

Diseño de un sistema basado en ETL para la migración de lógica de negocio contenida en hojas de cálculo

**ANDRES FELIPE BENAVIDES MONTOYA
JAIME RAFAEL NEGRETE PARRA**

Tutor:

Isaac Zúñiga Silgado

**Propuesta trabajo de grado para optar al título de
Ingeniero de sistemas y computación**

**UNIVERSIDAD TECNOLOGICA DE BOLIVAR
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS Y COMPUTACIONAL
CARTAGENA DE INDIAS D. T. Y C.
2023**

CONTENIDO

Capítulo 1: PROPUESTA DE TRABAJO DE GRADO

1.1 Estado del Arte

1.1.1 Antecedentes

1.1.2 Marco Conceptual

1.1.2.1 Recursos usados para la extracción y transformación de datos

1.1.2.2 Modelado del proceso ETL usando expresiones de mapeo

1.2 Planteamiento del Problema

1.3 Objetivos

1.3.1 Objetivo General

1.3.2 Objetivos Específicos

1.4 Justificación

1.5 Metodología

Capítulo 2: DESCRIPCIÓN DEL SISTEMA ACTUAL

2.1 Servidor de Procesamiento de Archivos

2.2 Flujo de Transformación de Datos

2.2 Implementación de Cambios

Capítulo 3: AUTOMATIZACIÓN DE LAS VALIDACIONES DE TIPO DE DATOS (TYPECHECKING)

3.1 Sistemas de TypeChecking

3.2 Nuevo Flujo de Trabajo con la Interfaz

3.3 JSON Schema & JSON

3.4 JSON Schema Validation (Problema de Validación)

3.5 JSON Schema: Implementación

3.6 Arquitectura Macro de la Interfaz Web

3.7 Implementación de la Interfaz Web

3.8 Análisis de Resultados

Capítulo 4: AUTOMATIZACIÓN DEL PROCESAMIENTO Y EJECUCIÓN DE FÓRMULAS

4.1 Mapeo de fórmulas

4.1.1 Casos de mapeo: Una hoja de cálculo - Sin fórmulas

4.1.2 Casos de mapeo: Más de una hoja de cálculo - Sin fórmulas

4.1.3 Casos de mapeo: Una hoja de cálculo - Con fórmulas

4.1.3.1 Fórmulas independientes

4.1.3.2 Fórmulas codependientes

4.1.4 Casos de mapeo: Más de una hoja de cálculo - Con fórmulas

4.1.4.1 Fórmulas relacionadas en una misma hoja de cálculo

4.1.4.2 Fórmulas relacionadas en múltiples hojas de cálculo

4.2 Transpilación de fórmulas

4.3 Apache Spark - PySpark

4.4 Flujo de transpiración

4.4.1 Definición de la fórmula

4.4.2 Parseo de la fórmula

4.4.3 Mapeo de fuente de datos

4.4.4 Generación de código python usando pyspark

4.4.5 Plan de ejecución - Apache spark

4.5 Diseño técnico e integración de soluciones

4.6 Análisis de Resultados

CAPÍTULO 5: DISEÑO DE UN SISTEMA DE LOTES DISTRIBUIDOS PARA AUMENTAR EL VOLUMEN DE IMPORTACIÓN SIMULTÁNEAS.

5.1 Proceso de implementación de importaciones distribuidas

5.2 Tecnologías y estrategias para implementar el proceso de importación

5.2.1 Marcos de procesamiento distribuido

5.2.2 Herramientas de integración de datos

5.2.3 Lenguajes de manipulación de datos

5.2.4 Plataformas de almacenamiento y procesamiento de datos

5.2.5 Herramientas de monitoreo y registro

5.3 Diseño técnico e integración de soluciones

5.3.1 Diseño técnico general para la integración de cambios

5.4 Análisis de Resultados

CAPÍTULO 6: CONCLUSIONES

CAPÍTULO 7: RECOMENDACIONES

8. BIBLIOGRAFÍA

Capítulo 1: PROPUESTA DE TRABAJO DE GRADO

1.1 Estado del Arte

1.1.1 Antecedentes

En la actualidad el procesamiento y manejo de datos es un componente crítico, dado que permite impactar el crecimiento del negocio con decisiones basadas en estos, lo cual permite entender el mercado, ayuda a mejorar las estrategias y procesos del negocio.

En pequeñas y medianas corporaciones, las hojas de cálculo como Excel son de uso frecuente para llevar a cabo la trazabilidad, registro y análisis de diversos comportamientos o eventos. Ejemplo de la aplicación de estas características la vemos evidenciada en el trabajo de Soares & Sousa (2017), que busca describir el desempeño de una empresa usando sistemas de puntuación.

Los primeros indicios de este tipo de arquitecturas que permiten modificar y tabular información se dieron con los trabajos de Mattessich (1961, 1964, 1989). Todo este trabajo permitió aplicar hojas de cálculo computarizadas en los sistemas de contabilidad y presupuesto de computadoras mainframe programadas en FORTRAN IV. Estas hojas de cálculo por lotes se ocuparon principalmente de la suma o resta de columnas o filas enteras de variables de entrada en lugar de celdas individuales.

Sin embargo, un sistema actual contiene una gran cantidad de reglas, y su comportamiento tiende a generar un alto tráfico de datos, lo que limitaría el uso de soluciones basadas en hojas de cálculo, ya que son pocas las que soportan tal cantidad de datos.

Por lo tanto, junto con la evolución del negocio, la falta de estructuración y a eso aunado la deuda técnica acumulada, se tienden a generar sistemas aislados o integraciones entre la forma tradicional de manipular los datos (hojas de cálculo) y soluciones de software a la medida (ej. sistemas web), obedeciendo al dominio y al flujo del negocio, realizando así una constante subida de datos a estos para su posterior análisis.

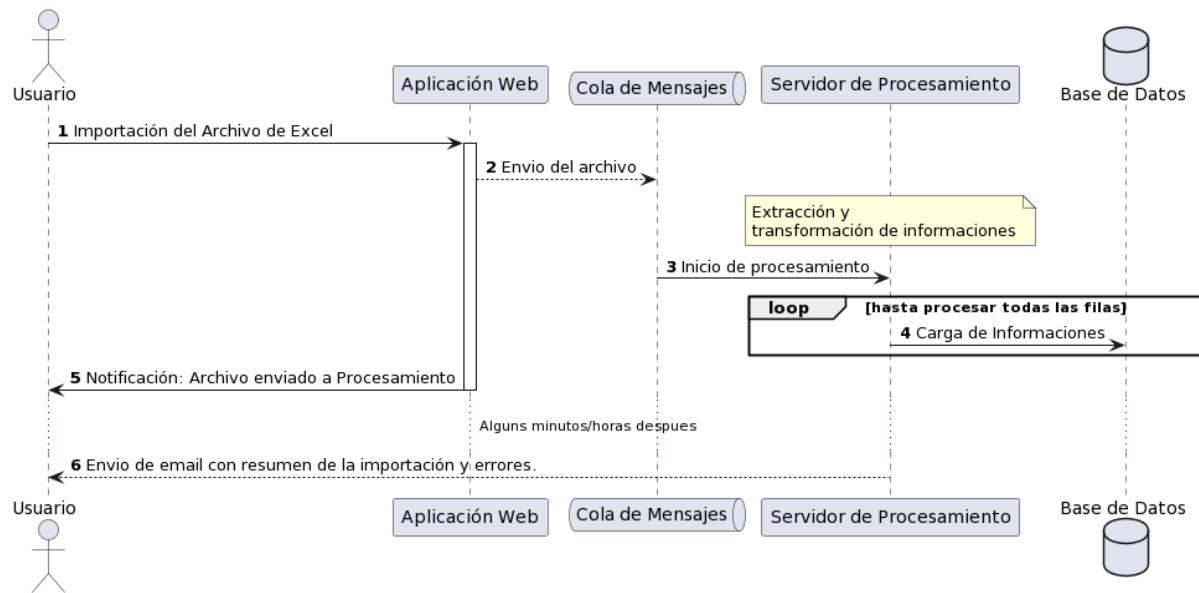
Es por ello, que a inicios del 2010 Ying et. al. expone *la creación de un sistema inteligente de importación de datos masivos de Excel apoyado en el diseño ETL* (Extract Transform Load - Extracción, Transformación y Carga) para la base de datos nacional de recursos de petróleo y gas, el sistema se encarga de importar datos de un Excel a una base de datos de Oracle con el objetivo de integrar y administrar los datos, la función del sistema incluye tres partes: recopilación de datos del Excel, procesamiento de datos y subida de datos a Oracle.

A estos sistemas que manejan gran cantidad de datos se les denomina *sistemas de Big Data*, los cuales necesitan estar disponibles en todo momento para la toma de decisiones a nivel gerencial, lo que significa que el tiempo de carga (L) y transformación (T) no se deberían percibir. Es por ello, que a medida que la información a analizar aumenta, las hojas de cálculo empiezan a ser poco útiles.

Ante lo anterior expuesto, estudios como el realizado por Hung, Benatallah, & Saint-Paul (2011). Propone un nuevo estándar para la transformación de datos a lo que él llamó “*Marco TranSheet*” donde se estudia la transformación de datos basada en hojas de cálculo permitiendo la modificación a datos requeridos por aplicaciones externas y la automatización de las mismas.

El siguiente diagrama de secuencia, explica de forma general, el flujo de importación de hojas de cálculos, siendo utilizado en el sistema web de la herramienta actual.

Figura 1. Flujo de importación de archivos excel en sistema web

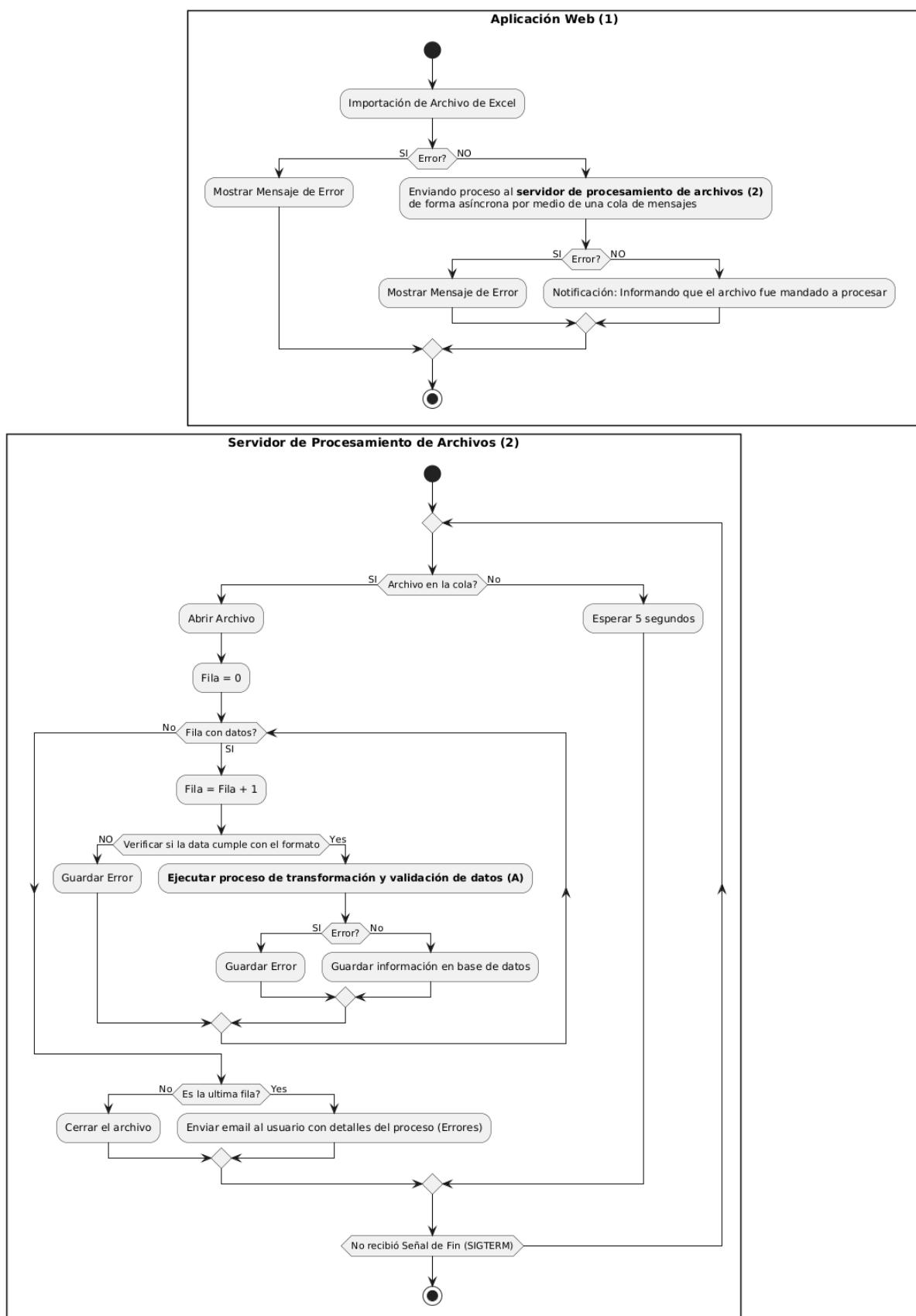


Fuente: Propia

El cual dependiendo del lenguaje de programación utilizado en el servidor de procesamiento (Figura 1), utilizará alguna librería para la lectura, carga, transformación y análisis de hojas de cálculo, por ejemplo pandas, openpyxl en python, o JExcel, Apache POI en el caso de Java, libxlsxwriter, ExcelFormat para C++ y Excelize, xlsx si se utiliza el lenguaje de programación Go.

Con el propósito de entender mejor el flujo ejecutado en el servidor de procesamiento, la siguiente figura describe el proceso de extracción, transformación y carga de datos de archivos Excel en una base de datos, que es ejecutado en este.

Figura 2: Flujo general del proceso.

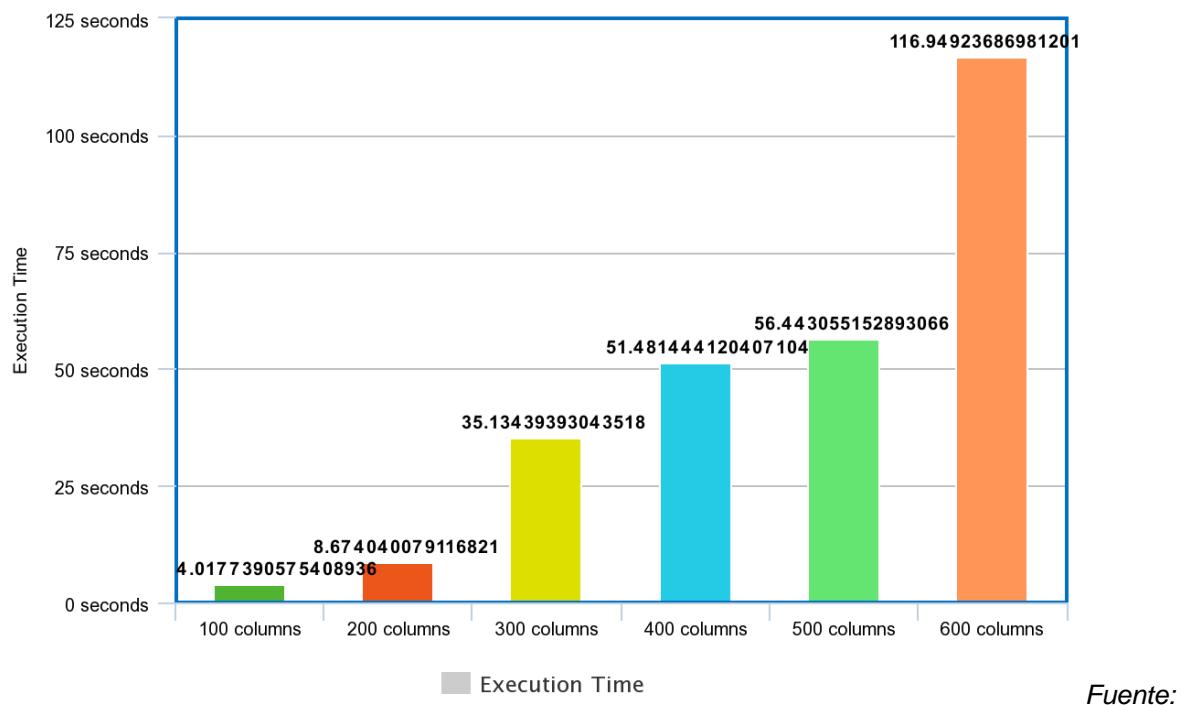


Fuente: Propia

Por ende, podemos observar que el sistema ejecuta un proceso secuencial al momento de extraer, transformar y cargar informaciones de una hoja de cálculo, el cual es programado con Python, utilizando una serie de librerías como openpyxl (2.1.4 - 2014), xlsxwriter (0.8.0, 2016), y data-importer (2.2.1 - 2015); esta última es la librería principal de la ejecución, que hace uso de openpyxl para la lectura de archivos formato xls oxlsx.

Los siguientes son los datos obtenidos después de importar en el sistema actual archivos de diferentes dimensiones

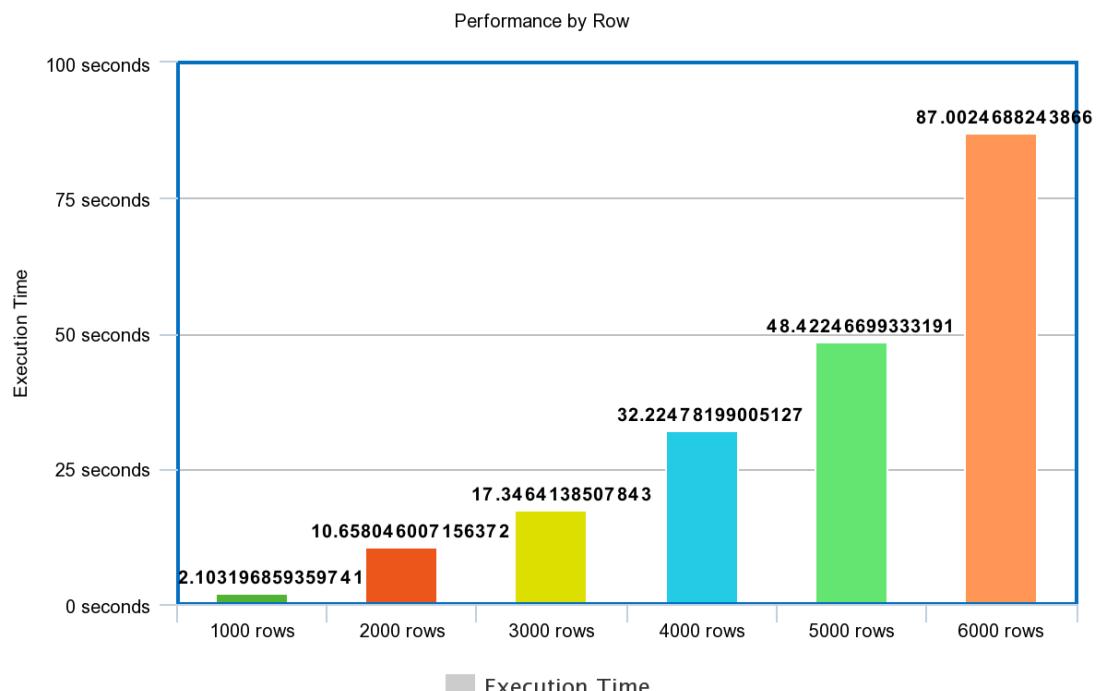
Figura 3. Tiempo de Carga de un archivo basado en el número de columnas y 1000 filas.



Propia

Fuente:

Figura 4. Tiempo de Carga de un archivo basado en el número de filas y 100 columnas.



Fuente: Propia

El sistema actual Inicialmente fue construido para llevar la información contenida en spreadsheet de datos tales como archivo con formato xls, xlsx o csv a una herramienta interna, con el tiempo se empezaron a necesitar procesar archivos con macros, xml e incluso html contenido en documentos xls or xlsx.

De modo que el tiempo de procesamiento de un archivo se ve impactado directamente por el número de filas, columnas, procesos de validación y sub estructuras contenidas en el mismo tales como fórmulas.

Por lo tanto, para ilustrar el impacto de estos, se calculó la variación en el tiempo de carga de un archivo al incrementar el número de columnas (Figura 3.) y filas (Figura 4.), donde la **carga del archivo** hace referencia al tiempo que lleva la librería data-importer (2.2.1 - 2015) cargando las informaciones del archivo a objetos procesables en Python, sin la ejecución de validaciones o transformaciones extras.

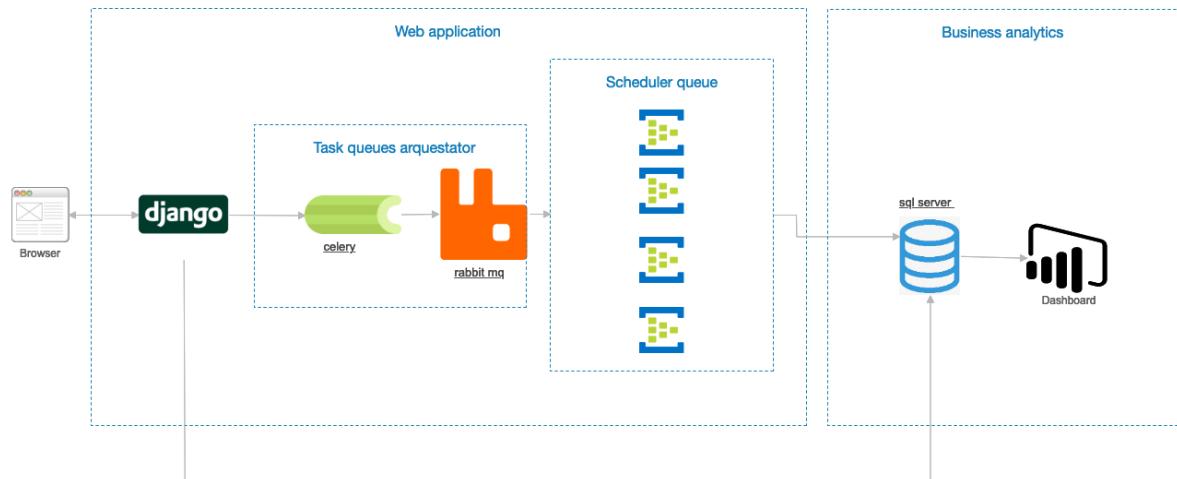
Tabla 1. Especificaciones de la máquina para análisis del tiempo de carga

Procesador	AMD® Ryzen 7 pro 3700u
Memoria RAM	13.5 GiB
SSD	512 GB
Sistema Operativo	Ubuntu 20.04.6 LTS

En consecuencia, tenemos un promedio de aumento del tiempo de carga de aprox. 137% en la agregación de 1000 filas de datos, y de aprox. 99% en la agregación de 100 columnas en el archivo, pruebas ejecutadas en la máquina descrita anteriormente (Tabla 1).

La herramienta, ha funcionado como facilitador entre el funcionario interno y la herramienta de analítica donde se realizan las operaciones, ya que le permite cargar grandes cantidades de información. Posterior a esto la información se encuentra en la herramienta con la posibilidad de consultar y bajar nuevamente el archivo si así se desea.

Figura 5. Diagrama de arquitectura del sistema actual de importación de archivos



Fuente: Propia

Se tiene disponible un servidor Web con Django que recibe las solicitudes de importación, consecuentemente estas solicitudes son redireccionadas a un sistema de gestión de colas, compuestos por Celery e RabbitMQ y finalmente son procesados

por un conjunto de workers para su carga en el sistema de almacenamiento (Figura 5).

Celery es el encargado de gestionar la ejecución y monitoramiento de las tareas, usando el sistema de mensajería RabbitMQ para la gestión y control de las colas de mensajes usadas por las máquinas, en este punto es donde encontramos la oportunidad de mejora, dado a que la actual implementación usa por defecto algoritmos tales como round robin para la planificación y el backend de RabbitMQ para evitar fallos por latencia.

En consecuencia, de lo anterior, podemos resaltar la ineficiencia del proceso ejecutado en el servidor de procesamiento al lidiar con archivos muy grandes (>50 mil filas, >100 columnas). Quedó evidenciado que el tamaño de los archivos impacta directamente en el tiempo de ejecución, además es importante destacar que las validaciones de los importadores comúnmente incluyen procesos como:

- 1 Transformación de tipo de datos.
- 2 Validación de información en la base de datos.
- 3 Validación en otros servicios conectados mediante HTTP.

Y todo esto agrega más tiempo de ejecución al procesar cada fila.

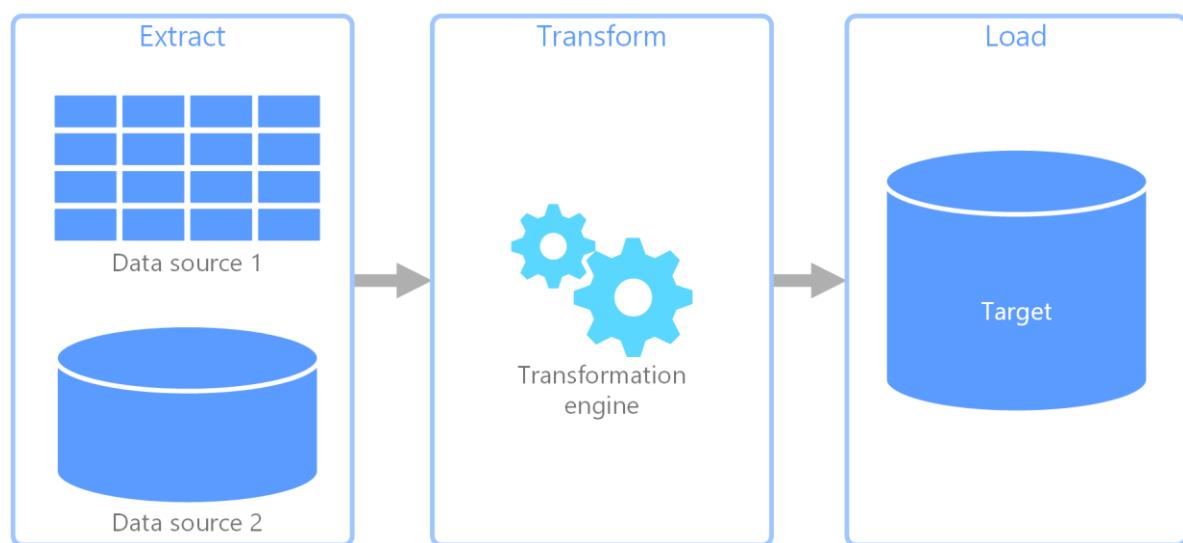
1.1.2 Marco Conceptual

Un **sistema en la nube** es el suministro de servicios informáticos a través de internet, dichos servicios incluyen almacenamiento, servidores, base de datos y más.

Un **Data Warehouse** es un conglomerado integrado de información orientada a la toma de decisiones estratégicas. Esto se logra reuniendo fuentes de información compuestas por datos referentes al caso de estudio.

ETL es un proceso que extrae los datos de diferentes sistemas fuente, luego transforma los datos (como aplicar cálculos, concatenaciones, etc.) y finalmente carga los datos en el sistema Data Warehouse, la siguiente figura ilustra a gran escala el flujo de la información en un sistema ETL.

Figura 6. Flujo de un sistema ETL



Fuente: <https://learn.microsoft.com/es-es/azure/architecture/data-guide/relational-data/etl> (12, 2022)

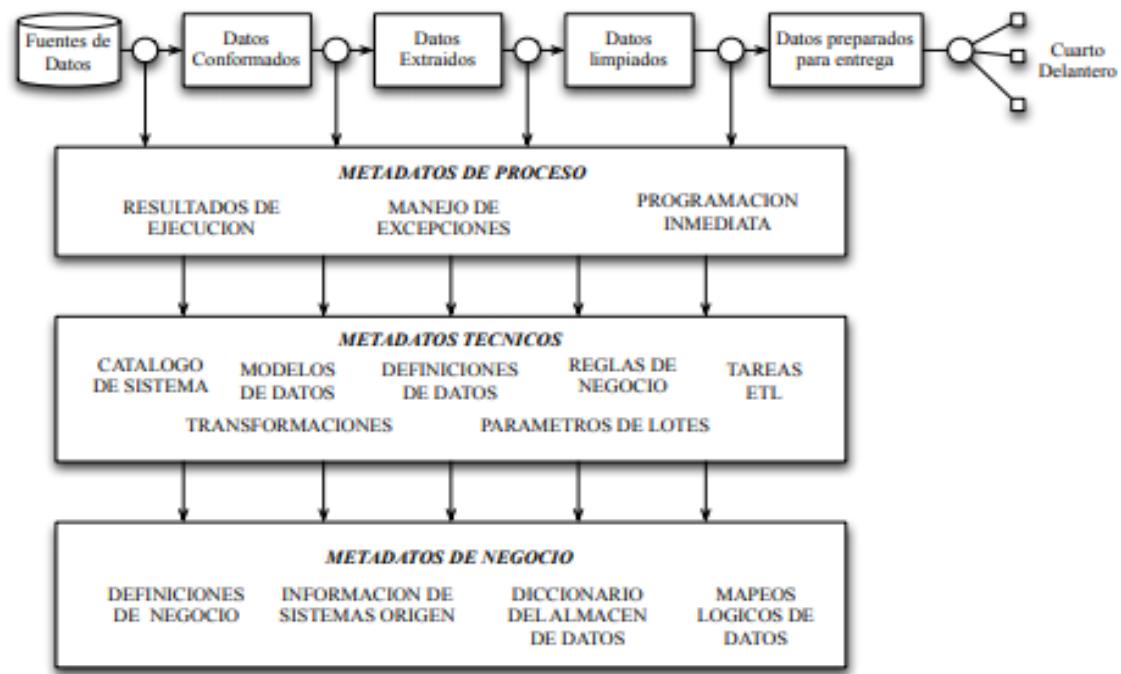
Por lo tanto, las herramientas ETL son piezas de software responsables de la extracción de datos, limpieza, personalización, reformato, integración, e inserción en un almacén de datos (Figura 6.).

Por otro lado, el concepto de Data Warehouse fue propuesto inicialmente por Edgar Frank Codd, este término cambió de forma significativa el concepto del almacenamiento de datos, ya que planteó la integración y unificación de las fuentes

de datos heterogéneas surgidas de los sistemas internos de cada organización, para así proporcionar información histórica, consolidada y fiable a soluciones analíticas más complejas.

Por esta razón, la construcción y diseño de los sistemas de almacenamiento de datos es fundamental, ciertamente hay que garantizar que los mecanismos y procesos para la definición de los datos extraídos y cargados en un Data Warehouse se realice con las mejores prácticas en el ETL.

Figura 7. Elementos y metadatos en un sistema ETL



Fuente: Propia

Se muestran en la Figura 7, los elementos, tareas, problemáticas y técnicas que sigue el actual proceso ETL y los metadatos que tiene.

