UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC
DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Sistemas Informáticos y Computación

Herramientas para la Composición de Flujos de Inferencia para Modelos de Inteligencia Artificial en la Nube

Trabajo Fin de Máster

Máster Universitario en Computación en la Nube y de Altas Prestaciones / Cloud and High-Performance Computing

AUTOR/A: Aguirre Ramírez, Diego Alejandro

Tutor/a: Moltó Martínez, Germán

Director/a Experimental: Calatrava Arroyo, Amanda

CURSO ACADÉMICO: 2023/2024

Departament de Sistemes Informàtics I Computació

Universitat Politècnica de València

Herramientas para la Composición de Flujos de Inferencia para Modelos de Inteligencia Artificial en la Nube

Trabajo Fin de Máster

Máster Universitario en Computación en la Nube y Altas Prestaciones / Cloud and High Performance Computing

Autor: Aguirre Ramírez, Diego Alejandro

Tutor: Moltó Martínez, Germán

Directora Experimental: Calatrava Arroyo, Amanda

2024

Resumen

Este trabajo de fin de máster explora la integración de herramientas generalmente utilizadas para la composición de workflows y pipeline, específicamente Elyra y Node-RED (este último a través de FlowFuse), para facilitar la implementación y gestión de de la ejecución de modelos de inteligencia artificial en entornos en la nube. La problemática central radica en la necesidad de simplificar los procesos de despliegue de estos modelos, que suelen ser complejos y requieren de conocimientos especializados en varias áreas de la informática. En este trabajo se propone una solución que combina las capacidades de Elyra, una extensión de JupyterLab para el manejo de pipelines de datos y *machine learning*, con la interfaz gráfica y facilidad de uso de Node-RED, permitiendo así una integración efectiva y accesible para usuarios de distintos niveles técnicos.

El trabajo detalla el proceso de configuración y uso de estas herramientas, demostrando cómo pueden simplificar la creación, gestión y despliegue de modelos de inferencia. Se examinan las ventajas de esta integración, como la reducción de tiempos de desarrollo, la automatización de tareas repetitivas y la posibilidad de realizar ajustes y mejoras de manera más ágil. Además, se aborda el impacto de estas tecnologías en la democratización del acceso a la inteligencia artificial (IA), permitiendo que usuarios sin profundos conocimientos en programación o data science puedan implementar y beneficiarse de modelos avanzados de IA.

A través de ejemplos prácticos y casos de uso, se evidencia la efectividad de Elyra y Node-RED en el ciclo de vida de los modelos de inferencia, desde su desarrollo hasta su despliegue y mantenimiento en producción. Finalmente, se discuten los retos encontrados durante la implementación de esta solución y se esbozan posibles líneas de investigación futura, tales como la integración de otras herramientas de desarrollo, la optimización de la eficiencia en la ejecución de los modelos y el fortalecimiento de la seguridad en los despliegues. Para la ejecución de los modelos de IA se utilizará la plataforma serverless OSCAR, basada en Kubernetes.

Palabras clave: Elyra; Node-RED; Inteligencia Artificial; Cloud Computing; Serverless.

Abstract

This master's thesis explores the integration of tools generally used for the composition of workflows and pipelines, specifically Elyra and Node-RED (the latter through FlowFuse), to facilitate the implementation and management of the execution of artificial intelligence models in cloud environments. The central issue lies in the need to simplify the deployment processes of these models, which are often complex and require specialised knowledge in various areas of computing. This thesis proposes a solution that combines the capabilities of Elyra, an extension of JupyterLab for handling data and machine learning pipelines, with the graphical interface and ease of use of Node-RED, thus enabling effective and accessible integration for users of varying technical levels.

The master's thesis details the process of configuring and using these tools, demonstrating how they can simplify the creation, management, and deployment of inference models. The advantages of this integration are examined, such as reduced development times, the automation of repetitive tasks, and the ability to make adjustments and improvements more agilely. Additionally, the impact of these technologies on democratising access to artificial intelligence (AI) is addressed, allowing users without deep knowledge in programming or data science to implement and benefit from advanced AI models.

Through practical examples and use cases, the effectiveness of Elyra and Node-RED in the inference model lifecycle is evidenced, from development to deployment and maintenance in production. Finally, the challenges encountered during the implementation of this solution are discussed, and possible lines of future research are outlined, such as the integration of other development tools, the optimisation of execution efficiency of the models, and the strengthening of security in deployments. The execution of AI models will utilise the serverless OSCAR platform, based on Kubernetes.

Keywords : Elyra; Node-RED; Artificial Intelligence; Cloud Computing; Serverless.

Tabla de contenidos

Tabla de figuras

1. Introducción

En la era actual de la tecnología, los modelos de inteligencia artificial (IA) han emergido como herramientas cruciales para impulsar innovaciones en diversas industrias. Sin embargo, implementar y gestionar estos modelos en la nube presenta desafíos significativos debido a la complejidad técnica en la configuración y operación de los flujos de inferencia necesarios para la toma de decisiones. Esta complejidad puede representar un obstáculo considerable, especialmente para quienes no poseen una formación técnica avanzada o los recursos necesarios para manejar infraestructuras complejas.

La ejecución de modelos de IA en la nube, que implica la aplicación de algoritmos pre entrenados sobre nuevos datos para obtener predicciones, demanda una infraestructura robusta y dinámica. Esta infraestructura debe ser capaz de ajustarse a las variaciones en la demanda de procesamiento y a los volúmenes de datos fluctuantes, asegurando la eficiencia y seguridad del proceso.

Ante estos desafíos, herramientas como Elyra y Node-RED ofrecen enfoques complementarios para facilitar la gestión de flujos de inferencia. Elyra amplía las capacidades de JupyterLab, proporcionando un entorno familiar para científicos de datos, lo que les permite utilizar librerías de Python y gestionar pipelines de machine learning a través de una interfaz basada en notebooks. Un notebook es un documento interactivo que combina código, texto explicativo y visualizaciones en un mismo entorno, lo que facilita el desarrollo y la documentación de experimentos de forma integrada.

Por otro lado, Node-RED destaca por su enfoque visual hacia la programación, permitiendo a los usuarios diseñar flujos de trabajo mediante una interfaz basada en nodos que simplifica la integración y automatización de tareas sin necesidad de codificación extensiva.

La plataforma OSCAR complementa estas herramientas, ofreciendo un entorno serverless, es decir, que se encarga de la gestión automatizada de los recursos de cómputo subyacentes, para la ejecución eficiente de aplicaciones de procesado de datos. Esto permite a los usuarios concentrarse exclusivamente en la entrada de datos y la manipulación de los resultados, sin preocuparse por la escalabilidad o el mantenimiento del sistema. Asimismo, OSCAR facilita operaciones avanzadas como el escalado automático de recursos, la gestión eficiente de cargas de trabajo y la integración segura de modelos de IA en aplicaciones más amplias.

Este trabajo se propone explorar cómo la combinación de Elyra, Node-RED y OSCAR puede simplificar significativamente el despliegue y la operación de modelos de IA en la nube. Se buscará demostrar cómo estas herramientas permiten a usuarios de variados niveles técnicos aprovechar las ventajas de la IA de manera efectiva y segura. Mediante una metodología de desarrollo híbrido

Agile-Iterativo, se analizarán las capacidades y beneficios de estas herramientas en diversos escenarios de uso, estableciendo un marco para su integración en procesos empresariales y operativos críticos.

Este trabajo se enmarca en el proyecto AI4EOSC, cuyo objetivo es habilitar una infraestructura europea de computación en la nube avanzada, que integre servicios de inteligencia artificial (IA) para su uso en ciencia abierta. AI4EOSC facilita el acceso y uso de herramientas de IA en la nube para científicos, investigadores y usuarios en diversas áreas. Este proyecto contribuye específicamente al componente AI4Compose[1] dentro de la arquitectura de AI4EOSC, utilizada para la creación y despliegue de flujos de trabajo de IA y pipelines. Para más información, puede consultarse la página web oficial del proyecto: AI4EOSC[2].

1.1 Motivación

El desarrollo acelerado de la tecnología y la ciencia de datos ha llevado a la creación de modelos de IA cada vez más complejos y potentes. Sin embargo, el proceso de desplegar estos modelos en entornos en la nube frecuentemente se encuentra con obstáculos significativos, particularmente en la gestión eficiente de flujos de inferencia. Estos flujos, que son críticos para la aplicación práctica de modelos de IA, implican no solo la ejecución del modelo en sí, sino también la preparación y el manejo de los datos de entrada y salida.

La principal motivación detrás de este Trabajo de Fin de Máster es simplificar a nivel programático la inferencia que se realiza en la nube, permitiendo que los científicos de datos y desarrolladores se enfoquen más en los datos de entrada y salida, y menos en los procesos intermedios. Estos últimos, aunque necesarios, pueden volverse engorrosos y no siempre aportan valor directo al objetivo principal de los estudios de IA, que es obtener resultados precisos y útiles de manera eficiente.

Este trabajo también se enmarca en el contexto del proyecto europeo AI4EOSC, cuyo objetivo es habilitar una infraestructura europea de computación en la nube que permita la integración y democratización de servicios de inteligencia artificial. Este TFM busca contribuir al proyecto AI4EOSC proporcionando herramientas que optimicen la implementación de flujos de inferencia en la nube mediante la integración de Elyra y Node-RED. Estas herramientas facilitarán a los usuarios con distintos niveles técnicos el despliegue de modelos de IA de manera eficiente y segura, aportando valor al componente AI4Compose de la arquitectura AI4EOSC, que se enfoca en la automatización y orquestación de flujos de trabajo en IA. Al simplificar los procesos técnicos y hacer más accesibles

[1] Repositorio de GitHub AI4COmpose: https://github.com/ai4os/ai4-compose
[2] Página web AI4EOSC: https://ai4eosc.eu

estas herramientas, se promoverá una mayor adopción de la IA en diversas áreas, desde la investigación científica hasta aplicaciones industriales. Más información sobre el proyecto AI4EOSC.

Este enfoque no solo busca mejorar la eficiencia y accesibilidad de los flujos de inferencia en la nube, sino que también apunta a democratizar el uso de la inteligencia artificial. Al simplificar los procesos técnicos, más científicos y analistas, independientemente de su experiencia en programación, pueden implementar y beneficiarse de modelos avanzados de IA. Así, este trabajo se centra en encontrar una solución que no solo sea efectiva sino que también sea accesible para un público más amplio, potenciando la adopción y el impacto de la IA en diversas áreas del conocimiento y la industria.

1.2 Objetivos

Con la motivación bien definida, vamos a detallar los objetivos que nos hemos propuesto para este proyecto:

1.2.1. Profundizar en el Funcionamiento de las Herramientas:

- **Elyra**: Vamos a explorar cómo Elyra amplía las funcionalidades de JupyterLab, facilitando la gestión de pipelines de datos y aprendizaje automático. Nos centraremos en entender el uso de Jupyter Notebook dentro de Elyra para descubrir cómo ayuda a simplificar los flujos de trabajo complejos.
- **Node-RED**: Analizaremos cómo Node-RED permite construir flujos de trabajo visualmente y cómo la integración con FlowFuse potencia la colaboración en equipo.

1.2.2. Desarrollo de Nodos Específicos:

- **Elyra**: Diseñaremos nodos que aprovechen librerías clientes en Python para interactuar con la plataforma OSCAR, integrando así nuestros flujos de trabajo con la nube de manera más efectiva.
- **Node-RED**: Crearemos nodos para gestionar peticiones HTTP, permitiendo la interacción con servicios web y facilitando la comunicación con plataformas como OSCAR.

1.2.3. Evaluación Comparativa de las Herramientas:

- Realizaremos pruebas prácticas para comparar cómo Elyra y Node-RED resuelven un mismo pipeline de inferencia de IA, evaluando su eficacia, facilidad de uso y rendimiento en situaciones reales.

- Estudiaremos qué herramienta resulta más adecuada según las necesidades del usuario, enfocándonos en la simplicidad para insertar datos y analizar resultados.

1.2.4. Foco en Usabilidad y Accesibilidad:

- Exploraremos cómo estas herramientas permiten a los usuarios concentrarse en los datos de entrada y salida sin tener que manejar los complejos procesos intermedios, lo cual es clave para hacer la tecnología de IA accesible a un público más amplio.

1.2.5. Contribución a AI4Compose

Otro de los objetivos clave de este proyecto es dejar preparadas herramientas y flujos de trabajo que puedan integrarse en el componente AI4Compose de AI4EOSC. AI4Compose tiene como objetivo simplificar el uso de la inteligencia artificial para un público diverso, permitiendo a usuarios con distintos niveles de conocimientos técnicos, incluidos aquellos sin experiencia en programación, utilizar y beneficiarse de modelos de inferencia de IA. A través de la implementación de herramientas como Elyra y Node-RED, se facilitará el acceso a modelos complejos de IA, permitiendo que científicos, analistas y otros profesionales puedan configurar y ejecutar inferencias en la nube sin la necesidad de involucrarse en los procesos técnicos detallados. Así, se garantiza una accesibilidad más amplia y efectiva de la tecnología de IA, democratizando su uso y potenciando la adopción de modelos de inferencia en distintas áreas de investigación y aplicaciones industriales.

1.3 Estructura del documento

Este Trabajo de Fin de Máster está organizado en los siguientes capítulos, diseñados para guiar al lector a través del estudio realizado sobre la composición de flujos de inferencia para modelos de inteligencia artificial en la nube, utilizando herramientas como Elyra, Node-RED, y FlowFuse en un entorno serverless proporcionado por OSCAR:

- **Capítulo 1: Introducción**
  - Este capítulo introduce el contexto general del estudio, incluyendo la motivación detrás del trabajo, los objetivos principales que se buscan alcanzar, y la estructura del documento para orientar al lector a lo largo del contenido.
- **Capítulo 2: Estado del Arte**
  - Se presenta un análisis detallado del estado actual de la tecnología en el ámbito de la inferencia en inteligencia artificial, destacando las herramientas y modelos relevantes. Se abordan conceptos clave como el modelo serverless, pipelines de inferencia y su aplicación

en la nube. Además se detallan descripciones sobre las herramientas que se utilizarán en este TFM, siendo estas OSCAR, Elyra, Node-RED y FlowFuse.

- **Capítulo 3: Metodología del Desarrollo**
  - Este capítulo describe la metodología utilizada para llevar a cabo el proyecto, detallando el enfoque híbrido Agile-Iterativo adoptado para el desarrollo, y cómo se integraron las herramientas y tecnologías seleccionadas, como OSCAR, Elyra, Node-RED, y FlowFuse.
- **Capítulo 4: Desarrollo de AI4Compose**
  - En este capítulo se describirán los pasos para la instalación y configuración de las herramientas empleadas en este Trabajo Fin de Máster, así como los entornos necesarios para su correcta implementación. Se detallará cómo preparar cada herramienta, asegurando que el entorno de desarrollo esté completamente preparado para permitir trabajar con Elyra y Node-RED.
- **Capítulo 5: Resultados**
  - En este capítulo se presentan los resultados obtenidos durante el desarrollo del proyecto, mostrando ejemplos prácticos de flujos de inferencia implementados, así como la validación de su funcionamiento y eficiencia.
- **Capítulo 6: Discusión**
  - Se realiza un análisis crítico de los resultados, evaluando la efectividad de las herramientas y metodologías utilizadas. Se discuten los desafíos encontrados y las implicaciones de los resultados en el contexto de la inteligencia artificial y la ciencia de datos.
- **Capítulo 7: Conclusión**
  - Se resumen las conclusiones generales del estudio, destacando las contribuciones principales del proyecto y su impacto en el campo de la inteligencia artificial.
- **Capítulo 8: Trabajo Futuro**
  - En este capítulo se proponen áreas para futuras investigaciones y desarrollos, como la creación de más ejemplos de uso, la integración de Elyra en entornos ejecutables en OSCAR, y el desarrollo de nodos adicionales para Node-RED.
- **Capítulo 9: Referencias**
  - Se proporciona un listado completo de todas las fuentes bibliográficas y recursos consultados para la elaboración del trabajo.

2. Estado del arte y tecnologías relacionadas

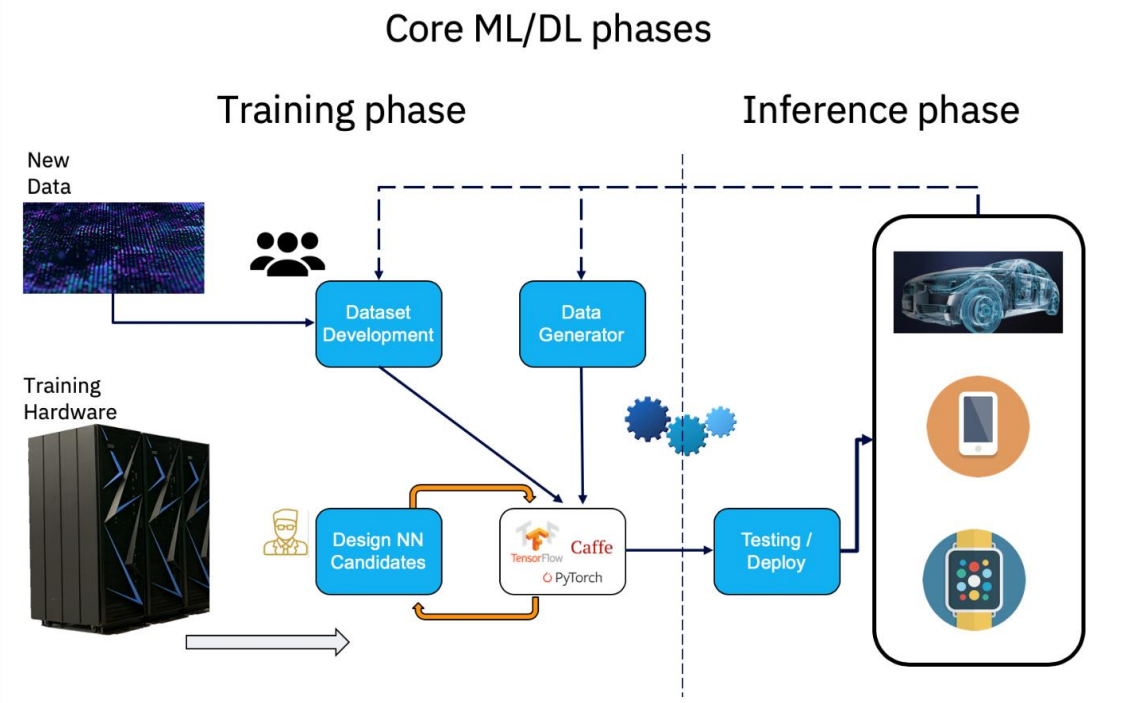

Figura 1. Diagrama de las fases del Machine Learning (Fuente: IBM)

En este apartado se analizarán los conceptos y tecnologías más relevantes en el campo de la inteligencia artificial y su aplicación en la inferencia, con un enfoque en los modelos serverless y pipelines, que son fundamentales para el desarrollo del trabajo presente.

2.1.1 Inferencia en Inteligencia Artificial

La inferencia en inteligencia artificial es el proceso en el que los modelos previamente entrenados son utilizados para hacer predicciones o tomar decisiones basadas en nuevos datos. Es una fase crítica en el ciclo de vida de la IA, que se activa después de que el modelo ha sido entrenado y preparado para su aplicación en entornos reales.

Importancia y Utilidad de la Inferencia

La inferencia permite a los modelos de IA, que han sido entrenados con grandes cantidades de datos y mediante cálculos complejos, ser aplicados de manera eficiente para analizar nuevos datos y proporcionar resultados en tiempo real o casi en tiempo real. Esto es esencial para aplicaciones que requieren respuestas rápidas, como sistemas de recomendación, asistentes virtuales y vehículos

autónomos. Un buen ejemplo de cómo Yahoo! Japón ha implementado inteligencia artificial en el uso de un motor de IA para la aprobación de anuncios en tiempo real. Anteriormente, la empresa enfrentaba retrasos significativos debido a la revisión manual de cada solicitud de anuncio, lo que afectaba la eficiencia operativa y generaba pérdidas de ingresos. Sin embargo, tras implementar un sistema basado en inteligencia artificial, Yahoo! Japan pudo reducir drásticamente el tiempo necesario para aprobar anuncios, pasando de varios días a solo unas pocas horas, con una mejora de velocidad de hasta 1,600% en la aprobación de solicitudes. Además, este sistema logró cumplir con las estrictas regulaciones gubernamentales sin necesidad de aumentar su plantilla laboral [3].

Funcionamiento de la Inferencia

En la práctica, la inferencia implica cargar un modelo ya entrenado y usarlo para procesar nuevos datos. Este proceso se puede llevar a cabo en servidores o dispositivos que no necesariamente tienen la misma capacidad computacional que aquellos utilizados para el entrenamiento. Mientras que el entrenamiento suele requerir potentes GPUs y grandes cantidades de datos, la inferencia puede realizarse en CPUs estándar o incluso en dispositivos móviles, dependiendo de la complejidad del modelo y la urgencia de la tarea. Tecnologías como el servidor de inferencia Triton de NVIDIA permiten la ejecución de modelos en diversas plataformas de hardware, optimizando los recursos y asegurando un rendimiento eficiente [1].

