadas en FL de forma fácil y rápida, el primer paso a realizar consistió en un análisis de las soluciones ya existentes sobre las que poder hacerlo. En este sentido, es importante tener en cuenta que el aprendizaje federado es un tema que evoluciona rápidamente y muchos proyectos de frameworks de FL se han iniciado en los últimos 5-6 años [32]. De acuerdo a esto, nueve han sido los principales frameworks que se han tenido en cuenta a la hora de analizar sus capacidades y comprobar su idoneidad con respecto a lo buscado:

- FATE (Federated AI Technology Enabler): es un framework de código abierto de FL permite colaborar con los datos protegiendo su seguridad y privacidad. Implementa protocolos de computación seguros basados en la encriptación homomórfica y la computación multiparte . Admite varios escenarios de FL y ofrece un conjunto de algoritmos, como la regresión logística, los algoritmos basados en árboles, el aprendizaje profundo y el aprendizaje por transferencia.[33]

- PaddleFL: es un framework de aprendizaje federado de código abierto basado en PaddlePaddle que permite a los investigadores replicar y comparar diferentes algoritmos de FL. Permite, además, desplegar un sistema de aprendizaje federado en clústeres distribuidos a gran escala [34].
- Clara training Framework proporciona un framework de entrenamiento que acelera el entrenamiento y la inferencia del ML para casos de uso de imágenes médicas. Permite a los investigadores y desarrolladores de imágenes médicas implementar modelos utilizando una API de alto nivel [**Clara2022**].
- IBM Federated Learning: solución propietaria de IBM para FL, es un framework en Python para un entorno empresarial.
- Flower Framework: framework para FL de código abierto y que permite utilizar la mayoria de frameworkws de ML [35].
- FedLearner: es un framework de aprendizaje automático colaborativo que permite el modelado conjunto de datos distribuidos entre instituciones.
- FedBiomed: es un proyecto de código abierto centrado en potenciar la investigación biomédica utilizando FL para el análisis estadístico. El proyecto se basa en Python y Pytorch [36].
- TensorFlow Federated: Solucion Nvidia de open source de ML con datos descentralizados para desarrollar fácilmente la investigación y experimentación con (FL), ML en el que varios nodos participantes conservan sus datos de entrenamiento de forma local entrenando con un modelo compartido [37].
- PySyft/PyGrid:librería de código abierto basada en Python 3 para FL, Fue desarrollada por la comunidad OpenMined y funciona principalmente con los frameworks PyTorch y TensorFlow [38] [39].

Una vez realizado el análisis de las soluciones ya existentes, el siguiente paso ha sido el de definir el conjunto de funcionalidades básicas que aquella que sea elegida debería cumplir con el fin de dar soporte a los objetivos de la plataforma a desarrollar:

- Construcción y despliegue del plan de entrenamiento. El marco deberá permitir la definición de las fases de entrenamiento (parámetros iniciales del modelo, definición de los datos, flujo de trabajo del entrenamiento), datos a intercambiar, lista de recursos del sistema.

- Ejecución y monitorización del plan de entrenamiento. El marco deberá permitir la evaluación de la calidad de los datos y centrarse tanto en el proceso de comprobación antes de la ejecución como en el seguimiento del progreso del entrenamiento.
- Bibliotecas de IA. El marco deberá ofrecer la posibilidad de utilizar diferentes bibliotecas de IA de acuerdo con las necesidades de los científicos.
- Estrategia de agregación. El sistema deberá permitirá determinar la combinación de la información de los pesos de los diferentes modelos locales al nodo central, incluyendo la posibilidad de personalizar la estrategia aparte de la predeterminada.
- Facilidad de personalización. Permitirá la posibilidad de incluir nuevas funcionalidades sobre la plataforma según necesidades.
- Documentación y ejemplos. El sistema deberá ofrecer un conjunto de información disponible para ayudar a los desarrolladores y científicos a instalar, ejecutar y controlar el marco.
- Código abierto. El marco FL seleccionado debe tener una licencia de código fuente que permita modificar y contribuir al código y que además sea comercialmente amigable para una posible explotación.

Aun cuando todas estas necesidades son cubiertas de un modo u otro por cada uno de los marcos analizados, dos han sido los aspectos clave adicionales que nos han permitido acotar más la selección y enfocarnos sólo en los más apropiados. De esta forma, uno de los aspectos más diferenciales es la necesidad de que el marco elegido sea compatible con el máximo número de librerías de AI de Python y que soporte pytorch, puesto que es la librería que se va a utilizar para validar la plataforma en el caso de uso. El siguiente ha sido la necesidad de que sea open source con el fin de reducir los costes en licencias y permitir auditar el código, aspecto muy importante teniendo en cuenta la particularidad de los datos clínicos. En relación a esto, además normalmente los open source tienen una comunidad que facilita la puesta en común del conocimiento, facilitando con ello su transferencia y documentación. Con estos dos aspectos clave se han podido descartar varios de los elementos iniciales de la lista, permitiendo quedarnos con solo tres: Flower, Fedbiomed y FATE.

FATE

FATE es un proyecto de código abierto iniciado por el Departamento de Inteligencia Artificial de Webank para proporcionar un marco de computación seguro que apoye el ecosistema de AI federada. Implementa múltiples protocolos de computación segura para permitir la colaboración de big data con el cumplimiento de la normativa de protección de datos. Una vista de su arquitectura puede verse en la Figura 1. FATE esta compuesto por los siguiente módulos:

- **FATE Flow**: es el núcleo del framework y componente principal, se encarga de preprocesar los datos, entrenarlos y probar los modelos.
- **FATE Board**: herramienta que muestra una interfaz que permite analizar y explorar los modelos visualmente.
- **FederatedML**: incluye la implementación de muchos modelos comunes de FL, así como herramientas que facilitan el ML como herramientas de codificación, módulos estadísticos, definiciones de parámetros y autogenerador de variables de transferencia, etc.
- **FATE Serving**: Un sistema de servidores escalable y de alto rendimiento para modelos de aprendizaje federados.
- **FATE Network**: la red para la comunicación, permite las comunicaciones cruzadas entre entre los nodos federadosFL
- **KuberFATE**: gestiona las cargas de trabajo de FL utilizando tecnologías nativas de la nube, como los contenedores. Permite que los trabajos de FL se ejecuten en entornos de nube pública, privada e híbrida.

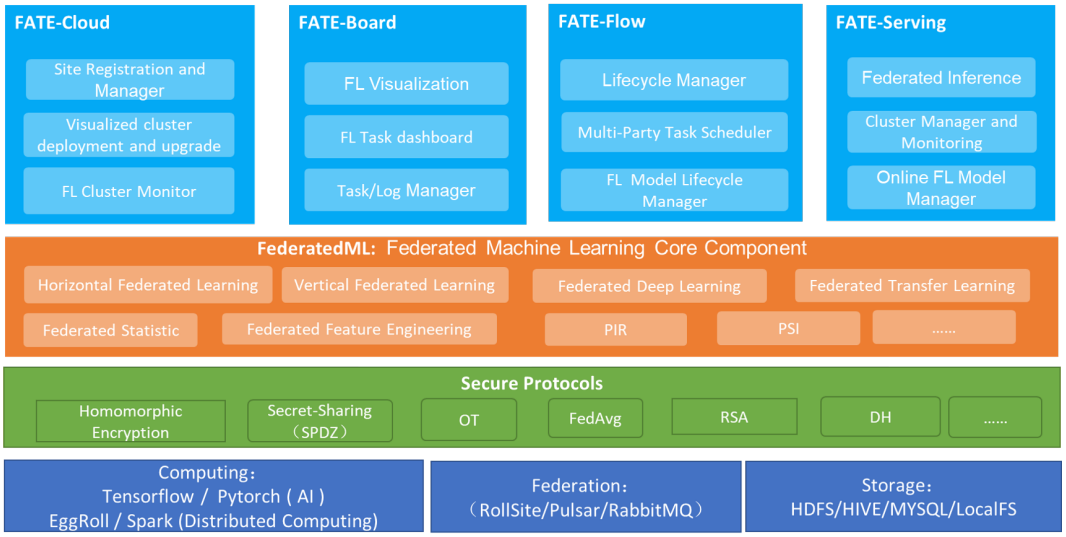

Figura 1: Arquitectura de FATE [33]

Flower

Flower es un framework de FL agnóstico y escalable de código abierto que maneja las comunicaciones a través del protocolo RPC (Remote Procedure Call) [35]. RPC tiene una buena interacción con los contenedores docker ya que no necesita publicar los puertos de los nodos trabajadores (clientes). Flower es sencillo de utilizar y admite numerosos frameworks o incluso código python personalizado y ofrece una API fácil de usar. La arquitectura de Flower se puede ver en la Figura 1.Las características principales de Flower son:

- **Personalizable**: Flower da la posibilidad de crear distintas configuraciones dependido de las necesidades de los distintos sistemas de FL.
- **Escalabilidad**: A la hora de diseñar Flower se pensó en su escalabilidad por lo que la mayoría de los componentes se pueden ampliar.
- **Agnóstico**: Una de las mayores ventajas de Flower es que no depende de ninguna librería de AI así que puede funcionar con PyTorch, TensorFlow, Keras, scikit-learn, MXNet, JAX,...
- **Usabilidad**: Fácil de aprender e intuitivo, con mucha documentación y una comunidad muy activa.

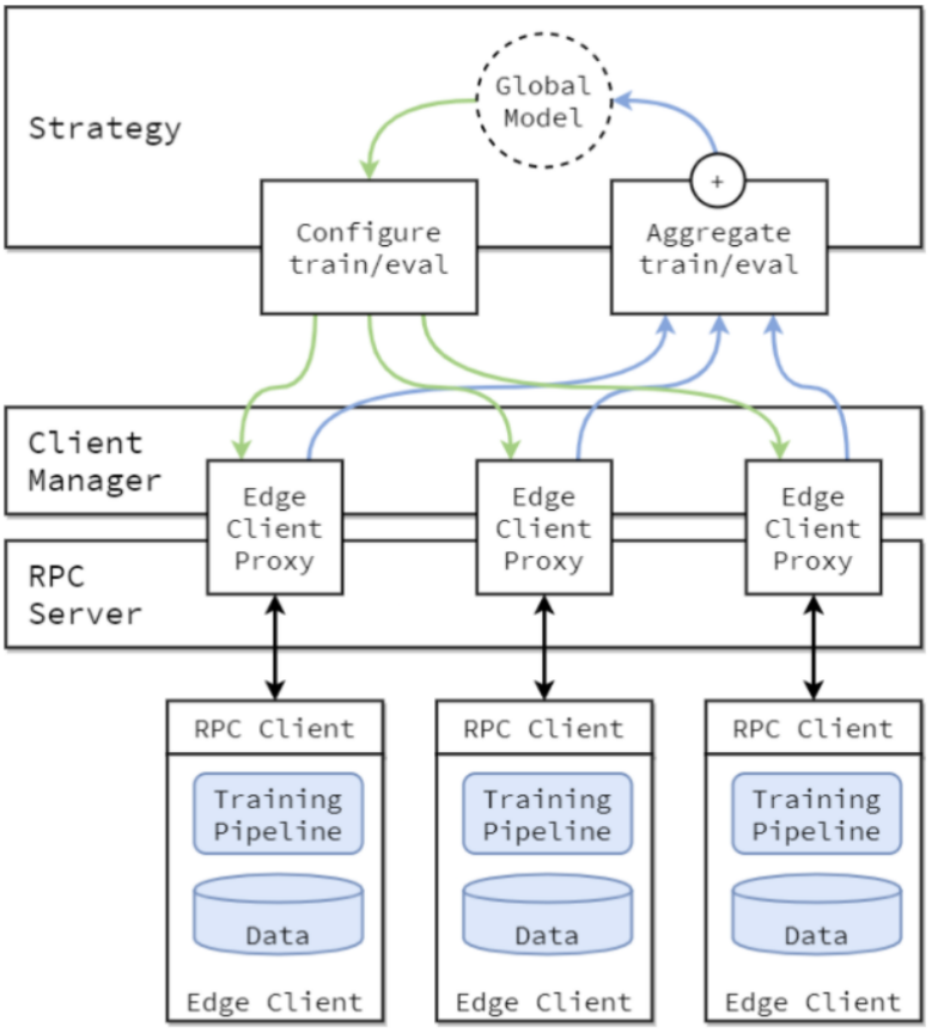

Figura 2: Arquitectura de Flower [35]

Fedbiomed

Fedbiomed es un proyecto de código abierto centrado en potenciar la investigación biomédica utilizando enfoques no centralizados para el análisis estadístico y el aprendizaje automático. El proyecto se basa actualmente en Python, PyTorch y Scikit-learn, y permite desarrollar y desplegar análisis de aprendizaje federado en aplicaciones de aprendizaje automático del mundo real. La arquitectura de Fedbiomed se puede observar en 3

El objetivo de Fedbiomed es proporcionar un entorno simplificado y seguro para:

- Desplegar fácilmente marcos de análisis de aprendizaje federado de última generación.
- Proporcionar una interfaz de usuario amigable para compartir datos para experimentos de aprendizaje federado.
- Permitir a los investigadores desplegar fácilmente sus modelos y métodos de análisis.
- Fomentar la investigación y las colaboraciones en el aprendizaje federado.

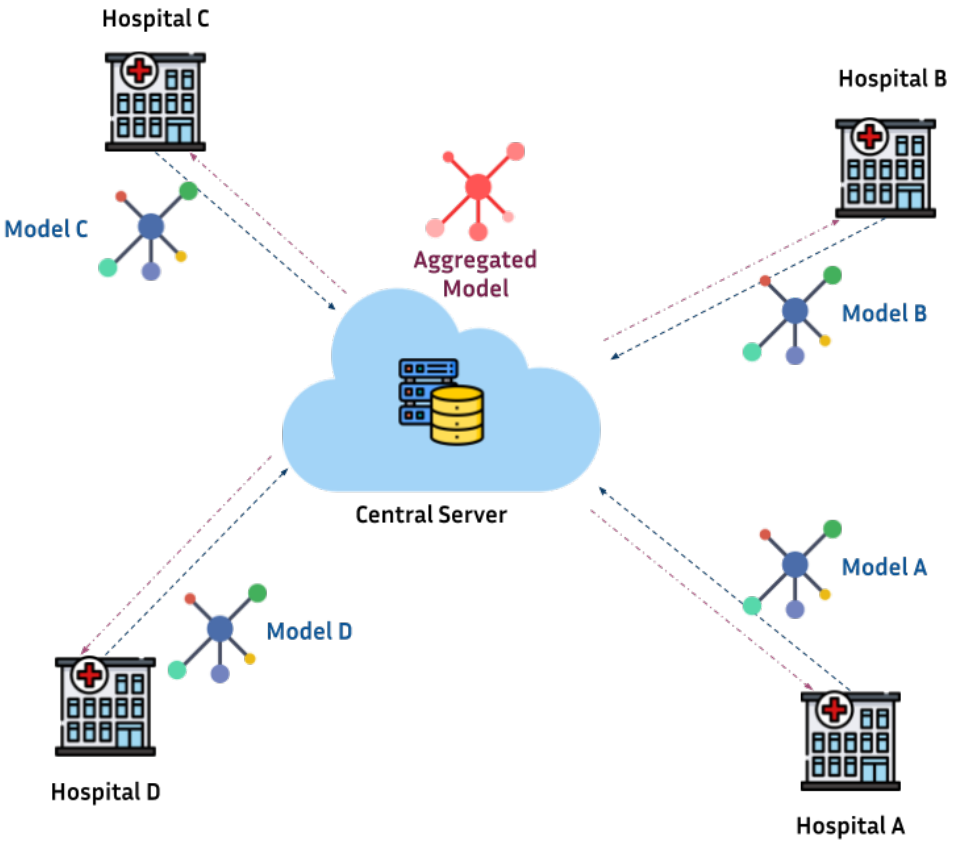

Figura 3: Arquitectura de Fedbiomed [36]

Comparativa

En esta sección se desglosan los diferentes aspectos comparados sobre las plataformas elegidas. Se puede ver un resumen de estas en la tabla 1

#	Criterio	FATE	Flower	FedBiomed
1	Construcción y despliegue del plan de entrenamiento	Sí	Sí	Sí
2	Ejecución y monitorización del plan de entrenamiento	Lenguaje específico de dominio para definir el plan Logs para monitorizar	Clase Python para definir el plan Logs para monitorizar	Lenguaje específico de dominio para definir el plan Logs para monitorizar
3	Seguridad de comunicaciones	-	SSL	VPN
4	Librerias AI		Todas las librerias de Python	PyTorch y un subconjunto de ScikitLearn
5	Estrategia de agregación	Sí, customizable	Sí, customizable	Sí, customizable
6	Extensibilidad del código	Media	Fácil	Media
7	Documentos y ejemplos	Medio	Medio	Bajo
8	Código abierto	Sí	Sí	Sí
9	Licencia	Apache 2.0	Apache 2.0	Apache 2.0 modificada FedBiomed propietario

Tabla 1: Comparativa de los tres frameworks elegidos

Construcción y despliegue del entrenamiento

- Fate: El primer paso consiste en definir un objeto parámetro, registrar la información de metadatos del nuevo componente y, a continuación, definir el objeto variable de transferencia.
- Flower: El plan de entrenamiento tiene que ser descargado desde un lugar central a los nodos.
- FedBiomed: El plan de entrenamiento se define junto con el modelo, y se almacena en un framework django, los nodos lo descargan del servidor una vez que inician el proceso FL.