Una clasificación más generalizada de los metadatos (Figura 7.), con base al perfil de los usuarios que los consumen son los siguientes:

- 1 **metadatos de negocio:** describen el significado de los datos desde el punto de vista de las reglas de negocio.

- 2 **metadatos técnicos:** representan los aspectos técnicos de los datos, incluidos los atributos, tipos de datos, longitudes, linaje. Son resultado de la elaboración de perfiles de datos.
- 3 **metadatos generados por procesos de ejecución:** guardan resultados estadísticos y de seguimiento del proceso ETL, elementos tales como: mapas de transformación entre fuentes de información, cantidad de registros extraídos y almacenados, registros no clasificados, tiempos de carga y extracción por mencionar algunos

1.1.2.1 Recursos usados para la extracción y transformación de datos

Archivos csv de texto plano

Los archivos de texto plano son aquellos que almacenan la información en filas y columnas para emular la estructura de una tabla de una base de datos. Si se trabaja bajo ambientes Windows o UNIX, los archivos están codificados en el estándar ASCII (American Standard Code for Information Interchange).

Los archivos planos pueden ser manipulados y procesados por algunas herramientas ETL o por lenguajes de secuencias de comandos como si se tratases de tablas de bases de datos, solo que en algunas ocasiones mucho más rápido que ellas. Las operaciones de ordenación, mezcla, eliminación, reemplazo y muchas otras funciones de migración de datos se ejecutan mucho más rápido sobre archivos de texto plano que sobre sistemas DBMS.

Una seria consideración que se debe tomar en cuenta cuando se trabaja con archivos de texto plano es que se debe de tener un correcto seguimiento y gestión de los metadatos. Este trabajo algunas veces se puede evitar si se emplea una herramienta ETL, de lo contrario es algo que tiene que llevarse a cabo puesto que resulta primordial en fases del proceso como en la transformación, mapeo y carga de la información. En la fase de entrega, los archivos planos son una excelente alternativa a las bases de datos relacionales, puesto que tiene un mejor desempeño y facilitan tareas como:

- Escritura en disco de los datos para monitoreo y seguimiento.
- Ordenación de la información.
- Filtrado de los datos.
- Reemplazo y sustitución de cadenas de texto.
- Aplicación de operaciones de agregación.
- Referenciación de fuentes de información.

Archivos XML (Lenguaje extensible de marcado)

El XML o lenguaje de marcas extensible es un metalenguaje que permite definir la gramática de lenguajes específicos. No es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para la definición son XHTML, SVG, MathML (*Glosario de Conceptos XML*, 2022).

Fue diseñado para describir datos de manera que sean legibles tanto para máquinas como para humanos. Por lo tanto es ampliamente usado para la definición de nuevos lenguajes, almacenamiento e intercambio de información entre sistemas (*XML 1.0 Specification*. Retrieved 22 August 2010).

La siguiente es la estructura de un documento formato xml

Figura 8. Partes de un XML.

Diagrama que muestra el código XML con las siguientes anotaciones:

- Prologo -- Declaracion**: Señala la parte del código que incluye `<?xml version="1.0"?>` y `<!DOCTYPE message SYSTEM "/xmlstuff/dtds/message.dtd"`. Una flecha apunta desde la parte derecha de la anotación al inicio de la sección de declaraciones.
- Raiz**: Señala la etiqueta `<message>` que es la raíz del documento. Una flecha apunta desde la parte derecha de la anotación al inicio de la etiqueta `<message>`.
- Elementos -- Atributos**: Señala la sección de contenido dentro de la etiqueta `<message>`, que incluye `<from>&contact;</from>`, `<to>&client;</to>`, `<subject>Increíble oferta</subject>`, `<body>&info; Atentamente &contact; </body>`. Una flecha apunta desde la parte derecha de la anotación al inicio de la etiqueta `<from>`.

```
<?xml version="1.0"?>
<!DOCTYPE message SYSTEM "/xmlstuff/dtds/message.dtd"
[
    <!ENTITY client "Fulanito de Tal y Cual">
    <!ENTITY info "Le ofrecemos la increíble oferta de bla bla bla , ...">
    <!ENTITY contact "Ramón Rubial">
]
<message> ← Raiz
<from>&contact;</from>
<to>&client;</to>
<subject>Increíble oferta</subject>
<body>&info;
Atentamente &contact;
</body>
</message>
```

Fuente: Propia

Un documento XML se estructura de forma jerárquica con base a etiquetas como se ilustra en la figura 6, sus partes son:

- **Prólogo:** es el primer parte de todo documento XML identificando el archivo com XLM y opcionalmente puede contener:
 - **DTD (Document Type Definition):** Es la etiqueta `<!DOCTYPE>` ilustrada en la figura 6 declarando el tipo de documento y parser.

- **Declaraciones especiales:** Información extra en el DTD para ser utilizadas por el parser.
- **Raíz:** Primera etiqueta después del prólogo, que encapsula todos las etiquetas/elementos del documento.
- **Elementos:** etiquetas dentro de la raíz para especificar informaciones, además pueden contener atributos.
- **Comentarios:** opcionales y con fines por lo general informativos, no tienen validez sintáctica para el documento.

1.1.2.2 Modelado del proceso ETL usando expresiones de mapeo

Rifaieh y Benharkat (2002) han definido un modelo que cubre diferentes tipos de expresiones cartográficas, utilizaron este modelo para crear una herramienta ETL activa. En este enfoque, las consultas se utilizarán para representar el mapeo entre los datos de origen y de destino; por lo tanto, permite que el DBMS desempeñe un papel ampliado como motor de transformación de datos y como almacén de datos.

Este enfoque permite una interacción completa entre los metadatos de mapeo y la herramienta de almacenamiento, además, aborda la eficiencia de una herramienta ETL de almacenamiento de datos basados en consultas sin sugerir ningún modelo gráfico, describiendo un generador de consultas para un Data Warehouse (DW) reutilizable y más eficiente.

Expresiones de mapeo

La expresión de mapeo de un atributo es la información necesaria para reconocer cómo se crea un atributo de destino a partir de los atributos de origen. A continuación, se enumeran ejemplos de las aplicaciones en las que se utilizan expresiones de mapeo:

- **Mapeo de esquemas (Madhavan et al., 2001):** para el mapeo de esquemas de base de datos, se necesita la expresión de mapeo para definir la correspondencia entre elementos emparejados.
- **Herramienta de almacenamiento de datos (ETL) (Staudt et al., 1999):** incluye un proceso de transformación donde se define la correspondencia entre los datos de origen y los datos DW de destino.
- **Mapeo de mensajes EDI:** se requiere la necesidad de una traducción de mensajes compleja para EDI, donde los datos deben transformarse de un formato de mensaje EDI a otro.
- **EAI (integración de aplicaciones empresariales):** la integración de sistemas de información y aplicaciones necesita un middleware para gestionar este

proceso (Stonebraker y Hellerstein, 2001). Incluye reglas de administración de las aplicaciones de una empresa, reglas de distribución de datos para las aplicaciones en cuestión y reglas de conversión de datos. De hecho, las reglas de conversión de datos definen la expresión de mapeo.

Para la llevar a cabo implementación de importador masivo de datos se utilizarán lenguajes como python para el manejo y orquestación de tareas junto con Scala y apache spark que nos permitirá escalar el procesamiento de datos a un entorno distribuido.

Se utilizará como base una arquitectura microservicios que permita separar las responsabilidades de cada paso en el proceso de validar y transformar la información, eliminando cuellos de botella y haciendo posible monitorear lo que está ocurriendo en el proceso de importación.

El importador tendrá la capacidad de procesar archivos de extensión csv, xls, xlsx, por esta razón se utilizan librerías de código libre escritas en python como, csv, openpyxl, xlrd, que permitirán realizar la lectura de estos archivos con base de código ya estandarizadas y actualmente soportadas por la comunidad.

Finalmente pueden ser implementados patrones de diseño, estructurales, creacionales y de comportamiento como explica Julià, Guiem. (2018, Feb). en “*patrones de diseño*” [9], lo que puede permitir realizar implementaciones legibles, entendibles y testables, ejemplos de estos son los siguientes.

Patrones creacionales

Los patrones creacionales sirven para solucionar problemas derivados de la creación de nuevos objetos. Estos patrones nos ayudan a encapsular y abstraer dicha creación, tenemos los siguientes patrones:

- **Abstract Factory (Fábrica abstracta):** Creación de objetos cuando se tiene múltiples criterios de creación.

- **Builder (Constructor):** Permite la creación de objetos complejos o compuestos por muchos otros sub-objetos.
- **Singleton:** Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Solo puede instanciar la clase a un único objeto.

Patrones estructurales

Son patrones que nos ayudan a modelar nuestra aplicación especificando la forma en la que unas clases se relacionan con otras. Algunos de los patrones estructurales son los siguientes:

- **Adapter o Wrapper (Adaptador):** Permite a dos clases con diferentes interfaces comunicarse entre ellas utilizando un objeto intermedio.
- **Bridge (Puente):** Desacopla la abstracción de implementación.
- **Decorator (Decorador):** Permite añadir funcionalidad extra a un objeto (de forma dinámica o estática).

Patrones de comportamiento

Para terminar, tenemos los patrones de comportamiento, estos se encargan de gestionar los algoritmos encapsulados por las clases, las relaciones y responsabilidades entre objetos. Tenemos los siguientes patrones:

- **Chain of responsibility (Cadena de responsabilidad):** Establece la línea de mensajes a seguir para realizar la tarea indicada.
- **Command (Comando):** Son objetos que encapsulan una acción, permitiendo ejecutar dicha operación sin necesidad de conocer el contenido de la misma.
- **Memento:** Este patrón permite volver a estados anteriores del sistema.

- **Observer (Observador):** Define una relación uno a muchos entre objetos y, cuando este es modificado, notifica a todos los objetos que dependen de él para realizar una acción determinada.
- **Strategy (Estrategia):** Dispone de varias soluciones para un problema y decide cuál de ellas es mejor en tiempo de ejecución.

1.2 Planteamiento del Problema

Las hojas de cálculo son de gran importancia en el ámbito empresarial, estas ayudan en el análisis, rastreo y transformación de la información. Pero a medida que la empresa crece, esta información también lo hace, lo anterior impacta el rendimiento de las hojas de cálculo al manejar grandes volúmenes de información. En consecuencia, provoca la migración de informaciones en Excel a sistemas en la nube, usando aplicativos web (Figura 1.), para hacer posible la migración de la información de Excel a esos nuevos sistemas para el posterior análisis.

El **Importador de datos masivo** actual utiliza el proceso descrito en la *Figura 2*. En el flujo de este proceso se presentan los siguientes inconvenientes:

- **Poca experiencia de usuario**

Cómo se logra apreciar en el módulo **Aplicación web** (Figura 2.), el alcance en términos de monitorización por parte del cliente es insuficiente, ya que solamente se le informa cuando el proceso terminó de procesar todas las líneas del archivo, lo cual puede provocar impaciencia o incertidumbre del proceso y el usuario opte por importar nuevamente pensando que el proceso no está en ejecución.

- **Acoplamiento de responsabilidades**

Como se puede observar en el módulo **Servidor de procesamiento de archivos** (Figura 2), el proceso ejecuta la fase de validación, limpieza, transformación y carga de información en diferentes etapas y componentes, por lo tanto la mantenibilidad de estos componentes por separado implican más trabajo para los desarrolladores dado el alto acoplamiento y falta de métricas de monitoreo que se presenta, lo que impacta de forma directa en el tiempo de desarrollo de nuevas funcionalidades, y al mismo tiempo la complejidad del existente.

- **Procesamiento secuencial**

Como se puede observar en el módulo **Servidor de procesamiento de archivos** (Figura 2.), Actualmente se realiza el procesamiento de los archivos

secuencialmente, es decir, línea a línea lo cual funciona hasta cierto punto con archivos pequeños, pero cuando el sistema necesita procesar archivos grandes estos demoran mucho en procesar, esto principalmente generado por la falta de optimización en el manejo de las tareas, por lo cual actualmente se limita el tamaño de archivo por importación, creando sobre trabajo para el usuario al tener que dividir los archivos manualmente.

Además del **proceso de validación y transformación de datos (A)** referenciado en la Figura 2, los usuarios de la herramienta comentan las siguientes debilidades:

- 1 “El tiempo de importación a la plataforma es demasiado largo, lo que ralentiza el proceso y logística del negocio.” - *Analista de Logística*.
- 2 “La falta de retorno acerca del estatus del proceso que se encuentra ejecutando, genera un sobrecosto a causa de la incertidumbre generada por la espera, lo que resulta muchas veces en sobre trabajo para la herramienta (usuario importa de nuevo pensando que no fue procesado) y consigo para el proceso en general.” - *Gerente de Soporte al usuario*.
- 3 “La integridad de la información se ve comprometida, ya que muchas de las reglas de negocio no son ejecutadas de la forma apropiada, lo que resulta en conclusiones erróneas por parte de la herramienta” - *Analista de Facturamiento*.
- 4 “El proceso de implementación de nuevos cambios es muy lento esto causado por la falta de documentación, lo que resulta en redundancia de información o funcionalidades.” - *Programador Senior*.
- 5 “La generación de reportes genera un sobrecosto dado el tiempo de generación de muchos reportes gerenciales y necesarios para la toma de decisiones, tanto a nivel logístico como operativo, esto como parte fundamental en la trazabilidad y validación del proceso” - *Analista de Procesos*.

- 6 “A nivel técnico la implementación de nuevas reglas de negocio resulta en malas prácticas de desarrollo y por ende en sobrecosto de planeación, esto dado a la poca configurabilidad y documentación” - *Programador Junior*.
- 7 “A nivel infraestructura la implementación de nuevos flujos de importación de datos al sistema provoca continuas pausas del sistema para llevar nuevos cambios al ambiente de producción, lo que genera en una disminución de disponibilidad de la plataforma y muy a menudo afectando la integridad de los procesos que se encuentran en ejecución” - *Arquitecto de Software*

A medida que más información debe ser procesada el importador de datos demanda más recursos de hardware y software que permitieran validar y transformar la información de entrada, esto principalmente para prevenir y minimizar el error humano, de formato y estructura en los datos que se intentan cargar a la herramienta. En este punto el rendimiento de la herramienta se encontraba condicionada por tres factores, cantidad de información, número de validaciones y transformaciones necesarias para cumplir los requerimientos del proceso.

Por lo tanto, considerando lo anterior, el presente proyecto propone diseñar un sistema de importación basado en reglas configurables que permitan describir un flujo de datos, logrando así el proceso de sincronización y migración de hojas de cálculo tradicionales a un sistema distribuido.

Todo esto con el objetivo de disminuir los tiempos de carga y análisis que el usuario actualmente enfrenta para llegar a una conclusión gerencial y operativa basada en los datos del negocio, paralelo a esto se propondrá impactar positivamente el tiempo de manutención del software, enfocando en el flujo de adicionar nuevos comportamientos por parte de un desarrollador.

Por lo anterior nos plantemos la siguiente pregunta problemática:

¿Cómo se debe diseñar una arquitectura que permita escalar micro flujos de negocio contenidos en una hoja de cálculo para la optimización del tiempo de procesamiento, desarrollo y evaluación del importador masivo?

1.3 Objetivos

1.3.1 Objetivo General

Diseñar un sistema ETL que permita disminuir la curva de desarrollo, mantenibilidad y mejorar el rendimiento, usabilidad del sistema de procesamiento de archivos, reduciendo los tiempos de espera respecto a la implementación actual.

1.3.2 Objetivos Específicos

- 1 Reducir el tiempo de desarrollo de cambios a las validaciones de tipo (TypeChecking) ejecutadas por los importadores, utilizando una interfaz para la definición de estas.
- 2 Reducir los tiempos de importación automatizando y paralelizando la configuración de las fórmulas contenidas en el archivo excel y la importación de los datos respectivamente.
- 3 Aumentar el volumen de importaciones simultáneas diarias, mediante el diseño de un sistema de procesamiento por lote distribuido.

1.4 Justificación

La utilización de hojas de cálculo para el manejo y monitorización de micro flujos de negocio a nivel financiero y logístico, presenta problemas de rendimiento y mantenibilidad cuando el conjunto de reglas es complejo o el conjunto de datos a tratar es muy grande, prueba de esto son las limitaciones respecto a los recursos de cada máquina, ya que esos software tienden a ser soluciones desktop, en estos casos las implementaciones empiezan a presentar problemas que generan cuellos de botella y errores en la integridad de datos, ralentizando el proceso que obedece al flujo del negocio, disminuyendo la velocidad de toma de decisiones, y finalmente pérdida de valor a nivel comercial.

Ante estos desafíos las empresas han optado por desarrollar software hechos a la medida que satisfagan las necesidades del negocio dando solución a los problemas de performance y mantenibilidad dispersa en cada hoja de cálculo, centralizando y manipulando en un solo lugar. Ejemplo de lo anterior son los desarrollos in-house donde se tiene disponibilidad de un equipo exclusivo para soporte al área en IT.

Sin embargo, los cuellos de botella generados y el constante cambio en el dominio del negocio hacen que sea tedioso de mantener; por ejemplo, aplicaciones para cálculo de impuestos donde sus controles en hojas de cálculo se ve afectado por decisiones gubernamentales (cambio de impuestos, aplicación de nuevas leyes según el régimen local, etc). Y en consecuencia los importadores necesitan ser actualizados.

Donde estos importadores hacen parte de un proceso ETL, que se enfoca en el manejo y manipulación de datos, y son responsables de:

1. La extracción de los datos apropiados de las fuentes.
2. El Transporte a un área de propósito especial donde será procesado.
3. La transformación de los datos fuente y el cálculo de nuevos valores que obedecen las reglas preestablecidas (fórmulas).
4. El aislamiento y limpieza de los datos, para garantizar que las reglas comerciales se cumplan y se respeten las restricciones de la base de datos.

- La carga de los datos limpios y transformados a la relación apropiada en el almacén, como fue explicado en Vassiliadis, P. (2009) “A survey of extract–transform–load technology.”[7].

Por lo tanto, es necesario una solución amigable, usable, flexible y escalable que ofrezca una hoja de cálculo convencional, pero con el poder y los recursos necesarios para trabajar con reglas de negocios complejas y flujo de datos grandes.

Tabla 2. Herramientas de ETL analizadas

Herramienta	Descripción	Puntos Negativos
Apache Airflow [38]	Una plataforma de código abierto para automatizar, programar y monitorear flujos de trabajo de manera programática.	Reglas del ETL y importaciones no serán intuitivamente configurable para el usuario final
Luigi [39]	Un módulo de Python que ayuda a construir complejos pipelines de trabajos en batch, manejando dependencias, gestión de flujos de trabajo y visualización	Reglas del ETL y importaciones no serán intuitivamente configurable para el usuario final
Bonobo [37]	Un framework ETL ligero para Python 3.5+ que se centra en la facilidad de uso.	Proyecto abandonado
Prefect [40]	Una plataforma de automatización de código abierto que es una alternativa moderna a Airflow, con un fuerte enfoque en Python.	Reglas del ETL y importaciones no serán intuitivamente configurables para el usuario final
Petl [36]	Herramienta para Extraer, Transformar y Cargar (ETL) tablas de datos en Python, con soporte para una amplia gama de formatos de datos.	Entrega un sistema de módulos pero que no es fácilmente desacoplable en un sistema distribuido
Apache Spark [19]	Plataforma de procesamiento de datos de código abierto que se utiliza principalmente para realizar análisis de datos a gran escala.	No es fácilmente configurable y no continúe todos los componentes de un ETL

Hoy en día en las organizaciones, una solución ETL que se enfoque en ambientes distribuidos, que venga junto a una capa de abstracción que permite ocultar al usuario la complejidad y tecnicismo de un lenguaje, que permitiría mantener la usabilidad, siendo configurable, mantenible, asimismo, disminuiría los cuellos de botella del flujo de negocio y garantiza la integridad de los datos para la toma de decisiones gerenciales.

Después de analizar algunas herramientas ETL (Tabla 2.), se resalta la configuración de un software in house con el diseño, siendo propuesto en este documento, debido a que ninguna de las herramientas gratuitas y open-source satisfacen todas las características mencionadas anteriormente principalmente la usabilidad y configuración por parte del usuario final, en el diseño que será descrito más adelante; se utilizan algunas de estas herramientas que en conjunto crean el ecosistema ETL configurable, distribuido y escalable.

Para la academia cartagenera, este proyecto presenta un estudio basado en el proceso de validación y transformación de datos que permita migrar y desacoplar información que contribuye a la mejora de los procesos de las empresas y sus flujos de negocio, lo cual se refleja económicamente con lo mencionado en el anterior párrafo, asimismo, al traer a estudio este tipo de soluciones a nuestro entorno local poder generar como consecuencia posteriores investigaciones sobre la misma.

Finalmente, la aplicación e implementación del importador de datos en ambientes donde sea necesario la validación y transformación de grandes cantidades de datos disminuirá el tiempo y costo en estos sistemas lo que afectará las operaciones logísticas y financieras a corto, mediano y largo plazo.

1.5 Metodología

La presente investigación se considera del tipo descriptiva y aplicada, debido a que utilizamos análisis estadísticos sobre los datos del sistema actual para diseñar la solución al problema encontrado, proponiendo una solución que puede ser implementada. De igual manera, se denominó la misma con un enfoque mixto; cuantitativo y cualitativo, a razón de que para la debida ejecución del estudio se observa el correcto comportamiento de la implementación antigua y basado en esos se define el diseño de una solución para mejorar los problemas de esta.

Por lo anterior, Inicialmente se identificarán las variables y parámetros que permitan el desarrollo del algoritmo destinado para el balance de las operaciones a realizar en un marco de producción hipotética.

Se definirá el tipo de lenguaje en el cual se representará el problema, además de establecer las restricciones que sean necesarias para el desarrollo del proceso, esto teniendo en cuenta el planteamiento inicial el cual tendrá como principal objetivo la transformación y control de grandes cantidades de datos utilizando técnicas de distribución y la utilización de algoritmos modernos.

Finalmente, se desarrollará el análisis de los datos arrojados en base a la fase anterior y se observará el comportamiento de cada iteración de acuerdo a las modificaciones definidas previamente, encontrando así una solución óptima que permita utilizar la cantidad de recursos necesarios para realizar la transformación requerida sobre los datos.

Se utilizarán principios encontrados en Shigarov, A. O., & Mikhailov, A. A. (2017). “*Rule-based spreadsheet data transformation from arbitrary to relational tables.*” [6].

Tipo De Investigación

Se realizarán observaciones diarias y mantenimiento a un sistema legado de importación de hojas de cálculo, lo que permitirá identificar puntos críticos a atacar

estableciendo objetivos que permitan planificar una solución distribuida basada en ETL.

Al diseñar un sistema padronizado de configuración basado en ETL, se tendrán en cuenta una serie de fases secuenciales relacionadas con los objetivos específicos, descritas a continuación:

Fase 1: Instaurar un método para hacer typechecking (Validación de tipo de datos): En esta etapa se validará si la información cargada corresponde el esquema planteado, lo que evitará gran cantidad de errores semánticos, al tratar de realizar operaciones con tipo de datos incompatible, además se asegura que una vez la información pase a la etapa de transformación la cantidad de validación disminuya, lo que significa un valor agregado al performance.

Para desarrollar la **Fase 1** y así dar solución a nuestro objetivo 2.2.1 (*Reducir el tiempo de desarrollo de cambios a las validaciones de tipo [TypeChecking] ejecutadas por los importadores, utilizando una interfaz para la definición de estas*) se realizarán las siguientes tareas:

- 1 Diseñar modelo de base de datos para guardar las configuraciones de columnas de las diferentes hojas de cálculo a manejar en el sistema.
- 2 Planeación de flujo para interpretar configuración de columnas registrada a proceso limpiador de datos.
- 3 Planeación de flujo de entrega de información después de limpiada.
- 4 Escoger motor de base de datos y tecnologías para el almacenamiento de las configuraciones de hojas de cálculo, proceso para ejecución de limpieza y entrega de esta.
- 5 Diseño del modelo y sistema general para realizar typechecking.

Fase 2: Crear un mínimo viable para escalar hojas de cálculo: En esta etapa se implementará un sistema de validación y transformación que permita de forma legible la migración de lógica matemática contenida en hojas de cálculo.

Para desarrollar la **Fase 2** y así dar solución a nuestro objetivo 2.2.2 (Reducir los tiempos de importación automatizando y paralelizando la configuración de las fórmulas contenidas en el archivo excel y la importación de los datos respectivamente.) se realizarán las siguientes tareas:

- 1 Elegir una herramienta de procesamiento distribuido que permita escalar la ejecución de los procesos.
- 2 Modelar la arquitectura del sistema junto con las limitaciones existentes con respecto a las tecnologías a usar.
- 3 Modelar arquitectura a nivel infraestructura para definir los límites de recursos para el funcionamiento.
- 4 Diseño del producto mínimo viable para pruebas de rendimiento.

Fase 3: Diseñar un sistema ETL: En esta etapa se integrarán los sistemas de carga y validación, junto con los de transformación, con el fin de consolidar un sistema que respete los principios de un ETL, lo que permitirá tener un control sobre el flujo del negocio, respecto a una estrategia existente, pero conformada por implementaciones separadas que brindan un valor agregado en usabilidad y performance.

Para desarrollar la **Fase 3** y así dar solución a nuestro objetivo 2.2.3 (Aumentar el volumen de importaciones simultáneas diarias, mediante el diseño de un sistema de procesamiento por lote distribuido.) se realizarán las siguientes tareas:

- 1 Modelar la arquitectura de un micro-lenguaje que permita la transformación de datos.
- 2 Diseñar un producto mínimo viable del micro-lenguaje para la transformación de datos.

Figura 9. Cronograma de Actividades

TAREAS	SEMANA 1	SEMANA 2	SEMANA 3	SEMANA 4	SEMANA 5	SEMANA 6	SEMANA 8	SEMANA 9	SEMANA 10	SEMANA 11	SEMANA 12
1 CREAR REPLICIA DEL SISTEMA ACTUAL PARA VALIDACION DE RENDIMIENTO											
2 INVESTIGACION DE TECNOLOGIAS PARA DISEÑAR SERVICIO DE TYPECHECKING											
3 INVESTIGACION DE TECNOLOGIAS PARA REALIZAR TRANSFORMACION Y REGRAS DE NEGOCIO SOBRE LOS DATOS											
4 INVESTIGACION DE TECNOLOGIAS PARA PROCESAMIENTO DISTRIBUIDO											
5 PLANEACION DE FLUJO PARA INTERPRETAR CONFIGURACION DE COLUMNAS REGISTRADA A PROCESO LIMPIADOR DE DATOS.											
6 MODELAR LA ARQUITECTURA DEL SISTEMA JUNTO CON LAS LIMITACIONES EXISTENTES CON RESPECTO A LAS TECNOLOGIAS A USAR											
7 PLANEACION DE FLUJO DE ENTREGA DE INFORMACION DESPUES DE LIMPIADA.											
8 MODELAR ARQUITECTURA A NIVEL INFRAESTRUCTURA PARA DEFINIR LOS LIMITES DE RECURSOS PARA EL FUNCIONAMIENTO.											
9 ESCOGER MOTOR DE BASE DE DATOS Y TECNOLOGIAS PARA EL ALMACENAMIENTO DE LAS CONFIGURACIONES DE HOJAS DE CALCULO, PROCESO PARA EJECUCION DE LIMPIEZA Y ENTREGA DE ESTA.											
10 DISEÑO DEL PRODUCTO MINIMO VIABLE PARA TRANSFORMACION Y REGRAS DE NEGOCIO SOBRE LOS DATOS											
11 MODELAR LA ARQUITECTURA DE UN MICRO-LENGUAJE QUE PERMITA LA TRANSFORMACION DE DATOS.											
12 DISEÑAR UN PRODUCTO MINIMO VIABLE DEL MICRO-LENGUAJE PARA LA TRANSFORMACION DE DATOS.											
13 ANALISIS DO DESIGN, RESALTANDO VENTAJAS Y DESVENTAJAS											

Fuente: Propia

Fuentes De Información

Fuentes de información primaria

Las principales fuentes consultadas y requeridas para este proyecto, es proporcionada por revistas científicas que suministran información destacada acerca de importación de datos e implantación de sistemas ETL.

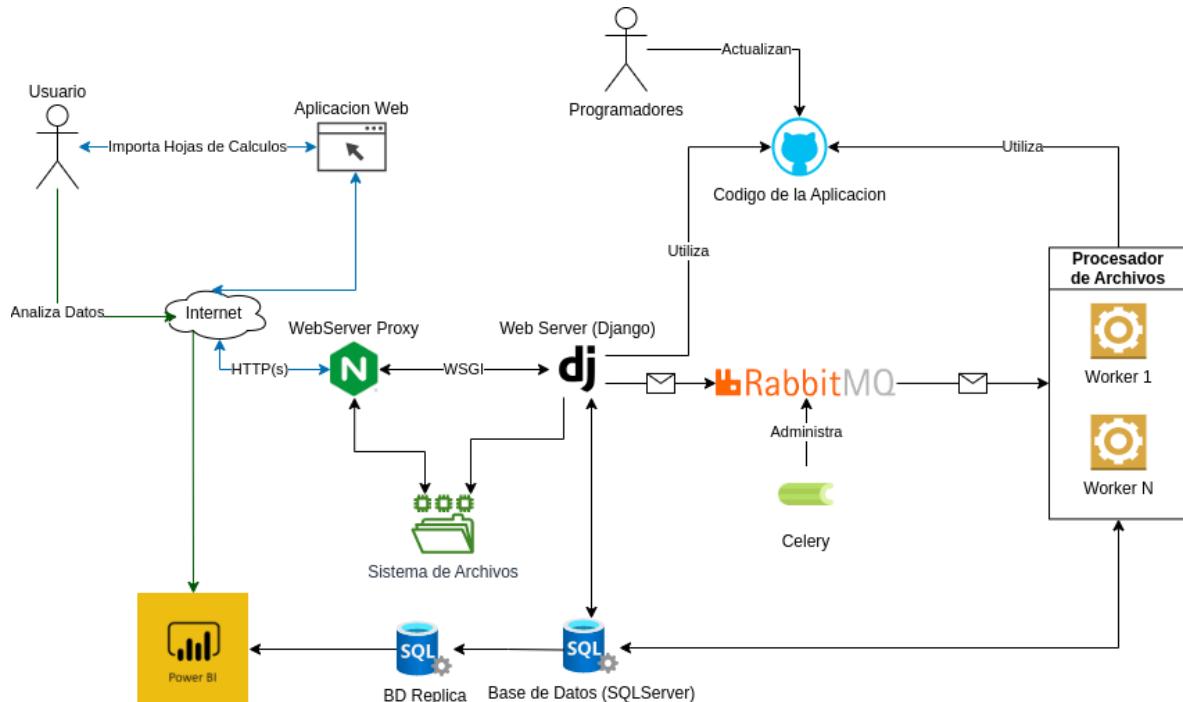
Fuentes de información secundarias

En segunda instancia la información será suministrada por libros, artículos de revistas, publicaciones e investigaciones relacionadas con el tema central, tal como “*Rule-Based spreadsheet Data Transformation from Arbitrary to Relational Tables. Information Systems*” desarrollado por Shigarov et al., (2017).