Desafíos de la Inferencia

Uno de los principales desafíos en la inferencia es optimizar los modelos para que funcionen eficientemente en hardware menos potente, sin comprometer la precisión o la velocidad de las predicciones. Técnicas como la poda de modelos, cuantificación y el uso de frameworks específicos permiten acelerar los procesos de inferencia sin la necesidad de recursos computacionales intensivos. Por ejemplo, Triton utiliza técnicas avanzadas para gestionar modelos grandes, como GPT-3, distribuyendo la carga de trabajo entre múltiples GPUs para mantener la eficiencia y rapidez en la inferencia [2].

Ejemplos de Aplicación

Las aplicaciones de la inferencia en IA abarcan diversos campos. En salud, se emplea para ayudar en el diagnóstico y tratamiento de enfermedades mediante el análisis rápido de grandes volúmenes de datos médicos. En el comercio, sistemas de recomendación basados en IA analizan patrones de compra para sugerir productos a los clientes en tiempo real. Empresas como American Express utilizan la inferencia de IA para la detección de fraudes en tiempo real, mejorando la seguridad y eficiencia de las transacciones financieras. Según Manish Gupta,

vicepresidente de Machine Learning y Ciencia de Datos en American Express, “nuestros algoritmos de fraude monitorean en tiempo real cada transacción de American Express alrededor del mundo, por un total de más de $1.2 billones gastados anualmente, y generamos decisiones de fraude en solo milisegundos. Mantener bajas nuestras tasas de fraude es clave para proteger a nuestros miembros y comerciantes"[3].

2.1.2 Modelo Serverless en Inferencia de IA

El modelo serverless es un enfoque en el cual la infraestructura es gestionada por un proveedor de servicios en la nube, permitiendo a los desarrolladores centrarse en la ejecución de código dirigido por eventos sin preocuparse por la administración de servidores.

Ventajas del Modelo Serverless

- **Escalabilidad Automática:** Las plataformas serverless escalan automáticamente para manejar picos de demanda, lo que es crucial para aplicaciones de inferencia en tiempo real.
- **Costos Eficientes:** En el modelo serverless, se paga solo por el tiempo de computación utilizado, reduciendo significativamente los costos operativos.
- **Gestión Simplificada:** La infraestructura es gestionada por el proveedor de servicios, reduciendo la carga administrativa y permitiendo a los desarrolladores concentrarse en la optimización de los modelos y en el desarrollo de aplicaciones.
- **Despliegue Rápido:** Facilita el despliegue de nuevas versiones de aplicaciones y modelos de IA, permitiendo iteraciones rápidas y mejoras continuas.

Integración con Pipelines de Inferencia

El modelo serverless se integra eficientemente con pipelines de inferencia, simplificando y optimizando el proceso de despliegue y operación de modelos de IA. Servicios como AWS Lambda, Google Cloud Functions y Azure Functions permiten desplegar modelos de inferencia sin la necesidad de gestionar servidores dedicados, facilitando la implementación y el mantenimiento. Además, estas funciones serverless escalan automáticamente para manejar cargas de trabajo intensivas, asignando recursos de manera dinámica y eficiente para cada tarea de inferencia.

2.1.3 OSCAR para Inferencia

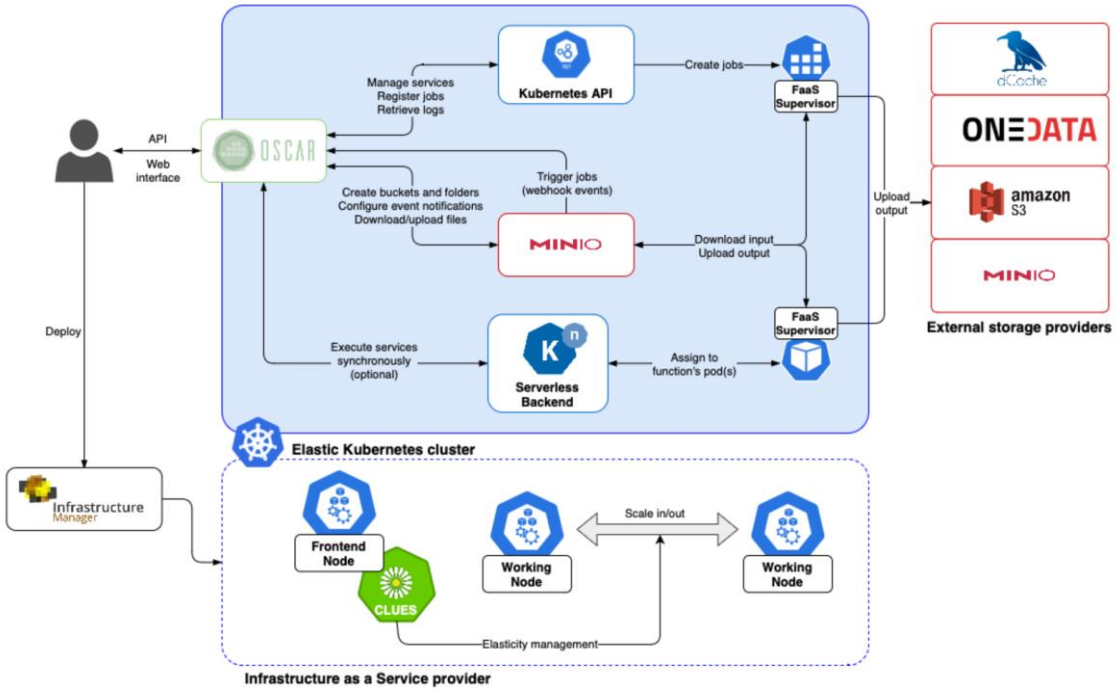

Figura 2. Diagrama del funcionamiento de OSCAR (Fuente: Página Oficial de OSCAR)

En este Trabajo de Fin de Máster, se ha optado por la plataforma OSCAR para la inferencia debido a mi familiaridad previa con ella, lo que ha permitido aprovechar su facilidad para desplegar servicios en la nube de manera eficiente. Su capacidad para gestionar servicios de inferencia a través de tokens facilita además su integración y llamada externa, lo que la convierte en una opción ideal para este proyecto.

Ventajas de OSCAR

1. **Despliegue Simplificado**: OSCAR utiliza archivos FDL (Function Description Language) para especificar detalles como la imagen Docker y los proveedores de almacenamiento para los buckets de entrada y salida. Esto facilita el despliegue automático y preciso de servicios sin necesidad de configuraciones manuales extensivas.
2. **Ejecuciones Síncronas y Eficientes**: La plataforma permite realizar ejecuciones síncronas que se integran directamente en pipelines automatizados. Con MinIO como almacenamiento de objetos, compatible con Amazon S3, la transferencia de datos de entrada y salida se realiza de manera eficiente y segura. Esto es crucial para aplicaciones que requieren manejo intensivo de datos, como análisis en tiempo real y procesamiento de grandes volúmenes de información.
3. **Optimización de Recursos**: OSCAR maximiza la utilización de recursos mediante la elasticidad automática de Kubernetes y herramientas

de gestión de elasticidad como CLUES. Esto asegura que los recursos se ajusten dinámicamente según la carga de trabajo, reduciendo costos y mejorando el rendimiento. Esta capacidad es vital para aplicaciones que experimentan variaciones en la carga de trabajo, garantizando que siempre se disponga de la cantidad adecuada de recursos.

4. **Accesibilidad y Funcionalidad**: OSCAR cuenta tanto con una interfaz de usuario (UI), que permite a los usuarios desplegar servicios de inferencia de forma amigable, como con una interfaz de línea de comandos (CLI), que facilita el despliegue de servicios de manera más ágil. Esto permite tanto a científicos de datos como a técnicos manipular y desplegar servicios para procesar datos en tiempo real sin enfrentar las complejidades asociadas con la gestión de la infraestructura.
5. **Compatibilidad y Flexibilidad**: La plataforma OSCAR puede desplegarse en múltiples proveedores cloud gracias al uso del Infrastructure Manager (IM).

2.1.4 Pipelines como Herramientas de Inferencia

Las pipelines de inferencia simplifican la interacción con servicios de IA, permitiendo a los usuarios componer flujos de trabajo de manera visual y con mínima necesidad de intervención técnica [5].

Ventajas de las Pipelines

Las pipelines de inferencia en la nube tienen varias características clave que las hacen esenciales para la gestión de flujos de trabajo de IA:

- **Modularidad:** Compuestas por múltiples módulos o nodos, cada uno responsable de una tarea específica, como el preprocesamiento de datos, ejecución de modelos y postprocesamiento de resultados.
- **Automatización:** Permiten la automatización de tareas repetitivas, reduciendo la intervención manual y minimizando errores.
- **Escalabilidad:** Pueden escalar automáticamente para manejar grandes volúmenes de datos y múltiples solicitudes de inferencia simultáneamente.
- **Interfaz Visual:** Herramientas como Node-RED y Elyra proporcionan interfaces gráficas para la creación y gestión de pipelines.
- **Integración:** Facilitan la integración de diversas herramientas y servicios, permitiendo combinar diferentes tecnologías y frameworks en un único flujo de trabajo.
- **Reusabilidad:** Los componentes de las pipelines son reutilizables, lo que permite la creación de bibliotecas de módulos para múltiples proyectos.
- **Gestión de Errores y Recuperación:** Incluyen mecanismos para la gestión de errores y la recuperación de fallos, asegurando la continuidad de los flujos de trabajo.

Estas características hacen de las pipelines herramientas poderosas para la implementación y gestión de flujos de trabajo de inferencia en la nube, mejorando la eficiencia y accesibilidad para los desarrolladores y científicos de datos.

2.1.5 Herramientas Principales para Pipelines de Inferencia

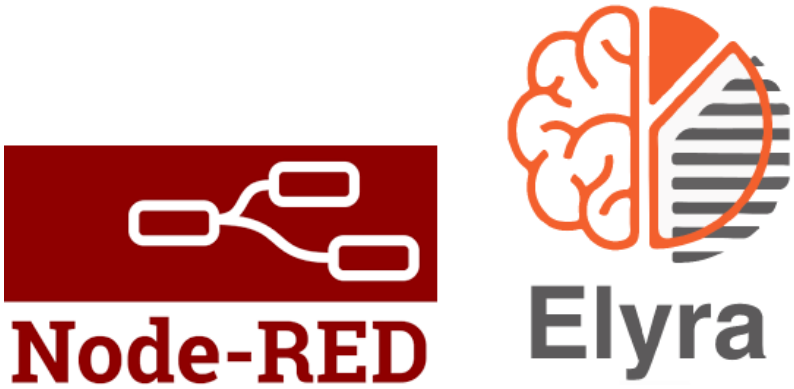

Figura 3. Node-RED y Elyra (Logos de las páginas oficiales)

Existen varias herramientas para la creación de pipelines de inferencia en inteligencia artificial, cada una con características que las hacen adecuadas para diferentes entornos y necesidades:

- **Node-RED:** Una herramienta de programación visual basada en flujo que permite conectar dispositivos, APIs y servicios mediante un editor de arrastrar y soltar. Es ampliamente utilizada en proyectos de automatización y su enfoque intuitivo la hace accesible a usuarios con distintos niveles técnicos. Además de contar con multitenancy gracias al soporte de FlowFuse.
- **Elyra:** Un proyecto de código abierto que proporciona una interfaz visual para gestionar pipelines de machine learning en Jupyter Notebooks, una plataforma comúnmente utilizada en la comunidad científica. Elyra permite una fácil integración con entornos de nube como Kubernetes y Kubeflow, facilitando la creación de pipelines escalables.
- **Kubeflow Pipelines:** Diseñada específicamente para Kubernetes, esta herramienta facilita la automatización y gestión de pipelines de machine learning a gran escala. Su capacidad de ejecución distribuida lo convierte en una opción ideal para proyectos que requieren procesamiento en infraestructuras nativas de la nube.
- **Apache Airflow:** Una plataforma versátil y robusta para la orquestación de flujos de trabajo. Airflow es ampliamente utilizado en la industria para gestionar y programar pipelines complejos y distribuidos, siendo particularmente adecuado para grandes volúmenes de datos y operaciones empresariales.

A pesar de las múltiples opciones disponibles, **Node-RED** y **Elyra** fueron seleccionadas para este proyecto por varias razones clave:

- **Elyra** fue elegida debido a su integración con **Jupyter Notebooks**, una plataforma ampliamente adoptada en la comunidad científica y de datos. Al proporcionar una interfaz visual para la creación de pipelines directamente en Jupyter, Elyra permite a los científicos de datos trabajar en un entorno familiar, lo que mejora la productividad y facilita la documentación de los experimentos. Además, la capacidad potencial de escalar en la nube con Kubernetes y Kubeflow la convierte en una herramienta poderosa para proyectos que requieren computación distribuida.
- **Node-RED** se seleccionó por su simplicidad y accesibilidad. Su diseño intuitivo basado en el flujo de trabajo visual facilita la creación de pipelines de inferencia incluso para usuarios con menos experiencia técnica. Node-RED está ampliamente extendido en la automatización y la integración de servicios, lo que lo convierte en una herramienta versátil para proyectos que requieren conexiones rápidas entre APIs y servicios en la nube. Además gracias a **FlowFuse**, una plataforma que gestiona múltiples instancias de Node-RED, facilita su uso en entornos colaborativos y escalables.

Ambas herramientas ofrecen la capacidad de escalabilidad y un entorno cómodo para distintos tipos de usuarios finales, lo que las hace idóneas para componer flujos de trabajo de inferencia de IA en el contexto de este TFM.

3. Metodología del Desarrollo

Este Trabajo Fin de Máster (TFM) adopta una metodología de desarrollo híbrida que combina elementos de metodologías ágiles, como Scrum, con prácticas de desarrollo iterativo y prototipado. Este enfoque es particularmente adecuado para proyectos de investigación y desarrollo en los que la flexibilidad, la adaptación continua y la integración de herramientas como OSCAR, Elyra y Node-RED son esenciales para alcanzar los objetivos propuestos.

3.1.1 Enfoque General

El proyecto sigue un enfoque híbrido Agile-Iterativo, que se inspira en las metodologías ágiles para manejar la planificación y ejecución de tareas, mientras que el desarrollo iterativo y el prototipado permiten adaptar el proyecto a medida que evoluciona la comprensión de las herramientas y tecnologías utilizadas.

Inspiración en Scrum:

- **Iteraciones y Revisiones Continuas:** Se adopta una estructura de sprints cortos, típica de Scrum, para gestionar el desarrollo de manera autónoma. Cada sprint se enfoca en alcanzar objetivos específicos relacionados con la implementación y validación de tecnologías como OSCAR, Elyra y Node-RED. Este enfoque permite realizar ajustes y mejoras basadas en el feedback continuo obtenido de las pruebas y experimentaciones.

Desarrollo Iterativo:

- **Proceso Iterativo:** El proyecto se desarrolla de manera iterativa, revisando y mejorando continuamente los conceptos y prototipos. Esto permite una adaptación constante del enfoque, especialmente a medida que se descubren nuevas funcionalidades o desafíos en el uso de las herramientas seleccionadas.
- **Flexibilidad:** La metodología se adapta dinámicamente en función de los resultados obtenidos en cada iteración, lo que es clave para la investigación y desarrollo tecnológico en un entorno cambiante.

Prototipado:

- **Pruebas de Concepto y Prototipos:** Se desarrollan prototipos y pruebas de concepto para validar rápidamente las ideas en un entorno práctico. Estas pruebas permiten identificar problemas y optimizar soluciones antes de integrarlas en el proyecto final.

3.1.2 Herramientas y Tecnologías Utilizadas

Una vez escogidas, el TFM se centrará en la implementación y desarrollo de las siguientes herramientas.

- **OSCAR:** Utilizado para el despliegue de servicios serverless orientados a eventos, optimizando el uso de recursos y facilitando la integración en pipelines automatizados.
- **Elyra:** Proporciona una interfaz visual para la creación y gestión de pipelines de machine learning en Jupyter Notebooks, integrando la gestión de flujos de trabajo en entornos de nube.
- **Node-RED:** Herramienta de programación visual basada en flujo que permite diseñar y gestionar flujos de trabajo interactivos y conectar diversas APIs y servicios.

Estas herramientas se seleccionan por su capacidad para simplificar y optimizar el proceso de inferencia en la nube, garantizando un desarrollo ágil y eficiente.

3.1.3 Proceso de Desarrollo

El proceso de desarrollo sigue los siguientes pasos:

1. **Configuración de Entornos:** Configuración de los entornos de trabajo para cada una de las herramientas, asegurando la compatibilidad y la correcta integración de OSCAR, Elyra y Node-RED.
2. **Desarrollo Iterativo:** Implementación de las funcionalidades clave de cada herramienta en sprints cortos, permitiendo iteraciones rápidas basadas en los resultados obtenidos y el feedback.
3. **Pruebas y Validación:** Realización de pruebas continuas en cada iteración para validar la funcionalidad y eficiencia de las soluciones desarrolladas. Se realizan ajustes según sea necesario para optimizar el rendimiento y la escalabilidad.
4. **Integración Final:** Integración de todas las herramientas y tecnologías en un flujo de trabajo coherente que cumpla con los objetivos del TFM, asegurando que todas las partes funcionen conjuntamente de manera eficiente.
5. **Documentación y Presentación:** Documentación detallada del proceso de desarrollo, resultados obtenidos y las conclusiones alcanzadas. Preparación de la presentación final del proyecto.

3.1.4 Justificación del Enfoque

Este enfoque híbrido se justifica por la necesidad de flexibilidad y adaptabilidad en un proyecto de investigación donde las variables pueden cambiar rápidamente. La metodología permite:

- **Maximizar la Eficiencia:** Optimizar el uso de tiempo y recursos, permitiendo ajustes rápidos basados en el aprendizaje continuo.
- **Enfoque en el Producto Final:** Garantizar que el proyecto de AI4Compose se mantenga centrado en desarrollar un producto final funcional y adaptado a las necesidades específicas, ajustando el enfoque según la evolución del proyecto.

La elección de esta metodología asegura que el proyecto pueda adaptarse y responder de manera efectiva a los desafíos que surjan durante el desarrollo, manteniendo un enfoque sistemático y estructurado.

4. Desarrollo de AI4COMPOSE

En este capítulo se explicarán el proceso utilizado para instalar, probar y desarrollar cada herramienta usada en este trabajo final de master.

4.1 Desarrollo de Elyra

Elyra es un proyecto de código abierto que ofrece una interfaz visual para la creación y gestión de pipelines de machine learning en Jupyter Notebooks. Ha sido escogido en este TFM por varias razones clave:

1. **Compatibilidad con Librerías de Python**: Elyra permite el uso de librerías de Python, lo cual es esencial para aprovechar las capacidades del cliente de Python para OSCAR [1] en la implementación de modelos de inferencia.
2. **Popularidad de Jupyter Notebooks**: Jupyter Notebooks es ampliamente utilizado en el entorno científico debido a su flexibilidad y capacidad para combinar código, visualizaciones y texto en un solo documento interactivo. Esto facilita la experimentación y el desarrollo colaborativo.
3. **Entorno Familiar**: Al utilizar Jupyter Notebooks, proporcionamos un entorno ya conocido y cómodo para los científicos de datos y técnicos. Esto reduce la curva de aprendizaje y permite un enfoque más directo en la resolución de problemas y el desarrollo de modelos.

Uso de Elyra con OSCAR Python

La integración de Elyra con OSCAR Python permite a los usuarios crear y gestionar pipelines de inferencia de manera eficiente, utilizando un entorno familiar y potente. Esto facilita el desarrollo, prueba y despliegue de modelos de IA, optimizando el flujo de trabajo y mejorando la productividad.

4.1.1 Características de Elyra

Elyra es una herramienta versátil y potente, ideal para la gestión de pipelines de machine learning en Jupyter Notebooks, gracias a varias características clave que optimizan su uso en entornos de ciencia de datos. Una de sus principales ventajas es su interfaz visual intuitiva, que permite a los usuarios crear y gestionar pipelines de manera gráfica, simplificando la composición de flujos de trabajo complejos sin necesidad de escribir código detallado. Esta facilidad de uso se combina con su integración nativa con Jupyter Notebooks, lo que le permite aprovechar las capacidades interactivas de Jupyter, ampliamente utilizado en el ámbito científico. Gracias a esto, los usuarios pueden combinar código,

visualizaciones y documentación en un único entorno, promoviendo la reproducibilidad y la colaboración.

Además, Elyra es totalmente compatible con librerías de Python, lo que resulta crucial para la implementación y gestión de modelos de inferencia a través de OSCAR Python. Esta compatibilidad asegura que los usuarios puedan continuar utilizando las librerías y frameworks a los que ya están acostumbrados, sin necesidad de modificar significativamente sus flujos de trabajo.

Otra de las funcionalidades destacadas de Elyra es la orquestación de pipelines de machine learning, permitiendo coordinar diversas tareas como la preparación de datos, el entrenamiento de modelos, la evaluación y el despliegue de inferencias. Esto se complementa con su integración con Kubeflow y Kubernetes, que ofrece una capacidad de escalabilidad y gestión de recursos eficiente en entornos de nube. De esta manera, es posible ejecutar pipelines de manera optimizada en infraestructuras escalables.

