MÁSTER EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

Automatización integral de procesos administrativos: Optimización de gestión de Datos y Servicios

Autor: Ángel Guzmán Torres

Director: Ignacio Martínez Vignardi

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Automatización Integral de Procesos Administrativos: Optimización de gestión de Datos y Servicios

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2023/2024 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo. Ángel Guzmán Torres Fecha: …16…/ …07…/ …2024…

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo. Ignacio Martínez Vignardi Fecha: …16…/ …07…/ …2024…

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Automatización integral de procesos administrativos: Optimización de gestión de Datos y Servicios

Autor: Ángel Guzmán Torres

Director: Ignacio Martínez Vignardi

Madrid

Automatización integral de procesos administrativos: Optimización de gestión de Datos y Servicios

Autor: Guzmán Torres, Ángel.
Director: Ignacio Martínez Vignardi.
Entidad Colaboradora: Nfq Advasory.

RESUMEN DEL PROYECTO

El presente proyecto surge de la necesidad de automatizar y optimizar el flujo de datos de una empresa gestora que ofrece servicios de operaciones. El objetivo es mejorar la precisión de los datos y reducir errores (automatización de procesos) de un servicio concreto ofrecido a un Servicer Inmobiliario. Se analiza la situación inicial y se propone un nuevo flujo de trabajo creando una base de datos centralizada.

Palabras clave: Base de datos, automatización de procesos, empresa gestora, servicer inmobiliario, servicio de operaciones.

1. Introducción

Este proyecto tiene como objetivo mejorar los servicios de una Empresa Gestora (EG) para un Servicer Inmobiliario (Servicer) mediante la optimización de procesos administrativos y operativos. Como consultor e ingeniero de datos, se realizará un estudio exhaustivo para identificar y resolver problemas de desempeño. A partir de este análisis, se diseñará un nuevo flujo de trabajo para minimizar el trabajo manual, principal causa de errores y retrasos. Se implementarán herramientas tecnológicas para automatizar procesos, mejorando la precisión y calidad de la información. Esto garantizará una gestión más eficiente y transparente, fortaleciendo la colaboración entre EG y el Servicer.

2. Definición del proyecto

El objetivo principal del proyecto es automatizar el flujo de trabajo y reducir al máximo las tareas manuales. Para lograr este objetivo principal, se han definido los siguientes objetivos secundarios:

- Comprender el flujo de trabajo inicial.
- Identificar problemas, errores y puntos de mejora.
- Reducción del trabajo manual.
- Mejora de precisión en los datos.
- Transparencia y trazabilidad.
- Satisfacción del Servicer.

La primera etapa del proyecto consiste en trabajar estrechamente con los diferentes departamentos de EG que están involucrados en el servicio ofrecido al Servicer. Esta

fase tiene una duración de dos semanas y su objetivo es comprender a fondo el flujo de trabajo existente.

Tras el análisis inicial, se elaborará una estrategia y un plan de mejora detallado. En esta fase se construirá desde cero una base de datos en SQL Server (en adelante, BBDD), que será la herramienta principal del nuevo flujo de trabajo propuesto. La estructura de la BBDD se diseñará para ofrecer diversos beneficios a largo plazo, tales como un mantenimiento sencillo y una fácil detección de problemas.Para poder cargar y extraer los datos en la BBDD, se desarrollarán varios scripts en Python utilizando librerías como Pandas, Selenium y Openpyxl, entre otras.

Dichos scripts facilitarán la manipulación de datos y la automatización de tareas repetitivas. Todos los archivos involucrados en este proyecto se organizarán en diferentes carpetas en Google Drive, donde los coordinadores de los distintos departamentos de EG tendrán acceso para poder recoger y dejar los ficheros de interés.Además, se desarrollarán procedimientos almacenados dentro de la BBDD para asegurar un flujo de datos preciso y eficiente. Estos procedimientos incluirán lógicas específicas para crear validaciones y alertas, así como proporcionar información valiosa a los coordinadores y gestores de EG.

Una vez completado el desarrollo del código, se realizarán pruebas manuales para corregir pequeños detalles y asegurar el correcto funcionamiento del sistema.

3. Contexto y Resultados

El proyecto se centra en la gestión de pagos de deuda y obtención de certificaciones de deuda cero de activos inmobiliarios que el Servicer Inmobiliario tiene en su cartera y llegan a la venta. EG trabaja esa parte administrativa y de gestión, para que el día de la venta en notaría el activo esté libre de deuda y se firme la venta con éxito.

EG verifica que todos los pagos estén al día tanto en el ámbito de la Comunidad de Propietarios (CCPP) como en el de Tributos. Sin embargo, el sistema actual presenta deficiencias significativas, como errores frecuentes, datos incoherentes y pérdida de información, causando insatisfacción en el Servicer. Además, la gestión manual de estos procesos consume una gran cantidad de recursos y tiempo.

Partes Involucradas:

Servicer: Administra una gran cantidad de carteras de activos inmobiliarios, gestionando un promedio de 900 ventas mensuales y exigiendo que EG gestione al menos el 95% de estas ventas. Esto requiere una coordinación diaria constante para garantizar eficiencia y precisión. El Servicer necesita reportes detallados y actualizados para tener una visión clara de las operaciones diarias.

Empresa Gestora (EG): Es responsable de gestionar los pagos de los Tributos y la Comunidad de Propietarios de los inmuebles vendidos.La estructura organizativa incluye coordinadores y gestores:

- Coordinadores: Dos coordinadores, uno para Tributos y otro para CCPP, asignan ventas a los gestores, mantienen contacto diario con el Servicer, corrección de errores, realizan análisis de datos, seguimiento de la facturación, aseguran la eficiencia de las gestiones, y lideran otros servicios ofrecidos a diferentes Servicers.
- Gestores: Divididos en equipos de Tributos y CCPP, los gestores trabajan directamente con administraciones públicas y administradores de fincas para gestionar pagos y asegurar que no haya deudas pendientes en los inmuebles.

Las gestiones realizadas por los Gestores se registran en un fichero Excel recibido por la mañana con las ventas a tratar, al final del día se reporta al Servicer el avance de las mismas.

Cabe destacar que el número de ventas varía a diario, por ello es necesaria la actualización del fichero de ventas por ambas partes: El Servicer actualiza las ventas presentes en el fichero y EG actualiza los campos de gestión de cada venta.

La automatización de tareas, especialmente la carga y validación de datos, ha permitido reducir significativamente el tiempo necesario para completar las actividades de los Coordinadores. Antes, la carga manual de ficheros y la revisión de errores consumían aproximadamente el 20% del tiempo de los coordinadores (1,5 horas al día). Con la implementación de scripts en Python y el uso de procedimientos almacenados en SQL Server, este tiempo se ha reducido a un 1% (5 minutos al día). La única tarea manual es corregir los errores presentes en los ficheros a última hora de la tarde (validaciones automáticas), lo que permite a los Coordinadores enfocarse en actividades de mayor valor añadido y dedicar más tiempo a la gestión de otros servicios, no solo a ventas.

El registro detallado de todas las gestiones en la BBDD ha mejorado la transparencia y trazabilidad de las operaciones. Ahora es posible realizar auditorías internas de manera más eficiente, identificando rápidamente cualquier discrepancia y corrigiéndola en tiempo real. Esto también ha fortalecido la confianza entre EG y el Servicer, ya que se pueden justificar las gestiones realizadas con datos precisos y actualizados, así como justificar las no gestionadas por cuestiones de tiempo.

Además, los Coordinadores pueden realizar tareas de análisis de manera mucho más sencilla gracias al acceso a PowerBI, donde pueden utilizar todas las gráficas y tablas basadas en los datos presentes en la BBDD. Estas tablas se crean en el procedimiento almacenado de generación de informes de la BBDD, proporcionando información valiosa de manera visual y accesible.

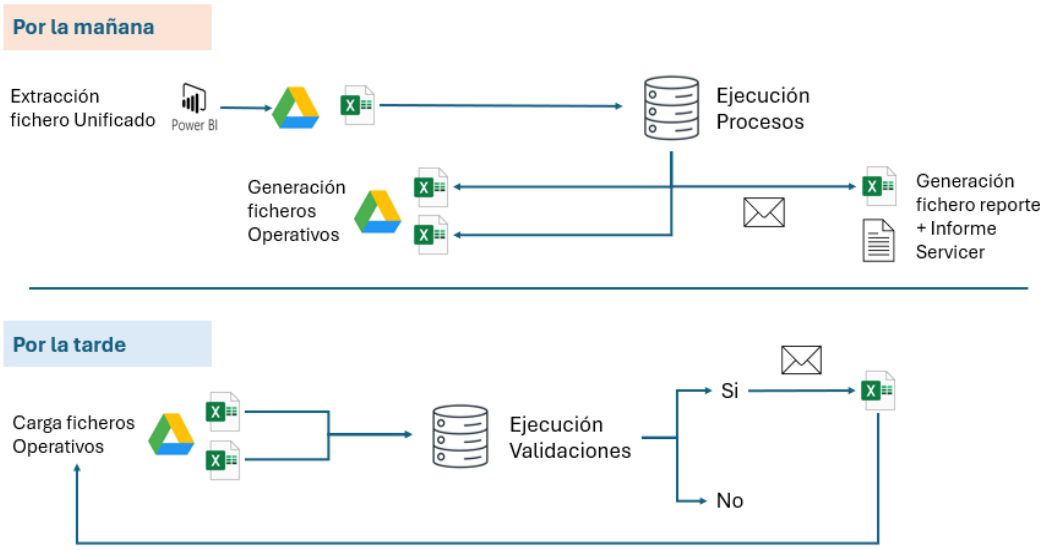

Ilustración 1: Flujo de trabajo optimizado. Fuente: elaboración propia.

Esta imagen muestra el trabajo realizado. Por la mañana se carga el fichero de ventas del Servicer en la BBDD y se generan tanto los ficheros operativos para que los gestores de Tributos y CCPP trabajen ese día, como el fichero de reporte al Servicer con los campos de gestión actualizados según los ficheros operativos del día anterior.

Por la tarde se cargan los ficheros operativos en la BBDD y se ejecutan las validaciones, en el caso de que haya errores, se devolverá por correo electrónico al Coordinador en cuestión para que lo corrija y se vuelva a cargar.

4. Conclusiones

Este proyecto ha permitido a EG operar de manera mucho más sólida, eficiente y profesional, lo que no solo mejora el servicio ofrecido, sino que también abre la puerta para atraer nuevos clientes. El tiempo ahorrado en la gestión de datos y la reducción de errores ha optimizado el uso de los recursos, beneficiando tanto a la empresa como al Servicer.

Aunque la implementación ha comenzado con el área de ventas, el objetivo es integrar todos los diferentes datos manejados en los distintos servicios de EG en la BBDD. Esto optimizará el trabajo diario y permitirá implementar lógicas más complejas y personalizadas, sentando las bases para una mejora continua.

EG ofrece muchos más servicios como Gestión Catastral, Requerimientos, Formalización, y Facturación. A largo plazo, se pretende incluir toda esta información en la BBDD para optimizar los recursos y llevar un seguimiento preciso y eficiente.

Se busca ampliar la red de clientes presentando estos proyectos con el sistema implementado, y asegurando que los gestores trabajen con un solo fichero con toda la

información. El objetivo es facilitar el trabajo y hacer los servicios más eficientes, posicionando a EG como una empresa innovadora y confiable en la gestión de activos inmobiliarios.

Integral automation of administrative processes: Optimization of data management and services.

Autor: Guzmán Torres, Ángel.
Director: Ignacio Martínez Vignardi.
Entidad Colaboradora: Nfq Advasory.

ABSTRACT

This project arises from the need to automate and optimize the data flow of a management company that offers operational services. The objective is to improve data accuracy and reduce errors (process automation) of a specific service offered to a Real Estate Servicer. The initial situation is analyzed and a new workflow is proposed by creating a centralized database.

Keywords: Database, process automation, management company, real estate servicer, operations service.

1. Introducción.

This project aims to improve the services of a Management Company (ME) for a Real Estate Servicer (Servicer) by optimizing administrative and operational processes. As a consultant and data engineer, a comprehensive study will be performed to identify and solve performance issues. From this analysis, a new workflow will be designed to minimize manual work, the main cause of errors and delays. Technological tools will be implemented to automate processes, improving the accuracy and quality of information. This will ensure a more efficient and transparent management, strengthening the collaboration between EG and the Servicer.

2. Definición del proyecto

The main objective of the project is to automate the workflow and reduce manual tasks as much as possible. To achieve this main objective, the following secondary objectives have been defined:

- Understand the initial workflow.
- Identify problems, errors and improvement points.
- Reduction of manual work.
- Improved data accuracy.
- Transparency and traceability.
- Servicer satisfaction.

The first stage of the project consists of working closely with the different departments of EG that are involved in the service offered to the Servicer. This phase lasts two weeks and its objective is to thoroughly understand the existing workflow.

After the initial analysis, a detailed strategy and improvement plan will be developed. In this phase, a SQL Server database (hereafter referred to as DB) will be built from scratch, which will be the main tool of the proposed new workflow. The structure of the database will be designed to offer several long-term benefits, such as easy maintenance and easy problem detection, in order to load and extract data into the database, several Python scripts will be developed using libraries such as Pandas, Selenium and Openpyxl, among others.

These scripts will facilitate data manipulation and automation of repetitive tasks. All the files involved in this project will be organized in different folders in Google Drive, where the coordinators of the different departments of EG will have access to pick up and drop off files of interest, and stored procedures will be developed within the DB to ensure an accurate and efficient data flow. These procedures will include specific logics to create validations and alerts, as well as provide valuable information to EG coordinators and managers.

Once code development is complete, manual testing will be performed to correct minor details and ensure the system is functioning properly.

3. Contexto y Resultados

The project focuses on the management of debt payments and obtaining zero debt certifications for real estate assets that Servicer Inmobiliario has in its portfolio and are for sale. EG works this administrative and management part, so that on the day of the sale at the notary's office the asset is free of debt and the sale is successfully signed.

EG verifies that all payments are up to date with the Community of Property Owners (CCPP) and Taxes. However, the current system has significant deficiencies, such as frequent errors, inconsistent data and loss of information, causing dissatisfaction in the Servicer. In addition, the manual management of these processes consumes a large amount of resources and time.

Involved Parties:

Servicer: Manages a large number of real estate asset portfolios, managing an average of 900 sales per month and requiring EG to manage at least 95% of these sales. This requires constant daily coordination to ensure efficiency and accuracy. The Servicer needs detailed and up-to-date reports to have a clear view of the daily operations.

Management Company (EG): Responsible for managing the Taxes and Community of Owners payments of the sold properties, the organizational structure includes coordinators and managers:

- Coordinators: Two coordinators, one for Taxes and one for CCPP, assign sales to the managers, maintain daily contact with the Servicer, correct errors, perform data analysis, follow up on invoicing, ensure the efficiency of the management, and lead other services offered to different Servicers.

- Managers: Divided into Tax and CCPP teams, the managers work directly with public administrations and property managers to manage payments and ensure that there are no outstanding debts on the properties.

The actions carried out by the Managers are recorded in an Excel file received in the morning with the sales to be processed, and at the end of the day the progress is reported to the Servicer.

It should be noted that the number of sales varies daily, so it is necessary for both parties to update the sales file: The Servicer updates the sales present in the file and EG updates the management fields for each sale.

The automation of tasks, especially data loading and validation, has significantly reduced the time required to complete the activities of the Coordinators. Previously, manual file uploading and error checking consumed approximately 20% of the coordinators' time (1.5 hours per day). With the implementation of Python scripting and the use of SQL Server stored procedures, this time has been reduced to 1% (5 minutes per day). The only manual task is to correct errors in the files late in the afternoon (automatic validations), which allows the Coordinators to focus on more value-added activities and spend more time managing other services, not just sales.

The detailed recording of all transactions in the database has improved transparency and traceability of operations. It is now possible to perform internal audits more efficiently, quickly identifying any discrepancies and correcting them in real time. This has also strengthened the trust between EG and the Servicer, since it is now possible to justify the transactions carried out with accurate and updated data, as well as to justify those not handled due to time constraints.

In addition, Coordinators can perform analysis tasks in a much easier way thanks to the access to PowerBI, where they can use all the graphs and tables based on the data present in the DB. These tables are created in the DB's reporting stored procedure, providing valuable information in a visual and accessible way.

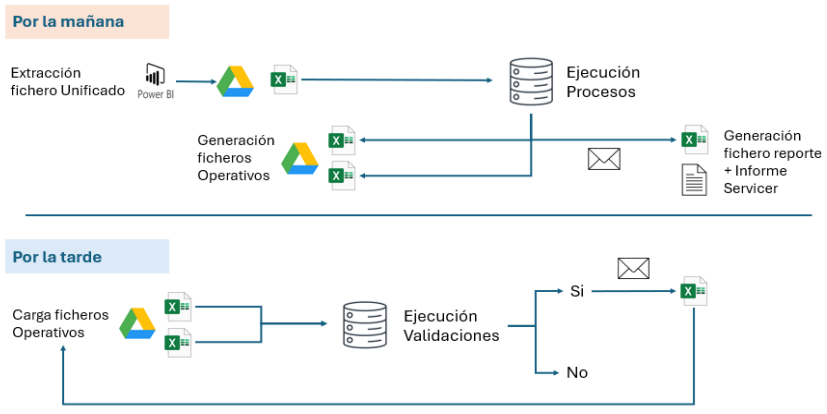

Ilustración 2: Flujo de trabajo optimizado. Fuente: elaboración propia.

This image shows the work done. In the morning, the Servicer sales file is loaded into the database and the operational files are generated for the Tax and CCPP managers to work that day, as well as the report file to the Servicer with the updated management fields according to the previous day's operational files.

In the afternoon, the operational files are uploaded to the database and the validations are carried out. In the event of errors, they will be returned by e-mail to the Coordinator in question for correction and reloading.

4. Conclusiones

This project has allowed EG to operate in a much more robust, efficient and professional manner, which not only improves the service offered, but also opens the door to attract new clients. The time saved in data management and the reduction of errors has optimized the use of resources, benefiting both the company and the Servicer.

Although the implementation has started with the sales area, the goal is to integrate all the different data handled in EG's various services into the DB. This will optimize the daily work and allow the implementation of more complex and customized logics, laying the foundation for continuous improvement.

EG offers many more services such as Cadastral Management, Requirements, Formalization, and Invoicing. In the long term, we intend to include all this information in the DB in order to optimize resources and keep an accurate and efficient follow-up.

The aim is to expand the network of clients by presenting these projects with the implemented system, and ensuring that the managers work with a single file with all the information. The objective is to facilitate the work and make the services more efficient, positioning EG as an innovative and reliable company in the management of real estate assets.

Índice de la memoria

Contenido

Índice de figuras

Capítulo 1. INTRODUCCIÓN

1.1 MOTIVACIÓN DEL PROYECTO

El presente proyecto surge de la necesidad de implementar mejoras significativas en un servicio profesional que actualmente ofrece una Empresa Gestora, en adelante referida como EG, a un Servicer Inmobiliario, en adelante referido como Servicer. Este proyecto surge a raíz de la necesidad de optimizar y eficientar los procesos administrativos y operativos entre ambas entidades.