Capítulo 2: DESCRIPCIÓN DEL SISTEMA ACTUAL

El siguiente diagrama ilustra a alto nivel los componentes del sistema actual.

Figura 10: Arquitectura actual extendida de la Figura 5.



Fuente: Propia

En este capítulo se explicará el estado actual del sistema, las tecnologías utilizadas, los flujos de trabajo realizados y cómo son implementadas las actualizaciones en la arquitectura.

Actualmente se cuenta con una plataforma WEB donde el usuario importa la hoja de cálculo o archivo de excel, el cual es guardado en el servidor y enviado para un cluster de workers (Process Server) que procesarán este archivo, como se explica en la Figura 1.

En la Figura 2, se ilustra el proceso de forma general que se realiza al momento que el servidor Web recibe la importación del archivo por parte del usuario y el flujo de validación, transformación y carga de las informaciones en la base de datos para el posterior análisis.

La arquitectura que permite realizar todos los procesos descritos anteriormente, se basa en un servidor web usando un sistema de mensajería para la comunicación con un conjunto de workers (Procesadores de Archivos), los cuales comparten el mismo código base, es decir, el código de los workers y servidor web está en el mismo lugar, como se puede observar en la Figura 7, la que describe las siguientes tecnologías:

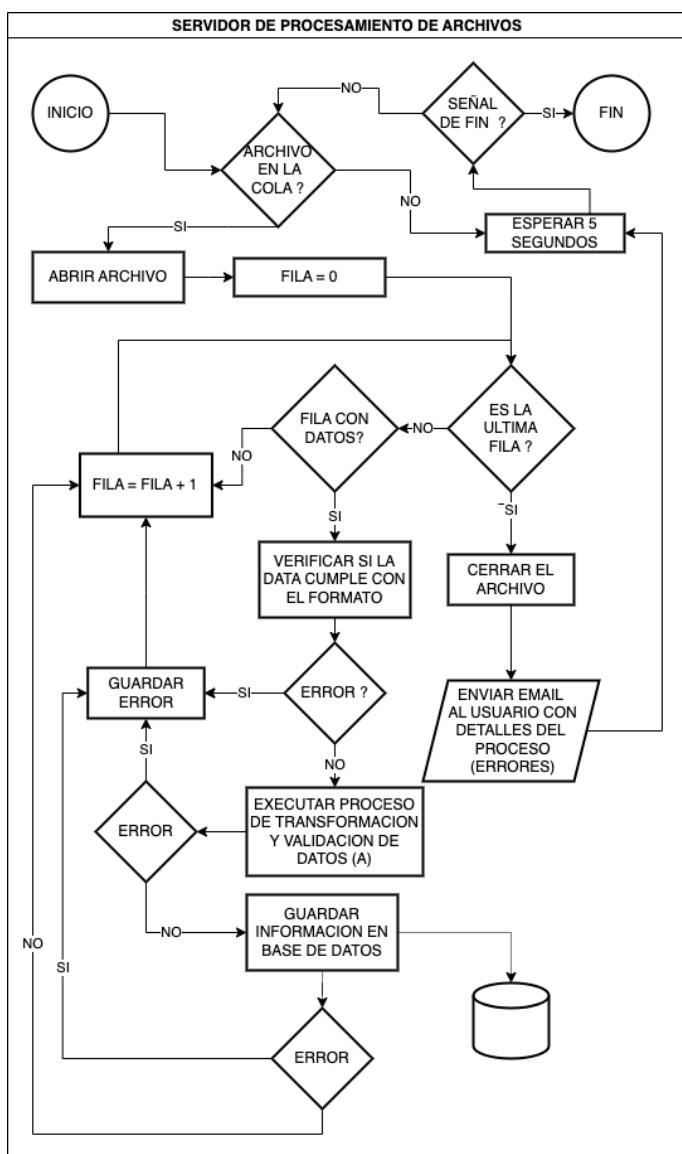
- **Web Server Proxy (Nginx):** Es el servicio web disponible públicamente y recibe las solicitudes HTTP(s) del cliente y las redirecciona para el WebServer excepto solicitudes de archivos estáticos (CSS, JS).
- **Web Server (Django):** Es el servidor web realizado con el lenguaje de programación Python, usando el framework Django, que contiene y ejecuta las reglas de negocio de las solicitudes HTTP(s).
- **WSGI (Web Server Gateway Interface):** Es la tecnología del lenguaje Python que permite la interconexión entre un servidor web tradicional (Nginx) y un servidor web *moderno* creado con frameworks en el ecosistema Python.
- **Sistema de Archivos:** Hace referencia al sistema de archivos del servidor.
- **Base de Datos (SQL Server):** RDBMS utilizado por el servidor web y workers para almacenar las informaciones en este caso se utiliza SQL Server Enterprise.
- **Celery + RabbitMQ:** Celery es una librería de python que permite la gestión y el manejo de workers, utilizando un broker de mensajería, en este caso RabbitMQ un software hecho en el lenguaje erlang que se encarga de orquestar, enviar y manejar los mensajes utilizando el protocolo AMQP.
- **Código de la Aplicación (Github):** Hace referencia al **sistema de gerenciamiento de versiones** donde se mantiene y actualiza el código de aplicación en este caso es utilizado GitHub al momento de realizar el documento.
- **Power BI:** Es el sistema de analítica utilizado por los usuarios para crear dashboards, reportes, etc, utilizando la información en el RDBMS, en este caso se utiliza una réplica para no impactar la instancia usada por el servidor Web.

Toda la arquitectura descrita anteriormente es ejecutada en un data-center interno de la empresa gestionada con OpenStack, responsabilidad de un equipo de infraestructura dedicado.

2.1 Servidor de Procesamiento de Archivos

El siguiente diagrama de flujo ilustra el proceso establecido para procesar un archivo.

Figura 11: Flujo del Servidor de procesamiento de Archivos de la Figura 10.



Fuente: Propia

Toda hoja de cálculo recibida por los workers descritos en la Figura 7, ejecuta el flujo descrito en la Figura 8, el cual describe un procesamiento lineal de las filas contenidas en las hojas de cálculo proporcionadas por el usuario, el cual podríamos dividirlo en 3 procesos:

- **Validación de formato:** mencionado en la acción ***VERIFICAR SI LA DATA CUMPLE CON EL FORMATO*** de la Figura 8, lo cual podemos denominar TypeChecking por lo que se valida el tipo de información en las celdas.
- **Transformación:** Se ejecutan las reglas de negocio definidas por los usuarios y se transforman los valores en base a estas, mencionada en la acción ***EXECUTAR PROCESO DE TRANSFORMACIÓN Y VALIDACIÓN DE DATOS*** de la Figura 8.
- **Carga de Datos:** mencionada en la acción ***GUARDAR INFORMACIÓN EN BASE DE DATOS*** de la Figura 8, hace referencia a la subida de la información ya procesada en el sistema de base de datos.

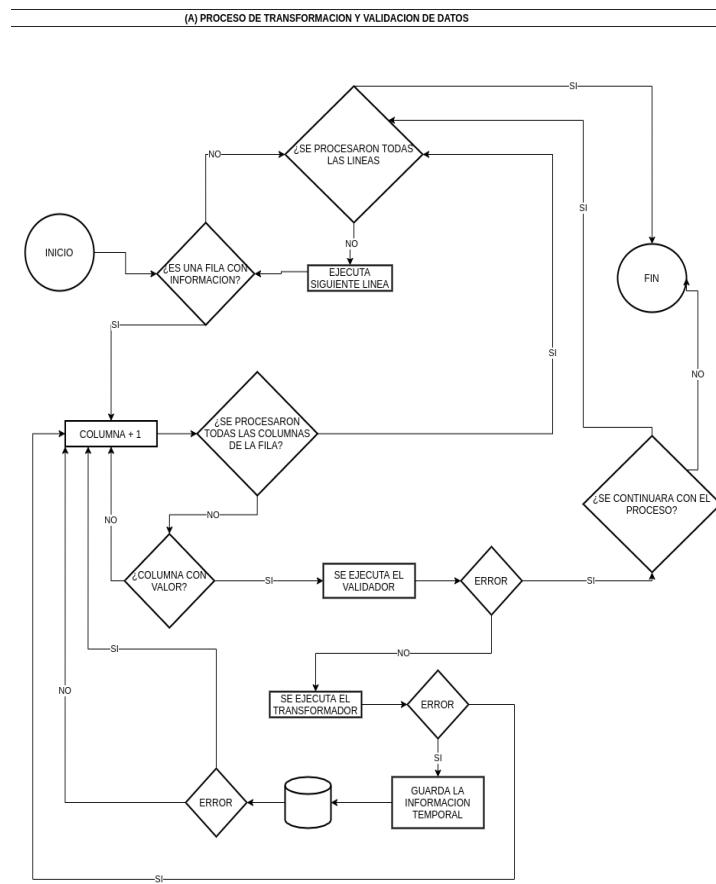
El flujo anterior es genérico para todas las hojas de cálculo en el sistema por lo que se creó una librería de python interna para este tipo de importadores llamada importer-data.

Algunas métricas de este sistema se han descrito en la Tabla 1, resaltando que el promedio de demora de las importaciones es de 35 minutos, debido a que el flujo genérico actual por archivo no cuenta con paralelización, es decir, los micro flujos internos (TypeChecking, Transformación y Carga de Datos) se ejecutan secuencialmente.

2.2 Flujo de Transformación de Datos

El siguiente diagrama de flujo ilustra el proceso establecido para ejecutar las validaciones y transformaciones contenidas en un archivo.

Figura 12: Flujo de Transformación de la Figura 11.



Fuente: Propia

El proceso de transformación hace referencia principalmente a la aplicación de lógica de negocios y validaciones de estas, en las hojas de cálculos, por ejemplo, imaginemos el siguiente escenario, el sistema procesa una hoja de cálculo con las siguientes columnas:

- 1 Identificación.
 - 2 Tipo de Empleado.
 - 3 Horas de trabajo.
 - 4 Salario.

Ciertamente el objetivo del importador además de validar el formato de las informaciones y cargar las informaciones en el sistema, debe generar la información de impuesto y horas extras trabajadas, basadas en el tipo de empleado y las horas de trabajo, este tipo de reglas en el procesamiento del archivo se ejecutan al momento de la transformación de los datos.

Además de lo anterior podrían existir reglas de transformación con dependencia entre columnas, ejemplo, el salario no podría pasar de \$5'000.000 millones para empleados de tipo individual.

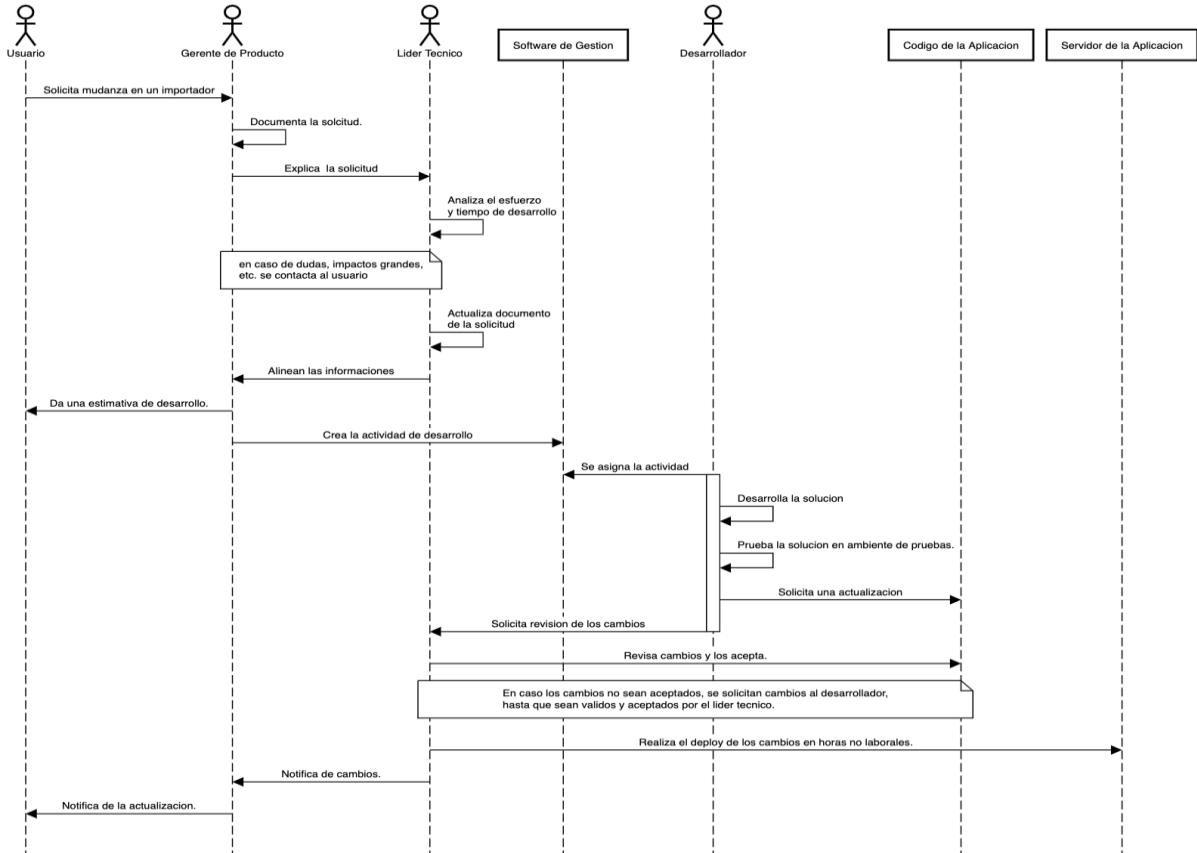
Por lo anterior, estas son realizadas a nivel de columna por cada fila de las hojas de cálculo, como ilustrado en el flujo de la Figura 9.

En resumen, este es uno de los procesos de mayor impacto en los tiempos de ejecución por lo que generalmente se intercomunica con otros sistemas para realizar este tipo de validaciones, ya sean base de datos externas, API 's, cálculos matemáticos, consultas en otras tablas de la base datos, etc.

2.2 Implementación de Cambios

En el siguiente diagrama de secuencia se representa gráficamente el proceso definido para llevar a cabo modificaciones en el sistema, incorporando decisiones relacionadas con producto, desarrollo y despliegue.

Figura 13: Flujo de actualización del sistema



Fuente: Propia

Todo sistema o plataforma web con usuarios, necesita tener establecido un framework o flujo de trabajo para la implementación de cambios en esta, varía mucho entre cada tipo de empresas, pero en el sistema actual esta actualización se realiza a partir de una solicitud del usuario que utiliza los importadores en la plataforma, ya sea porque mudo una regla del negocio o identificar un problema en el proceso de importación.

Por lo anterior, se podría definir 4 actores en este flujo que son:

- **Usuarios:** personas utilizando los importadores de la plataforma.
- **Gerentes de Proyecto:** persona encargada de la administración y gerencia de los clientes que utilizan los módulos de la plataforma.
- **Líder Técnico:** Profesional del área de computación y sistemas gestor de un grupo de desarrolladores.

- **Desarrollador(es):** Profesionales encargados de la implementación, actualización y manutención de los módulos de la plataforma gestionados por un líder técnico.

El cual inicia de una solicitud del usuario hacia el gerente de proyecto, como descrito en la Figura 10.

Para concluir, el **Tiempo de desarrollo al implementar una mejora** de 2 semanas descrito en la Tabla 1, está directamente relacionado al tiempo de actualización de código, pruebas y aprobación que tiene el flujo actual para la implementación de un cambio (Figura 10).

Hasta una actualización de TypeChecking podría llevar en torno de 1 semana o 2 dependiendo, un ejemplo de esta puede ser:

El usuario percibe que el importador está aceptando valores de tipo texto en una columna que solamente debería aceptar valores de tipo numéricos

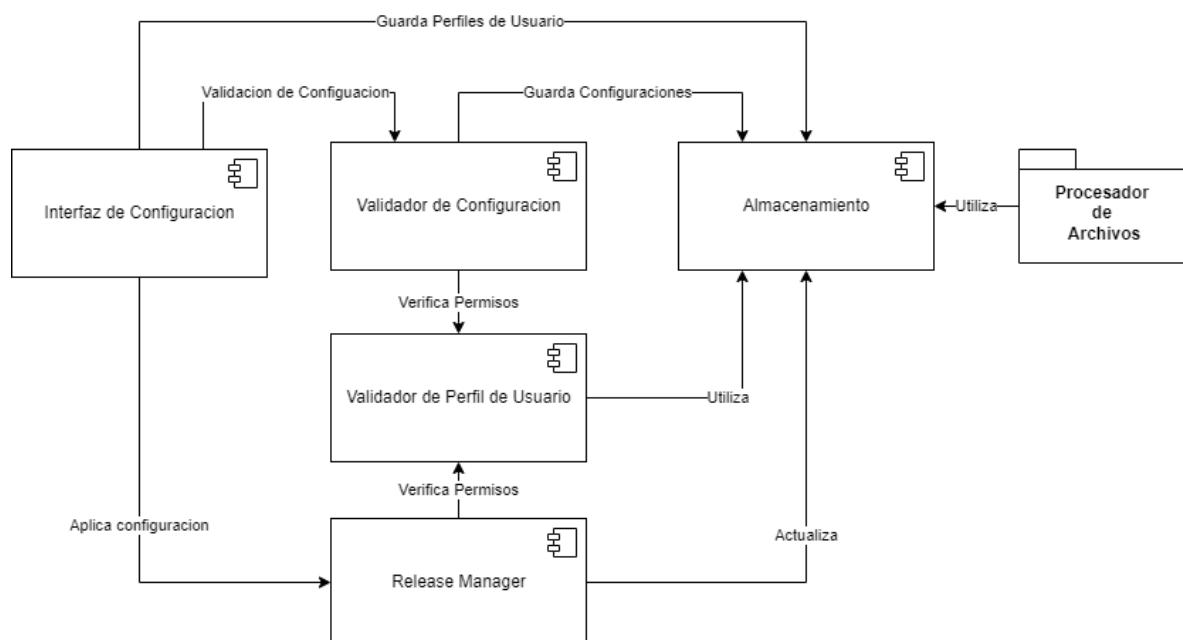
El desarrollador necesitará actualizar el código base, verificar que todo se mantenga funcionando estable, solicitar aprobación, realizar pruebas automáticas y realizar el deploy en horas no laborales para no impactar el uso de la herramienta, este último es responsabilidad del líder técnico junto con el equipo de infraestructura.

Capítulo 3: AUTOMATIZACIÓN DE LAS VALIDACIONES DE TIPO DE DATOS (TYPECHECKING)

Se plantea la creación de una interfaz web que posibilite la definición de reglas de validación de tipos y formatos para columnas específicas en hojas de cálculo con el propósito de facilitar las actualizaciones mediante un panel o dashboard en línea. Esto contribuirá a evitar la sobrecarga operativa relacionada con el desarrollo, las pruebas y la implementación en la infraestructura del servicio de validación, tal como se detalla en la Figura 13.

En el siguiente diagrama representa gráficamente los componentes propuestos para automatizar la validación de tipo de datos.

Figura 14. Componentes del Sistema para Interfaz de TypeChecking



Fuente: Propia

3.1 Sistemas de TypeChecking

Los sistemas de typechecking se describen normalmente a través de un formalismo particular basado en afirmaciones llamadas juicios como fue mencionado por Cardelli, L. (1996) en Type Systems, En este caso estos **juicios de tipos** definirán las reglas que una columna A deberá cumplir con un tipo B.

- Números Enteros Positivos y Negativos.
- Números Decimales Positivos y Negativos.
- Texto (Con/Sin patrones).
- Fechas (Con/Sin tiempo).

Por lo cual, se debe saber los tipos de datos que normalmente utilizan estas hojas de cálculo, descritos en la lista anterior descrita por ERWIG, Martin. Typed Table Transformations.

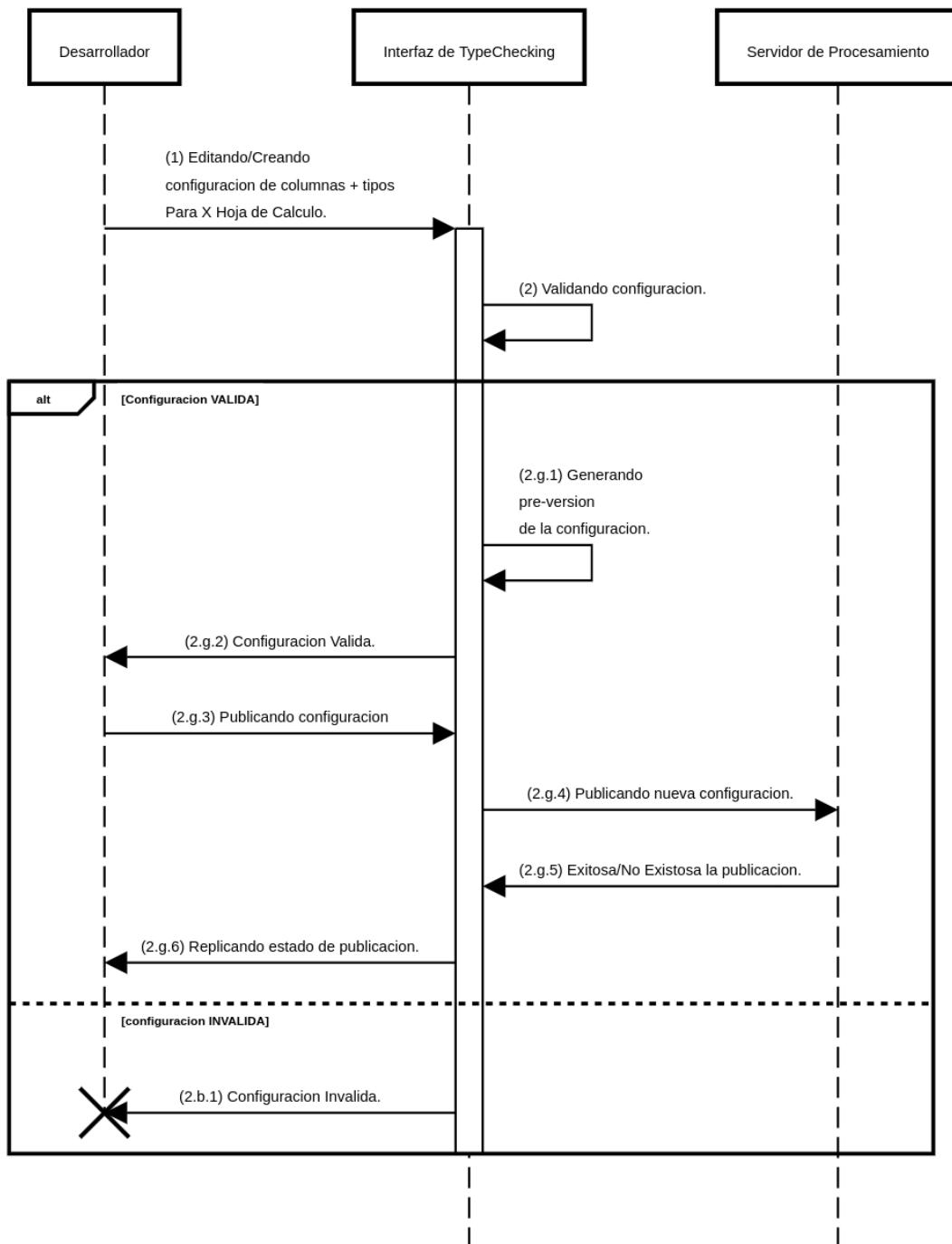
Por lo anterior, este sistema funcionará para validar columnas de hojas de cálculo con una estructura tabular, donde los datos están representados en una tabla de 2 dimensiones (filas y columnas) que representan datos (ERWIG, Martin. Typed Table Transformations), y estas columnas tienen asociado un tipo.

En consecuencia, está direccionado a hojas de cálculo que almacenan información tipada de forma tabular en la totalidad de la hoja (1 hoja representa 1 tabla tipada).

3.2 Nuevo Flujo de Trabajo con la Interfaz

En el siguiente diagrama flujo ilustra el proceso propuesto para llevar a cabo la actualización de datos, habilitando la validación de tipos de datos.

Figura 15. Flujo de Desarrollo de mudanzas en reglas de validación



Fuente:

Propia

Un desarrollador podrá configurar de forma fácil e intuitiva los juicios de tipos que deben cumplir las columnas de cierta hoja de cálculos, algunas podrían ser hojas de cálculo con información de transportes marítimos, carros a venta en una

concesionaria, lista de productos en un almacén, etc, haciendo uso de los componentes mostrados en la Figura 7.

Estas serán validadas por el servidor de procesamiento (Figura 2), el cual utilizará las reglas de validación configuradas, validadas y publicadas por el desarrollador, como mostrado en la Figura 8.

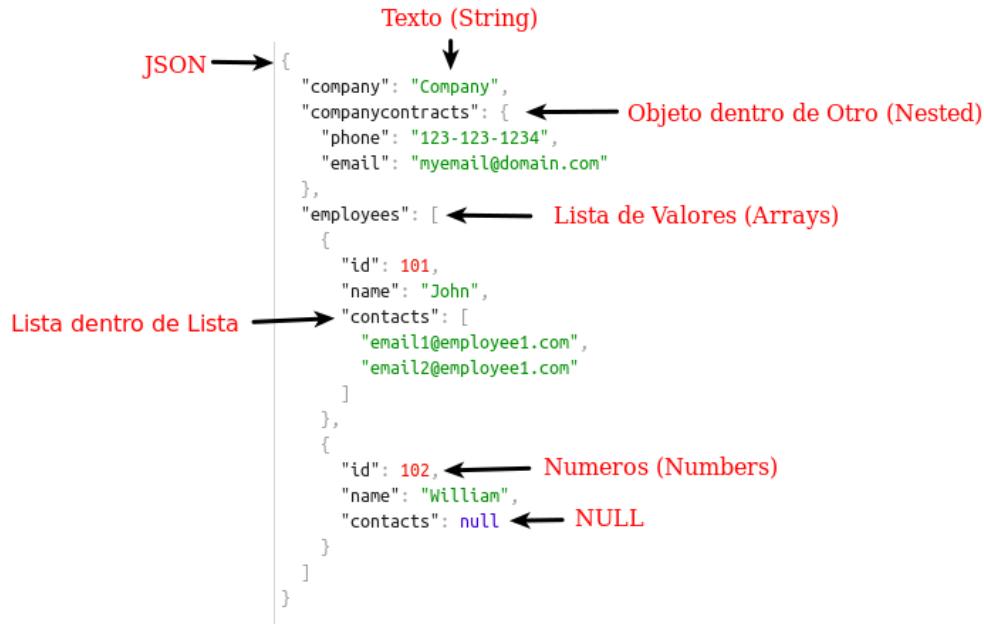
Mencionadas configuraciones para cierto tipo de hojas de cálculo, ayudará al servidor de procesamiento a ejecutar de una forma rápida la validación de tipos sobre los datos en las filas de la hoja de cálculos, y en caso de se presentar una falla por la nueva configuración el componente de release (Figura 7) se encargará automáticamente restaurar la última configuración estable.

Lo anterior será posible debido al formato en el cual se propone almacenar y publicar estas configuraciones utilizado por el componente de Validación de Configuración (Figura 7), el cual es **JSON Schema**.

3.3 JSON Schema & JSON

La siguiente imagen describe la estructura de un archivo de formato json

Figura 16. Ejemplo de un JSON



Fuente: Propia

JSON Schema es un lenguaje para declarar la estructura de un JSON válido (Francis Galiegue and Kris Zyp. 2013), este último (JSON) mostrado en la figura 9, es un formato de serialización de datos que es ampliamente adoptado para almacenar informaciones en disco o enviar data sobre la red (Francis Galiegue and Kris Zyp. 2013).

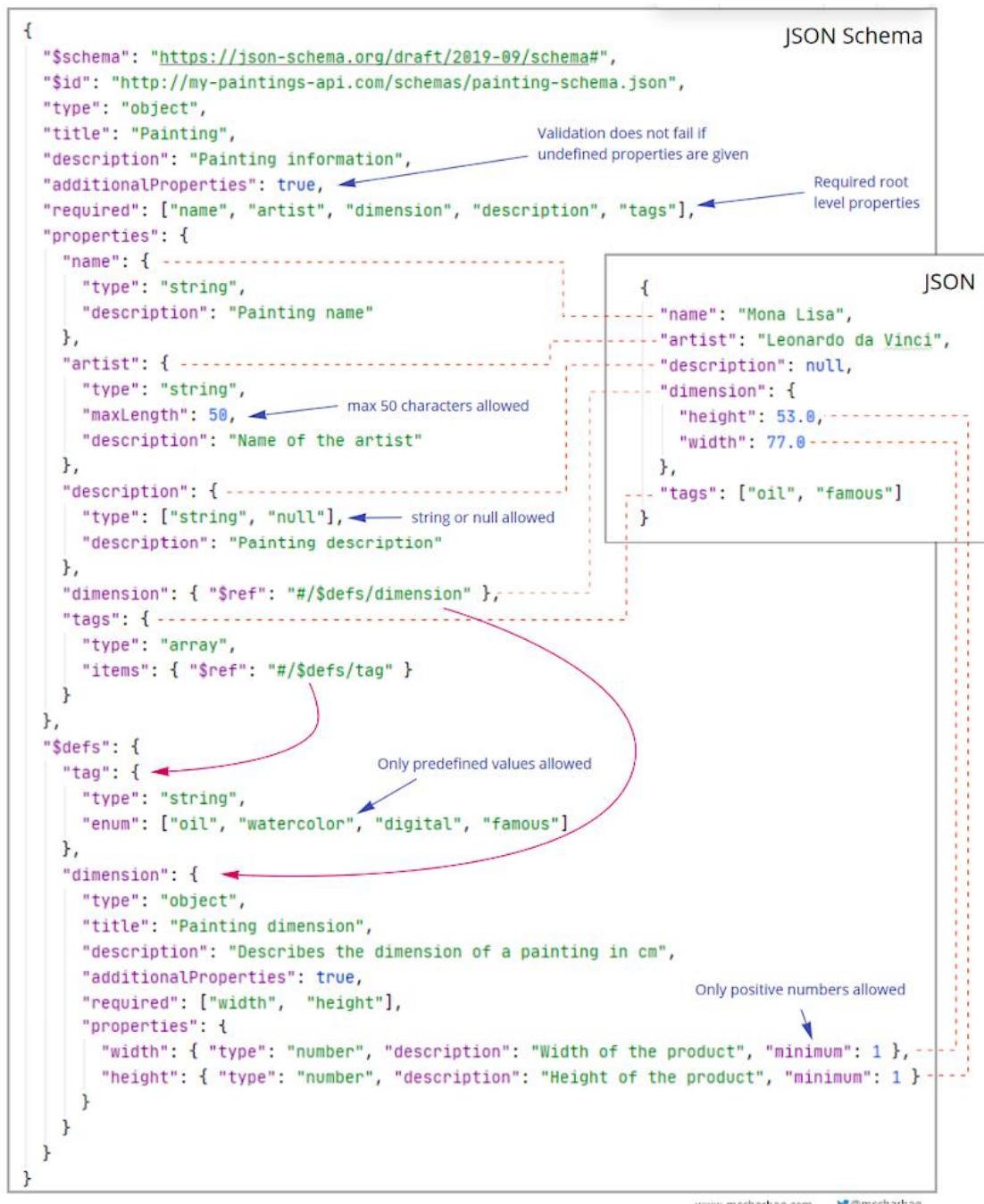
El cual soporta varios tipos de datos primitivos como:

- Texto (strings)
- Números (numbers)
- Booleanos (Booleans)
- NULL
- Lista de valores (Arrays) - Anidados
- Objetos (Objects) - Anidados

Por lo tanto, es compatible con los tipos de datos que se validaron en las hojas de cálculo configuradas en la interfaz.

