POC

**Balance de Energía para la empresa X**

*Resumen:*

Con este proyecto se entregará un proceso completamente automatizado que genere un reporte con el balance diario de una compañía generadora de energía, tomando como entrada un archivo que indica la capacidad de generación diaria de las diferentes plantas activas dentro de la compañía y la información estándar de referencia del ente regulador sabremos si la compañía está generando más, suficiente o menos energía de la que debería.

*Objetivos del proyecto:*

* Establecer formatos y entandares para los archivos de entrada y salida del proceso.
* Automatizar el proceso de extracción y comparación de la información necesaria.
* Analizar los datos para ser capaz de concluir si la capacidad de generación de energía diaria es suficiente.
* Reducir al máximo la intervención humana en el proceso, permitiendo así a los analistas de mercado centrarse en otras de sus funciones.

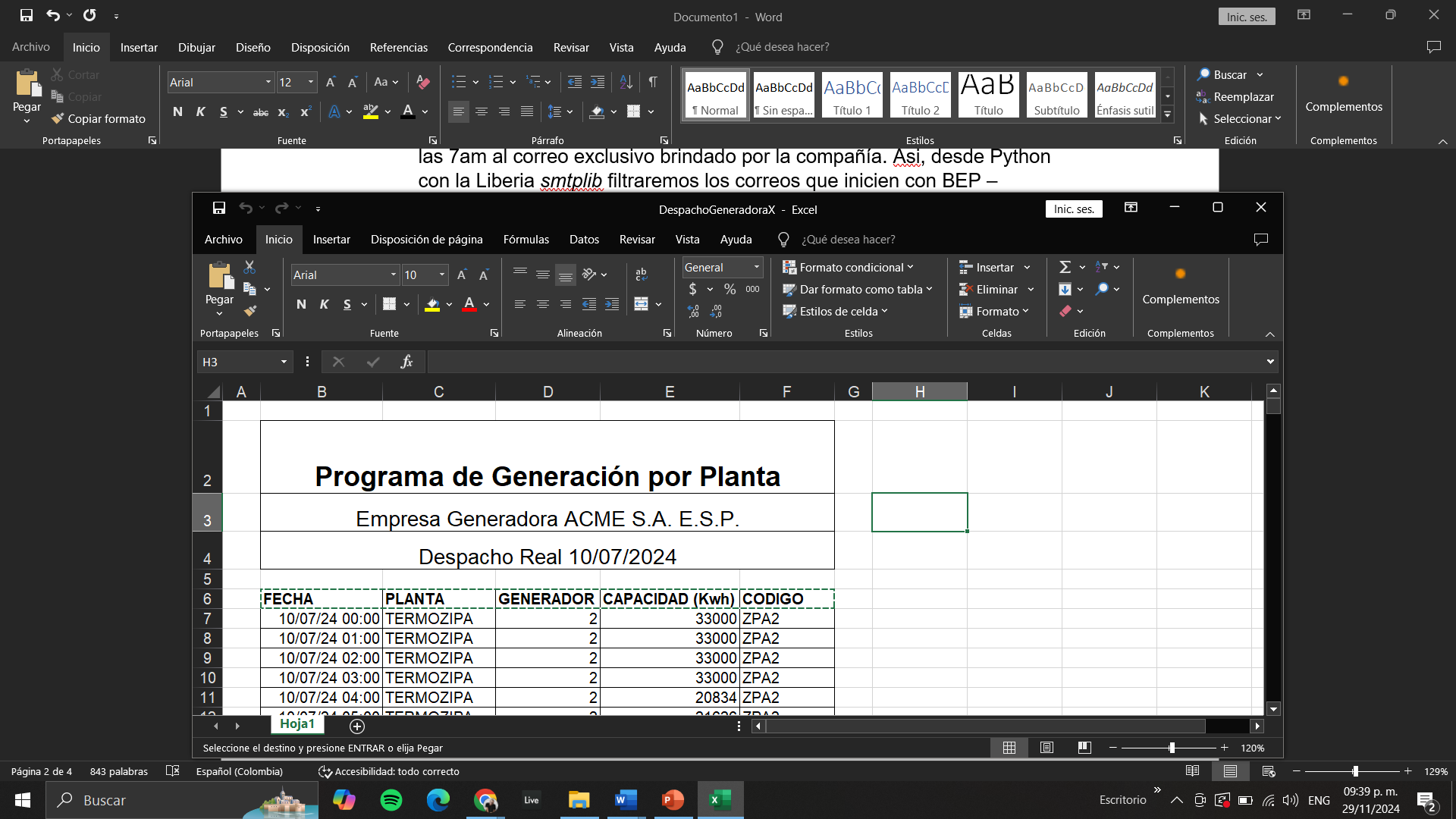
*Proceso actual:*

Python se va usar como núcleo para esta POC porque su versatilidad lo convierte en la mejor opción para manejar todo el pipeline de datos de manera integrada. Además, su ecosistema maduro y su excelente compatibilidad permiten realizar una implementación confiable y escalable.