Como consultor e ingeniero de datos, se plantea llevar a cabo un exhaustivo y minucioso estudio de la funcionalidad actual del proyecto. Este estudio tendrá como objetivo principal identificar con precisión la raíz de los problemas que afectan el desempeño del servicio. A través de un análisis detallado, se busca comprender las debilidades y los puntos críticos que requieren intervención.

Con los resultados obtenidos, se procederá a definir y diseñar un nuevo flujo de trabajo. Este nuevo esquema de operación estará orientado a minimizar de manera drástica el trabajo manual, el cual es una de las principales fuentes de errores y retrasos. Además, se implementarán estrategias y herramientas tecnológicas que permitan automatizar los procesos, garantizando así una mayor precisión y reduciendo la posibilidad de errores humanos.

Este enfoque también contribuirá a mejorar la precisión y la calidad de la información gestionada. Al mismo tiempo, asegurará una gestión más eficiente y transparente tanto para EG como para el Servicer, lo cual es fundamental para fortalecer la confianza y la colaboración entre ambas partes.

1.2 CONTRIBUCIÓN A LOS ODS

El presente proyecto tiene una importante contribución con varios de los ODS, a continuación, se detalla una breve explicación de cómo es esa contribución:

Ilustración 3: Objetivos de Desarrollo Sostenible. Fuente: Google

ODS 8: Trabajo Decente y Crecimiento Económico

La automatización de tareas reduce la carga laboral repetitiva y mejora la eficiencia, permitiendo que los empleados se enfoquen en actividades de mayor valor añadido. Esto no solo aumenta la productividad, sino que también mejora las condiciones laborales y la satisfacción de los trabajadores.

ODS 9: Industria, Innovación e Infraestructura

La implementación de nuevas tecnologías y la creación de una base de datos robusta fortalece la infraestructura digital de EG. Se mejora la precisión y confiabilidad de los datos gracias a la innovación aplicada en el proyecto, con esto se consigue que el trabajo realizado sea mucho más eficiente.

ODS 12: Producción y Consumo Responsables

Al minimizar el trabajo manual y automatizar los procesos, se reducen significativamente los errores. Esto lleva a una gestión más responsable de la información.

ODS 16: Paz, Justicia e Instituciones Sólidas

Este proyecto fortalece la relación entre EG y el Servicer, a través de la mejora en la precisión de la información se aseguran procesos más transparentes y confiables.

1.3 OBJETIVOS DEL PROYECTO

El objetivo principal del proyecto es automatizar el flujo de trabajo y reducir al máximo las tareas manuales, con el fin de que EG pueda ofrecer un servicio más eficiente y de alta calidad al Servicer. Para lograr este objetivo principal, se han definido los siguientes objetivos secundarios:

- Comprender el flujo de trabajo inicial: Realizar un análisis detallado del flujo de trabajo actual para entender cada uno de los pasos involucrados, desde la recepción de información hasta la entrega de los resultados finales al Servicer.
- Identificar problemas, errores y puntos de mejora: Llevar a cabo una evaluación exhaustiva para detectar los problemas y errores recurrentes, así como identificar las áreas que necesitan mejora.
- Reducción del trabajo manual: Implementar automatizaciones que minimicen las tareas manuales, con el objetivo de disminuir la carga de trabajo y reducir los errores humanos.
- Mejora de precisión en los datos: Asegurar que los datos manejados en el proceso sean precisos y estén actualizados. Se implementarán para ello validaciones automatizadas.
- Transparencia y trazabilidad: Establecer un sistema que permita una gestión transparente de todas las actividades y transacciones.

- Satisfacción del Servicer: Asegurar que las mejoras implementadas resulten en un servicio que cumpla con las expectativas y necesidades del Servicer, incrementando así su satisfacción y confianza en EG.

1.4 METODOLOGÍA UTILIZADA

La metodología utilizada en el proyecto se basa en los objetivos mencionados anteriormente, asegurando un enfoque sistemático y estructurado para lograr la automatización y optimización del flujo de trabajo. A continuación, se detalla el proyecto en sus diferentes etapas:

1.4.1 PRIMERA ETAPA: ANÁLISIS Y COMPRENSIÓN DEL FLUJO DE TRABAJO

La primera etapa del proyecto consiste en trabajar estrechamente con los diferentes departamentos de EG que están involucrados en el servicio ofrecido al Servicer. Esta fase tiene una duración de dos semanas y su objetivo es comprender a fondo el flujo de trabajo existente. Durante este período, se llevarán a cabo reuniones con los coordinadores de cada departamento para identificar todas las casuísticas y particularidades del proceso actual. Esto permitirá tener una visión detallada de las operaciones, detectando puntos críticos y posibles áreas de mejora.

1.4.2 SEGUNDA ETAPA: ESTRATEGIA Y PLAN DE MEJORA

Tras el análisis inicial, se elaborará una estrategia y un plan de mejora detallado. En esta fase se construirá desde cero una base de datos en SQL Server (en adelante, BBDD), que será la herramienta principal del nuevo flujo de trabajo propuesto. La estructura de la BBDD se diseñará para ofrecer diversos beneficios a largo plazo, tales como un mantenimiento sencillo y una fácil detección de problemas. La creación de esta base de datos sentará las bases para la automatización de procesos.

1.4.3 TERCERA ETAPA: DESARROLLO DE SCRIPTS Y PROCEDIMIENTOS

Para poder cargar y extraer ficheros en la BBDD, se desarrollarán varios scripts en Python utilizando librerías como Pandas, Selenium y Openpyxl, entre otras. Estos scripts facilitarán la manipulación de datos y la automatización de tareas repetitivas. Todos los archivos involucrados en este proyecto se organizarán en diferentes carpetas en Google Drive, donde los coordinadores de los distintos departamentos de EG tendrán acceso para poder recoger y dejar los ficheros de interés.

Además, se desarrollarán procedimientos almacenados dentro de la BBDD para asegurar un flujo de datos preciso y eficiente. Estos procedimientos incluirán lógicas específicas para crear validaciones y alertas, así como proporcionar información valiosa a los coordinadores y gestores de EG. El desarrollo del código y los scripts tendrá una duración de seis semanas.

1.4.4 CUARTA ETAPA: PRUEBAS Y RETROALIMENTACIÓN

Una vez completado el desarrollo del código, se realizarán pruebas manuales para corregir pequeños detalles y asegurar el correcto funcionamiento del sistema. Durante esta fase, se recibirá retroalimentación de los coordinadores, quienes habrán recibido previamente formación sobre el nuevo flujo de trabajo, para asegurar que entienden las lógicas aplicadas. Este feedback será crucial para ajustar y perfeccionar el sistema, garantizando que cumpla con las expectativas y necesidades de todos los usuarios.

1.4.5 QUINTA ETAPA: AUTOMATIZACIÓN Y MEJORAS CONTINUAS

Con el sistema funcionando a la perfección, se desarrollará un schedule en Python para automatizar todo el proceso, eliminando la necesidad de intervención manual. Esta automatización completa permitirá una operación más eficiente y fiable del sistema. A partir de este punto, se implementarán mejoras continuas para alcanzar un servicio profesional y completo, asegurando que EG pueda ofrecer un servicio altamente eficiente y preciso al Servicer.

Capítulo 2. DESCRIPCIÓN DEL PROYECTO

2.1 ANTECEDENTES Y CONTEXTO

El presente proyecto surge del siguiente contexto: EG es una empresa que ofrece servicios de ejecución de procesos integrales, aportando a sus clientes valor tangible dentro del sector financiero, inmobiliario y de seguros.

En este caso, ofrece al Servicer un servicio integral de gestión de pagos y certificaciones relacionados con la venta de activos inmobiliarios. Este servicio incluye la verificación de que todos los pagos estén al corriente y EG es el encargado de gestionar y tramitar con quien corresponda para conseguir los justificantes de pago o certificados de deuda cero de los inmuebles, tanto en el ámbito de la Comunidad de Propietarios (CCPP) como en el de Tributos.

El sistema actual utilizado por EG presenta diversas deficiencias, tales como errores frecuentes, datos incoherentes y pérdida de información, lo que ha resultado en insatisfacción por parte del Servicer. Además, la gestión manual de estos procesos consume una cantidad significativa de recursos, tiempo y, por ende, de dinero, lo que ha llevado a la necesidad de una revisión y mejora del sistema.

El consultor (autor de este trabajo) se involucra en el proyecto con el objetivo de automatizar el mayor número posible de procedimientos, reduciendo así la carga laboral y garantizando un registro fiable de todas las gestiones realizadas. La automatización de estos procesos permitirá no solo mejorar la eficiencia operativa de EG, sino también proporcionar un servicio más transparente y preciso al Servicer.

En este contexto, el proyecto se centra en la creación de una nueva base de datos (BBDD) y el desarrollo de scripts y procedimientos automatizados que optimicen el flujo de trabajo y aseguren la integridad y consistencia de los datos.

2.2 PARTES INVOLUCRADAS

2.2.1 SERVICER

El Servicer es una entidad que administra una gran cantidad de carteras de activos inmobiliarios, gestionando un promedio de 900 ventas mensuales. La necesidad del proyecto surge de la exigencia del Servicer de que EG gestione con éxito al menos el 95% de estas ventas. Este alto volumen de operaciones demanda una coordinación diaria constante entre el Servicer y los coordinadores de EG para garantizar la eficiencia y precisión en la gestión de ventas.

Cada día, el Servicer da de alta nuevas ventas mientras otras se cancelan, lo cual requiere una actualización constante y precisa de la información. Esto implica la necesidad de un análisis diario detallado de las gestiones realizadas. Dado el volumen y la naturaleza dinámica de las operaciones, es crucial que haya una comunicación fluida y continua entre el Servicer y los coordinadores de EG. Esta comunicación garantiza que todas las gestiones se realicen correctamente y dentro de los plazos establecidos, minimizando errores y retrasos que podrían afectar negativamente el rendimiento del servicio.

La gestión efectiva de estos activos no solo requiere habilidades operativas, sino también la capacidad de manejar datos de manera precisa y en tiempo real. Por ello, el proyecto se enfoca en mejorar los sistemas y procesos de gestión de EG para que puedan manejar el alto volumen de ventas con la eficiencia requerida por el Servicer. Esta mejora no solo busca cumplir con las expectativas del Servicer en términos de volumen y precisión, sino también fortalecer la relación y colaboración entre ambas entidades, asegurando que el servicio proporcionado sea de la más alta calidad y confiabilidad.

Además, el Servicer necesita reportes detallados y actualizados que le permitan tener una visión clara y precisa de las operaciones diarias. Esto incluye la capacidad de identificar rápidamente cualquier problema o retraso en las gestiones para tomar acciones correctivas inmediatas. La capacidad de EG para adaptarse a estos requisitos y proporcionar información precisa y actualizada es esencial para garantizar el éxito continuo del proyecto.

2.2.2 EMPRESA GESTORA (EG)

EG es una empresa que ofrece servicios de operaciones y gestión a múltiples empresas, centrándonos en este proyecto, es responsable de gestionar los pagos de los Tributos y la Comunidad de Propietarios de los inmuebles que se venden en la cartera gestionada por el Servicer, para así obtener las certificaciones o justificantes necesarios a presentar en notaria el día de la venta del activo.

La estructura de las personas en EG está organizada de la siguiente manera:

Coordinadores:

EG cuenta con dos Coordinadores, uno encargado del departamento de Tributos y otro del departamento de Comunidad de Propietarios (CCPP). Sus funciones son:

- Asignar ventas a los gestores.
- Mantener contacto diario con el Servicer.
- Realizar análisis de datos y asegurar que las gestiones se lleven a cabo eficientemente.
- Seguimiento de la facturación: cada oportunidad de venta se factura en base a las gestiones realizadas, cuando la gestión pasa a estar OK se factura (en el momento que se pagan las cartas de pago o, en su defecto, se obtenga el certificado de deuda cero).
- Liderar otros servicios ofrecidos a diferentes Servicers, como pagos de voluntaria, formalización, requerimientos, etc.

Gestores:

Dentro de cada departamento, existen una serie de gestores.

- Equipo de Tributos: Compuesto por 5 gestores. Este equipo trabaja directamente con administraciones públicas para gestionar pagos de tasas como agua, basura y alcantarillado, así como el Impuesto de Bienes Inmuebles (IBI). Los gestores presentan la documentación necesaria en las sedes electrónicas de las

administraciones públicas para obtener cartas de pago o justificantes de pago y mantienen un contacto continuo con los ayuntamientos y diputaciones de España.

- Equipo de CCPP: Compuesto por 4 gestores. Este equipo trabaja con administradores de fincas, verificando que los pagos de la comunidad de propietarios estén al día. Los gestores revisan y confirman que no existan deudas pendientes en las comunidades.

Si hubiese que realizar pagos en algún caso, son los gestores los encargados de realizar esta función, que será diferente en base a la Sociedad Propietaria del activo. Dentro de los activos bajo gestión, los gestores se distribuyen en dos grupos según la Sociedad Propietaria: Banco o Third Parties (en adelante, TP), esto se hace para que los gestores estén especializados en un tipo de gestión y sean más eficientes, ya que cada empresa (cartera) trabaja de una forma diferente.

El objetivo del trabajo de los gestores de EG es asegurar que, para el día en que se realicen las ventas de cada activo - fecha posicionamiento - , todos los pagos estén al corriente y no haya problemas durante la firma en notaría. Esto implica un esfuerzo constante y coordinado para mantener la información actualizada, tramitar pagos según corresponda con la sociedad propietaria del activo, y resolver cualquier incidencia que pueda surgir en el proceso. Los responsables del proyecto son los propios Coordinadores.

Capítulo 3. ANÁLISIS INICIAL

3.1 JUSTIFICACIÓN Y SITUACIÓN INICIAL

El proceso inicial de gestión llevado a cabo por EG en colaboración con el Servicer es un procedimiento complejo y manual. Este proceso es diario y comienza a las 8:00 am, cuando el Servicer envía un correo en el que se adjunta un fichero Excel (fichero de ventas) junto con un resumen diario (informe) a los Coordinadores de EG.

Los Coordinadores reciben dicho fichero y lo dividen manualmente en dos subficheros:

- Fichero Operativo CCPP: contiene los campos necesarios para tramitar las gestiones relacionadas con la Comunidad de Propietarios.
- Fichero Operativo Tributos: contiene los campos necesarios para tramitar las gestiones relacionadas con los Tributos (tasas e IBI).

Esta división facilita el trabajo de los gestores, asegurando que cada grupo maneje únicamente la información relevante para su área de responsabilidad.

El fichero original del Servicer, llamado Fichero Unificado o Fichero de Ventas, contiene una fila por cada oportunidad de venta y presenta tres tipos de campos distintos:

- Campos de información del activo: Detalles específicos del activo inmobiliario, como el nombre del comprador, la fecha de venta, la localización, y la Sociedad Propietaria del activo, la referencia de dicho activo, entre otras cosas.
- Campos de gestión de Tributos: Información y estados relacionados con los pagos de tasas y el Impuesto sobre Bienes Inmuebles (IBI), incluyendo fechas de inicio, última gestión y fin de gestión, estado de la gestión, y ciertos valores necesarios para la llevar a cabo la gestión de la venta con éxito, como nombres de remesa y fechas de envío a pago.

- Campos de gestión de CCPP: Datos y estados sobre los pagos de la Comunidad de Propietarios, similares a los campos de Tributos, pero para una gestión diferente.

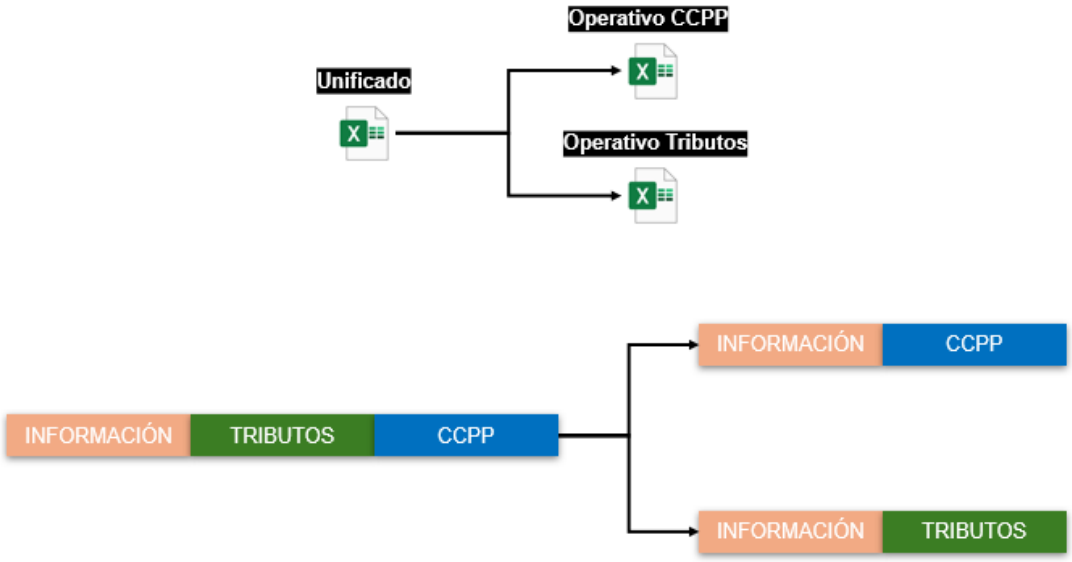

Ilustración 4: División de los ficheros con sus respectivos tipos de campos. Fuente: elaboración propia

Para evitar confusiones entre los gestores, EG divide estos campos en dos subficheros. Uno contiene los campos de información del activo y los campos de gestión de CCPP, y el otro los campos de información del activo y los campos de gestión de Tributos. Esta separación se realiza manualmente. Una vez completadas las gestiones diarias, los subficheros se vuelven a unificar en uno solo para devolverlo al Servicer.

Los Gestores actualizan los ficheros operativos con el progreso de las gestiones y los Coordinadores validan la información antes de enviarla de vuelta.

A final de mes, EG presenta la facturación al Servicer basada en las gestiones completadas. La facturación se realiza de forma individualizada por cada gestión, lo que requiere que los Coordinadores revisen diariamente qué gestiones pueden ser facturadas. Esta revisión diaria implica comprobar la fecha de la última gestión y confirmar si se ha proporcionado el justificante de pago o si las deudas se han pagado.

Además de los ficheros operativos, EG trabaja con dos ficheros adicionales por cada departamento (cuatro extras en total):

- Fichero de Ventas Especiales: Son ventas que el Servicer no puede incluir en su fichero de ventas actualizado diariamente debido a que son ventas de activos que pertenecen a carteras gestionadas aparte. Por ello, el Servicer envía un correo a los Coordinadores para que incluyan la venta en sus respectivos ficheros de ventas especiales.
- Fichero de Postventas Fuera de Fichero: Son ventas que, a pesar de darse de baja del fichero de ventas (unificado) porque ya se han vendido, la gestión aún no ha finalizado. El Coordinador debe estar pendiente de estas ventas porque, si el activo ya se ha vendido, habrá que terminar la gestión. Estas ventas son prioritarias.

La razón de usar ficheros diferentes es para mantener la consistencia en el número de ventas reportadas al Servicer. Las Ventas Especiales y Postventas se gestionan aparte para asegurar que el fichero de ventas contenga el mismo número de ventas que los ficheros operativos, ya que luego habrá que reportarlo al Servicer.