Asimismo, Elyra facilita la gestión de componentes reutilizables, lo que permite a los usuarios crear bibliotecas modulares que pueden ser utilizadas en múltiples proyectos, mejorando la eficiencia y evitando la duplicación de esfuerzos. Finalmente, la herramienta incluye capacidades de documentación y monitoreo integradas, permitiendo rastrear y documentar cada paso del pipeline, lo que facilita tanto la transparencia como la colaboración entre equipos.

4.1.2 Despliegue de Elyra

Requisitos Previos e Instalación

Para utilizar Elyra, es necesario contar con un entorno adecuado que incluya:

1. **JupyterLab**: Elyra se integra con JupyterLab, por lo que es necesario tenerlo instalado en el sistema.
2. **Python**: Una versión actualizada de Python instalada superior a la versión 3.8, en nuestro caso hemos usado la versión 3.8 para las pruebas.
3. **Acceso a Internet**: Para descargar paquetes y dependencias necesarios.

Pasos de Instalación

1. Instalar JupyterLab:

```
pip install jupyterlab
```

2. Instalar Elyra

```
pip install elyra
```

3. Recompilar Jupyter lab

```
jupyter lab build
```

4. Verificar Instalación

```
jupyter lab
```

Si todo funciona correctamente, JupyterLab debería abrirse con las funcionalidades de Elyra integradas, listo para su uso en el desarrollo de flujos de trabajo de ciencia de datos y otras aplicaciones.

4.1.3 Entorno de Elyra

Una vez que Elyra está instalado y configurado correctamente en JupyterLab, el entorno de trabajo muestra varias características y herramientas adicionales que facilitan el desarrollo de flujos de trabajo y la edición de código. A continuación se describen los elementos clave de la interfaz:

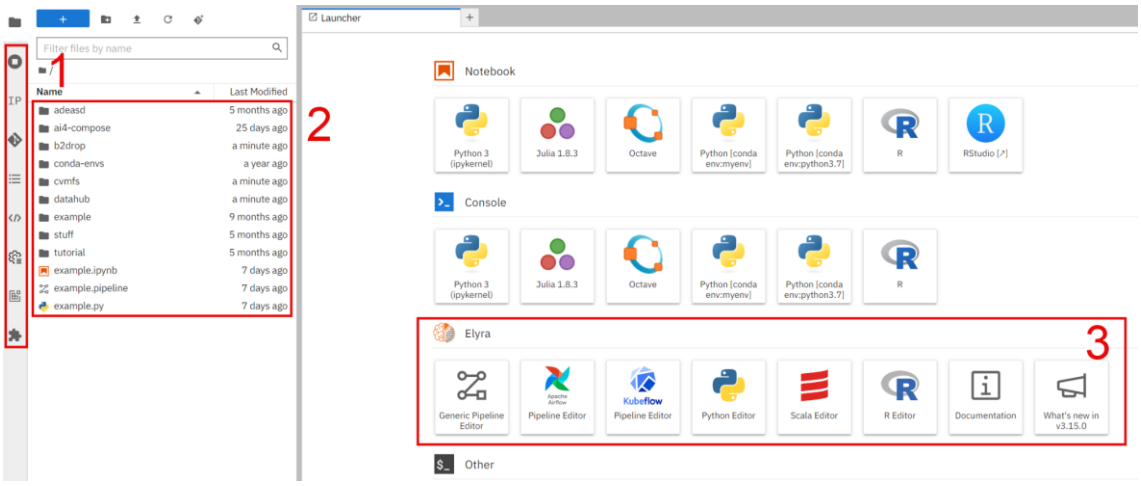

Figura 4. Entorno de Elyra

1. **Widgets Integrados**: En la parte izquierda de la interfaz, se encuentran los widgets integrados (marcado con el número 1 en la imagen), como el de GitHub, que permiten acceder y gestionar repositorios de código, entre otras funcionalidades adicionales. Estos widgets facilitan la integración con otras herramientas y servicios directamente desde el entorno de JupyterLab.
2. **Panel de Archivos**: También en la parte izquierda de la interfaz, se encuentra el panel de archivos (marcado con el número 2 en la imagen). Este panel permite navegar por el sistema de archivos, abriendo y gestionando notebooks, scripts y otros documentos necesarios para el desarrollo de proyectos.

3. **Herramientas de Elyra**: En la parte inferior del launcher, se encuentran las herramientas específicas de Elyra (marcadas con el número 3 en la imagen). Estas herramientas incluyen:
    - **Generic Pipeline Editor**: Un editor visual para crear y gestionar pipelines de datos.
    - **Pipeline Editor**: Herramientas específicas para la integración con Apache Airflow y Kubeflow.
    - **Python Editor**: Un editor especializado para scripts de Python.
    - **Scala Editor**: Un editor para código en Scala.
    - **R Editor**: Un editor para código en R.
    - **Documentation**: Acceso a la documentación de Elyra.

    - **What's new in v3.15.0**: Información sobre las novedades de la última versión de Elyra.

4.1.4 Elementos del Pipeline en Elyra:

Barra de Herramientas del Pipeline: La barra de herramientas en la parte superior (marcada con el número 1 en la imagen) proporciona varias opciones importantes para gestionar el pipeline:

- **Iniciar el Pipeline**: Permite ejecutar el pipeline actual.
- **Guardar**: Guarda el pipeline.
- **Exportar**: Exporta el pipeline en un formato específico.
- **Borrar**: Elimina el pipeline actual.
- **Abrir Runtime**: Permite configurar y seleccionar el runtime para el pipeline.
- **Abrir Componente de Catálogos**: Accede a componentes predefinidos y configuraciones.
- **Deshacer/Rehacer**: Deshacer (Ctrl + Z) y rehacer (Ctrl + Y) acciones realizadas.
- **Cortar, Copiar, Pegar**: Funciones estándar de edición.
- **Comentar**: Agrega comentarios a componentes del pipeline.
- **Eliminar**: Elimina componentes seleccionados del pipeline.
- **Organizar Horizontalmente/Verticalmente**: Organiza los componentes del pipeline de manera horizontal o vertical para mejor visualización.

Atajos para Crear Documentos: En la parte izquierda (marcada con el número 2 en la imagen), se encuentran los atajos para crear nuevos notebooks y scripts:

- **Notebook**: Ideal para desarrollar y documentar análisis de datos, incluyendo visualizaciones y explicaciones detalladas.

- **Script de Python**: Útil para escribir y ejecutar scripts de Python específicos y funciones pequeñas. Por ejemplo, convertir una imagen a formato base64.
- **Script de R**: Para los usuarios que prefieren utilizar R para sus análisis de datos.

La diferencia principal entre usar un notebook y un script radica en el propósito y el alcance de la tarea. Los notebooks son más adecuados para trabajos que requieren documentación extensa y visualizaciones, mientras que los scripts son más eficientes para funciones pequeñas y tareas automatizadas que no necesitan interacción con el usuario.

Área de Trabajo del Pipeline: En la parte central (marcada con el número 3 en la imagen), se encuentra el área de trabajo donde se arrastran y configuran los componentes del pipeline. Esta área permite construir visualmente el flujo de trabajo al arrastrar componentes desde el panel de archivos o desde el catálogo de componentes.

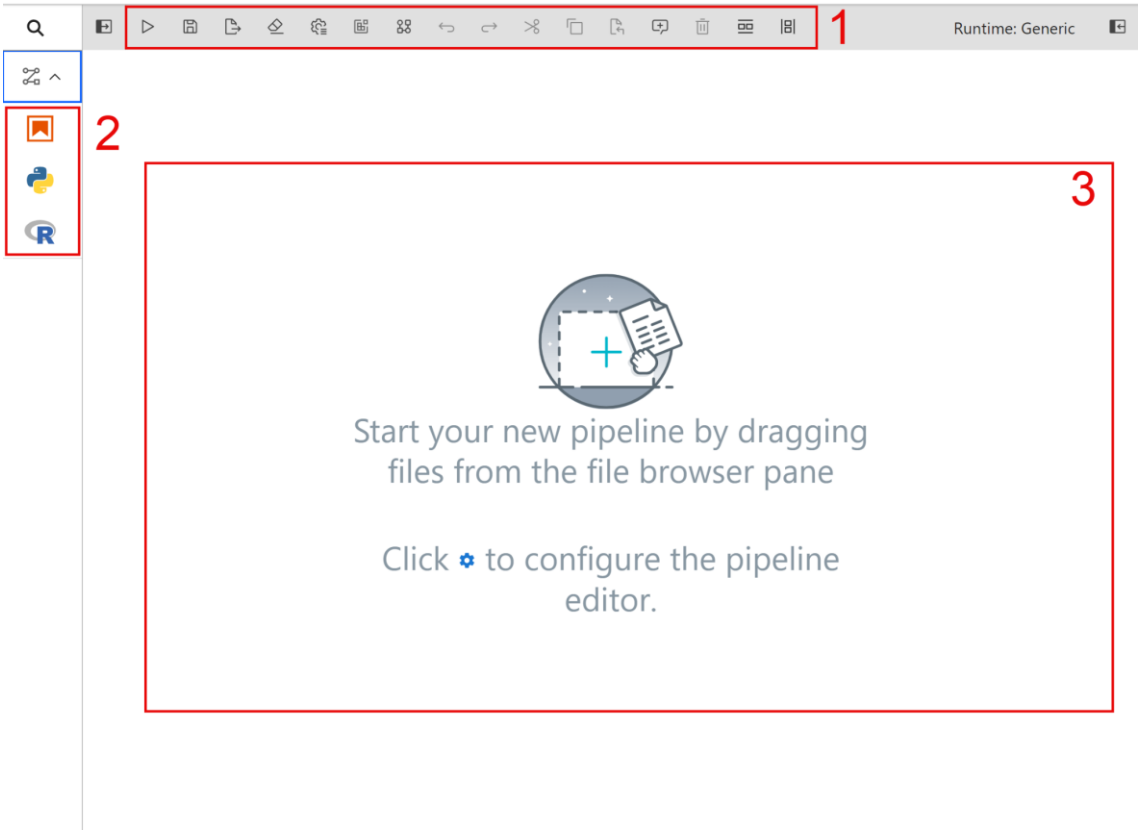

Figura 5. Espacio de Trabajo de Elyra

4.1.5 Uso de nodos en Elyra:

En Elyra, los nodos representan diferentes tareas que se pueden encadenar para formar un pipeline de datos. A continuación se explican dos tipos de nodos comunes: los scripts de Python y los notebooks.

Nodos de Script de Python y Notebooks

1. **Nodos de Script de Python**: Los nodos de script de Python son ideales para ejecutar tareas específicas y pequeñas que no requieren una gran interacción con el usuario. Por ejemplo, un script que convierte una imagen a formato Base64. En Elyra, estos nodos se configuran para ejecutar scripts de Python y pueden recibir entradas y producir salidas que se pasan a otros nodos en el pipeline.
2. **Nodos de Notebook**: Los notebooks son más adecuados para tareas de análisis de datos que requieren visualización y documentación detallada. En Elyra, un nodo de notebook ejecuta un Jupyter Notebook y puede tomar entradas de nodos anteriores y pasar salidas a nodos posteriores.

A continuación, en la figura 6, se muestra un ejemplo de pipeline con un nodo de script de Python conectado a un nodo de notebook:

Figura 6. Ejemplo de un pipeline en Elyra

4.1.6 Configuración de nodos en Elyra

Cada nodo en Elyra tiene un panel de configuración donde se pueden definir varias propiedades y parámetros. A continuación, se describe cada sección del panel de configuración (ver Figura 4):

Label: Permite asignar un nombre descriptivo al nodo para identificarlo fácilmente en el pipeline.

Inputs:

- **Filename**: Especifica el archivo que se ejecutará, como un script de Python o un notebook.

- **Runtime Image**: Selecciona la imagen de entorno de ejecución que contiene todas las dependencias necesarias para ejecutar el script o notebook.
- **CPU** y **RAM (GB)**: Define los recursos de CPU y memoria asignados al nodo.
- **GPU** y **GPU Vendor**: Especifica los recursos de GPU si son necesarios para la tarea.
- **File Dependencies**: Permite agregar archivos adicionales que el script o notebook puede necesitar.
- **Include Subdirectories**: Opción para incluir subdirectorios en las dependencias de archivos.

Outputs:

- **Output Files**: Define los archivos que se generarán como salida del nodo.

Additional Properties:

Aunque Elyra presenta muchas propiedades y funciones adicionales tales como poder trabajar con Kubernetes, en este TFM solo se explorará el uso de las variables de entorno para utilizarlo como rutas de entradas y salidas de datos.

- **Environment Variables**: Aquí es donde se configuran las variables de entorno que pueden ser usadas para pasar información entre nodos. Por ejemplo, se pueden definir variables para especificar rutas de entrada y salida de datos, parámetros de configuración, etc.

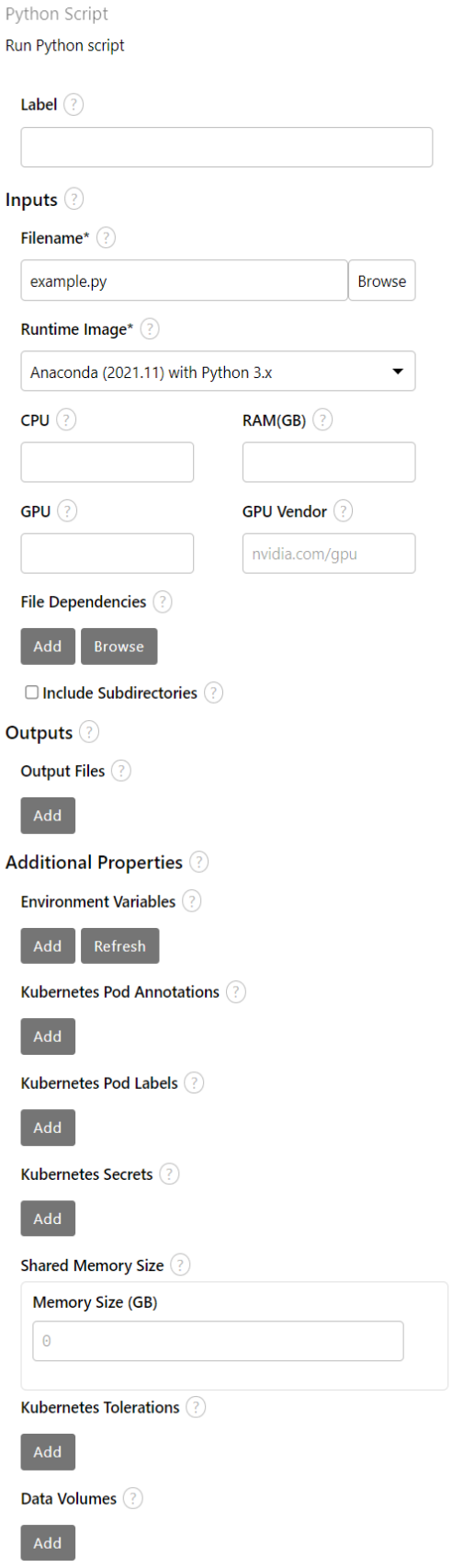

Figura 7. Tabla de configuraciones de Elyra

4.1.7 Ejemplo de Configuración de Variables de Entorno

Las variables de entorno son una forma flexible de pasar información entre nodos en Elyra. A continuación, se muestra un ejemplo de cómo configurar una variable de entorno:

```
environment_variables:
  - name: INPUT_FILE
    value: "/path/to/input/file"
  - name: OUTPUT_FILE
    value: "/path/to/output/file"
```

En este ejemplo, el nodo puede acceder a *INPUT_FILE* y *OUTPUT_FILE* dentro de su script o notebook, permitiendo que los nodos compartan información y trabajen de manera conjunta.

Con estos conceptos y configuraciones, Elyra permite construir y gestionar pipelines de datos complejos de manera eficiente, facilitando el desarrollo, la ejecución y la administración de flujos de trabajo de datos.

4.1.8 Uso de OSCAR Python en Elyra

OSCAR Python es una librería diseñada para facilitar la interacción con clústeres y servicios de OSCAR a través de solicitudes HTTP asíncronas. A continuación, se explica cómo importarlo y utilizarlo dentro de los notebooks de Jupyter.

Importación de OSCAR Python y OIDC

Para importar OSCAR Python en un notebook de Jupyter, primero es necesario instalar la librería junto con sus dependencias. Esto se puede hacer ejecutando los siguientes comandos:

La instalación de liboidcagent es esencial para gestionar la autenticación segura mediante OpenID Connect (OIDC), que permite acceder a servicios protegidos, como OSCAR, mediante tokens de autenticación sin exponer credenciales.

```
!{sys.executable} -m pip install oscar-python
!pip install liboidcagent
from oscar_python.client import Client
```

Funciones Principales

Una vez importada la librería se pueden utilizar varias funciones clave para interactuar con los servicios. A continuación, se describen algunas de las funciones más importantes:

1. **Crear un Cliente de ejemplo**
   Para empezar con las pruebas primero crearemos un cliente con basic auth para de esta manera utilizar OSCAR Python. Siendo este cliente el punto de partida para todas las interacciones con los servicios. Posteriormente, ya terminadas las pruebas utilizaremos OIDC para una gestión más segura del cliente. (Ambos métodos se ilustran en los dos siguientes recuadros de código.)

```
options_basic_auth = {
   'cluster_id': 'cluster-id',
   'endpoint': 'https://cluster-endpoint',
   'user': 'username',
   'password': 'password',
   'ssl': 'True'
}
client = Client(options=options_basic_auth)
```

```
options_oidc = {
   'cluster_id': 'oscar-egi-cloud',
   'endpoint': 'https://inference.cloud.ai4eosc.eu',
   'oidc_token': access_token,
   'ssl': 'True'
}
oscar_client = Client(options=options_oidc)
```

2. Crear un Servicio

Para crear un nuevo servicio, se utiliza el método *create_service*. Este método permite especificar los detalles del servicio que se desea crear. El servicio puede ser definido a través de una ruta a un archivo o mediante un JSON.

```
response = client.create_service("path_to_fdl_or_json")
if response is None:
   print("Servicio creado exitosamente")
else:
   print("Error al crear el servicio:", response)
```

3. Llamar a un Servicio

Para interactuar con un servicio ya creado se puede utilizar el método *run_service*. Este método permite enviar solicitudes al servicio y obtener las respuestas correspondientes.

```
response = client.run_service(service_name='mi_servicio', input_data={'key': 'value'})
if response.status_code == 200:
   print("Respuesta del servicio:", response.text)
else:
   print("Error al llamar al servicio:", response.status_code)
```

4. Obtener Información del Servicio

Para obtener la definición de un servicio específico, se utiliza el método *get_service*.

```
service_info = client.get_service("mi_servicio")
print("Información del servicio:", service_info.text)
```

5. Borrar un Servicio

Si se necesita eliminar un servicio, se puede utilizar el método remove_service. Este método elimina el servicio especificado.

```
response = client.remove_service("mi_servicio")
if response.status_code == 200:
    print("Servicio eliminado exitosamente")
else:
    print("Error al eliminar el servicio:", response.status_code)
```

4.1.9 Pruebas y Validación

En este apartado describiremos el proceso de pruebas y validación utilizando OSCAR Python para interactuar con un servicio de demostración llamado *cowsay*[3]. Este servicio toma un mensaje de entrada y devuelve el mensaje junto con un ASCIImoji de una vaca que lo presenta en un bocadillo. A continuación, se detallan los pasos y la arquitectura del proceso.

Arquitectura del Proceso

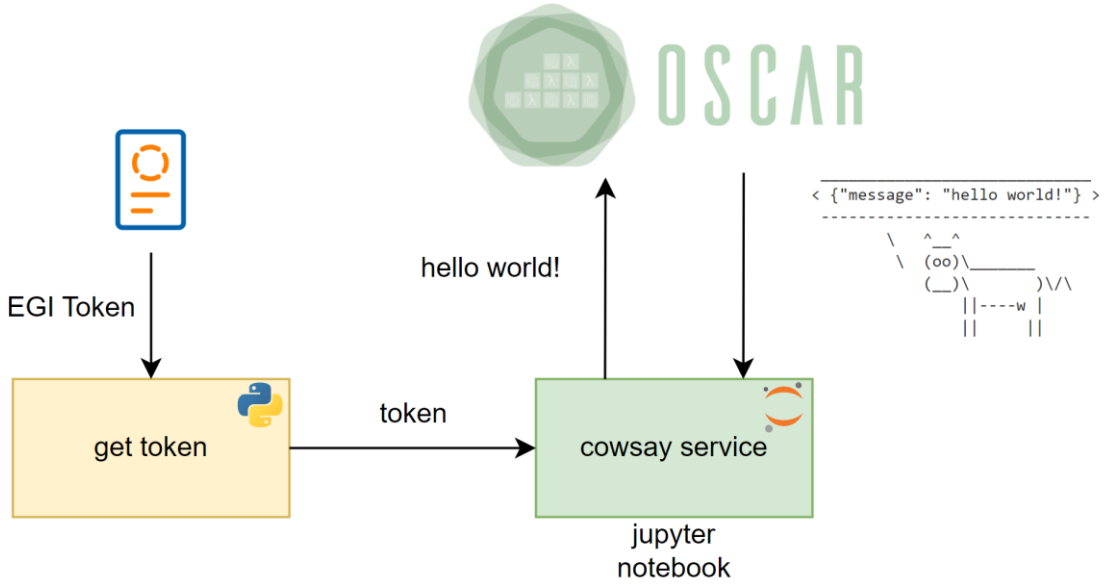

Figura 8. Arquitectura de Cowsay

1. **Generación de Token**:
    - El script de Python obtiene un token OIDC desde EGI Notebooks. Un token OIDC (OpenID Connect) es un identificador seguro utilizado para autenticar y autorizar el acceso a servicios protegidos. En este contexto, el token actúa como una credencial que verifica la

[3] Ejemplo del repositorio de GitHub: https://github.com/ai4os/ai4-compose/tree/main/elyra/examples/cowsay

identidad del usuario y le otorga permisos para interactuar con servicios como OSCAR.