Ejecución y monitorización del entrenamiento

- Fate: La ejecución se realiza mediante el envío de un trabajo a través del Lenguaje Específico de Dominio (DSL) de Fate y la monitorización se realiza mediante la comprobación de los archivos de registro.
- Flower: La ejecución se realiza llamando a la clase flower de python y toda la información relevante del proceso de entrenamiento se imprime en el log.
- FedBiomed: La ejecución se realiza utilizando el DSL de Fedbiomed y toda la información relevante se imprime en el log.

Autenticación / seguridad

- Fate: No está claro en la documentación cómo se implementa la seguridad.
- Flower: La comunicación GRPC entre los nodos y el servidor se puede hacer vía SSL, y se necesita más trabajo para implementar también un nivel de autenticación basado en token.
- FedBiomed: La seguridad se proporciona mediante la creación de una VPN entre los nodos.

Bibliotecas de IA

- Fate: basada en Python. En las últimas versiones está empezando a soportar algunas características de Pytorch, tensorflow y ScikitLearn.
- Flower: Basada en Python. Bibliotecas generales: TensorFlow, PyTorch, scikit, sklearn, modelo basado en numpy, muy flexible.
- FedBiomed: Sólo Pytorch por ahora y algunos modelos de scikit learn.

Estrategia de agregación

- Fate: La clase agregadora puede ser extendida.
- Flower: El servidor Flower no prescribe una forma de persistir las actualizaciones del modelo o los resultados de la evaluación,no guarda (todavía) automáticamente las actualizaciones del modelo en el lado del servidor. Está en la hoja de ruta proporcionar una manera incorporada de hacer esto. Las actualizaciones de los modelos pueden persistir en el lado del servidor mediante la personalización de los métodos de estrategia. La implementación de estrategias personalizadas es siempre una opción, pero para muchos casos puede ser más conveniente simplemente personalizar una estrategia existente.
- FedBiomed: Actualmente trabajan en proveer varias soluciones para esta heterogeneidad. Hasta ahora, soportan una función llamada FedAverage() que realiza el esquema de agregación estándar en el aprendizaje federado: el promedio federado. Realiza una media ponderada de los parámetros del modelo local basada en el tamaño de los conjuntos de datos específicos de los nodos. Esta operación se produce después de cada ronda de entrenamiento en los nodos. Es posible crear su agregador personalizado creando una nueva clase que herede de la clase Aggregator.

Personalización

- Fate: Es un marco bastante complejo y cubre muchos aspectos de un marco FL.
- Flower: Flower es un framework "minimalista"que ofrece las funcionalidades básicas de un framework FL. Esta filosofía/enfoque permite una fácil personalización e integración con otras herramientas/módulos. Se puede construir fácilmente sobre él.
- FedBiomed: Algunos módulos pueden ser fácilmente extendidos/personalizados, como el cargador de datos, mientras otros como la distribución del modelo a los nodos son más difíciles de personalizar.

Documentación y ejemplos

- Fate: Documentación de tamaño medio. Sin embargo, en las pruebas iniciales para instalar el framework no se pudo instalar por problemas sin solución documentada. Parte de la documentación no está traducida al inglés.
- Flower: Documentación de tamaño medio. Incluye varios ejemplos.
- Documentación breve con los pasos básicos para ejecutar el framework.

Por qué Flower

Aunque FedBiomed es framework de FL muy prometedor y completo que cubre muchas de las necesidades, hay varios aspectos que obligaron a su descarte como la gestión de los derechos de propiedad intelectual, ya que todas las contribuciones al framework deben transferir la propiedad de los derechos de propiedad intelectual a INRIA (The National Institute for Research in Digital Science and Technology) [40]y también el hecho de que por el momento sólo pytorch está totalmente soportado. Además, otra fuente de riesgo fue el uso del protocolo MQTT (MQ Telemetry Transport) dado que, aunque es óptimo para mensajes cortos, no lo es para mensajes más pesados.

En el caso de FATE, aunque es un framework de FL muy completo y se gestiona bajo la fundación Linux, hay muchos módulos y características y no siempre la documentación cubre bien todos los aspectos, por lo que la extensión y personalización del framework FL para los propósitos de este trabajo puede ser un

reto. Además, no está claro si FATE puede soportar todas las librerías python de AI y cuánto esfuerzo hay que hacer para añadir nuevas librerías, dado que por el momento está muy centrado en TensorFlow y PyTorch. De todas formas, es un framework de FL que puede ser utilizado como implementación de referencia para muchos aspectos y podría ser considerado como una segunda opción en el caso de que Flower no rinda como se espera.

Flower es un framework FL minimalista construido sobre bases sólidas como gRPC (GoogleGoogle Remote Procedure Calls) y protobuf, además es muy flexible en cuanto a las librerías de AI de python y las estrategias de agregación, necesitará integrarse con otros módulos ya existentes para cubrir todas las necesidades que requerirá la plataforma y parece que es el framework FL más fácil de personalizar o ampliar. Dado que los requisitos definidos son genéricos, una solución más madura podría plantear algunos problemas para adaptarla a las necesidades del trabajo. Por este motivo, entre otros aspectos expuestos anteriormente, la solución seleccionada es Flower.

Diseño

Análisis de requisitos

Requisitos Funcionales

Los requisitos funcionales determinan los servicios ofrecidos por el software. En este caso, la plataforma completa se compone tanto del Sandbox general como de la funcionalidad específica de FL. Los requisitos se pueden descomponer siguiendo la siguiente jerarquía:

1. Nodo central.

    a) Definir el plan de entrenamiento, incluyendo el modelo de ML que reside en tiempo de ejecución en los nodos satélite junto con la estrategia de agregación, el conjunto de datos federado a utilizar y el flujo de entrenamiento. Prioridad: necesario.

    b) Orquestar el despliegue de los contenedores de cómputo en los nodos satélite. Prioridad: necesario.

    c) Monitorizar la ejecución en cada nodo empleando el manejo de excepciones cuando sea necesario. Prioridad: necesario.

    d) Modificar el estado de ejecución de una tarea a cancelada. Prioridad: necesario.

    e) Guardar el estado final de agregación de la tarea de FL. Prioridad: necesario.

    f) Enviar el modelo agregado a los nodos satélite para comenzar una nueva iteración. Prioridad: necesario.

    g) Generación de documentación pública para la REST(transferencia de estado representacional) API siguiendo un estándar como OpenAPI. Prioridad: recomendable.

h) Ofrecer una interfaz de usuario sencilla y funcional para garantizar el acceso a los servicios sin necesidad de utilizar acciones REST. Prioridad: necesario.

i) Operaciones CRUD sobre los conjuntos de datos federados. Prioridad: necesario.

j) Operaciones CRUD sobre los usuarios y organizaciones. Prioridad: necesario.

k) Scripts de despliegue para facilitar la suscripción de nuevos nodos. Prioridad: recomendable.

l) Los estados de las tareas se guardan en un almacenamiento persistente. Prioridad: recomendable.

m) Las comunicaciones entrantes y salientes se cifran utilizando un protocolo seguro. Prioridad: necesario.

n) La estrategia de agregación opera con datos que pueden proceder de cualquier modelo escrito en Python independiente de la librería de la cual provenga. Prioridad: necesario.

2. Nodo satélite.

a) Lanzar la ejecución de un cliente de Flower asociado al subconjunto de datos federado de ese nodo satélite. Prioridad: necesario.

b) Terminar la ejecución del cliente Flower descrito en el punto anterior. Prioridad: necesario.

c) Limpiar el entorno de contenedores de cómputo para permitir el despliegue de una nueva tarea. Prioridad: necesario.

d) Generación de una clave para autorizar comunicaciones entrantes del nodo central. Prioridad: recomendable.

e) Generación de documentación pública para la REST API siguiendo un estándar como OpenAPI. Prioridad: recomendable.

f) Las comunicaciones entrantes y salientes se cifran utilizando un protocolo seguro. Prioridad: necesario.

g) Los contenedores que ejecutan el cliente de Flower son flexibles respecto a las librerías utilizadas. Prioridad: necesario.

Requisitos no funcionales

Los requisitos no funcionales más importantes son los siguientes:

1. Utilizar el protocolo gRPC, basado en HTTP2, para transmitir la información de los modelos entre los contenedores de cómputo de Flower.
2. Utilizar el protocolo HTTPS para cifrar las comunicaciones de las REST API.
3. Utilizar contenedores para desplegar los servicios y así garantizar el aislamiento, seguridad e independencia del hardware.
4. El almacenamiento persistente utiliza una base de datos relacional.

Arquitectura

Para el desarrollo de esta plataforma se han utilizado diversas tecnologías. Para poder entender la arquitectura se procede a describir brevemente cada una de ellas.

Técnología utilizadas

Contenerización

La contenerización es la virtualización a nivel de sistema operativo o de aplicación sobre múltiples recursos de red para que las aplicaciones de software puedan ejecutarse en espacios de usuario aislados llamados contenedores en cualquier entorno, independientemente del tipo o del hardware. En esta plataforma se va a utilizar para desplegar los distintos módulos, de forma que cada uno sera independiente. Se va a utilizar Docker para la contenerización. Todos los módulos tienen un archivo Dockerfile. Docker construye la imagen del contenedor leyendo el archivo Dockerfile, que consiste en una serie de instrucciones o comandos necesarios para construir dicha imagen. Para desplegar,combinar y configurar múltiples contenedores al mismo tiempo docker utiliza el archivo docker-compose.yml. En este archivo estará la configuración que comparten los distintos módulos de la plataforma.

Interfaz de programación de aplicaciones (API)

Una API es una forma para que dos o más programas informáticos se comuniquen entre sí. Es un tipo de interfaz de software que ofrece un servicio a otra pieza de software[1]. Un documento o estándar que describe cómo construir o utilizar dicha conexión o interfaz se denomina especificación API. Se dice que un sistema informático que cumple esta norma implementa o expone una API. El término API puede referirse tanto a la especificación como a la implementación. En la plataforma de federated learning se va a utilizar para la comunicación entre los distintos nodos.

API Gate Away

Una API gate away es una herramienta de gestión de APIs que se sitúa entre un cliente y un conjunto de servicios backend. Una pasarela de API actúa como un proxy inverso para aceptar todas las llamadas a la API, agregar los distintos servicios necesarios para cumplirlas y devolver el resultado apropiado. Kong es una API gateaway utilizado en la plataforma para unificar los servicios de lo distintos módulos en un solo proxy.

Frameworks

Un Framework es una herramienta que proporciona componentes listos o soluciones que se personalizan para acelerar el desarrollo y construir sobre ellos una solución. En esta plataforma se han utilizado los siguientes frameworks:

- **Flower** es un framework para construir sistemas de Federated Learning, explicado en el apartado 2.3.2.
- **Fast API** es un framework basado en python para construirAPIs
- **Flask** es un framework basado en python para construir aplicaciones web.

Sandbox

Se ha llamado sandbox a la plataforma generada para el entrenamiento de algoritmos de machine learning de manera federada. El sandbox esta compuesto por un nodo central y dos o más nodos satélites. El nodo central es el encargado de gestionar todo el proceso y los nodos satélites son los encargados de alojar los datos y ejecutar los algoritmos de forma local. Cada nodo esta compuesto por distintos módulos. Cada módulo es un contenedor independiente con su propio Dockerfile con las instrucciones necesarias para construir su imagen. Estos módulos comparten un archivo Docker-compose.yml en el se configura como debe interactuar cada contenedor. Cuando se ejecuta una tarea, varios módulos temporales son desplegados, esto es, el servidor central de Flower en el nodo central y clientes de Flower en los nodos satélites. La comunicación entre los distintos nodos se produce utilizando el protocolo HTTP, que para su seguridad esta protegido por API-Keys.

Nodo Central

El nodo central está compuesto por una base de datos el orquestador y el manager, que a su vez esta compuesto por una interfaz de usuario y una API.Cuando

una tarea se esa ejecutando, el nodo central crea un módulo temporal con un contenedor para el despliegue del servidor central de Flower. Se puede ver con mas detalle en la Figura 3.1.

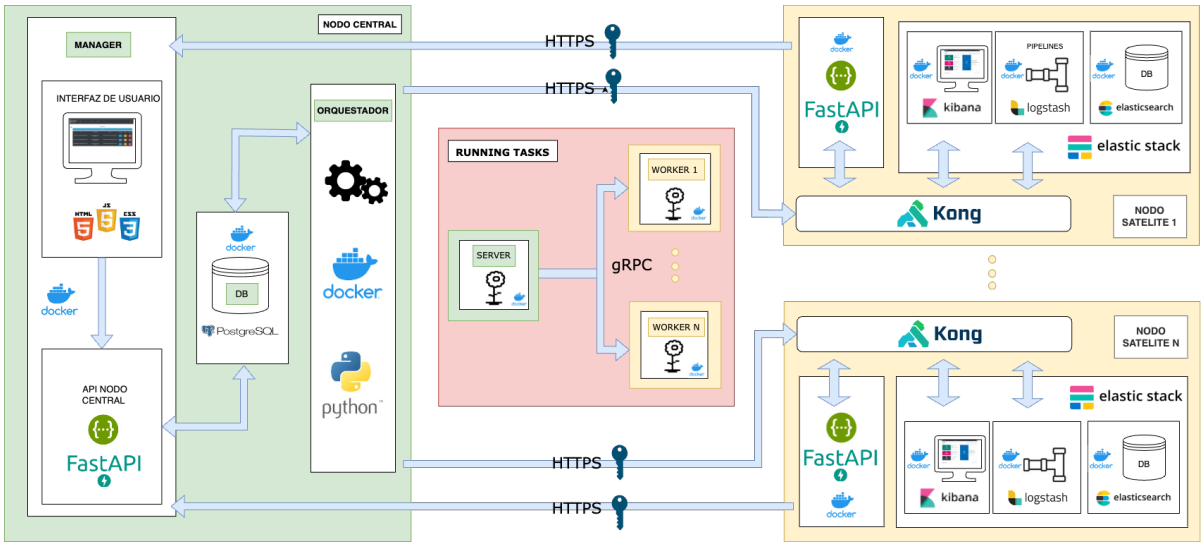

Figura 4: Arquitectura del Sandbox

Manager

Es el encargado de la gestión de la plataforma y de presentar la interfaz para el usuario. El manager permite gestionar las siguientes funcionalidades:

- **Usuarios** permite las operaciones crear, leer, actualizary borrar (CRUD) sobre los usuarios de la plataforma.
- **Groupos** permite las operaciones CRUD sobre los grupos de la plataforma.
- **Nodos** permite las operaciones CRUD sobre los nodos de la plataforma.
- **Índices** permite las operaciones CRUD sobre los índices ya existentes en los nodos.
- **Tareas** permite las operaciones CRUD sobre las tareas de la plataforma, así como su validación para la ejecución. sobre los índices ya existentes en los nodos.
- **Historial** permite consultar el historial de las tareas y recuperar los datos de los entrenamientos.

- **Monitorización** permite consultar la cola de tareas y las tareas que están ejecutándose actualmente.

API La API del manager esta desarrollada en FastAPI y su objetivo es cumplir tres funcionalidades:

1. permitir a la interfaz comunicarse con el backend del manager
2. permitir al servidor central de Flower comunicarse con el backend del manager
3. permitir a los nodos satélites comunicarse con el nodo central

Las únicas llamadas expuestas en el openAPI son, por un lado, la utilizada por los nodos satélites para comunicar la finalización de una tarea en su nodo y, por otro, la utilizada por el servidor central de Flower para comunicar la finalización de la tarea. FastAPI crea automáticamente la documentación de la API, tal y como se muestra en la Figura 5

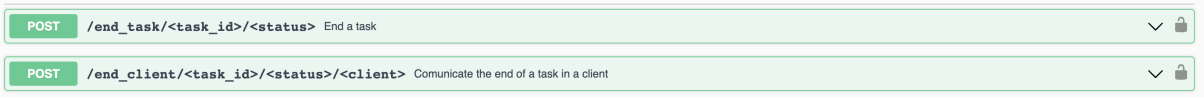

Figura 5: Especificacion de API Nodo Central

Interfaz de Usuario Para la interfaz se ha utilizado Jinja [41], que es un motor de plantillas web utilizado ampliamente en Python. Por su parte las plantillas han sido desarrolladas en HTML+CSS+JS. La interfaz se compone de las siguientes pantallas:

login Esta pantalla sirve para que el usuario inserte sus credenciales y acceda a la plataforma . Consta de un formulario para el envío de las credenciales como puede verse en la Figura 6 . No existe pantalla de registro, es el administrador de la plataforma quien tiene que crear los usuarios y asignarlos a los grupos.

