**DISEÑO E IMPLEMENTACIÓN DE MODULO FUNCIONAL EN PYTHON PARA LOS PROCESOS DE POST CÁLCULO PARA LOS MODELOS SUPPLY CHAIN CCUS Y H2**

**DESARROLLO DE PROTOTIPOS PARA LA INTEGRACIÓN LENGUAJE DE PROGRAMACIÓN PYTHON GAMS**

**Jorge Eduardo Diaz Sierra**

Ingeniero químico

Esp. Ingeniería de Yacimientos

Desarrollador Python-Java

**Crisóstomo Alberto Barajas Solano**

Ingeniero de Sistemas

Magister en Ingeniería de Sistemas e Informática

PhD (c) en Ingeniería, área Ingeniería Electrónica

**Dirigido a:**

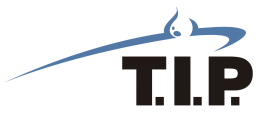
**Luz Marina Rozo Vargas**

Coordinador Técnico – TIP

**Ariel Uribe**

Líder Técnico del Proyecto – Ecopetrol





**Contrato 3041167**

**Orden de Servicio No. ODS 137\_ 5603028 Item 8 Entregable No 1**

**Servicios Técnicos Especializados para el Centro de Innovación**

**y Tecnología del Instituto Colombiano del Petróleo de Ecopetrol S.A.**

**20 de junio de 2022**



**TABLA DE CONTENIDO**

[RESUMEN EJECUTIVO 4](#_Toc106270128)

[1. INTRODUCCIÓN 4](#_Toc106270129)

[2. ANÁLISIS Y DISEÑO MODULO DE POST CALCULO 4](#_Toc106270130)

[3. MODULO DE GENERACIÓN DE ARCHIVOS JSON 8](#_Toc106270131)

[3.1. TECNOLOGÍAS USADAS 8](#_Toc106270132)

[3.2. INTERFAZ GRÁFICA 8](#_Toc106270133)

[4. MODULO DE EXTRACCIÓN TRANSFORMACION Y CARGA 10](#_Toc106270134)

[4.1. TECNOLOGIAS USADAS. 10](#_Toc106270135)

[4.2. EXTRACCIÓN 11](#_Toc106270136)

[4.2.1 ARCHIVOS TIPO JSON DE INGRESO. 12](#_Toc106270137)

[4.2.2. NOMBRES DE LOS ARCHIVOS (.json) 13](#_Toc106270138)

[4.2.3. TRANSFORMACIÓN 14](#_Toc106270139)

[4.2.4. CARGA 14](#_Toc106270140)

[4.3. DESCRIPCIÓN DE ALGORITMO PARA LA EXTRACCIÓN Y POSTCALCULO DE INFORMACION DE ARCHIVOS TIPO db GENERADOS POR GAMS. 14](#_Toc106270141)

[4.3.1. DESCRIPCION DEL CODIGO GENERAL 14](#_Toc106270142)

[4.3.2. DIAGRAMA DE SECUENCIA ALGORITMO 15](#_Toc106270143)

[4.3.2.1. CLASE FORMA DE EJECUCIÓN (MainExecutionDataframeJsonFiles) 17](#_Toc106270144)

[4.3.3. EJECUCIÓN NATIVA SQL (MainExecutionSQLQuery) 17](#_Toc106270145)

[4.3.4. EJECUCIÓN TIPO COLUMNA (MainExecutionColumnsFunc) 18](#_Toc106270146)

[4.3.4.1. EJECUCIÓN SQL COLUMNA (FunctionReadPandasDFChuckIns) 18](#_Toc106270147)

[4.3.4.2. EJECUCIÓN PARAMETROS-VARIABLES-FORMULA (FunctionReadPandasDFChuckIns) 19](#_Toc106270148)

[4.3.4.3. OPERACIÓN MERGE (MergeDataSetsColumns) 20](#_Toc106270149)

[4.4. INTERFAZ GRÁFICA 21](#_Toc106270150)

[5. CONCLUSIONES Y RECOMENDACIONES 23](#_Toc106270151)