Responsabilidades de los Coordinadores de EG:

Como se ha comentado brevemente en el anterior apartado, los coordinadores de EG en este proyecto concreto tienen las siguientes responsabilidades:

- Asignación de ventas: Determinar qué nuevas ventas deben ser gestionadas y asignarlas a los gestores adecuados.
- Validación y facturación: Revisar y validar las gestiones completadas para asegurar que cumplen con los criterios de facturación.
- Actualización y división de ficheros: Dividir manualmente el fichero recibido del Servicer en subficheros y unificarlos nuevamente para su envío tras las gestiones diarias.
- Contacto diario con el Servicer: Mantener una comunicación constante para resolver incidencias y corregir errores.

- Gestionar con éxito el 95% de las oportunidades de ventas presentes en el fichero unificado que tengan la fecha posicionamiento en el mes actual (que se venden en el mes en curso). Este número de oportunidades de venta es variable ya que cada día hay altas y bajas.
- Analizar datos para repartir a los gestores entre diferentes proyectos o funciones específicas, ¿sobra gente en el equipo?
- Liderar otros servicios: Además de las tareas específicas del servicio de ventas, los Coordinadores lideran otros servicios ofrecidos a diferentes Servicers, añadiendo complejidad a su trabajo diario.

Responsabilidades del Servicer:

El Servicer tiene ciertas responsabilidades y uno de los objetivos del proyecto es también reducir el trabajo manual y las tareas repetitivas del Servicer. Sus responsabilidades en el contexto del proyecto en cuestión son las siguientes:

- Actualización de altas: Mantener actualizado el fichero con las nuevas altas de ventas de activos y sus campos de información.
- Eliminación de bajas: Quitar del fichero las ventas que se dan de baja por diversas razones.
- Envío diario del fichero de ventas: Enviar diariamente el fichero actualizado a EG.
- Informe de consecución: Proporcionar un informe diario que muestre el progreso de las gestiones en cada departamento (Tributos y CCPP), recordando que el objetivo es llegar al 95% de éxito y reportando las prioridades y las gestiones en la que hay que poner foco.

3.2 EVALUACIÓN DE PROCESOS

Tras trabajar con los diferentes departamentos y evaluar el estado del proyecto se determina lo siguiente.

Los datos, por lo general, no son muy precisos. Al haber trabajo manual excesivo y no tener una estrategia automatizada de validación, no se tiene un claro registro de las ventas que se dan de baja, se presentan los datos al Servicer de manera incompleta por parte de los Gestores y esto provoca que el Servicer pueda quedar descontento con el servicio.

La facturación mensual presenta problemas recurrentes, con procesos que pueden extenderse más de tres días debido a la falta de un sistema automatizado integrado. Es muy difícil llevar un registro propio de las gestiones que se van finalizando día a día para una posterior facturación, se trabaja con muchos datos y hacerlo de manera manual es muy propenso a cometer errores.

Los Coordinadores se enfrentan a una carga de trabajo manual considerable. La falta de automatización ha llevado a la pérdida ocasional de información y a la duplicación de esfuerzos. A raíz de esto se ha detectado el siguiente problema: cuando una venta se da de baja y luego se vuelve a dar de alta, la información de las gestiones previas puede perderse, obligando a los gestores a repetir el trabajo. Es crucial tener un registro claro de los activos que se dan de baja, ya que algunos de ellos pueden volver a estar darse de alta en un futuro. Las gestiones realizadas antes de la baja deben ser adecuadamente documentadas y, en caso de que se reactive la venta de ese activo, las gestiones previas deben ser consideradas para la facturación.

La falta de medición de los tiempos medios de gestión y el intervalo entre la fecha de alta de un activo y su fecha de venta han dificultado el cumplimiento de los objetivos por parte de EG. Cabe mencionar que se han detectado situaciones en las que el Servicer deja poco tiempo de gestión y esto afecta al rendimiento de EG. El Servicer es el que está en contacto con los compradores de los inmuebles y el que asigna las fechas de venta, fecha para la cual la gestión debe quedar finalizada por parte de EG para que la venta se realice con éxito. Los cambios en las fechas de venta realizados por el Servicer sobre los activos ya presentes en el fichero no son capturados adecuadamente por parte de EG, lo que puede llevar a confusiones y afectar la planificación y la gestión de las ventas.

Existe una tendencia preocupante de acumulación de fechas de ventas hacia el final del mes por parte del Servicer, lo que crea picos de trabajo y presión sobre los recursos de EG. Es crucial implementar medidas preventivas para distribuir mejor la carga de trabajo a lo largo del mes y mitigar estos efectos.

El uso de tres ficheros diferentes por cada equipo hace que se dupliquen gestiones y que aumente el tiempo de la gestión de las ventas, además de que el fichero de postventas se rellena de forma manual, lo cual invita a que haya errores e inconsistencias, es decir, los Coordinadores deben llevar un registro de las postventas que se dan de baja en el fichero de ventas y no se han terminado de gestionar para incluirlas en el fichero de postventas.

3.3 NECESIDADES Y REQUISITOS DE AUTOMATIZACIÓN

Tras el análisis se destacan múltiples áreas donde la automatización puede mitigar las deficiencias operativas y mejorar significativamente la eficiencia del servicio ofrecido al Servicer.

Es fundamental implementar una base de datos centralizada donde se integre diariamente la información del fichero proporcionado por el Servicer. Esta base de datos servirá como núcleo para aplicar lógicas de negocio que faciliten el registro y seguimiento de la facturación día a día. Además, se deben establecer validaciones automáticas para asegurar que los datos ingresados por parte de los Gestores sean completos y coherentes, minimizando así errores y omisiones, se van a implementar validaciones de negocio, campos obligatorios, de tipos de datos y ciertas validaciones de seguridad para evitar problemas.

La BBDD también debe mantener un registro detallado de las fechas de venta de cada activo, así como de los estados de gestión por parte de CCPP y Tributos, actualizándose continuamente para permitir un análisis preciso. Esto facilitará la justificación al Servicer sobre las gestiones no realizadas debido a cambios en las fechas de venta, proporcionando transparencia y claridad en la consecución de objetivos.

Para facilitar la supervisión y análisis por parte de los Coordinadores, se debe registrar detalladamente el trabajo realizado por cada gestor. Esto permitirá evaluar fácilmente el progreso individual y global del equipo, identificar áreas de mejora y reconocer el desempeño destacado. Para ello se implementan una serie de lógicas necesarias para mostrar posteriormente los informes de interés.

Se propone automatizar completamente el proceso de recepción del fichero del Servicer. Esto implica integrar sistemas que accedan de manera automatizada al fichero de ventas en el entorno del Servicer, descargarlo y cargarlo directamente en la BBDD de EG. Además, el sistema deberá generar informes detallados sobre la consecución de las ventas que hoy en día genera el Servicer, por lo que el proyecto también busca la automatización de las tareas del cliente para mejorar la experiencia.

Se establecerá un sistema para la generación automática de informes y alertas diarios internos para los coordinadores de EG. Estos informes proporcionarán una visión clara de la facturación día a día y alertas sobre situaciones anómalas, como activos que en un pasado se dieron de baja y vuelven a aparecer en el fichero de ventas como alta, para que puedan revisar si se tiene que realizar la gestión de nuevo o no, así como un aviso de los activos a los que el Servicer cambia la fecha de venta a menos de una semana según la fecha en curso, ya que no daría tiempo físico a realizar la gestión. Además, se automatizará el envío del fichero actualizado al Servicer por correo electrónico día a día, quitándole esa tarea a los Coordinadores.

La organización de los ficheros de carga y extracción en la BBDD se propone realizarla en carpetas accesibles mediante un Google Drive compartido, simplificando así el proceso para los Coordinadores. Su única tarea será recoger el fichero de gestión según el departamento generado por la BBDD por la mañana y devolverlo al final del día, después de que los Gestores trabajen, para su procesamiento en la BBDD y posterior envío del fichero de ventas actualizado al Servicer.

Capítulo 4. DISEÑO DE PROYECTO

4.1 DISEÑO DEL FLUJO DE TRABAJO

4.1.1 MAPEO DE PROCESOS ACTUALES

El proceso actual de gestión en EG se inicia con la recepción diaria del Fichero de Ventas y el Informe de Consecución por parte del Servicer, enviados vía correo electrónico. Una vez recibidos, los Coordinadores de EG revisan minuciosamente el informe para priorizar las actividades del día. Posteriormente, dividen el Fichero Unificado (mismo Fichero de Ventas enviado por el Servicer) en dos subficheros separados: uno destinado a la gestión de Comunidades de Propietarios (CCPP) y otro para la gestión de Tributos.

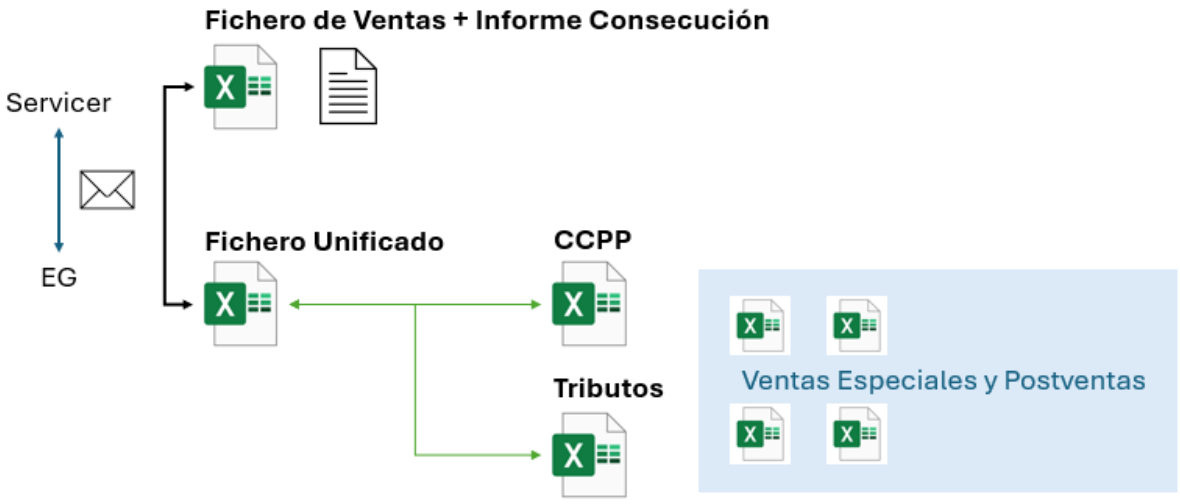

Ilustración 5: Flujo de trabajo inicial. Fuente: elaboración propia

Cada departamento dentro de EG trabaja de manera paralela con estos subficheros, junto con los ficheros adicionales de Ventas Especiales y Postventas. Durante el día, los gestores correspondientes gestionan las ventas y actualizan los campos de gestión del fichero según el departamento, hay ciertos gestores encargados de gestionar las ventas especiales y postventas (1 gestor por departamento).

Al concluir la jornada laboral, los Coordinadores consolidan nuevamente el Fichero Unificado y lo devuelven al Servicer. Este último procede a actualizar su propio sistema con la información de gestión proporcionada por EG. Este proceso manual implica un alto grado de intervención humana y conlleva desafíos significativos, como la posibilidad de errores en la consolidación de datos y la falta de actualización inmediata de los registros.

Además, la gestión de Ventas Especiales y Postventas se realiza de manera independiente y manual, incrementando la complejidad operativa y abriendo la puerta a errores y retrasos en la facturación y en la gestión de informes. Esta estructura actual del flujo de trabajo subraya la urgente necesidad de automatización y optimización para mejorar la precisión, eficiencia y transparencia en todas las etapas del proceso de gestión de activos inmobiliarios.

Las validaciones se realizan de manera manual identificando las gestiones realizadas en el día a través de la fecha de última gestión, pero ¿quién asegura que el Gestor esté actualizando esa fecha? El hecho de llevar un seguimiento manual de la facturación tiene el mismo problema, ¿cómo tienen la certeza de qué día la venta pasó a estar gestionada y por lo tanto se factura?

4.1.2 PROPUESTA DE NUEVO FLUJO DE TRABAJO

Tras un estudio minucioso del flujo de trabajo y, pensando en cómo poder diseñar un nuevo flujo que abarque todas las necesidades, se establece lo siguiente:

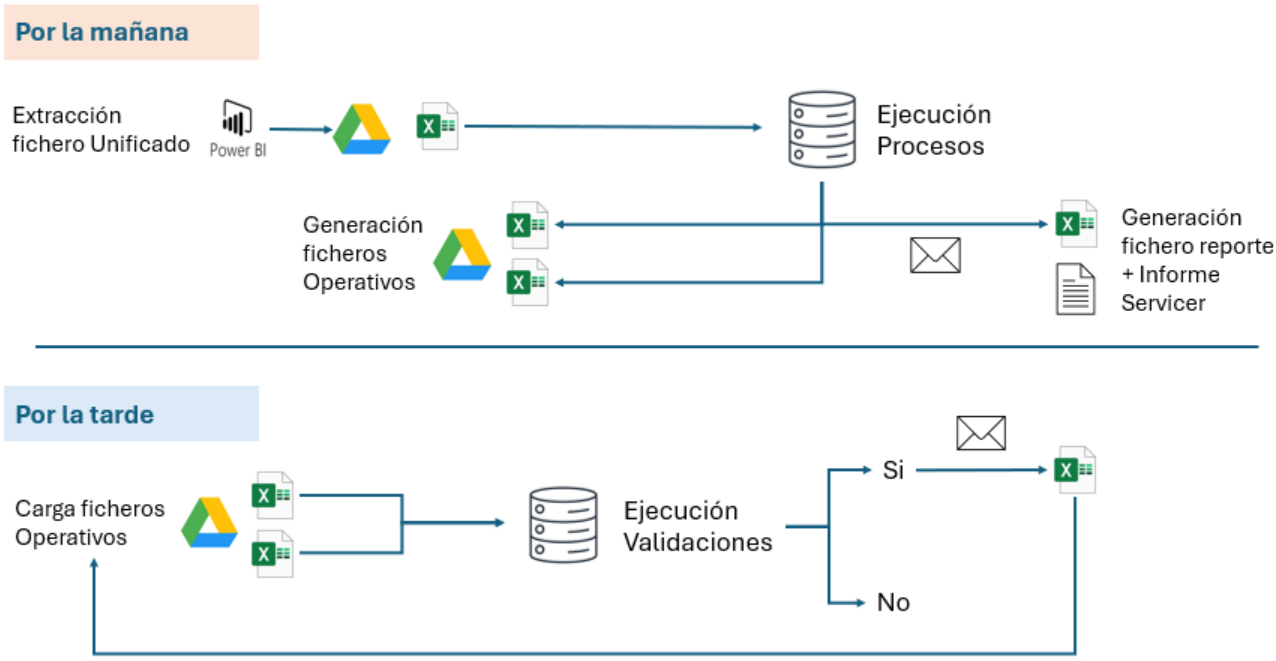

Ilustración 6: Nuevo flujo de trabajo. Fuente: elaboración propia

El nuevo flujo de trabajo se divide en dos etapas:

4.1.2.1Mañana

El Fichero de Ventas por parte del Servicer se actualiza a las 7:30 am indicando a diario las nuevas ventas dadas de alta y las ventas que se dan de baja, este fichero se puede extraer a Excel accediendo a la interfaz propia de Power BI del Servicer, la idea es extraer ese fichero a la ruta correspondiente del Google Drive para llevar a cabo su posterior carga en la BBDD, una vez cargado, se ejecutaran los diferentes procedimientos almacenados para tratar el dato según el diseño establecido y poder extraer de la BBDD tres ficheros Excel:

<u>Ficheros Operativos</u>

Son los ficheros con los que los Gestores trabajarán día a día, por lo que hay uno para CCPP y otro para Tributos, cada uno muestra los campos de gestión correspondientes a su departamento (campos a actualizar por parte de los Gestores), así como la información del activo incluida por el Servicer y ciertos campos adicionales de interés.

En estos ficheros se incluyen los tres tipos de ventas diferentes:

- Ventas presentes en el fichero de ventas del Servicer, o lo que es lo mismo, Ventas Normales: estas son las ventas que posteriormente se tienen que reportar al Servicer, las que se incluyen en el fichero descargado por la mañana en Power BI (Fichero Unificado).
- Ventas Especiales: indicadas por la columna Venta Especial, rellenadas con un "si", estas son las ventas que pertenecen a activos con una Sociedad Propietaria a gestionar aparte, por eso se identifican de esta forma.
- Postventas Fuera de Fichero: indicadas por la columna Postventa, son las ventas que se dieron de baja en el Fichero de Ventas porque ya se vendieron, pero no se llegó a finalizar la gestión.

Gracias a esto, cada departamento reduce el uso a un único fichero, evitando que haya oportunidades de venta duplicadas y agilizando los procesos de gestión.

Fichero de Reporte

Al igual que se generan los Fichero Operativos, también se genera un Fichero de Reporte para el Servicer, en este caso, tan solo se incluyen las Ventas Normales y se actualizan los campos de gestión con la información de los Ficheros Operativos del día anterior, actualizando los campos de gestión para el Servicer. A través de la información proporcionada en este fichero, se realiza el Informe de Consecución que también se reporta al Servicer, así como a los Coordinadores de cada departamento.

Este Fichero de Reporte se hace llegar a los Coordinadores y al Servicer por correo electrónico automatizado, así como el Informe de Consecución.

4.1.2.2 Tarde

Después de trabajar durante todo el día actualizando los campos de gestión sobre los Ficheros Operativos, los Coordinadores son los encargados de subir el Fichero correspondiente a su departamento en la ruta específica del Google Drive para su posterior

carga en la BBDD. Una vez cargado, se ejecutan las validaciones a tener en cuenta y, en el caso de que exista alguna validación a corregir, el fichero operativo correspondiente se volverá a extraer de la BBDD indicando en una hoja complementaria las validaciones (errores) presentes, el Coordinador deberá corregir dichas validaciones y volver a dejar el fichero en la ruta para volver a cargarlo, este proceso se repetirá hasta que el Fichero Operativo no presente validaciones y pueda quedar cargado en la BBDD hasta el día siguiente por la mañana, donde se repetirá el mismo proceso.

En el caso de que existan validaciones, se le hace llegar un correo electrónico automatizado al Coordinador correspondiente para que lo corrija y lo vuelva a dejar en la ruta, en el caso de no existir validaciones, se enviará también un correo de confirmación, asegurando que el fichero ha quedado perfectamente cargado y validado.

4.2 ESTRUCTURA BASE DE DATOS

4.2.1 MODELADO DE LOS DATOS

La estructura de la BBDD implementada para este proyecto se organiza en cuatro fases distintas: STG, HOM, HIS y CALC. Cada fase tiene una función específica en el procesamiento y gestión de los datos, garantizando que se mantenga la integridad, precisión y coherencia a lo largo del flujo de trabajo. A continuación, se detalla cada una de estas fases:

Fase STG (Staging):

Esta es la fase inicial, es decir, los datos recién cargados desde un fichero (Excel) a la BBDD. Todos los datos se guardan como cadenas de texto para evitar problemas de carga relacionados con errores de formato. Durante esta fase, se rellena la columna Fec_fichero, que corresponde a la fecha indicada en el nombre del fichero Excel a cargar o, en caso de no indicarse, a la fecha de carga. Esta etapa sirve como una zona temporal para el preprocesamiento de los datos antes de su transformación.