La siguiente imagen describe el esquema de un archivo de formato json

Figura 17. Ejemplo de un JSON Schema.



www.mscharhag.com @mscharhag

Fuente: <https://www.mscharhag.com/api-design/json-schema> - 21/06/2022

Débito a que JSON Schema también utiliza el formato JSON para describir estructuras como mostrado en la figura 10 y es un Draft del IETF (Internet Engineering Task Force) en continua evolución, actualmente en el draft-2019-09 al momento de escribir el documento y el propuesto a utilizar es el Draft-04 (Francis Galiegue and Kris Zyp. 2013. JSON Schema draft 04), que es el más adoptado globalmente.

Al ser una especificación normalizada (Felipe Pezoa, etc 2016. at Foundations of JSON Schema), cuenta con una serie de algoritmos validadores listos para ser usados en diferentes lenguajes (JSON Schema Validators). los cuales verifican si un documento JSON **J** cumple la estructura definida en un esquema **S**, a este problema se le denomina **JSON Schema validation** el cual resolverá nuestro Procesador de Archivos (Figura 7) para cada JSON de las líneas de un archivo.

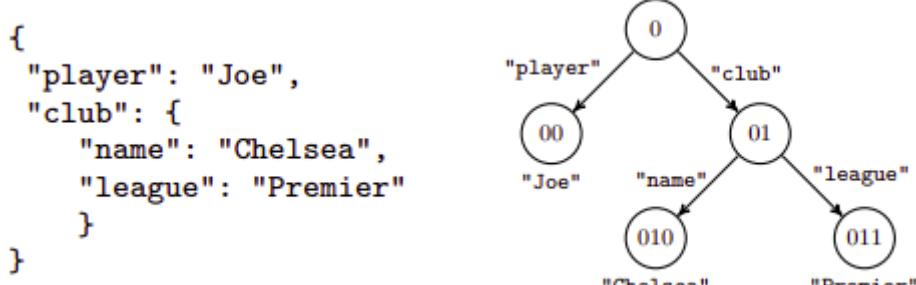
3.4 JSON Schema Validation (Problema de Validación)

es un problema computacional que es resuelto con algoritmos o programas conocidos como validadores, y es demostrado que el problema está siempre en **PTIME** y puede ser resuelto en **tiempo lineal** con respecto tanto al documento (J) y al schema (S) siempre que el schema no utilice **uniqueItems** (Felipe Pezoa, etc 2016. at Foundations of JSON Schema).

Aclarando que PTIME hace referencia a una **clase de complejidad** algorítmica, indicando que los problemas de decisión pueden ser resueltos en tiempos polinomios por una máquina de turing determinista (P (clase de complejidad) - 2022, 20 de junio), como lo es el problema de decisión: **JSON Schema validation**.

Y que los algoritmos de validación se ejecutan en tiempo O(J.S) (Felipe Pezoa, etc 2016. at Foundations of JSON Schema), ya que nuestros JSON Schemas no contarán con la presencia de un uniqueItems por lo que el tipo de dato Array no es soportado en las hojas de cálculo, sin embargo, en el peor escenario del uso de **uniqueItems** la complejidad se mantiene igual (Felipe Pezoa, etc 2016. at Foundations of JSON Schema).

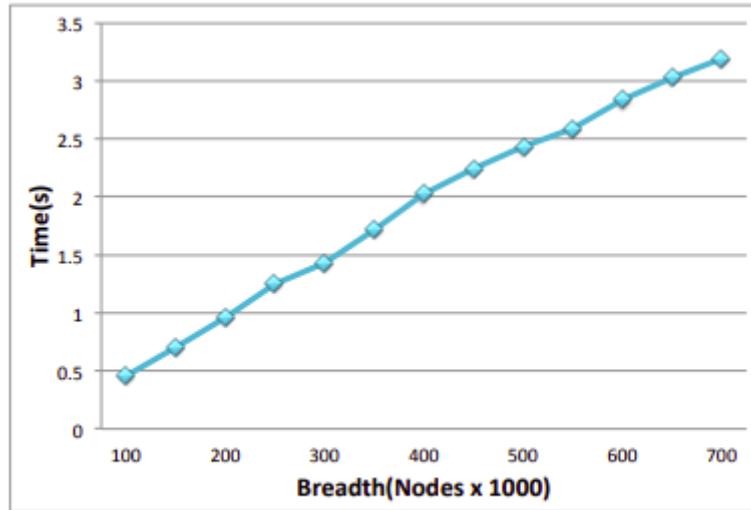
Figura 18. Un JSON (J) y la representación como árbol ordenado y sin clasificar T(J)



Fuente: Foundations of JSON Schema, Figura 4 (Felipe Pezoa, etc 2016.)

Un JSON (sin array) puede ser representado en un árbol sin orden ni clasificación como mostrado en la Figura 11, por esta razón, es válido afirmar que los JSON generados a partir de información tabular de las líneas de una hoja de cálculo serán árboles con múltiples hijos teniendo una altura máxima de 1.

Figura 19. Prueba de rendimiento sobre validaciones de tipos sobre atributos de JSONs



(c) Trees with multiple children

Fuente: *Foundations of JSON Schema, Figura 5.c (Felipe Pezoa, etc 2016.)*

Por supuesto, el número de columnas de una hoja de cálculo afectaría el tiempo de validación de las líneas y consecuentemente de todo el archivo, debido a que aumentara el número de atributos que el JSON tendrá y por ende validarán, tal cual como sucede en los sistemas de hoy.

Pero el rendimiento y tiempo de las validaciones es significativamente aceptable como mostrado en la *Figura 12*, menos de 3 segundos para 1 fila que pertenecería a un archivo con 700 columnas, además JSON schema incrementa la interoperabilidad, mensurabilidad y estandariza el lenguaje utilizado en la definición de validaciones, enfatizando nuevamente en la posibilidad de ser paralelizable.

3.5 JSON Schema: Implementación

Para utilizar JSON Schema sobre las hojas de cálculos a ser procesadas, la información de cada línea deberá ser transformada, generando un JSON similar al mostrado en la figura 8, con la excepción de que tendrá solamente un nivel, es decir, solo tendrá una llave (nombre del header de la columna) y un valor (valor de la celda de la fila transformada), no contará con tipos anidados complejos (Arrays, Objetos).

Por lo que el procesador de archivos, a partir de una hoja de cálculo generará varios JSON que contendrán y representarán el valor de cada fila del archivo original, lo que nos permitirá poder procesarlos de forma individual y paralelizable.

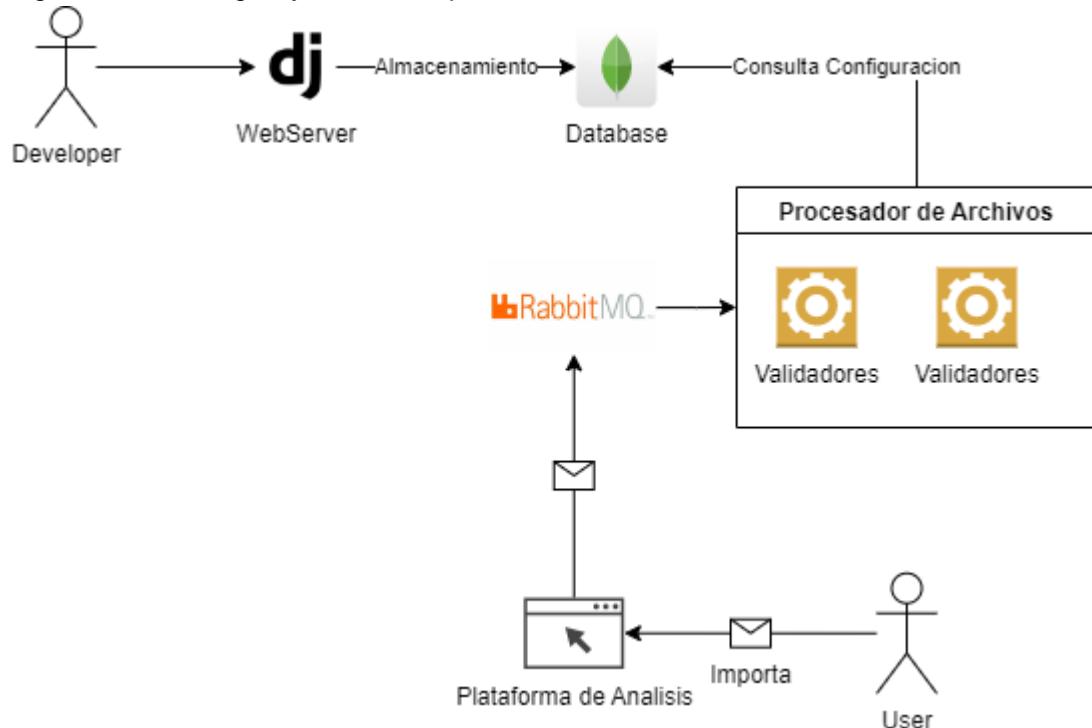
Debido a que las validaciones de tipo no cuentan con dependencias entre filas o columnas, simplemente se validarán el tipo del valor en una celda que pertenece a una columna a la que le fue definida un tipo en la configuración.

Por lo cual el sistema contará con un flujo de versionamiento como se puede observar desde el paso 2.g.1 hasta el 2.g.6 de la *Figura 7*, administrado por el componente de Release Manager (Figura 6), para el control de la publicación simultáneas de nuevas reglas por parte de múltiples desarrolladores y un sistema automático de validación de JSON Schemas ejecutado en el paso 2 de la *Figura 7* para solamente aceptar JSON Schemas válidos hechos por los desarrolladores.

3.6 Arquitectura Macro de la Interfaz Web

En la imagen siguiente se detallan los componentes y tecnologías que integran tanto la interfaz como el servicio web.

Figura 20. Tecnologías y Macro Componentes de la Interfaz Web.



Fuente: Propia

A continuación se describen las tecnologías y macro-componentes que utilizará la Interfaz de TypeChecking descritas en la *Figura 13*.

- **WebServer:** Servidor Web que controlara toda la lógica de la interfaz y almacenará de manera persistente las versiones de las reglas de validación, conteniendo los componentes (Figura 7):
 - Interfaz de Configuración.
 - Validador de configuración.

- Validador de perfiles de usuario.
- Release Manager.
- **Database:** Base de datos para almacenar los JSON Schema y demás recursos, en este caso en particular se propone MongoDB una base de datos NOSQL, que sería el componente de Almacenamiento (Figura 7)
- **Procesador de Archivos:** Cluster de validadores que recibirán el archivo importado por el usuario, consultará la configuración definida por el desarrollador para ese tipo de archivos y realizará las validaciones pertinentes.

Los demás componentes como rabbitmq y el flujo de importación del archivo continúan siendo los mismos descritos en la Figura 3.

A fin de continuar utilizando los beneficios de JSON Schema incluso a nivel de base de datos, se propone la utilización de MongoDB una base de datos basada en documentos JSON, que pueden beneficiarse de JSON Schema (Felipe Pezoa, etc 2016), ya cuentan con la tecnología mencionada integrada (MongoDB Schema Validation), lo que permite la fácil definición de reglas para validaciones de las configuraciones.

En la imagen siguiente se presenta el esquema de datos del documento que almacena la información relacionada con la validación del tipo de dato.

Figura 21: Descripción del documento que almacenará los json_schemas configurados.

importer_typecheck_schema
id (index)
file_type (index)
import_name
import_slug (index)
dev_profile
active_schema: {...json_schema_v3}
schemas_releases: [...json_schema_v1, ...json_schema_v2]}

Cada json_schema contara con la informacion del desarrollador que lo realizo.

Fuente: propia

El cual permitirá almacenar todos los schemas definidos para un importador en formato JSON como ilustrado en la Figura 14, lo que permitirá flexibilidad y consultas rápidas al momento de verificar cual es el JSON Schema activo a ser usado en la validación de los archivos.

3.7 Implementación de la Interfaz Web

Al implementar la interfaz web para configuraciones de validaciones de tipo sobre las hojas de cálculo a ser procesada por los importadores, esta es transparente para el usuario por lo que no muda la forma en que el usuario interactúa con la herramienta de análisis para hacer sus importaciones, ya que todo se mantiene exactamente igual.

El impacto se lleva a cabo en los desarrolladores de la plataforma que mantienen los importadores, con la interfaz web al momento de implementar un cambio o configurar las validaciones de tipo de un nuevo importador, el desarrollador solamente tendría que realizar los siguientes pasos:

- 1 Editar la configuración del importador en la interfaz de typechecking.
 - 1.a Se creará en caso no exista y debe definir el nombre y un archivo de pruebas como ejemplo.
 - 1.b Caso sea necesario proporcionará un archivo de test nuevo.
- 2 Guardar la configuración del importador y esperar que la validación de esta finalice.
- 3 Esperar los resultados de la ejecución de las validaciones sobre el archivo de prueba.
- 4 Si todo está bien, procede a hacer el release o activación del importador.

Lo anterior es ejecutado en cuestiones maximo 2 horas, a diferencia de 2 semanas que podria tardar el flujo anterior (Tabla 1), ya que este necesitaba los siguientes pasos descritos en la Figura 10:

- 1 Actualización del código base de la plataforma de análisis.
- 2 Hacer pruebas de los cambios y verificar que lo existente aun se mantuviera estable.
- 3 Subir los cambios al sistema de control de versiones.
- 4 Realizar la actualización del código de producción de la plataforma de análisis.
- 5 Esperar la implementación de sus cambios.

De los cuales los más demorados son el paso 1 y 2, debido a que la interfaz web se encarga directamente de activar o no el importado para la plataforma de análisis no

se requiere un esfuerzo del desarrollador para mencionada tarea, solamente mantener el monitoreo usual de los importadores.

3.8 Análisis de Resultados

La interfaz web para configuración de TypeChecking ha disminuido drásticamente el tiempo de actualizaciones y desarrollo del proceso de validación de los importadores para la plataforma de análisis, llevando este de ser semanas para horas.

Además de lo anterior, al ser un servicio completamente aislado de la plataforma de análisis, es decir, sin depender de esta plataforma, el servicio puede ser extendido a otras herramientas internas de la empresa, mejorando los tiempos de desarrollo en estos sistemas por igual.

Claramente, como todo sistema tiene sus ventajas y desventajas, y una de las que más resalta en el sistema de typechecking es la complejidad agregada para todo el ecosistema que los ingenieros necesitan se preparar para mitigar.

Al permitir flexibilidad y dinamismo en las configuraciones de importadores siendo almacenadas como JSON Schemas en un sistema de base de datos, siendo un **SSOT** (*Single source of truth. In Wikipedia*), como ingenieros, debemos mitigar pérdidas de información, evitar brechas de aseguranzas tanto a nivel de base de datos como en el servicio web, definir políticas para los desarrolladores solamente editar/crear configuraciones bajo autorización, asegurar una alta disponibilidad del servicio.

Al ser un servicio crítico para otras herramientas, debe permanecer casi siempre disponible, y se deben preparar contramedidas para una posible indisponibilidad del servicio, ya que con seguridad en algún momento él quedará indisponible por un periodo de tiempo, no hay sistema perfecto.

Estas pueden ser mitigadas, implementando backups recurrentes, sistema de permisos en la interfaz, incluir sistema de monitoreo del servicio, métricas, y definir varios nodos atendiendo la comunicación con otras herramientas.

Aunque agregue complejidad al sistema, este reduce los tiempos de implementación de nuevos cambios de una forma intuitiva y extensible, permitiendo que los esfuerzos de los desarrolladores, sean dirigidos a sistemas y funcionalidades más importantes dentro de la empresa.

En conclusión, la interfaz de TypeChecking podría disminuir los tiempos de implementación de los cambios en los importadores en relación a las validaciones de tipos en las hojas de cálculo procesadas por estos, e impactando no solamente

positivamente a la plataforma de análisis sino también a otras posibles herramientas que quieran implementar importadores.

La siguiente es una tabla comparativa entre el sistema actual y el propuesto, detallando los criterios comparables respecto al presente capítulo.

Criterio	Sistema actual	Sistema propuesto
Modificabilidad	Se necesitan múltiples pasos, ya que un desarrollador con conocimientos en el lenguaje en el que está escrita la herramienta debe crear una solicitud de extracción (pull request) o de fusión (merge request) para introducir nuevas validaciones o actualizar las existentes.	Los cambios se realizan a través de una interfaz gráfica por un miembro del equipo de soporte, lo que hace el proceso de modificabilidad simple.
Control de cambios	Se utiliza Git y un repositorio de código para mantener un registro de estos cambios. Sin embargo, dado que es una herramienta que principalmente usan los equipos de ingeniería, no está disponible para los equipos de soporte.	El control de cambios en las validaciones se puede consultar a través de la interfaz de usuario (UI), y está accesible para todos.
Disponibilidad	Debido a que estos cambios se realizan a través de código, existe la posibilidad de introducir errores en el proceso de creación o actualización de los mismos.	Debido a que el cambio se introduce mediante la interfaz de usuario (UI) no existe posibilidad de introducir errores críticos.
Tiempo de implementación	Debido a la cantidad de cambios, el proceso requiere más tiempo y abarca varias etapas, que incluyen la	El proceso de implementación es considerablemente más rápido, ya que es llevado a cabo y validado por un

	<p>implementación, la creación de propuestas de cambios en el repositorio de código, la espera de aprobación, la validación de cambios en el entorno de pruebas y, finalmente, el despliegue.</p>	miembro del equipo de soporte a través de la interfaz de usuario (UI).
--	---	--

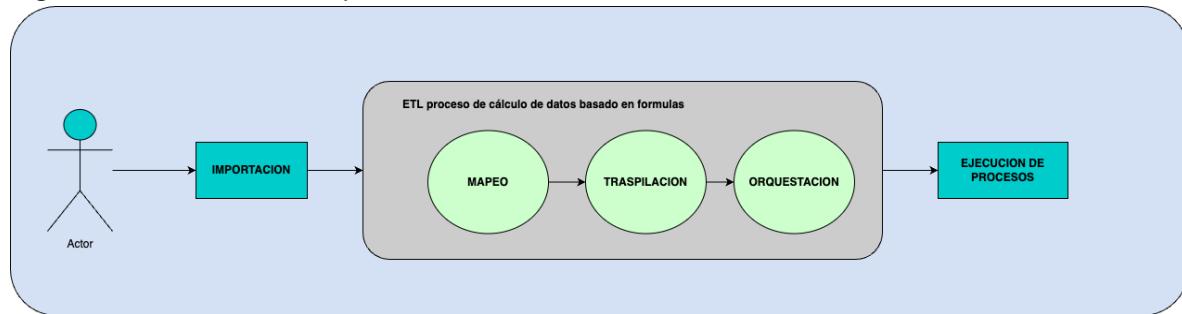
Capítulo 4: AUTOMATIZACIÓN DEL PROCESAMIENTO Y EJECUCIÓN DE FÓRMULAS

En este capítulo se propone un sistema de mapeo, transpiración y orquestación de procesos que permitirá realizar los cálculos definidos en una hoja de cálculo utilizando fórmulas de excel, de tal forma que se podría mapear cada proceso definiendo un plan de ejecución y consigo escalando la ejecución utilizando la nube.

Las hojas de cálculo son herramientas muy útiles para almacenar, analizar y correlacionar datos, pero se encuentran limitadas por los recursos disponibles, ya que en mayormente son aplicación desktop como microsoft excel, con la propuesta definida se propone eliminar la mayor parte de tiempos de espera que muchos colaboradores experimentan hoy, ya que diariamente deben realizar por ejemplo de análisis financieros para estatus gerenciales, seguimiento de productos, etc.

Para dar solución a esto nos enfrentamos a los siguientes retos técnicos representados en la siguiente imagen.

Figura 22: Proceso de ETL para cálculo de datos basado fórmulas



Fuente: Propia

4.1 Mapeo de fórmulas

Parte de la solución es no interrumpir ni modificar el proceso que hoy el colaborador sigue para estructurar sus datos, ejemplo de esto es definir tablas, sub-hojas de cálculo y fórmulas en una hoja o muchas.

Por esta razón se define una fase de mapeo en la que el artefacto de entrada será el documento que contiene las hojas de cálculo definidas por el colaborador y la única acción a realizar por parte de éste será la subida al sistema de procesamiento.

Para lograr esto se definen los siguientes **casos de mapeo** y los componentes involucrados.

4.1.1 Casos de mapeo: Una hoja de cálculo - Sin fórmulas

Figura 23: Hoja de cálculo sin fórmulas

A2	fx	Angel
	A	B
1	Name	Age
2	Angel	19
3	Pedro	10
4	Jose	12
5	Carlos	30

Fuente: Propia

Este es el más simple de los casos donde se verá involucrado el componente de importación de datos a una base de datos, no será necesario ningún mapeo de fórmulas, ya que son datos crudos que posiblemente serán utilizados para un proceso posterior, como es mostrado en la Figura 16.

4.1.2 Casos de mapeo: Más de una hoja de cálculo - Sin fórmulas

Figura 24: Múltiples hojas de cálculo con información pero sin fórmulas

B2	fx	19	
	A	B	C
1	Name	Age	
2	Angel	19	
3	Pedro	10	
4	Jose	12	
5	Carlos	30	

+ Hoja 1 Hoja 2

Fuente: Propia

Este es un caso similar al anterior donde se verá involucrado el componente de importación de datos a una base de datos, pero teniendo en cuenta que la información subida debe estar categorizada, ya que son diferentes hojas de cálculo con diferentes datos.

Tal como el paso anterior no será necesario ningún mapeo de fórmulas, ya que son datos crudos que posiblemente serán utilizados para un proceso posterior, como es mostrado en la Figura 17.

4.1.3 Casos de mapeo: Una hoja de cálculo - Con fórmulas

En este caso nos encontramos con datos contenidos en una hoja de cálculo y además con la presencia de alguna fórmula calculando los datos de una o más celdas.

Por lo tanto, se podrían presentar varios casos dentro de esta

4.1.3.1 Fórmulas independientes

Figura 25: Fórmula de excel no correlacionada

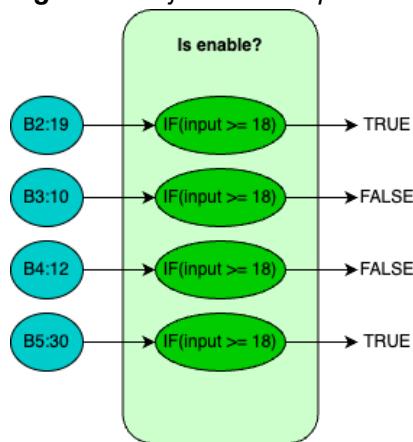
C2	A	B	C
1	Name	Age	Is enable?
2	Angel	19	TRUE
3	Pedro	10	FALSE
4	Jose	12	FALSE
5	Carlos	30	TRUE

Fuente: Propia

Este caso hace referencia a una o más fórmulas no correlacionadas generando un resultado basado en una o más celdas que no contienen una fórmula, por lo tanto generando procesos no codependientes.

Como resultado la hoja de cálculo genera el plan de ejecución representado en la siguiente imagen.

Figura 26: Ejecución de procesos generados para realizar el cálculo de la columna C(Is enable?)



Fuente: Propia

En este caso se generan 4 procesos que recibiendo como entrada la información de la columna B generan el resultado de la columna C.

4.1.3.2 Fórmulas codependientes

En este caso nos encontramos con fórmulas code-dependientes lo que significa que una fórmula tiene como entrada la información de una celda que a la vez contiene otra fórmula, por lo tanto debe existir un plan de ejecución ordenado para que una vez la primera fórmula genere el resultado la segunda lo reciba como entrada.

A continuación, se puede ver un ejemplo de este caso.

Figura 27: Fórmula excel definiendo si la edad es mayor o igual a 18.

C2	A	B	C	D
1	Name	Age	Is enable?	Should send ticket?
2	Angel	19	TRUE	TRUE
3	Pedro	10	FALSE	FALSE
4	Jose	12	FALSE	FALSE
5	Carlos	30	TRUE	TRUE

Fuente: Propia

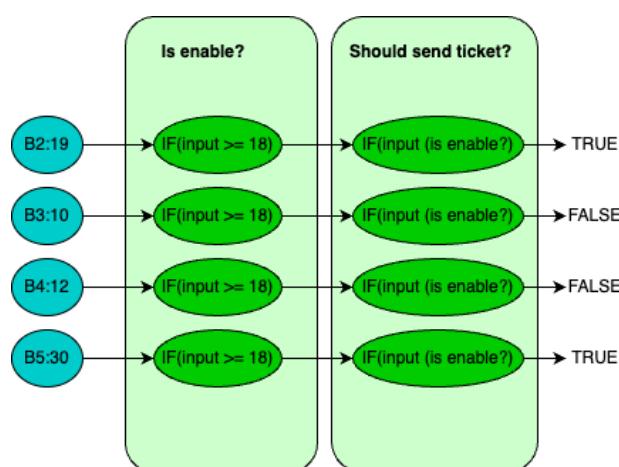
Figura 28: Fórmula excel definiendo si debería ser enviado el tiquete basado en la columna C, que está siendo calculada con otra fórmula de excel.

D2	A	B	C	D
1	Name	Age	Is enable?	Should send ticket?
2	Angel	19	TRUE	TRUE
3	Pedro	10	FALSE	FALSE
4	Jose	12	FALSE	FALSE
5	Carlos	30	TRUE	TRUE

Fuente: Propia

Como resultado la hoja de cálculo genera el plan de ejecución representado en la siguiente imagen.

Figura 29: Ejecución de procesos generados para realizar el cálculo de la columna D(Should send ticket?)



Fuente: Propia

En este caso se generan inicialmente 4 procesos para dar resultado a la columna B(is enable?) Después de esto es posible generar 4 nuevos procesos para calcular la

columna **C(Should send ticket?)** Por lo tanto fueron necesarios 8 procesos para dar el resultado teniendo los primeros 4 como dependencias de los próximos 4.

4.1.4 Casos de mapeo: Más de una hoja de cálculo - Con fórmulas

En este caso nos encontramos con datos contenidos en varias hojas de cálculo y justo a esto fórmulas que podrían estar correlación de datos entre una o varias hojas de cálculo; por lo tanto, se generan dos casos.

4.1.4.1 Fórmulas relacionadas en una misma hoja de cálculo

Este caso es similar al caso anterior con la única diferencia que podrían existir otras hojas de cálculo conteniendo información no relacionada a la hoja que está siendo calculada. Por lo tanto, la hoja calculada se importará y se mapean las fórmulas y la demás se importaran o se mapean sus fórmulas según el caso.

4.1.4.2 Fórmulas relacionadas en múltiples hojas de cálculo

En este caso nos encontramos con varias hojas de cálculo y entre estas fórmulas está relacionado información de una hoja para generar el resultado en otra hoja; por lo tanto, se derivan dos nuevos casos.

Fórmula con información de otra hoja de cálculo

En este caso nos encontramos con una fórmula recibiendo como entrada la información de otra hoja de cálculo que no contiene una fórmula en la celda.

A continuación, se puede ver un ejemplo de este caso.

Figura 30: Hoja 1 de excel con información de usuarios

	B2	fx	19
1	Name	Age	
2	Angel	19	
3	Pedro	10	
4	Jose	12	
5	Carlos	30	
6			
7			
8			
9			

+ Hoja 1 Hoja 2

Fuente: Propia

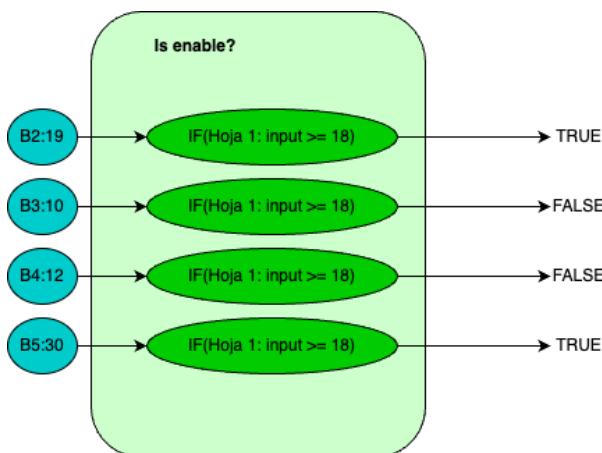
Figura 31: Hoja de cálculo con fórmula consumiendo información de la Hoja 1.

A2		<i>f(x)</i>	=IF('Hoja 1'!B2>=18, TRUE, FALSE)	
	A	B	C	D
1	Is enable?			
2		TRUE		
3		FALSE		
4		FALSE		
5		TRUE		
6				
7				
8				
9				

Fuente: Propia

Como resultado la hoja de cálculo genera el plan de ejecución representado en la siguiente imagen.

Figura 32: Ejecución de procesos generados para realizar el cálculo de la columna Hoja2:A(Is enable?).



Fuente: Propia

En este caso se generan 4 procesos recibiendo como entrada la información de la columna B de la **Hoja 1** generando el resultado de la columna A de la **Hoja 2**.

Fórmula recibiendo información de otra forma en otra hoja

En este caso nos encontramos con varias hojas de cálculo y en ellas fórmulas relacionadas con celdas de otra hoja de cálculo que a la vez contiene una fórmula.

A continuación, se puede ver un ejemplo de este caso.

Figura 33: Hoja 1 conteniendo información de usuario y calculando la columna Hoja1:C(is enable?)

C2	A	B	C
1	Name	Age	Is enable?
2	Angel	19	TRUE
3	Pedro	10	FALSE
4	Jose	12	FALSE
5	Carlos	30	TRUE
6			

+ ⌂ Hoja 1 ▾ Hoja 2 ▾

Fuente: Propia

Figura 34: Hoja 2 calculando columna A(should send ticket?) basándose en el resultado de la columna Hoja1:C(is enable?).