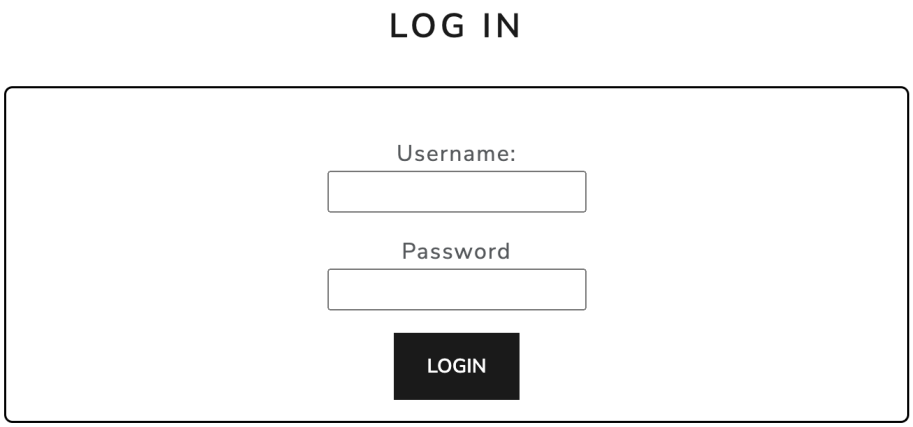

Figura 6: Pantalla de Login

Menú El menú es una barra de navegación que pueden ver todos los usuarios logueados de la plataforma y que sirve para navegar por ella. En la Figura 7 se ve la barra de navegación con las distintas acciones: Consola, Current, History, New task, Validation y Logout. Las acciones no resaltadas son opciones que no están disponibles, bien por que no se tiene permiso de administrador o no existe contenido en esas pestañas.

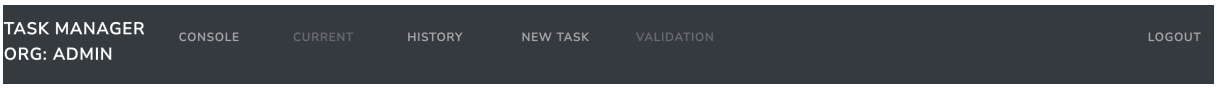

Figura 7: Menú de la aplicación

Menú lateral El menú lateral es el que pueden ver solo los administradores de la plataforma y que sirve para navegar por las funcionalidades para gestionar los diferentes recursos de los administradores, que son Usuarios, Organizaciones, Tareas, Nodos e Índices como se puede ver en la Figura 8.

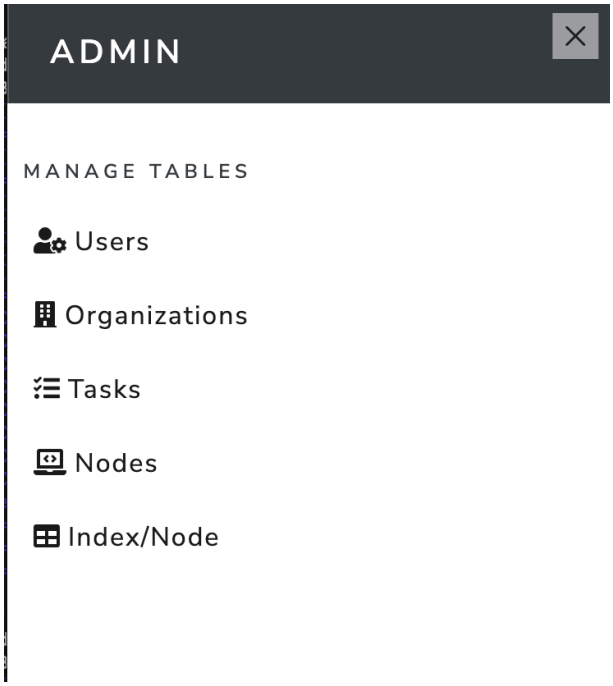

Figura 8: Menú de administrador

Usuarios En esta pantalla el administrador podrá gestionar los usuarios, pudiendo ver un listado de los mismos, editarlos y borrarlos o crear nuevos, como se puede ver en la Figura 9. El formulario para la creación de usuarios puede verse en la Figura 10

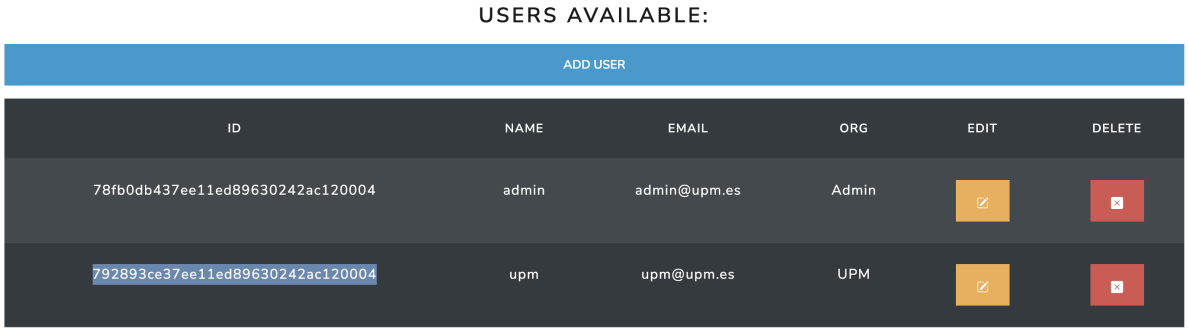

Figura 9: Pantalla de usuarios

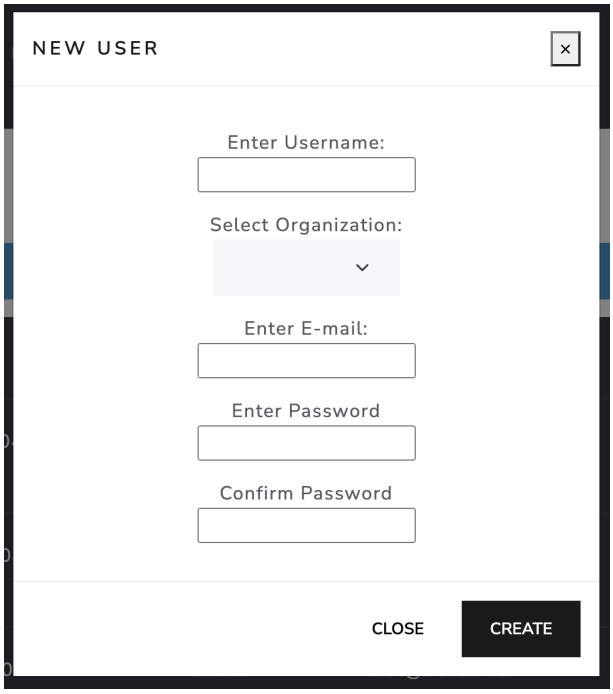

Figura 10: Detalle Pop up añadir usuario

Organizaciones En esta pantalla el administrador podrá gestionar los grupos, pudiendo ver un listado, editar, borrar o crear nuevos, como se puede ver en la Figura En la Figura 11, mientras la Figura 12 muestra el formulario para la creación de nuevos grupos.

Figura 11: Pantalla de organizaciones

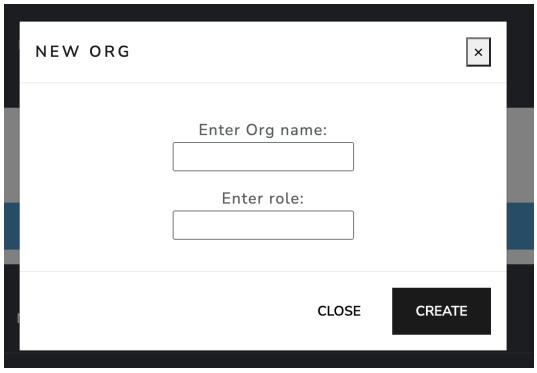

Figura 12: Detalle Pop up añadir grupos

Nodos En esta pantalla el administrador podrá gestionar los nodos satélites que vayan a componer la plataforma. En la Figura 13 se puede ver el listado de nodos existentes y las acciones a realizar sobre ellos, editar y borrar, y, además, se podrán crear nuevas con el formulario que se ve en la Figuras 14.

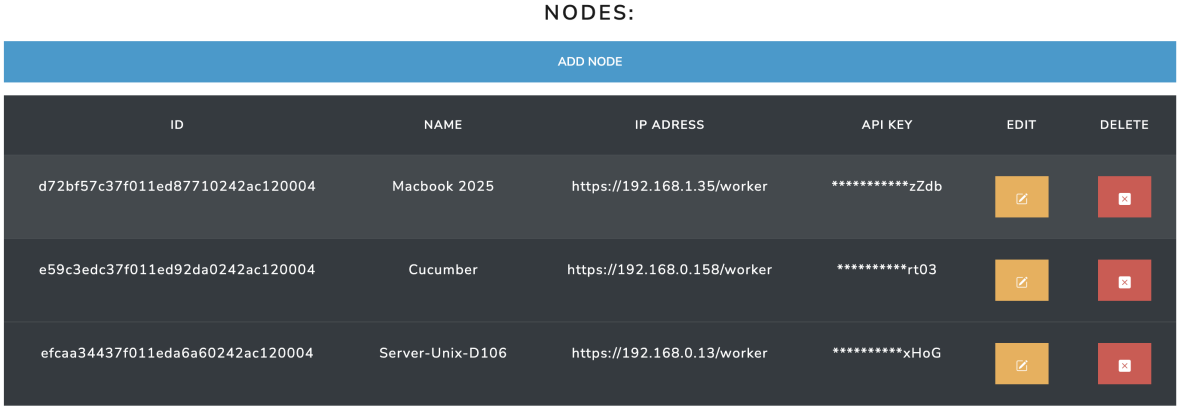

Figura 13: Pantalla de nodos

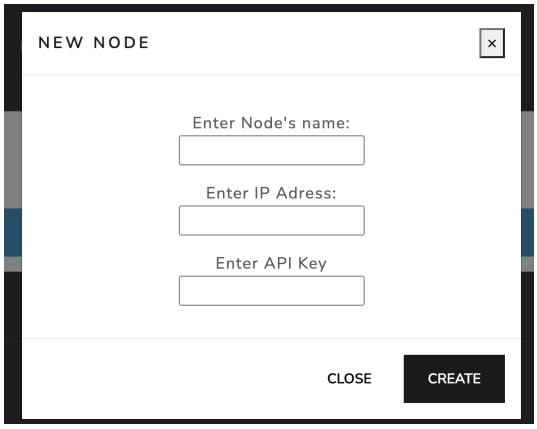

Figura 14: Detalle Pop up añadir nodos

Tareas En esta pantalla el administrador podrá gestionar y consultar la información de todas las tareas. En la Figura 15 se observa el listado de tareas, y el detalle de una de ellas, ofreciendo la posibilidad de editarlas o borrarlas.

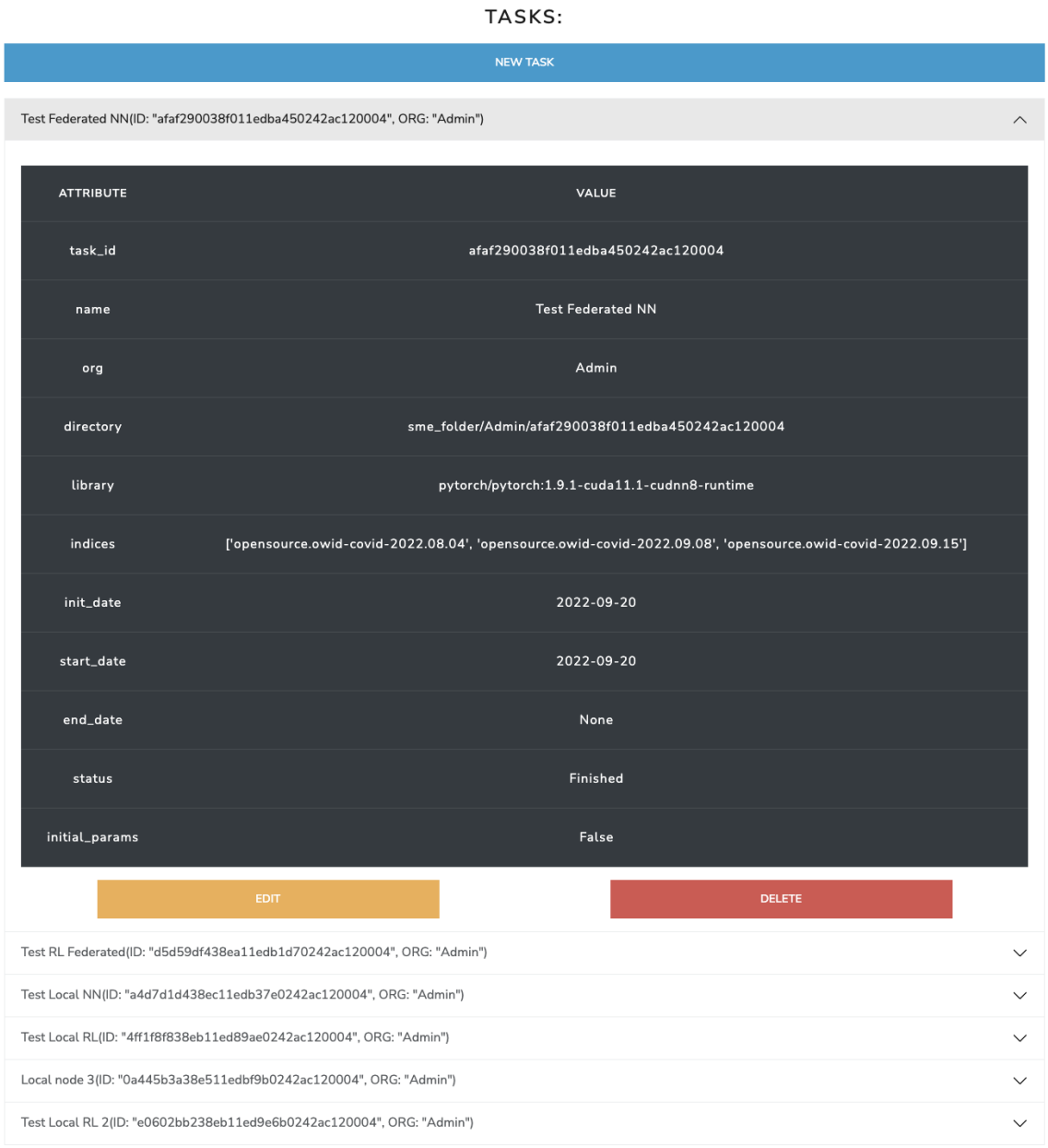

Figura 15: Pantalla de tareas

Índices En esta pantalla el administrador podrá gestionar los índices de los datasets en Elasticsearch(detallado en el apartado) disponibles en los nodos satélites. En la Figura 16 se ve un listado de todos los índices existentes con las acciones de enlazar con una organización, editar o borrar. En la Figura 17 se observa un formulario para la creación de un índice, donde habrá que indicar el

nodo en el que esta alojado.