[ANEXO A DIAGRAMA DE SECUENCIA DE EJECUCION EXTRACCIÓN TRANSFORMACION Y CARGA 25](#_Toc106270152)

**LISTA DE TABLAS Y FIGURAS**

[Tabla 1. Sintaxis para nombrar los archivos .json 14](#_Toc106270103)

[Tabla 2. Explicación extracción y transformación de datos 15](#_Toc106270104)

[Figura 1. Casos de usos para el módulo de configuración 6](#_Toc106270105)

[Figura 2. Diagrama de ejecución para el uso de introducción y selección de operaciones de datos 6](#_Toc106270106)

[Figura 3. Diagrama de ejecución para el uso de selección de modelo y versión 7](#_Toc106270107)

[Figura 4. Casos de usos para el módulo de ejecución 8](#_Toc106270108)

[Figura 5. Diagrama de ejecución para el uso de ejecución de sabanas de datos 8](#_Toc106270109)

[Figura 6. Diagrama de ejecución para el uso de cruce de sabanas de datos 9](#_Toc106270110)

[Figura 7. Interfaz principal modulo de generación de archivos JSON. 10](#_Toc106270111)

[Figura 8. Ventana de creación de columnas tipo parámetro/formula 11](#_Toc106270112)

[Figura 9. Ventana de creación de columnas tipo query sql 11](#_Toc106270113)

[Figura 10. Ejemplo de archivo (.json) para la consulta de los archivos (.db). 13](#_Toc106270114)

[Figura 11. Ejemplo de archivo (.json) sentencia SQL nativo. 13](#_Toc106270115)

[Figura 12. Ejemplo de archivo (.json) consulta tipo COLUMNA. 14](#_Toc106270116)

[Figura 13. Ejemplo de extracción y transformación de columnas consultados por medio de archivo (.json) 15](#_Toc106270117)

[Figura 14. Arquitectura base para la aplicación de extracción de datos y POST-CÁLCULO. 16](#_Toc106270118)

[Figura 15. Diagrama de secuencia 16](#_Toc106270119)

[Figura 16. Tipo de ejecución SQL nativo o Columna 18](#_Toc106270120)

[Figura 17. Ejecución SQL nativo. 19](#_Toc106270121)

[Figura 18. Diagrama de Secuencia MainExecutionColumnsFunc. 19](#_Toc106270122)

[Figura 19. Diagrama de ejecución tipo QUERY en Columna 20](#_Toc106270123)

[Figura 20. Algoritmo de extracción y transformación tipo Columna-Parametro-Variable-Formula. 21](#_Toc106270124)

[Figura 21. Ejecución operación merge. 22](#_Toc106270125)

[Figura 22. Interfaz gráfica para la prueba y validación de las rutinas generadas 23](#_Toc106270126)

[Figura 23. Estructura de archivos resultados 23](#_Toc106270127)

RESUMEN EJECUTIVO

La extracción de datos es un tema muy importante en todas las industrias dado que facilita el análisis de los factores requeridos, esto permite comparar resultados entre corridas e implementar algoritmos de las mejores soluciones entre otros.

En este documento se resume la elaboración del código para la lectura y selección de datos de corridas de los modelos de la cadena de valor CCUS y CO2 respectivamente. En este documento se muestran las tecnologías para la multi lectura de los archivos SQL de acuerdo a los archivos tipo (.***json***) de ingreso con sus respectivos archivos (***.csv y .xlsx***) como resultado del filtrado.

1. INTRODUCCIÓN

Los modelos *supply chain* de CCUS y H2 generan una cantidad ingente de resultados numéricos. Estos son generados como archivos GDX o DB, los cuales deben ser analizados y procesados una vez generados. Este proceso, llamado post cálculo, es necesario para obtener información útil para la toma de decisiones.

El módulo de post cálculo se diseñó para ser parametrizado, escalable y adaptable a las necesidades de los usuarios. Para esto utiliza una serie de archivos de configuración en texto plano llamados **JSON** (*JavaScript Object Notation* por sus siglas en inglés) (JSON.org, s.f.). Un archivo JSON es similar en su estructura a un archivo XML dado que tiene una serie de parejas nombre-valor para presentar información estructurada.

