**REPORTE DE PRUEBA TECNICA**

1. **Flujo General**

El objetivo es llevar un dataset mediante un proceso de Extracción, Carga y transformación (ELT) el cual se puede evidenciar en la figura 1. Que nuestra fuente de información proviene de la aplicación Meet up, mediante el uso una api se extrae la información dicha dataset para posteriormente cargarla en una zona raw alojada en un datalake hospedado en Google Cloud (GCP), para ser procesado y transformado en datos de interés para el negocio, de esta manera ser visualizados en una db de snowflake.

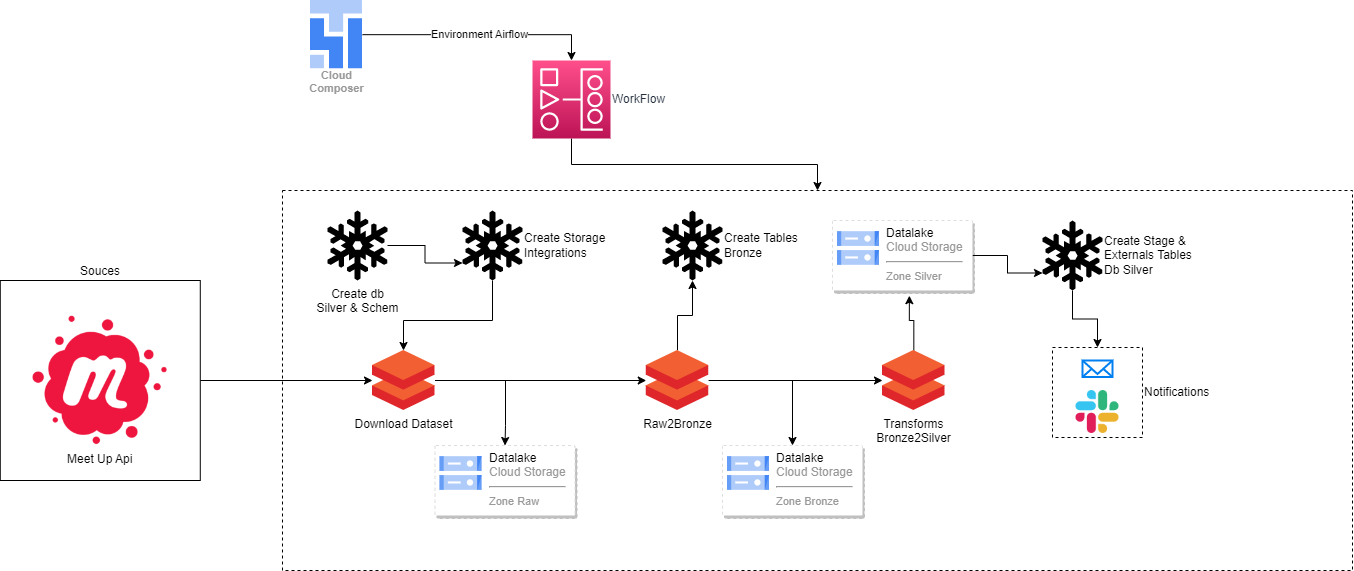


Figura 1. Arquitectura del proceso de ELT

1. **Servicios Cloud**

En la siguiente tabla se evidenciar los servicios usados:

|  |  |  |
| --- | --- | --- |
| **Service** | **Service Name** | **Descripción** |
| Kaggle | Kaggle-api | Framework mediante el cual se realiza la consulta y extracción de los datos puros |
| SnowFlake | SnowFlake | Motor de almacenamientos de datos |
| Cloud Storage | Prueba-loliveros | Servicio de almacenamiento blob |
| Composer | Pruebas-loliveros-Airflow | Entorno distribuido por kubernetes, el cual permite desplegar las diferentes instancias necesarias para los 3 componentes de airflow (Web-Server, Schedule, DB) |
| Airflow | Airflow | Herramienta que permite diseñar pipelines, realizar una orquestación de tareas |
| Databricks | Prueba-rappi | Entorno de desarrollo para procesar grandes volúmenes de datos |

1. **Carga Fuentes**

Para la carga de información se uso una función Figura 2. La cual realiza una petición a la api de kaggle para extraer lo información y posteriormente cargarla en la zona raw del datalake