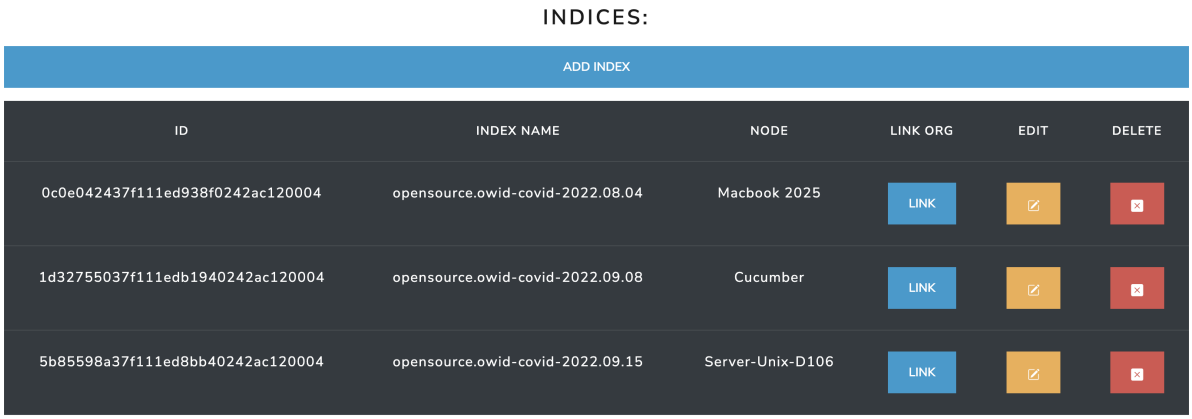

Figura 16: Pantalla de índices

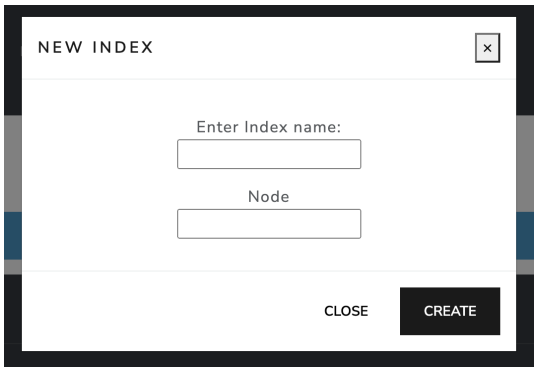

Figura 17: Detalle Pop up añadir índice

Creación de Tareas En esta pantalla los usuarios podrán crear tareas, para lo que tendrán que proporcionar los archivos clientes y servidor del algoritmo federado que quieren entrenar. Además, deberán seleccionar una de las imágenes preconstruída de docker con la librería elegida para el algoritmo, teniendo en la plataforma las opciones de Pytorch y TensorFlow. Por último tendrán que seleccionar los índices de datos en los que quieren ejecutar los algoritmos. Como campo opcional se puede subir un archivo con los pesos iniciales del modelo. En la Figura 18 se puede observar el formulario para poder crear la tarea y añadir la información antes explicada.

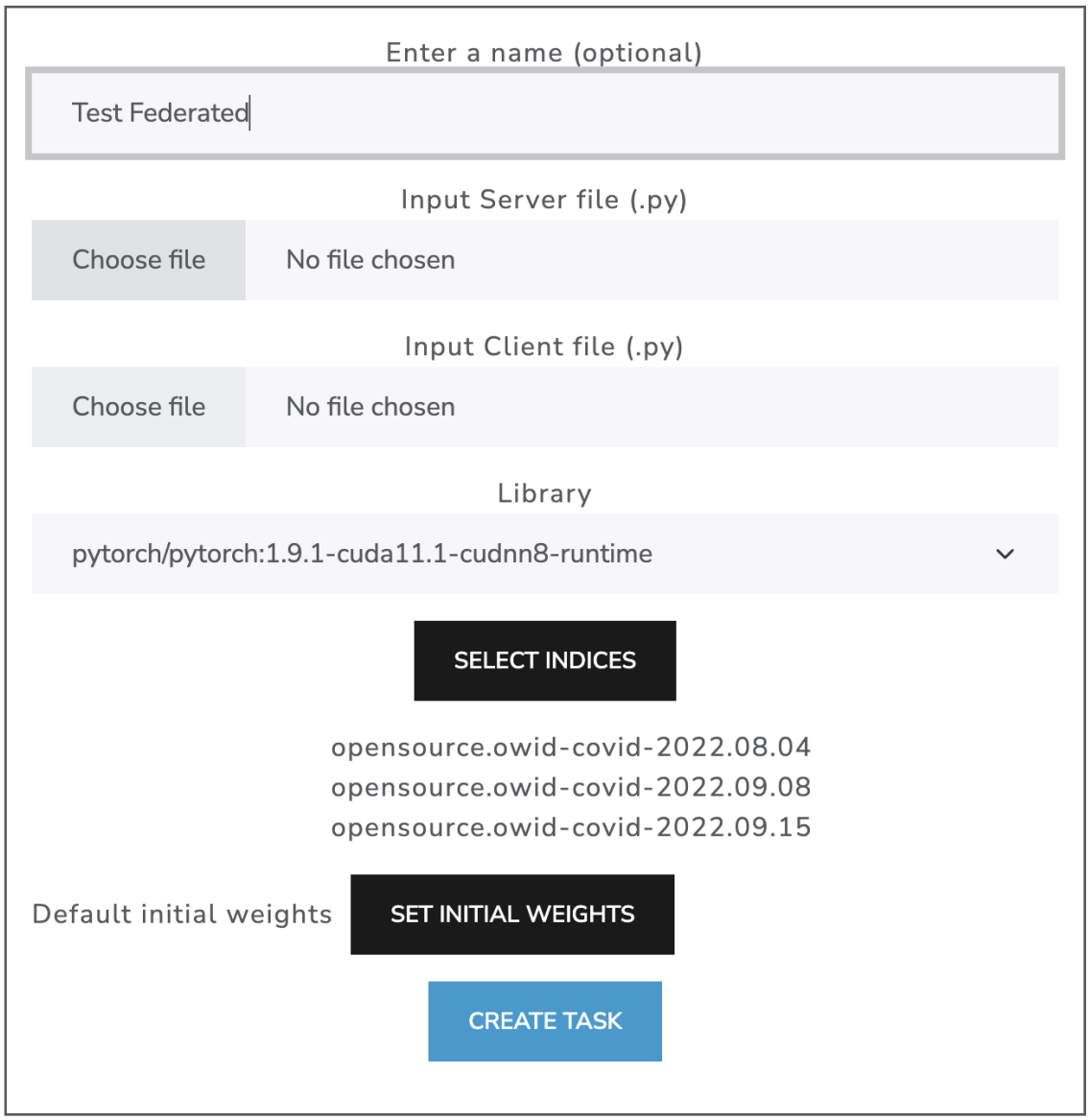

Figura 18: Pantalla de creación de tarea

Validación En esta pantalla el administrador podrá comprobar el código de las tareas subidas a la plataforma y validarlas para que las tareas sean añadidas a la cola de ejecución. En la Figura 19 se puede ver el listado de tareas a validar y en la Figura, 20 el detalle de una tarea, con su código y las acciones sobre esta, que son validar o rechazar.

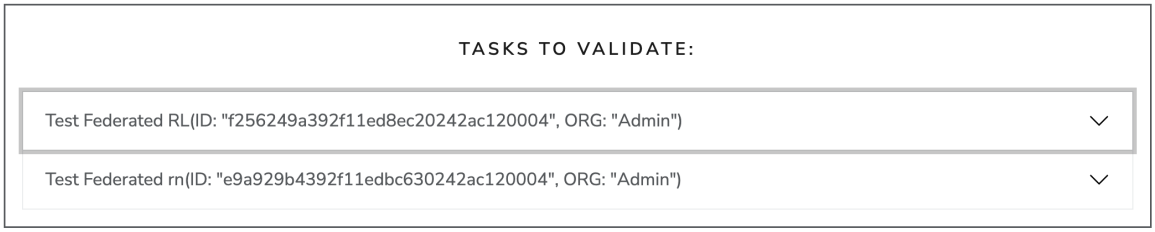

Figura 19: Pantalla de validación

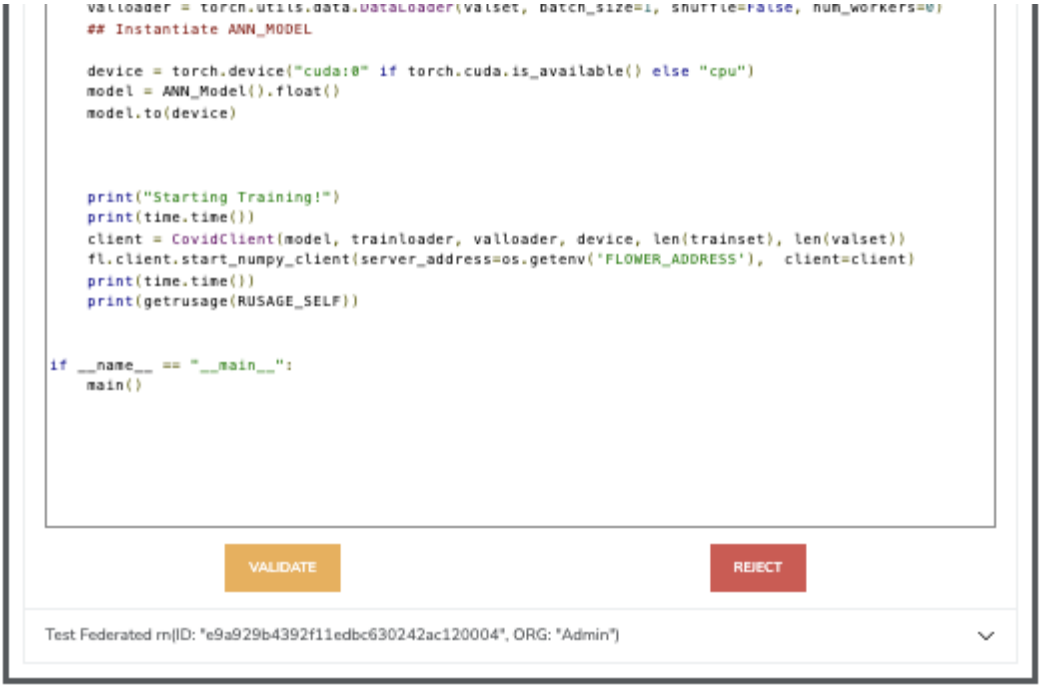

Figura 20: Detalle de validación de una tarea

Consola En esta pantalla el usuario podrá ver el estado de la cola de tareas, apareciendo en verde aquellas que se están ejecutando. En la Figura 21 se puede ver la cola de tareas, donde la que se está ejecutando tiene un fondo verde, mientas que las que están a la espera tienen fondo negro.

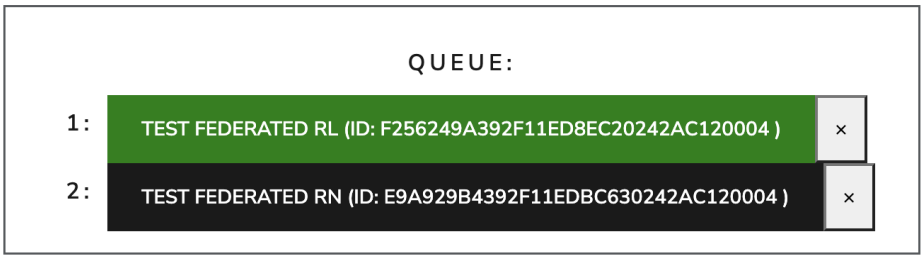

Figura 21: Pantalla de consola

Histórico En esta pantalla el usuario podrá ver el histórico de tareas finalizadas como se puede ver en la Figura 22. También podrá acceder al sumario de la tarea donde encontrará los archivos con los resultados de los entrenamientos y los archivos punto py del cliente y servidor como puede observarse en la Figura 23).

Figura 22: Pantalla de histórico

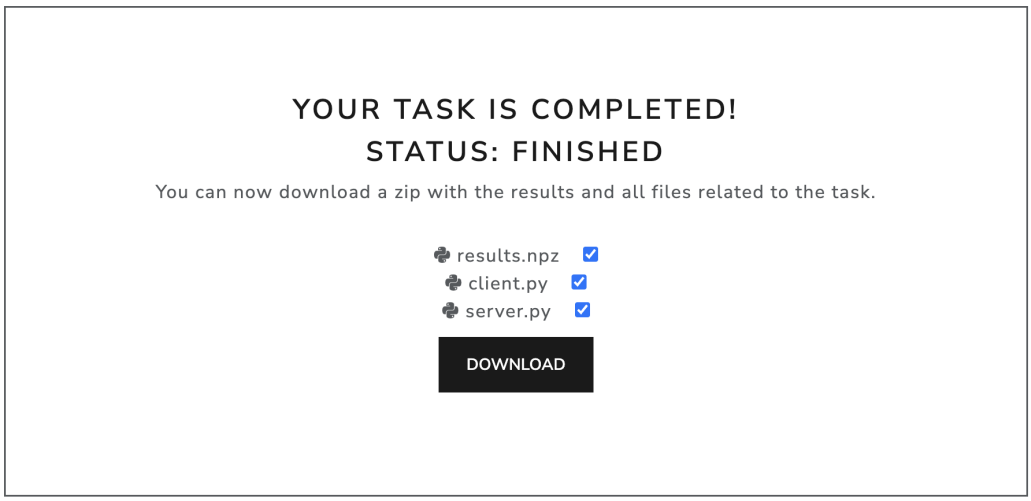

Figura 23: Detalle de finalización de tarea

Orquestador

El orquestador es el encargado de hacer el trabajo que no se ve de la plataforma, esto es, recibir comunicaciones del mánager y comunicarse con los nodos satélites. Una vez validada una tarea, el orquestador toma el mando y añade esa tare a la cola de ejecución, que queda a la espera de que sea su turno. Cuando este llega, crea en el nodo central un nuevo contenedor, utilizando la imagen de docker seleccionada a la hora de crear la tarea, para levantar el servidor central de Flower, y, a su vez, el orquestador se comunica con el nodo satélite a través de su API para comunicarle que tiene que desplegar un cliente de Flower, enviándole el nombre de la imagen y el archivo con el algoritmo. El orquestador está preparado para reprogramar, además, la cola de tareas en caso de errores a la hora de desplegar los contenedores. Una vez lanzada la tarea en los contenedores temporales se queda a la espera de mensajes por parte del mánager para dar por terminada la tarea.

Base de datos

El nodo central precisa de una base de datos para guardar persistentemente la información necesaria para la gestión de la plataforma, para lo que se ha utilizado una base de datos relacional del tipo PostgreSql [**Postgres2022**]. El diagrama de entidad/relación se pude ver en la Figura 24

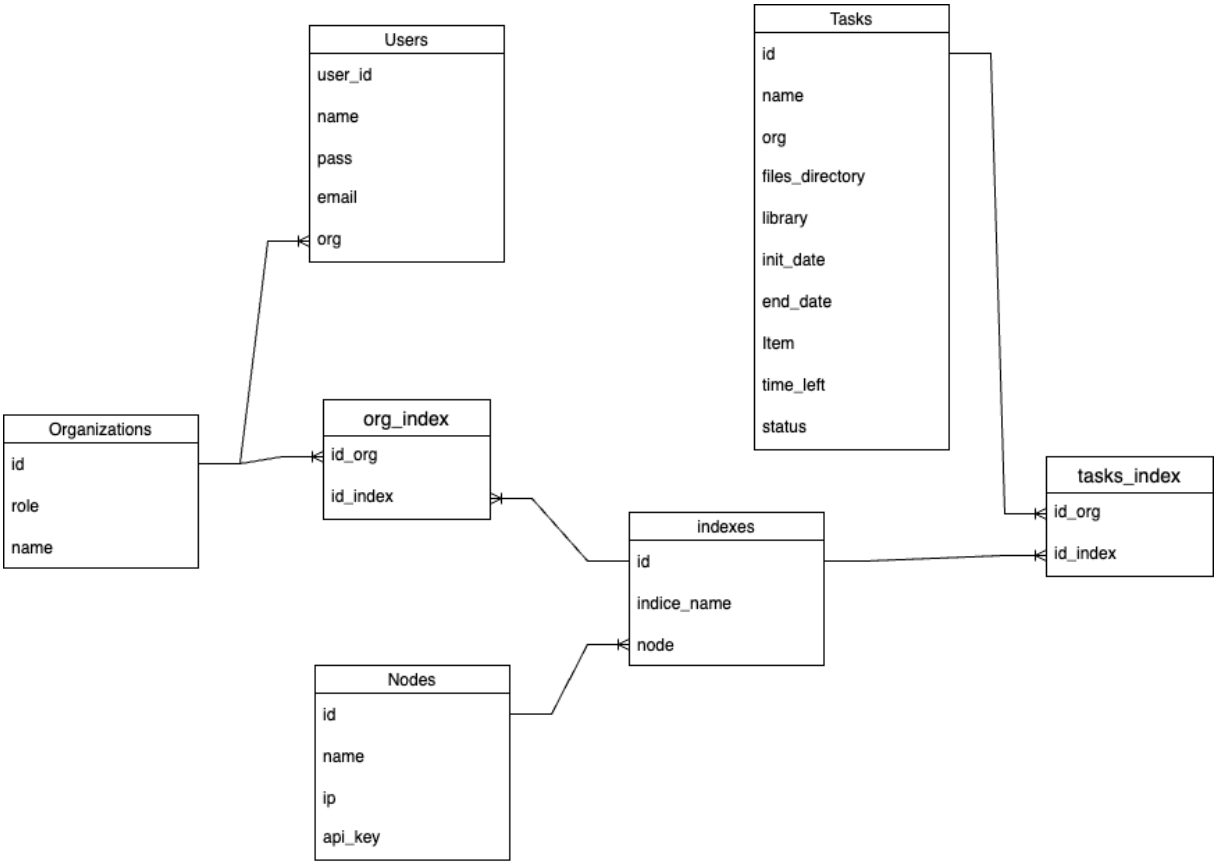

Figura 24: Diagrama Entidad / Relación

Servidor central de Flower

A la hora de ejecutar una tarea, se crea un contenedor virtual temporal que se basa en una imagen que contiene la librería elegida para esa tarea. En la plataforma solo se pueden elegir imágenes de pytorch y tensorflow aunque se podrían añadir más imágenes fácilmente, tal y como se buscaba. El contenedor ejecuta el servidor central que ha debido de ser añadido a la tarea como un archivo python. Este archivo debe seguir la documentación de Flower y será el encargado de agregar los resultados de los diferentes nodos satélites y actualizar el modelo con los nuevos valores. El contenedor se borra automáticamente cuando se ha terminado el entrenamiento.