Fase HOM (Homogeneización):

En la fase de homogeneización, los datos se transforman para imponer el tipo de dato correspondiente a cada campo. Esto asegura que los datos cumplan con los formatos y tipos esperados para su procesamiento posterior. Se rellenan dos columnas adicionales:

- Checksum: Un número generado a partir de los campos de cada registro, utilizado para detectar cambios en los datos con respecto al registro del día anterior.
- Concat_SQL: Un campo único en la tabla, creado a partir de la concatenación de los principales campos de un registro, que se utilizará para historificar cada registro de manera única.

Fase HIS (Historificación):

En la fase de historización, se alimenta un histórico que muestra todos los distintos registros según el fichero. Cada registro se clasifica según su columna Estado:

- Alta: El registro aparece en HOM, pero no se encontraba previamente en el histórico.
- Baja: El registro se encontraba en el histórico, pero ya no aparece en HOM.
- Modificación: Se ha modificado algún campo, detectando cambios entre el último registro del histórico y el registro en HOM a través de la columna Checksum.
- Realta: En el último registro del histórico, el registro figuraba como baja y ahora aparece nuevamente en HOM.

Se muestra un ejemplo de ciertas columnas de la tabla HIS perteneciente a Tributos de un activo aleatorio para entender cómo se rellena, ordenado por Fec_fichero de manera ascendente.

Results | Messages | Client Statistics

	FEC_FICHERO	CONCAT_SQL	ESTADO_TRIBUTOS	ESTADO
1	2024-06-19	BR850005 / 00107303	APORTADO_JUSTIFICANTE	BAJA
2	2024-06-14	BR850005 / 00107303	APORTADO_JUSTIFICANTE	MODIFICACION
3	2024-06-12	BR850005 / 00107303	APORTADO_JUSTIFICANTE	MODIFICACION
4	2024-06-06	BR850005 / 00107303	SOLICITADO_SEDE_ELECTRÓNICA	MODIFICACION
5	2024-05-30	BR850005 / 00107303	PTE_GESTION	MODIFICACION
6	2024-05-21	BR850005 / 00107303	PTE_GESTION	MODIFICACION
7	2024-05-14	BR850005 / 00107303	PTE_GESTION	MODIFICACION
8	2024-05-13	BR850005 / 00107303	PTE_GESTION	MODIFICACION
9	2024-05-07	BR850005 / 00107303	PTE_GESTION	ALTA

Ilustración 7: Ejemplo de tabla HIS para un activo concreto. Fuente: SQL Server

Se puede observar la fecha del fichero, como en la fila 9 se dio de alta y cada línea muestra las veces que ha habido algún cambio en alguno de sus campos (en la imagen no se muestran todos los campos), hasta que finalmente el activo se da de baja. En el caso de que ese activo volviese a darse de alta, se añadiría sobre el histórico una nueva fila indicando que es una Realta.

Fase CALC (Cálculo y Lógicas):

Esta fase se centra en la generación de tablas a partir de las lógicas implementadas en procedimientos almacenados. En esta fase se encuentran los Maestros de CCPP y Tributos y ciertas tablas de interés para realizar posteriores análisis o cálculos de interés.

Esquemas secundarios

- REP (Reporting): esquema utilizado para tablas que van a ser exportadas de la BBDD. Este tipo de tablas se rellenan en procedimientos almacenados a partir de las tablas CALC.
- COD (Paramétricas): esquema utilizado para tablas paramétricas, es decir, tablas que se relacionan con diferentes procedimientos almacenados para poder llevar ejecuciones a cabo.

- DQ (Validaciones): esquema utilizado para tablas que muestran los errores de validación presentes en las diferentes tablas que se han cargado. Estas tablas se rellenan ejecutando los procedimientos almacenados procedentes.

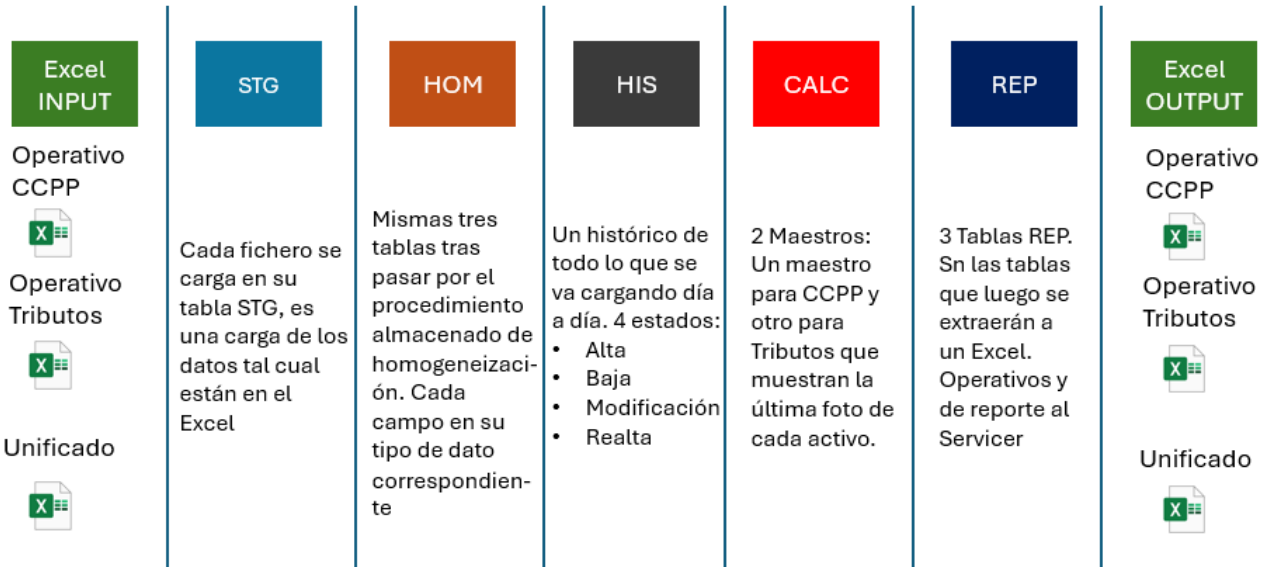

Ilustración 8: Resumen tratamiento del dato en SQL Server. Fuente: elaboración propia

A modo de resumen, se incluye una imagen para entender más fácilmente el siguiente apartado. Es el flujo que recorren los datos dentro de la BBDD y para pasar de un esquema a otro se ejecutan diferentes procedimientos almacenados estandarizados, programados en SQL Server. Esto asegura tener un buen registro de los datos y demás información de interés detallada posteriormente.

Se muestra tan solo el flujo de los datos de los ficheros que se cargan, pero existen muchas más tablas y procedimientos que más adelante se explicarán en detalle, esto sirve a modo de introducción.

4.2.2 ORGANIZACIÓN Y PARAMETRIZACIÓN DEL SERVICIO

Con el objetivo de profesionalizar este servicio y facilitar la implementación de futuros proyectos, se ha estructurado la base de datos (BBDD) en torno a tres tablas paramétricas principales. Estas tablas actúan como parámetros de entrada para la ejecución de todos los procedimientos estandarizados, asegurando una gestión eficiente y ordenada.

Tablas Paramétricas:

- COD.CLIENTES: Esta tabla relaciona id_cliente con nombre_cliente. En el contexto de este proyecto, el Servicer específico se identifica con id_cliente = 1.
- COD.HITOS: Esta tabla vincula el id_hito con el nombre_hito. En este proyecto se trabajan con tres hitos principales:
    - Tributos: Se refiere a las tablas que contienen los campos de gestión de tributos. Identificado como id_hito = 1.
    - CCPP: Se refiere a las tablas que contienen los campos de gestión de la Comunidad de Propietarios (CCPP). Identificado como id_hito = 2.
    - Tributos – CCPP: Se refiere a las tablas que incluyen tanto campos de gestión de tributos como de CCPP. Identificado como id_hito = 7.
- COD.SERVICIOS: Esta tabla relaciona el id_servicio con el nombre_servicio. En este caso, el servicio de ventas se identifica con id_servicio = 1.

Para este proyecto específico, al ser el primero implementado dentro del sistema, solo se utilizarán los parámetros mencionados anteriormente. Sin embargo, la estructura está diseñada para permitir la incorporación de nuevos proyectos en el futuro, simplemente rellenando las tablas paramétricas pertinentes.

Estos parámetros son utilizados como inputs en los diversos procedimientos y scripts. Por ejemplo, para transferir datos de STG.CCPP_OPE a HOM.CCPP_OPE, se ejecuta el procedimiento HOM.HOMOGENEIZACION con los parámetros 1, 2, 1, que hacen referencia a id_cliente, id_hito e id_servicio, respectivamente. Comprender este esquema es crucial para entender el funcionamiento de los diferentes scripts y procedimientos almacenados.

Las tablas paramétricas en la BBDD, de las que dependen los distintos procedimientos, también incluyen las columnas id_cliente, id_hito e id_servicio. Los procedimientos filtrarán los registros correspondientes en función de estos parámetros de entrada, garantizando que solo se utilicen los datos relevantes para cada operación específica.

Las carpetas en Google Drive utilizadas para cargar y extraer ficheros también se organizan siguiendo esta estructura parametrizada. Existen tres carpetas principales:

- Inputs: Aquí se colocan los ficheros que se cargarán en la BBDD.
- Outputs: Aquí se generan y almacenan los ficheros extraídos de la BBDD.
- Plantillas: Aquí se guardan las plantillas utilizadas para generar los ficheros de Outputs.

Dentro de cada carpeta principal, se subdividen en carpetas específicas para cada cliente, hito y servicio. Esta organización permite un acceso ordenado y eficiente a los ficheros de interés y facilita la automatización de diferentes servicios e hitos para cada cliente en el futuro.

4.2.3 ARQUITECTURA BBDD: TABLAS Y PROCEDIMIENTOS

Antes de explicar las diferentes tablas y procedimiento utilizados dentro de la BBDD, se muestra una imagen que explica el flujo de trabajo de las diferentes tablas, así como, el momento en el que se ejecuta cada procedimiento (enumeración dentro de la ilustración). Haciendo referencia a la imagen del nuevo flujo de trabajo, se muestra internamente el flujo de los siguientes puntos de trabajo:

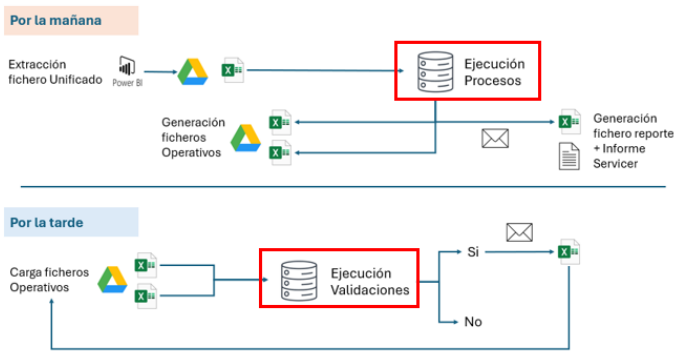

Ilustración 9: diseño nuevo flujo de trabajo. Fuente: elaboración propia

Los números mostrados en las siguientes imágenes hacen referencia al orden en el que se van ejecutando los diferentes procedimientos, se utiliza un alias para hacer referencia al procedimiento ejecutado para luego hacer una descripción de este.

4.2.3.1Ejecución validaciones.

Empezando por las ejecuciones de por la tarde, para que sea mas sencillo de entender el procedimiento:

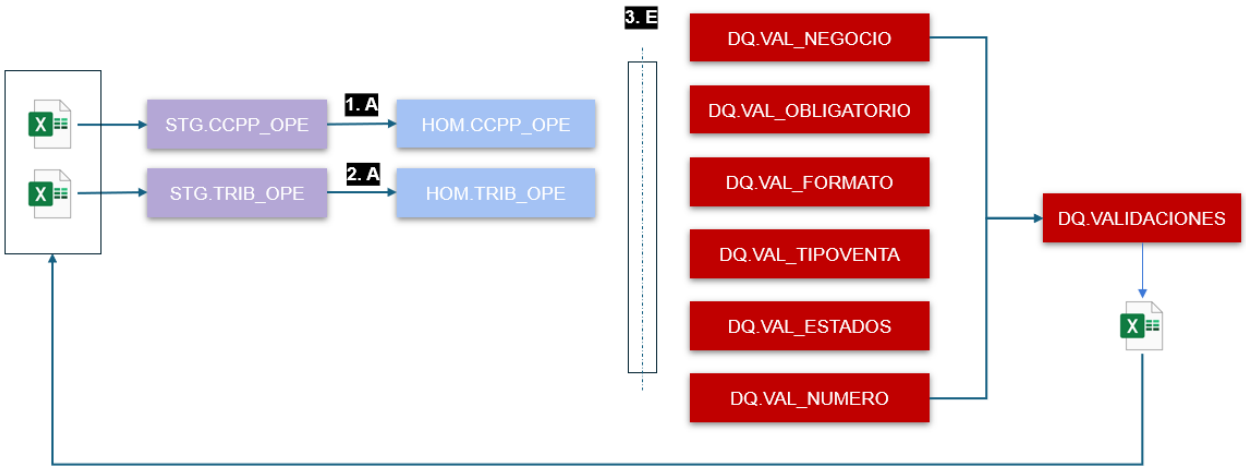

Ilustración 10: Flujo interno BBDD procedimientos por la tarde. Fuente: elaboración propia.

Como se ha explicado anteriormente, los ficheros operativos de CCPP y Tributos se cargan por la tarde en STG, a partir de ahí se ejecuta el procedimiento de homogeneización (A), quedando ambas tablas en su fase HOM y, sobre estas tablas, se ejecuta el procedimiento referente a las validaciones (E). este proceso se repetirá hasta que la tabla DQ.VALIDACIONES no presente registros para ventas y así cada fichero quedaría cargado en su fase HOM sin errores.

4.2.3.1.1 Procedimientos almacenados presentes:

HOM.HOMOGENEIZACION (A):

El procedimiento almacenado HOM.HOMOGENEIZACION es fundamental para el proceso de gestión de datos en la BBDD. Este procedimiento se encarga de transformar una tabla de la fase STG (Staging) a la fase HOM (Homogeneización). Su estandarización

permite su aplicación a cualquier tabla, siempre y cuando se hayan completado correctamente las tablas paramétricas que contienen la información necesaria para su ejecución.

Tablas paramétricas dentro del procedimiento:

- COD.PARAM_HOMOGENEIZACION_CAMPOS: Esta tabla paramétrica es esencial para la conversión de datos. Cada fila de esta tabla se refiere a un campo específico de una tabla particular. En ella se especifica la tabla de origen, la tabla de destino y el tipo de dato al cual se debe convertir cada campo. Esto asegura que los datos se transformen correctamente según los requisitos de la tabla destino.

ID_CLIENTE	ID_SERVICIO	ID_HITO	TABLA_ORIGEN	TABLA_DESTINO	CAMPO	TRANSFORMACION
1	7	1	STG.AAM_BI_VENTAS	HOM.AAM_BI_VENTAS	NMB_COMPRADOR	TRY_CAST(TRIM(NMB_COMPRADOR) AS NVARCHAR(255))
1	2	1	STG.AAM_OPE_CCPP_VENTAS	HOM.AAM_OPE_CCPP_VENTAS	VENTA_ESPECIAL	TRY_CAST(VENTA_ESPECIAL AS NVARCHAR(2))
1	7	1	STG.AAM_BI_VENTAS	HOM.AAM_BI_VENTAS	FEC_ULT_GESTION_TRIB	TRY_CAST(TRIM(FEC_ULT_GESTION_TRIB) AS DATE)
1	1	1	STG.AAM_OPE_TRIB_VENTAS	HOM.AAM_OPE_TRIB_VENTAS	VENTA_ESPECIAL	TRY_CAST(VENTA_ESPECIAL AS NVARCHAR(2))

Ilustración 11: ejemplo ciertos registros tabla COD.PARAM_HOMOGENEIZACION_CAMPOS

- COD.PARAM_CONCAT_SQL: Esta tabla define cómo se debe realizar la concatenación de diferentes campos para llenar el campo CONCAT_SQL. Este campo es un valor único por cada venta y es crucial para la historificación de los registros. Dependiendo de la tabla que se está homogeneizando, se aplicará una lógica de concatenación específica para asegurar la unicidad y consistencia de los datos.

ID_CLIENTE	ID_SERVICIO	ID_HITO	TABLA	CONCAT_SQL	
1	1	1	HOM.AAM_OPE_TRIB_VENTAS	CASE	WHEN CHARINDEX('/', REF_UE) = 0 A...
1	2	1	HOM.AAM_OPE_CCPP_VENTAS	CASE	WHEN CHARINDEX('/', REF_UE) = 0 A...
1	7	1	HOM.AAM_BI_VENTAS	CASE	WHEN CHARINDEX('/', REF_UE) = 0 A...

Ilustración 12: ejemplo ciertos registros tabla COD.PARAM_CONCAT_SQL

Además, el procedimiento HOM.HOMOGENEIZACION también tiene la función de rellenar la tabla de validación DQ.VAL_FORMATO. Si el tipo de dato especificado en la tabla paramétrica no coincide con el dato presente en la fase STG, el procedimiento insertará un registro en la tabla de validación DQ.VAL_FORMATO, indicando un error de formato.

Esto permite detectar y corregir inconsistencias de datos de manera eficiente, asegurando la integridad y calidad de la información procesada.

Funcionamiento resumido:

Se declara un cursor, por si hubiese varias tablas a homogeneizar con los mismos parámetros de entrada, y se guardan en él los nombres de las diferentes tablas origen y tablas destino (STG y HOM) según los parámetros de entrada.