2. **Almacenamiento de Credenciales**:
    - Las credenciales generadas se guardan como variables de entorno.
    - Estas variables de entorno son leídas por el siguiente nodo, que es un notebook de Jupyter.
3. **Llamada al Servicio Cowsay**:
    - El notebook de Jupyter utiliza las credenciales usando el token, para crear un nuevo cliente de OSCAR.
    - Con el cliente creado, se llama al servicio *cowsay* utilizando la función *run_service*.
    - Se pasa un mensaje JSON como entrada, que se define también como una variable de entorno.
    - El servicio *cowsay* , que se ejecuta en el cluster de OSCAR remoto y no en el propio entorno de ejecución del Notebook, procesa el mensaje y devuelve la respuesta con el emoji de la vaca. Esto permite externalizar tareas computacionalmente complejas a clusters de OSCAR desplegados en infraestructuras distribuidas remotas (como un proveedor Cloud).

Implementación del proceso:

Una vez ya tenemos clara la arquitectura en el apartado de implementación, se crean dos nodos: uno para un script de Python y otro para un notebook de Jupyter. A continuación, se describe la lógica de cada uno y cómo se configuran en Elyra.

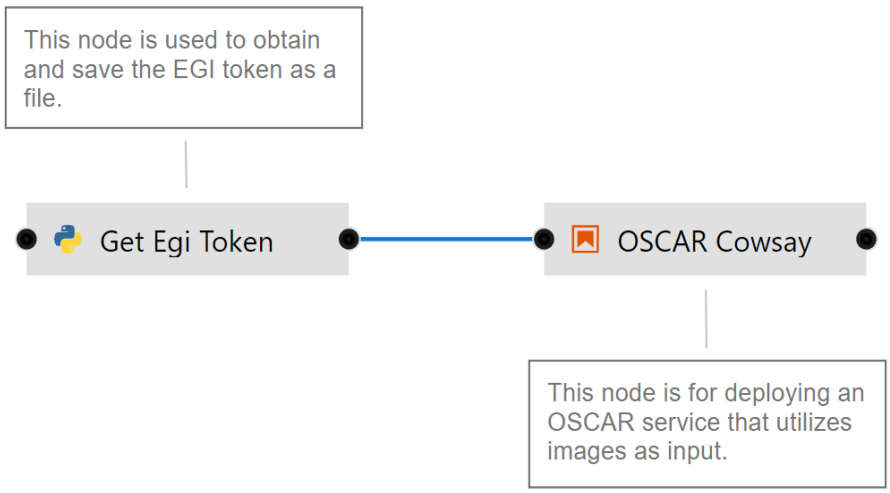

Figura 9. Despliegue del flujo de trabajo de Cowsay

Nodo 1: Script de Python

El primer nodo es un script de Python que lee un token de EGI, lo codifica en Base64 y guarda el token codificado en un archivo. Este nodo utiliza variables de entorno para definir la ubicación donde se guardará el token y el nombre del archivo.

Explicación del Script:

1. **Lectura del Token**: Se lee el token de acceso desde una ubicación específica.
2. **Codificación en Base64**: El token se codifica en Base64 para mayor seguridad durante la transmisión.
3. **Guardado del Token Codificado**: El token codificado se guarda en un archivo cuya ubicación y nombre se definen mediante variables de entorno.

Variables de Entorno:

- *ENCODED_TOKEN_DIR*: Directorio donde se guardará el archivo del token codificado.
- *ENCODED_TOKEN_FILE:* Nombre del archivo del token codificado.

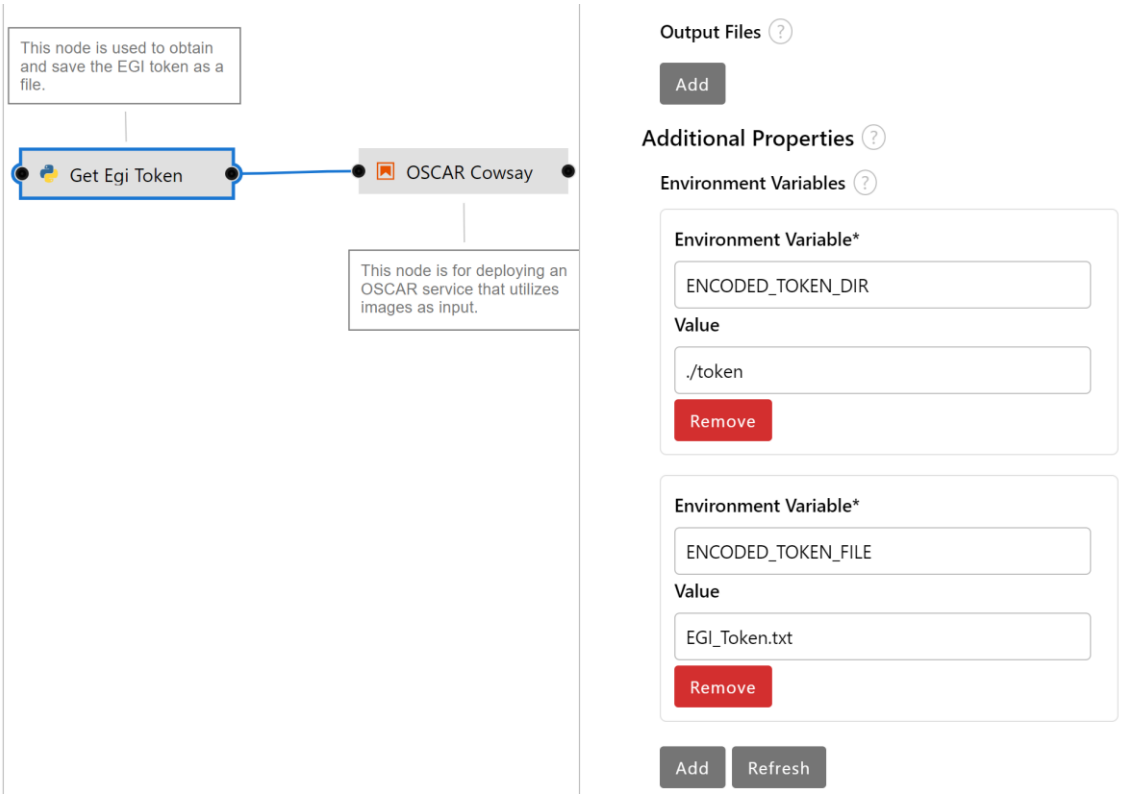

Figura 10. Configuración de los nodos de Elyra 1

Nodo 2: Notebook de Jupyter

El segundo nodo es un notebook de Jupyter que utiliza las credenciales generadas para llamar al servicio *cowsay*. Este nodo recibe dos variables de entorno: el mensaje que se usará para el servicio *cowsay* y la ubicación del token codificado.

Explicación del Notebook:

1. **Lectura de Credenciales**: El notebook lee las credenciales codificadas desde un archivo.
2. **Configuración del Cliente OSCAR**: Utiliza las credenciales para configurar un cliente de OSCAR.
3. **Llamada al Servicio Cowsay**: Utiliza el cliente para llamar al servicio *cowsay* con el mensaje de entrada definido.

Variables de Entorno:

- *ENCODED_TOKEN_PATH:* Ruta al archivo que contiene el token codificado.
- *COWSAY_MESSAGE:* Mensaje que se enviará al servicio *cowsay*.

Configuración del Pipeline en Elyra

1. **Nodo 1 (Script de Python)**:
    - **Variables de Entorno**:
        - *ENCODED_TOKEN_DIR:* Directorio donde se guardará el archivo del token codificado.
        - *ENCODED_TOKEN_FILE:* Nombre del archivo del token codificado.
2. **Nodo 2 (Notebook de Jupyter)**:
    - **Variables de Entorno**:
        - *ENCODED_TOKEN_PATH:* Ruta al archivo que contiene el token codificado.
        - *COWSAY_MESSAGE:* Mensaje que se enviará al servicio *cowsay*.

Con estas configuraciones los nodos y variables de entorno en Elyra, nos aseguramos una integración fluida y efectiva entre los diferentes componentes del pipeline, permitiendo que un usuario final pueda realizar pruebas y validaciones de manera eficiente.

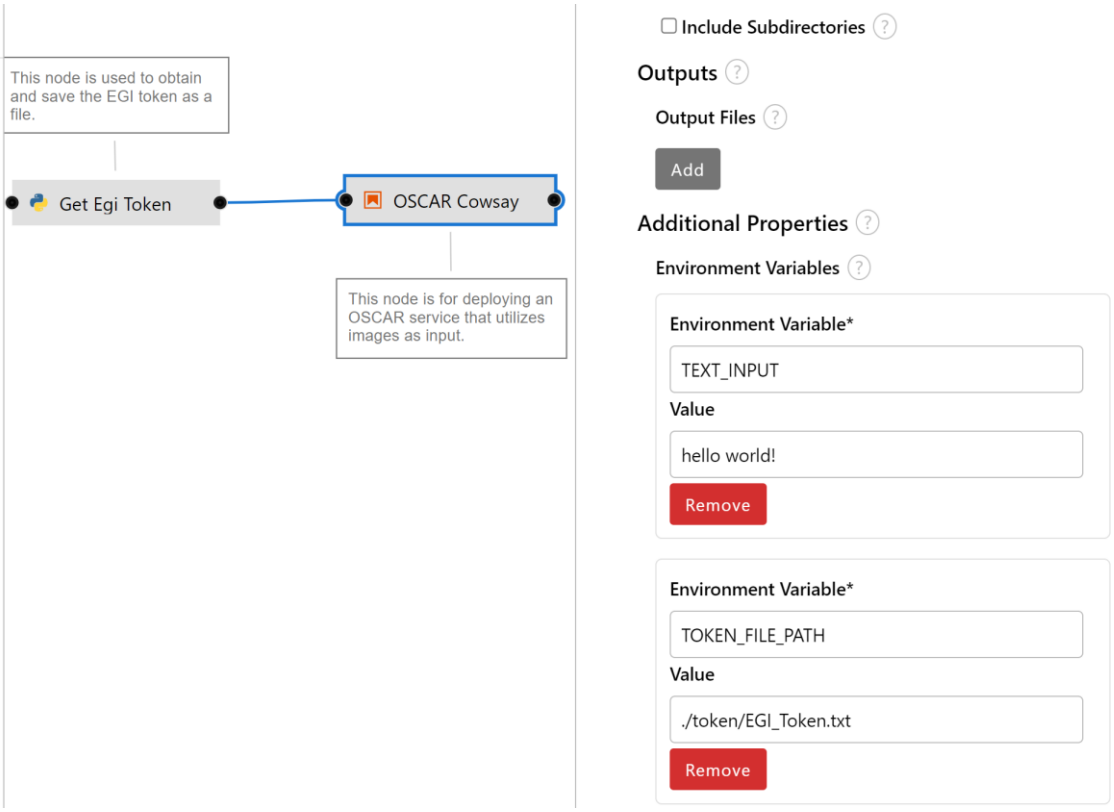

Figura 11. Configuración de los nodos de Elyra 2

Ya teniendo todas las configuraciones y código implementado es hora de arrancar el pipeline, esto iniciará la ejecución de los nodos en orden de dependencia, empezando primero por el script de Python que genera el archivo que contiene el token que usará el segundo nodo para poder generar las credenciales para interactuar con el servicio de OSCAR. En las figuras 12 y 13 se puede ver el proceso de inicio.

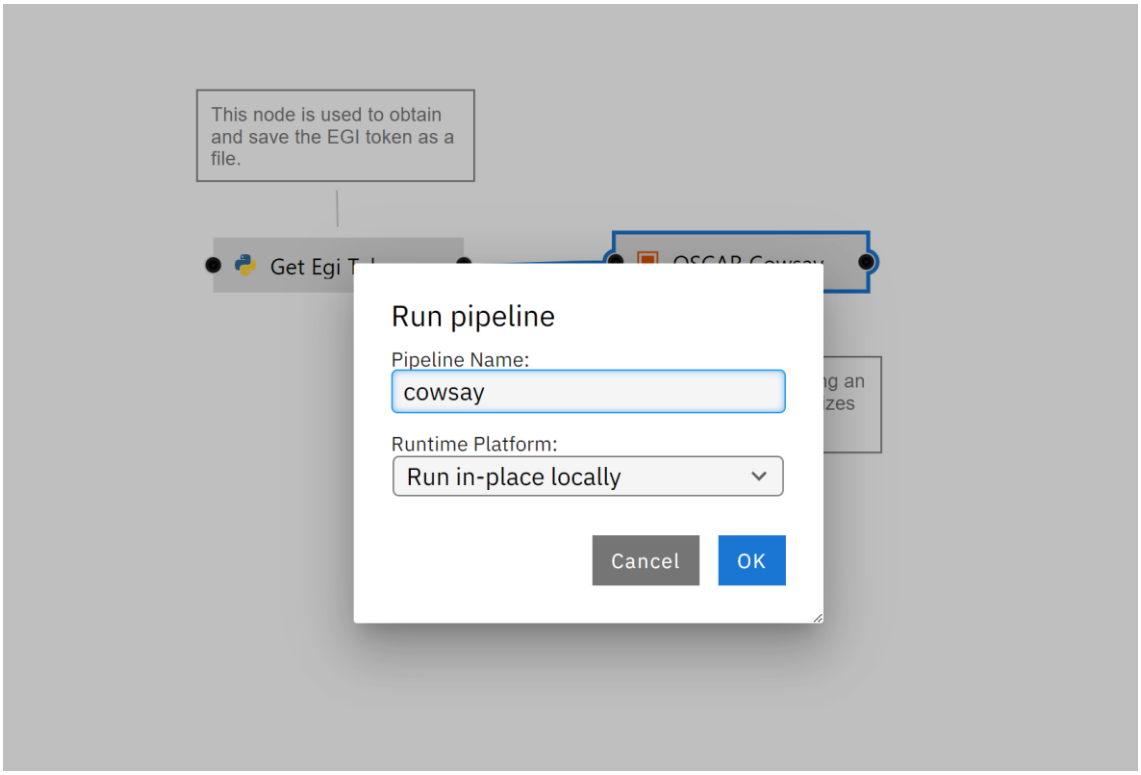

Figura 12. Proceso de inicio del flujo de trabajo en Elyra

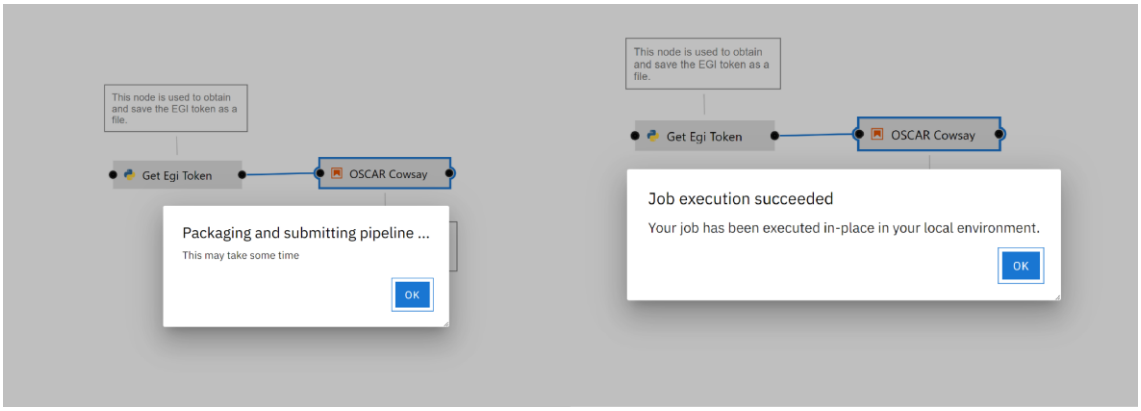

Figura 13. Proceso de ejecución del flujo de trabajo en Elyra

Cuando el servicio ha finalizado, podemos interactuar con el nodo del notebook para abrir el mismo y ver cómo se ha ejecutado de forma correcta, de tal manera que obtenemos el resultado esperado. Con esto confirmamos que hemos configurado todo de forma correcta. Ya, con esta prueba realizada es posible crear nodos más complejos que utilicen inferencia, como por ejemplo un clasificador de plantas.

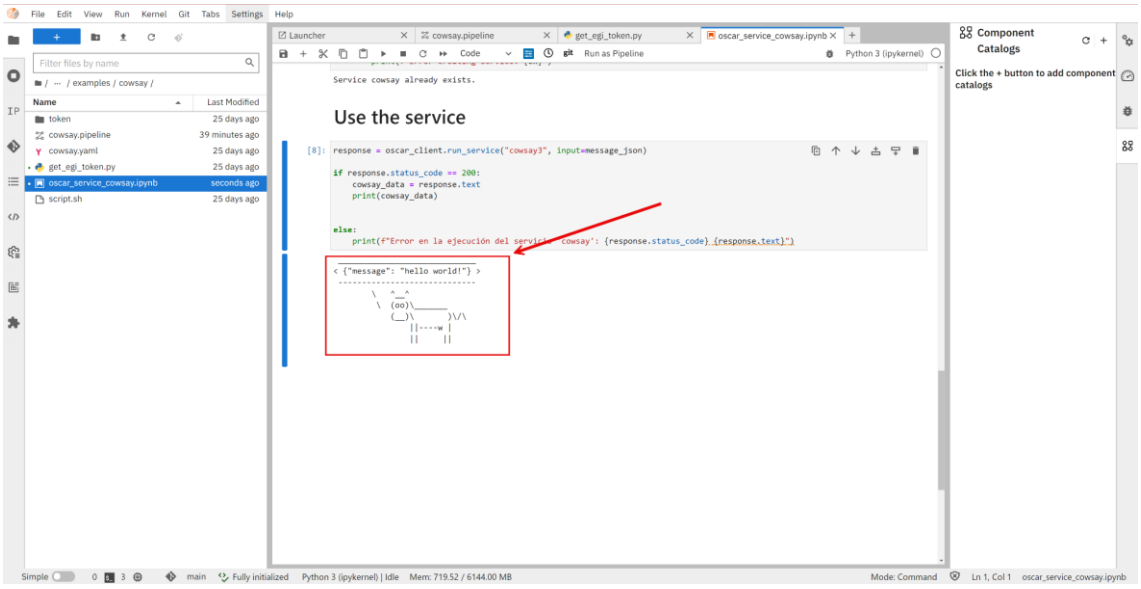

Figura 14. Resultado de Cowsay

4.1.10 Clasificación de Plantas en Elyra:

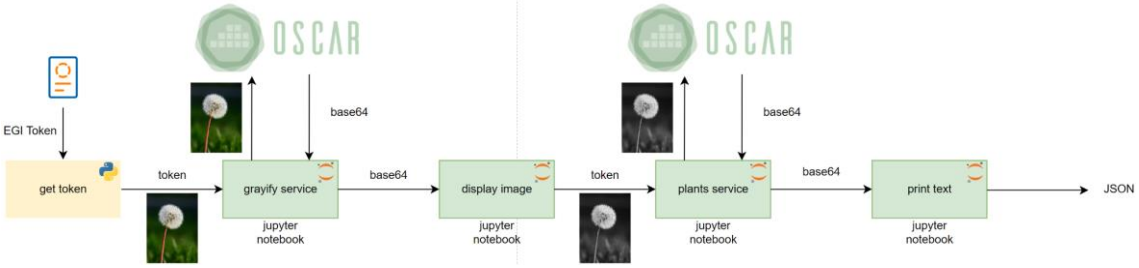

Figura 15. Arquitectura de Clasificación de Plantas en Elyra

El pipeline de clasificación de plantas[4] en Elyra incorpora diversos nodos para procesar imágenes utilizando un modelo de inferencia. Este escenario reutiliza el nodo de generación de token y emplea servicios de OSCAR para la transformación de imágenes y clasificación de plantas. La implementación está diseñada para demostrar cómo Elyra puede manipular y transformar datos e imágenes, integrando diferentes herramientas y servicios. A continuación, se describe cada nodo y su función en el pipeline.

Pipeline y Flujo de Datos

1. **Nodo: Get EGI Token**
    - **Función**: Este nodo es responsable de obtener y guardar el token de EGI como un archivo. Este token es esencial para autenticar y generar las credenciales necesarias para interactuar con los servicios de OSCAR.
2. **Nodo: Grayify Service**

[4] Ejemplo del repositorio de GitHub: https://github.com/ai4os/ai4-compose/tree/main/elyra/examples/plants-classification

- **Función**: Recibe una imagen y el token de EGI, genera las credenciales para interactuar con OSCAR y envía la imagen al servicio *Grayify*. Este servicio convierte la imagen a escala de grises. El resultado es un archivo de texto cifrado en Base64.