Nodo Satélite

El nodo satélite es donde se realiza el entrenamiento real sobre los datos locales y está compuesto por una stack ELK(Elasticsearch Logstash kibana) [42] para el

almacenamiento y visualización de datasets, una API para la comunicación y una API gate away que hace de proxy de los diferentes módulos. Por último, también tiene un módulo temporal que es creado a la hora de ejecutar las tareas.

API

La API de un nodo satélite permite al nodo central comunicarse con el a través de llamadas Http. Cada nodo al iniciarse crea una API key única para ese nodo y que sirve para autenticar las llamadas. La API solo tiene dos llamadas, una para lanzar una tarea (run) y otra para pararla (stop). La llamada run levanta un contenedor temporal dentro del nodo que funcionará como cliente de Flower, mientras que la llamada stop cancelará la ejecución de ese contenedor y lo borrará. Está desarrollada en FastTAPI que crea automáticamente la documentación de la API, se pude ver la interfaz creada en la Figura 25

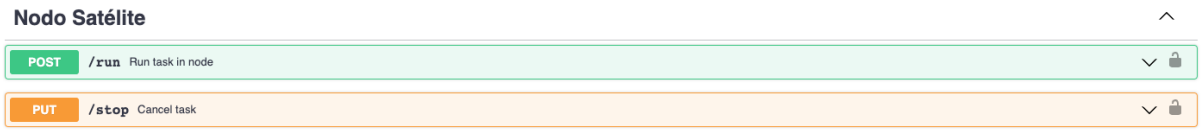

Figura 25: Especificación de la API del nodo satélite

Base de datos

Para el desarrollo de esta plataforma se ha elegido el stack ELK para el almacenamiento de los datasets [42].ELK es una colección compuesta por tres productos opensource:

- **ElasticSearch**Elasticsearch almacena e indexa los datos. Es una base de datos NoSQL basada en el motor de búsqueda de código abierto de Lucene.
- **logstash** Se utiliza para recopilar, analizar, transformar y almacenar en el búfer los datos de una variedad de fuentes. Los datos recogidos por Logstash son enviados a Elasticsearch mediante pipelines.
- **Kibana** actúa como una capa de análisis y visualización sobre Elasticsearch. Kibana puede utilizarse para buscar, visualizar e interpretar los datos almacenados en Elasticsearch y se ha utilizado el paquete de seguridad para proteger las datos con autenticación y autorización. Se puede ver una ejemplo de la interfaz en la Figura 26

En Elasticsearch los datos se guardan en índices, que se pueden crear y alimentar de datos a través de pipelines de logstash usando tanto ficheros como streams de datos .

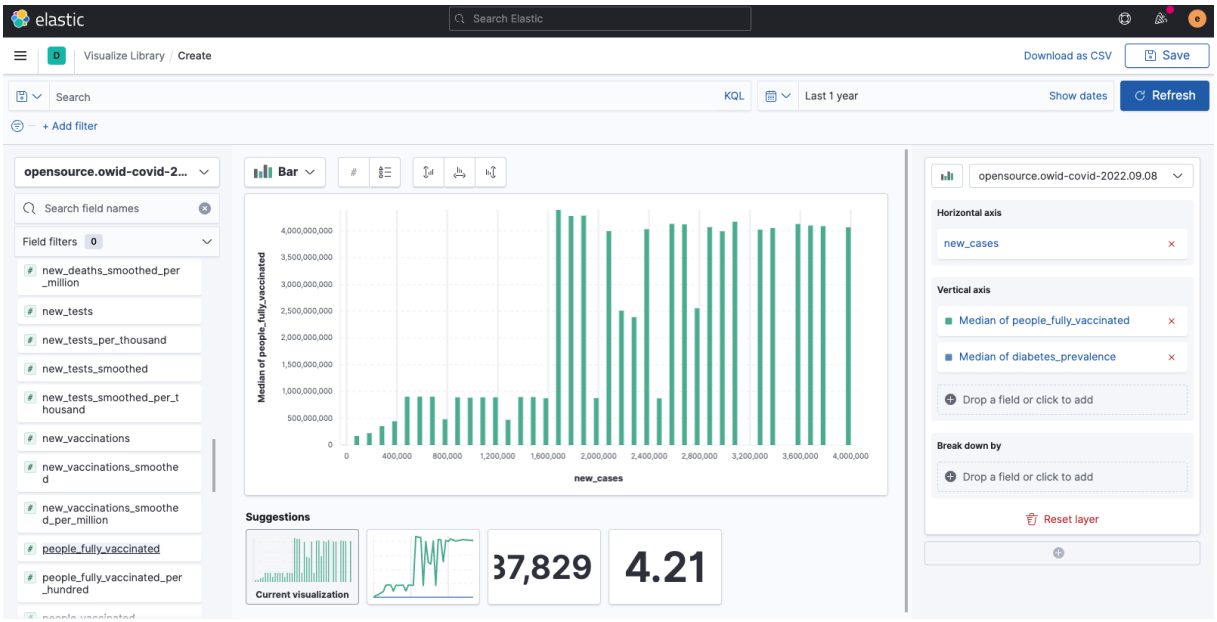

Figura 26: Pantalla de Kibana en uno de los nodos

API Gateaway

El API gateaway esta creado con Kong y conecta con los módulos unificándolos en el puerto 443 para acceder mediante HTTPS. La configuración de rutas y módulos se puede ver en la tabla 3.1

Nº	ruta	Modulo	llamadas
1	\	Kibana	GET POST PUT DELETE
2	$\backslash worker$	API	POST DELETE
3	$\backslash worker \backslash docs$	Especificacion de la api	GET
4	$\backslash api \backslash ingest\_stream \backslash \{index\}$	Logstash	POST
5	$\backslash api \backslash ingest\_file \backslash \{index\}$	Logstash	POST

Tabla 2: Tabla de rutas del nodo satélite

Cliente de Flower

Como en el nodo cental, a la hora de ejecutar una tarea se debe crear un contenedor temporal. El contenedor ejecuta un cliente con el script que ha debido de ser añadido a la tarea como un archivo python. Este archivo debe seguir la

documentación de Flower y será el encargado de entrenar y alimentar al nodo central con los resultados del entrenamiento y a su vez recibirá los nuevos pesos agregados para la siguiente interacción de entrenamiento. El contenedor se borra automáticamente cuando se ha terminado el entrenamiento.

Despliegue, configuración y flujo de entrenamiento

Para el despliegue y configuración de la plataforma y la ejecución de una tarea de entrenamiento se deben seguir estos pasos:

1. Despliegue de nodos. El sandbox necesita tener desplegado el nodo central y mínimo dos nodos satélites. Una vez desplegados los nodos tenderíamos una arquitectura como la mostrada en la Figura 27

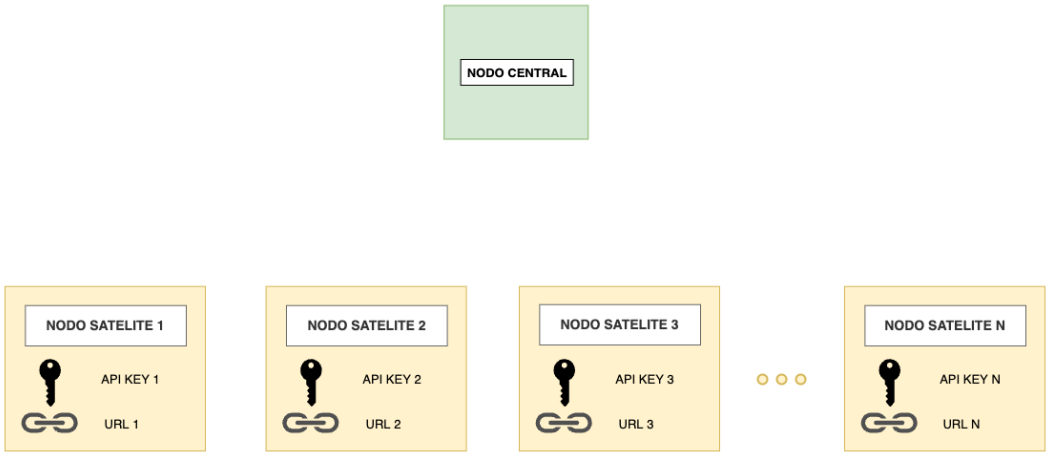

Figura 27: Paso 1 Despliegue de nodos

2. Asociación de nodos: A través de la interfaz de usuario del nodo central(Figura 14) se deben añadir los nodos satélite a la base de datos, indicando su url (Ej:https://[IP NODO 1]/worker) y la API Key generada al desplegar el nodo

3. Creación de los dataset: Usando el módulo de kibana o las APIs de los diferentes nodos satélites, se deben crear y alimentar con datos los datasets en índices de Elasticsearch.

4. Asociación de índices a nodos: A través de la interfaz de usuario del nodo central (Figura 17) se deben añadir los índices de los diferentes nodos satélite a la base de datos, indicando el nombre del índice y a qué nodo pertenece. En este estado la Arquitectura quedaría como se ve en la Figura 28

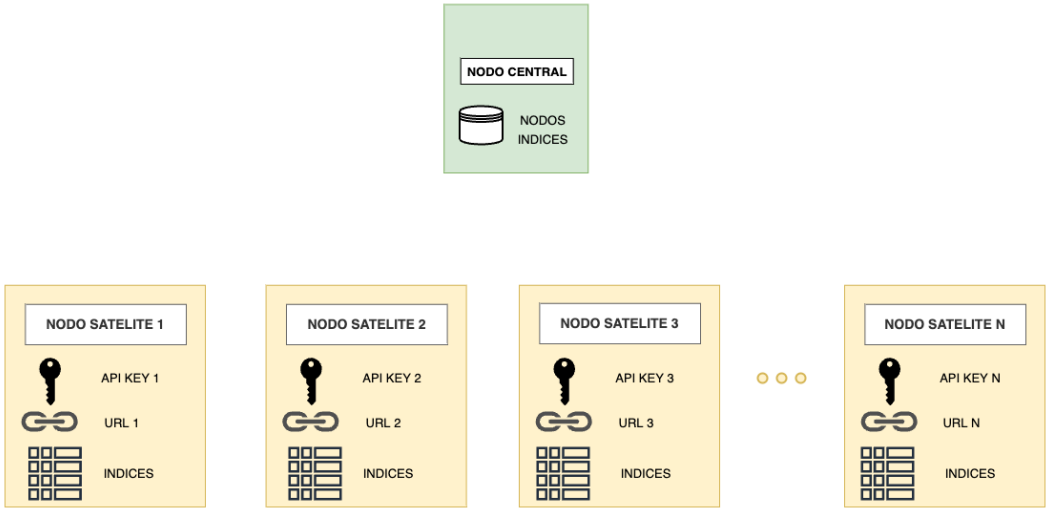

Figura 28: Arquitectura de la plataforma después del paso 4

5. Creación de una tarea: Se debe crear una tarea de entrenamiento como se ve en la Figura 18 indicando los índices a utilizar, la librería que se ha utilizado (Pytorch o Tensorflow) y proporcionar los scripts de Flower (cliente y servidor) en lenguaje python.

6. Validación de una tarea: Un administrador con conocimiento técnico deberá revisar y validar el código antes de añadirlo a la cola de ejecución. Se puede ver un detalle de la interfaz en la Figura 20

7. Lanzamiento de la tarea: El orquestador del nodo central comienza el despliegue del contenedor del servidor de Flower y se comunica con cada uno de los nodos involucrados en la tarea vía HTTPS, enviando el API Key en la cabecera y usando la llamada run para desplegar los contenedores con los clientes de Flower.En la figura 29, se puede ver el momento del lanzamiento de una tarea.

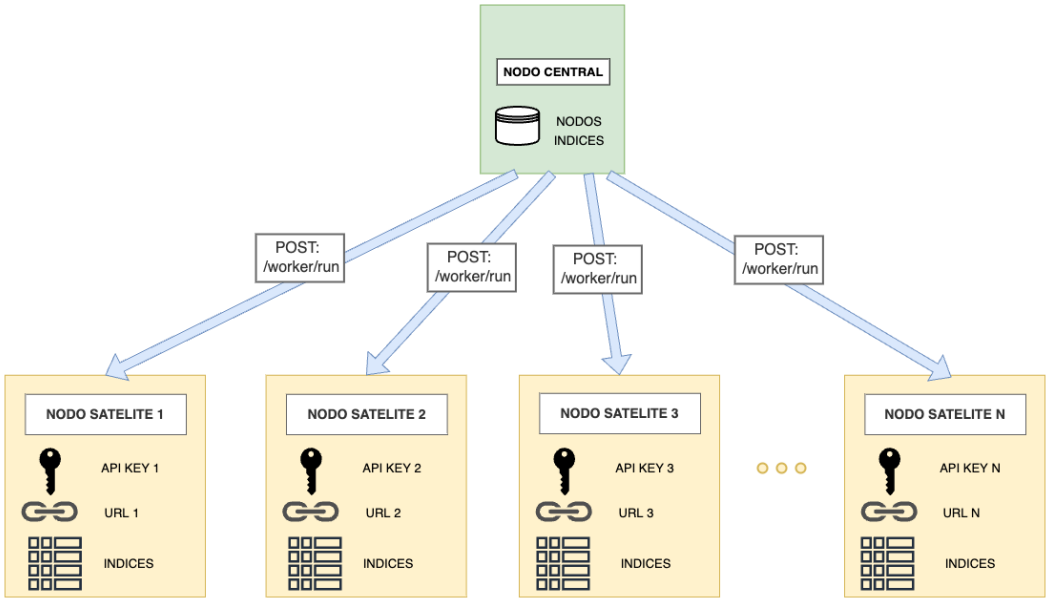

Figura 29: Paso 7 Lanzamiento de tarea

8. Despliegue de contenedores temporales: El nodo central y los nodos satélites despliegan los contenedores servidor y cliente de Flower y ejecutan sus respectivos scripts de python. La librería de Flower se encarga de establecer la comunicación entre los contenedores clientes con el contenedor servidor mediante gRPC.

9. Entrenamiento: Los clientes empiezan a entrenar el algoritmo con sus propios datasets y envían los resultados al servidor que los agrega al modelo y actualiza los modelos de cada cliente.