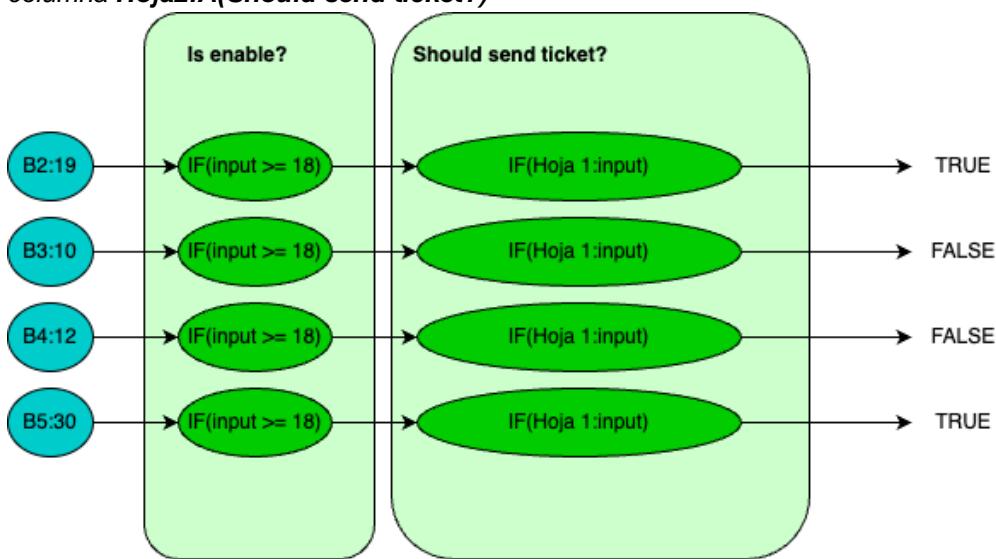
A2	A	B	C
1	Should send ticket?		
2	TRUE		
3	FALSE		
4	FALSE		
5	TRUE		
6			

+ ⌂ Hoja 1 ▾ Hoja 2 ▾

Fuente: Propia

Como resultado la hoja de cálculo genera el plan de ejecución representado en la siguiente imagen.

Figura 35: Ejecución de procesos generados entre diferentes hojas para realizar el cálculo de la columna Hoja2:A(Should send ticket?)



Fuente: Propia

En este caso se generan inicialmente 4 procesos para dar resultado a la columna **B(is enable?) de la Hoja 1**, Después de esto es posible generar 4 nuevos procesos para calcular la columna **C(Should send ticket?) de la Hoja 2**, Por lo tanto fueron necesarios 8 procesos para dar el resultado teniendo los primeros 4 como dependencias de los próximos 4.

4.2 Transpilación de fórmulas

Una vez finalizada la fase de mapeo ya tenemos la información contenida en las hojas de cálculo y la meta información tal como las fórmulas y el orden topológico, lo que nos permitirá modelar las interacciones entre los datos y las fórmulas para dar como resultado nuevos datos.

En esta fase es donde podremos optimizar los procesos y eliminar los tiempos de espera que se presentan en la ejecución local, esto generando un plan de ejecución adecuado que permita paralelizar los procesos disponibles y disminuyendo la latencia al momento de acceder a los datos.

Excel fórmulas - Parser

Excel fórmulas es una solución pragmática incluida en las hojas de cálculo, que permite operaciones para el análisis o generación de datos, así como cualquier otro lenguaje tienen una sintaxis definida, un conjunto de token reservados, reglas semánticas etc.

Sin embargo, buscamos migrar toda dependencia de este lenguaje tanto desde el punto de vista sintáctico como de plataforma, es por eso que se propone realizar una transpilacion a un lenguaje que permita escalar estas operaciones tanto como sea necesario.

El primer componente necesario para realizar una transpiración es una estructura de datos que defina la estructura de la sintaxis de excel formula como lenguaje, para lograr esto podemos utilizar la gramática libre de contexto ya definida en el trabajo de **Efthimia Aivaloglou (Parsing Excel formulas: A grammar and its application on four large dataset)**

Figura 36: Token léxicos usados en la gramática

Token Name	Description	Contents	Priority
BOOL	Boolean literal	TRUE FALSE	0
CELL	Cell reference	\$? [A-Z]+ \$? [1-9][0-9]*	2
DDECALL	Dynamic Data Exchange link	' ([^'] ")+'	0
ERROR	Error literal	#NULL! #DIV/0! #VALUE! #NAME? #NUM! #N/A	0
ERROR-REF	Reference error literal	#REF!	0
EXCEL-FUNCTION	Excel built-in function name followed by '('	(Any entry from the function list ³) \()	5
FILE	External file reference	\ [0-9]+ \]	5
HORIZONTAL-RANGE	Range of rows	\$? [0-9]+ : \$? [0-9]+	0
NR	Named range	[A-Z_]\□ ₁][□ ₄]*	-2
NR-COMBINATION	Named range which starts with a string that could be another token	(TRUE FALSE [A-Z]+ [1-9][0-9]*[A-Z]_.?□ ₁]) [□ ₄]+	3
SR-COLUMN	Column definition in structured references	[w\.]+	-3
NUMBER	An integer, floating point or scientific notation number literal	[0-9]+ .? [0-9]* (e [0-9]+)?	0
REF-FUNCTION	Excel built-in reference-returning function '('	(INDEX OFFSET INDIRECT)\()	5
REF-FUNCTION-COND	Excel built-in conditional reference function '('	(IF CHOOSE)\()	5
RESERVED-NAME	An Excel reserved name	_xlnm\.[A-Z_]+	-1
SHEET	The name of a worksheet	□ ₂ + !	5
SHEET-QUOTED	A sheet reference in single quotes	(□ ₃ ")* ' !	5
MULTIPLE-SHEETS	A reference to multiple sheets	□ ₂ + : □ ₂ + !	1
MULTIPLE-SHEETS-QUOTED	A multiple sheets reference in single quotes	(□ ₃ ")+ : (□ ₃ ")+ ' !	1
STRING	String literal	" ([^"] "")* "	0
UDF	User Defined Function followed by '('	(_xll\.)? [A-Z_][A-Z0-9_\\.\□ ₁]* \()	4
VERTICAL-RANGE	Range of columns	\$? [A-Z]+ : \$? [A-Z]+	0
Placeholder character	Placeholder for	Specification	
□ ₁	Extended characters	Non-control Unicode characters x80 and up	
□ ₂	Sheet characters	Any character except ' * [] \:/ ? () ; { } # " = < > & + - * / ^ % , _	
□ ₃	Enclosed sheet characters	Any character except ' * [] \:/ ?	
□ ₄	Valid named range characters	A-Z0-9_\.?□ ₁	

Fuente:

https://www.researchgate.net/publication/319609929_Parsing_Excel_formulas_A_grammar_and_its_application_on_4_large_datasets_AIVALOGLOU_et_al

Figura 37: Gramática libre de contexto

```

⟨Start⟩ ::= ⟨Formula⟩
| '=' ⟨Formula⟩
| '{=' ⟨Formula⟩ '}'

⟨Formula⟩ ::= ⟨Constant⟩
| ⟨Reference⟩
| ⟨FunctionCall⟩
| '(' ⟨Formula⟩ ')'
| ⟨ConstantArray⟩
| RESERVED-NAME

⟨Constant⟩ ::= NUMBER | STRING | BOOL | ERROR

⟨FunctionCall⟩ ::= EXCEL-FUNCTION ⟨Arguments⟩ ')'
| ⟨UnOpPrefix⟩ ⟨Formula⟩
| ⟨Formula⟩ '%'
| ⟨Formula⟩ ⟨BinOp⟩ ⟨Formula⟩

⟨UnOpPrefix⟩ ::= '+' | '-'

⟨BinOp⟩ ::= '+' | '-' | '*' | '/' | '^' | '&' |
| '<' | '>' | '=' | '<=' | '>=' | '<>'

⟨Arguments⟩ ::= ⟨Argument⟩ { ',' ⟨Argument⟩ } | ε

⟨Argument⟩ ::= ⟨Formula⟩ | ε

⟨Reference⟩ ::= ⟨ReferenceItem⟩
| ⟨RefFunctionCall⟩
| '(' ⟨Reference⟩ ')'
| ⟨Prefix⟩ ⟨ReferenceItem⟩
| FILE '!' DDECALL

⟨RefFunctionCall⟩ ::= ⟨Union⟩
| ⟨RefFunctionName⟩ ⟨Arguments⟩ ')'
| ⟨Reference⟩ ':' ⟨Reference⟩
| ⟨Reference⟩ '₋' ⟨Reference⟩

⟨ReferenceItem⟩ ::= CELL
| ⟨NamedRange⟩
| VERTICAL-RANGE
| HORIZONTAL-RANGE
| UDF ⟨Arguments⟩ ')'
| ERROR-REF
| ⟨StructuredReference⟩

⟨Prefix⟩ ::= SHEET
| FILE SHEET
| FILE '!'
| MULTIPLE-SHEETS
| FILE MULTIPLE-SHEETS
| "" SHEET-QUOTED
| "" FILE SHEET-QUOTED
| "" MULTIPLE-SHEETS-QUOTED
| "" FILE MULTIPLE-SHEETS-QUOTED

⟨RefFunctionName⟩ ::= REF-FUNCTION
| REF-FUNCTION-COND

⟨NamedRange⟩ ::= NR | NR-COMBINATION

⟨Union⟩ ::= '(' ⟨Reference⟩ { ',' ⟨Reference⟩ } ')'

⟨StructuredReference⟩ ::= ⟨SRElement⟩
| '[' ⟨SRExpression⟩ ']'
| NR ⟨SRElement⟩
| NR '[' ']'
| NR '[' ⟨SRExpression⟩ ']'

⟨SRExpression⟩ ::= ⟨SRElement⟩
| ⟨SRElement⟩ (' : ' | ',') ⟨SRElement⟩
| ⟨SRElement⟩ ',' ⟨SRElement⟩ (' : ' | ',') ⟨SRElement⟩
| ⟨SRElement⟩ ',' ⟨SRElement⟩ ',' ⟨SRElement⟩ ':' ⟨SRElement⟩

⟨SRElement⟩ ::= '[' (NR | SR-COLUMN) ']'
| FILE

⟨ConstantArray⟩ ::= '{' ⟨ArrayColumns⟩ '}'

⟨ArrayColumns⟩ ::= ⟨ArrayRows⟩ { ';' ⟨ArrayRows⟩ }

⟨ArrayRows⟩ ::= ⟨ArrayConst⟩ { ',' ⟨ArrayConst⟩ }

⟨ArrayConst⟩ ::= ⟨Constant⟩
| ⟨UnOpPrefix⟩ NUMBER
| ERROR-REF

```

Fuente: https://www.researchgate.net/publication/319609929_Parsing_Excel_formulas_A_grammar_and_its_application_on_4_large_datasets_AIVALOGLOU_et_al

Figura 38: Diagrama de sintaxis de la fórmula con las reglas de producción expandida

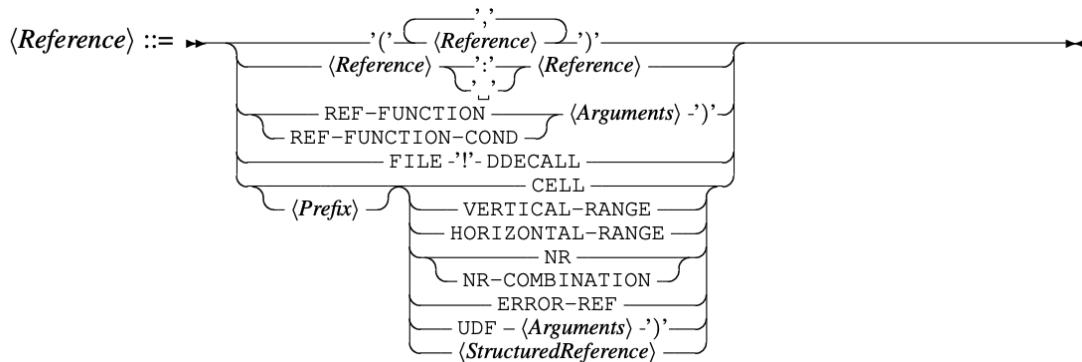
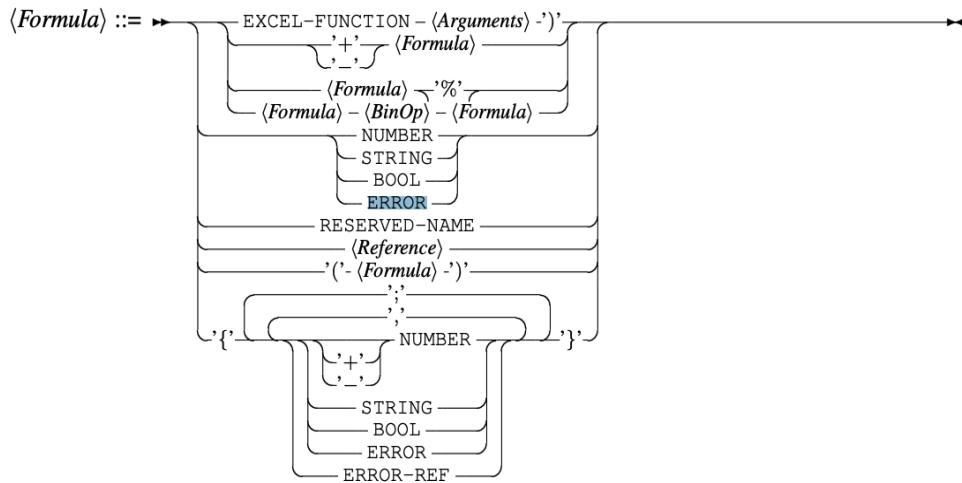


Figura 39: Diagrama de sintaxis de la referencia con las reglas de producción expandida:



Fuente:

https://www.researchgate.net/publication/319609929_Parsing_Excel_formulas_A_grammar_and_its_application_on_4_large_datasets_AIVALOGLOU_et_al

Una vez definida la gramática libre de contexto con las sentencias principales soportadas, podremos autogenerar un parser utilizando herramienta como ANTLR 4 que es un generador de “parsers” para leer, procesar, ejecutar y traducir texto estructurado o archivos binarios.

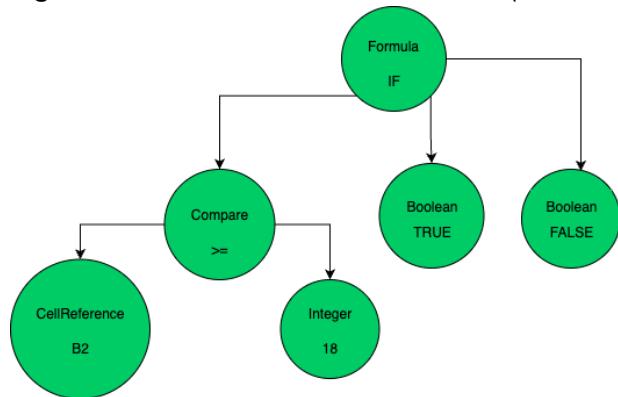
De esta forma podremos tomar una fórmula y recorrer el árbol de sintaxis abstracta que la define pudiendo obtener la meta información de esta tal como la información de entrada, la localización de las celdas y del resultado.

Figura 40: Condicional formula excel

=IF(B2>=18, True, False)

Fuente: Propia

Figura 41: Parser Tree de la fórmula = IF(B2>18, TRUE, FALSE)



Fuente: Propia

Finalmente junto con el parser generado por ANTLR4 podemos programáticamente acceder a el árbol de sintaxis abstracta y obtener toda la información que necesitamos para transpirar a un nuevo lenguaje.

4.3 Apache Spark - PySpark

([Apache Spark](#), Wikipedia) Apache Spark es un framework de computación (entorno de trabajo) en clúster open-source. Fue desarrollada originalmente en la Universidad de California, en el AMPLab de Berkeley. El código base del proyecto Spark fue donado más tarde a la Apache Software Foundation que se encarga del mantenimiento desde entonces. Spark proporciona una interfaz para la programación de clusters completos con Paralelismo de Datos implícito y tolerancia a fallos.

Apache Spark se puede considerar un sistema de computación en clúster de propósito general y orientado a la velocidad. Proporciona APIs en Java, Scala, Python y R. También proporciona un motor optimizado que soporta la ejecución de gráficos en general. También soporta un conjunto extenso y rico de herramientas de alto nivel entre las que se incluyen Spark SQL (para el procesamiento de datos estructurados basada en SQL), MLlib para implementar machine learning, GraphX para el procesamiento de gráficos y Spark Streaming.

Por lo tanto se propone transpilar las operaciones encontradas en las fórmulas al framework apache spark, utilizando como cliente el SDK de python, ya que nos permite escalar el procesamiento de los datos dependiendo de los recursos de infraestructura disponibles, tal como se menciona en el estudio realizado por George K. Thiruvathukal(*A Benchmarking Study to Evaluate Apache Spark on Large-Scale Supercomputers*) apache spark nos permitirá escalar el procesamiento de largos conjuntos de datos.

Python spark - Flujo de datos

Apache spark puede ser utilizado a través de python utilizando la librería pyspark, permitiéndonos definir planes de ejecución que luego serán enviados a los diferentes clúster distribuidos.

1 Definiendo un conjunto de datos

Se puede crear un PySpark DataFrame a través de **pyspark.sql.SparkSession.createDataFrame** normalmente pasando una lista de listas, tuplas, diccionarios y pyspark.sql.Rows, un pandas DataFrame y un RDD que consta de dicha lista. **pyspark.sql.SparkSession.createDataFrame** toma el argumento del esquema para especificar el esquema del DataFrame. Cuando se omite,

PySpark infiere el esquema correspondiente tomando una muestra de los datos.

Código 1: Definiendo datos estáticos a un RDD

```
rdd = spark.sparkContext.parallelize([
    (1, 2., 'string1', date(2000, 1, 1), datetime(2000, 1, 1, 12, 0)),
    (2, 3., 'string2', date(2000, 2, 1), datetime(2000, 1, 2, 12, 0)),
    (3, 4., 'string3', date(2000, 3, 1), datetime(2000, 1, 3, 12, 0))
])
df = spark.createDataFrame(rdd, schema=['a', 'b', 'c', 'd', 'e'])
df
```

Código 2: Datos definidos representados en una tabla

```
df.show()
df.printSchema()
+---+---+---+---+
| a | b | c | d |
+---+---+---+---+
| 1 | 2.0 | string1 | 2000-01-01 | 2000-01-01 12:00:00 |
| 2 | 3.0 | string2 | 2000-02-01 | 2000-01-02 12:00:00 |
| 3 | 4.0 | string3 | 2000-03-01 | 2000-01-03 12:00:00 |
+---+---+---+---+
```

2 Aplicando una transformación sobre el conjunto de datos

PySpark admite varias UDF y API para permitir a los usuarios ejecutar funciones nativas de Python. Consulte también las últimas UDF de Pandas y las API de funciones de Pandas. Por ejemplo, el siguiente ejemplo permite a los usuarios usar directamente las API en una serie pandas dentro de la función nativa de Python

Código 3: Aplicando operación sobre datos definiendo una función de python

```
import pandas as pd
from pyspark.sql.functions import pandas_udf

@pandas_udf('long')
def pandas_plus_one(series: pd.Series) -> pd.Series:
    # Simply plus one by using the pandas Series.
    return series + 1

df.select(pandas_plus_one(df.a)).show()
+---+
|pandas_plus_one(a)|
+---+
| 2 |
| 3 |
| 4 |
```

3 Resilient Distributed Dataset (RDD)- Operaciones

Los RDD admiten dos tipos de operaciones: transformaciones, que crean un nuevo conjunto de datos a partir de uno existente, y acciones, que devuelven un valor al programa controlador después de ejecutar un cálculo en el conjunto de datos. Por ejemplo, el mapa es una transformación que pasa cada elemento del conjunto de datos a través de una función y devuelve un nuevo RDD que representa los resultados. Por otro lado, reduce es una acción que agrega todos los elementos del RDD usando alguna función y devuelve el resultado final al programa controlador (aunque también existe un **reduceByKey** paralelo que devuelve un conjunto de datos distribuido).

Todas las transformaciones en Spark son perezosas, ya que no calculan sus resultados de inmediato. En cambio, solo recuerdan las transformaciones aplicadas a algún conjunto de datos base (por ejemplo, un archivo). Las transformaciones solo se calculan cuando una acción requiere que se devuelva un resultado al programa controlador. Este diseño permite que Spark funcione de manera más eficiente. Por ejemplo, podemos darnos cuenta de que un conjunto de datos creado a través del mapa se usará en una reducción y devolverá solo el resultado de la reducción al controlador, en lugar del conjunto de datos mapeado más grande.

De forma predeterminada, cada RDD transformado se puede volver a calcular cada vez que se ejecuta una acción en él. Sin embargo, también puede conservar un RDD en la memoria usando el método persistente (o caché), en cuyo caso Spark mantendrá los elementos en el clúster para un acceso mucho más rápido la próxima vez que lo consulte. También hay soporte para RDD persistentes en el disco o replicados en múltiples nodos.

4 Resilient Distributed Dataset (RDD) - Transformaciones Comunes

Una transformación es cada operación de Spark que devuelve un DataFrame, un Dataset o un RDD. Cuando creamos una cadena de transformaciones, agregamos componentes básicos al trabajo de Spark, pero no se procesa ningún dato. Eso es posible porque las transformaciones se ejecutan de forma perezosa. Spark calculará el valor cuando sea necesario.

Transformación	Descripción
----------------	-------------

map(func)	Devuelve un nuevo conjunto de datos distribuido formado al pasar cada elemento de la fuente a través de una función func.
filter(func)	Devuelve un nuevo conjunto de datos formado al seleccionar aquellos elementos de la fuente en los que func devuelve verdadero.
flatMap(func)	Aquí, cada elemento de entrada se puede asignar a cero o más elementos de salida, por lo que func debería devolver una secuencia en lugar de un solo elemento.
mapPartitions(func)	Es similar a map, pero se ejecuta por separado en cada partición (bloque) del RDD, por lo que func debe ser de tipo Iterator<T> => Iterator<U> cuando se ejecuta en un RDD de tipo T.
mapPartitionsWithIndex(func)	Es similar a mapPartitions que proporciona func con un valor entero que representa el índice de la partición, por lo que func debe ser de tipo (Int, Iterator<T>) => Iterator<U> cuando se ejecuta en un RDD de tipo T.
sample(withReplacement, fraction, seed)	Muestra la fracción fracción de los datos, con o sin reemplazo, utilizando una semilla generadora de números aleatorios dada.
union(otherDataset)	Devuelve un nuevo conjunto de datos que contiene la unión de los elementos del conjunto de datos de origen y el argumento.
intersection(otherDataset)	Devuelve un nuevo RDD que contiene la intersección de elementos en el conjunto de datos de origen y el argumento.
distinct([numPartitions])	Devuelve un nuevo conjunto de datos que contiene los distintos elementos del conjunto de datos de origen.
groupByKey([numPartitions])	Devuelve un conjunto de datos de pares (K, Iterable) cuando se llama a un conjunto de datos de pares (K, V).
reduceByKey(func, [numPartitions])	Cuando se invoca en un conjunto de datos de pares (K, V), devuelve un conjunto de datos de pares (K, V) donde los valores de cada clave se agregan utilizando la función de reducción dada, que debe ser del tipo (V, V) => v
aggregateByKey(zeroValue)(seqOp, combOp, [numPartitions])	Cuando se invoca en un conjunto de datos de pares (K, V), devuelve un conjunto de datos de pares (K, U) donde los valores de cada clave se agregan utilizando las funciones de combinación dadas y un valor "cero" neutral.
sortByKey([ascending], [numPartitions])	Devuelve un conjunto de datos de pares clave-valor ordenados por claves en orden ascendente o descendente, como se especifica en el argumento ascendente booleano.

join(otherDataset, [numPartitions])	Cuando se invoca en conjuntos de datos de tipo (K, V) y (K, W), devuelve un conjunto de datos de pares (K, (V, W)) con todos los pares de elementos para cada clave. Las combinaciones externas se admiten a través de leftOuterJoin, rightOuterJoin y fullOuterJoin.
cogroup(otherDataset, [numPartitions])	Cuando se invoca en conjuntos de datos de tipo (K, V) y (K, W), devuelve un conjunto de datos de (K, (Iterable, Iterable)) tuplas. Esta operación también se llama groupWith.
cartesian(otherDataset)	Cuando se invoca en conjuntos de datos de tipos T y U, devuelve un conjunto de datos de pares (T, U) (todos los pares de elementos).
pipe(command, [envVars])	Canalice cada partición del RDD a través de un comando de shell, p. un script Perl o bash.
coalesce(numPartitions)	Disminuye el número de particiones en el RDD a numPartitions.
repartition(numPartitions)	Reorganiza los datos en el RDD aleatoriamente para crear más o menos particiones y equilibrarlo entre ellas.
repartitionAndSortWithinPartitions(partitioner)	Reparticionar el RDD de acuerdo con el particionador dado y, dentro de cada partición resultante, ordena los registros por sus claves.

5 Resilient Distributed Dataset (RDD) - Acciones Comunes

Las acciones, por otro lado, no se ejecutan con pereza. Cuando ponemos una acción en el código y Spark llega a esa línea de código al ejecutar el trabajo, tendrá que realizar todas las transformaciones que conducen a esa acción para producir un valor.

Action	Description
reduce(func)	Agrega los elementos del conjunto de datos usando una función func (que toma dos argumentos y devuelve uno). La función debe ser conmutativa y asociativa para que pueda calcularse correctamente en paralelo.
collect()	Devuelve todos los elementos del conjunto de datos como una matriz en el programa controlador. Esto suele ser útil después de un filtro u otra operación que devuelve un subconjunto suficientemente pequeño de los datos.
count()	Devuelve el número de elementos en el conjunto de datos.
first()	Devuelve el primer elemento del conjunto de datos (similar a take(1)).

take(n)	Devuelve una matriz con los primeros n elementos del conjunto de datos.
takeSample(w ithReplaceme nt, num, [seed])	Devuelve una matriz con una muestra aleatoria de num elementos del conjunto de datos, con o sin reemplazo, opcionalmente especificando previamente una semilla generadora de números aleatorios.
takeOrdered(n , [ordering])	Devuelve los primeros n elementos del RDD usando el orden natural o un comparador personalizado.
saveAsTextFil e(path)	Se utiliza para escribir los elementos del conjunto de datos como un archivo de texto (o un conjunto de archivos de texto) en un directorio determinado en el sistema de archivos local, HDFS o cualquier otro sistema de archivos compatible con Hadoop. Spark llama a String en cada elemento para convertirlo en una línea de texto en el archivo.
saveAsSeque nceFile(path) (Java and Scala)	Se utiliza para escribir los elementos del conjunto de datos como Hadoop SequenceFile en una ruta determinada en el sistema de archivos local, HDFS o cualquier otro sistema de archivos compatible con Hadoop.
saveAsObject File(path) (Java and Scala)	Se usa para escribir los elementos del conjunto de datos en un formato simple usando la serialización de Java, que luego se puede cargar usando SparkContext.objectFile().
countByKey()	Solo está disponible en RDD de tipo (K, V). Por lo tanto, devuelve un hashmap de pares (K, Int) con el recuento de cada clave.
foreach(func)	Ejecuta una función en cada elemento del conjunto de datos para efectos secundarios, como actualizar un acumulador o interactuar con sistemas de almacenamiento externos.

4.4 Flujo de transpiración

Ya definidas las características que nos brinda apache spark, permitiéndonos escalar el procesamiento de los datos de forma distribuida, definiremos un caso de uso mostrando el flujo de transpiración de las fórmulas excel a la sintaxis de python usando pyspark para definir nuestras acciones y transformaciones.

4.4.1 Definición de la fórmula

Figura 42: Condicional formula excel

=IF(B2>=18, True, False)

Fuente: Propia

Esta fórmula es la declaración de un condicional, donde la operación binaria a ser evaluada es **B2>=18**, donde el segmento izquierdo hace referencia a la información de una celda y el segmento derecho a un valor numérico entero.

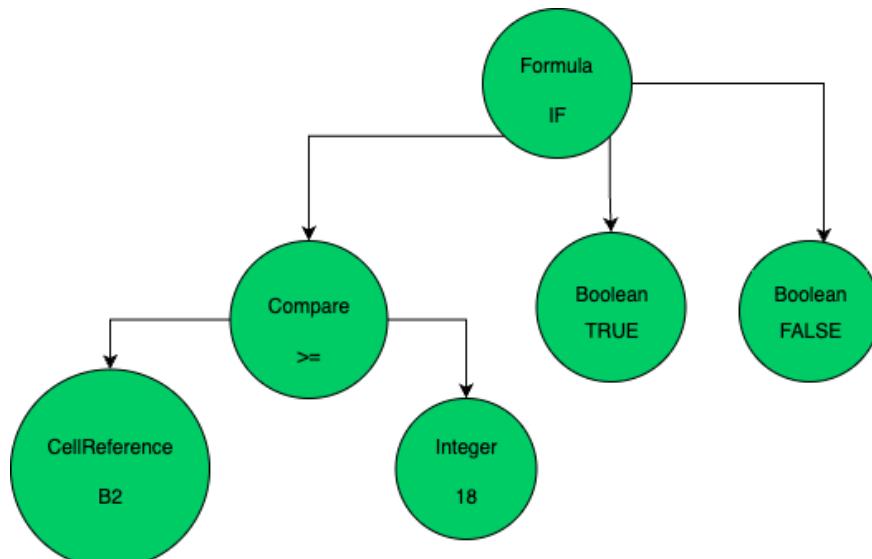
Si es la expresión ubicada en la condición da como resultado verdadera, se la expresión ubicada como segundo parámetro, de lo contrario se evaluará la que está ubicada en el tercer parámetro.

4.4.2 Parseo de la fórmula

Con ayuda del parser generado por ANTLR4 podremos validar y recorrer el árbol generado por la expresión y empezar a reconocer los datos, operaciones binarias, nombre reservados y empezar la transpiración a python usando pyspark.

A Continuación podemos ver un ejemplo de este caso.

Figura 43: Parser Tree de la fórmula =IF(B2>18, TRUE, FALSE)



Fuente: Propia

Ya teniendo los nodos identificados podemos entender que tipo de fórmula se está intentando analizar, en este caso es una condición y además entender si existen referencias a otras celdas, lo que significa que es necesario traer el valor ubicado en esas celdas o generar una dependencia ya que podría existir otra fórmula en ese lugar.

4.4.3 Mapeo de fuente de datos

Dado a que las fórmulas necesitan tener acceso a datos para realizar las transformaciones o acciones, debemos tener la referencia a la información que ya fue importada por el componente de importación.

La información se encontrará disponible en un base de datos definida por el componente de importación, por lo tanto, solo necesitaremos acceso a esta.

En este punto ya tendremos un RDD que contiene la información requerida para realizar la acción o transformación que queremos.