Se eliminan los registros presentes en la tabla HOM a homogeneizar, para que la tabla quede vacía y finalmente queden registrados únicamente los registros presentes en STG.

```
--#######################################
--# 1.OBTENCIÓN TABLAS ORIGEN Y DESTINO #
--#######################################
EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION', '1.OBTENCION TABLAS ORIGEN Y DESTINO', NULL

--1.1. Declaración de cursor
EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION', '1.1.DECLARACION DE CURSOR', NULL
IF CURSOR_STATUS('global','tablas_origen_destino')>=-1
BEGIN
    DEALLOCATE tablas_origen_destino
END

DECLARE tablas_origen_destino CURSOR FOR
SELECT DISTINCT
    TABLA_ORIGEN,
    TABLA_DESTINO
FROM COD.PARAM_HOMOGENEIZACION_CAMPOS
WHERE ID_CLIENTE = @Cliente AND ID_SERVICIO = @Servicio AND ID_HITO = @Hito

OPEN tablas_origen_destino;
FETCH NEXT FROM tablas_origen_destino INTO @Tabla_Origen, @Tabla_Destino;
```

Una vez dentro del cursor, se seleccionan los campos a homogeneizar según la tabla paramétrica COD. PARAM_HOMOGENEIZACION_CAMPOS

```
--2.4. Seleccion de campos a homogeneizar
EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION', '2.4.SELECCION CAMPOS A HOMOGENEIZAR', NULL
SELECT
    CR.TABLA_ORIGEN AS TABLE_NAME,
    CR.CAMPO AS COLUMN_NAME,
    CR.TABLA_DESTINO AS TABLE_DEST,
    CR.TRANSFORMACION AS TRANSFORMACION,
    ROW_NUMBER() OVER(ORDER BY CR.TABLA_ORIGEN, CR.CAMPO, CR.TRANSFORMACION) AS RN
INTO #TMP_RELACION_CAMPOS
FROM [COD].[PARAM_HOMOGENEIZACION_CAMPOS] CR
WHERE CR.TABLA_ORIGEN=@Tabla_Origen
```

Se incluyen los datos de la tabla STG sobre una tabla temporal, esto se hace para evitar problemas con los tipos de datos, ya que, si se rellenase directamente la tabla HOM y algún campo no tuviese el formato correcto, daría error.

```
--2.6. Creacion de tabla temporal para realizar el proceso de homogeneización
EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION', '2.6.CREACION TABLA TEMPORAL STG';
SET @SqlInsert = 'SELECT *
INTO ##TMP_ORIGEN
FROM ' + @Tabla_Origen + '
;'
```

Mientras la columna a homogeneizar sea inferior al total de columnas de la tabla origen, se actualiza el campo de la tabla temporal aplicando la transformación del tipo de dato presente en la tabla paramétrica.

```
EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION', '2.7.INICIO TRANSFORMACION DEL DATO';
WHILE @contador <= @totalcolumnas
BEGIN
    BEGIN TRY
        --2.7.1 Inicializacion campos a transformar
        EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION', '2.7.1.INICIALIZACION CAMPOS A TRANSFORMAR';
        SELECT
            @tablename = TABLE_NAME,
            @columnname = COLUMN_NAME,
            @transformation = TRANSFORMACION
        FROM #TMP_RELACION_CAMPOS
        WHERE RN = @contador
        --2.7.2 Registro en log inicio de ejecución a nivel de campo
        SET @message = LEFT('2.7.2.INICIO HOMOGENEIZACION TABLA ' + @tablename + ' CAMPO ' + @columnname, 100)
        EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION', @message

        SET @query = 'UPDATE TABLA SET ' + @columnname + ' = ' + @transformation + ' FROM ##TMP_ORIGEN AS TABLA;'
        EXEC sp_executesql @query
        SET @contador = @contador + 1
    END TRY
    BEGIN CATCH
        --2.7.3 Insertar en tabla de log las excepciones
        SET @message = '2.7.3.ERROR HOMOGENEIZACION ' + @tablename + ' - ' + @columnname
        SET @error = ERROR_MESSAGE()
        EXEC DQ.LOG 'ERROR', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION', @message, @error
        SET @contador = @contador + 1
    END CATCH
END
```

Se incluyen los campos con errores de formato sobre la tabla DQ.VAL_FORMATO, esto es sencillo de identificar ya que la transformación aplicada sobre los campos se realiza con un TRY_CAST, función que cambia el tipo de dato y, en caso de no cuadrar, elimina el campo dejándolo en NULL. Por lo que tras aplicar las transformaciones, se establece un bucle que

recorra todos los campos de la tabla y compare el dato en STG y la temporal, detectando los campos eliminados y, por ende, con errores de formato.

```
SET @sqlQuery = '
    INSERT INTO DQ.VAL_CAMPOS_HOM
    SELECT
        '+CAST(@CLIENTE AS VARCHAR)+' AS ID_CLIENTE,
        '+CAST(@SERVICIO AS VARCHAR)+' AS ID_SERVICIO,
        '+CAST(@HITO AS VARCHAR)+' AS ID_HITO,
        ''' + @Tabla_Origen + ''' AS TABLA_ORIGEN,
        ''' + @Tabla_Destino + ''' AS TABLA_DEST,
        A.' + @ID_Unico + ' AS VALOR_ID,
        ''' + @campo_stg + ''' AS CAMPO,
        A.' + @campo_stg + ' AS VALOR_ORIGEN,
        B.' + @campo_stg + ' AS VALOR_DEST,
        GETDATE() AS FCH_DATOS
    FROM
        ' + @Tabla_Origen + ' A
    LEFT JOIN
        ##TMP_ORIGEN B ON A.' + @ID_Unico + ' = B.' + @ID_Unico + '
    WHERE
        A.' + @campo_stg + ' IS NOT NULL
        AND B.' + @campo_stg + ' IS NULL;'
    EXEC sp_executesql @sqlQuery
```

Se termina de rellenar la tabla HOM, generando las columnas checksum y concat_sql sobre la temporal y haciendo un insert de lo datos de la tabla temporal sobre la tabla destino. Tras tener la tabla destino rellena con los campos checksum y concat_sql vacíos, se procede a rellenarlos, haciendo referencia a la tabla paramétrica COD.PARAM_CONCAT_SQL y la función CHECKSUM.

```
--2.9. Completitud tabla HOM
EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION', '2.9.INICIO VOLCADO A TABLA HOM';
--2.9.1 Inclusion checksum y concat_sql
EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION', '2.9.1.INCLUSION CHECKSUM Y CONCAT_SQL';
ALTER TABLE ##TMP_ORIGEN
ADD CHECKSUM NVARCHAR(MAX),CONCAT_SQL NVARCHAR(MAX)
--2.9.2 Volcado a HOM
EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION', '2.9.2.VOLCADO A TABLA HOM';
SET @SqlInsert1 = 'INSERT INTO ' + @Tabla_Destino + ' ' +'SELECT * FROM ##TMP_ORIGEN;'
EXEC sp_executesql @SqlInsert1
--2.9.3 Completitud CONCAT_SQL en HOM
EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION', '2.9.3.COMPLETITUD CONCAT_SQL EN HOM';
BEGIN TRY
    SELECT @concatSQL=CONCAT_SQL
    FROM COD.PARAM_CONCAT_SQL
    WHERE ID_CLIENTE=@Cliente AND ID_SERVICIO=@SERVICIO AND ID_HITO=@HITO AND TABLA=@Tabla_Destino
    SET @sqlUpdateConcat = '
        UPDATE ' + @Tabla_Destino + '
        SET CONCAT_SQL = '+ @concatSQL +';'
    EXEC sp_executesql @sqlUpdateConcat;
END TRY
BEGIN CATCH
    SET @message = '2.9.3.ERROR EN EL RELLENO DE CONCAT_SQL DE LA TABLA ' + @Tabla_Destino
    SET @error = ERROR_MESSAGE()
    EXEC DQ.LOG 'ERROR', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION', @message, @error
END CATCH


--2.9.6 Completitud Checksum
EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION', '2.9.6.COMPLETITUD CHECKSUM EN HOM'
BEGIN TRY
    SELECT @columns = STRING_AGG('[' + COLUMN_NAME + ']', ', ')
    FROM INFORMATION_SCHEMA.COLUMNS
    WHERE TABLE_NAME = @tableDestino AND TABLE_SCHEMA= @schemaDestino AND COLUMN_NAME NOT IN('FEC_FICHERO','CHECKSUM')
    SET @checksum = 'UPDATE ' + @Tabla_Destino + ' ' + 'SET CHECKSUM = CHECKSUM(' + @columns + ');'
    EXEC sp_executesql @checksum
END TRY
BEGIN CATCH
    SET @message = '2.9.6.ERROR EN LA COMPLETITUD CHECKSUM EN HOM ' + @Tabla_Destino
    SET @error = ERROR_MESSAGE()
    EXEC DQ.LOG 'ERROR', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION', @message, @error
END CATCH
FETCH NEXT FROM tablas_origen_destino INTO @Tabla_Origen, @Tabla_Destino;
```

Al final del código se puede observar como el cursor pasa a realizar el mismo proceso sobre la siguiente tabla a homogeneizar en caso de que exista.

DQ.ORQUESTADOR_VALIDACIONES (E)

Es un procedimiento que llama a varios procedimientos almacenados, cada uno para ejecutar una validación específica. Como se puede observar en la ilustración 12, se presentan las siguientes validaciones:

- Validaciones de negocio: Estas validaciones aseguran que los campos rellenados por los gestores tengan coherencia y sentido lógico. Se definen mediante lógicas entre

diferentes campos, especificadas en una tabla paramétrica. El procedimiento utiliza esta tabla paramétrica para aplicar las validaciones adecuadas según la tabla que se esté evaluando.

- Validaciones de campos obligatorios: Este conjunto de validaciones verifica que todos los campos que no pueden estar nulos estén correctamente rellenados. Si algún campo obligatorio está vacío, se inserta una entrada en la tabla de validaciones correspondiente, indicando la falta.
- Validaciones de tipo de venta: Dado que se consolidaron los ficheros operativos de tres por departamento a uno solo, esta validación garantiza que cada activo tenga su tipo de venta claramente definido (Venta Especial y Postventa). Esto es crucial ya que las lógicas internas dependen de estos campos.
- Validaciones de estados: Se asegura que los estados indicados por los gestores sean correctos y estén predefinidos. No se permite la invención de estados intermedios que no existan en la configuración establecida.
- Validaciones de número de ventas: Dado que el fichero operativo ahora incluye todos los tipos de venta diferentes, es fundamental validar que las ventas normales coincidan entre sí y con las ventas del fichero del Servicer. Esto es esencial para asegurar que los datos reportados al Servicer sean precisos y consistentes.

Una vez que todas estas validaciones se han realizado y las tablas de validaciones se han rellenado, se ejecuta el procedimiento DQ.REP_VALIDACIONES. Este procedimiento compila los resultados de todas las validaciones en la tabla DQ.VALIDACIONES, proporcionando una vista consolidada de todos los errores presentes en cada tabla.

Funcionamiento resumido:

Cada validación funciona de una manera diferente en base al tipo de validación a realizar. Todas las tablas de validación secundarias se rellenan con la misma estructura para poder mostrar todos los errores de las diferentes tablas sobre la tabla DQ.VALIDACIONES de manera ordenada.

Hay ciertas validaciones que se ejecutan en base a tablas paramétricas, debido a que siguen ciertas lógicas, como las validaciones de negocio, de campos obligatorios o la validación de los estados rellenados por los gestores. Luego, hay otras validaciones que se ejecutan en base a lógicas directas, comparando las tablas de reporte de por la mañana con lo cargado por la tarde, como el número de ventas o la validación del tipo de venta.

Se incluye una ilustración de la tabla paramétrica de los campos de negocio para mostrar cómo se ejecuta la lógica. Sobre esta tabla se pueden añadir o eliminar validaciones según sea necesario. Las validaciones de campos obligatorios y de estado funcionan de la misma manera, utilizando sus respectivas tablas paramétricas.

ID_SERVICIO	ID_HITO	ID_CLIENTE	TABLA	CAMPO	VALIDACION
1	1	1	AAM_OPE_TRIB_VENTAS	FEC_INI_GESTION_TRIB	FEC_INI_GESTION_TRIB > FEC_FIN_GESTION_TRIB
7	1	1	AAM_BI_VENTAS	FEC_INI_GESTION_TRIB	FEC_INI_GESTION_TRIB > FEC_ULT_GESTION_TRIB
1	1	1	AAM_OPE_TRIB_VENTAS	FEC_ULT_GESTION_TRIB	FEC_ULT_GESTION_TRIB > FEC_FIN_GESTION_TRIB
7	1	1	AAM_BI_VENTAS	FEC_ENVIO_REMESA	(FEC_ENVIO_REMESA IS NOT NULL OR FEC_ENVIO_REMES...
1	1	1	AAM_OPE_TRIB_VENTAS	FEC_INI_GESTION_TRIB	FEC_INI_GESTION_TRIB IS NULL AND ESTADO_TRIB='OK'
1	1	1	AAM_OPE_TRIB_VENTAS	IMP_IBI_ANUALIZADO	ESTADO_TRIBUTOS IN ('APORTADO_JUSTIFICANTE', 'ENVIA...
7	1	1	AAM_BI_VENTAS	FEC_ENVIO_PAGO_TRIB	(FEC_ENVIO_PAGO_TRIB IS NOT NULL OR FEC_ENVIO_PAG...
1	1	1	AAM_OPE_TRIB_VENTAS	NUM_REMESA_TRIB	ESTADO_TRIBUTOS IN ('PTE_AUTORIZACION_PM', 'ENVIADO...
7	1	1	AAM_BI_VENTAS	NUM_REMESA_TRIB	FEC_ENVIO_REMESA IS NOT NULL AND NUM_REMESA_TRI...
2	1	1	AAM_OPE_CCPP_VENTAS	ETI_REMESA	ETI_REMESA IS NULL AND ESTADO_COMUNIDADES IN ('EN...

Ilustración 13: ejemplo ciertos registros tabla paramétrica COD.PARAM_CAMPOS_NEGOCIO

4.2.3.2Ejecución procesos

Una vez explicado las ejecuciones realizadas por la tarde, se presentan las ejecuciones de por la mañana. Cabe recordar, que los ficheros operativos se quedaron en el estado HOM sin errores del día anterior.

Una vez extraído el fichero de ventas del Servicer del Power BI y cargado el STG, los procedimientos y diferentes tablas involucradas son las siguientes:

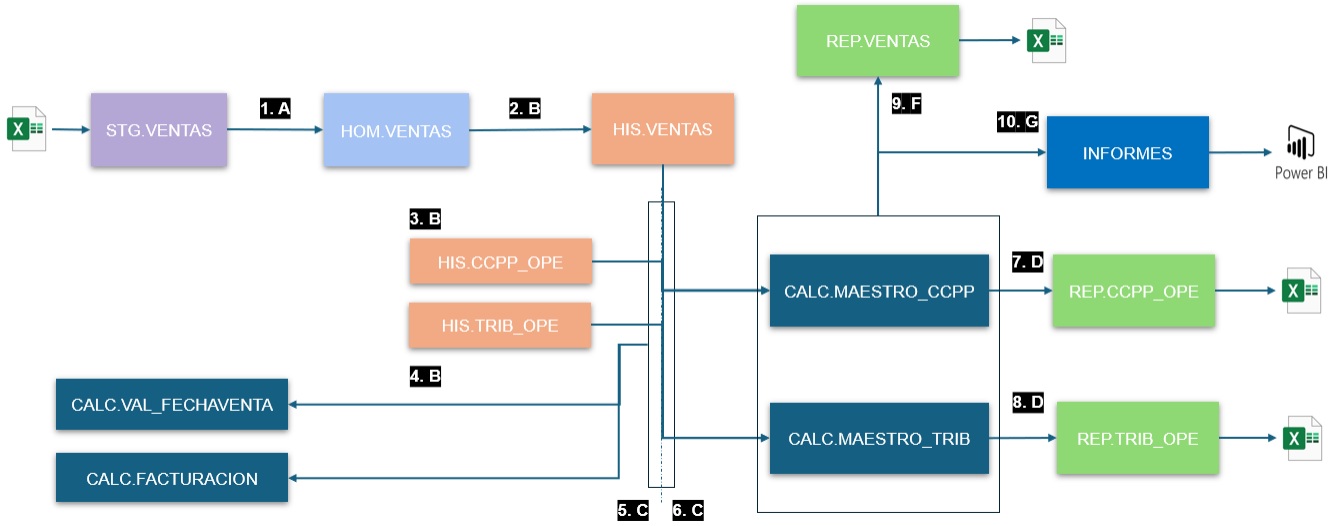

Ilustración 14: Flujo interno BBDD procedimientos por la mañana. Fuente: elaboración propia.

Nada más empezar el día, teniendo el fichero de ventas del Servicer y los ficheros operativos de CCPP y Tributos (con la información del final del día anterior) en HOM, se ejecuta la historificación para cada uno de ellos. Con los últimos registros de los históricos se actualizan los Maestros de CCPP y Tributos y, a partir de los Maestros, se generan los reportes a extraer de la BBDD.

La razón por la cual se hace de esta manera es debido a la actualización que hace el Servicer sobre su fichero de ventas. Cuando se llega a HIS.VENTAS, se actualizan las ventas dadas como Altas y Bajas, de manera que esta información se actualiza posteriormente sobre los Maestros: se insertan las Altas, se actualizan las Bajas (para que no salgan en los posteriores reportes) y los campos de gestión con los últimos registros de los históricos de los ficheros operativos. A partir de los maestros, se rellenan las diferentes tablas REP que posteriormente son extraídas a Excel, así como ciertas tablas que servirán de informes para mostrar los datos en un Power BI para los Coordinadores.

4.2.3.2.1 Procedimientos almacenados presentes:

HIS.HISTORIFICACION (B)

Este procedimiento se utiliza para actualizar las tablas histórico, para cada tabla cargada en STG y HOM, hay un histórico donde se acumulan todos los registros diarios. Este procedimiento se utiliza para rellenar la columna Estado, donde se indica si es Alta, Baja, Modificación o Realta. Como ya se ha explicado antes, la historificación se realiza según el campo CONCAT_SQL.