10. Finalización: Los clientes terminan el entrenamiento y el contenedor se auto destruye. El servidor espera a que terminen todos los clientes y guarda los resultados. Al finalizar la ejecución la arquitectura se queda como en la Figura 30

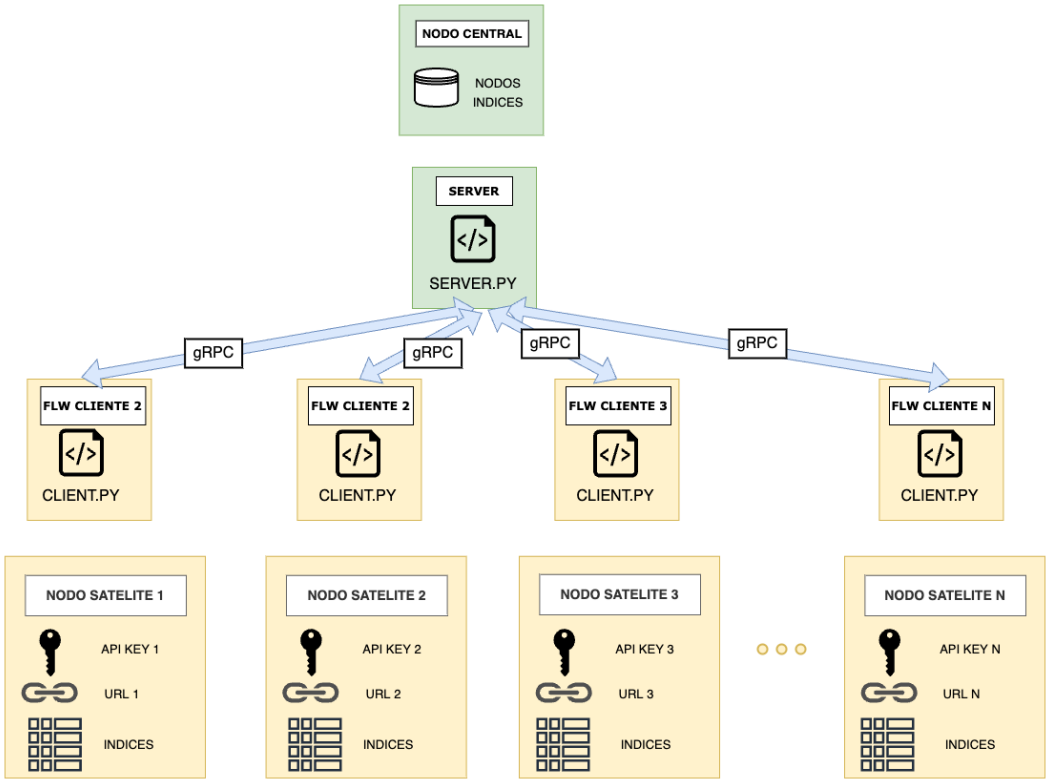

Figura 30: Estado de la arquitectura durante el entrenamiento

Validación

Metodología de validación

Se ha decido validar la plataforma probando el funcionamiento de un algoritmo federado y comparar su rendimiento con los resultados en local. Para ello se ha desplegado una plataforma con un nodo central y tres nodos satélites. Los tres nodos satélites tienen diferentes prestaciones y diferentes datos de un mismo dataset como se puede ver en la siguiente tabla:

Servidor	Memoria Rm	Procesador	Tarjeta Gráfica	Tamaño del dataset
Macbook	16 GB	2.20 GHz intel i7	NO	12441
Cucumber	48 GB	2.80GHz intel i9	NO	49767
D106 Unix Server	32 GB	3.50GHz intel 7	NVIDIA TITAN RTX 24GB	62208

Tabla 3: Tabla de propiedades de los nodos satélite

Dataset Covid19

Para esta validación se ha elegido federar un algoritmo sobre el dataset público de Our world in data (OWID) sobre el Covid 19 (https://github.com/owid/covid-19-data). Este dataset contiene datos de la evolución de la pandemia con datos diarios por país sobre estadísticas sanitarias, datos demográficos, número de infectados, vacunados, etc. y tratar de predecir el numero de nuevos casos. Para ello hemos elegido 7 variables que estaban en presentes en la mayoría de los datos y que pueden ser normalizadas fácilmente, pues simplifica el trabajo de limpieza de datos. Éstas son:

- population: población del país.
- male_smokers: número de fumadores varones del país.
- female_smokers: número de fumadores mujeres del país.

- reproduction_rate: velocidad de reproducción.
- life_expectancy: esperanza de vida.
- cardiovasc_death_rate: tasa de mortalidad cardiovascular.
- diabetes_prevalence: prevalencia de diabetes.

Y se intentará predecir:

- new_cases: nuevos casos de covid

Los datos se han subido a la plataforma utilizando un pipeline de logstash que periódicamente baja los datos de OWID y los actualiza en el índice de logstash. Cabe señalar que antes de procesar los datos se ha realizado una limpieza, quitando aquellas filas que no contengan los 7 variables seleccionadas o la variable a predecir. Además, se ha realizado una normalización de los datos con la media y la desviación típica. Para simular datos distintos en cada nodo, en uno de los nodos se ha utilizado solo 10 % de los datos, en otro 50 % y en el último el 40 %. Finalmente, para que los datos no estén ordenados, puesto que están alfabéticamente, se barajan siempre con la misma clave para que aunque desordenados, los tres nodos tengan el mismo orden y por tanto distintos datos.

Modelo

Para la validación de la plataforma se ha decidido utilizar una red neuronal profunda (NN) sencilla para la predicción, debido a que puede tener como entrada un vector con los datos y se pueden realizar operaciones complejas dentro de las capas para resolver tareas de regresión. Las NN se componen de una capa de entrada, que será del tamaño de los datos de entrada, unas hiden layers, que añaden complejidad y profundidad a la red y, por último, la capa de salida, que deberá ajustarse a las necesidades de cada problema a resolver, dependiendo por ejemplo de si es una regresión o una clasificación.

Para la red de este trabajo la capa de entrada contiene 7 valores correspondientes a los datos del dataset referenciados en el punto anterior. Dentro de la red propiamente dicha, existen tres capas, la primera con 16 neuronas, la segunda con una función de activación ReLu(Rectified Linear Unit) que activa las neuronas solo si estas estan por encima de 0, y la última realiza un Dropout del 30 % de neuronas en cada iteración durante el entrenamiento, con la finalidad de reducir el

sobre entrenamiento u overfitting. Finalmente la capa de salida es de una neurona que ayudara a predecir el numero de nuevos casos. Se puede ver un esquema de la red en la Figura 31

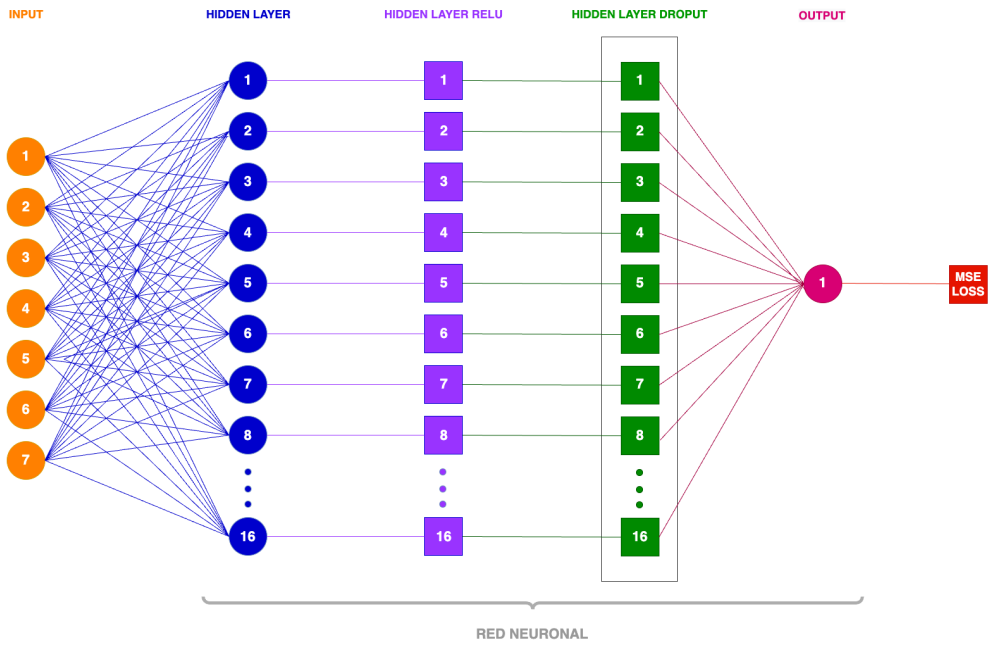

Figura 31: Capas de la red Neuronal

Se ha seleccionado la métrica de error medio absoluto como función de pérdida (MAE) 1y un optimizador Adam con Learning Rate 0.001 [43].

$$MAE = \sum_{i=1}^{D} |x_i - y_i| \tag{1}$$

Para validar el modelo y, con ello, la propia plataforma, se ha ejecutado una regresión lineal (LR) sobre el mismo datatset de manera local en cada nodo, intentando así hacer la misma predicción que con la NN, para así comprobar que, aunque el modelo sea simple, ofrece una mejora sobre la LR.

Experimentos

Para realizarlos experimentos en local con los modelo de LR y de NN se han establecido que debido a la simplicidad de los modelos un total de 20 epochs, es decir, el dataset iterará 20 veces completas durante el entrenamiento, mientras que en federado para el modelo de NN se han establecido 20 rondas y 1 epoch, es decir, que el algoritmo se iterará 20 veces pero después de cada epoch se realizará la estrategia de agregación para actualizar los pesos del modelo. Se ha elegido como estrategia de agregación el Federated Average [44], que consiste en hacer una media de los pesos. Como es un modelo sencillo y la predicción que se requiere no es muy compleja, se ha considerado que 20 iteraciones sobre el dataset es suficiente para probar su efectividad. El dataset de cada nodo se separará en un 80 por ciento para el entrenamiento del modelo y un 20 por ciento para la validación del modelo, siendo la separación por orden de las filas del dataset.

Resultados entrenamiento Local

Regresión Lineal

En las siguientes gráficas de las Figuras 32, 33 y 34 y la Tabla 4 se pueden observar los resultados de la LR en los distintos nodos.

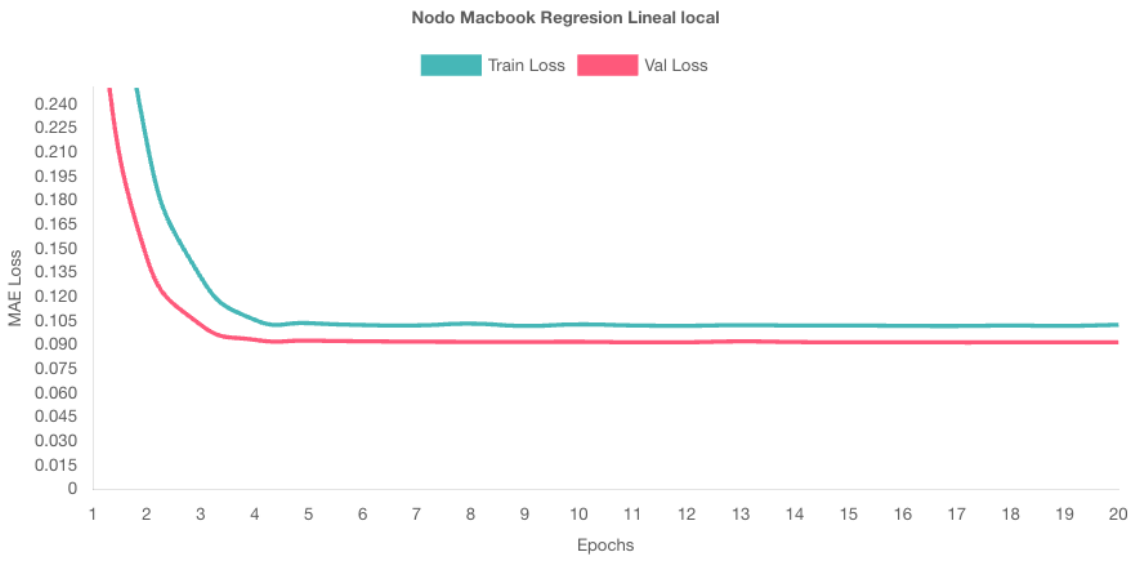

Figura 32: Resultados LR en nodo Macbook

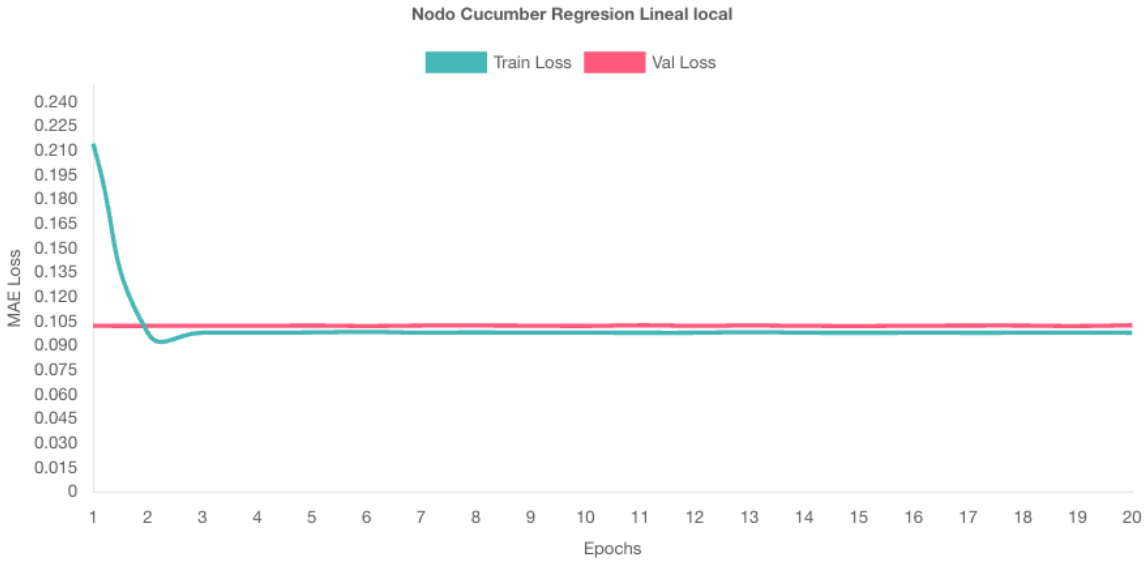

Figura 33: Resultados LR en nodo Cucumber

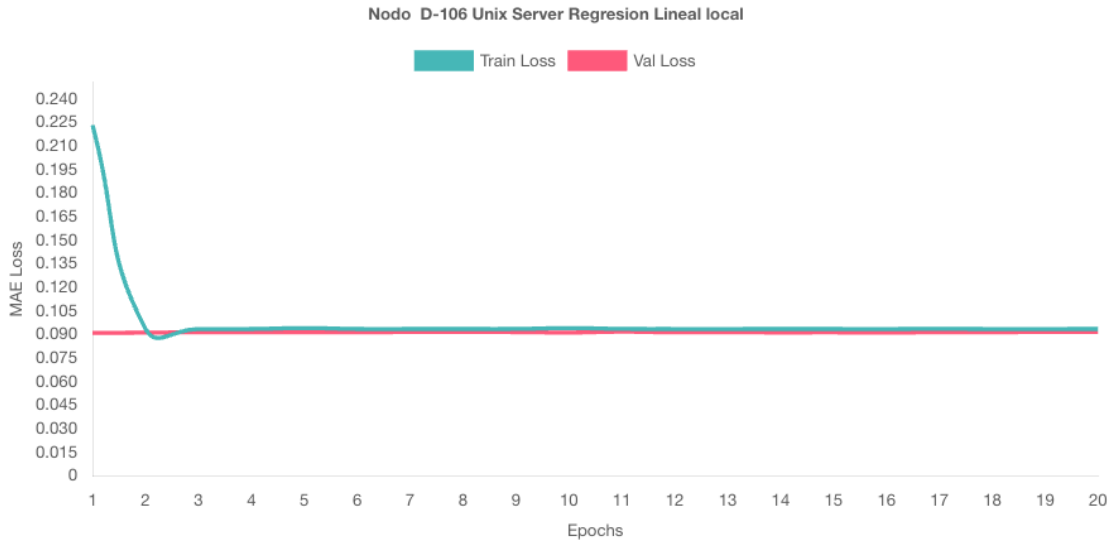

Figura 34: Resultados LR en nodo D106

Servidor	Train Loss	Validation Loss	Tiempo	Uso maximo de ram
Macbook	0.10228102328255773	0.09145525925466846	15s	802908 kb
Cucumber	0.09775427257100996	0.10235898230417212	28s	1451360 kb
D106 Unix Server	0.09332569608115292	0.09140661202117069	52s	1449932

Tabla 4: Resultados y estadísticas del entrenamiento LR

En este experimento se puede observar que obtiene buenos resultado muy rápidamente y luego se estanca. Esto puede deberse a que es un modelo muy simple y un dataset completo y extenso.

Red Neuronal

En las siguientes gráficas de las Figuras 35, 36 y 37 y la Tabla 5 se pueden observar los resultados de la NN.