3. **Nodo: Display Image**
   - **Función**: Este nodo toma el archivo de texto en formato base64 del nodo anterior y lo convierte en una imagen utilizando librerías de Python. Esto demuestra la capacidad de Elyra para manipular y visualizar imágenes.
4. **Nodo: Plants Service**
   - **Función**: Utiliza la imagen en escala de grises junto con el token para enviarla al servicio de clasificación de plantas. Este servicio analiza la imagen y devuelve un archivo de texto en formato base64 con la clasificación de la planta.
5. **Nodo: Print Text**
   - **Función**: Similar al nodo de *Display Image*, este nodo toma el archivo de texto base64 y lo convierte a texto plano. El resultado contiene información detallada sobre la planta, como la probabilidad de su clasificación y enlaces a recursos adicionales como Wikipedia.

Descripción de la Implementación

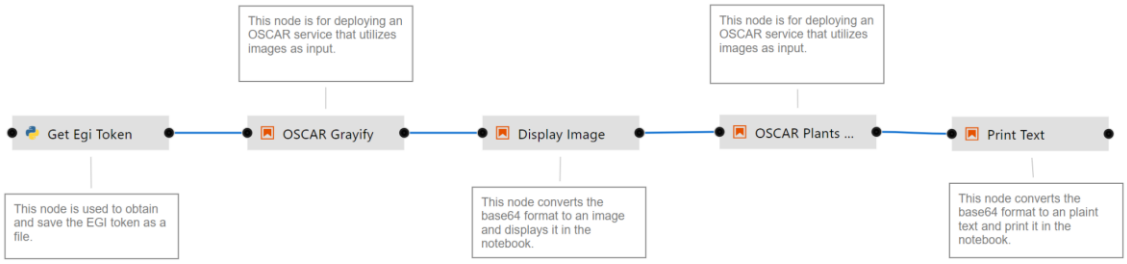

Figura 16. Flujo de trabajo de clasificación de plantas en Elyra

El pipeline comienza con la adquisición de un token de seguridad que permite la interacción autenticada con los servicios de OSCAR. La primera transformación se realiza en el servicio *Grayify*, que prepara la imagen para la clasificación al convertirla a escala de grises, una técnica común en el preprocesamiento para análisis de imágenes.

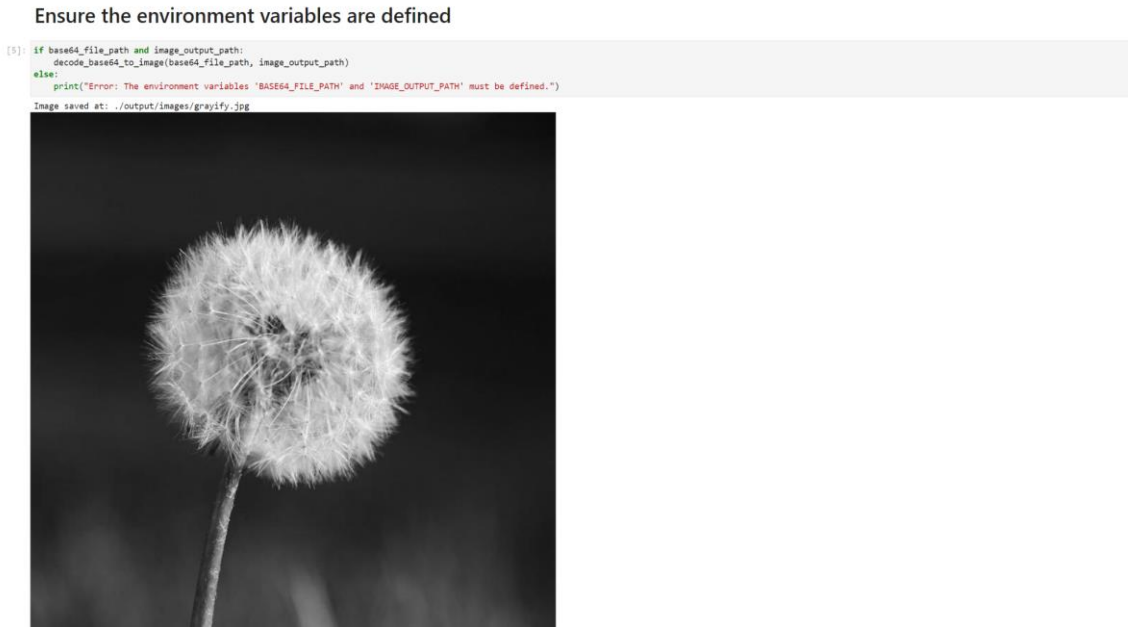

Figura 17. Resultado de Grayify

Posteriormente, la imagen se procesa en el servicio de clasificación de plantas, que utiliza un modelo de aprendizaje automático para identificar características específicas de la planta en la imagen.

Es importante recordar que el proceso de inferencia se puede ejecutar en un cluster de OSCAR remoto que ofrezca capacidades de cómputo adicionales no disponibles en el entorno de ejecución del Notebook de Jupyter, como puede ser el soporte a GPUs.

El resultado de la clasificación se presenta en formato JSON, proporcionando la identificación de la planta y también enlaces útiles para información adicional. Todos estos datos podrían utilizarse para hacer medias o incluso entrenar futuros modelos, demostrando así la utilidad de usar Elyra para inferencia en inteligencia artificial.

Figura 18. Resultado de Clasificación de Plantas

Visualización y Análisis

Cada nodo no solo ejecuta una función específica, sino que también sirve como punto de control para visualizar y verificar los datos intermedios. Esto es crucial para el *debugging* y la validación del flujo de datos a través del pipeline. La capacidad de convertir y mostrar imágenes y texto directamente dentro de Elyra facilita una comprensión más profunda del proceso de transformación de datos y los resultados del modelo.

4.1.11 Consideraciones Finales de Elyra

Elyra ofrece un entorno familiar para los científicos de datos, permitiendo programar de manera sencilla ejemplos de inferencia mediante pipelines. Una de sus grandes ventajas es que estos pipelines pueden ser almacenados en repositorios o guardados localmente para su reutilización, lo que facilita su mantenimiento y compartición. Además, Elyra permite integrar herramientas avanzadas de Python para análisis de datos e inteligencia artificial, lo que amplía las posibilidades de obtener resultados más completos y personalizados en los procesos de inferencia.

4.2 Desarrollo de Node-RED

Node-RED es una herramienta de programación visual, mediante pipelines, que permite conectar dispositivos, APIs y servicios en línea mediante un editor de arrastrar y soltar. En este TFM, Node-RED se utiliza para crear flujos de trabajo de inferencia de IA, facilitando la integración con OSCAR y otros servicios. Además, su multitenancy gracias a FlowFuse permite gestionar múltiples usuarios y proyectos de manera eficiente, haciéndolo ideal para entornos colaborativos. Otra ventaja clave es su amplia comunidad, que proporciona una gran cantidad de tutoriales y nodos creados por los propios usuarios, lo que simplifica la implementación y mejora la accesibilidad para desarrolladores con diferentes niveles de experiencia.

Ventajas de Node-RED

1. **Programación Visual**:
    - La interfaz de arrastrar y soltar simplifica la creación de flujos de trabajo, permitiendo a los usuarios diseñar y gestionar procesos complejos sin necesidad de escribir código extenso.
2. **Amplia Biblioteca de Nodos**:
    - Node-RED ofrece una extensa biblioteca de nodos predefinidos para interactuar con diferentes servicios y APIs, lo que facilita la integración y automatización de tareas.
3. **Flexibilidad y Extensibilidad**:

    - Los usuarios pueden crear nodos personalizados utilizando JavaScript, lo que permite adaptar la herramienta a necesidades específicas y ampliar sus capacidades según los requisitos del proyecto.
4. **Escalabilidad**:
    - Node-RED puede ejecutarse en diversas plataformas, desde dispositivos de borde hasta entornos de nube, escalando según las necesidades del flujo de trabajo de inferencia.
5. **Comunidad Activa**:
    - Node-RED cuenta con una comunidad activa que contribuye con nuevos nodos y mejoras, lo que asegura un crecimiento constante y soporte para nuevas tecnologías y servicios.

Integración de Node-RED con OSCAR

Utilizando Node-RED, los usuarios pueden crear flujos de trabajo que integren servicios de inferencia de OSCAR de manera sencilla. Esto implica:

- **Configuración de Nodos OSCAR**: Nodos específicos para interactuar con OSCAR, donde se puede configurar la imagen, los datos del servicio (endpoint, nombre del servicio, token) y recuperar los resultados.
- **Monitorización y Gestión**: Integrar nodos de monitorización para supervisar el estado y rendimiento de los flujos de trabajo, asegurando una operación eficiente y fiable.

4.2.1 Características de Node-RED

Node-RED es una herramienta versátil y potente para la creación de flujos de trabajo a través de una interfaz visual intuitiva que permite diseñar y gestionar flujos sin necesidad de escribir código complejo. Una de sus principales ventajas es su extensa biblioteca de nodos predefinidos, que facilita la interacción con APIs, servicios, bases de datos, y dispositivos. Esto simplifica la integración y automatización de procesos. Además, ofrece la posibilidad de crear nodos personalizados utilizando JavaScript, lo que permite adaptarlo a necesidades específicas y ampliar sus capacidades.

Node-RED es extremadamente flexible, pudiendo ejecutarse en diversas plataformas, desde dispositivos de borde como Raspberry Pi hasta entornos en la nube, permitiendo escalar según las necesidades del proyecto. También cuenta con una comunidad activa que contribuye continuamente con nuevos nodos y mejoras, ofreciendo un soporte constante y acceso a recursos y tutoriales. La herramienta facilita la integración con servicios externos para la creación de soluciones eficientes en entornos como IoT y automatización de procesos.

Otra ventaja es su capacidad para monitorizar y gestionar el rendimiento de los flujos de trabajo, lo que garantiza una operación fiable. Node-RED permite además modularizar los flujos complejos en partes más manejables, facilitando su mantenimiento y escalabilidad. En entornos colaborativos, FlowFuse añade una funcionalidad clave: el multitenancy, permitiendo gestionar múltiples usuarios y proyectos de manera eficiente y segura. Por último, ofrece opciones de seguridad y autenticación para proteger los flujos y controlar el acceso, asegurando que solo los usuarios autorizados puedan hacer cambios o acceder a datos sensibles.

Por otro lado, pese a que es posible instalar Node-RED en tu propio dispositivo, nos centraremos en su instalación a través de FlowFuse.

4.3 Desarrollo de FlowFuse

Figura 19. Logo FlowFuse de su página oficial

FlowFuse (anteriormente conocida como FlowForge) es una plataforma diseñada para permitir el uso colaborativo de Node-RED, proporcionando una infraestructura que facilita el trabajo conjunto en la creación y gestión de flujos de trabajo. Entre sus principales ventajas se encuentra la capacidad de colaboración en tiempo real, permitiendo que varios usuarios trabajen simultáneamente en el mismo flujo, lo que agiliza el desarrollo y reduce los tiempos de implementación. Además, FlowFuse cuenta con un sistema de gestión de usuarios y roles, lo que permite asignar permisos específicos a los miembros del equipo, asegurando un control adecuado sobre quién puede realizar cambios en los flujos de trabajo.

4.3.1 Características de FlowFuse

Una característica fundamental de FlowFuse es su control de versiones, que permite rastrear cambios, revertir a versiones anteriores y gestionar el historial de desarrollo, garantizando una mayor seguridad y capacidad de recuperación. FlowFuse también facilita la integración de prácticas de integración continua y despliegue (CI/CD), lo que asegura que los cambios se prueben y desplieguen de manera eficiente sin interrumpir el flujo de trabajo.

En términos de escalabilidad y desempeño, FlowFuse está diseñado para gestionar proyectos complejos y grandes volúmenes de datos, asegurando un rendimiento óptimo conforme crecen las necesidades del proyecto. Esto lo

convierte en una solución robusta y escalable para el desarrollo colaborativo de flujos de trabajo de inferencia de IA.

En este proyecto se utilizará la versión gratuita de FlowFuse, que, aunque permite la colaboración y las funcionalidades básicas, tiene ciertas limitaciones, como la falta de acceso a características avanzadas de CI/CD y la restricción en la cantidad de usuarios y flujos que pueden ser gestionados simultáneamente. Sin embargo, para los propósitos de este Trabajo de Fin de Máster, esta versión es suficiente para demostrar su utilidad en un entorno colaborativo.

4.3.2 Desarrollo de Node-RED utilizando FlowFuse

Requisitos Previos

Para utilizar Node-RED a través de FlowFuse, se necesitan los siguientes elementos:

1. **Docker:** FlowFuse se despliega como un contenedor Docker, por lo que es necesario tener Docker instalado en tu sistema. Puedes instalar Docker siguiendo las instrucciones en Docker Documentation.
2. **Docker Compose:** Es recomendable usar Docker Compose para gestionar los contenedores de FlowFuse. Puedes instalarlo siguiendo las instrucciones en Docker Compose Documentation.
3. **Acceso a Internet**: Para descargar paquetes y dependencias necesarios.

Pasos de Instalación

Aunque es posible instalar Node-RED como un servicio independiente, en este TFM se ha optado por utilizar FlowFuse, una plataforma que permite la gestión de múltiples instancias de Node-RED. FlowFuse ofrece funcionalidades avanzadas como la colaboración en tiempo real, la gestión de usuarios y la integración continua, lo que es especialmente útil para equipos de desarrollo y científicos de datos que requieren un entorno flexible y escalable para el desarrollo colaborativo. A continuación, se detallan los pasos para la instalación y configuración de FlowFuse, que se utilizará para orquestar y gestionar las instancias de Node-RED en este proyecto.

1. **Clonar el Repositorio de FlowFuse:**
    - Se descarga e instala Node.js desde nodejs.org, lo que también instalará npm.

2. **Despliegue y Configuración de FlowFuse:**
    - FlowFuse se despliega y configura mediante Docker Compose, lo que permite un despliegue sencillo y rápido. Para iniciar FlowFuse, ejecuta:

```
git clone https://github.com/flowforge/flowforge.git
cd flowforge
```

3. **Despliegue y Configuración de FlowFuse:**
    - FlowFuse se despliega y configura mediante Docker Compose, lo que permite un despliegue sencillo y rápido. Para iniciar FlowFuse, ejecuta:

```
docker-compose up -d
```

Este comando descarga las imágenes necesarias y levantará los contenedores de FlowFuse en segundo plano.

Nota sobre la Configuración: FlowFuse permite una amplia gama de configuraciones adaptables según las necesidades específicas del entorno. Aunque este TFM no detalla configuraciones particulares debido a la naturaleza específica del despliegue utilizado, FlowFuse ofrece la flexibilidad de ajustar parámetros como puertos, bases de datos y autenticación según sea necesario para cada caso. Existen configuraciones básicas en la página oficial que permite incluso desplegar FlowFuse en local como un servicio.

4. Acceso a la Interfaz Web:

- Una vez que los contenedores estén en funcionamiento, se accede a la interfaz web de FlowFuse desde tu navegador utilizando la dirección:

```
http://localhost:3000
```

5. Configuración Inicial:

- En el primer acceso a FlowFuse, se pedirá crear una cuenta de administrador. Como se muestra en la ilustración 20 y 21

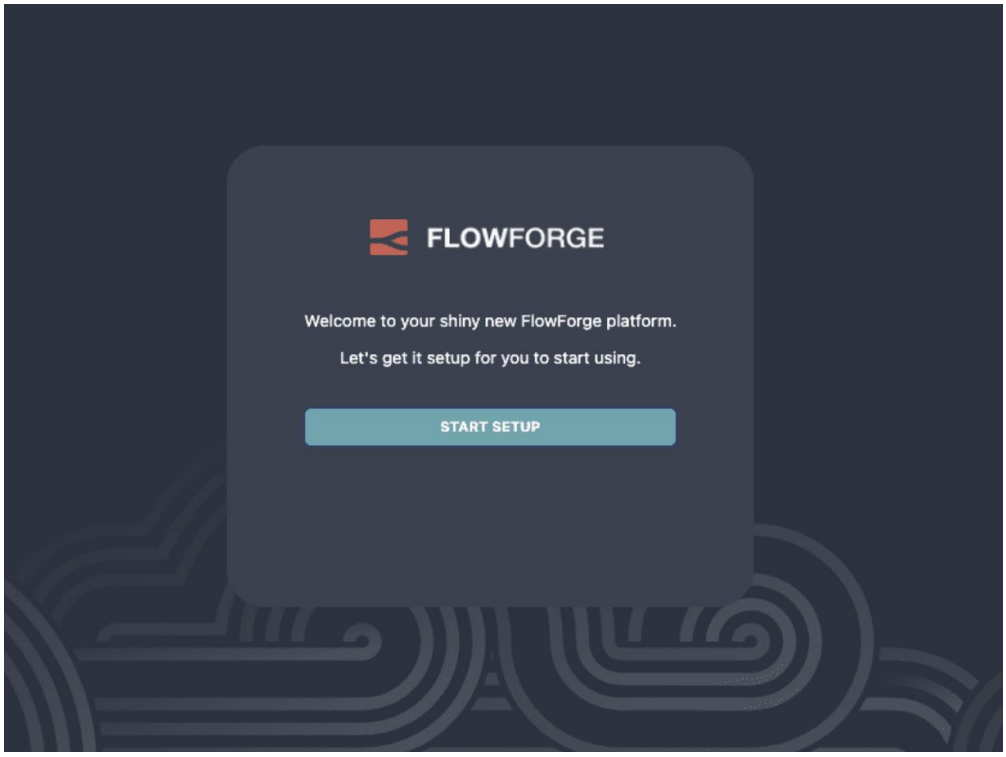

Figura 20. Creando cuenta de administrador 1

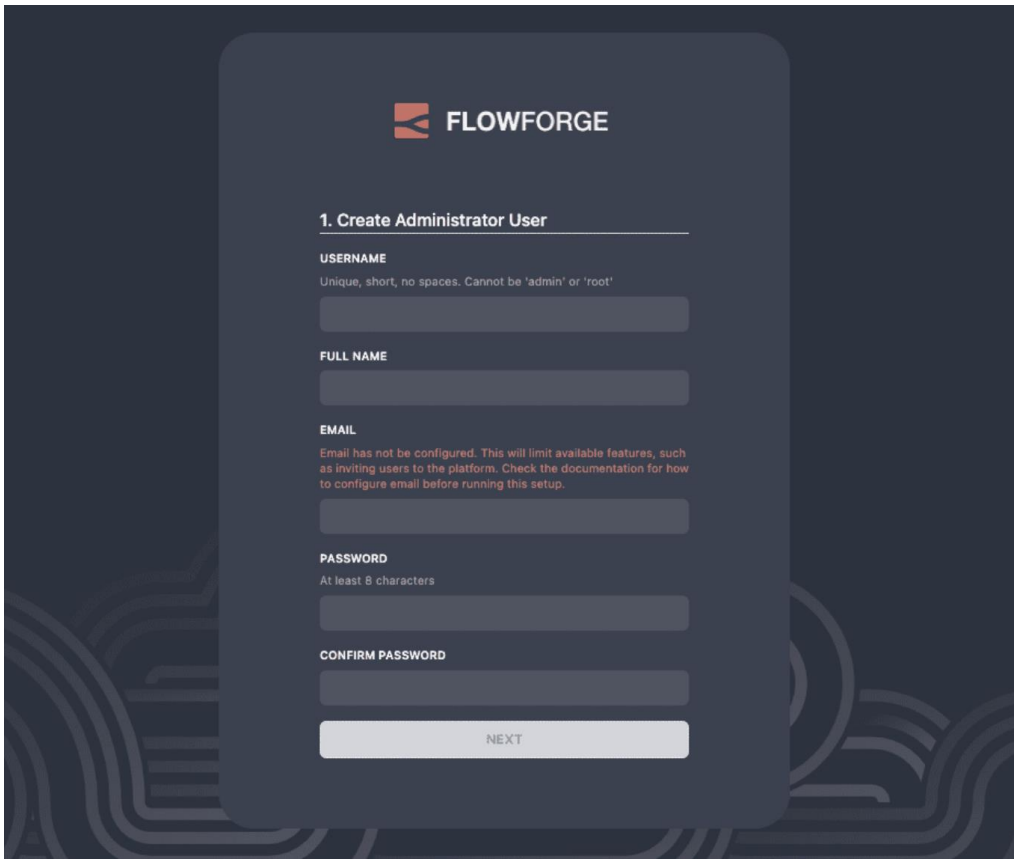

Figura 21. Creando cuenta de administrador 2

6. Creación de Instancias de Node-RED:

Desde la interfaz de FlowFuse, puedes crear y gestionar múltiples instancias de Node-RED. Esto es ideal para entornos colaborativos donde diferentes usuarios o equipos necesitan trabajar en paralelo en distintos proyectos.