Figura 44: Query a base de datos para cargar información en un objeto RDD

```
_select_sql = "(select B from testdb.test_data)"  
df_select = spark.read.jdbc(url=url, table=_select_sql, properties...)
```

Fuente: Propia

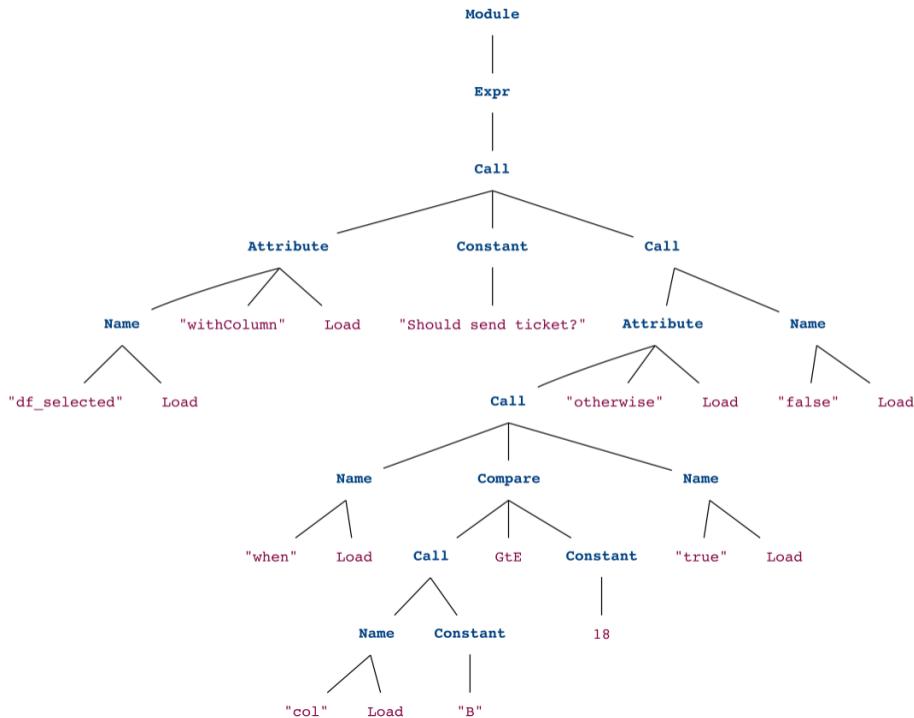
Con ayuda de este comando podremos traer solo la información necesaria para realizar los cálculos que la fórmula realizará, en este caso, la fórmula solo necesita la columna **B** que hace referencia a la columna con la cabecera **Age**.

Nota: Existen otras estrategias tales desde crear el data frame directamente, leer un archivo(JSON, CSV, PLAIN TEXT, PARQUET, HIVE, AVRO) etc. Sin embargo apache spark se encuentra optimizado para realizar operaciones con SQL junto con esto tener una base de datos externa nos permitirá escalar en el momento que la cantidad de datos aumente.

4.4.4 Generación de código python usando pyspark

Cuando tenemos mapeados el tipo de fórmula y los datos necesarios, podemos iniciar con la transpiración de datos, que tomará como entrada el árbol de la fórmula en excel y lo transformaremos en un árbol que represente la operación usando la sintaxis de python.

Figura 45: Árbol de sintaxis abstracta generado a partir de las transpiración usando el árbol de la Figura 38



Fuente: Propia

Generado el árbol de sintaxis abstracta que sigue las reglas sintácticas de python usando pyspark, podemos generar el texto plano de lo que sería el código final que podrá ser ejecutado para dar resultado a nuestra operación.

Figura 46: Código python generado a partir de la formula excel

```
df_selected.withColumn("Should send ticket?", when(col("B") >= 18, true)
    .otherwise(false))
```

Fuente: Propia

Definido esto podemos ejecutar las operaciones delegando gran parte de la lógica de orquestación y utilización de recursos a Spark.

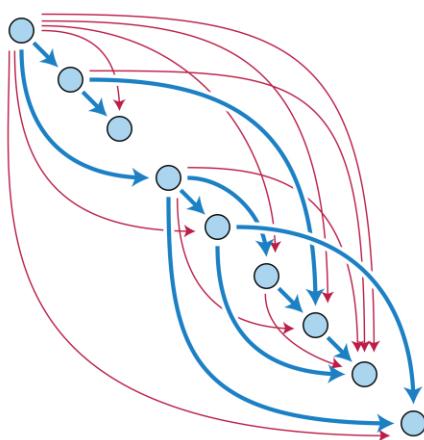
4.4.5 Plan de ejecución - Apache spark

Apache Spark tiene como núcleo una estructura de datos llamada RDD(Resilient Distributed Datasets) la cual es una colección distribuida inmutable de objetos. Cada conjunto de datos en RDD se divide en particiones lógicas, que se pueden calcular en diferentes nodos del clúster. Los RDD pueden contener cualquier tipo de objetos de Python, Java o Scala, incluidas las clases definidas por el usuario.

Esta estructura de datos permite orquestar la ejecución de los procesos a través de cluster distribuidos definiendo planes de ejecución que son representados en un grafo directo acíclico.

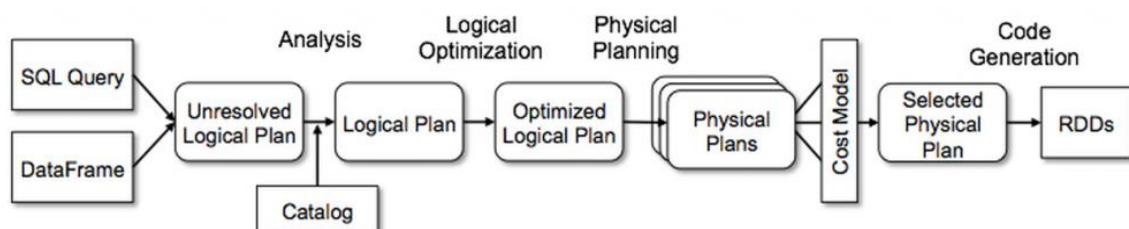
En matemáticas, particularmente en teoría de grafos e informática, un gráfico acíclico dirigido (DAG) es un gráfico dirigido sin ciclos dirigidos. Es decir, consta de vértices y aristas (también llamados arcos), con cada arista dirigida de un vértice a otro, de modo que seguir esas direcciones nunca formará un bucle cerrado. Un gráfico dirigido es un DAG si y sólo si se puede ordenar topológicamente, organizando los vértices como un orden lineal que sea consistente con todas las direcciones de los bordes. Los DAG tienen numerosas aplicaciones científicas y computacionales, que van desde la biología (evolución, árboles genealógicos, epidemiología) hasta las ciencias de la información (redes de citas) y la computación (programación).

Figura 47: Grafo acíclico dirigido



Fuente: Directed Acyclic Graph. (Unknown). En Wikipedia. Recuperado de https://en.wikipedia.org/wiki/Directed_acyclic_graph

Figura 48: Resilient Distributed Datasets plan



Fuente: Datalex. Sparks Logical and Physical Plans: When, Why, How, and Beyond. Medium .Recuperado de <https://medium.com/datalex/sparks-logical-and-physical-plans-when-why-how-and-beyond-8cd1947b605a>

Una vez ejecutado el plan definido por spark podemos verificar que la información se calculó correctamente.

```
df.show()
df.printSchema()
+---+---+-----+

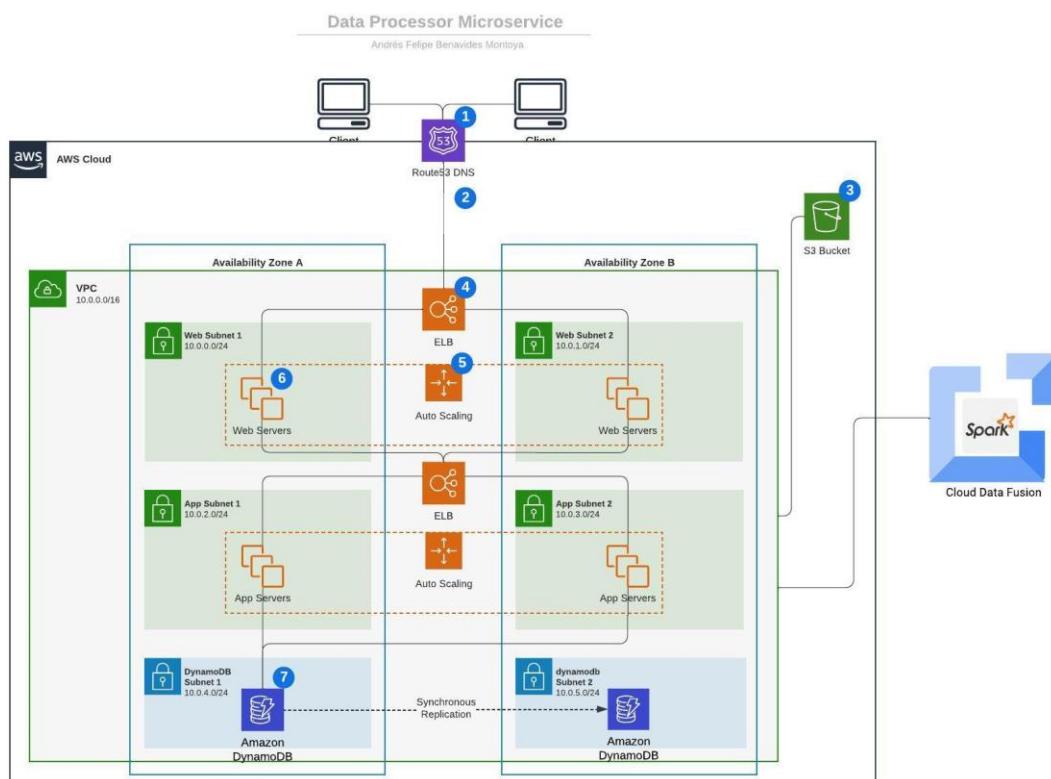
```

Age	Should send ticket?
19	true
10	false
12	false
30	true

4.5 Diseño técnico e integración de soluciones

En el procesamiento y análisis de datos moderno, aprovechar la infraestructura en la nube para manejar flujos de trabajo complejos de manera eficiente es crucial. La figura 48.1 describe una arquitectura de microservicios diseñada para procesar archivos, extraer y ejecutar fórmulas de Excel, y posteriormente transpilar estos flujos de trabajo para su ejecución en Apache Spark utilizando Cloud Data Fusion, almacenar los datos en dynamodb finalmente general un archivo en S3. Este enfoque asegura escalabilidad, fiabilidad y la capacidad de manejar tareas de procesamiento de datos a gran escala.

Figura 48.1: Diagrama de infraestructura de alto nivel para el microservicio encargado de procesar las fórmulas contenidas en el archivo importado



fuente: propia

El microservicio de procesamiento de datos basado en fórmulas está construido en la infraestructura de la nube de AWS, incorporando varios servicios para operaciones eficientes y escalables. El proceso se inicia cuando un cliente sube un archivo que contiene fórmulas de Excel, el cual es procesado por una serie de componentes interconectados.

Componentes

1. **Clients:** Usuarios que suben archivos de Excel para ser procesados.
2. **Route53 DNS:** Este servicio dirige el tráfico de los clientes a la aplicación. Actúa como el sistema de nombres de dominio que conecta la solicitud del usuario con los recursos de AWS.
3. **S3 Bucket:** Amazon S3 (Simple Storage Service) se utiliza para almacenar los archivos subidos de manera segura. Actúa como el repositorio para todos los archivos de datos entrantes.
4. **Elastic Load Balancer (ELB):** Distribuye el tráfico entrante entre múltiples objetivos, como instancias EC2, en múltiples zonas de disponibilidad para garantizar alta disponibilidad.
5. **Auto Scaling:** Ajusta automáticamente el número de instancias de EC2 para manejar la carga de trabajo de manera eficiente y proporcionar un rendimiento constante al menor costo.
6. **VPC (Virtual Private Cloud):** Aísla la configuración de red, proporcionando una mayor seguridad.
7. **Servidores Web y Aplicaciones:** Estos servidores, distribuidos en múltiples zonas de disponibilidad, manejan el procesamiento del frontend y backend de la aplicación.
8. **DynamoDB:** Almacén de datos para guardar los resultados obtenidos del ETL ejecutado en data fusion, permitirá consultar la información de forma rápida y eliminar los datos al cabo de un tiempo con el fin de optimizar almacenamiento.
9. **Cloud Data Fusion:** Un servicio gestionado para construir y gestionar pipelines de ETL/ELT. Se integra con Apache Spark para ejecutar flujos de trabajo de datos.

Flujo de Trabajo:

1. **Carga de Archivos por el Cliente:** Los usuarios suben archivos de Excel que contienen varias fórmulas a través de una interfaz web.
2. **Enrutamiento DNS:** La solicitud de carga de archivos se dirige a través del DNS de AWS Route53 al balanceador de carga.
3. **Almacenamiento de Archivos:** Los archivos subidos se almacenan en un bucket S3 para asegurar durabilidad y disponibilidad-

4. **Disparador de Procesamiento:** Una vez que el archivo se sube, una notificación (por ejemplo, a través de un evento S3) dispara los servidores web para que recojan el archivo para su procesamiento.
5. **Servidores Web:** Estos servidores son responsables de extraer datos y fórmulas de los archivos de Excel. El autoescalado asegura que el sistema pueda manejar cargas variables de manera eficiente.
6. **Servidores de Aplicaciones:** Estos servidores manejan las tareas computacionales, transpilan las fórmulas extraídas a un formato que se puede ejecutar en Apache Spark-
7. **Ejecución del Flujo de Trabajo de Datos:** Los flujos de trabajo transpildos se envían a Cloud Data Fusion. Utilizando Apache Spark, los flujos de trabajo se ejecutan para procesar los datos a gran escala.
8. **Almacenar datos:** Se guardaran los datos de las filas junto con los resultados de los ETL ejecutados

Ejecutar flujos de trabajo de datos en Apache Spark utilizando Cloud Data Fusion es una solución robusta y escalable. Esta arquitectura se beneficia de la fiabilidad de los servicios de AWS, las capacidades de autoescalado y el equilibrado de carga eficiente para manejar volúmenes grandes de datos sin problemas. Finalmente Cloud Data Fusion para gestionar pipelines de ETL asegura la flexibilidad y escalabilidad requeridas en las tareas modernas de procesamiento de datos.

4.6 Análisis de Resultados

Parte de escalar un sistema requiere entender qué cambios son los necesarios respecto a las necesidades que se afrontan, las hoja de cálculo son una excelente herramienta que nos permite correlacionar datos, realizar análisis estadístico y en algunos casos hacer que funcione como una base de datos, sin embargo mucha de esta información se encuentra tiene como finalidad enriquecer información que ya se encuentra recopilada en software enterprise al interior de cada empresa, sin embargo la tarea de organización, análisis y validación es en gran parte manual y a medida que la cantidad de datos es mayor, estos tiempos aumentan, llegando un punto en el que no es viable ya que se encuentran limitados a estrategias y recursos de laptops es por eso que con la solución.

- Se agrego el componente de la disponibilidad y seguimientos de los datos ya que esta información será almacenada en la nube
- Se mejoró los tiempos de procesamiento ya que se contó con estrategias que optimizan la ejecución de los procesos, utilizando computación distribuida en comparación con el sistema legacy que es serial.
- Se mantuvo la misma interfaz de organización y definición de fórmulas para que sea transparente al colaborador todo lo que ocurre.

La siguiente es una tabla comparativa entre el sistema actual y el propuesto, detallando los criterios comparables respecto al presente capítulo.

Criterio	Sistema actual	Sistema propuesto
Modificabilidad	Se requieren múltiples pasos, ya que un desarrollador con conocimiento en el lenguaje de la herramienta debe crear una solicitud de fusión (pull request, merge request) para agregar nuevas validaciones o actualizar las existentes.	No es necesario realizar cambios en la mayoría de los casos, dado que el sistema transpila automáticamente las fórmulas de Excel encontradas en el archivo.
Control de cambios	Las modificaciones relacionadas con la transformación de datos se versionan y se incorporan en la base de código.	No es necesario realizar cambios en la mayoría de los casos, ya que las hojas de cálculo contienen la información necesaria.
Disponibilidad	datos que este cambio se realiza a través de código, existe la posibilidad de introducir errores en el proceso de creación o actualización de los cambios.	No se debe realizar cambios, por lo tanto la posibilidad de introducir errores críticos es nula
Tiempo de implementación	Dado que hay numerosos cambios, este proceso lleva más tiempo e implica: implementación, creación de propuestas de cambios en el repositorio de código, espera de aprobación, validación de cambios en el ambiente de pruebas y despliegue.	No es necesario implementarlo, ya que la herramienta lo define automáticamente basándose en el contenido de la hoja de cálculo por sí misma.
Velocidad de ejecución	El desempeño está vinculado a la velocidad	Debido a que se transpila y se ejecuta en Apache

	del lenguaje en el que está implementado y a los recursos disponibles.	Spark en un entorno distribuido, es escalable por sí mismo, esto gracias a las optimizaciones incorporadas durante la transpilación y a las capacidades inherentes de la herramienta (Apache Spark).
--	--	--

CAPÍTULO 5: DISEÑO DE UN SISTEMA DE LOTES DISTRIBUIDOS PARA AUMENTAR EL VOLUMEN DE IMPORTACIÓN SIMULTÁNEAS.

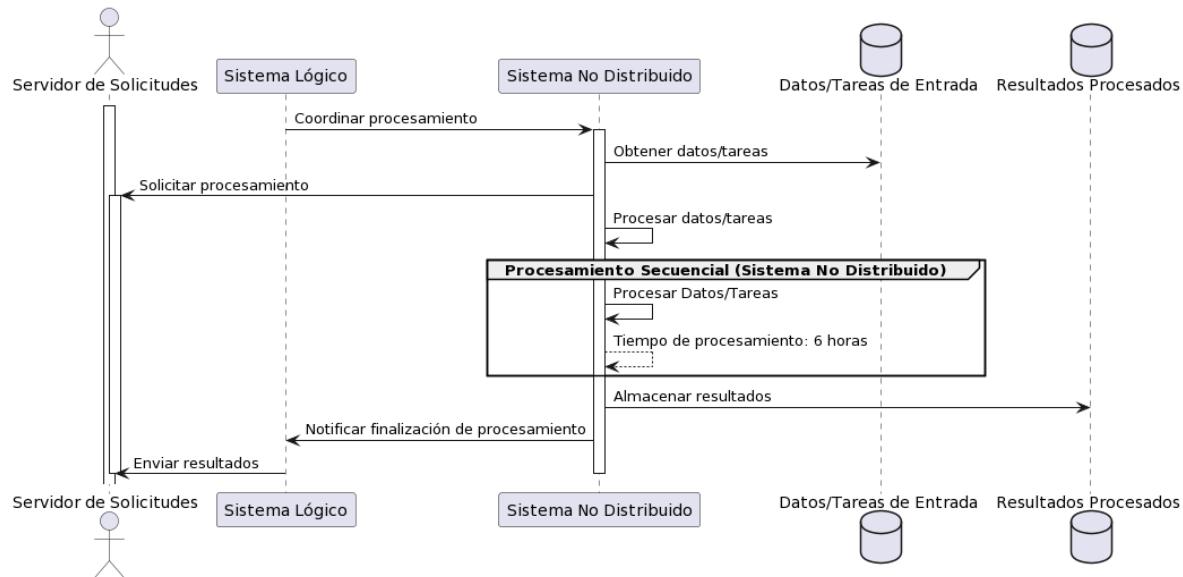
Un sistema de lotes distribuidos, es una estrategia de ejecución basada en nodos, que permite aumentar el volumen de procesos simultáneos en un ecosistema, para procesar una gran cantidad de datos o tareas en paralelo, dividiéndolas en lotes más pequeños y distribuyéndolas entre varias unidades de procesamiento (nodos). Esto permite que el sistema maneje un mayor volumen de datos o tareas en un período de tiempo más corto, en comparación con una sola unidad de procesamiento que trabaja en los mismos datos o tareas de forma secuencial.

En el contexto de la importación, se podría utilizar este para aumentar el volumen de importaciones simultáneas **mientras el proceso sea paraleizable**, dividiendo el proceso de importación en lotes más pequeños y distribuyéndolos entre varias unidades de procesamiento. Esto podría ser útil para escenarios donde hay un gran volumen de datos o tareas que deben importarse, y una sola unidad de procesamiento no podría manejarlos de manera eficiente. Mediante el uso de este tipo de estrategias, sería posible procesar las importaciones de manera más rápida y eficiente.

En los siguientes diagramas se demuestra como con 4 nodos es posible dividir y procesar un conjunto de datos por lotes disminuyendo los tiempos de procesamiento a una cuarta parte, es decir el sistema con una estrategia de procesamiento serial le tomó procesar la información 6 horas, mientras que el sistema distribuido con 4 nodos ejecutó la tarea en 1.5 horas dado a que dividió la carga de procesamiento entre los nodos.

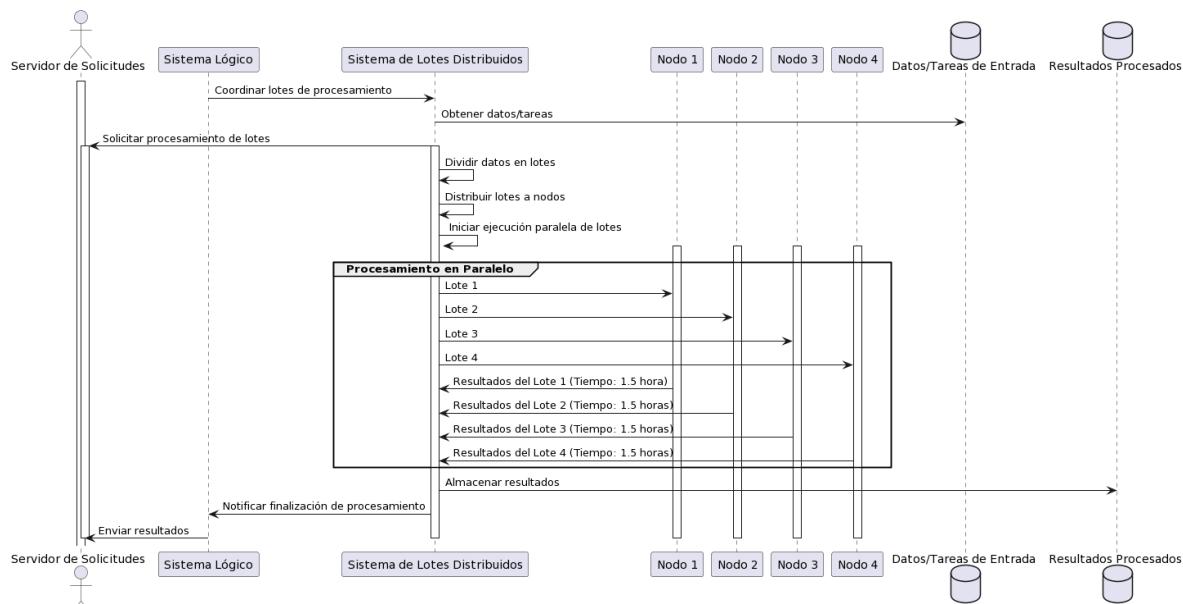
El siguiente diagrama de secuencia ilustra el proceso de importación del sistema actual

Figura 49: Sistema no distribuido o serial



El siguiente diagrama de secuencia ilustra el proceso de importación del sistema propuesto.

Figura 50: Sistema distribuido



5.1 Proceso de implementación de importaciones distribuidas

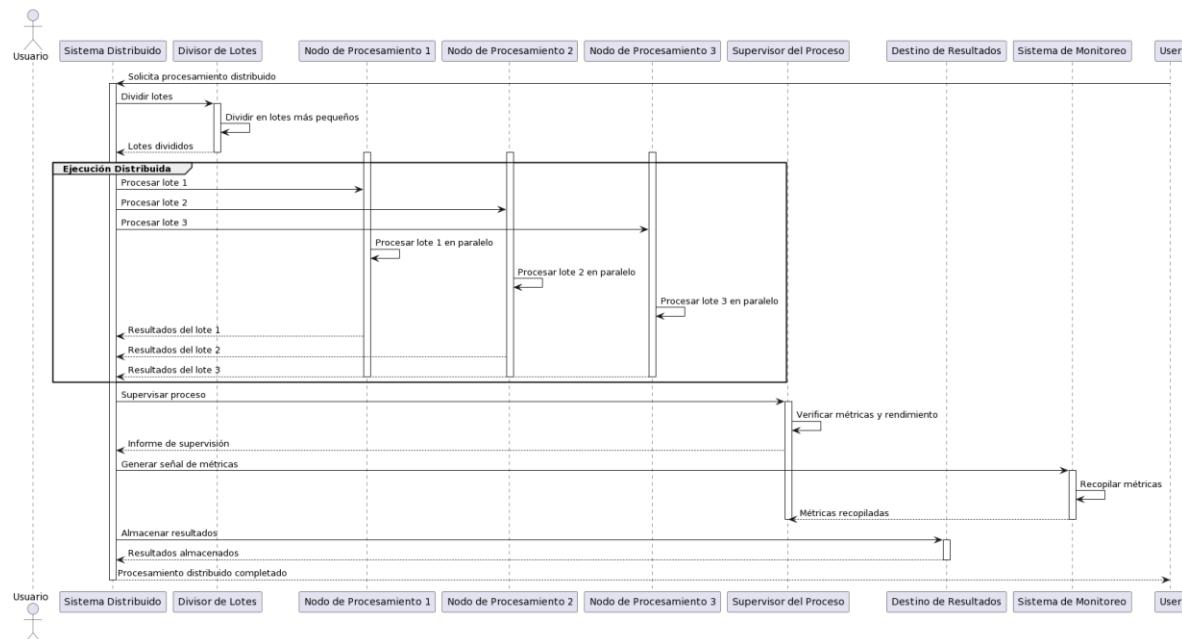
Se definieron varios pasos a seguir para implementar el proceso de importación de datos distribuidos:

- **Identificación de la fuente de datos:** El primer paso es determinar de dónde provienen los datos que desea importar. Esto podría ser una base de datos, un archivo o un servicio externo.
- **Definición de plan de proceso de importación:** A continuación, debe definir el proceso para importar los datos. Esto podría implicar definir el formato de los datos, la transformación o el procesamiento que debe aplicarse a los datos y el destino donde se almacenarán los datos.
- **Configuración de una infraestructura informática distribuida:** Se debe tener una infraestructura informática distribuida, compuesta por un grupo de computadoras físicas o en la nube que representarán las unidades de procesamiento (nodos).
- **División del proceso de importación en lotes más pequeños:** Se deberá dividir el proceso de importación en lotes de datos más pequeños, esto permitirá distribuir el proceso entre las unidades de procesamiento de la infraestructura.
- **Distribución del proceso de importación entre las unidades de procesamiento:** Se deben distribuir los lotes entre las unidades de procesamiento de la infraestructura distribuida, lo cual puede ser realizado usando una herramienta o marco diseñado para el procesamiento distribuido, como Apache Spark.

- **Coordinación del proceso de importación:** Para garantizar que el proceso de importación se desarrolle sin problemas y de manera eficiente, se deberá coordinar el procesamiento de los lotes de datos y la importación de los datos en el destino para mantener la integridad y consistencia de estos, esto podría implicar el uso de una herramienta o marco para administrar el proceso, o escribir código personalizado para coordinar las diferentes partes del proceso.
- **Supervisión del proceso de importación:** Por último, es importante supervisar el proceso de importación para asegurarse de que se ejecuta sin problemas e identificar cualquier problema que pueda surgir. Esto podría implicar el uso de herramientas de registro o monitoreo para rastrear el progreso del proceso de importación y alertar si se detecta algún problema.

El siguiente diagrama de secuencia ilustra a detalle el proceso de importación en el sistema propuestos

Figura 51: Diagrama de secuencia detallado - Sistema distribuido



5.2 Tecnologías y estrategias para implementar el proceso de importación

Se usan diferentes tecnologías y estrategias de software para implementar un proceso de importación de datos distribuidos, explicados más adelante en esta sección.

5.2.1 Marcos de procesamiento distribuido

Estas son herramientas que están diseñadas para dividir una gran tarea de procesamiento en subtareas más pequeñas y distribuirlas entre un grupo de computadoras u otras unidades de procesamiento. Los ejemplos incluyen **Apache Hadoop**, **Apache Spark** y **Google Cloud Dataflow**. Estos marcos se pueden usar para dividir el proceso de importación en lotes más pequeños y distribuirlos entre las unidades de procesamiento en la infraestructura informática distribuida.

Por consiguiente, se decidió usar apache spark por el soporte de la comunidad y fuerte conexión con el lenguaje de programación python, lo que permitirá una fácil integración con el servicio web diseñado en Django. (Fuente: *PySpark Apache spark support* https://spark.apache.org/docs/latest/api/python/getting_started/index.html)

5.2.2 Herramientas de integración de datos

Estas son herramientas que están diseñadas para extraer datos de diferentes fuentes y cargarlos en un sistema de destino.

Los ejemplos incluyen herramientas ETL (extraer, transformar, cargar) como **Talend**, **Pentaho**, y plataformas de integración de datos como **Google Cloud Data Fusion** y **Azure Data Factory**, estas herramientas se pueden utilizar para definir el proceso de importación y automatizar la extracción y transformación de los datos.

Para nuestro caso de uso se propone utilizar Google Cloud Data Fusion por las siguientes ventajas

- Interfaz visual y sencilla que facilita el despliegue sin código de flujos de procesamiento de datos de extracción, transformación y carga (ETL) o de extracción, carga y transformación (ELT)
- Extensa biblioteca gratuita con más de 150 transformaciones y conectores preconfigurados
- Integración nativa con los excepcionales servicios de Google Cloud
- Linaje integral de datos para analizar causas y efectos
- Diseño con un núcleo de código abierto (CDAP) para facilitar la portabilidad de flujos de procesamiento

Además, se planea la creación de diferentes conectores que nos permitan obtener la información de diferentes fuentes de datos, sin embargo, la principal característica deseada es poder extraer la información de una spreadsheet, por lo tanto, se decide incorporar al servicio web un módulo de lectura y mapeo de estos datos usando la librería **openpyxl** diseñada para realizar la lectura de archivos xls, csv, xlsx que son las principales extensiones de archivos usadas en spreadsheets.

Google Cloud Data Fusion vs Talend: ¿Cuáles son las diferencias?

Google Cloud Data Fusion y Talend son plataformas populares para la integración y transformación de datos. Sin embargo, existen varias diferencias clave entre las dos:

Facilidad de Uso: Google Cloud Data Fusion proporciona una interfaz visual sin código que permite a los usuarios configurar fácilmente tuberías de integración de datos. Ofrece una interfaz simple de arrastrar y soltar para construir flujos de datos y transformaciones, lo que lo hace adecuado para usuarios con conocimientos técnicos limitados. Por otro lado, Talend requiere algunos conocimientos de programación y ofrece un entorno de desarrollo más complejo, requiriendo que los usuarios escriban código para las tareas de integración de datos.

Escalabilidad y Rendimiento: Google Cloud Data Fusion utiliza la infraestructura y los recursos de Google Cloud, permitiendo una alta escalabilidad y rendimiento. Puede manejar grandes volúmenes de datos y se puede escalar fácilmente hacia arriba o hacia abajo según la demanda. Talend también proporciona opciones de escalabilidad, pero puede no ofrecer el mismo nivel de rendimiento que Google Cloud Data Fusion debido a sus limitaciones de infraestructura.

Integración con Servicios en la Nube: Google Cloud Data Fusion tiene una integración perfecta con otros servicios de Google Cloud como BigQuery, Cloud Storage y Dataflow. Esto permite a los usuarios procesar y analizar datos fácilmente utilizando varias herramientas y servicios de Google Cloud. En contraste, Talend puede requerir una configuración adicional y ajustes para integrarse con servicios en la nube, lo que lo hace potencialmente más laborioso y complejo.