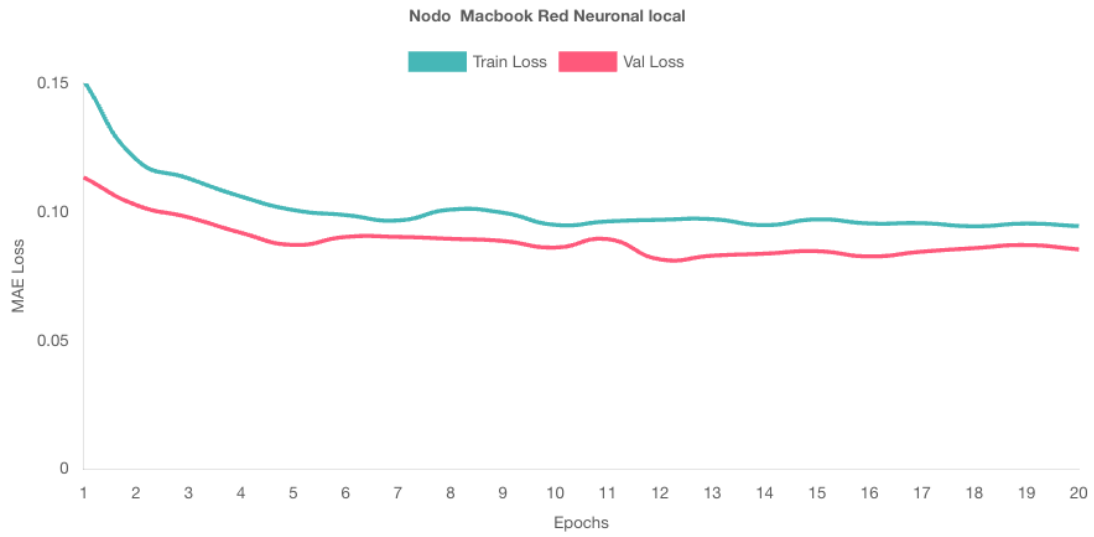

Figura 35: Resultados NN en nodo Macbook

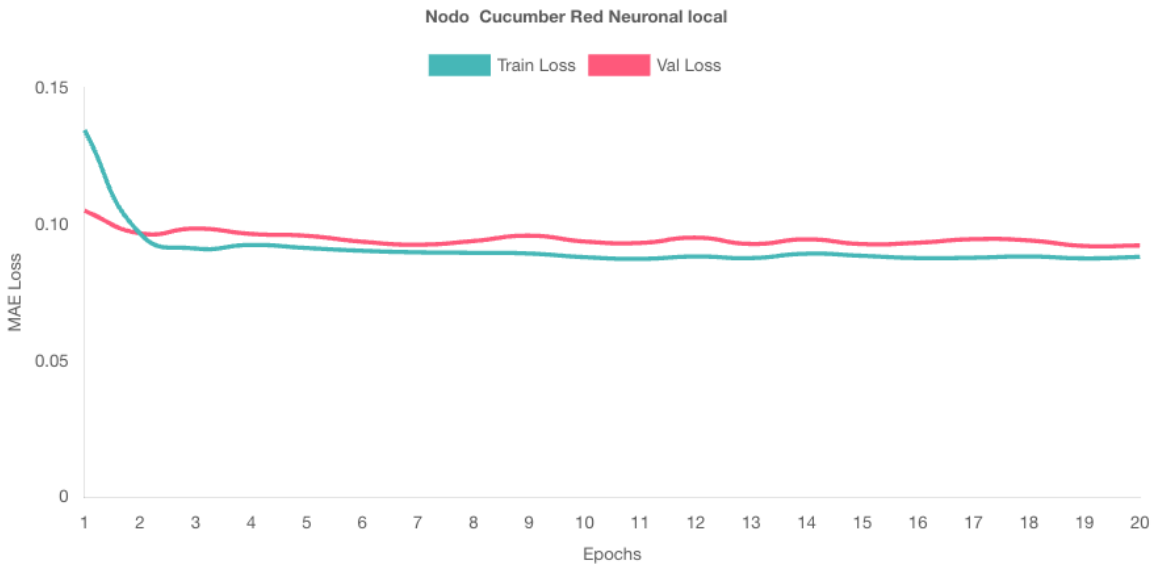

Figura 36: Resultados NN en nodo Cucumber

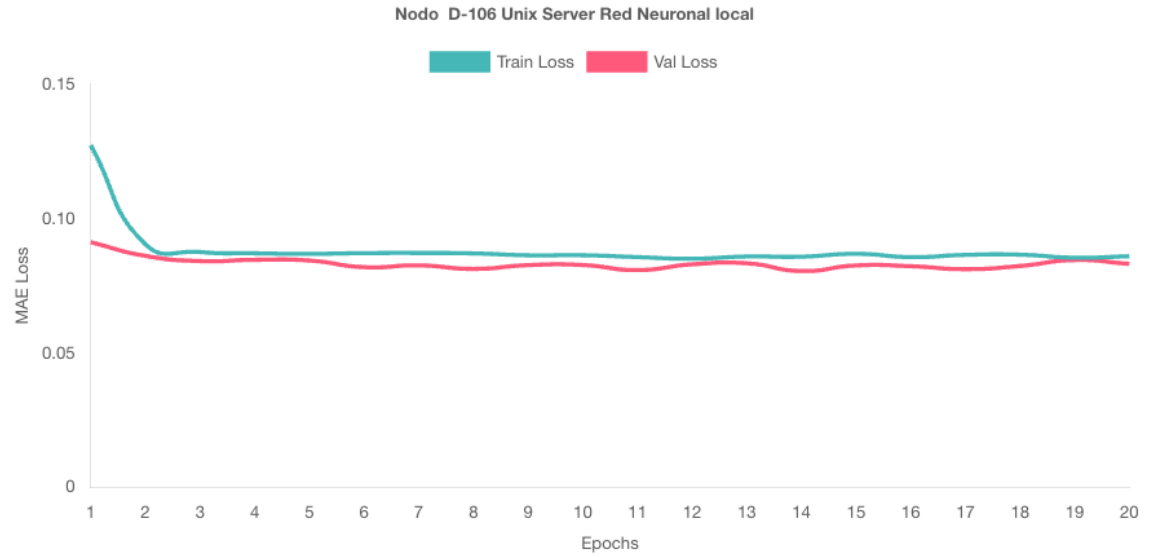

Figura 37: Resultados LR en nodo D106

Servidor	Train Loss	Validation Loss	Tiempo	Uso maximo de ram
Macbook	0.09443732692549626	0.08537145514847908	22s	783752 kb
Cucumber	0.08811016902282691	0.09229286031613022	37s	1450772 kb
D106 Unix Server	0.0860148249720738	0.0831677703407836	70s	1450772 kb

Tabla 5: Resultados y estadísticas del entrenamiento NN

Las gráficas muestran que aprende rápidamente, pero esta vez no se estanca sino que la perdida va bajando. Si continuáramos con más épocas lo mas probable es que la perdida siguiera bajando, aunque sea muy lentamente. Se observa también que el tiempo de entrenamiento se alarga.

Resultados entrenamiento federado

En las siguientes gráficas de las Figuras 38, 39 y 40 y la Tabla 6 se pueden observar los resultados de la NN federado.

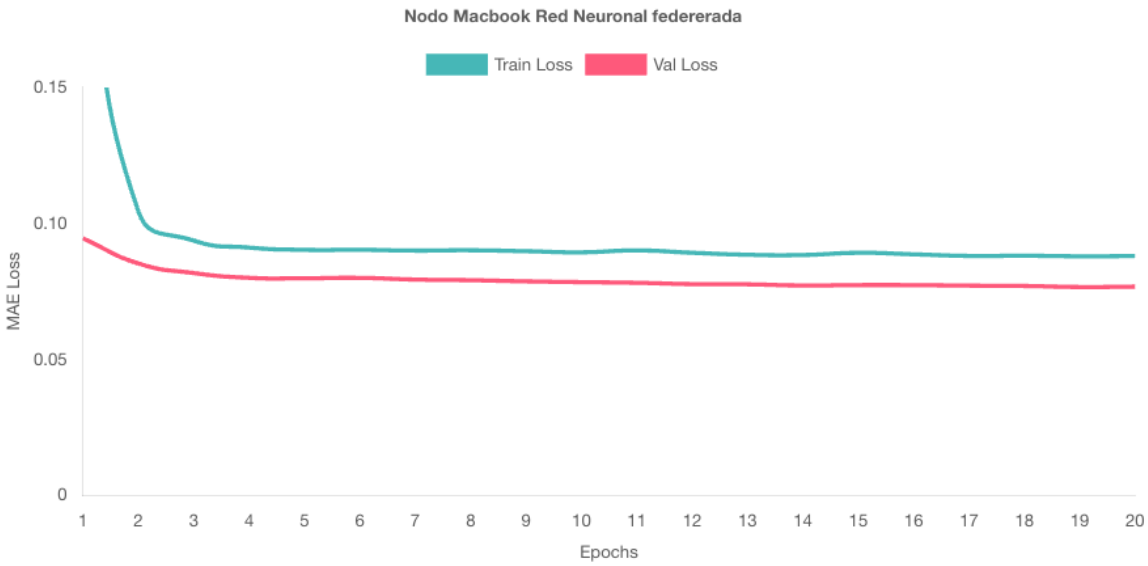

Figura 38: Resultados NN federado en nodo Macbook

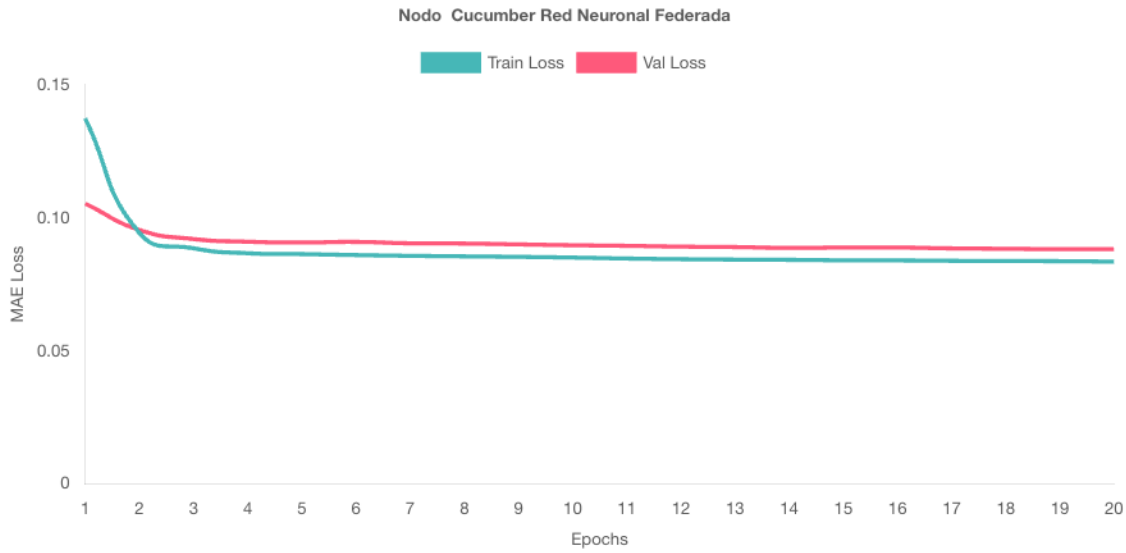

Figura 39: Resultados NN federado en nodo Cucumber

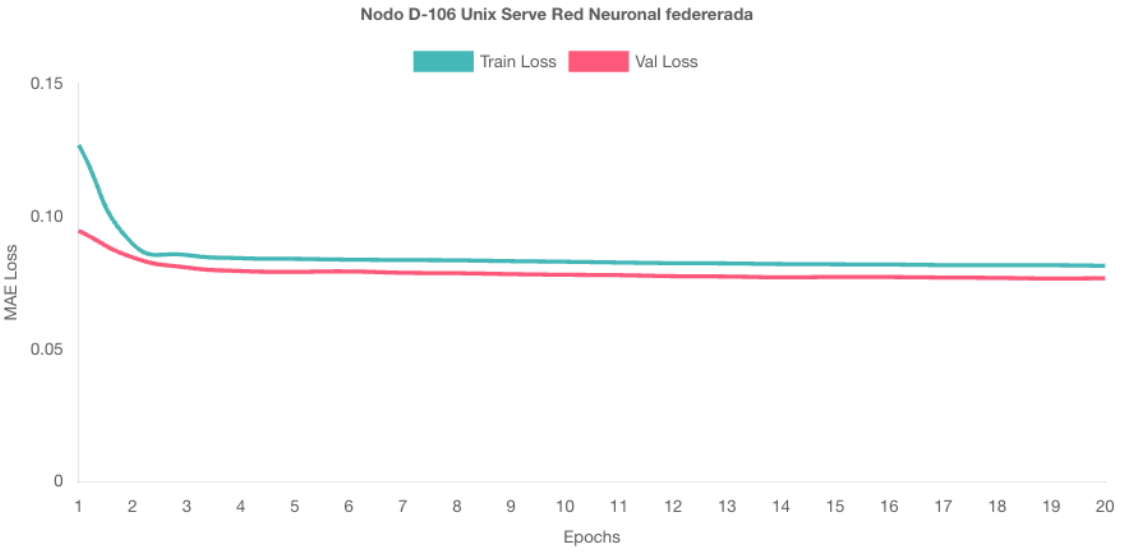

Figura 40: Resultados LR federado en nodo D106

Servidor	Train Loss	Validation Loss	Tiempo	Uso maximo de ram
Macbook	0.0879784690705725	0.07672443793837375	60s	805388 kb
Cucumber	0.08353178828572003	0.0882484419576832	69s	1460764 kb
D106 Unix Server	0.08136714673613062	0.07666280675579741	67s	1459412 kb

Tabla 6: Resultados y estadísticas del entrenamiento NN federado

Las gráficas muestran que se aprende rápidamente y luego la perdida va bajando muy despacio. Se muestra también que el nodo con menos datos el aprendizaje es mejor que en los otros. Como era de esperar, los tiempos de entrenamiento se igualan al del nodo con mayor número de datos.

Comparación de resultados

Comparando los resultados se puede confirmar que la NN mejora la LR a costa de un mayor tiempo de entrenamiento, lo cual es normal pues la red neuronal es mas compleja que la regresión lineal. Esta mejora valida la NN para utilizarla en el experimento. La federación de la NN también mejora los resultados al entrenamiento en local a costa de requerir más tiempo en algunos nodos, ajustándose por arriba al nodo que mayor tiempo necesita, puesto que los nodos que utilizan menos tiempo tienen que esperar a los nodos mas lentos para realizar la agregación de los pesos. Aunque la diferencia de tiempo es mayor que respecto a la solución en local, estos nodos se ven recompensados con unos mejores resultados. También conviene destacar que el uso de memoria es muy parecido para todos los casos, por lo que el uso de la federación no supondría un aumento de uso de recursos de los servidores. Con estos experimentos y sus resultados se valida la funcionalidad de la plataforma, pues esta funcionando para el entrenamiento de algoritmos federados.

Validación de objetivos

El objetivo principal definido en este trabajo era el de crear una plataforma que permitiera utilizar FL de manera sencilla, con el fin de que los usuarios finales trabajaran con los datos clínicos y los algoritmos de ML sin necesidad de preocuparse de la privacidad, de la arquitectura y del despliegue de los servidores. Para comprobar que se han alcanzado, se ha hecho una comparación de los diferentes pasos que debería hacer un científico de datos una vez desarrollado un modelo de FL tanto usando solo Flower, como usando la plataforma de la solución. Esta comparación se puede ver en 7

#	Acción	Flower Framework	Plataforma desarrollada
1	Creación de entorno de trabajo en nodo central	1. Conexión con ssh en nodo central 2. Creación de entorno virtual de python 3. Instalación de dependencias	Transparente para el usuario (Automatizado por la plataforma)
2	Creación de entorno de trabajo en nodos satélite	1. Conexión con ssh en nodo satélite 2. Creación de entorno virtual de python 3. Instalación de dependencias	Transparente para el usuario (Automatizado por la plataforma)
3	Despliegue de nodo central de Flower	1. Copia de server.py a nodo central via scp 2. Ejecución de script de python server.py en nodo central	1. Loguearse en la plataforma. 2. Crear tarea en Interfaz gráfica. (Figura \ref{3.15})
4	Despliegue de nodo central de Flower	1. Copia de client.py py a nodo central via scp 2. Ejecución de script de python client.py en nodo central	
5	Obtención de resultados	1. Copia desde nodo central de archivo de resultados via scp	1.Descarga de resultados desde interfaz gráfica (Figura \ref{3.20})

Tabla 7: Comparativa de pasos para el entrenamiento de un modelo

Como puede verse en esta tabla, el número de pasos que cualquier científico de datos tiene que hacer para poder poner a funcionar sus algoritmos se ve ampliamente reducido. Además, aquellos que tienen que ser realizados son mucho más simples, evitando con ello que el usuario tenga que disponer de conocimientos avanzados en la gestión de este tipo de redes. De esta forma, podemos derivar que la disminución de la complejidad se alcanza de forma cualitativa y cuantitativa.

Además, en relación a esto y tal y como se esperaba, el software se proporciona de forma en la que su instalación se facilita puesto que se reduce a la necesidad de ejecutar un script en bash que se ha llamado "deploy.sh". Este script únicamente recibe un parámetro que puede tomar dos valores, ya sea server o worker dependiendo si se quiere instalar un nodo central o un nodo satélite, y se encarga de desplegar los contenedores necesarios para cada caso, cumpliendo con el objetivo buscado.