Este módulo se divide en dos partes: el módulo de generación de los archivos JSON, diseñada para usuarios con un mínimo de conocimientos técnicos sobre los modelos CCUS y H2; y el módulo de ejecución de las consultadas contenidas en los JSON generados

2. ANÁLISIS Y DISEÑO MODULO DE POST CALCULO

Tal y como se mencionó anteriormente, el módulo de post cálculo se divide en dos herramientas dependientes y separadas. El primero es el módulo de configuración y creación de archivos JSON. Este se encarga de organizar la información contenida en los archivos DB generados a partir de los modelos GDX, y presentar el contenido del modelo a usuarios con un mínimo de conocimientos sobre el modelo CCUS y H2.

La figura 1 presenta el diagrama de casos de uso con las posibles interacciones del usuario con este primer módulo; mientras que las figuras 2 y 3 presenta los diagramas de secuencia para la creación de los archivos JSON.

Diagrama

Descripción generada automáticamente

Figura 1. Casos de usos para el módulo de configuración

Diagrama

Descripción generada automáticamente

Figura 2. Diagrama de ejecución para el uso de introducción y selección de operaciones de datos

Diagrama

Descripción generada automáticamente con confianza media

Figura 3. Diagrama de ejecución para el uso de selección de modelo y versión

El segundo módulo es el de ejecución. Este toma los resultados de los modelos GAMS y los post procesa en función de los archivos JSON especificados. La figura 4 muestra los casos de uso para las posibles interacciones con el usuario, mientras que las figuras 5 y 6 muestran los diagramas de secuencia para los usos de ejecución y cruce de sabanas de datos.

Diagrama

Descripción generada automáticamente

Figura 4. Casos de usos para el módulo de ejecución

Imagen que contiene Texto

Descripción generada automáticamente

Figura 5. Diagrama de ejecución para el uso de ejecución de sabanas de datos

Diagrama

Descripción generada automáticamente

Figura 6. Diagrama de ejecución para el uso de cruce de sabanas de datos

3. MODULO DE GENERACIÓN DE ARCHIVOS JSON

3.1. TECNOLOGÍAS USADAS

* Pandas

Pandas es una herramienta de análisis y manipulación de datos de código abierto rápida, potente, flexible y fácil de usar, construido sobre el lenguaje de programación Python (Pandas, 2022).

* PyQT5

PyQt5 es un conjunto completo de enlaces de Python para Qt v5. Se implementa como más de 35 módulos de extensión y permite que Python se use como un lenguaje de desarrollo de aplicaciones alternativo a C++ en todas las plataformas compatibles, incluidos iOS y Android. PyQt5 también puede integrarse en aplicaciones basadas en C++ para permitir que los usuarios de esas aplicaciones configuren o mejoren la funcionalidad de esas aplicaciones (PyQT.org, 2022).

3.2. INTERFAZ GRÁFICA

El módulo de generación de archivos JSON cuenta con 3 ventanas. La ventana principal mostrada en la figura 7

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 7. Interfaz principal modulo de generación de archivos JSON.

La ventana principal contiene la información específica de la sabana, tales como el nombre, el modelo, la versión, el número de la corrida y el método a usar para procesar las columnas creadas.

Las siguientes dos ventanas, mostradas en las figuras 8 y 9, permiten las creaciones de columnas tipo parámetro/formula y query sql, respectivamente. Es importante que el usuario tenga cuidado al momento de crear las columnas, dado que las dimensiones deben coincidir para todas y cada una de las columnas creadas.

Tabla

Descripción generada automáticamente

Figura 8. Ventana de creación de columnas tipo parámetro/formula

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 9. Ventana de creación de columnas tipo query sql

4. MODULO DE EXTRACCIÓN TRANSFORMACION Y CARGA

4.1. TECNOLOGIAS USADAS.

En el desarrollo de la aplicación se utiliza las tecnologías listadas a continuación:

* Pandas

Pandas es una herramienta de análisis y manipulación de datos de código abierto rápida, potente, flexible y fácil de usar, construido sobre el lenguaje de programación Python (Pandas, 2022).