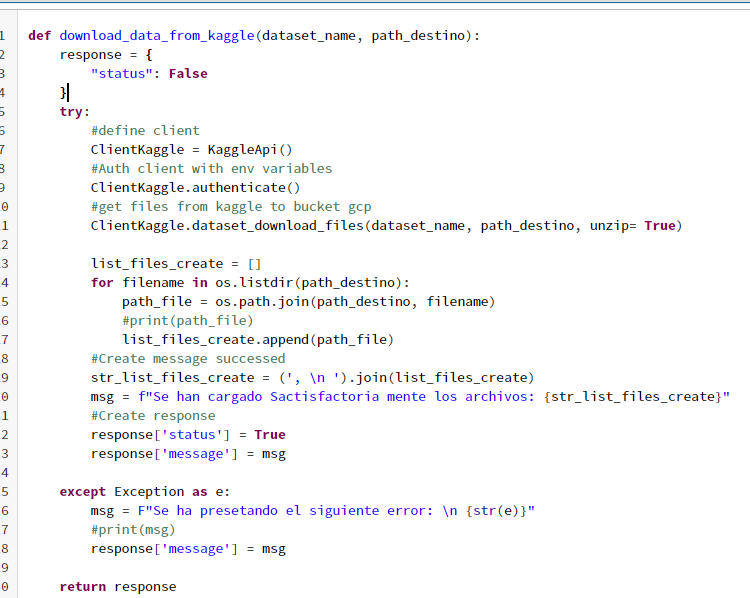


Figura 2. Función de Descarga de Información

**Nota**: Antes de la ejecución de esta función se debe definir la base de datos y el storage integrations de la zona silver, la cual se va usar de stagging

1. **Raw a Bronze**

En la zona raw se tienen los datos de manera nativa y en formatos de almacenamiento no óptimos para su procesamiento de manera masiva, para esto se debe convertir estos formatos en esquemas mas óptimos para el procesamiento, en este ejercicio se hace necesario el uso del formato delta para la zona bronze. Esta transformación se realiza mediante la función Figura 3 y carga estas tablas en la base de datos bronze de snowflake.