Funcionamiento resumido:

Se crea un cursor para historificar todas las tablas que hacen referencia a los parámetros introducidos en la ejecución del procedimiento.

```
EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logtitulo, '2.OBTENCION TABLAS A HISTORIFICAR'
IF CURSOR_STATUS('global', 'tablas')>=-1
BEGIN
    DEALLOCATE tablas
END

DECLARE tablas CURSOR FOR
SELECT DISTINCT TABLA_DESTINO
FROM COD.PARAM_HOMOGENEIZACION_CAMPOS
WHERE ID_CLIENTE = @Cliente AND ID_SERVICIO = @Servicio AND ID_HITO = @Hito

OPEN tablas
FETCH NEXT FROM tablas INTO @Tabla_Destino
```

Se crean dos tablas temporales, #FOTO_ACTUAL_BAJAS y #FOTO_ACTUAL, y se rellenan con el dato más actualizado (ROW_NUMBER) de cada registro presente en el histórico. La primera tabla temporal incluye los registros que están dados de baja y la segunda incluye los registros que no están dados de baja.

```
EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logtitulo,'5.OBTENCION FOTO ACTUAL'
EXEC('SELECT *
      INTO ##FOTO_ACTUAL
      FROM (
            SELECT *,
                  ROW_NUMBER() OVER(PARTITION BY '+@id_unico+' ORDER BY FEC_FICHERO DESC) AS R_N
            FROM HIS.'+@Tabla+'
        ) A
    WHERE R_N=1 AND ESTADO<>''BAJA''
    ')

EXEC('SELECT *
      INTO ##FOTO_ACTUAL_BAJAS
      FROM (
            SELECT *,
                  ROW_NUMBER() OVER(PARTITION BY '+@id_unico+' ORDER BY FEC_FICHERO DESC) AS R_N
            FROM HIS.'+@Tabla+'
        ) A
    WHERE R_N=1 AND ESTADO=''BAJA''
    ')
```

Una vez se rellenan las tablas temporales, se crea una tercera tabla temporal #BAJAS donde se insertarán los registros presentes en la temporal #FOTO_ACTUAL y no presentes en HOM, lo cual indica que es un registro que el día anterior no estaba dado de baja y, al no estar presente en HOM, hay que darlo de baja. A estos registros se les actualiza la columna Fec_fichero para que indique la fecha en la que se dio de baja. Finalmente, se insertan los registros de #BAJAS sobre la tabla HIS correspondiente,

```
--**************************
--6.1.Historificacion bajas *
---**************************
EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logtitulo,'6.1.HISTORIFICACION BAJAS'
--6.1.1.Seleccion bajas
EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logtitulo,'6.1.1.SELECCION BAJAS'
EXEC('SELECT HIS.*
      INTO ##BAJAS
      FROM ##FOTO_ACTUAL AS HIS
      LEFT JOIN HOM.' + @Tabla + ' HOM ON HIS.'+@id_unico+' = HOM.'+@id_unico+'
      WHERE HOM.'+@id_unico+' IS NULL;')
--6.1.2. Actualizacion FEC_FICHERO
EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logtitulo,'6.1.2.ACTUALIZACION FECHA FICHERO'
UPDATE ##BAJAS
SET FEC_FICHERO=@fchFichero,
    ESTADO='BAJA',
    FEC_SISTEMA=DATEADD(HH, 1, GETDATE())

--6.1.3.Insercion en tabla histórica
EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logtitulo,'6.1.3.INSERCION EN TABLA HISTORICA'
EXEC('INSERT INTO HIS.' + @Tabla + ' SELECT * FROM ##BAJAS');
```

Seguidamente se procede a historificar las altas. Estas se detectan de manera simple ya que son registros que no están presentes en HIS y sí están en HOM, lo cual indica que es un registro nuevo.

```
--*****************************
--6.2.Historificacion ALTAS *
---*****************************
EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logtitulo,'6.2.HISTORIFICACION ALTAS'
--6.2.1.Insercion altas
EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logtitulo,'6.2.1.INSERCION ALTAS'

EXEC('INSERT INTO HIS.' + @Tabla + '
      SELECT HOM.*,
             ''ALTA'' AS ESTADO,
             DATEADD(HH, 1, GETDATE()) AS FEC_SISTEMA
      FROM HOM.' + @Tabla + ' HOM
      LEFT JOIN HIS.' + @Tabla + ' HIS ON HOM.'+@id_unico+' = HIS.'+@id_unico+'
      WHERE HIS.'+@id_unico+' IS NULL;
```

El siguiente paso será historificar las modificaciones. El proceso inserta los registros sobre HIS indicando en la columna estado que se trata de una modificación. Estos casos se detectan gracias a la columna checksum rellenada en el procedimiento de homogeneización. Compara los registros presentes en #FOTO_ACTUAL y en HOM, de manera que si el campo checksum tiene valores diferentes, significa que alguno de ellos ha cambiado y por lo tanto ha habido una modificación en el registro.

```
--*************************************
--6.3.Historificacion MODIFICACIONES    *
---*************************************
EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logtitulo,'6.3.HISTORIFICACION MODIFICACIONES'
--6.3.1.Seleccion modificaciones
EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logtitulo,'6.3.1.INSERCION MODIFICACIONES'
EXEC('
    INSERT INTO HIS.' + @Tabla + '
    SELECT HOM.*,
        ''MODIFICACION'' AS ESTADO,
        DATEADD(HH, 1, GETDATE()) AS FEC_SISTEMA
    FROM ##FOTO_ACTUAL HIS
    INNER JOIN HOM.' + @Tabla + ' HOM ON HIS.'+@id_unico+' = HOM.'+@id_unico+'
    WHERE HOM.CHECKSUM <> HIS.CHECKSUM;
');
```

Por último, se procede a insertar sobre HIS las realtas. Estos son casos presentes en #FOTO_ACTUAL_BAJAS que a su vez están presentes en HOM, lo cual quiere decir que

el último registro del histórico indica que es una baja y se ha vuelto a dar de alta porque está presente en HOM.

```
--*******************************
--6.4.Historificacion REALTAS   *
---*******************************
EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logtitulo,'6.4.HISTORIFICACION REALTAS'
--6.4.1.Seleccion modificaciones
EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logtitulo,'6.4.1.INSERCION REALTAS'
EXEC('
        INSERT INTO HIS.' + @Tabla + '
        SELECT HOM.*, ''REALTA'' AS ESTADO, DATEADD(HH, 1, GETDATE()) AS FEC_SISTEMA
        FROM HOM.' + @Tabla + ' HOM
        INNER JOIN ##FOTO_ACTUAL_BAJAS ACT
        ON HOM.'+@id_unico+' = ACT.'+@id_unico+'')
```

CALC.GENERAR_MAESTRO_VENTAS (C)

Este procedimiento es el más importante del proyecto, ya que se encarga de actualizar los Maestros, tanto el de CCPP como el de Tributos.

Los Maestros muestran el registro más reciente de cada venta y son las tablas principales de toda la BBDD, permitiendo la implementación de diversas lógicas adicionales para obtener información extra de interés:

- Fecha de entrada: La fecha en que el activo fue dado de alta.
- Fecha de salida: La fecha en que el activo fue dado de baja.
- Tiempos de gestión: Calcula el tiempo de gestión de cada activo. Se subdividen en tres:
    - Tiempo desde que la venta se da de alta hasta que se comienza la gestión.
    - Tiempo desde que la venta se comienza a gestionar hasta que se factura.
    - Tiempo desde que la venta se da de alta hasta que el Coordinador asigna un Gestor.
- Fecha de facturación: Indica cuándo fue facturado el activo.
- Revisar: Una columna que se rellena cuando ocurre una Realta. En este caso, se copia el estado del último registro del maestro para que el Coordinador pueda determinar

si el activo parte desde ese punto de gestión o si se considera una nueva gestión. Esto se tendrá en cuenta en la facturación.

Además, en el Maestro también se tiene un claro registro de lo que está dado de Baja, y es la tabla principal de donde se extrae la información para generar los reportes.

En este procedimiento, además de actualizar los Maestros, se actualizan otras tablas de interés que servirán para realizar ciertos informes. Cabe destacar que estas tablas se actualizan al inicio del procedimiento, permitiendo comparar la información presente en los históricos (información más reciente) con la información presente en los maestros (información del día anterior por la mañana, ya que aún no se ha actualizado).

- CALC.VAL_FECHAVENTA: hace una comparativa al principio del procedimiento entre la fecha de venta indicada en el histórico del fichero unificado y la fecha de venta indicada en el Maestro. Si hay un cambio entre estos dos registros, se insertará una fila indicando el activo, la fecha antigua y la fecha de venta nueva, así como la fecha en la que ocurre el cambio, para luego poder sacar conclusiones.
- CALC.FACTURACION: siguiendo la misma lógica que en el punto anterior, se hace una comparativa del estado de gestión indicado en los históricos de los ficheros operativos y el estado de gestión indicado en el Maestro. Gracias a esta comparativa, se puede hacer un seguimiento de la facturación cuando el estado de gestión pasa de no facturable a facturable.
- CALC.VAL_ESTADOS_VENTAS: hace una comparativa de los estados de gestión en los históricos de los ficheros operativos con los estados de gestión del Maestro, introduciendo una fila en la tabla indicando el estado nuevo, el antiguo, el gestor que gestiona la venta, cierta información del inmueble y la fecha del cambio del estado. Con esto se consigue tener un registro diario del trabajo realizado por cada gestor.

Como se ha explicado anteriormente, la tabla Maestro rellena columnas nuevas en base a ciertas lógicas implementadas y, nunca se eliminan registros, siempre se muestra el más reciente según las actualizaciones del día a día en los históricos.

Funcionamiento resumido:

La primera parte del código es estandarizada y según los parámetros de entrada, 1 1 1 ó 1 2 1, se definen ciertas variables que hacen referencia al Maestro de Tributos o CCPP, respectivamente.

Comienza con un reseteo del Maestro, para que en el caso de que ocurra algún error, se pueda volver a versiones anteriores. Hay un parámetro de entrada extra que es @FCH_DATOS, la fecha para la cual se quiere ejecutar el Maestro, esto sirve para actualizar los Maestros con los datos más actuales de los tres históricos en función de esa fecha.

```
--###############################
--# 2.RESETEO MAESTRO POR FECHA #
--###############################
EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, '2.RESETEO MAESTRO POR FECHA'
--2.1. Borrado de todo lo que ha entrado al maestro después
EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, '2.1.BORRADO MAESTRO CON FECHA ENTRADA POSTERIOR A LA FECHA'
Set @sql='DELETE FROM CALC.AAM_MAESTRO_'+@SERV+'_VENTAS WHERE
         (FEC_ENTRADA_FC>='''+ CAST(@FCH_DATOS_1 AS VARCHAR)+''' AND FEC_ENTRADA_BI>='''+CAST(@FCH_DATOS AS VARCHAR)+''') OR
         (FEC_ENTRADA_FC IS NULL AND FEC_ENTRADA_BI>='''+CAST(@FCH_DATOS AS VARCHAR)+''') OR
         (FEC_ENTRADA_FC>='''+ CAST(@FCH_DATOS_1 AS VARCHAR)+''' AND FEC_ENTRADA_BI IS NULL)
    '
EXEC sp_executesql @sql
```

Además de eliminar los registros posteriores a la fecha, se hace una actualización de las columnas generadas en el propio procedimiento: fecha de entrada, fecha de salida, fecha de facturación, estado (alta, baja, modificación). Asegurando que el Maestro quede con la información correcta.

También se hace un reseteo de las tablas extras que se rellenan en este procedimiento:

```
--2.3.Borrado tablas posteriores a maestro
EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, '2.3.BORRADO TABLAS POSTERIORES A MAESTRO'
DELETE FROM CALC.VAL_FEC_POSICIONAMIENTO WHERE ID_CLIENTE=@CLIENTE AND ID_SERVICIO=@SERVICIO AND ID_HITO=@HITO AND FEC_CAMBIO>@FCH_DATOS
DELETE FROM CALC.VAL_ESTADOS_VENTAS WHERE ID_CLIENTE=@CLIENTE AND ID_SERVICIO=@SERVICIO AND ID_HITO=@HITO AND FEC_DATOS>@FCH_DATOS
```

Una vez completado el reinicio de todas las tablas afectadas (si lo hubiese), se crean 4 tablas temporales que se rellenan de manera estandarizada según sea CCPP o Tributos.

#FOTO_ACTUAL_BI: datos más recientes del histórico del fichero unificado (Servicer) que no estén dados de baja, es decir, lo que se podría llamar “vivo”.

```
EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito, @logtitulo, '3.INICIO CAMBIOS EN LOS REGISTROS'
--3.1.Obtencion perimetro vivo BI
EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, '3.1.OBTENCION VIVO BI'
SET @sql='SELECT *
          INTO ##FOTO_ACTUAL_BI
          FROM (
                SELECT *,
                       ROW_NUMBER() OVER (PARTITION BY CONCAT_SQL, ID_OPORTUNIDAD ORDER BY FEC_FICHERO DESC) AS RN
                FROM HIS.AAM_BI_VENTAS
                WHERE FEC_FICHERO<='''+CAST(@FCH_DATOS AS VARCHAR)+'''
               ) AS HIS_BI
          WHERE HIS_BI.RN = 1
          AND HIS_BI.ESTADO <> ''BAJA''
        '
EXEC sp_executesql @sql
```

#FOTO_ACTUAL_SERVICIO: datos más recientes del histórico del fichero operativo en cuestión que no estén dados de baja.

```
--3.2.Obtencion perimetro vivo servicio
EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, '3.2.OBTENCION VIVO SERVICIO'
SET @sql='SELECT *
           INTO ##FOTO_ACTUAL_SERVICIO
           FROM (
                 SELECT *,
                        ROW_NUMBER() OVER (PARTITION BY CONCAT_SQL ORDER BY FEC_FICHERO DESC) AS RN
                 FROM HIS.AAM_OPE_'+@SERV+'_VENTAS
                 WHERE FEC_FICHERO<='''+(CAST((@FCH_DATOS_1) AS VARCHAR))+'''
                ) AS HIS_BI
           WHERE HIS_BI.RN = 1
           AND HIS_BI.ESTADO <> ''BAJA''
         '
EXEC sp_executesql @sql
```

#FOTO_ACTUAL_BAJAS y #FOTO_ACTUAL_SERVICIO_BAJAS: Los registros dados de baja en el fichero unificado y en el operativo, respectivamente.

Las bajas del fichero unificado son las ventas que se han caído, esto lo actualiza el Servicer a diario y estos registros se usarán para dar de baja esas ventas en el Maestro y que no se extraigan en los ficheros de reporte.

Con respecto a las bajas del fichero operativo, tan solo se tienen en cuenta las que son Ventas Especiales y Postventas, ya que las altas y bajas de las ventas normales las indica el Servicer.

```
--3.3.Activos perimetro bajas en el BI (Hoy)
EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, '3.3.OBTENCION BI BAJAS DIARIO'
SET @sql='SELECT *
        INTO ##FOTO_ACTUAL_BI_BAJAS
        FROM (
              SELECT *, ROW_NUMBER() OVER (PARTITION BY CONCAT_SQL, ID_OPORTUNIDAD ORDER BY FEC_FICHERO DESC) AS RN
              FROM HIS.AAM_BI_VENTAS
              WHERE FEC_FICHERO='''+CAST(@FCH_DATOS AS VARCHAR)+'''
            ) A
        WHERE A.RN = 1 AND ESTADO = ''BAJA''';
EXEC sp_executesql @sql

--3.4.Activos perimetro bajas en el operativo CCPP (Hoy) - Ventas Especiales y Postventas fuera de fichero
EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, '3.4.OBTENCION SERVICIO BAJAS DIARIO'
SET @sql='SELECT *
         INTO ##FOTO_ACTUAL_SERVICIO_BAJAS
         FROM (
              SELECT *, ROW_NUMBER() OVER (PARTITION BY CONCAT_SQL ORDER BY FEC_FICHERO DESC) AS RN
              FROM HIS.AAM_OPE_'+@SERV+'_VENTAS
              WHERE FEC_FICHERO='''+CAST(@FCH_DATOS_1 AS VARCHAR)+'''
           ) A
           WHERE A.RN = 1 AND ESTADO = ''BAJA'' AND (VENTA_ESPECIAL=''SI'' OR POSTVENTA=''SI, NO PRESENTE EN EL BI'')'
EXEC sp_executesql @sql
```

Una vez rellenadas las tablas temporales con la información de interés, se ejecutan los siguientes procesos:

Inserción de registros en la tabla CALC.VAL_FEC_POSICIONAMIENTO, indicando en cada fila insertada información de interés sobre cambios en las fechas de ventas por parte del Servicer.

```
--4.1.Cambios en la fecha de posicionamiento
EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito, @logtitulo, '4.1.REGISTRO CAMBIOS FEC_POSICIONAMIENTO'
SET @sql='INSERT INTO CALC.VAL_FEC_POSICIONAMIENTO
        (ID_CLIENTE,ID_SERVICIO,ID_HITO,CONCAT_SQL,
        ID_OPORTUNIDAD, FEC_POSICIONAMIENTO_NEW, PRIORIDAD_PATRIMONIO_NEW,
        FEC_POSICIONAMIENTO_OLD, PRIORIDAD_PATRIMONIO_OLD, FEC_CAMBIO)
          SELECT '+CAST(@CLIENTE AS VARCHAR)+' AS ID_CLIENTE,
                 '+CAST(@SERVICIO AS VARCHAR)+' AS ID_SERVICIO,
                 '+CAST(@HITO AS VARCHAR)+' AS ID_HITO,
                 CALC.CONCAT_SQL,
                 CALC.ID_OPORTUNIDAD,
                 HIS.FEC_POSICIONAMIENTO AS FEC_POSICIONAMIENTO_NEW,
                 HIS.PRIORIDAD_PATRIMONIO AS PRIORIDAD_PATRIMONIO_NEW,
                 CALC.FEC_POSICIONAMIENTO AS FEC_POSICIONAMIENTO_OLD,
                 CALC.PRIORIDAD_PATRIMONIO AS PRIORIDAD_PATRIMONIO_OLD,
                 '''+CAST(@FCH_DATOS AS NVARCHAR)+''' AS FEC_CAMBIO
          FROM ##FOTO_ACTUAL_BI HIS
          INNER JOIN CALC.AAM_MAESTRO_'+@SERV+'_VENTAS AS CALC
          ON HIS.ID_OPORTUNIDAD = CALC.ID_OPORTUNIDAD
          AND HIS.CONCAT_SQL = CALC.CONCAT_SQL
          WHERE (CALC.FEC_POSICIONAMIENTO <> HIS.FEC_POSICIONAMIENTO
             OR (CALC.FEC_POSICIONAMIENTO IS NULL AND HIS.FEC_POSICIONAMIENTO IS NOT NULL)
             OR (CALC.FEC_POSICIONAMIENTO IS NOT NULL AND HIS.FEC_POSICIONAMIENTO IS NULL))
             AND CALC.ESTADO<>''BAJA'';'
    EXEC sp_executesql @sql
```

Inserción de registros en la tabla CALC.VAL_ESTADOS_VENTAS, indicando en cada fila insertada información de interés sobre el avance de los estados de gestión en cada venta, provienen de los ficheros operativos (campos de gestión actualizados).

```
EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito, @logtitulo, '4.2.REGISTRO CAMBIOS ESTADOS'
SET @sql='INSERT INTO CALC.VAL_ESTADOS_VENTAS
            (ID_CLIENTE,ID_SERVICIO,ID_HITO,CONCAT_SQL,
            ID_OPORTUNIDAD, GESTOR, ESTADO_NEW, ESTADO_OLD, FEC_DATOS)
            SELECT '+CAST(@CLIENTE AS VARCHAR)+' AS ID_CLIENTE,
                '+CAST(@SERVICIO AS VARCHAR)+' AS ID_SERVICIO,
                '+CAST(@HITO AS VARCHAR)+' AS ID_HITO,
                CALC.CONCAT_SQL,
                CALC.ID_OPORTUNIDAD,
                HIS.GESTOR_'+@SERV+' AS GESTOR,
                HIS.ESTADO_'+@servicio_nmb+' AS ESTADO_NEW,
                CALC.ESTADO_'+@servicio_nmb+' AS ESTADO_OLD,
                '''+CAST(@FCH_DATOS AS NVARCHAR)+''' AS FEC_DATOS
            FROM ##FOTO_ACTUAL_SERVICIO HIS
            INNER JOIN CALC.AAM_MAESTRO_'+@SERV+'_VENTAS CALC
            ON COALESCE(HIS.ID_OPORTUNIDAD,1) = COALESCE(CALC.ID_OPORTUNIDAD,1)
            AND HIS.CONCAT_SQL = CALC.CONCAT_SQL
            WHERE CALC.ESTADO_'+@servicio_nmb+' <> HIS.ESTADO_'+@servicio_nmb+'
            AND CALC.ESTADO<>''BAJA'';'
    EXEC sp_executesql @sql
```

Una vez terminada la parte del código estandarizada, se procede a la actualización del Maestro en cuestión, como estas tablas tienen diferentes columnas entre si (CCPP y Tributos), no se puede realizar de manera estandarizada.