* sqlite3

SQLite es una biblioteca de C que provee una base de datos ligera basada en disco que no requiere un proceso de servidor separado y permite acceder a la base de datos usando una variación no estándar del lenguaje de consulta SQL. Algunas aplicaciones pueden usar SQLite para almacenamiento interno. También es posible prototipar una aplicación usando SQLite y luego transferir el código a una base de datos más grande como PostgreSQL u Oracle (org, 2022).

* Dash-Plotly

Dash es un marco de Python creado por Plotly para crear aplicaciones web interactivas. Dash está escrito en la parte superior de Flask, Plotly.js y React.js. Con Dash, no tiene que aprender HTML, CSS y Javascript para crear tableros interactivos, solo necesita Python. Dash es de código abierto y la creación de la aplicación utilizando este marco se ve en el navegador web (Science, 2022).

* Archivo base de datos (.db)

Un archivo DB es un archivo de base de datos genérico. Almacena datos en un formato estructurado, generalmente compuesto por un conjunto de tablas, campos de tablas, tipos de datos de campos y valores de campos. Muchas aplicaciones crean y utilizan diferentes tipos de archivos DB específicos de la aplicación (Info, 2022).

* Lenguaje SQL nativo.

SQL se utiliza para comunicarse con una base de datos. Según ANSI (American National Standards Institute), es el lenguaje estándar para los sistemas de gestión de bases de datos relacionales. Las declaraciones SQL se utilizan para realizar tareas como actualizar datos en una base de datos o recuperar datos de una base de datos. Algunos sistemas comunes de gestión de bases de datos relacionales que utilizan SQL son: Oracle, Sybase, Microsoft SQL Server, Access, etc (Course, 2022).

4.2. EXTRACCIÓN

4.2.1 ARCHIVOS TIPO JSON DE INGRESO.

Se crearon los siguientes archivos tipo *(.json)* para usar como interfaz el algoritmo de extracción desarrollado con los respectivos archivos de bases de datos o tipo *(.db)* en este archivo se insertan el modelo que se quiere consultar, la versión, los archivos *(.db)* del modelo y la respectiva ruta donde se encuentran los archivos *(.db)* como se muestra en la figura 1*.*

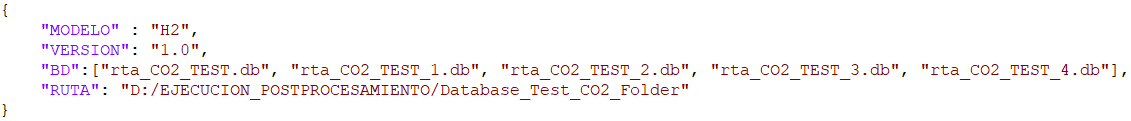


Figura 10. Ejemplo de archivo (.json) para la consulta de los archivos (.db).

Para la consulta de las bases de datos se elaboro el archivo tipo *(.json)* donde también se establece el nombre del modelo versión, nombre de la sabana, dimensiones y sentencia SQL nativo de donde se extrae la sabana de datos como se muestra en la figura 2.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Figura 11. Ejemplo de archivo (.json) sentencia SQL nativo.

Para la extracción por columnas se elaboró el siguiente archivo tipo ***(.json)*** donde cada espacio titulo columna debe contener la información como se muestra en la figura 3, este puede ser tipo SQL, FORMULA, PARAMETRO O VARIABLE respectivamente. Cada columna tiene los nombres de entrada ***(NOMBRE\_E\_S)***, el tipo de campo GAMS que se desea extraer ***(TIPO\_CAMPO)***, nombre que se le da a la columna extraída ***(NOMBRE\_COLUMNA)*** y finalmente la formula que se implementara en la extracción de la columna ***(FORMULA)***.

Diagrama

Descripción generada automáticamente

Figura 12. Ejemplo de archivo (.json) consulta tipo COLUMNA.