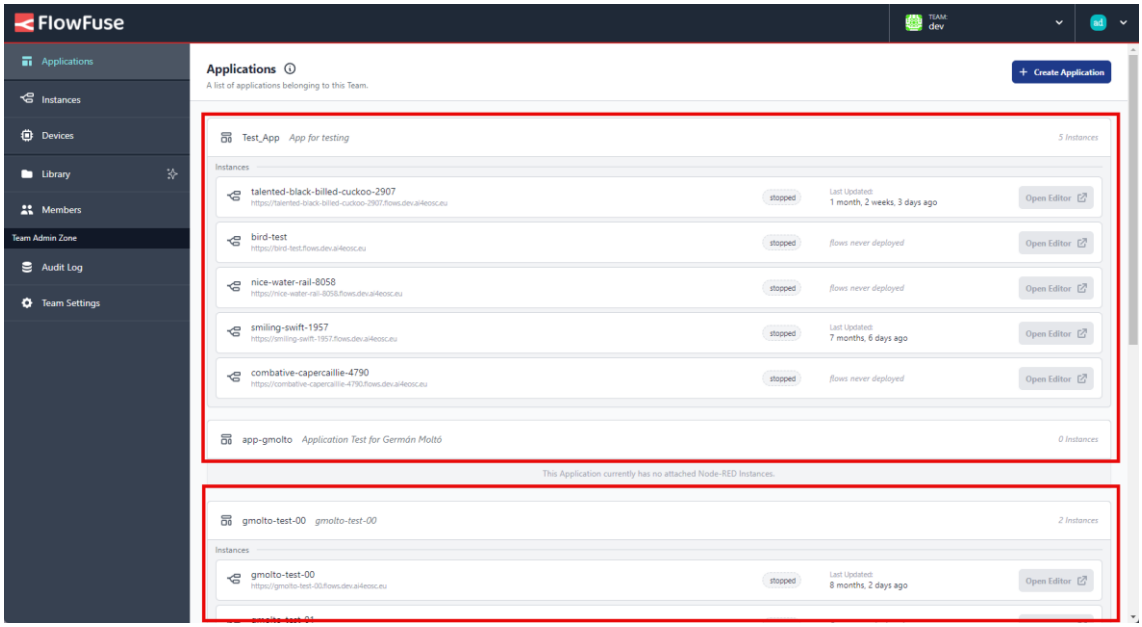

Figura 22. Entorno de FlowFuse

4.3.3 Entorno de Flowfuse

A continuación, se describe brevemente el entorno de FlowFuse, que es la plataforma utilizada para gestionar múltiples instancias de Node-RED en este proyecto. Aunque el entorno de FlowFuse proporciona una infraestructura robusta para la administración y colaboración, el foco principal de este TFM estará en el entorno de Node-RED, ya que es donde se desarrollan y ejecutan los flujos de trabajo clave. Sin embargo, para comprender el contexto en el que operan las instancias de Node-RED, es importante familiarizarse con los elementos fundamentales de FlowFuse. (Figura 22.)

1. **Panel de Navegación Lateral (izquierda):**
    - **Applications:** Este es el acceso principal para gestionar las aplicaciones dentro de FlowFuse. Desde aquí, los usuarios pueden ver todas las aplicaciones disponibles, crear nuevas y administrar las existentes.
    - **Instances:** Permite la gestión de las instancias de Node-RED asociadas a las aplicaciones. Aquí se pueden visualizar, iniciar, detener y configurar las diferentes instancias según sea necesario.

- **Devices:** Este apartado está destinado a la administración de dispositivos conectados, permitiendo al usuario integrar y gestionar hardware adicional que interactúe con las instancias de Node-RED.
- **Library:** Contiene una colección de flujos predefinidos, nodos personalizados y otros recursos reutilizables que pueden ser implementados en las aplicaciones y flujos de trabajo.
- **Members:** Este apartado permite gestionar los miembros del equipo, otorgando permisos y acceso a las distintas funcionalidades de FlowFuse, lo cual es esencial para la colaboración en equipo.
- **Team Admin Zone:** Sección destinada a la administración avanzada del equipo, donde se pueden configurar roles, permisos y acceder a configuraciones más detalladas del entorno.
- **Audit Log:** Proporciona un registro detallado de las acciones realizadas dentro de la plataforma, útil para auditorías y seguimiento de actividades.
- **Team Settings:** Aquí se encuentran las configuraciones generales del equipo, incluyendo aspectos como la personalización del entorno y las políticas de acceso.

2. **Área Principal de Trabajo (centro):**
   - **Lista de Aplicaciones e Instancias:** En el centro de la pantalla se muestra una lista de aplicaciones existentes, junto con sus instancias de Node-RED asociadas. Cada aplicación puede expandirse para mostrar las instancias activas, con información sobre su estado, última actualización y opciones para acceder al editor de flujos de trabajo de Node-RED.
   - **Botón "Create Application":** En la parte superior derecha del área principal, este botón permite al usuario crear nuevas aplicaciones dentro de FlowFuse, iniciando el proceso de configuración de una nueva instancia o flujo de trabajo.
3. **Barra Superior (derecha):**
   - **Selección de Equipo:** En la esquina superior derecha, el usuario puede ver y seleccionar el equipo activo con el que está trabajando. Esta funcionalidad es útil para usuarios que forman parte de múltiples equipos o proyectos dentro de FlowFuse.
   - **Menú de Usuario:** Justo al lado del selector de equipo, se encuentra el menú de usuario, donde se pueden acceder a las configuraciones de la cuenta personal, cerrar sesión, y otros ajustes relacionados con el perfil del usuario.

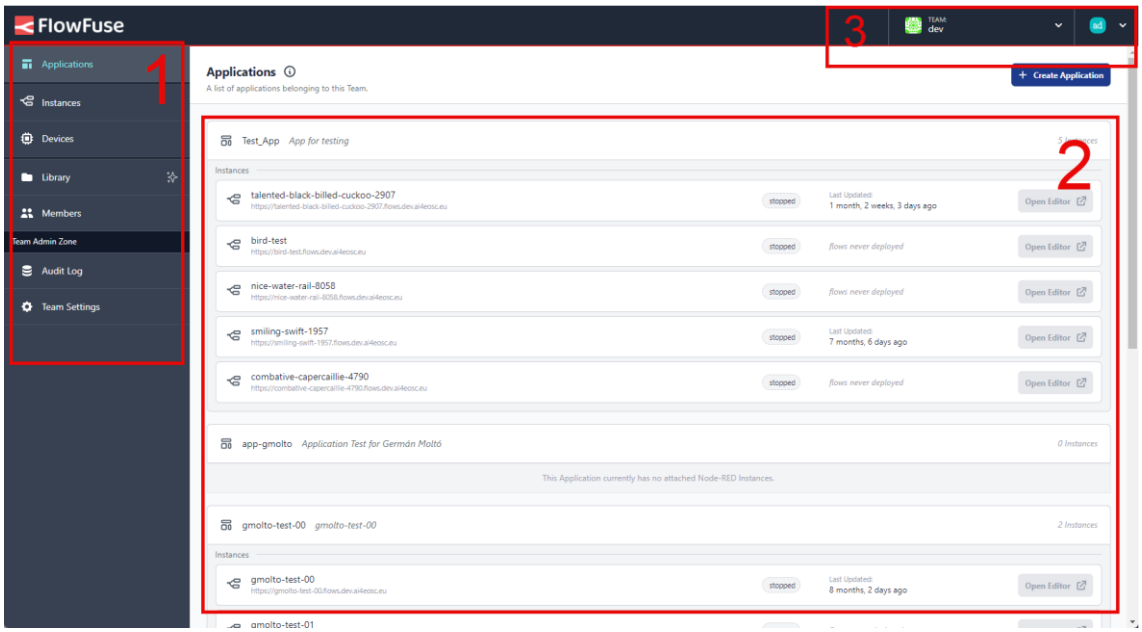

Figura 23. Componente del entorno de FlowFuse

4.3.3 Entorno de Node-RED

Al acceder a la interfaz de Node-RED (figura 24) desde FlowFuse, el usuario se encuentra con un entorno de desarrollo visual que facilita la creación y gestión de flujos de trabajo a través de nodos conectados. La interfaz de Node-RED se organiza en varios componentes clave que permiten una navegación intuitiva y un diseño eficiente de los flujos. A continuación, se describen los elementos principales visibles en la interfaz:

1. **Panel de Nodos (Izquierda):**
    - **Lista de Nodos:** En el lado izquierdo de la pantalla, se encuentra el panel de nodos, que contiene una amplia variedad de nodos organizados en diferentes categorías. Estos nodos son los bloques de construcción que se utilizan para crear los flujos. Algunos de los nodos más comunes incluyen nodos de entrada, procesamiento, y salida, como *switch, slider, gauge, y chart*.
    - **Nodos Personalizados (MinIO y OSCAR):** En la parte inferior del panel, se pueden ver nodos personalizados específicos para las integraciones utilizadas en este proyecto, como los nodos MinIO, que permiten interactuar con el almacenamiento de objetos, y los nodos OSCAR, que son específicos para ejecutar funciones y flujos relacionados con la clasificación de plantas y otros servicios de OSCAR. Estos nodos personalizados son fundamentales para la implementación de funcionalidades específicas dentro del TFM.
2. **Área de Trabajo (Centro):**
    - **Lienzo de Flujos:** El área central de la interfaz es el lienzo donde se diseñan y conectan los flujos. Aquí, los usuarios arrastran y

sueltan los nodos desde el panel de la izquierda y los conectan mediante cables para definir cómo los datos fluyen de un nodo a otro. Este lienzo permite una visualización clara y organizada de las secuencias lógicas que componen cada flujo de trabajo.

- **Flujo Actual:** En la parte superior del lienzo, se indica el nombre del flujo actualmente abierto, en este caso, *Flow 1*. Los flujos se pueden organizar en diferentes pestañas para gestionar múltiples procesos o proyectos dentro de la misma instancia.

3. **Panel de Información y Configuración (Derecha):**
   - **Pestañas de Información y Configuración:** En la parte derecha de la interfaz, se encuentra un panel que permite acceder a varias funciones cruciales:
     - **Info:** Muestra detalles sobre el nodo seleccionado, incluyendo su tipo, configuración actual y opciones de personalización.
     - **Debug:** Proporciona una vista en tiempo real de los mensajes y datos que pasan a través de los nodos durante la ejecución del flujo, lo que es esencial para la depuración y prueba de los flujos.
     - **Deploy:** En la parte superior derecha, *el botón Deploy* permite desplegar los cambios realizados en los flujos. Este botón es esencial para aplicar las modificaciones y hacer que los flujos estén activos y funcionales.
   - **Gestión de Flujos y Subflujos:** El panel también permite gestionar los diferentes flujos y subflujos dentro de la instancia de Node-RED. Esto incluye la posibilidad de agregar, eliminar o modificar flujos y subflujos, así como gestionar nodos de configuración global.

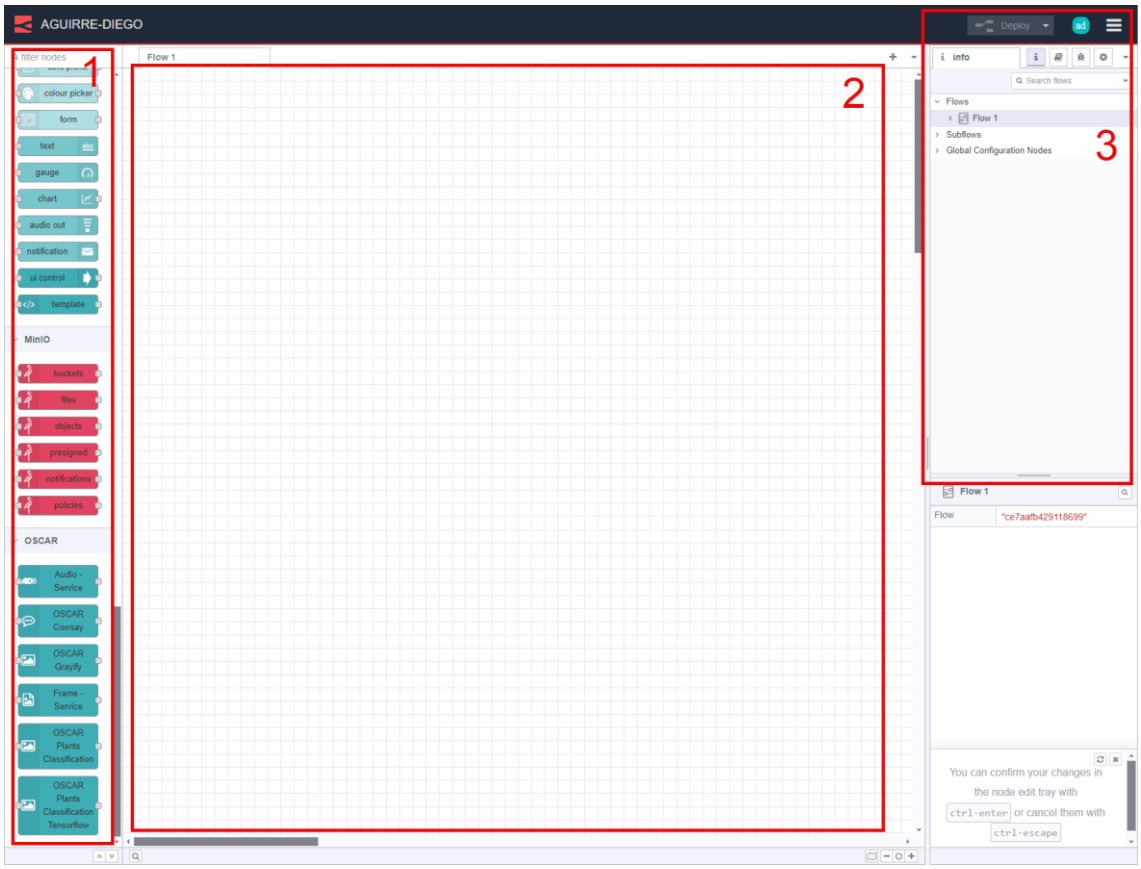

Figura 24. Área de trabajo de Node-RED

4.3.4 Elementos de un nodo en Node-RED

En Node-RED, los nodos son los bloques fundamentales que se utilizan para construir flujos de trabajo. Cada nodo tiene características y funcionalidades específicas que permiten procesar datos, interactuar con servicios externos o realizar tareas lógicas. A continuación, se describen los elementos principales de un nodo en Node-RED:

1. **Entrada y Salida:**
    - **Entrada:** La entrada de un nodo se representa por un pequeño punto en el lado izquierdo del nodo en el lienzo de flujo. Esta entrada permite que el nodo reciba mensajes de otros nodos conectados. Cada mensaje contiene datos que pueden ser procesados por el nodo según su configuración y funcionalidad.
    - **Salida:** Similar a la entrada, la salida de un nodo se representa por un punto en el lado derecho del nodo. Los nodos pueden tener una o varias salidas, dependiendo de su función. La salida permite enviar los resultados procesados a otros nodos dentro del flujo. Esto establece la secuencia de operaciones y permite que los datos fluyan a través de diferentes nodos en el flujo de trabajo.
2. **Opciones de Configuración:**

- **Nombre del Nodo:** Cada nodo puede ser asignado un nombre personalizado para facilitar la identificación y organización dentro del flujo. Asignar nombres descriptivos a los nodos ayuda a entender mejor la función de cada nodo y a documentar el flujo de manera clara.
- **Datos de Entrada:** Los nodos pueden configurarse para aceptar ciertos tipos de datos de entrada. Esta configuración puede incluir definir qué propiedades del mensaje se utilizarán, establecer parámetros específicos o incluso filtrar los datos antes de ser procesados.
- **Metadatos:** Además de los datos de entrada, los nodos pueden tener metadatos asociados, como etiquetas o descripciones adicionales que proporcionen más contexto sobre su función o cómo se deben utilizar dentro del flujo. Los metadatos son útiles para documentar el propósito y las características del nodo.
- **Apariencia del Nodo:** Node-RED permite personalizar la apariencia de los nodos para mejorar la visualización y organización de los flujos. Se pueden cambiar colores, íconos y etiquetas visuales, lo cual es útil para diferenciar nodos que realizan funciones similares o para destacar nodos clave en un flujo complejo.

3. **Configuración Específica del Nodo:**
   - Algunos nodos tienen configuraciones avanzadas que se pueden ajustar según las necesidades específicas del flujo de trabajo. Por ejemplo, un nodo *HTTP request* puede requerir la configuración de la URL de destino, métodos de autenticación y encabezados de solicitud. Cada tipo de nodo puede tener sus propios parámetros de configuración, que se ajustan a través de una interfaz de configuración accesible haciendo doble clic en el nodo (Figura 25).

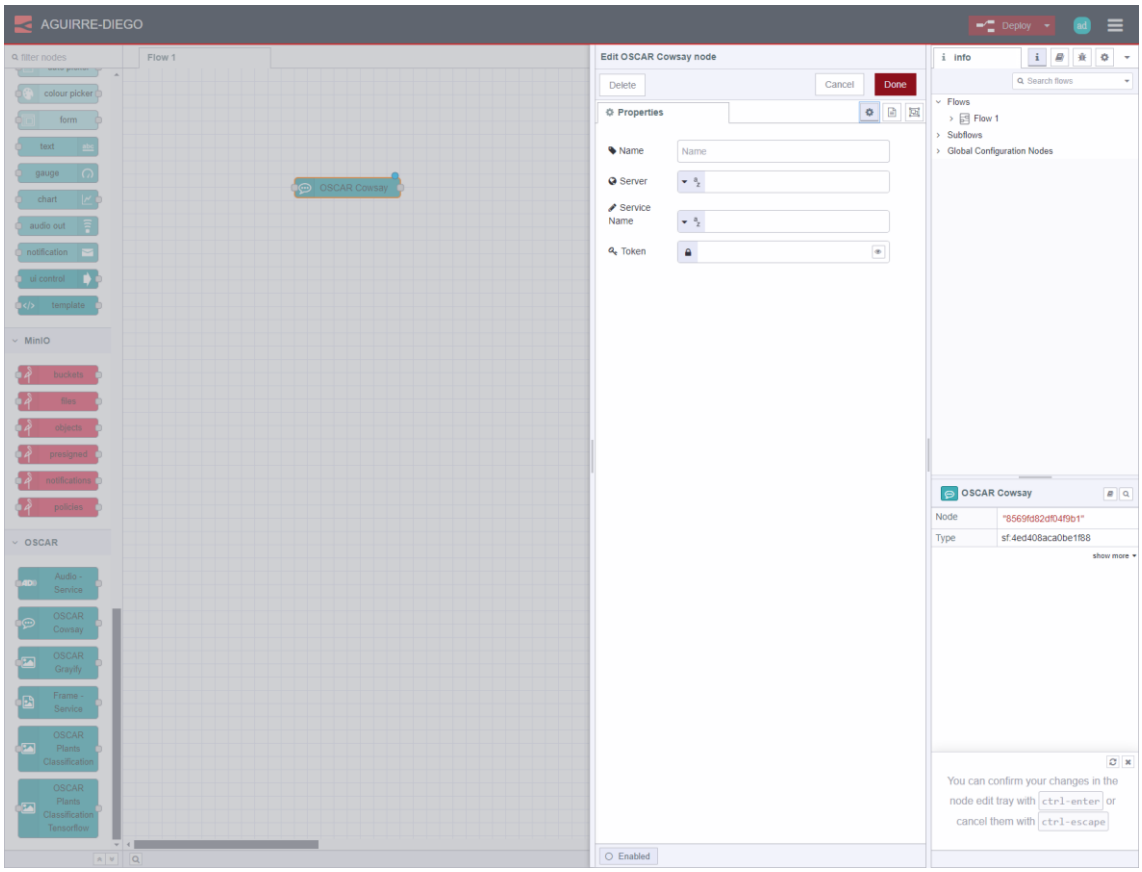

Figura 25. Configuración de un nodo en Node-RED

4.3.5 Uso y configuración de nodos en Node-RED

En Node-RED, la creación de flujos de trabajo se realiza conectando nodos de diferentes tipos para definir el procesamiento de datos. A continuación, se describe un ejemplo básico de uso que incluye los *nodos de inject y debug*, los cuales son fundamentales para iniciar flujos y visualizar resultados.