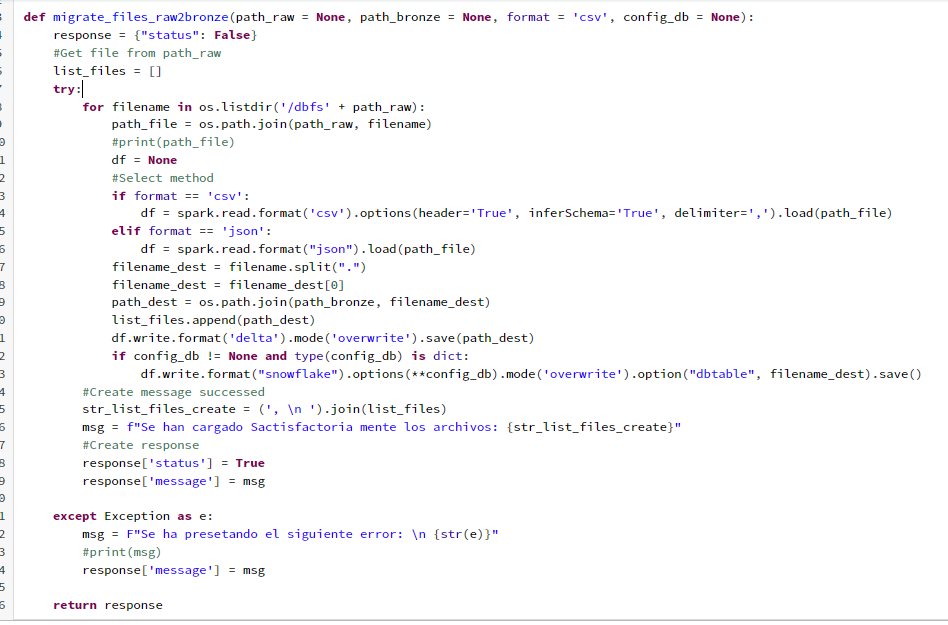


Figura 3. Función de migración Raw2Bronze

1. **Transformaciones**

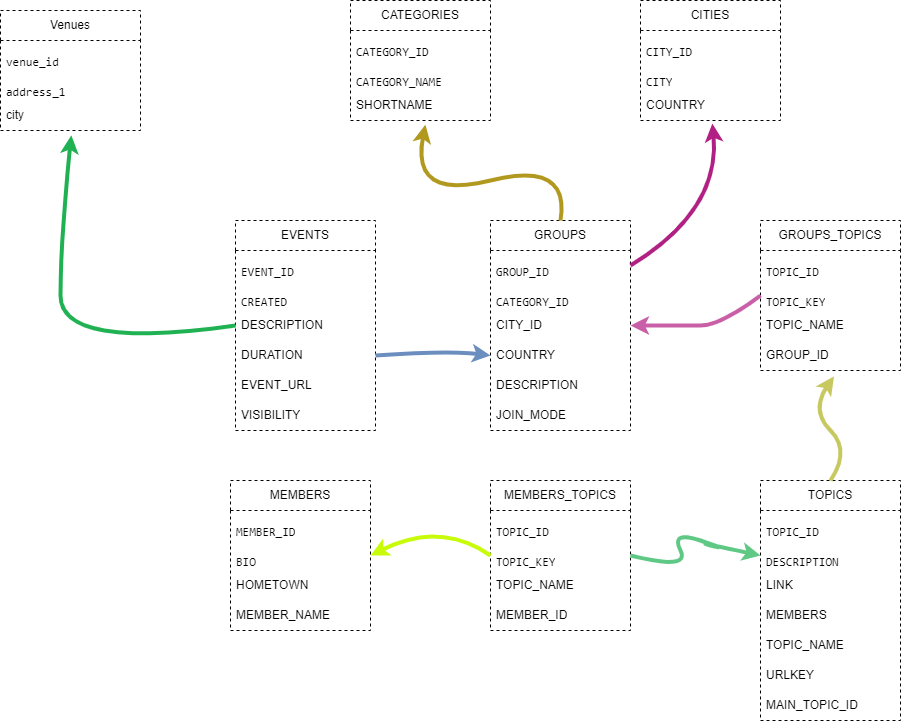


Figura 4 . Arquitectura de Datos

A continuación, se presenta un diagrama relacional Figura 4. de las tablas cargadas en la base de datos, por temas visualización algunas tablas solo cuentan con una muestra de los campos más relevantes.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura 5. Transformaciones silver

Se realizan las trasformaciones solicitadas en la prueba usando el framework de spark, estas se realizan a partir de los datos almacenados en la zone bronze que se encuentran en formato delta. Finalizando esta transformación se procede almacenarlos en la zone silver del datalake y a su vez a linkearlos como tablas externas en la base de datos Silver de snowflake. Ver detalle en el notebook de transform\_silver.

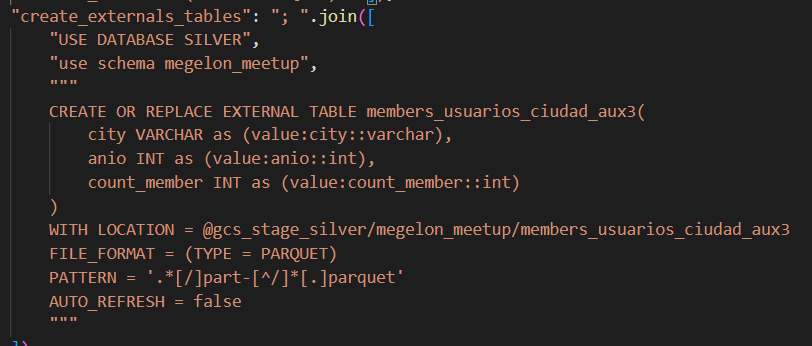


Figura 6. Creación de tablas externas mediante stage y storage intragration Snowflake

1. **Notificaciones**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 7. Notificaciones de error

Se usa dos tipos de notificaciones Canal de slack y email, las notificaciones mediante el canal de slack son principalmente para informar al equipo de desarrollo que el proceso ha fallado o se ha ejecutado de manera exitosa, mientas que la notificación por correo electrónico únicamente va dirigidas al usuario final para informarle que su proceso ha sido exitoso.

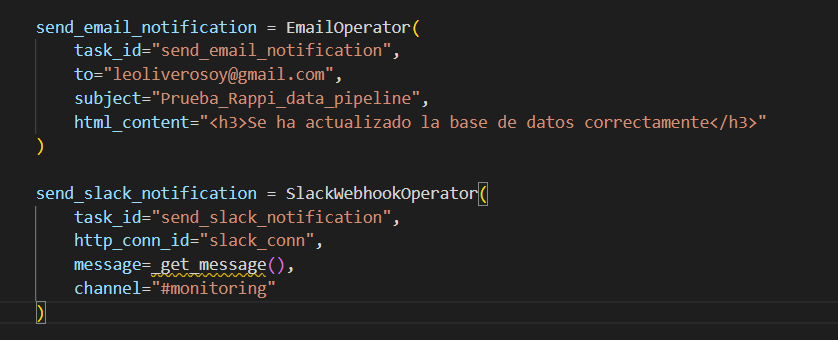


Figura 7. Notificaciones en caso de éxito

1. **DAG Y SCHEDULE AIRFLOW**

Se desplego un environment del servicio Google composer en la versión 1.20 y airflow 2.3.4, en este entorno se crearon variables de entorno se creó por líneas de comando (Ver archivos config\_environments\_composer.txt). además, se instalaron los paquetes necesarios (ver Archivos requirements.txt). Se procede a la programación del pipeline (DAG ver archivo Meetup\_pipeline.py) cuyo grafo se muestra en la figura 8.

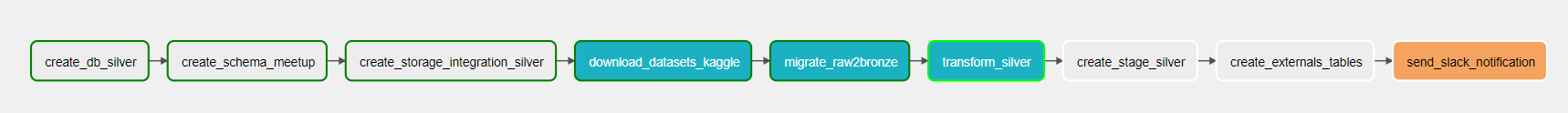


Figura 8. Grafo del Pipeline