Modelo de Precios: Google Cloud Data Fusion sigue un modelo de precios de pago por uso, donde los usuarios solo pagan por los recursos que consumen. Esto puede ser rentable para organizaciones con necesidades variables de procesamiento e integración de datos. Talend, por otro lado, típicamente sigue un modelo de precios basado en suscripción que puede no ser tan flexible según el uso, lo que potencialmente conlleva mayores costos para ciertas organizaciones.

Conectores Pre-construidos: Google Cloud Data Fusion ofrece una amplia gama de conectores pre-construidos para varias fuentes de datos y sistemas, simplificando el proceso de integración. Estos conectores incluyen bases de datos populares, como MySQL, Oracle y SQL Server, así como servicios en la nube como Salesforce y Google Analytics. Talend también proporciona un conjunto rico de conectores, pero la disponibilidad de los conectores puede variar dependiendo de la versión y edición específica en uso.

Soporte y Comunidad: Google Cloud Data Fusion se beneficia de los amplios recursos de soporte de Google y de una gran comunidad de usuarios. Los usuarios pueden aprovechar la documentación, los foros y los canales de soporte de Google para obtener asistencia. Talend también proporciona servicios de soporte y tiene una

comunidad activa, pero el nivel de soporte y recursos puede variar dependiendo de la edición y licencia específica.

En resumen, Google Cloud Data Fusion ofrece una plataforma de integración de datos fácil de usar, escalable y rentable con integración perfecta a los servicios de Google Cloud. Talend, por otro lado, proporciona una plataforma más flexible y personalizable con una amplia gama de conectores y opciones de soporte. La elección entre los dos depende de los requisitos específicos, la experiencia técnica y las preferencias de la organización.

Google Cloud Data Fusion vs Azure Data Factory : ¿Cuáles son las diferencias?

Azure Data Factory y Google Cloud Data Fusion son dos servicios populares de integración de datos basados en la nube que ofrecen capacidades para orquestar y gestionar flujos de trabajo de datos. Veamos las diferencias clave entre ellos.

Escalabilidad: Azure Data Factory aprovecha la infraestructura en la nube de Azure, permitiendo a los usuarios escalar según sus necesidades. Ofrece opciones flexibles para el movimiento y la transformación de datos, permitiendo una integración sin inconvenientes con varias fuentes y destinos de datos. Por otro lado, Google Cloud Data Fusion ofrece escalabilidad incorporada y puede manejar grandes volúmenes de datos fácilmente, gracias a la infraestructura masiva de Google. Ofrece una interfaz visual de no-código para procesos ETL (Extracción, Transformación y Carga), haciéndola accesible para usuarios no técnicos.

Integración con Servicios Nativos: Azure Data Factory está estrechamente integrado con otros servicios de Azure como Azure Databricks, Azure Synapse Analytics y Azure Machine Learning. Esta integración permite a los usuarios construir tuberías de datos de extremo a extremo que abarcan varios servicios de Azure. Google Cloud Data Fusion, por otro lado, está diseñado para integrarse sin problemas con los servicios de la Plataforma de Google Cloud como BigQuery, Pub/Sub y Dataproc. Esta integración permite a los usuarios aprovechar al máximo el ecosistema de Google Cloud para el procesamiento y análisis de datos.

Facilidad de Uso: Azure Data Factory proporciona una interfaz visual para diseñar tuberías de datos utilizando un enfoque de arrastrar y soltar. También ofrece capacidades avanzadas de transformación de datos a través de su característica de flujos de datos de mapeo. Google Cloud Data Fusion proporciona un entorno sin código para diseñar y desplegar tuberías de datos. Ofrece una amplia gama de conectores y transformaciones que se pueden configurar fácilmente a través de una interfaz visual, haciendo que sea fácil para los usuarios construir y gestionar flujos de trabajo de datos complejos.

Modelo de Precios: Azure Data Factory sigue un modelo de precios basado en el consumo, donde los usuarios pagan por los recursos que consumen, como el movimiento de datos, la transformación de datos y la ejecución de tuberías. Los precios se basan en factores como el número de ejecuciones de tuberías, el volumen de movimiento de datos y la complejidad de la transformación de datos. Google Cloud Data Fusion, por otro lado, sigue un modelo de precios fijo basado en el tamaño y la complejidad de las tuberías de datos. Los usuarios son facturados en función del número de nodos de tuberías, el volumen de movimiento de datos y el uso de características adicionales como la transformación de datos.

Monitorización y Gestión: Azure Data Factory proporciona un conjunto rico de características de monitorización y gestión, incluyendo la monitorización de tuberías, alertas y reinicios automáticos. Se integra con Azure Monitor y Azure Log Analytics para la recopilación y el análisis de métricas y registros de tuberías. Google Cloud Data Fusion ofrece capacidades de monitorización incorporadas que proporcionan información en tiempo real sobre las tuberías de datos, incluyendo métricas sobre la ingestión, transformación y salida de datos. También se integra con los servicios de monitorización y registro de Google Cloud para la gestión y monitorización centralizada.

Ecosistema e Integraciones de Terceros: Azure Data Factory se beneficia de ser parte del ecosistema más amplio de Azure, que incluye una amplia gama de servicios y herramientas para análisis de datos, IA y aprendizaje automático. Se integra sin problemas con servicios como Azure Data Lake Storage, Azure SQL Database y Power BI. Google Cloud Data Fusion tiene un ecosistema en crecimiento de

conectores de terceros e integraciones, permitiendo a los usuarios conectarse a varias fuentes y destinos de datos. Además, se integra con servicios populares de Google Cloud como AutoML, Cloud Pub/Sub y Bigtable.

En resumen, Azure Data Factory proporciona una integración sólida con el ecosistema de Azure y ofrece capacidades avanzadas de transformación de datos, mientras que Google Cloud Data Fusion destaca en escalabilidad y facilidad de uso, con un enfoque en la integración nativa con los servicios de Google Cloud Platform.

Google Cloud Data Fusion vs Pentaho Data Integration: ¿Cuáles son las diferencias?

Google Cloud Data Fusion y Pentaho Data Integration son plataformas populares para la integración y transformación de datos. A continuación, exploramos las diferencias clave entre ellas:

Facilidad de Uso: Google Cloud Data Fusion ofrece una interfaz visual sin código, permitiendo a los usuarios configurar fácilmente las tuberías de integración de datos. La interfaz de arrastrar y soltar facilita la construcción de flujos de datos y transformaciones, haciéndolo accesible para usuarios con conocimientos técnicos limitados, por otro lado Pentaho Data Integration ofrece tanto una interfaz gráfica de usuario como la opción de script de código aunque tiene una interfaz visual similar, puede requerir más experiencia técnica para aprovechar todas sus capacidades avanzadas.

Escalabilidad y Rendimiento: Google Cloud Data Fusion aprovecha la infraestructura de Google Cloud, permitiendo una alta escalabilidad y rendimiento, también maneja grandes volúmenes de datos y se puede escalar fácilmente según la demanda, por otro lado Pentaho Data Integration ofrece opciones robustas de escalabilidad, incluyendo la capacidad de ejecutarse en clústeres locales y entornos en la nube, sin embargo, el rendimiento puede depender de la infraestructura subyacente y la configuración específica.

Integración con Servicios en la Nube

Google Cloud Data Fusion se integra perfectamente con otros servicios de Google Cloud como BigQuery, Cloud Storage y Dataflow. Esto permite a los usuarios procesar y analizar datos utilizando diversas herramientas y servicios de Google Cloud. Por otro lado Pentaho Data Integration puede integrarse con varios servicios en la nube y herramientas de terceros, incluyendo AWS, Azure y Google Cloud, pero puede requerir más configuración y ajustes además. Ofrece conectores nativos para diferentes bases de datos y servicios, pero la integración puede ser más compleja en comparación con Google Cloud Data Fusion.

Modelo de Precios: Google Cloud Data Fusion sigue un modelo de precios basado en el consumo, donde los usuarios pagan por los recursos que consumen. Esto puede ser rentable para organizaciones con necesidades variables de procesamiento e integración de datos. Por otro lado, Pentaho Data Integration generalmente sigue un modelo de precios basado en suscripción o licencia perpetua. El costo puede ser más elevado para organizaciones que no necesitan capacidades avanzadas de integración.

Conectores Pre-construidos: Google Cloud Data Fusion ofrece una amplia gama de conectores pre-construidos para varias fuentes de datos y sistemas. Incluye bases de datos populares como MySQL, Oracle y SQL Server, así como servicios en la nube como Salesforce y Google Analytics. Pentaho Data Integration incluye un extenso conjunto de conectores para bases de datos, aplicaciones y servicios. La disponibilidad y la variedad de conectores pueden ser mayores, pero su configuración puede requerir más tiempo y conocimiento técnico.

Sopporte y Comunidad

Google Cloud Data Fusion cuenta con los recursos de soporte extensivos de Google y una gran comunidad de usuarios. Los usuarios pueden aprovechar la documentación de Google, foros y canales de soporte para obtener asistencia. Pentaho Data Integration proporciona servicios de soporte a través de Hitachi Vantara y tiene una comunidad activa. El nivel de soporte y recursos puede ser sólido, pero variará según las necesidades específicas de la edición y licencia.

En resumen Google Cloud Data Fusion ofrece una plataforma de integración de datos fácil de usar, escalable y rentable con integración perfecta a los servicios de Google Cloud. Pentaho Data Integration, por otro lado, proporciona una plataforma más flexible con más opciones de personalización y una amplia variedad de conectores y capacidades de

integración. La elección entre las dos dependerá de los requisitos específicos, la experiencia técnica y las preferencias de la organización.

5.2.3 Lenguajes de manipulación de datos

Estos son lenguajes de programación que se pueden usar para manipular y transformar datos. Los ejemplos incluyen **SQL**, **Python** y **R**, estos lenguajes se pueden usar para escribir código personalizado para transformar los datos a medida que se importan, o para coordinar el proceso de importación y administrar el flujo de datos entre el origen y el destino.

Se propone usar Python como lenguaje de programación principal dado debido a la extensa integrabilidad y soporte con la herramienta para realizar el procesamiento de datos Apache Spark y el framework web usado, Django.

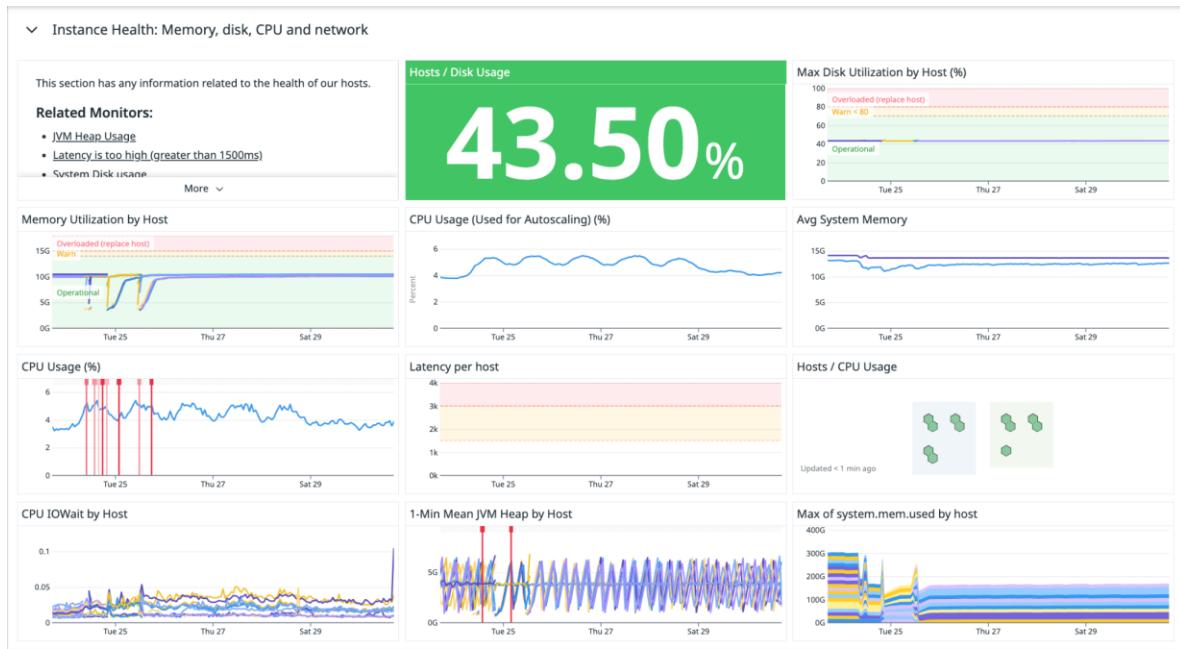
5.2.4 Plataformas de almacenamiento y procesamiento de datos

Estas son plataformas que proporcionan una forma de almacenar y procesar datos a escala. Los ejemplos incluyen almacenes de datos como **Amazon Redshift**, **Snowflake** o **Amazon S3** y **Azure Data Lake**. Estas plataformas se pueden utilizar como destino de los datos importados y pueden proporcionar la infraestructura necesaria para almacenar y procesar los datos a escala.

Se propone utilizar Amazon S3 dado el soporte y escalabilidad cuando la cantidad de datos crezca, lo que permite un cobro bajo demanda y no fijo.

5.2.5 Herramientas de monitoreo y registro

Figura 52: Monitoreo de métricas de sistema a través de datadog



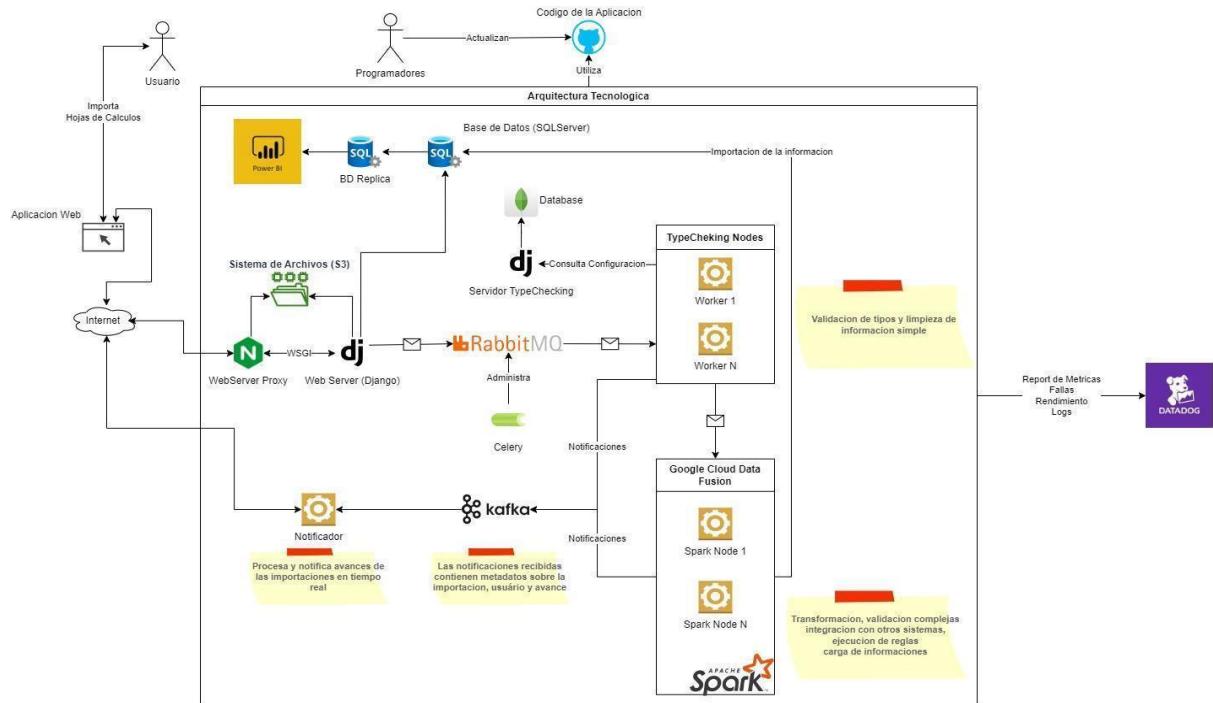
Estas son herramientas que se pueden usar para monitorear el proceso de importación e identificar cualquier problema que pueda surgir. Los ejemplos incluyen herramientas de registro como **Splunk** y **ELK Stack**, y herramientas de monitoreo como **Datadog** y **New Relic**. Estas herramientas se pueden usar para rastrear el progreso del proceso de importación y alertar si se detecta algún problema.

Se propone utilizar **Datadog** como herramienta de monitoreo por la extensibilidad e integrabilidad con lenguajes de programación como python además del alto uso en la industria para monitoreo de tareas.

5.3 Diseño técnico e integración de soluciones

El siguiente diagrama ilustra a alto nivel los componentes y tecnologías del sistema propuesto.

Figura 53: Arquitectura a alto nivel con sistema de typechecking y procesamiento distribuido con spark integrado



Fuente: Propia

Para integrar el sistema de validación de datos en el proceso de importación distribuido, se diseñó el proceso de validación para poder ejecutarse en paralelo con el proceso de importación (Figura 53.). Esto implica dividir los datos que se van a importar en lotes más pequeños y luego ejecutar el proceso de validación en cada lote al mismo tiempo que el proceso de importación.

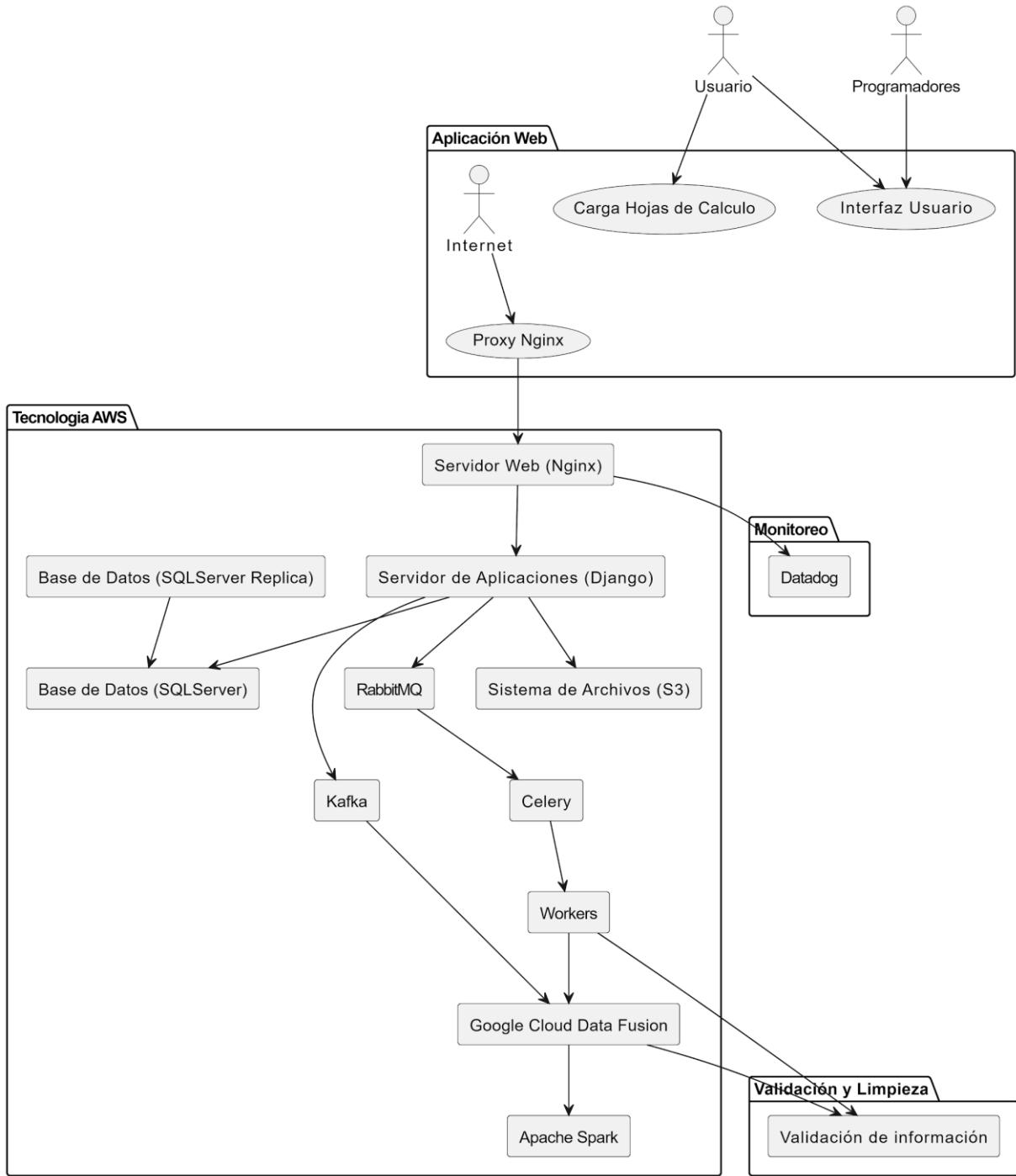
La forma propuesta de hacerlo es utilizar un sistema de procesamiento distribuido, como RabbitMQ y Celery, este proporciona un marco para ejecutar aplicaciones distribuidas en un clúster de computadoras e incluye herramientas para dividir datos en lotes más pequeños y distribuirlos entre las computadoras del clúster.

Para integrar el proceso de validación de datos en el proceso de importación mediante Celery, primero debe definir las reglas de validación que desea aplicar a los datos importados. Estas reglas podrían implementarse utilizando un lenguaje de programación como Java o Python, y podrían incluir comprobaciones de tipos de datos, formatos y otros criterios que los datos deben cumplir para que se consideren válidos.

Una vez que se han definido las reglas de validación, se usa Celery para dividir los datos que se van a importar en lotes más pequeños y distribuirlos entre las computadoras del clúster. El proceso de validación se puede ejecutar simultáneamente con el proceso de importación, aplicando las reglas de validación a cada lote de datos a medida que se importa.

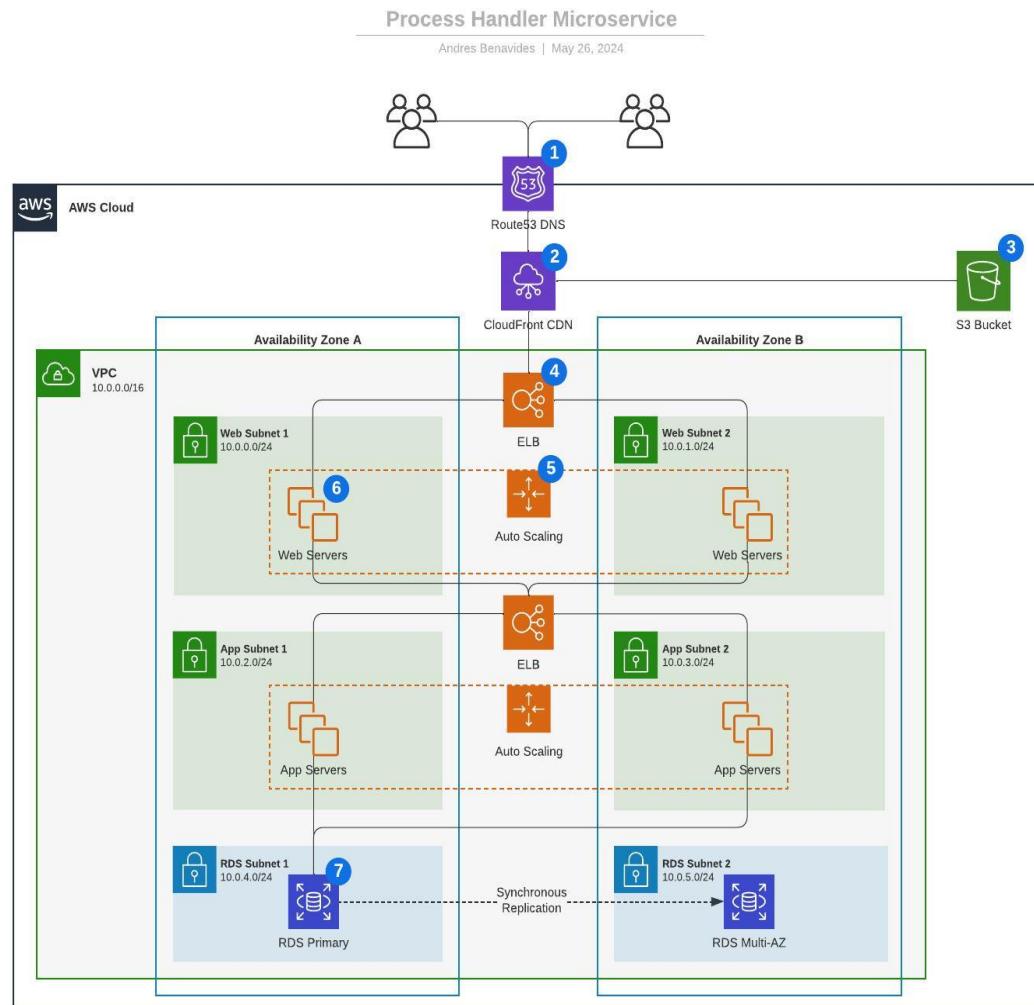
Si el proceso de validación encuentra algún error o problema con los datos importados, se envía una notificación o activa una alerta para notificar a las partes correspondientes, de modo que se pueda abordar el problema y corregir el proceso de importación. Esto puede ayudar a garantizar la precisión y la calidad de los datos importados y mejorar la eficiencia del proceso de importación en general.

Figura 54: Diagrama de componentes y relación con los actores



Fuente: Propia

Figura 55: Diagrama de infraestructura de alto nivel para el microservicio encargado de manejar las peticiones de importación de datos



Fuente: Propia

El diagrama describe la arquitectura de un Microservicio de Gestor de Procesos desplegado en la nube de AWS, detallando sus diversos componentes y cómo interactúan dentro de una configuración de alta disponibilidad. Aquí tienes una descripción paso a paso:

1. Route53 DNS:

- **Función:** Gestiona y dirige el tráfico hacia el dominio del servicio.

- **Usuarios:** Las solicitudes DNS de los usuarios son gestionadas por Route 53.

2. CloudFront CDN:

- Función: CloudFront (Red de Entrega de Contenidos) distribuye y almacena en caché contenido para los usuarios finales.
- Propósito: Baja latencia y altas velocidades de transferencia de datos.

3. Bucket S3:

- **Función:** Almacena activos estáticos como archivos o copias de seguridad.
- **Acceso:** Accesible por diversos componentes dentro de la arquitectura para operaciones de lectura/escritura.

4. Balanceador de Carga Elástico (ELB):

- Función: Distribuye el tráfico web entrante entre múltiples servidores web.
- Ubicación: Desplegado en las Zonas de Disponibilidad A y B.

5. Auto Escalado:

- Función: Ajusta automáticamente el número de servidores web y de aplicaciones para coincidir con la carga y mantener el rendimiento.
- Estrategia: Garantiza escalabilidad y disponibilidad al lanzar o terminar instancias basadas en políticas definidas.

6. Servidores Web y de Aplicaciones:

- Subredes Web (1 y 2):
 - Hospedadas en las Zonas de Disponibilidad A y B.
 - Sirven como servidores web frontend, manejando solicitudes HTTP entrantes.
- Subredes de Aplicaciones (1 y 2):
 - Hospedadas en las Zonas de Disponibilidad A y B.
 - Manejan la lógica de la aplicación y el procesamiento en segundo plano.
- **Propósito:** Distribuidos a través de múltiples subredes y zonas de disponibilidad para mayor redundancia.

7. RDS (Servicio de Base de Datos Relacional):

- RDS Primaria (Zona de Disponibilidad A):

- Contiene la instancia primaria de la base de datos.
- **RDS Multi-AZ (Zona de Disponibilidad B):**
 - Instancia secundaria de la base de datos para alta disponibilidad.
 - La replicación síncrona asegura la consistencia de los datos entre la instancia primaria y la secundaria.

VPC (Nube Privada Virtual):

- Bloque CIDR^{**}: 10.0.0.0/16.
- Función: Aísla recursos, proporcionando seguridad de red y control sobre el enrutamiento.

Resumen:

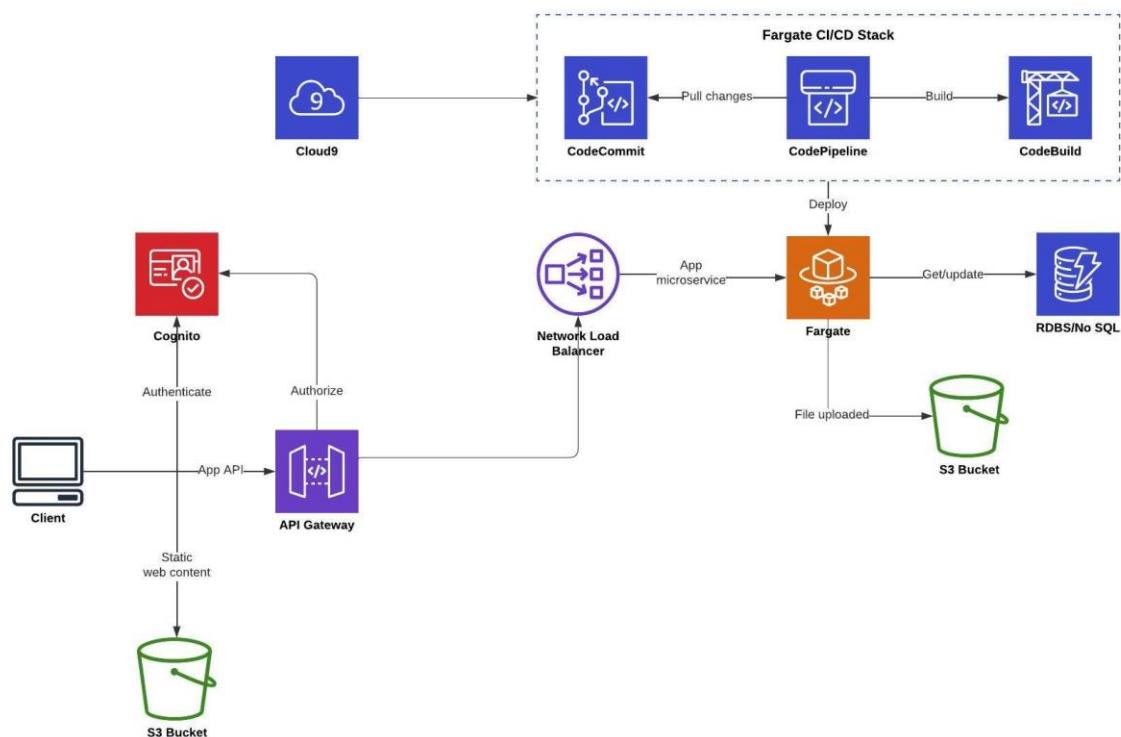
- **Alta Disponibilidad:** La arquitectura utiliza múltiples zonas de disponibilidad (A y B) para asegurar alta disponibilidad y tolerancia a fallos.
- **Escalabilidad:** Con Auto Escalado y Balanceadores de Carga, puede manejar de manera eficiente una carga variable.
- **Consistencia y Redundancia de Datos:** RDS con despliegue Multi-AZ asegura que los datos se respalden y estén disponibles de manera consistente incluso en caso de fallo.
- **Alcance Global:** CloudFront garantiza una entrega rápida de contenido a usuarios en todo el mundo.