1. ***Nodo Inject:* Iniciar el Flujo**
    - *El nodo inject* es uno de los nodos más simples y útiles en Node-RED. Su función principal es enviar un mensaje al flujo de trabajo, iniciando así el proceso de ejecución. Este nodo puede configurarse para enviar datos específicos, como cadenas de texto, números, booleanos, o incluso objetos JSON, que luego serán procesados por los nodos subsiguientes en el flujo.
    - **Configuración del Nodo *Inject:*** Al hacer doble clic en el nodo *inject,* se puede abrir la ventana de configuración. Aquí, el usuario puede definir:

        - **Payload:** El contenido del mensaje que se enviará. Puede ser un valor fijo o un dato generado en el momento de la inyección.
        - **Timestamp:** Puede configurarse para enviar un sello de tiempo actual, lo cual es útil para flujos basados en tiempo.
        - **Intervalo de Tiempo:** Permite programar *el nodo inject* para que envíe mensajes de forma automática a intervalos regulares, útil para monitoreo y tareas repetitivas.
2. ***Nodo Debug:* Visualizar Resultados y Logs**
    - *El nodo debug* se utiliza para visualizar los resultados de un flujo en tiempo real. Al conectar *un nodo debug* a la salida de otro nodo, los mensajes que pasan a través de ese nodo se mostrarán en el panel de depuración, ubicado en la parte derecha de la interfaz de Node-RED.
    - **Configuración del *Nodo Debug:*** Al hacer doble clic en *el nodo debug*, se puede ajustar su configuración para:
        - **Output:** Definir qué parte del mensaje se desea visualizar (por ejemplo, *el payload* completo o una propiedad específica del mensaje).
        - **Name:** Asignar un nombre al nodo para identificar fácilmente la salida en el panel de depuración.
3. **Conectando Nodos para Crear un Flujo**
    - Para construir un flujo básico, se conecta la salida del *nodo inject* a la entrada del nodo que se desea probar o utilizar. Luego, se conecta la salida de este nodo *al nodo debug*. Esta configuración simple permite iniciar el flujo enviando un mensaje desde *el nodo inject*, procesar ese mensaje a través de uno o más nodos intermedios, y finalmente visualizar el resultado en *el nodo debug*.
    - Este tipo de flujo es extremadamente útil para la depuración, ya que el *panel de debug* muestra los mensajes en tiempo real. Si hay errores o fallos en el flujo, los mensajes de depuración proporcionan información detallada sobre qué nodos fueron afectados y qué datos se estaban procesando en ese momento.
4. **Uso de Logs para Detección de Fallos**
    - *El nodo debug* no solo muestra los resultados exitosos de los flujos, sino que también es una herramienta crucial para la detección y resolución de problemas. Al revisar los mensajes de depuración, se pueden identificar errores en la lógica del flujo, datos inesperados, o problemas de conectividad. Además, es posible configurar múltiples *nodos debug* en diferentes puntos del flujo para rastrear cómo se transforman los datos a medida que pasan por distintos nodos.
5. **Expandiendo la Funcionalidad**

- Aunque este ejemplo básico se centra en *los nodos inject y debug*, Node-RED permite la integración de muchos otros nodos que pueden realizar operaciones más complejas, interactuar con servicios externos, manejar datos en tiempo real, y mucho más. La combinación adecuada de nodos permite crear flujos de trabajo sofisticados que pueden automatizar tareas, procesar grandes volúmenes de datos, y facilitar la integración entre diferentes sistemas y aplicaciones.

Este enfoque de uso básico con *los nodos inject y debug* es ideal para aprender los fundamentos de Node-RED y establecer un flujo de trabajo inicial. A partir de aquí, se puede ir añadiendo complejidad y funcionalidad para satisfacer las necesidades específicas del proyecto, utilizando la vasta colección de nodos disponibles en Node-RED.

4.3.6 Creación de Nodos Personalizados en Node-RED mediante FlowFuse

En esta sección, se explicará cómo crear nodos personalizados en Node-RED utilizando subflujos y cómo estos pueden ser gestionados y distribuidos mediante FlowFuse. Esta técnica facilita la reutilización de flujos de trabajo y la estandarización de componentes en proyectos colaborativos.

Subflujos en Node-RED

Node-RED permite encapsular conjuntos de nodos en un subflujo, creando un bloque reutilizable que puede ser tratado como un nodo individual. Esta funcionalidad es útil para simplificar flujos complejos y para crear componentes estándar que pueden ser utilizados en diferentes partes de un proyecto o en múltiples proyectos.

- **Creación de Subflujos:** Un subflujo se crea seleccionando un grupo de nodos en el editor de Node-RED y agrupándolos en un subflujo. Esta acción convierte el grupo de nodos en un único nodo que puede ser reutilizado. Los subflujos pueden tener entradas y salidas definidas, lo que permite integrar los subflujos con otros nodos en el flujo principal.
- **Configuración del Subflujo:** Una vez creado, el subflujo puede ser configurado para aceptar diferentes parámetros de entrada y proporcionar salidas específicas. Esto se hace definiendo las propiedades del subflujo, lo que permite su personalización y adaptación a diferentes escenarios.

Exportación de Subflujos como Módulos

Una vez que se ha creado y configurado un subflujo, es posible exportarlo como un módulo de Node-RED. Esto implica empaquetar el subflujo para que pueda ser instalado y utilizado por otros usuarios de Node-RED. La exportación de

subflujos como módulos es especialmente útil para compartir funcionalidades específicas con la comunidad de Node-RED o para estandarizar componentes en proyectos grandes.

- **Empaquetado como Módulo:** Node-RED permite empaquetar subflujos en módulos npm, que son los paquetes estándar utilizados para instalar nodos adicionales en Node-RED. Al empaquetar un subflujo como un módulo, se crean archivos de configuración (package.json) que describen el módulo y sus dependencias. Estos archivos aseguran que el subflujo funcione correctamente cuando se instale en otras instancias de Node-RED.
- **Publicación en el Marketplace:** Una vez empaquetado, el módulo puede ser publicado en el marketplace de Node-RED. Esto hace que el subflujo esté disponible para otros desarrolladores, quienes pueden instalarlo directamente desde la interfaz de Node-RED, facilitando su reutilización y la colaboración.

Gestión y Publicación de Subflujos con FlowFuse

FlowFuse simplifica la administración y publicación de subflujos como nodos reutilizables en Node-RED. Permite a los desarrolladores gestionar varias instancias de Node-RED, asegurando que los subflujos y nodos personalizados estén siempre actualizados y disponibles para los equipos.

- **Integración de FlowFuse:** Al integrar FlowFuse con Node-RED, los subflujos creados pueden ser gestionados y publicados de manera centralizada. FlowFuse proporciona una interfaz para administrar subflujos y nodos, facilitando su distribución y mantenimiento.
- **Ventajas de Usar FlowFuse:**
    - **Colaboración Mejorada:** FlowFuse permite a los equipos colaborar en tiempo real, compartiendo y editando subflujos en un entorno controlado.
    - **Control de Versiones:** Con FlowFuse, es posible mantener un control de versiones de los subflujos y nodos personalizados, asegurando que los cambios se realicen de manera controlada y que se puedan revertir si es necesario.
    - **Publicación Sencilla:** FlowFuse facilita la publicación de subflujos empaquetados en el marketplace de Node-RED o en repositorios internos, simplificando el proceso de distribución.

4.3.7 Publicación de Nodos Personalizados en el Marketplace de Node-RED

Después de crear y empaquetar un subflujo como un módulo reutilizable en Node-RED, el siguiente paso es publicar este módulo para que esté disponible

para otros usuarios. La publicación en el Marketplace de Node-RED permite que los nodos personalizados sean fácilmente instalables y utilizables por la comunidad de Node-RED, lo que fomenta la colaboración y la reutilización de componentes.

Preparación del Módulo para Publicación:

- **Empaquetado del Subflujo:** Una vez creado el subflujo y configurado como un módulo, se debe empaquetar siguiendo las convenciones de npm (Node Package Manager). Esto implica crear un archivo package.json que contiene metadatos sobre el módulo, como su nombre, versión, descripción, autor, y dependencias necesarias. Este archivo es fundamental para la gestión y distribución del módulo.

```
{
  "name": "node-red-example-subflow",
  ...
  "node-red": {
    "nodes": {
      "example-node": "cowsay.js"
    },
    "dependencies": [
      "node-red-node-random"
    ]
  },
  "dependencies": {
    "node-red-node-random": "1.2.3"
  }
}
```

Estructura del Directorio: El directorio del módulo debe contener los archivos necesarios, incluyendo package.json, el archivo principal del nodo (por ejemplo, cowsay.js), y cualquier otro recurso adicional (como íconos o archivos de estilo).

Publicación en el Registro de npm:

- **Creación de una Cuenta en npm:** Para publicar el módulo, primero debemos tener creada previamente una cuenta en npm. Esto se puede hacer desde la página oficial de npm[5].

Inicio de Sesión en npm desde la Terminal: En una terminal se debe ejecutar el siguiente comando para iniciar sesión en npm:

```
npm login
```

- Tras ello se debe introducir un nombre de usuario, contraseña y correo electrónico asociados con una cuenta de npm.

Publicación del Módulo: Una vez iniciada la sesión, se debe navegar al directorio donde se encuentra el módulo y ejecutar el comando para publicar:

```
npm publish
```

- Este comando subirá el módulo al registro de npm, haciendo que esté disponible para su instalación a través de Node-RED y el marketplace de npm.
- **Verificación en el Marketplace de Node-RED:**
  - Tras publicar el módulo, verificamos su disponibilidad en el catálogo de nodos de Node-RED. Los usuarios de Node-RED pueden buscar nuestro nodo utilizando las palabras clave definidas en package.json y pueden instalarlo directamente desde la interfaz de Node-RED.
  - **Revisión de Listado:** Se revisa el listado del nodo en el Marketplace para asegurarnos de que toda la información se muestra correctamente y que los enlaces proporcionados, como el repositorio de GitHub o la página de documentación, funcionan adecuadamente.
- **Mantenimiento y Actualización:**
  - **Actualización de Versiones:** A medida que realizamos mejoras o correcciones en nuestro nodo, actualizamos la versión del módulo modificando el número de versión en package.json y publicamos nuevamente el módulo usando npm publish. Esto permite que las

[5] Página oficial de NPM: https://www.npmjs.com/

actualizaciones se reflejen en el Marketplace, y los usuarios puedan acceder a la versión más reciente.
    - **Gestión de Feedback y Soporte:** Node-RED cuenta con un repositorio de GitHub[6] y foro[7] para poder ofrecer soporte a los usuarios.

Uso de FlowFuse para Publicación y Distribución

FlowFuse ha facilitado la gestión de subflujos y su publicación como nodos reutilizables en Node-RED. A través de FlowFuse, hemos podido administrar múltiples instancias de Node-RED, asegurando que los subflujos y nodos personalizados estén actualizados y disponibles para los equipos de trabajo.

4.3.8 Pruebas y Validación

Para validar la integración y funcionalidad de los flujos en Node-RED utilizando OSCAR, se llevará a cabo una prueba utilizando un servicio de ejemplo conocido como *Cowsay (figura 26)*. A diferencia de las pruebas realizadas con Elyra, donde se utilizaba una librería de Python para crear un cliente y realizar las peticiones, en este caso, las peticiones se realizarán directamente a través de métodos HTTP estándar (como POST y GET), que son métodos comunes para la comunicación con servicios web. Estos métodos permiten enviar y recibir datos de manera estructurada entre diferentes servicios, en este caso, para interactuar con un servicio en un clúster.

Proceso de Prueba con el Servicio Cowsay

1. **Configuración del Nodo *Cowsay* de OSCAR:**
    - Para comenzar, se utilizará el nodo *Cowsay* proporcionado por OSCAR, que está preconfigurado para enviar peticiones al servicio *Cowsay* en un clúster de inferencia. El clúster OSCAR específico utilizado para esta prueba es el clúster de inferencia desplegado para el proyecto AI4EOSC y ubicado en: https://inference.cloud.ai4eosc.eu.
2. **Datos de Configuración:**
    - En el nodo *Cowsay*, se deben preparar y configurar los datos que se enviarán con la petición. Estos datos son esenciales para establecer una comunicación correcta con el servicio y obtener una respuesta válida:
        - **Endpoint del Servidor de OSCAR:** Este es el punto de entrada para las peticiones HTTP al clúster de inferencia.

[6] GitHub de Node-RED: https://github.com/node-red
[7] Foro de Node-RED: https://discourse.nodered.org/

- **Nombre del Servicio:** El servicio específico de *Cowsay* en el clúster debe ser identificado. Para esta prueba, el nombre del servicio es C*owsay*. Este nombre debe ser configurado en el nodo para que la petición sea dirigida al servicio correcto.
- **Token del Servicio:** Este token del servicio OSCAR contiene el modelo de inferencia y es necesario para poder interactuar con el servicio de manera externa. Este debe ser proporcionado y configurado en el nodo *Cowsay* para que la petición sea autorizada por el servidor de OSCAR.

3. **Realización de la Petición:**
   - Una vez configurados los parámetros necesarios, el nodo *Cowsay* puede ser activado para realizar una petición HTTP. En este flujo, se utilizará *un nodo inject* para iniciar la petición y enviar un mensaje de texto, el cual será el contenido que el servicio *Cowsay* procesará.
   - El flujo se completa conectando la salida del nodo *Cowsay* a *un nodo debug*, permitiendo así ver la respuesta del servicio en el panel de depuración de Node-RED. El servicio debería devolver el mismo texto introducido, pero estilizado con el formato característico de *Cowsay*.
4. **Validación del Resultado:**
   - El éxito de la prueba se valida observando la salida en *el nodo debug*. Si el servicio *Cowsay* devuelve correctamente el texto estilizado en respuesta a la petición, se considera que la integración y configuración del nodo de OSCAR en Node-RED han sido exitosas.
   - Este flujo de prueba también sirve para verificar que la comunicación con el clúster de inferencia se realiza correctamente y que los datos necesarios (endpoint, nombre del servicio y token) se manejan adecuadamente para interactuar con los servicios de OSCAR.

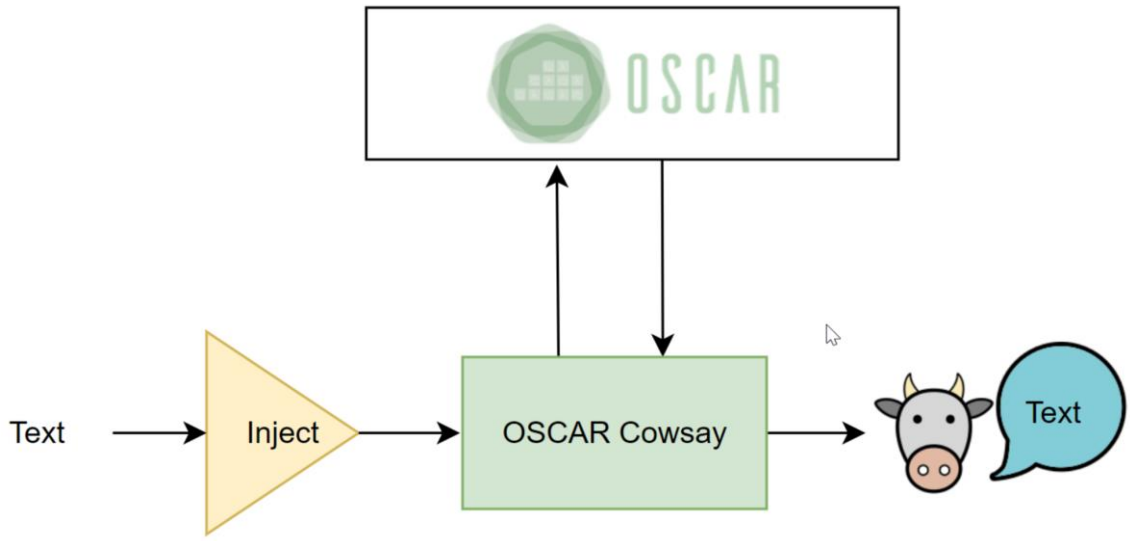

Figura 26. Arquitectura de Cowsay en Node-RED

Finalmente, como se puede ver en la figura 27, ya tenemos montado el flujo de trabajo, solo es necesario darle a iniciar al nodo inject que contiene el string que se enviará al nodo de Cowsay. De esta manera, tras ser ejecutado el proceso podremos ver el resultado en la pestaña de debug.

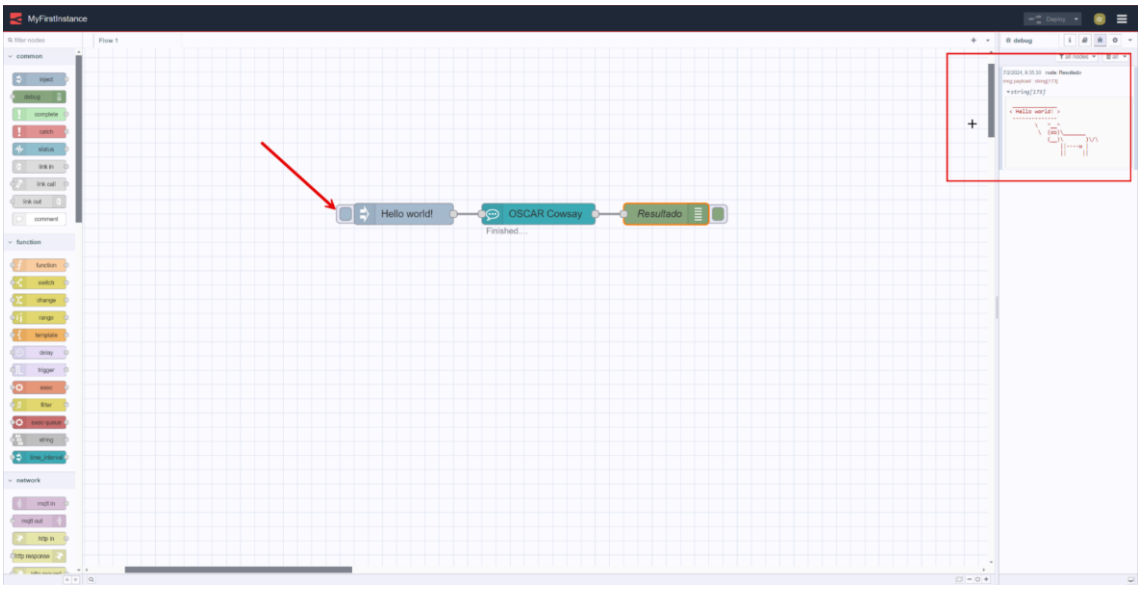

Figura 27. Flujo de trabajo de Cowsay en Node-RED

4.3.9 Clasificación de Plantas en Node-RED

Siguiendo el enfoque de validación usado con Elyra, se realizará una prueba de clasificación de plantas utilizando los nodos de OSCAR en Node-RED. En esta prueba se integrarán varios nodos para realizar tareas específicas, comenzando por la adquisición de una imagen desde internet, su procesamiento para convertirla a blanco y negro, y finalmente la clasificación de la planta presente en

la imagen. El objetivo es demostrar cómo Node-RED puede manejar un flujo de procesamiento de imágenes y clasificación utilizando servicios de OSCAR.

Proceso de Prueba con Clasificación de Plantas y Grayify

1. **Adquisición de la Imagen:**
    - El flujo comienza con *un nodo de tipo http request* configurado para realizar una petición GET. Este nodo se utiliza para obtener una imagen desde una URL pública en internet. La imagen servirá como entrada para los procesos de conversión y clasificación.
    - La URL de la imagen se especifica en la configuración del *nodo http request*. Este nodo descarga la imagen y la envía al siguiente nodo en el flujo.
2. **Conversión de Imagen a Blanco y Negro con *Graify:***
    - Una vez obtenida la imagen, se envía a un nodo de OSCAR llamado OSCAR *Graify*. Este nodo está configurado para recibir la imagen y procesarla, convirtiéndola a blanco y negro.
    - **Configuración del Nodo OSCAR *Graify*:** Para realizar la conversión, el nodo necesita los siguientes datos:
        - **Endpoint del Servidor OSCAR:** Similar a la configuración previa, el endpoint para el clúster de inferencia es https://inference.cloud.ai4eosc.eu.
        - **Nombre del Servicio:** El servicio de *Graify* que se utilizará para la conversión tiene un nombre específico en el clúster de OSCAR. Este nombre se configura en el nodo para asegurar que la petición se dirija al servicio correcto.
        - **Token del Servicio:** Un token de seguridad se utiliza para autenticar la petición y permitir que se realice la conversión.
3. **Clasificación de Plantas:**
    - Después de la conversión a blanco y negro, la imagen procesada se envía a otro nodo de OSCAR, en este caso, *el nodo OSCAR Plant Classification*. Este lanza un servicio de clasificación que funciona en el mismo cluster de OSCAR.
    - **Configuración del Nodo *OSCAR Plant Classification*:** Similar a los pasos anteriores, este nodo requiere:
        - **Endpoint del Servidor OSCAR:** El mismo endpoint del clúster de inferencia.
        - **Nombre del Servicio:** El servicio de clasificación específico para plantas en el clúster de OSCAR. Este nombre debe coincidir con el servicio de clasificación disponible.
        - **Token del Servicio:** Para autenticar y autorizar la petición de clasificación.
4. **Visualización del Resultado:**

- Finalmente, los resultados de la clasificación de plantas se envían a un *nodo debug* en Node-RED. Este nodo permite visualizar el resultado en el panel de depuración, mostrando la categoría o clase de planta identificada por el modelo.
- Esta visualización no solo muestra el resultado final de la clasificación, sino que también sirve como una herramienta de validación para verificar que todo el flujo de procesamiento de imágenes y clasificación funciona correctamente.