4.2.2. NOMBRES DE LOS ARCHIVOS (.json)

Para que el algoritmo reconozca el tipo de archivo que ingresa requiere la siguiente sintaxis en los nombres de los archivos:

Tabla 1. Sintaxis para nombrar los archivos .json

|  |  |  |
| --- | --- | --- |
| Tipo Archivo | Número | Ejemplo |
| Archivo lectura (.db) | 99 | 99\_JSON\_CCUS.json |
| Archivo (.json) tipo sentencia SQL nativa | 98 | 98\_SENTENCIA\_SQL\_WHERE.json |
| Archivo (.json) tipo columna | 98 | 98\_sCaptura.json |

4.2.3. TRANSFORMACIÓN

Los datos de la columna se pueden transformar por medio del campo formula asignando una operación entre los datos consultados, por ejemplo:

Tabla 2. Explicación extracción y transformación de datos

|  |  |
| --- | --- |
| *Datos para extraer* | “P\_CCC y P\_CCamax” |
| *Tipo dato GAMS* | *“value”* |
| *Nombre del campo consultado* | *P\_CCC\_1* |
| Formula |  |

Texto

Descripción generada automáticamente con confianza media

Figura 13. Ejemplo de extracción y transformación de columnas consultados por medio de archivo (.json)

4.2.4. CARGA

Se crean las carpetas en la misma ubicación de directorio donde se encuentran las bases de datos con la siguiente sintaxis:

*NombreCarpeta = 'DT' + '\_' + nombreBaseDatos + '\_' + Modelo + '\_' + Version*

4.3. DESCRIPCIÓN DE ALGORITMO PARA LA EXTRACCIÓN Y POSTCALCULO DE INFORMACION DE ARCHIVOS TIPO db GENERADOS POR GAMS.

4.3.1. DESCRIPCION DEL CODIGO GENERAL

La aplicación de extracción de datos y post calculo tiene dos capas como se muestra en la figura 5. En la capa de presentación se encuentra las secciones HTML y CSS de la aplicación

Diagrama

Descripción generada automáticamente con confianza baja

Figura 14. Arquitectura base para la aplicación de extracción de datos y POST-CÁLCULO.

Por medio de la función *(MainExecutionDatamart)* se conecta la interfaz principal a las acciones de extracción, transformación y carga respectivamente. La figura 6 muestra el diagrama de secuencia de las funciones.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Figura 15. Diagrama de secuencia

4.3.2. DIAGRAMA DE SECUENCIA ALGORITMO

En el *ANEXO A* se encontrará el diagrama de secuencia donde se muestra la forma de consumo de las clases elaboradas. A continuación, se describen las mismas:

* *MainGetJsonFiles*

Clase que da lectura a los archivos *(.json)* segregándolos según tabla 1.

* *SelectProcessPostProcessing.*

Clase que selecciona el tipo de ejecución directo sobre GAMS o sobre archivo *(.db)* por el momento se encuentra sobre el archivo *(.db)* es decir manual.

* *ExecuteSelectProcessPostProcessingPath*

Valida si la ruta ingresada en archivo *(.json)* existe.

* *ConectPathsDBFiles*

Valida si existen archivos (.db) en la ruta ingresada del archivo (.json).

* *ValidateDBFiles*

Organiza los resultados de la lectura de los archivos *(.json).*

* *ResultsValidationFilesDB*

Revisa si los archivos (.db) existen.

* *ValidationReadJsonDataMarkFunc*

Extrae la información de los archivos *(.json)*.

* *ResultValidationJsonFiles*

Elabora tabla de instrucciones para conectar archivos (.db) y extraer los datos según los requerimientos de los archivos (.json) de ingreso.

* *ValidateJsonSentenceSQLDataMarkFunc*

Valida si existe sentencia SQL en caso de ejecutar SQL nativo.

* *SelectExecuteChooseSQLOther*

Valida tipo de ejecución tipo SQL nativo o columna.

* *MainExecutionDataframeJsonFiles*

Extrae, transforma los datos que se quieren extraer de los resultados del modelo CCUS y H2 respectivamente.

* *CreateFolderDatamark*

Crea los directories donde se guardarán las sábanas de datos resultantes respectivamente.