Esta configuración tiene como objetivo proporcionar una arquitectura de microservicio robusta y resiliente con alta disponibilidad, escalabilidad y eficiencia, capaz de manejar una carga significativa y asegurar la redundancia de datos.

5.3.1 Diseño técnico general para la integración de cambios

En el entorno dinámico y competitivo de hoy, las organizaciones requieren sistemas robustos que faciliten el desarrollo ágil, la integración continua y la entrega rápida de software. Una solución bien diseñada no solo debe ser eficiente y escalable, sino también capaz de manejar datos en tiempo real para obtener información procesable. En esta tesis, proponemos una arquitectura basada en servicios de AWS (Amazon Web Services) que mejora significativamente la gestión del ciclo de vida del desarrollo del software, la escalabilidad y la integración de datos en tiempo real.

Figura 56: Herramientas utilizadas para la integración de cambios para los microservicios



fuente: propia

Propuesta y Funcionamiento

1. Desarrollo y Gestión de Código:

- **Cloud9** : Es el entorno de desarrollo integrado (IDE) donde los desarrolladores escriben y prueban el código en un entorno basado en la nube.
- **CodeCommit**: Se utiliza para almacenar y gestionar el código fuente del proyecto. Es un repositorio de control de versiones que facilita la colaboración entre equipos.

2. Integración y Entrega Continua (CI/CD):

- **CodePipeline:** Coordina los diferentes pasos del proceso de CI/CD. Toma el código desde CodeCommit, ejecuta construcciones, pruebas y despliegos de manera automática.
- **CodeBuild:** Construye el código fuente, ejecuta pruebas y empaqueta los artefactos necesarios para el despliegue.

3. Despliegue:

- **Fargate:** Despliega los contenedores construidos y los ejecuta de forma serverless, eliminando la necesidad de gestionar la infraestructura subyacente.
- **Network Load Balancer:** Distribuye de manera eficiente el tráfico entrante entre los diferentes microservicios desplegados en Fargate.

4. Gestión de Datos:

- **RDBS /NO SQL:** Proporciona una base de datos Relational o No SQL altamente escalable y de alto rendimiento para almacenar y acceder a los datos de la aplicación, basada en la necesidad de cada microservicio.

5. Interacción del Cliente:

- **Cognito:** Gestiona la autenticación y autorización de los usuarios. Facilita la creación y gestión de usuarios y permite una autenticación segura.
- **API Gateway:** Funciona como un punto de entrada unificado para que las solicitudes del cliente sean redirigidas a los microservicios en Fargate.

6. Contenido Estático:

- **S3 Bucket (arriba a la izquierda):** Ofrece almacenamiento de contenido estático, como archivos HTML, CSS y JS, que pueden ser servidos directamente al cliente.

Ventajas de Este Sistema

1. Escalabilidad:

- Fargate y el Network Load Balancer permiten que la aplicación escale automáticamente en respuesta a la demanda, mejorando la disponibilidad y rendimiento sin necesidad de gestión manual de servidores.

2. Seguridad:

- Cognito proporciona una capa integral de seguridad para la autenticación y autorización de usuarios, asegurando que solo usuarios autorizados puedan acceder a la aplicación.

3. Gestión eficiente del Ciclo de Vida del Desarrollo:

- CodePipeline, CodeBuild y CodeCommit permiten una integración y entrega continua (CI/CD) eficaz, reduciendo el tiempo de despliegue y mejorando la calidad del código.

4. Desarrollo y Despliegue Serverless:

- Servicios Fargate reducen la necesidad de gestionar infraestructura, permitiendo a los desarrolladores centrarse en la lógica de negocio y funcionalidades clave.

5. Almacenamiento y Procesamiento de Datos en Tiempo Real:

- DynamoDB, junto con Kinesis Data Firehose, proporciona soluciones eficientes para el almacenamiento y procesamiento en tiempo real, asegurando que los datos sean precisos y estén siempre actualizados.

6. Alta disponibilidad y Resiliencia:

- La arquitectura basada en AWS ofrece alta disponibilidad y resistencia a fallos, garantizando la fiabilidad de la aplicación.

La arquitectura propuesta ofrece una solución integral que mejora significativamente la integración de datos actuando en tiempo real y optimizando el ciclo de vida del desarrollo de software. La elección de una arquitectura serverless y servicios gestionados libera a los equipos de desarrollo de la carga de administración de la infraestructura, permitiendo que se concentren en impulsar la innovación y la calidad del producto. Al implementar un flujo de CI/CD robusto y soluciones de

almacenamiento y procesamiento de datos eficaces, las organizaciones pueden responder rápidamente a las necesidades cambiantes del mercado y ofrecer experiencias de usuario excepcionales, todo ello mientras se optimizan los costos operativos y se asegura la escalabilidad y seguridad del sistema.

5.4 Análisis de Resultados

Una vez realizada la integración de los procesos se obtiene como resultado un pipeline de procesamiento que permite escalar el tamaño de datos a procesar, este construido bajo la tecnología Apache Spark que permitirá disminuir, los tiempos de procesamiento, recursos utilizados, fallos durante la ejecución.

La evidencia de esto son los resultados encontrados a través de los diferentes estudios y análisis ejecutados por diferentes empresas. En estos se detalla y demuestra como Apache Spark impacta en las diferentes métricas de un sistema de procesamiento por lotes distribuido. Estas se detallan a continuación.

- **Rendimiento:** Apache Spark utiliza computación en memoria y optimizaciones avanzadas, como el almacenamiento en columnas y la inserción de predicados, para procesar grandes cantidades de datos mucho más rápido que MapReduce tradicional. La API de PySpark proporciona una integración perfecta con Python, lo que facilita que los científicos e ingenieros de datos utilicen Spark usando código de Python.

Un estudio de referencia realizado por IBM mostró que Apache Spark procesó una carga de trabajo por lotes 100 veces más rápido que Hadoop MapReduce y 10 veces más rápido que los sistemas tradicionales basados en disco. (Fuente: [IBM, "Apache Spark vs Hadoop", A comprehensive performance analysis of Apache Hadoop and Apache Spark for large scale data sets using HiBench](#)).

- **Latencia:** Apache Spark utiliza un motor de ejecución DAG (gráfico acíclico dirigido) avanzado, que le permite procesar datos mucho más rápido y con menor latencia que MapReduce tradicional. El motor DAG de Spark procesa datos en paralelo, lo que reduce el tiempo necesario para procesar grandes cantidades de datos y aumenta el rendimiento general. (Fuente: *UC Berkeley RAD Lab, "Spark: Cluster Computing with Working Sets", Low latency analytics for streaming traffic data with Apache Spark*)
- **Disponibilidad:** Apache Spark tiene soporte integrado para tolerancia a fallas y puede recuperarse de fallas de nodos automáticamente. Spark vuelve a ejecutar automáticamente las tareas fallidas en otros nodos, lo que garantiza que el trabajo continúa incluso si falla un nodo. Esto es particularmente importante para procesar grandes cantidades de datos, ya que elimina la necesidad de intervención manual y reduce el tiempo de inactividad. (Fuente: CSAIL, "[Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing](#)")
- **Escalabilidad:** Apache Spark está diseñado para escalar horizontalmente, lo que le permite procesar cantidades crecientes de datos a medida que se agregan más nodos al clúster. Spark puede ejecutarse en un clúster de hardware básico, lo que facilita el escalado horizontal a medida que crece la cantidad de datos. Spark también admite la asignación dinámica de recursos, lo que le permite asignar recursos automáticamente según sea necesario en función de las demandas de la aplicación.

Apache Spark puede escalar horizontalmente para manejar cantidades crecientes de datos a medida que se agregan más nodos al clúster. Un análisis de referencia realizado por IBM mostró que Apache Spark podía escalar linealmente, procesando más de 100 TB de datos en un clúster de más de 10000 núcleos. (Fuente: [IBM, "How Spark Beat Hadoop 100 TB Daytona GraySort Challenge"](#))

- **Tolerancia a fallas:** Apache Spark tiene soporte integrado para la tolerancia a fallas, lo cual es importante para procesar grandes cantidades de datos. Spark vuelve a ejecutar automáticamente las tareas fallidas en otros nodos, lo que garantiza que el trabajo continúe incluso si falla un nodo. Esto elimina la necesidad de intervención manual y reduce el tiempo de inactividad.

Apache Spark tiene soporte integrado para la tolerancia a fallas, con funciones como la recuperación automática de tareas y la conmutación por error automática. Un estudio de la universidad Berkeley mostró que Apache Spark puede ofrecer 20 veces más disponibilidad que otras soluciones en las implementaciones de producción. (Fuente: *Universidad de Berkeley "[Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing](#)"*)

- **Utilización de recursos:** Apache Spark tiene soporte integrado para la administración de clústeres y la asignación de recursos, lo que le permite asignar recursos automáticamente según sea necesario en función de las demandas de la aplicación. Spark también proporciona un control detallado sobre los recursos asignados a cada tarea, lo que permite a los científicos e ingenieros de datos optimizar la utilización de recursos y minimizar el desperdicio.

Apache Spark proporciona un control detallado sobre la asignación de recursos, lo que permite a los científicos e ingenieros de datos optimizar la utilización de los recursos y minimizar el desperdicio. Un estudio de referencia realizado por M. Bellafkih mostró que Apache Spark pudo lograr una mejor utilización de los recursos del clúster hasta un 50 % en comparación con Hadoop MapReduce. (Fuente: *Researchgate, "[Leveraging resource management for efficient performance of Apache Spark](#)"*)

La siguiente es una tabla comparativa entre el sistema actual y el propuesto, detallando los criterios comparables respecto al presente capítulo.

Criterio	Sistema actual	Sistema propuesto
Rendimiento	Dado que la ejecución está vinculada al lenguaje de programación de la implementación y a los recursos disponibles, no es escalable.	La ejecución se delega a Apache Spark en un entorno distribuido, lo que posibilita aumentar el porcentaje de respuestas exitosas y ejecuciones exitosas.
Latencia	La ejecución, al depender del lenguaje de programación de la implementación y de los recursos disponibles, junto con la dificultad para escalar, provoca un aumento de la latencia durante picos de tráfico que desafían el rendimiento del lenguaje y framework web usado.	Cuando la importación de datos se realiza en un entorno distribuido, se observa una notable reducción en la latencia ya que se paralleliza el procesado de datos.
Disponibilidad	La disponibilidad de la plataforma, y en particular de los procesos en ejecución, se vuelve inestable debido a la dependencia de recursos y servidores locales (on-premise).	En un entorno distribuido, la importación de datos se realiza de manera que, en caso de fallos, otros nodos asumirán el proceso que falló, lo que resulta en un aumento de la disponibilidad.
Escalabilidad	Dado que los recursos son locales (on-premise), solo podemos escalar utilizando los recursos disponibles. Además, el proceso de escalado implica varios pasos y presupuesto.	En un entorno en la nube (AWS), es sencillo escalar cuando se producen picos de tráfico o se requiere importar una cantidad mayor de datos de lo habitual.
Tolerancia a fallas	Cuando ocurre una falla en la importación, es necesario limpiar los datos y realizar una nueva importación de forma manual.	Durante la ejecución, se detectan los lotes que fallan y se inicia un proceso de reintento, cambiando el nodo de ejecución.

Utilización de recursos	Todos los recursos se ejecutan en el mismo sistema, y esto, junto con la falta de optimización en las implementaciones, podría causar fallos en procesos cercanos	Los procesos que gestionan las solicitudes, validan los datos y realizan la importación se ejecutan en máquinas separadas en la nube
Trazabilidad	Genera métricas exclusivamente relacionadas con los servidores en cuanto a métricas de sistema se refiere.	Utiliza herramientas como Datadog para realizar un seguimiento de las métricas de sistema y de aplicación. Datadog te permite generar métricas de negocio al realizar agregaciones.

En comparación, la utilización Celery con Django y algunas máquinas locales en el proceso actual, carece de la capacidad de escalar eficientemente de forma distribuida y es posible que no pueda orquestar grandes cantidades de datos de manera eficiente. Las capacidades de rendimiento y tolerancia a fallas también son limitadas.

En conclusión, el uso de Apache Spark y Python para procesar grandes cantidades de datos ofrece un mejor rendimiento, latencia, disponibilidad, escalabilidad, tolerancia a fallas y administración de recursos en comparación con el uso de Celery con Django y tres máquinas locales.

CAPÍTULO 6: CONCLUSIONES

Reducir el tiempo de desarrollo de cambios a las validaciones de tipo (TypeChecking) ejecutadas por los importadores, utilizando una interfaz para la definición de estas.

La iniciativa de implementar una interfaz para la definición de validaciones de tipo (TypeChecking) en los importadores ha resultado ser una estrategia altamente beneficiosa para abordar los desafíos previos que enfrentaba el equipo de desarrollo. Anteriormente, agregar cualquier validación de tipo implicaba una serie de pasos engorrosos y una considerable inversión de tiempo. Los desarrolladores debían modificar directamente el código relacionado con los importadores, lo que implicaba crear ramas, generar un pull request (PR) y someterse a exhaustivas revisiones de código antes de que cualquier cambio pudiera ser finalmente desplegado. Este proceso se volvía especialmente complejo y lento cuando se requerían múltiples validaciones, lo que ralentizaba la entrega de nuevas funcionalidades y generaba un cuello de botella en el ciclo de desarrollo.

Con la nueva propuesta de integrar una interfaz que permita gestionar las validaciones de tipo, todo este proceso ha experimentado una notable transformación. La interfaz proporciona a los desarrolladores una herramienta centralizada y accesible para definir, configurar y actualizar las validaciones necesarias sin la necesidad de intervenir directamente en el código de los importadores. Esta interfaz, en forma de una intuitiva interfaz de usuario, permite definir las reglas y condiciones de validación de manera mucho más sencilla y rápida, lo que ha eliminado la necesidad de manipular directamente el código y ha reducido significativamente la posibilidad de errores en el proceso.

Además de simplificar la definición de validaciones, la interfaz también ha agilizado el proceso de revisión y aprobación de cambios. Anteriormente, cada modificación de código requería una revisión minuciosa para garantizar que no afectará negativamente otras partes del sistema. Ahora, al poder gestionar las validaciones a través de la interfaz, los cambios son más granulares y más fáciles de verificar, lo que

ha permitido acelerar el proceso de revisión y aumentar la confianza en la calidad del software entregado.

Un beneficio adicional y significativo es la flexibilidad que brinda esta interfaz en el proceso de mantenimiento y actualización del sistema. Antes, si una validación requería modificaciones o ajustes posteriores, el equipo debía repetir todo el proceso de desarrollo y despliegue, lo que implicaba una dedicación de tiempo considerable. Con la interfaz, las actualizaciones de validaciones son rápidas y simples, lo que facilita la adaptación a nuevos requisitos y necesidades del negocio sin retrasar el lanzamiento de nuevas versiones.

En resumen, la implementación de esta interfaz para la definición de validaciones de tipo ha revolucionado el enfoque del equipo de desarrollo, optimizando el flujo de trabajo y reduciendo drásticamente el tiempo de desarrollo y despliegue de cambios. La capacidad de gestionar las validaciones desde una interfaz de usuario ha agilizado significativamente el proceso de desarrollo, eliminando la necesidad de modificar directamente el código y simplificando las revisiones y aprobaciones. Asimismo, la flexibilidad de la interfaz ha permitido al equipo adaptarse rápidamente a nuevas necesidades y requisitos, lo que se traduce en una mayor capacidad de respuesta y una mayor eficiencia en la entrega de funcionalidades. En definitiva, esta innovadora propuesta ha demostrado ser una solución sólida y eficaz para mejorar la calidad y productividad del equipo de desarrollo, sentando las bases para un futuro desarrollo de software más ágil y eficiente.

Reducir los tiempos de importación automatizando y paralelizando la configuración de las fórmulas contenidas en el archivo excel y la importación de los datos respectivamente.

En conclusión, al reducir los tiempos de importación automatizando y paralelizando la configuración de las fórmulas contenidas en el archivo Excel y la importación de los datos respectivamente, se logra implementar un pipeline de procesamiento escalable utilizando Apache Spark. Los beneficios de utilizar Apache Spark incluyen un

rendimiento superior, menor latencia, mayor disponibilidad, escalabilidad efectiva, tolerancia a fallas integrada y una mejor utilización de los recursos del clúster.

Esto se respalda con los resultados de diversos estudios y análisis realizados por diferentes empresas e instituciones. Por ejemplo, según un estudio de referencia realizado por (Fuente: IBM, "[How Spark Beat Hadoop 100 TB Daytona GraySort Challenge](#)"), Apache Spark procesó una carga de trabajo por lotes 100 veces más rápido que Hadoop MapReduce y 10 veces más rápido que los sistemas tradicionales basados en disco. Además, el motor de ejecución DAG de Apache Spark, mencionado por UC Berkeley RAD Lab, [Low latency analytics for streaming traffic data with Apache Spark](#)), permite procesar datos en paralelo, lo que reduce el tiempo necesario para procesar grandes cantidades de datos y mejora el rendimiento general.

Apache Spark también ofrece tolerancia a fallas incorporada, lo cual es crucial al procesar grandes volúmenes de datos. El sistema puede recuperarse automáticamente de las fallas de los nodos y volver a ejecutar las tareas en otros nodos, evitando así la intervención manual y reduciendo el tiempo de inactividad.

La escalabilidad también es un factor destacado, ya que Apache Spark puede escalar horizontalmente al agregar más nodos al clúster, lo que permite procesar mayores volúmenes de datos. Un análisis de referencia realizado por (Fuente: IBM, [A comprehensive performance analysis of Apache Hadoop and Apache Spark for large scale data sets using HiBench](#)), demostró que Apache Spark puede escalar linealmente, procesando más de 100 TB de datos en un clúster de más de 10,000 núcleos.

La administración de recursos es otra ventaja de Apache Spark, ya que proporciona control detallado sobre la asignación de recursos, lo que optimiza la utilización de los recursos y minimiza el desperdicio. En comparación con el uso de Celery con Django y máquinas locales, Apache Spark supera en rendimiento, tolerancia a fallas y utilización de recursos.

En resumen, la implementación de Apache Spark y Python para procesar grandes volúmenes de datos ofrece mejoras significativas en rendimiento, latencia, disponibilidad, escalabilidad, tolerancia a fallas y gestión de recursos en comparación con soluciones como Celery con Django y máquinas locales. Estos beneficios respaldan la reducción de los tiempos de importación y permiten un procesamiento eficiente y escalable de los datos.

Aumentar el volumen de importaciones simultáneas soportadas diariamente en la plataforma al implementar un sistema de procesamiento por lote distribuido.

En conclusión, al implementar un sistema de procesamiento por lote distribuido, se logra aumentar significativamente el volumen de importaciones simultáneas soportadas diariamente en la plataforma. Esta mejora es especialmente relevante considerando que el sistema legacy utilizado previamente carecía de tecnologías modernas y estrategias de optimización, lo que limitaba la capacidad para manejar grandes volúmenes de importaciones de manera eficiente.

El uso de un sistema de procesamiento por lote distribuido, como Apache Spark, permite realizar importaciones de forma simultánea y paralela, distribuyendo la carga de trabajo entre múltiples nodos del clúster. Esto se traduce en una mayor capacidad para procesar múltiples archivos al mismo tiempo, lo que aumenta la productividad y la eficiencia de la plataforma.

La capacidad de paralelismo del sistema distribuido optimiza la utilización de los recursos disponibles, acelerando el procesamiento de los archivos de importación. Al aprovechar las ventajas de tecnologías modernas, como Apache Spark, se obtiene un rendimiento superior, menor latencia y escalabilidad efectiva, lo que permite manejar volúmenes cada vez mayores de importaciones diarias, todo esto corroborado por diferentes análisis tales como, "[How Spark Beat Hadoop 100 TB Daytona GraySort Challenge](#)" realizado por IBM y [Low latency analytics for streaming traffic data with Apache Spark](#) realizado por UC Berkeley RAD Lab.

En resumen, la implementación de un sistema de procesamiento por lote distribuido mejora significativamente la capacidad de la plataforma para soportar un mayor

volumen de importaciones simultáneas diariamente. Esto se logra al aprovechar el paralelismo, la optimización y las capacidades de escalabilidad ofrecidas por tecnologías modernas como Apache Spark. En comparación con el sistema legacy, que carecía de estas capacidades, el nuevo enfoque brinda una solución más eficiente y capaz de satisfacer las demandas crecientes de importaciones en la plataforma.

CAPÍTULO 7: RECOMENDACIONES

En este capítulo, se presentan recomendaciones clave para la utilización efectiva de la plataforma diseñada en esta tesis, que combina herramientas como Apache Spark, Django, AWS, Datadog y técnicas de importación de datos desde archivos Excel a un almacén de datos. Estas recomendaciones se basan en la experiencia adquirida durante la investigación y el desarrollo de la propuesta, y tienen como objetivo mejorar la eficiencia, la seguridad y la gestión de datos en el contexto de la plataforma.

Buenas prácticas en la importación de datos

Validación de datos: Antes de importar datos desde archivos Excel, se recomienda implementar un proceso de validación riguroso para asegurarse de que los datos sean coherentes y cumplan con los requisitos de la base de datos.

Control de duplicados: Establecer mecanismos para detectar y manejar duplicados en los datos importados. Esto garantizará la integridad de la base de datos y evitará problemas posteriores.

Monitoreo y métricas

Implementación de Datadog: Hacer un uso completo de las capacidades de Datadog para monitorear el rendimiento y la salud de la plataforma. Definir métricas y alertas relevantes que permitan una respuesta rápida ante posibles problemas.

Logging efectivo: Establecer registros (logs) detallados y estructurados para rastrear eventos y actividades en la plataforma. Los registros deben incluir información relevante para la solución de problemas y auditorías.

Seguridad y gestión de acceso

Seguridad de la plataforma: Implementar medidas de seguridad robustas en todas las capas de la plataforma, desde la infraestructura de AWS hasta la aplicación Django. Mantener actualizadas las bibliotecas y componentes de software para evitar vulnerabilidades conocidas.

Gestión de acceso y roles: Definir roles y permisos de usuario adecuados en la plataforma. Limitar el acceso a datos y funcionalidades según las responsabilidades de cada usuario.

Escalabilidad y rendimiento

Optimización de Spark: Si la cantidad de datos a importar crece significativamente, considere estrategias de optimización de Apache Spark para garantizar un rendimiento óptimo.

Escalabilidad en AWS: Diseñar la infraestructura en AWS de manera que sea fácilmente escalable para manejar aumentos en la carga de trabajo. Utilizar servicios como Amazon RDS o Amazon Redshift según sea necesario.

Mantenimiento y documentación

Documentación completa: Mantener una documentación actualizada y exhaustiva que describa la arquitectura de la plataforma, los procedimientos de importación de datos, las políticas de seguridad y los pasos para la solución de problemas.

Mantenimiento proactivo: Establecer un programa de mantenimiento proactivo que incluya actualizaciones regulares de software, copias de seguridad programadas y pruebas de recuperación ante desastres.

Estas recomendaciones están diseñadas para ayudar a los usuarios y administradores de la plataforma a aprovechar al máximo de funcionalidad y a mantenerla en un estado óptimo de rendimiento y seguridad. Al seguir estas pautas,

se puede garantizar una experiencia exitosa en la importación de datos y la gestión de la plataforma en general.

8. BIBLIOGRAFÍA

- 1 Vieira, A., Soares, N., & Sousa, S. D. (2017, December). *Implementing the balanced scorecard in excel for small and medium enterprises*. In 2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM) (pp. 2361-2365). IEEE.
- 2 Hung, V., Benatallah, B., & Saint-Paul, R. (2011, October). *Spreadsheet-based complex data transformation*. In Proceedings of the 20th ACM international conference on Information and knowledge management (pp. 1749-1754).
- 3 Mattessich, R. (1961). *Budgeting models and system simulation*. *The Accounting Review*, 36(3), 384.
- 4 Mattessich, R. (1989). *Accounting and the input—output principle in the prehistoric and ancient world*. *Abacus*, 25(2), 74-84.
- 5 Mattessich, R. (1964). *Simulation of the firm through a budget computer program*. RD Irwin.
- 6 Shigarov, A. O., & Mikhailov, A. A. (2017). *Rule-based spreadsheet data transformation from arbitrary to relational tables*. *Information Systems*, 71, 123-136.
- 7 Vassiliadis, P. (2009). *A survey of extract–transform–load technology*. *International Journal of Data Warehousing and Mining (IJDWM)*, 5(3), 1-27.
- 8 Ying, L., Yu-feng, H., Li-zhou, F., & Yang, W. (2010, July). *Design and implementation of excel massive data intelligent import system*. In 2010 3rd International Conference on Computer Science and Information Technology (Vol. 2, pp. 328-330). IEEE.
- 9 Julia, Guiem. (2018, Feb). Patrones de diseño. GoDevel. recuperado de <https://godevel.com/blog/patrones-de-diseno/>.
- 10 Cardelli, L. (1996). *Type systems*. *ACM Computing Surveys (CSUR)*, 28(1), 263-264. [link](#)
- 11 ERWIG, Martin. *Typed Table Transformations*. arXiv preprint arXiv:1809.02746, 2018. [link](#)
- 12 Francis Galiegue and Kris Zyp. 2013. JSON Schema draft 04. <http://json-schema.org/draft-04/json-schema-validation.html>
- 13 Felipe Pezoa, Juan L. Reutter, Fernando Suarez, Martín Ugarte, and Domagoj Vrgoč. 2016. *Foundations of JSON Schema*. In International Conference on World Wide Web (WWW). 263–273. <https://doi.org/10.1145/2872427.2883029>
- 14 JSON Schema Validators. Disponible en: <<https://json-schema.org/implementations.html#validators>>. Accesado el 21/06/2022.

- 15 *P (clase de complejidad)*. (2022, 20 de junio). Wikipedia, La enciclopedia libre. Fecha de consulta: 01:58, julio 21, 2022 desde [https://es.wikipedia.org/w/index.php?title=P_\(clase_de_complejidad\)&oldid=144303605](https://es.wikipedia.org/w/index.php?title=P_(clase_de_complejidad)&oldid=144303605).
- 16 MongoDB Schema Validation Disponible en:
<<https://www.mongodb.com/docs/manual/reference/operator/query/jsonSchema/>>, Acesado el 21/07/2022.
- 17 George K. Thiruvathukal, Cameron Christensen, Xiaoyong Jin, François Tessier, Venkatram Vishwanath. Type safety with JSON subschemaA Benchmarking Study to Evaluate Apache Spark on Large-Scale Supercomputers. arXiv preprint arXiv:1904.11812, 2019. [link](#)
- 18 Aivaloglou, Efthimia & Hoepelman, David & Hermans, Felienne. (2017). Parsing Excel formulas: A grammar and its application on 4 large datasets: AIVALOGLOU et al .. Journal of Software: Evolution and Process. 29. e1895. 10.1002/smrv.1895. [link](#)
- 19 Apache spark, wikipedia. [link](#)
- 20 Directed acyclic graph, wikipedia. [link](#)
- 21 Wikipedia contributors. (2022, November 7). Single source of truth. In Wikipedia, The Free Encyclopedia. Retrieved 10:29, November 14, 2022, from https://en.wikipedia.org/w/index.php?title=Single_source_of_truth&oldid=1120485355
- 22 Glosario de Conceptos XML. Glosario de Conceptos. (n.d.). Retrieved December 2, 2022, from <https://www.ine.es/DEFIne/es/concepto.htm?c=3881&op=30169&p=1&n=20>
- 23 "XML 1.0 Specification". World Wide Web Consortium. Retrieved 22 August 2010.
- 24 Openpyxl 2.1.4, 2014, openpyxl - A Python library to read/write Excel 2010 xlsx/xlsm files, <https://foss.heptapod.net/openpyxl/openpyxl>
- 25 XlsxWriter 0.8.0, 2019 - XlsxWriter is a Python module for creating Excel XLSX files, <https://github.com/jmcnamara/XlsxWriter>
- 26 data-importer 2.2.1 - 2015, a tool which allow you to transform easily a CSV, XML, XLS and XLSX file into a python object or a django model instance, <https://github.com/valdergallo/data-importer>
- 27 Spark: Cluster computing with working sets - University of California ... Available at: <https://amplab.cs.berkeley.edu/wp-content/uploads/2011/06/Spark-Cluster-Computing-with-Working-Sets.pdf>

- 28 Low latency analytics for streaming traffic data with Apache Spark. Available at:
https://www.researchgate.net/publication/283896069_Low_latency_analytics_for_streaming_traffic_data_with_Apache_Spark
- 29 Resilient distributed datasets: A fault-tolerant abstraction for in ... Available at:
https://people.csail.mit.edu/matei/papers/2012/nsdi_spark.pdf
- 30 Advanced apache spark meetup: How spark beat hadoop @ 100 TB daytona graysort challenge (no date a) PPT. Available at: <https://www.slideshare.net/cfregly/advanced-apache-spark-meetup-how-spark-beat-hadoop-100-tb-daytona-graysort-challenge>
- 31 Education, I.C. (2021) Hadoop vs. spark: What's the difference?, IBM Blog. Available at:
<https://www.ibm.com/blog/hadoop-vs-spark>
- 32 Ahmed, N. et al. (2020) A comprehensive performance analysis of Apache Hadoop and Apache Spark for large scale data sets using HiBench - Journal of Big Data, SpringerOpen. Available at:
<https://journalofbigdata.springeropen.com/articles/10.1186/s40537-020-00388-5>
- 33 Staudt, M., Vaduva, A., Vetterli, T., 1999. Metadata Management and Data Warehousing. Technical Report, The Department of Information Technology (IFI) at the University of Zurich.
- 34 Madhavan, J., Bernstein, P.A., Rahm, E., 2001. Generic schema matching with cupid. In: Proceedings of the 27th International Conferences on Very Large Databases, pp. 49–58.
- 35 Rifaiyah, R., Benharkat, N.A., 2002. Query-based data warehousing tool. In: Proceedings of the Fifth ACM International Workshop on Data Warehousing and OLAP, November 2002.
- 36 petl - Python Extract Transform and Load Tables of Data, <https://github.com/petl-developers/petl>
- 37 Bonobo - Bonobo is a data-processing toolkit for python 3.5+, <https://www.bonobo-project.org/>
- 38 Apache Airflow - a platform created by the community to programmatically author, schedule and monitor workflows, <https://airflow.apache.org/>
- 39 Luigi - a Python module that helps you build complex pipelines of batch jobs. It handles dependency resolution, workflow management, visualization etc. It also comes with Hadoop support built in., <https://github.com/spotify/luigi>

*40 Perfect - Modern workflow orchestration for data and ML engineers,
<https://www.prefect.io/>*