*Extracción de la informacion:*

*Documento Capacidad Energética.*

* Buscando la menor intervención humana, se propone obtener el reporte de la capacidad energética diaria por medio de un correo electrónico con el encabezado “BEP- Capacidad energética YYYY-MM-DD” enviado antes de las 7am al correo exclusivo brindado por la compañía. Así, desde Python con la Librería *smtplib* filtraremos los correos que inicien con BEP – Capacidad energética y trabajaremos con el que coincida con la fecha de ejecución. De acá tomaremos el archivo Excel adjunto, este deberá tener un tab llamado ‘energía’ con 5 columnas iniciando en la celda B6: fecha, planta, generador, capacidad(kWh), como se muestra a continuación:



Todos estos datos, con pandas se guardarán en un Dataframe llamado GeneracionPlanta.

*API ente regulador*

Se va a consumir la API con la librería requests por su simplicidad y robustez. Con este podemos hacer la consulta con la fecha para cada uno de los datasets. El JSON resultante lo convertiremos a un dataframe con pandas.

¿Por qué pandas?

* Como el proceso se corre diariamente, los conjuntos de datos no van a ser tan grandes.
* La API de pandas es más sencilla e intuitiva, lo que reduce la complejidad de implementación.

*Transformación de la informacion:*

Todo el proceso de limpieza y transformación de los datos los haremos con pandas.

¿Qué se busca?

* Filtrar/Eliminar los registros de la respuesta de la API que no brindan informacion importante para la consulta de ese día, teniendo en cuenta el tipo de plantas, fecha y hora.
* Transformar los datos de fecha para llevarlos al mismo formato en los tres dataframes YYYY-MM-DD HH:MM:SS.
* Asegurar los tipos de datos en todas las columnas y hacer las conversiones pertinentes.

*Carga:*

Con pandas se va a extraer la informacion de los tres dataframes a tres tablas.

¿Por qué tablas?

Con la integración de este proyecto, también se busca ir recolectando informacion para que en el futuro la empresa no solo pueda hacer los reportes diarios si no también un histórico de informacion en el balance de la anergia y su comportamiento a lo largo del tiempo.

Por esta razón, elegimos un Datawarehouse para el almacenamiento de la informacion, ya que es ideal para acumular datos y realizar análisis a largo plazo, permitiendo gestionar no solo la extracción diaria, sino también el historial con consultas rápidas y optimizadas para análisis posteriores.

Buscando usar una base de datos relacional, se proponen PostgreSQLo MySQL, que aunque no son warehouses en su definición tradicional puede actuar como un pequeño Data Warehouse si no se realizan análisis complejos.

En este caso, elegiremos **PostgreSQL** por sus características avanzadas para análisis y manejo de datos históricos y su futuro escalable.

***Análisis***

Al tener la informacion en una base de datos SQL (datos estructurados) usaremos *SQL* para las operaciones como sumas, restas y cruces de la informacion Con este se garantiza la integridad en los cálculos y se aprovecha la optimización interna del Warehouse.

Los resultados se guardarán en un dataframe llamado results y se agregarán a una tabla que tenga el histórico por fecha de los resultados.

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha (YYYY-MM-DD) | Energia\_Generada | Energia\_Requerida | Precio |

Con open **openpyxl** para generar el archivo Excel final que intenta el resultado este proceso y pueda ser entregado por medio de un correo electrónico.

**Resumen Tecnologías Propuestas**

|  |  |  |
| --- | --- | --- |
| **Etapa** | **Tecnología** | **Justificación** |
| Lectura de datos | pandas | Leer reporte Excel |
|  | requests | Consumir API del ente regulador |
| Transformación | pandas | Limpieza, transformación y validación de datos en DataFrames |
| Carga | SQLAlchemy | Conexión para cargar los datos |
| Warehouse | PostgreSQL | Usado para volúmenes pequeños y análisis de datos |
| Cálculos y análisis | SQL |  |
| Report - Export | SQL + Pandas |  |
| Correo electrónico | smtplib | Interactuar con el correo electrónico (recibir y enviar) |