* *CreateFolderFunction*

Se crean los archivos .csv o Excel de las sábanas de datos resultantes.

4.3.2.1. CLASE FORMA DE EJECUCIÓN (MainExecutionDataframeJsonFiles)

Esta clase se encarga de extraer la información de los archivos (.db), posteriormente transformarla si se requiere y hacer las respectivas operaciones MERGE. Como resultado se obtiene la tabla resultante, que luego será guardada como archivo (.csv o .xlsx)

En la figura muestra el tipo de ejecución y de acuerdo a esta se ejecutarán clases diferentes *(MainExecutionSQLQuery)* o *(MainExecutionColumns)* respectivamente.

Diagrama

Descripción generada automáticamente

Figura 16. Tipo de ejecución SQL nativo o Columna

4.3.3. EJECUCIÓN NATIVA SQL (MainExecutionSQLQuery)

En esta clase se lee la ruta de los archivos *(.db)*, hace la conexión a los mismos y ejecuta la sentencia como se muestra en la figura 8

Diagrama

Descripción generada automáticamente

Figura 17. Ejecución SQL nativo.

4.3.4. EJECUCIÓN TIPO COLUMNA (MainExecutionColumnsFunc)

La ejecución tipo columna consta de varias clases que se muestran en el diagrama de secuencia de la figura 9:

Diagrama

Descripción generada automáticamente

Figura 18. Diagrama de Secuencia MainExecutionColumnsFunc.

4.3.4.1. EJECUCIÓN SQL COLUMNA (FunctionReadPandasDFChuckIns)

De manera similar a la ejecución SQL nativo ejecuta la sentencia SQL desde el campo formula.

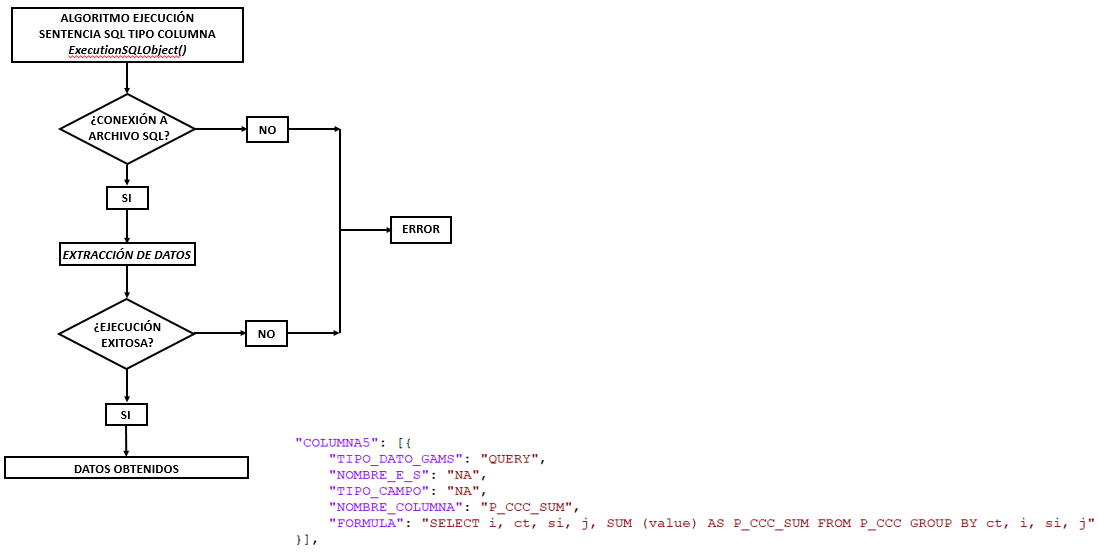


Figura 19. Diagrama de ejecución tipo QUERY en Columna

4.3.4.2. EJECUCIÓN PARAMETROS-VARIABLES-FORMULA (FunctionReadPandasDFChuckIns)

Para la extracción y transformación con tipo FORMULA-PARAMETERS-VARIABLE, se ejecutan las clases ValidateDimensionsColumn(), Join\_Dimension\_Sets() que extraen la información de los conjuntos de Gams y luego hacen su respectiva combinación. Teniendo esta tabla de resultados se pueden extraer los datos de cada columna que se desea. Importante por cantidad de datos se logro la unión de un máximo de 4 conjuntos por la cantidad de datos.