El código completo se encuentra en los Anexos, pero se resume la lógica que sigue la actualización de los Maestros:

1. Actualización de la facturación: Comparativa entre el estado de gestión de los registros en la temporal del servicio y el Maestro. En el caso de que la gestión esté OK en la temporal y KO en el Maestro, significa que esa venta se factura ese mismo día.

```
--*******************************
--5.2.Actualizacion facturacion *
--*******************************
EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito, @logtitulo, '5.2.ACTUALIZACION FACTURACION'
--5.2.1.Añadir las columnas de facturación del maestro sobre el operativo
EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito, @logtitulo, '5.2.1.ADICION COLUMNAS FACTURACION A LA FOTO ACTUAL DEL SERVICIO'
ALTER TABLE ##FOTO_ACTUAL_SERVICIO
ADD FEC_FACT_TRIB DATE,
    FACT_TRIB INT;
--5.2.2.Copiar sobre la temporal las columnas de facturación
EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito, @logtitulo, '5.2.2.IDENTIFICACION DE ACTIVOS YA FACTURADOS'
UPDATE O
SET O.FEC_FACT_TRIB = M.FEC_FACT_TRIB,
    O.FACT_TRIB = M.FACT_TRIB
FROM ##FOTO_ACTUAL_SERVICIO AS O
INNER JOIN CALC.AAM_MAESTRO_TRIB_VENTAS AS M ON O.CONCAT_SQL = M.CONCAT_SQL
WHERE M.ESTADO<>'BAJA';

--5.2.4.Facturar los activos que pasan de KO a OK
EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito, @logtitulo, '5.2.3.ACTUALIZACION FECHA FACTURACION PARA LO FACTURABLE'
UPDATE OPE
SET OPE.FACT_TRIB= 1,
    OPE.FEC_FACT_TRIB = @FCH_DATOS_1
FROM ##FOTO_ACTUAL_SERVICIO OPE
LEFT JOIN COD.ESTADO E
ON OPE.ESTADO_TRIBUTOS = E.ESTADO
WHERE E.OK_KO='OK' AND E.FACT=1
AND (OPE.REVISAR_TRIB IS NULL OR OPE.REVISAR_TRIB IN ('NUEVA_GESTION','GESTION_ANTERIOR'))
AND OPE.FEC_FACT_TRIB IS NULL
AND OPE.FACT_TRIB IS NULL
AND EXISTS (
        SELECT 1
        FROM CALC.AAM_MAESTRO_TRIB_VENTAS M
        LEFT JOIN COD.ESTADO E2 ON M.ESTADO_TRIBUTOS=E2.ESTADO
        WHERE OPE.CONCAT_SQL=M.CONCAT_SQL
            AND E2.OK_KO='KO' AND M.ESTADO<>'BAJA'
)
```

2. Inserción de Altas de Ventas Especiales (#FOTO_ACTUAL_SERVICIO): Se insertan registros que están presentes en la tabla temporal y no están presentes en el Maestro, con la condición de que Venta Especial sea "sí".
3. Inserción de Altas de Ventas Normales (#FOTO_ACTUAL): Se insertan registros que están presentes en la tabla temporal y no están presentes en el Maestro.
4. Actualización de Bajas de Ventas Especiales y Postventas (#FOTO_ACTUAL_SERVICIO_BAJAS): Para los registros presentes en la tabla temporal, se actualiza el campo Estado a "Baja" en el Maestro.
5. Actualización de Bajas de Ventas Normales (#FOTO_ACTUAL_BAJAS): Para los registros presentes en la tabla temporal, se actualiza el campo Estado a "Baja" en el Maestro.

6. Actualización de los campos de gestión (#FOTO_ACTUAL_SERVICIO): Se realiza una unión entre la tabla temporal y el Maestro, actualizando los campos de gestión en el Maestro.
7. Actualización de campos a revisar: Si en la tabla temporal (#FOTO_ACTUAL) existe un registro como Alta y ese registro ya estaba presente en el Maestro, se inserta un registro adicional duplicado, copiando los campos de gestión del registro anterior y actualizando la columna Revisar a "Revisar". El registro anterior se marcaría como Baja si aún no lo está. El Coordinador del departamento debe indicar en la columna Revisar si se trata de una nueva gestión o si continúa desde el punto de gestión anterior. Si es una nueva gestión y previamente estaba gestionada, el activo se facturaría dos veces.
8. Actualización de campos de información del activo (#FOTO_ACTUAL): Los campos actualizados por el Servicer también deben actualizarse en el Maestro.
9. Actualización de los campos de fechas de entrada y salida: Hay dos tipos de fechas de entrada y dos tipos de fechas de salida, diferenciándose en si se refiere a una Venta Normal o no. Esto es necesario para poder hacer un seguimiento detallado, ya que puede haber un caso en el que una Venta Especial se dé de alta posteriormente en el fichero del Servicer como una Venta Normal.

```
--****************************************
--5.11.Actualizacion fecha de entrada   *
--****************************************
EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito, @logtitulo, '5.11.ACTUALIZACION FECHA ENTRADA'
UPDATE CALC.AAM_MAESTRO_TRIB_VENTAS
SET FEC_ENTRADA_FC = CASE WHEN (VENTA_ESPECIAL='SI' OR POSTVENTA='SI, NO PRESENTE EN EL BI') AND FEC_ENTRADA_FC IS NULL THEN @FCH_DATOS_1
                          ELSE FEC_ENTRADA_FC END,
    FEC_ENTRADA_BI = CASE WHEN (VENTA_ESPECIAL='NO' AND (POSTVENTA IS NULL OR POSTVENTA ='SI')) AND FEC_ENTRADA_BI IS NULL THEN @FCH_DATOS
                          ELSE FEC_ENTRADA_BI END,
    FEC_SALIDA_FC = CASE WHEN (VENTA_ESPECIAL='SI' OR POSTVENTA='SI, NO PRESENTE EN EL BI') AND FEC_SALIDA_FC IS NULL AND ESTADO='BAJA' THEN @FCH_DATOS_1
                          ELSE FEC_SALIDA_FC END,
    FEC_SALIDA_BI = CASE WHEN (VENTA_ESPECIAL='NO' AND (POSTVENTA IS NULL OR POSTVENTA ='SI')) AND FEC_SALIDA_BI IS NULL AND ESTADO='BAJA' THEN @FCH_DATOS
                          ELSE FEC_SALIDA_BI END

UPDATE CALC.AAM_MAESTRO_TRIB_VENTAS
SET FEC_SALIDA_FC = CASE WHEN FEC_ENTRADA_FC<FEC_ENTRADA_BI THEN FEC_ENTRADA_BI
                          ELSE NULL END
WHERE FEC_SALIDA_FC IS NULL AND FEC_ENTRADA_BI IS NOT NULL AND FEC_ENTRADA_FC IS NOT NULL
```

10. Actualización de los campos que miden los diferentes tiempos de gestión descritos anteriormente.

```
--3.14.3.Rellenar tiempos medios
UPDATE A
SET AGING_MEDIO_VENTA=
    CASE
        WHEN FEC_ENTRADA_BI IS NOT NULL AND FEC_ENTRADA_FC IS NOT NULL
            THEN DATEDIFF(DAY, IIF(FEC_ENTRADA_BI < FEC_ENTRADA_FC, CAST(FEC_ENTRADA_BI AS DATE),
                CAST(FEC_ENTRADA_FC AS DATE)),FEC_POSICIONAMIENTO)
        WHEN FEC_ENTRADA_BI IS NOT NULL AND FEC_ENTRADA_FC IS NULL
            THEN DATEDIFF(DAY, CAST(FEC_ENTRADA_BI AS DATE),FEC_POSICIONAMIENTO)
        WHEN FEC_ENTRADA_FC IS NOT NULL AND FEC_ENTRADA_BI IS NULL
            THEN DATEDIFF(DAY, CAST(FEC_ENTRADA_FC AS DATE),FEC_POSICIONAMIENTO)
        ELSE NULL
    END,
    AGING_MEDIO_GESTION =
    CASE
        WHEN FEC_INI_GESTION_TRIB IS NOT NULL AND FEC_FACT_TRIB IS NOT NULL
            THEN DATEDIFF(DAY, FEC_INI_GESTION_TRIB,FEC_FACT_TRIB)
        ELSE NULL
    END,
    AGING_MEDIO_INICIO_GESTION =
    CASE
        WHEN FEC_ENTRADA_BI IS NOT NULL AND FEC_ENTRADA_FC IS NOT NULL
            THEN DATEDIFF(DAY, IIF(FEC_ENTRADA_BI < FEC_ENTRADA_FC, CAST(FEC_ENTRADA_BI AS DATE),
                CAST(FEC_ENTRADA_FC AS DATE)),FEC_INI_GESTION_TRIB)
        WHEN FEC_ENTRADA_BI IS NOT NULL AND FEC_ENTRADA_FC IS NULL
            THEN DATEDIFF(DAY, CAST(FEC_ENTRADA_BI AS DATE),FEC_INI_GESTION_TRIB)
        WHEN FEC_ENTRADA_FC IS NOT NULL AND FEC_ENTRADA_BI IS NULL
            THEN DATEDIFF(DAY, CAST(FEC_ENTRADA_FC AS DATE),FEC_INI_GESTION_TRIB)
        ELSE NULL
    END
FROM CALC.AAM_MAESTRO_TRIB_VENTAS A
```

Al final del Maestro, se ejecuta el procedimiento para actualizar la tabla, a modo de reporte, que muestra la facturación total, insertando las ventas facturadas día a día.

```
--##################################################
--# 7.ACTUALIZACION TABLA REP.AAM_FACTURACION_VENTAS #
--##################################################
EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito, @logtitulo, '7.ACTUALIZACION TABLA REP.AAM_FACTURACION_VENTAS'
EXEC REP.ACTUALIZAR_FACTURACION_VENTAS @cliente, @servicio, @hito, @FCH_DATOS_1
```

REP.GENERAR_REPORTES_OPERATIVOS (D)

Procedimiento que se encarga de rellenar las tablas REP de lo que van a ser los ficheros operativos de CCPP y Tributos que posteriormente van a ser exportadas a Excel.

Se extraen todos los diferentes tipos de venta presentes en el Maestro que no estén dados de Baja, es decir, todo lo que está “vivo”.

REP.GENERAR_REPORTE_SERVICER (F)

Lo mismo que el anterior procedimiento, pero para rellenar la tabla REP de lo que va a ser el fichero que se reporta al Servicer. En este caso se hace una unión de ambos maestros para tener en un mismo fichero los tres tipos de campos a reportar (información del activo, campos de CCPP y campos de Tributos) y se filtran por las ventas normales no dadas de baja, lo cual tiene que coincidir con lo mismo que se cargó en STG.VENTAS, pero con los campos de gestión actualizados con el trabajo realizado el día anterior, para que el Servicer pueda actualizar esta información sobre su fichero.

CALC.GENERAR_INFORMES_VENTAS (G)

En este procedimiento se generan diferentes tablas que serán extraídas al Power BI donde tendrán acceso los Coordinadores. Se muestran los siguientes informes:

- REP.VENTAS_GESTOR: es una tabla donde se muestra un análisis de las gestiones que realizan cada Gestor, indicando el número de ventas asignadas y la consecución propia de cada uno, así como las ventas facturadas por Gestor.
- REP.VENTAS_CARTERAS: como las ventas se dividen en diferentes carteras en las cuales hay subequipos de Gestores (Banco o TP), también hay un informe detallado de la consecución y número de ventas diferenciado por cartera.
- REP.VENTAS_GESTOR_DIARIO: muestra las gestiones realizadas en cada día del mes, para tener más información de cara a repartir trabajo entre los Gestores.
- REP.REPORTING_VENTAS: una tabla que muestra el número de gestiones mes a mes y la consecución de cada mes, para ver de un simple vistazo como se llevan las gestiones del año en curso.

Capítulo 5. IMPLEMENTACIÓN DEL PROYECTO

5.1 DESARROLLOS Y AUTOMATIZACIONES

Con el objetivo de automatizar el flujo de trabajo, se desarrollan ciertos scripts en Python. La estructura del programa es simple y ordenada, de manera que sea sencillo encontrar errores de ejecución y se puedan establecer condiciones que, dependiendo de los parámetros de entrada en la ejecución de diferentes scripts, se pudieran aplicar las casuísticas a considerar o realizar automatizaciones extra.

El proyecto se compone de diferentes códigos divididos en clases, Jobs de ejecución y un Schedule.

5.1.1 EJECUCIONES EN EL FLUJO DE TRABAJO

Para explicar el funcionamiento de las automatizaciones en Python, se aplicará la misma estructura que en el apartado anterior, en base al orden de ejecución según el flujo de trabajo, se desarrolla una explicación de cada función implicada en el funcionamiento del sistema.

5.1.1.1 Tarde: Job_carga_operativos.py

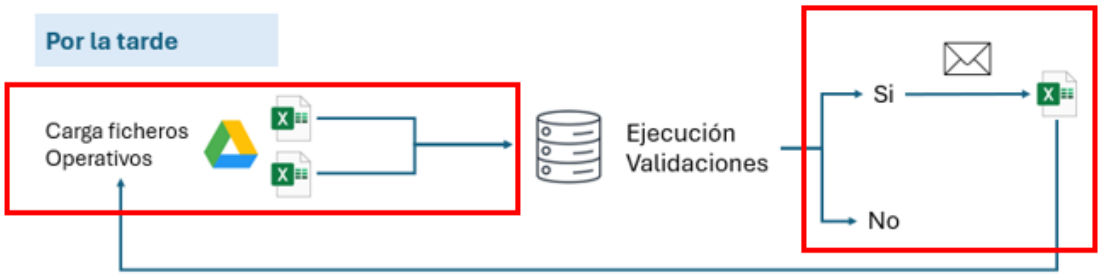

Ilustración 15: Flujo de trabajo por la tarde. Automatización de procesos. Fuente: elaboración propia

Para la ejecución de este proceso, se elabora un único script llamado Job_carga_operativos.py en el que se reúnen las diferentes funciones a emplear para ejecutar desde la carga de los ficheros hasta la validación de errores y su posterior extracción, para que el Coordinador en cuestión pueda corregirlas y el fichero quede cargado sin errores.

Para realizar este proceso, se ejecuta el script indicando los parámetros de entrada correspondientes al fichero a cargar. Según estos parámetros, se ejecutan las siguientes funciones:

Funciones Principales

```
#3.Carga ficheros operativos
ejecutarCarga(id_cliente,id_servicio,id_hito,id_tipo_fichero_carga,log)
#4.Homogeneización ficheros operativos
ejecutarHomogeneizacion(conexion,id_cliente,id_servicio,id_hito,log)
#5.Validación ficheros operativos
ejecutarValidaciones(conexion,id_cliente,id_servicio,id_hito,log)
#6.Reporting Operativos
count=ejecutarReporting(conexion,id_cliente,id_servicio,id_hito,id_tipo_fichero_rep,log)
#7.Volcado a ficheros
ejecutarVolcado(id_cliente,id_servicio,id_hito,id_tipo_fichero_rep,log,count)
#8.Envío correo validaciones
envioMail(conexion,id_cliente,id_servicio,id_hito,id_tipo_fichero_rep,count)
#9.Resetear reportes para volver a ejecutar validaciones
ejecutarReporteMaestro(conexion, id_cliente,id_servicio,id_hito,id_tipo_fichero_rep,log, count)
```

EjecutarCarga

La función ejecutarCarga inicializa la clase ejecutorCarga() y llama a la función ejecutar, esta clase se encuentra en el script Carga.py.

El script Carga.py se utiliza para procesar y cargar datos desde archivos en la BBDD. Este proceso se realiza mediante la clase ejecutorCarga, que contiene varias funciones para gestionar la carga de datos según los parámetros de entrada.

Descripción de las Funciones en Carga.py:

- __init__: Función inicializadora de la clase. Configura los parámetros de entrada y carga las configuraciones necesarias desde un archivo Config.ini, donde se encuentran los parámetros genéricos para todos los procedimientos.
- obtenerFicheros: Realiza una consulta en la tabla COD.FICHEROS de la BBDD para identificar los datos necesarios para la carga, como el nombre del archivo, la hoja a cargar, la tabla de destino en SQL, la ruta del archivo y las columnas a cargar, entre otros.

- regularizarFechas: Formatea las fechas en los datos de entrada para asegurar que tengan un formato uniforme (YYYY-MM-DD). Utiliza expresiones regulares para detectar y transformar diferentes formatos de fecha.
- obtener_nombres_columnas_sql: Obtiene los nombres de las columnas de la tabla SQL de destino (STG) para asegurarse de que los datos se carguen con los nombres correctos.
- excelToDataframe: Convierte los archivos de Excel, CSV o TXT en un dataframe de pandas. Este dataframe se usa posteriormente para cargar los datos en la base de datos.
- borrado_previo: Borra los datos existentes en la tabla de destino STG antes de cargar los nuevos datos.
- cargar: Realiza la carga de los datos en la tabla de destino en SQL. Inserta la columna Fec_fichero (fecha presente en el propio nombre del archivo), transforma los datos y carga los datos en lotes para optimizar el rendimiento. Mueve los archivos procesados a una carpeta "OLD".
- ejecutor: Coordina todo el proceso de carga. Obtiene los archivos, procesa los datos, realiza el borrado previo, obtiene los nombres de las columnas y llama a la función de carga. Registra cada paso y maneja errores si no se encuentran archivos o si ocurre algún problema durante el procesamiento.

EjecutarHomogeneizacion

Ejecuta el procedimiento HOM.HOMOGENEIZACION.

EjecutarValidaciones

Ejecuta el procedimiento DQ.ORQUESTADOR_VALIDACIONES

EjecutarReporting

Ejecuta el procedimiento DQ.REP_VALIDACIONES, para rellenar la tabla DQ.VALIDACIONES con los respectivos errores presentes y cuenta el número de registros

sobre esta tabla que hacen referencia a errores de la tabla que se ha cargado recientemente, este numero se introduce en la variable count.

En caso de que count sea 0, no existen validaciones y la ejecución habrá finalizado, quedando la tabla del fichero operativo cargado sobre HOM. En el caso contrario, existen validaciones y por lo tanto se continúa con el proceso.

Para el caso en el que count>0, se ejecuta el procedimiento REP.GENERAR_REPORTE_OPERATIVOS indicando un parámetro de entrada extra que hace que el relleno de la tabla que luego se extraerá sobre el Excel lo haga desde la tabla HOM en el lugar del Maestro, para que los datos queden actualizados según lo que se ha cargado. Este Excel extraído tiene una hoja llamada Validaciones donde se incluirán los errores a corregir por el Coordinador.

EjecutarVolcado

Esta función inicializa la clase generadorFicheros() y llama a la función generacionFicheros, esta clase se encuentra en el script GeneradorFicheros.py.

El script GeneradorFicheros.py se utiliza para generar y formatear archivos de reporte en Excel basados en datos de la BBDD y presenta diferentes funciones.

Descripción de las Funciones

- __init__: Inicializa la clase con los parámetros de entrada y configura las conexiones y rutas necesarias.
- obtencionParametrosFicheros: Realiza una consulta la tabla COD.FICHEROS para identificar los parámetros necesarios para la generación de los archivos de reporte, como el nombre del archivo, la hoja a generar, la ruta de destino, las columnas de fechas e importes a formatear
- obtencionRutas: Genera las rutas de origen y destino de las plantillas basándose en los parámetros obtenidos.

- copiadoPlantilla: Copia la plantilla de archivo desde la ruta de origen (ruta donde se encuentra la plantilla) a la ruta de destino, indicando en el nombre del fichero la fecha de extracción. Si el archivo ya existe en la ruta de destino, lo elimina antes de copiar la nueva plantilla.
- formatearImportes: Convierte las columnas especificadas como importes a tipo numérico dentro del DataFrame. Dentro de esta función se ejecuta es_numero, que comprueba si un valor es numérico.
- convertirNumero: Convierte las celdas especificadas a formato numérico dentro de una hoja de Excel. Para evitar errores de formato en columnas con códigos formados por números enteros que no son importes.
- formatearFechas: Formatea las columnas de fechas en el DataFrame o en la hoja de Excel (fichero de extracción) según el parámetro recibido.
- create_validations: Crea validaciones de datos para ciertas celdas de la hoja de Excel. A partir de los valores introducidos en la hoja Aux de la plantilla, se crean desplegables sobre las columnas indicadas.
- insertarDatos: Inserta los datos del DataFrame en la hoja de Excel especificada.
- generacionFicheros: Coordina todo el proceso de generación de archivos. Obtiene los parámetros, copia las plantillas, inserta los datos y aplica las validaciones y formateos.