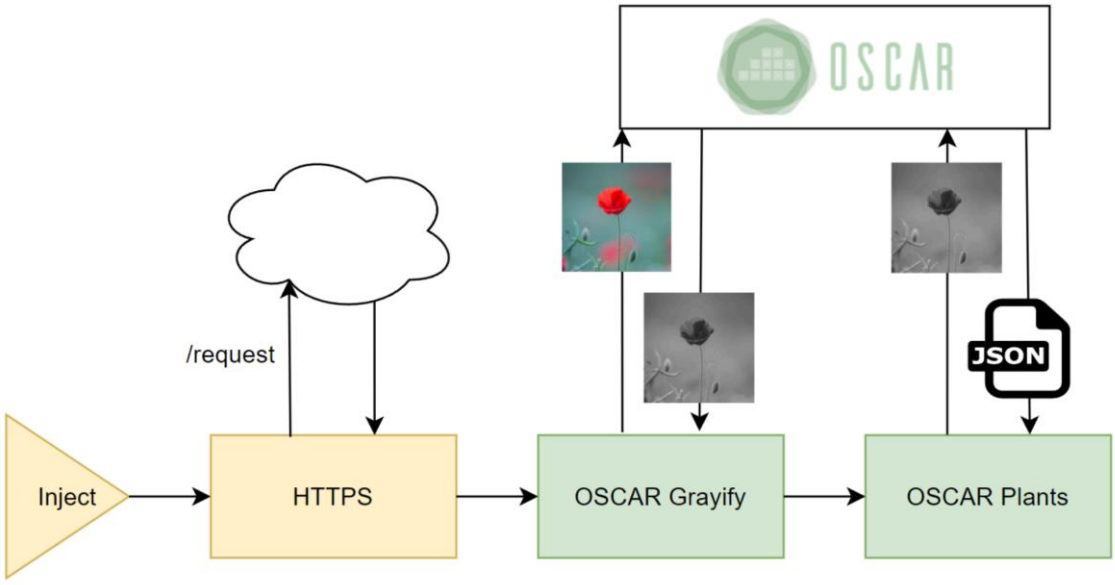

Figura 28. Arquitectura de Clasificación de Plantas en Node-RED

Como se observa en la figura x, tenemos un flujo de trabajo que permite nuevamente mediante el uso del nodo inject iniciamos los procesos para adquirir

una imagen de internet, aplicarle el filtro de grayify para finalmente ser clasificada por el nodo de clasificación de plantas.

En la figura 30 se observa con más claridad el resultado al convertir la visualización del resultado a una cadena de texto.

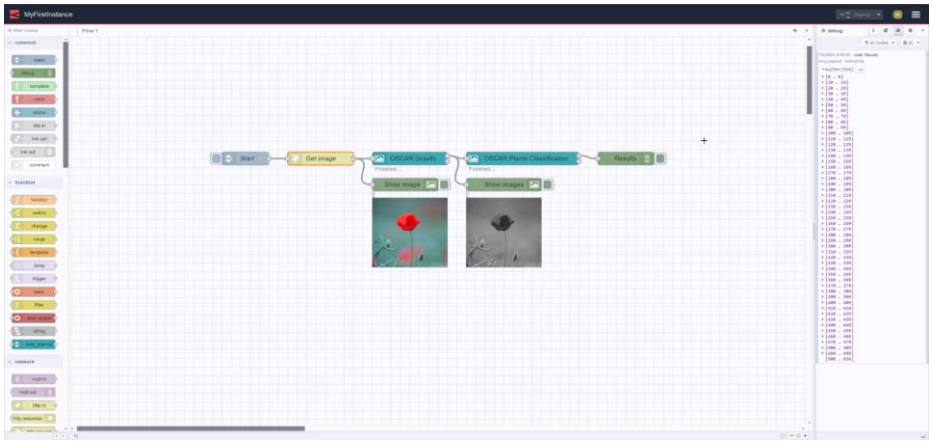

Figura 29. Flujo de trabajo de Clasificación de Plantas en Node-RED

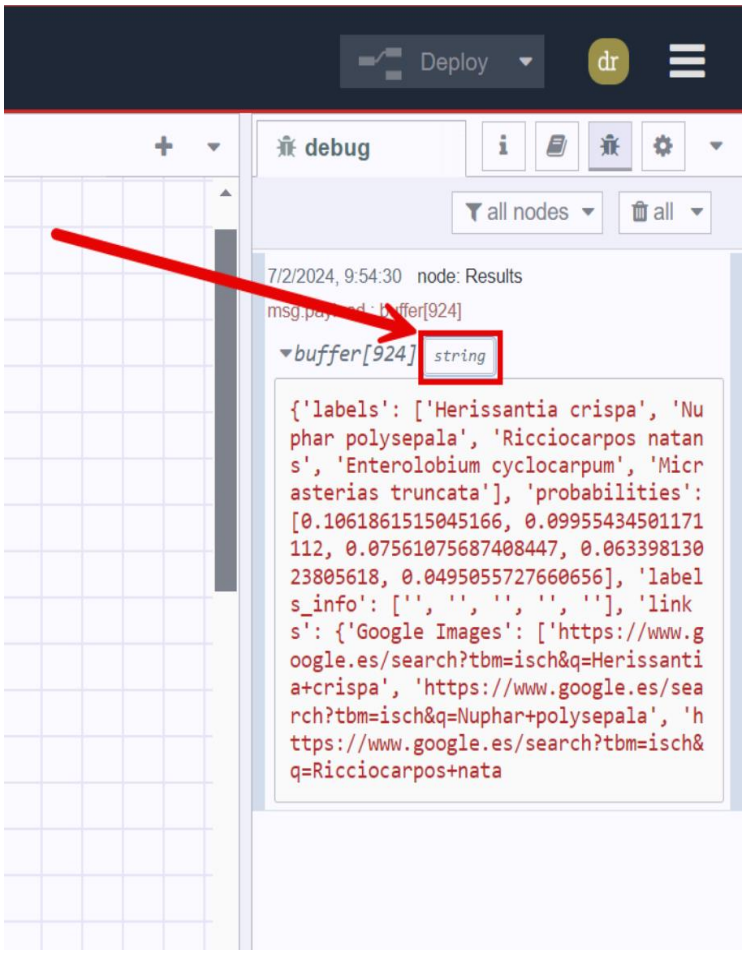

Figura 30. Resultado en cadena de texto.

4.3.10 Consideraciones Finales en Node-RED

A lo largo de este proyecto, se ha demostrado cómo Node-RED, en combinación con los servicios de OSCAR, puede utilizarse para diseñar y ejecutar flujos de trabajo de inferencia complejos de manera efectiva y eficiente. Utilizando nodos básicos y personalizados, hemos implementado flujos de procesamiento que integran adquisición de datos, transformación, y clasificación de información en tiempo real.

Las pruebas realizadas con servicios de ejemplo como *Cowsay* y los flujos de procesamiento de imágenes para la clasificación de plantas han validado la capacidad de Node-RED para interactuar con servicios de inferencia a través de peticiones HTTP estándar. Esta aproximación permite a los desarrolladores y científicos de datos trabajar en un entorno visual que simplifica la configuración, el despliegue, y la supervisión de tareas de inferencia, sin necesidad de profundos conocimientos de programación.

5. Resultados

En este apartado se presentan los principales resultados obtenidos a lo largo del desarrollo de este proyecto, destacando las implementaciones y colaboraciones realizadas, así como las metodologías utilizadas para integrar herramientas y servicios dentro del ecosistema de AI4EOSC.

5.1 Contribución a AI4EOSC

Uno de los resultados más destacados de este TFM es la participación en el proyecto AI4EOSC, que ha permitido la creación de un repositorio con ejemplos de uso y nodos personalizados basados en los propios ejemplos de OSCAR. Esto garantiza que todos los ejemplos presentados en el documento, así como futuros casos y nodos, puedan ser replicados y utilizados por cualquier persona interesada en el proyecto.

5.2 Creación de Nodos OSCAR en Node-RED mediante Subflujos

Otro resultado clave de este proyecto fue la creación de nodos personalizados de OSCAR dentro de la librería de Node-RED[8] a partir de subflujos. De esta manera personas ajenas al proyecto pueden descargarlas y utilizarlas a través de la plataforma, consiguiendo así un mayor alcance de usuarios finales. En la actualidad hay un total de 8 subflujos.

5.3 Colaboración con EGI Notebooks para Integración de Elyra

Se llevó a cabo una colaboración exitosa con EGI Notebooks para agregar el soporte de Elyra a su plataforma que proporciona acceso a entornos de Jupyter Notebooks. Para ello se modificó el Dockerfile que instala todos los componentes de Jupyter Lab, incluyendo los notebooks y otras herramientas de análisis de datos.

Implementación de Elyra: La integración de Elyra en EGI Notebooks facilitó la ejecución de flujos de trabajo complejos en un entorno colaborativo y accesible, mejorando la capacidad de los científicos de datos para diseñar, probar y desplegar modelos de machine learning.

[8] Librería de Node-RED: https://flows.nodered.org/

- **Beneficios de la colaboración:** Esta colaboración permite a los usuarios de EGI Notebooks hacer uso de Elyra para realizar diferentes tareas de Inferencia.

5.4 Despliegue del Servicio de FlowFuse

El despliegue de FlowFuse fue otro logro significativo del proyecto, permitiendo el uso de Node-RED tanto de forma pública como en entornos controlados como parte de los proyectos de AI4EOSC.

- **Acceso Público y Controlado:** El servicio de FlowFuse se configuró para permitir acceso tanto a usuarios públicos como a miembros específicos de proyectos de AI4EOSC. Esto proporciona flexibilidad en el uso de Node-RED, permitiendo a los equipos colaborar en proyectos compartidos o desarrollar de manera independiente.
- **Integración en AI4EOSC:** La integración de FlowFuse dentro del proyecto de AI4EOSC facilita la administración y escalabilidad de los servicios de Node-RED, asegurando un entorno robusto para el desarrollo de aplicaciones de inferencia y procesamiento de datos.

6. Discusión

6.1 Eficacia de las Herramientas Utilizadas

Uno de los aspectos clave de este proyecto fue la elección de las herramientas: Node-RED, Elyra, OSCAR, y FlowFuse. Cada una de estas herramientas aportó funcionalidades únicas al proyecto, permitiendo una integración fluida de servicios de inferencia y facilitando la gestión de flujos de trabajo.

- **Node-RED:** Su interfaz visual de arrastrar y soltar demostró ser una herramienta efectiva para construir y gestionar flujos de trabajo. Por otro lado, el hecho de tener un *market place* y foro permite tener acceso a herramientas y feedback de su comunidad.
- **Elyra:** Integrar Elyra con EGI Notebooks proporcionó un entorno potente para diseñar y gestionar pipelines de machine learning, mejorando la reproducibilidad y escalabilidad de los experimentos. Por otro lado, gracias a ser integrado en EGI Notebooks permite aumentar su alcance a usuarios dentro de la comunidad científica.
- **OSCAR:** Los nodos de OSCAR proporcionaron una funcionalidad robusta para la inferencia, demostrando la eficacia de integrar servicios basados en clústeres en flujos de trabajo visuales.
- **FlowFuse:** Esta herramienta permitió la gestión centralizada de múltiples instancias de Node-RED, facilitando la colaboración en tiempo real y la administración de proyectos en un entorno compartido. La implementación de FlowFuse demostró ser una solución efectiva para gestionar proyectos que requiera *pipelines* de Node-RED en AI4EOSC.

6.2 Desafíos Enfrentados

Durante el desarrollo del proyecto, se encontraron varios desafíos que proporcionan lecciones valiosas para proyectos futuros:

- **Gestión de Dependencias:** La instalación y gestión de dependencias, especialmente en Node-RED, presentó algunos desafíos. Asegurar la compatibilidad entre nodos y versiones de Node-RED es crucial para evitar problemas de compatibilidad. Automatizar la gestión de dependencias mediante scripts de configuración o contenedores podría simplificar el despliegue en diferentes entornos.
- **Colaboración y Control de Versiones:** Aunque GitHub se utilizó para el control de versiones y almacenamiento de flujos y notebooks, integrar de manera efectiva el trabajo colaborativo en tiempo real sigue siendo un desafío. Herramientas como FlowFuse y la integración de Elyra en EGI Notebooks ayudaron a mitigar este problema, pero la coordinación y

sincronización en proyectos con múltiples colaboradores puede beneficiarse de soluciones adicionales de gestión de proyectos y comunicación.

6.3 Impacto y Aplicaciones Futuras

Los resultados de este proyecto no solo validan la viabilidad de utilizar Node-RED, Elyra, OSCAR y FlowFuse en proyectos de inteligencia artificial, sino que también abren la puerta a nuevas aplicaciones y mejoras:

- **Expansión de Nodos Personalizados:** La creación de más nodos personalizados basados en subflujos puede ampliar la funcionalidad de Node-RED, permitiendo integrar una gama más amplia de servicios y modelos de inferencia. Esto puede ser particularmente útil en aplicaciones de IoT y automatización industrial.
- **Mejoras en la Documentación y Tutoriales:** Proporcionar documentación detallada y tutoriales para configurar y utilizar estas herramientas podría facilitar la adopción por parte de nuevos usuarios. Esto incluye no solo instrucciones de configuración, sino también ejemplos de casos de uso y mejores prácticas.
- **Colaboración con Otros Proyectos de IA:** Las capacidades demostradas en este proyecto pueden ser transferidas a otros proyectos de inteligencia artificial y procesamiento de datos dentro de la comunidad de AI4EOSC. Colaborar con otras plataformas y proyectos para integrar estas herramientas podría mejorar la interoperabilidad y potenciar la innovación.

6.4 Consideraciones de Escalabilidad y Mantenimiento

A medida que más usuarios y proyectos adopten estas herramientas, será crucial considerar la escalabilidad y el mantenimiento de la infraestructura:

- **Escalabilidad de FlowFuse y Node-RED:** A medida que crece el número de usuarios y aplicaciones, es importante asegurarse de que FlowFuse y las instancias de Node-RED puedan escalar adecuadamente para manejar la carga de trabajo. Esto podría incluir la optimización de recursos de hardware y la implementación de soluciones de monitoreo y ajuste automático.
- **Mantenimiento de Repositorios:** Mantener actualizados los repositorios de GitHub con los últimos cambios, correcciones de errores y nuevas funcionalidades es esencial para garantizar que las herramientas sigan siendo útiles y seguras. Establecer un ciclo de revisión y actualización periódica podría ayudar a mantener la relevancia y eficacia de los flujos y notebooks compartidos.

7. Conclusión

En este Trabajo Fin de Máster se ha demostrado la integración efectiva de diversas herramientas y tecnologías de código libre, específicamente Node-RED, Elyra, OSCAR y FlowFuse, para la construcción de flujos de trabajo de inferencia en inteligencia artificial. A través de la implementación de casos de prueba y la creación de nodos personalizados, se ha validado la capacidad de estas herramientas para facilitar la construcción, gestión y ejecución de flujos de trabajo complejos, al tiempo que se promueve la colaboración y la reutilización de componentes en proyectos de ciencia de datos.

Los resultados obtenidos destacan la versatilidad de Node-RED para integrar servicios de inferencia mediante métodos HTTP, permitiendo un enfoque visual y simplificado en la gestión de flujos de datos. La incorporación de Elyra en EGI Notebooks ha demostrado cómo se pueden utilizar herramientas avanzadas de gestión de pipelines de machine learning en entornos colaborativos de Jupyter Notebooks. Además, el uso de FlowFuse ha proporcionado un marco robusto para la gestión centralizada de instancias de Node-RED, mejorando la escalabilidad y el control en entornos de desarrollo compartidos.

Por otro lado, el proyecto ha permitido comprender cómo estas herramientas pueden ser adaptadas y configuradas para satisfacer las necesidades de diferentes proyectos de inteligencia artificial y ciencia de datos. La adopción y correcta integración de estas tecnologías sienta una base sólida para la expansión y el crecimiento de aplicaciones colaborativas y escalables en el ecosistema de AI4EOSC.

8. Trabajo Futuro

Basándose en los logros y conocimientos adquiridos en este proyecto, se identifican varias áreas de trabajo futuro para mejorar la integración y funcionalidad de las herramientas utilizadas, dentro de los límites de no poder modificar el código base:

1. **Creación de Más Ejemplos de Uso:**
    - Desarrollar una biblioteca más extensa de ejemplos de uso tanto para Node-RED como para Elyra, abarcando una variedad de casos de inferencia y procesamiento de datos. Estos ejemplos pueden incluir aplicaciones en áreas como el procesamiento de lenguaje natural, la visión por computadora y la automatización de tareas en IoT. Los ejemplos facilitarán a otros usuarios la implementación de flujos de trabajo complejos y promoverán la adopción de estas herramientas.
2. **Integración Directa de Elyra en OSCAR:**
    - Explorar la posibilidad de integrar Elyra directamente en un entorno Jupyter que pueda ser ejecutado en OSCAR. Esto permitiría a los usuarios diseñar, ejecutar y gestionar pipelines de machine learning dentro de OSCAR, aprovechando las capacidades de escalabilidad y procesamiento distribuido del clúster de OSCAR. Aunque no se puedan modificar directamente Elyra o OSCAR, se pueden explorar configuraciones y scripts de integración que faciliten esta interoperabilidad.
3. **Desarrollo de Nodos Adicionales para Node-RED:**
    - Crear más nodos personalizados basados en subflujos para integrar otros servicios de OSCAR y herramientas de IA. Por ejemplo, nodos para servicios de análisis de sentimiento, detección de objetos en imágenes y procesamiento de grandes volúmenes de datos en tiempo real. Estos nodos facilitarían la integración de capacidades avanzadas de IA en flujos de trabajo de Node-RED y podrían ser compartidos en la comunidad para su reutilización.
4. **Mejoras en la Documentación y Tutoriales:**
    - Aunque no se puedan mejorar directamente las interfaces de usuario de FlowFuse o Node-RED, se puede trabajar en mejorar la documentación y crear tutoriales que faciliten su uso. Esto incluye proporcionar guías detalladas sobre la configuración de los nodos, ejemplos de integración, y mejores prácticas para la gestión de flujos de trabajo colaborativos.

5. **Automatización del Despliegue y Mantenimiento:**

- Implementar scripts de automatización para el despliegue y mantenimiento de instancias de Node-RED y Elyra en entornos de producción. Esta automatización facilitaría la adopción y escalabilidad de estas herramientas en proyectos de gran envergadura, asegurando un despliegue coherente y eficiente sin necesidad de modificar el código base de las herramientas.

6. **Colaboración con Otras Comunidades y Proyectos:**
   - Extender la colaboración con otras plataformas y proyectos de IA y ciencia de datos para integrar estas herramientas en más entornos. Esto podría incluir colaboraciones con iniciativas de open science, centros de investigación y consorcios industriales, para promover la interoperabilidad y la adopción de estas soluciones en un ámbito más amplio.
7. **Estudios de Usabilidad y Retroalimentación de Usuarios:**
   - Realizar estudios de usabilidad para obtener retroalimentación de los usuarios sobre las herramientas y flujos desarrollados. Esta retroalimentación podría usarse para identificar áreas de mejora y desarrollar nuevas funcionalidades que respondan mejor a las necesidades de los usuarios, ajustando configuraciones y flujos de trabajo según los resultados de estos estudios.

9. Referencias

[1] Elyra. (n.d.). Elyra AI Toolkit for Jupyter Notebooks: Pipelines and Extensions. Recuperado de https://elyra.readthedocs.io/en/latest/

[2] NVIDIA. (n.d.). Simplifying AI Inference in Production with Triton. Recuperado de https://developer.nvidia.com/blog/simplifying-ai-inference-in-production-with-triton/

[3] NVIDIA. (n.d.). Deploying AI Deep Learning Models with Triton Inference Server. Recuperado de https://developer.nvidia.com/blog/deploying-ai-deep-learning-models-with-triton-inference-server/

[4] Yahoo! JAPAN. (n.d.). Ad Quality. Recuperado de https://global-marketing.yahoo-net.jp/products/ad_quality

[5] Iberdrola. (n.d.). Machine learning: ¿qué es y cómo revoluciona el mundo? Recuperado de https://www.iberdrola.com/innovacion/machine-learning-aprendizaje-automatico

[6] Elyra AI. (n.d.). elyra. Recuperado de https://github.com/elyra-ai/elyra

[7] IBM, 2019. Inteligencia Artificial: Hoy, Datos, Entrenamiento e Inferencias. *IBM Blog*. https://www.ibm.com/blogs/systems/mx-es/2019/12/inteligencia-artificial-hoy-datos-entrenamiento-e-infrerencias/

[8] Noticias AI. (n.d.). Inferencia en inteligencia artificial: ¿Qué es y por qué es crucial? Recuperado de https://noticias.ai/inferencia-en-inteligencia-artificial-que-es-y-por-que-es-crucial/

[9] Rouse, M. (2021). ¿Qué es la computación serverless y por qué importa? Recuperado de https://searchcloudcomputing.techtarget.com/es/definicion/Que-es-la-computacion-sin-servidor-y-por-que-importa

[10] FlowFuse. (n.d.). Node-RED management with FlowFuse. Recuperado de https://flowfuse.com/docs/

[11] Jupyter. (n.d.). Jupyter Notebooks: A Guide to Collaborative Development. Recuperado de https://jupyter.org/

[12] AI4EOSC. (n.d.). Workplan. Recuperado de https://ai4eosc.eu/about/workplan/

[13] AI4OS. (n.d.). ai4-docs. Recuperado de https://github.com/ai4os/ai4-docs

[14] AI4OS. (n.d.). ai4-compose. Recuperado de https://github.com/ai4os/ai4-compose

[15] EGI Federation. (n.d.). notebooks. Recuperado de https://github.com/EGI-Federation/notebooks

[16] Oscar. (n.d.). Documentación de Oscar. Recuperado de https://docs.oscar.grycap.net/

[17] Galea, A. (2018). *Applied Data Science with Python and Jupyter*. 1st edition. Packt Publishing.

[18] Kanikathottu, H. (2019). *Serverless Programming Cookbook*. 1st edition. Packt Publishing.