A close-up of a logo

Description automatically generated

**Sprint 2**

**Grupo 2: María Carreño Nin de Cardona, Raquel Fernández Esquinas, Teresa X. Garvía Gallego e Isabel V. Morell Maudes**

**Tecnologías de Procesamiento Big Data**

**3º Grado en Ingeniería Matemática e Inteligencia Artificial**

# Índice

[Introducción 3](#_Toc177138896)

[Metodología 4](#_Toc177138897)

[Resultados 5](#_Toc177138898)

[Conclusión 7](#_Toc177138899)

# Introducción

# Este sprint supone la segunda práctica del proyecto final de la asignatura Tecnologías de Procesamiento Big Data sobre las criptomonedas. En este sprint del proyecto, daremos un paso crucial para guardar y leer los metadatos relacionados con los datos históricos almacenados en S3 mediante la creación de un crawler utilizando el servicio de AWS Glue Data Catalog y AWS Glue Crawler.

El objetivo principal de este sprint es desarrollar un script en Python que automatice tanto la creación del catálogo de datos, como la configuración del crawler.

Dado que la seguridad es un pilar fundamental para el cliente, el código subido no debe contener ningún tipo de credencial o información sensible.

Este proceso permitirá mejorar la gobernanza de los datos almacenados en S3, asegurando una gestión eficiente y segura de los metadatos necesarios para el análisis en futuras fases del proyecto.

Adjuntamos el enlace al repositorio de Git donde se encuentra toda la información, ficheros, datos…

sobre este Sprint 2. Toda la información se encuentra en la rama de develop:

https://github.com/tgarviagallego/Proyecto\_BigData.git

# Metodología

Para completar la segunda historia de usuario hemos utilizado Glue de AWS que es un servicio de integración de datos ‘serverless’ que facilita la detección, preparación y la combinación de los datos para analística, machine learning y desarrollo de aplicaciones.

Comenzamos utilizando Glue Data Catalog para almacenar y organizar los metadatos de las tablas que hicimos en el sprint anterior desde la página web. Para ello, hemos creado un rol de glue que permita acceder a todos sus servicios gracias a la política de AWSGlueServiceRole y que nos permitirá leer los datos almacenados en el bucket que creamos en el Sprint 1 gracias a la política de AmazonS3ReadOnlyAccess. A continuación, creamos la base de datos con nombre “trade\_data\_imat3a02” y añadimos una tabla utilizando un crawler que hemos llamado “crawler\_imat02”. Este crawler ha tomado los datos del bucket de s3 “s3://trading-view-data” y los ha almacenado en la tabla de nombre “trade\_data\_trading\_view\_data”.

Después, hemos hecho un script de Python llamado create\_bbdd.py que utiliza la librería boto3 para conseguir la funcionalidad pedida. Comenzamos creando un cliente de glue con boto3.client que recibe el servicio de “glue” y las diferentes credenciales necesarias. Éstas se encuentran en un fichero llamado credenciales.py que tenemos almacenado en local con el fin de no hacerlas públicas. El cliente comienza creando una base de datos con nombre “trade\_data\_imat3a02” y, si salta un error porque está ya creada se maneja el error. Luego, creamos un crawler al que pasamos el enlace donde se encuentra el bucket. Así conseguimos crear la base de datos con las diferentes tablas con los metadatos de los diferentes ficheros del buckets de s3.

# Resultados

En primer lugar, decidimos hacer a mano la creación de la database, el datalog y el crawler en Glue de AWS, siguiendo el tutorial con el fin de afianzarnos más con el tema.

Una vez que ya interiorizamos qué teníamos que hacer, hicimos un programa de Python que nos permitía crear todo lo anterior con solo ejecutarlo, como ya hemos explicado anteriormente.

A la hora de crear el crawler se nos pedía la creación de una política, donde tuvimos que crear un rol y asociárselo a esa política, donde se le permitía al usuario poder hacer cualquier operación de lectura en el bucket. Para la creación de la política también añadimos las de Glue.

Como resultados, podemos observar las siguientes creaciones que hemos hecho.

En primer lugar, vemos como el rol que hemos creado para la creación del crawler está asociado a las 2 políticas mencionadas anteriormente, la de Glue y la de lectura de los buckets de s3.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

La política final sería la que observamos en la imagen de debajo, la de trading-view que usaremos a la hora de crear el crawler.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Una vez que ya tenemos las políticas y el rol definido podemos crear el crawler. Pero antes, creamos el datalog donde se va a almacenar el formato de la estructura de los datos del bucket. (trade\_data\_imat3a02).

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Tras la implementación del código, donde realizamos una conexión al bucket, creamos el datalog y crawler que contienen los metadatos de nuestros datos, la forma con la que se ve en el AWS es:

**Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.**

Dentro de dicha base de datos encontramos el Datalog, donde se almacenan los metadatos de los datos del bucket, donde podemos ver su esquema en la imagen de debajo:

**Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.**

Como podemos observar en las imágenes anteriores, conseguimos obtener todos los requisitos necesarios, con muchas dificultades, pero conseguido. Hemos conseguido que el sistema funcione de una forma correcta.

# Conclusión

Este sprint ha sido un paso clave para la gestión y gobernanza de los datos relacionados con el proyecto de criptomonedas, permitiéndonos aprovechar las capacidades de AWS Glue para organizar y estructurar los metadatos de manera eficiente. A lo largo del proceso, hemos aprendido a utilizar AWS Glue Data Catalog y AWS Glue Crawler para integrar, analizar y gestionar datos almacenados en S3.

La automatización lograda mediante el script en Python desarrollado con la librería boto3 no solo cumple con los requisitos establecidos en la historia de usuario, sino que también establece una base sólida para futuras implementaciones. Este enfoque nos permite optimizar el flujo de trabajo y garantizar que los metadatos estén correctamente organizados y accesibles para análisis futuros.

Además, se han tomado medidas rigurosas para mantener la seguridad de las credenciales, asegurando que el código sea seguro y cumpla con las mejores prácticas. Gracias a estos avances, hemos sentado las bases para un gobierno efectivo de los datos y mejorado la capacidad del equipo para abordar problemas más complejos en las siguientes fases del proyecto.