Conclusiones

Conclusiones

Tal y como se buscaba y se ha descrito a lo largo de esta memoria, se ha cumplido el objetivo principal de este trabajo, esto es, desarrollar una plataforma que permita crear una red para el entrenamiento federado con datos sin que estos salgan de los servidores en los que están alojados. De una manera muy sencilla, se puede instalar el software sin necesidad de tener ningún conocimiento de contenerización de sistemas, red, etc., dado que la ejecución de un script bash es suficiente para desplegar los nodos. Para el gestión, mantenimiento,agregación de nodos, ejecución de tareas,consulta de resultados, etc. este trabajo propone una interfaz gráfica sencilla, por lo que no se necesita ningún perfil técnico especifico para gestionarla y los científicos de datos no se tienen que preocupar más que en crear sus algoritmos y federarlos. Se puede decir que esto es una mejora en cuanto a las plataformas actuales, pues se aísla el entorno de entrenamiento del modelado de algoritmos. Se ha demostrado en este proyecto que la plataforma funciona y que ha utilizado FL para mejorar los resultados, sin que los datos hayan salido de sus respectivos nodos. La arquitectura de esta plataforma puede dar pie a un nuevo modo de compartir datos sensibles par un uso secundario de los datos clínicos, pudiendo ser una semilla para un estándar que simplifique los trámites burocráticos y legales. Abriendo esta puerta a la investigación se puede alcanzar una mayor interoperabilidad, mejorando la rapidez para mejorar tratamientos o para tomas de decisiones, vital en situaciones como la vivida con la pandemia del Covid 19. En el plano económico, la plataforma puede dar una oportunidad también de ahorrar costes mediante un crecimiento en horizontal, que suele ser menos costoso, con los datos repartidos en muchos nodos con equipos de prestaciones modestas, en vez de un crecimiento vertical con menos equipos pero con hardware más caro.

Trabajo Futuro

Aun cuando el presente trabajo ofrece una solución completa y funcional, varias son las líneas de investigación que se pueden seguir para aumentar su impacto:

- En el caso de uso para validar la plataforma, los resultados han sido buenos, pero cabe la duda de que hubiera pasado si los datos no hubieran sido el resultado de dividir un solo dataset, si no que los datos procedieran distintos datasets en distintos nodos heterogéneos. Habría que investigar como se comportaría en estos casos y ver la posibilidad de proveer en la plataforma un a estrategia de agregación que tenga en cuenta esta anomalías a la hora de calcular los pesos.
- En el caso de FL el entrenamiento de un modelo puede tener fugas de privacidad al compartir con el nodo central los pesos del modelo. Se podría investigar el añadir la privacidad diferencial (DP) a la plataforma. DP es un método utilizado para limitar y cuantificar la fuga de privacidad de los datos sensibles al realizar tareas de aprendizaje.
- Otra de las lineas de investigación podría ser la de incorporar a la plataforma un modo sin servidor central en el que la comunicación sea peer to peer salvando así los problemas derivados de la comunicación con este.
- La plataforma esta en un estado muy precoz, llegando a ser una primera versión alpha, por lo que esta abierta a crecer con pequeñas funcionalidades, como ampliar el numero de imágenes del contenedor creado para el entrenamiento como pueden ser sci-kit o sklearn y no centrarse únicamente en TensorFlow y PyTorch. Otra funcionalidad podría ser la de poder ver las gráficas de los resultados en vez de descargarse el archivo con los resultados.

Planificación temporal

Se ha realizado un Diagrama de Gantt para la planificación temporal desglosando las distintas tareas y los plazos. Se ha usado la herramienta Project de Microsoft Office. El proyecto tiene una duración de 152 días con una dedicación de 2 horas al día, con un total aproximado de 300 horas.

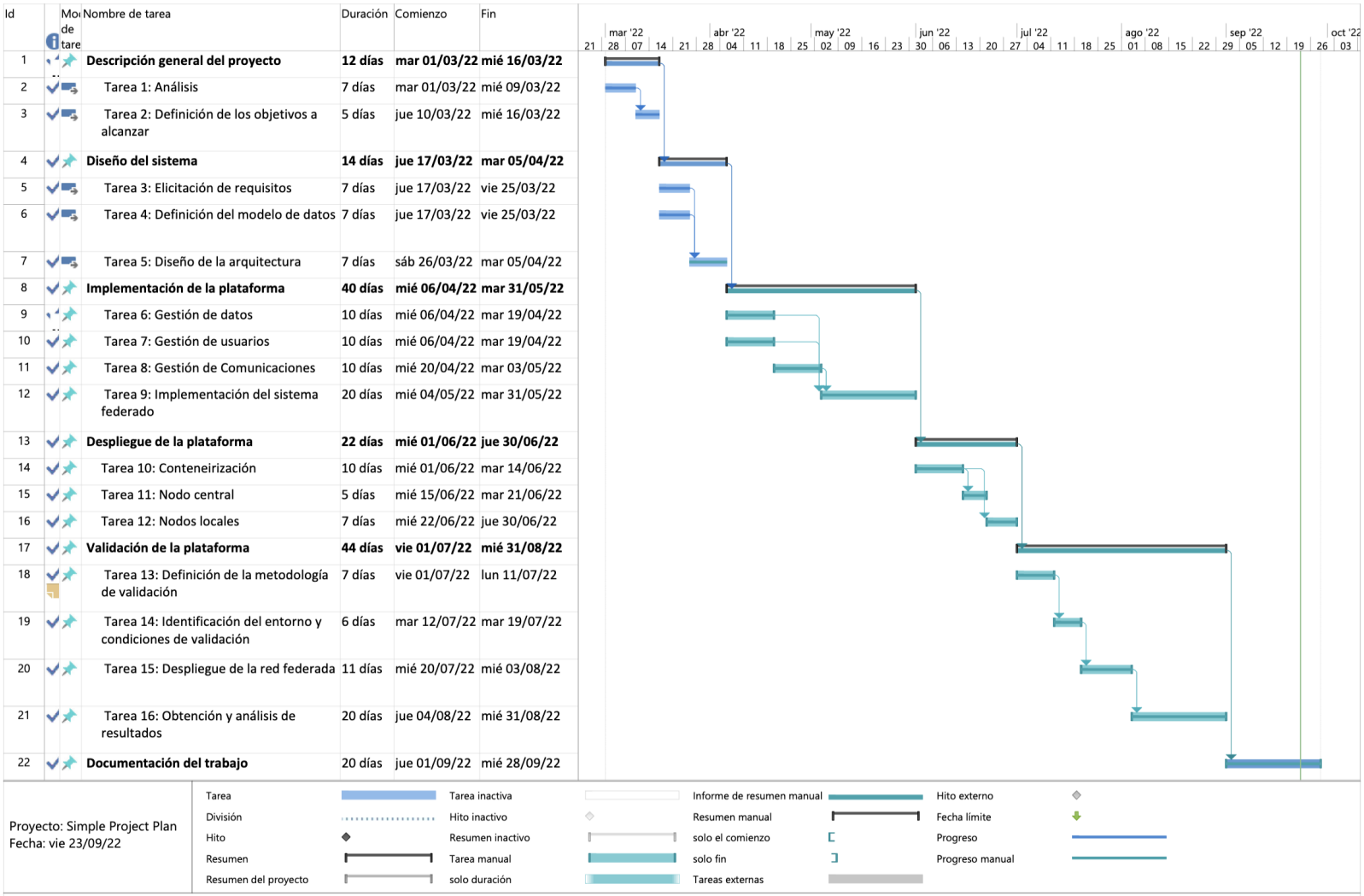

Figura 41: Diagrama de Gantt

Costes

COSTE DE MANO DE OBRA (coste directo)	**Horas**	**Precio/hora**	**Total**
	300	14.375€	4312.5€

COSTE DE RECURSOS MATERIALES (coste directo)	**Precio de Compra**	**Uso en meses**	**Amortización (en años)**	**Total**
Portatil Macbook Pro 2022 32GB RAM desarollo y Servidor central	3849€	6	5	384.9€
Servidor 48 GB intel 9 2.80Ghz con GPU 24GB para nodo satelite	3000€	4	5	250€
Servidor 32 GB intel 7 3,50Ghz con GPU 24GB para nodo satelite	4500€	4	5	375€
Portatil Macbook Pro Mid 2015 16GB RAM para nodo satelite	2200€	4	7	104.76€
Software Open source	-	-	-	0€

COSTE TOTAL DE RECURSOS MATERIALES	1114.66€

GASTOS GENERALES (costes indirectos)	15 %	Sobre CD		814.07€
BENEFICIO INDUSTRIAL	8 %	Sobre CD+CI		499.29€
SUBTOTAL PRESUPUESTO				6740.52€
IVA APLICABLE			21 %	1415.5€

TOTAL PRESUPUESTO	8156.03€

Bibliografía

[1] Christopher V Cosgriff, Daniel K Ebner y Leo Anthony Celi. "Data sharing in the era of COVID-19". En: *The Lancet Digital Health* 2.5 (2020), e224. ISSN: 2589-7500. DOI: https://doi.org/10.1016/S2589-7500(20)30082-0. URL: https://www.sciencedirect.com/science/article/pii/S2589750020300820.

[2] Elizabeth Gourd. "GDPR obstructs cancer research data sharing". En: *The Lancet Oncology* 22.5 (2021), pág. 592. ISSN: 1470-2045. DOI: https://doi.org/10.1016/S1470-2045(21)00207-2. URL: https://www.sciencedirect.com/science/article/pii/S1470204521002072.

[3] Russell L. Ackoff. "From data to wisdom.." en. En: *Journal of applied systems analysis* 16.1 (1989), págs. 3-9.

[4] *Data volume of internet of things (IoT) connections worldwide in 2019 and 2025*. 2022. URL: https://www.statista.com/statistics/1017863/worldwide-iot-connected-devices-data-size/ (visitado 25-09-2022).

[5] *Los tres candados que una empresa debe poner sobre los datos especialmente protegidos*. 2022. URL: https://www.sage.com/es-es/blog/los-tres-candados-que-una-empresa-debe-poner-sobre-los-datos-especialmente-protegidos/ (visitado 25-09-2022).

[6] Thomson Civitas. "Guía de protección de datos personales para Servicios Sanitarios Públicos." es. En: *gencia de Protección de datos de la Comunidad de Madrid (APDCM)* (2004), pág. 69.

[7] *General Data Protection Regulation*. 2022. URL: https://gdpr-info.eu/ (visitado 25-09-2022).

[8] Van Panhuis Willem G. et al. "A systematic review of barriers to data sharing in public health." en. En: *BMC public health* 14 (2014), pág. 33.

[9] Jie et al. "Federated learning for healthcare informatics." en. En: *Journal of Healthcare Informatics Research* 5.1 (2021), págs. 1-19.

[10] Sun J. et al. "Vertical Federated Learning without Revealing Intersection Membership." en. En: (2021).

[11] Tian Li et al. "Federated Learning: Challenges, Methods, and Future Directions." en. En: *IEEE Signal Processing Magazine* 37(3) (2020), págs. 50-60.

[12] *HL7 Healthcare Privacy and Security Classification System (HCS)*. 2014. URL: https://www.hl7.org/implement/standards/product_brief.cfm?product_id=345 (visitado 22-09-2022).

[13] *HL7 Version 2*. 2014. URL: https://www.hl7.org/implement/standards/product_brief.cfm?product_id=185 (visitado 22-09-2022).

[14] Lorena González-Castro et al. "CASIDE: A data model for interoperable cancer survivorship information based on FHIR". En: *Journal of Biomedical Informatics* 124 (2021), pág. 103953. ISSN: 1532-0464. DOI: https://doi.org/10.1016/j.jbi.2021.103953. URL: https://www.sciencedirect.com/science/article/pii/S1532046421002823.

[15] *HL7 Version 2*. 1989. URL: https://www.hl7.org/implement/standards/product_brief.cfm?product_id=185 (visitado 22-09-2022).

[16] *What is an HL7 Interface and Why is It Still the Standard in Healthcare Interoperability?* 2022. URL: https://kms-healthcare.com/what-is-an-hl7-interface-and-why-is-it-still-the-standard-in-healthcare-interoperability/ (visitado 22-09-2022).

[17] Friedman Charles et al. "Achieving a nationwide learning health system." en. En: *Science translational medicine* 2.57 (2010).

[18] Crowson Matthew et al. "A systematic review of federated learning applications for biomedical data." en. En: *PLOS Digital Health* 15 (2022), págs. 700-708.

[19] Antunes Rodolfo et al. "Federated Learning for Healthcare: Systematic Review and Architecture Proposal." en. En: *ACM Transactions on Intelligent Systems and Technology (TIST)* 13.4 (2022), págs. 1-23.

[20] Roy A. et al. "a peerto-peer environment for decentralized federated learning". en. En: *arXiv* (2019).

[21] Li W. et al. "Privacy-preserving federated brain tumour segmentation". En: *International Workshop on Machine Learning in Medical Imaging*. 2019.

[22] Sheller M. J. et al. "Multi-institutional deep learning modeling without sharing patient data: a feasibility study on brain tumor segmentation". en. En: *In International MICCAI Brainlesion Workshop*. 2018.

[23] Flores Mona et al. "Federated Learning used for predicting outcomes in SARS-COV-2 patients." en. En: *Research Square* ().

[24] Balachandar Niranjan et al. "Accounting for data variability in multi-institutional distributed deep learning for medical imaging." en. En: *Journal of the American Medical Informatics Association* 27.5 (2020), págs. 700-708.

[25] *Joint Imaging Platform (Jip)*. 2020. URL: https://jip.dktk.dkfz.de/jiphomepage/ (visitado 19-09-2022).

[26] Data Science Institute. *2020 annual report*. Data Science Institute, 2021.

[27] *Labelia Foundation*. 2019. URL: https://www.labelia.org/en/healthchain-project (visitado 19-09-2022).

[28] Santiago Silva et al. "Fed-BioMed: A General Open-Source Frontend Framework for Federated Learning in Healthcare". En: *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning: Second MICCAI Workshop, DART 2020, and First MICCAI Workshop, DCL 2020, Held in Conjunction with MICCAI 2020, Lima, Peru, October 4–8, 2020, Proceedings*. Lima, Peru: Springer-Verlag, 2020, págs. 201-210. ISBN: 978-3-030-60547-6. DOI: 10.1007/978-3-030-60548-3_20. URL: https://doi.org/10.1007/978-3-030-60548-3_20.

[29] *GAIA-X*. 2022. URL: https://gaia-x.eu/what-is-gaia-x/core-elements/data-spaces/ (visitado 19-09-2022).

[30] *Yee, Sandra*. 2022. URL: https://www.newswire.com/news/rhino-health-platform-powers-hospital-based-federated-learning (visitado 19-09-2022).

[31] Rath Ana et al. "Representation of rare diseases in health information systems: the Orphanet approach to serve a wide range of end users." en. En: *Human mutation* 33.5 (2022), págs. 803-808.

[32] Kairouz Peter et al. "Advances and open problems in federated learning." en. En: *arXiv preprint arXiv* (2019).

[33] *FATE Framework*. 2022. URL: https://fate.readthedocs.io/en/latest/architecture/ (visitado 25-09-2022).

[34] *PaddleFL Framework*. 2022. URL: https://paddlefl.readthedocs.io/ (visitado 25-09-2022).

[35] *FLower Framework*. 2022. URL: https://flower.dev/docs/architecture.html (visitado 19-09-2022).

[36] *FedBiomed Framework*. 2022. URL: https://fedbiomed.gitlabpages.inria.fr/ (visitado 25-09-2022).

[37] *TensorFlow Federated*. 2022. URL: https://docs.nvidia.com/clara/clara-train-sdk/ (visitado 25-09-2022).

[38] *PyGrid*. 2022. URL: https://github.com/OpenMined/PyGrid (visitado 25-09-2022).

[39] *PySyft*. 2022. URL: https://github.com/OpenMined/PySif (visitado 25-09-2022).

[40] *FedBiomed License*. 2021. URL: https://gitlab.inria.fr/fedbiomed/fedbiomed/-/blob/develop/LICENSE.md (visitado 25-09-2022).

[41] *Jinja Framework*. 2022. URL: https://jinja.palletsprojects.com/en/3.1.x/ (visitado 25-09-2022).

[42] *ELK Stack*. 2022. URL: https://www.elastic.co/es/what-is/elk-stack (visitado 25-09-2022).

[43] Diederik P. Kingma y Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. DOI: 10.48550/ARXIV.1412.6980. URL: https://arxiv.org/abs/1412.6980.

[44] Tao Sun, Dongsheng Li y Bao Wang. "Decentralized Federated Averaging". En: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022), págs. 1-12. DOI: 10.1109/TPAMI.2022.3196503.