A continuacion de la ejecución de la unión de los conjuntos se extraen los datos de las columnas del campo *NOMBRE\_E\_S* de acuerdo al TIPO\_CAMPO sobre este resultado se ejecuta una operación merge para unir los resultados de la columna con la tabla resultante de los conjuntos. La figura 11 resume este algoritmo.

Diagrama

Descripción generada automáticamente

Figura 20. Algoritmo de extracción y transformación tipo Columna-Parametro-Variable-Formula.

4.3.4.3. OPERACIÓN MERGE (MergeDataSetsColumns)

La operación MERGE se ejecuta por medio del DATAFRAME Pandas implementando ***merge.*** En Esta operación ingresan todas las tablas resultantes y las une hasta que se obtenga una sola. En la figura 12 se muestra la implementación de este algoritmo.Diagrama

Descripción generada automáticamente

Figura 21. Ejecución operación merge.

4.4. INTERFAZ GRÁFICA

Las rutinas anteriormente mencionadas fueron asociadas a una interfaz gráfica, diseñada con Dash Plotly, para su prueba y validación tal y como se muestra en la figura 13. Finalmente, los resultados generados se guardan en disco duro siguiendo la estructura de archivos mencionada anteriormente, tal y como se muestra en la figura 14

Interfaz de usuario gráfica

Descripción generada automáticamente

Figura 22. Interfaz gráfica para la prueba y validación de las rutinas generadas

Tabla

Descripción generada automáticamente

Figura 23. Estructura de archivos resultados

5. CONCLUSIONES Y RECOMENDACIONES

* Se elaboro la herramienta con el uso de los archivos (.db) dado que no se tiene un licenciamiento de Gams y las máquinas de Ecopetrol no tienen acceso a un uso de Python. Si se puede interactuar directamente con GAMS se disminuiría el tiempo de ejecución usando un DATAFRAME como pandas directamente sobre la respuesta del modelo extrayendo y transformando directamente sobre las nuevas versiones de GAMS.
* Probar tecnologías como SPARKS y Hadoop sobre las diferentes respuestas sobre hardware y arquitectura de CLOUD es prioritario.
* Los archivos tipo (.json) permitieron la integración de los procesos de configuración y de ejecución.
* Complementar el prototipo actual con archivos (.json) que personalicen los conjuntos *(sets)* sobre cada columna ayudaría a otro tipo de cálculos que esta versión no presta.

# **BIBLIOGRAFÍA**

Course, S. (14 de 06 de 2022). *SQL Course*. Obtenido de https://www.sqlcourse.com/beginner-course/what-is-sql/

Forbes, E. (2017). *Learning Concurrency in Python.* Packt.

ibm. (02 de 12 de 2021). *IBM API*. Obtenido de https://www.ibm.com/cloud/learn/api

Info, F. (14 de 06 de 2022). *File Info*. Obtenido de https://fileinfo.com/extension/db

JSON.org. (s.f.). *Introducción a JSON*. Recuperado el Junio de 2022, de https://www.json.org/json-es.html

org, d. p. (14 de 06 de 2022). *docs.python.org*. Obtenido de https://docs.python.org/es/3/library/sqlite3.html

Pandas. (14 de 06 de 2022). Obtenido de https://pandas.pydata.org/

PyQT.org. (Junio de 2022). *PyQT5*. Obtenido de https://pypi.org/project/PyQt5/

Romano, F. (2019). *Getting Started with Python.* Packt.

Science, T. D. (14 de 06 de 2022). Obtenido de https://towardsdatascience.com/dash-for-beginners-create-interactive-python-dashboards-338bfcb6ffa4

ANEXO A DIAGRAMA DE SECUENCIA DE EJECUCION EXTRACCIÓN TRANSFORMACION Y CARGA

|  |
| --- |
| Gráfico  Descripción generada automáticamente con confianza media |