EnvioMail

En el caso de que existan validaciones, se envía un correo automatizado al Coordinador en cuestión indicando que debe corregir las validaciones y subir de nuevo el fichero a la ruta de carga. En el propio correo se adjunta el fichero operativo con las validaciones a corregir.

EjecutarReporteMaestro

Una vez se envía el correo, se vuelve a ejecutar el procedimiento REP.GENERAR_REPORTE_OPERATIVOS para que esta vez se ejecute con el Maestro, para que queden los mismos registros que en el fichero de extracción de por la mañana y las validaciones se ejecuten de manera correcta cuando se vuelva a cargar el fichero.

5.1.1.2Mañana: Job_Carga_BI.py, Job_Historificacion.py, Job_Maestros.py, Job_Reporting.py y Job_BI_Nforce.py.

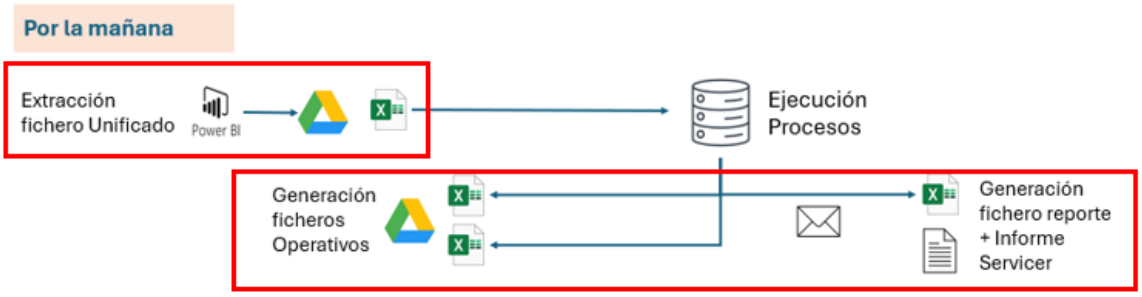

Ilustración 16: Flujo de trabajo por la mañana. Automatización de procesos. Fuente: elaboración propia

Para la ejecución de este proceso, se elaboran diferentes Jobs de ejecución que realizan funciones específicas dentro del flujo de trabajo. Se ha querido separar en varios debido a que por la mañana se ejecutan muchos procesos e interesa tener un orden a modo de facilidad de compresión.

5.1.1.2.1 Job_Carga_BI.py

Utilizado para realizar la extracción del fichero unificado y cargarlo en la BBDD.

Funciones principales

```
#3.Extraccion ficheros bi
ejecutarExtaccion(id_cliente,id_servicio,id_hito,log)
#4.Carga fichero BI
ejecutarCarga(id_cliente,id_servicio,id_hito,id_tipo_fichero_carga,log)
#5.Homogeneización fichero bi
ejecutarHomogeneizacion(conexion,id_cliente,id_servicio,id_hito,log)
```

EjecutarExtraccion

Esta función inicializa la clase extractorBI y llama a la función extracionBI, la clase se encuentra en el script Carga_BI.py.

Este script utiliza selenium para navegar a través de una cuenta externa por la interfaz del Power BI del Servicer y, con la definición de coordenadas y haciendo clics, llegar a la descarga del fichero unificado.

Descripción de las funciones

- __init__: Inicializa la clase extractorBI con los parámetros de entrada. Configura las opciones del navegador y carga los parámetros necesarios desde un archivo de configuración (Config.ini).
- extraccion_bi: Realiza todo el proceso de extracción de datos desde el sitio web:
    - Configura las opciones del navegador (tamaño de ventana, deshabilitación de GPU, preferencias de descarga).
    - Inicia el navegador y abre la URL de destino.
    - Inicia sesión utilizando las credenciales proporcionadas.
    - Navega por la interfaz web para seleccionar y exportar la tabla deseada.
    - Espera a que la descarga del archivo se complete.
    - Renombra y mueve el archivo descargado a la ubicación de destino (ruta input para cargar).
    - Cierra el navegador.

EjecutarCarga

Inicializa la clase de carga y carga el fichero en STG como anteriormente se ha explicado.

EjecutarHomogenizacion

Ejecuta el procedimiento HOM.HOMOGENEIZACION, terminado el script con la tabla HOM del fichero unificado rellena.

5.1.1.2.2 Job_Historificacion.py

Este script está específicamente para ejecutar el procedimiento HIS.HISTORIFICACION según los parámetros de entrada introducidos.

5.1.1.2.3 Job_Maestros.py

Este script ejecuta el procedimiento CALC.GENERAR_MAESTRO_VENTAS según los parámetros de entrada introducidos.

5.1.1.2.4 Job_Reporting.py

Una vez están actualizados ambos Maestros, se ejecutan el job de reporting que realiza unas funciones u otras dependiendo de los parámetros de entrada introducidos. Las funciones principales serán ejecutar los procedimientos para rellenar las tablas REP a exportar sobre Excel y, en el caso de se trate del reporting a cliente, se genera un mail automático que hace llegar a los diferentes Coordinadores y personas implicadas del Servicer con un informe detallado de las ventas con fecha de posicionamiento el mes actual, adjuntando los ficheros de interés, como se puede observar en la imagen del flujo de trabajo.

Funciones principales

```
#3.Reporting Operativos
ejecutarReporting(conexion,id_cliente,id_servicio,id_hito,id_tipo_fichero_rep,log)
#4.Volcado a ficheros
ejecutarVolcado(id_cliente,id_servicio,id_hito,id_tipo_fichero_rep,log)
#5.Envio mail a responsables
envioMail(conexion,id_cliente,id_servicio,id_hito,id_tipo_fichero_rep)
```

EjecutarReporting

Función que ejecuta el procedimiento REP.GENERAR_REPORTE_OPERATIVOS o REP.GENERAR_REPORTE_SERVICER en función del parámetro id_servicio, este procedimiento alimenta las tablas de reporte que luego se exportarán a Excel.

EjecutarVolcado

Inicializa la clase generadorFichero y ejecuta la función para extraer los ficheros operativos o de reporte a cliente en sus respectivas rutas.

EnvioMail

En caso de que sean los ficheros operativos los que se van a extraer, los propios Coordinadores podrán encontrar el fichero en la ruta Outputs y empezar a trabajar con ellos, en caso de que se trate del fichero a reportar al cliente, se ejecuta la función envioMail, que como se ha explicado anteriormente, mandará un correo con información de interés adjuntando el fichero de reporte con los campos de gestión actualizados.

5.1.1.2.5 Job_BI_Nforce.py

Este script se encarga de ejecutar el procedimiento que actualiza las tablas de informe que posteriormente se muestran en un Power BI interno donde tendrán acceso los Coordinadores para realizar cualquier tipo de análisis. También proporciona información de interés a nivel interno de EG.

Funciones principales

```
#3.Generación informes ventas
ejecutarqueryBI(conexion, fch_datos,log)
enviarcorreo(conexion,log,ruta)
```

EjecutarqueryBI

Esta función ejecuta el procedimiento CALC.GENERAR_INFORMES_VENTAS para actualizar las diferentes tablas presentes en este procedimiento posteriormente mostradas a los Coordinadores para realizar sus propios análisis, establecer prioridades y llevar un seguimiento del trabajo realizado por los diferentes gestores de cada departamento.

EnviarCorreo

A modo de seguridad, se hace una comprobación de que los números de ventas normales en los diferentes ficheros de reporte coincidan y, en el caso de que no lo haga, mandará un correo indicando la tabla que descuadra y que se debe revisar la BBDD. En el caso contrario, mandará otro correo que presenta un informe con un pequeño resumen en relación con las ventas presentes en los ficheros operativos y adjunta un fichero que muestra la facturación del mes actual, así como otro informe indicando de donde provienen las ventas a facturar.

Adjunto Informe diario de ventas:

Fecha del informe: 2024-06-14
Nº de ventas en BI: 3417
Altas BI: 95
Bajas BI: 18
Operativo de Tributos: - Ventas Totales: 3670
- Postventas fuera de fichero: 231
- Ventas fuera de fichero: 22
- Activos a Revisar: 33
Operativo de CCPP: - Ventas Totales: 3946
- Postventas fuera de fichero: 483
- Ventas fuera de fichero: 46
- Activos a Revisar: 33
Facturación total mes actual Tributos: 299
- Facturadas día anterior: 14
Facturación total mes actual CCPP: 327
- Facturadas día anterior: 25

Ilustración 17: Informe interno a modo de ejemplo. Resumen de los tipos de ventas dentro de los ficheros operativos.

5.2 INTEGRACIÓN DE SISTEMAS

Para poder ejecutar todos los procedimientos de manera automática, EG contrata un servicio en Google Cloud para crear una instancia en una máquina virtual. Esto se hace con el objetivo de que el sistema quede completamente automatizado y no necesite intervención humana.

5.2.1 CLASES PYTHON NECESARIAS PARA LA INTEGRACIÓN

Para poder ejecutar los procedimientos de la BBDD a través de Python, se ha creado un script llamado sqlconnector.py en el que se encuentra la clase MS_DB que, a través de ciertos parámetros pertenecientes a la BBDD utilizada (usuario, contraseña, etc.), se ejecuta la conexión y se define una función para poder ejecutar querys en SQL Server:

```python
def execute_query(self, query):
    with self.open_db_connection() as connection:
        result = connection.execute(text(query))
        return result.fetchall()
```

Otro script necesario para poder integrar la API de Gmail y enviar correos automatizados es SenderEmail.py. En este script se encuentra la clase Sender y presenta las siguientes funciones:

- __init__: Inicializa la clase Sender en función de los parámetros de entrada. Configura las credenciales de Gmail y establece el servicio de Gmail.
- __addBody: Añade el cuerpo del mensaje y, opcionalmente, imágenes (tablas de informe) al correo electrónico a través de HTML.
- __addAttachment: Añade archivos adjuntos al correo electrónico.
- sendMail: Compone y envía el correo electrónico.

Esta clase se utiliza para mandar los diferentes correos electrónicos, tantos internos como externos.

También se ha programado un script llamado supervisor.py, en el que se encuentra la clase Supervisor, el objetivo de esta clase es ejecutar una función que realiza un bucle desde las 16:30 hasta las 20:00 verificando continuamente si los ficheros operativos están en la ruta input para ser cargados en la BBDD (CheckDoc).

Los coordinadores deberán dejar los ficheros en la ruta a las 16:30, el bucle trabaja hasta las 20:00 por si hubiese algún problema y no lo puedan entregar antes. En el caso que no esté el fichero a las 20:00, se les envía un correo al Coordinador en cuestión indicando que debe dejar el fichero operativo en la ruta (envioMail).

En el caso de que, una vez cargado el fichero y ejecutados los procedimientos de validación, existan errores en algún fichero operativo, se mandará un correo al Coordinador indicando las validaciones y se volverá a ejecutar el supervisor automáticamente esperando a que se vuelva a dejar en la ruta con las validaciones corregidas, en ese momento se ejecutará la carga.

5.2.2 SCHEDULER.PY: SCRIPT CLAVE PARA LA INTEGRACIÓN DE SISTEMAS

Este script se utiliza para ejecutar todos los procesos de manera automatizada y en el orden deseado. Gracias a este script, no es necesario iniciar manualmente cada proceso, ya que se ejecuta continuamente y llama a los diferentes jobs según el horario configurado. El script utiliza la biblioteca schedule para programar tareas y la biblioteca os para ejecutar comandos del sistema.

Resumen funcionamiento

La ejecución del script es un bucle que revisa si la fecha actual se trata de un día laboral a partir de la función es_dia_laboral. Esta función tiene en cuenta los días festivos y fines de semana.

Si es día laboral, se ejecuta el schedule

```
while True:
    try:
        if es_dia_laboral(datetime.datetime.now()):
            schedule.run_pending()
        time.sleep(1)
    except Exception as e:
        break
```

La siguiente parte del script sería esta:

```
schedule.every().day.at("07:45").do(CargaBI)
schedule.every().day.at("07:55").do(Hist_Maestro_Reporting_Ventas)
schedule.every().day.at("16:30").do(CargaOperativos_Ventas)
```

En el momento que llega la hora, ejecuta las diferentes funciones:

CargaBI

Ejecuta el job_Carga_BI.py 1 7 1: parámetros de entrada del Servicer, ya que este job se encarga de extraer el fichero unificado (campos CCPP y Tributos) del Power BI del Servicer y cargalo en la BBDD.

Esta ejecución se hace a las 07:45 de la mañana y tarda 6 minutos.

```
def CargaBI():
    ejecucion=os.system(rf" python job_Carga_BI.py 1 7 1")
    if ejecucion==1:
        raise Exception
```

Si esta función falla por alguna razón, saltará la excepción y el Schedule se detiene.

Hist_Maestro_Reporting_Ventas

Como su propio nombre indica, se encarga de historificar los datos de las tres tablas (fichero unificado y ficheros operativos), actualizar los dos Maestros y extraer los tres ficheros de reporte (operativos y unificado actualizados).

Esta ejecución se realiza a las 07:55 de la mañana y tarda 8 minutos.

```
def Hist_Maestro_Reporting_Ventas():
    os.system("job_Historificacion 1 1 1")
    os.system("job_Historificacion 1 2 1")
    os.system("job_Historificacion 1 7 1")
    fecha_actual = datetime.datetime.now()
    fecha_formateada = fecha_actual.strftime("%Y%m%d")
    os.system(f"job_Maestro 1 1 1 {fecha_formateada}")
    os.system(f"job_Maestro 1 2 1 {fecha_formateada}")
    os.system(rf" python job_Reporting.py 1 1 1")
    os.system(rf" python job_Reporting.py 1 2 1")
    os.system(rf" python job_Reporting.py 1 7 1")
    os.system(f" python job_BI_Nforce.py {fecha_formateada}")
```

Se puede observar que los parámetros de entrada hacen referencia siempre al id_cliente=1 (Servicer), id_servicio=1,2,7 (Tributos, CCPP, Unificado), id_hito=1 (ventas)

CargaOperativos_Ventas

Ese mismo día a las 16:30, se ejecutará esta función, que se encarga de ejecutar los job de carga para Tributos y CCPP, este job lleva integrado el supervisor para estos parámetros, por lo que espera en bucle hasta que los ficheros operativos se encuentren en sus respectivas rutas. Al igual que la carga del fichero unificado, si esta falla, saltará la excepción y detiene el proceso.

```
def CargaOperativos_Ventas():
    ejecucion1=os.system(rf" python job_Carga_Operativos.py 1 1 1")
    ejecucion2=os.system(rf" python job_Carga_Operativos.py 1 2 1")
    if ejecucion1 == 1 or ejecucion2 == 1:
        raise Exception
```

5.3 *PRUEBAS Y AJUSTES (LOG)*

A la hora de realizar las pruebas, se creó un procedimiento en SQL llamado DQ.LOG_EJECUCION. Este procedimiento se encarga de insertar registros en la tabla DQ.LOG, permitiendo así llevar un control detallado de las acciones que realiza el sistema en cada punto de ejecución. Al incorporar esta funcionalidad, se facilita la identificación y corrección de errores, ya que se puede acceder directamente a la sección del código donde ocurrió el fallo.

	FCH_DATOS	LEVEL	ID_CLIENTE	ID_SERVICIO	ID_HITO	PROCESO	FASE	ERROR
1	2024-06-27 16:14:11.000	INFO	1	2	1	ENVIO MAIL AUTOM...	6.ADICION BODY	
2	2024-06-27 16:14:11.000	INFO	1	2	1	ENVIO MAIL AUTOM...	8.ENVIO MAIL	
3	2024-06-27 16:14:10.000	INFO	1	2	1	ENVIO MAIL AUTOM...	3.INICIALIZACION SERVICIO	
4	2024-06-27 16:14:10.000	INFO	1	2	1	ENVIO MAIL AUTOM...	4.INICIALIZACION MENSAJE	
5	2024-06-27 16:14:10.000	INFO	1	2	1	ENVIO MAIL AUTOM...	5.INICIALIZACION VARIABLES M...	
6	2024-06-27 16:14:09.000	INFO	1	2	1	ENVIO MAIL AUTOM...	2.OBTENCION CREDENCIALES	
7	2024-06-27 16:14:08.000	INFO	1	2	1	CARGA OPERATIVOS	98.NO EXISTEN VALIDACIONES ...	
8	2024-06-27 16:14:08.000	INFO	1	2	1	ENVIO MAIL AUTOM...	1.INICIALIZACION SENDER	
9	2024-06-27 16:14:07.000	INFO	1	2	1	CARGA OPERATIVOS	6.2.COMPROBACION VALIDACIO...	

Ilustración 18: Registros ejemplo tabla DQ.LOG en SQL Server.

La clase EventLogger en Python es la encargada de ejecutar este procedimiento SQL desde los distintos scripts. Además de insertar registros en la base de datos, EventLogger también imprime mensajes en la terminal (print). De este modo, se puede visualizar en tiempo real qué procesos se están ejecutando, lo cual es muy útil para monitorear y depurar el sistema.

```python
class EventLogger():
    def __init__(self):
        config_obj = configparser.ConfigParser()
        config_obj.read("Config.ini")

        event_logger = config_obj["eventlogger"]
        dbparam = config_obj["sqlserver"]

        self.myuserid=dbparam["user"]
        self.mypassword=dbparam["password"]
        self.mydatabase=dbparam["db"]
        self.myserver=dbparam["host"]
        self.connectionString = f'DRIVER={{SQL Server}};SERVER={self.myserver};DATABASE={self.mydatabase};UID={self.myuserid};PWD={self.mypassword}'

        self.log_table = event_logger["log_table"]

    #Utiliza la conexión propia del script GeneradorFicheros.py
    def writeLog(self, level, id_cliente, id_servicio, id_hito, proceso , mensaje, error):
        err=error.replace("'","")
        mensaje=mensaje.replace("'","")
        query = """INSERT INTO
        {} VALUES ('{}','{}','{}',{}, {}, {}, '{}', '{}', '{}')
        """.format(self.log_table, datetime.now().strftime('%Y-%m-%d %H:%M:%S'),
                   str(self.myuserid), str(level), id_cliente, id_servicio, id_hito, str(proceso), str(mensaje), str(err))
        print(query)
        self.conn = sqlserver.connect(self.connectionString)
        self.cur = self.conn.cursor()
        self.cur.execute(query)
        self.cur.commit()
        self.cur.close()
        self.conn.close()
```

En la terminal de Python, se visualiza lo siguiente (ejemplo):

```
('2024-07-01 12:56:03','sqlserver','INFO',1, 1, 1, 'CARGA OPERATIVOS', '1.INICIO EJECUCION', '')
('2024-07-01 12:56:04','sqlserver','INFO',1, 1, 1, 'CARGA OPERATIVOS', '2.CONEXION A BASE DE DATOS', '')
('2024-07-01 12:56:05','sqlserver','INFO',1, 1, 1, 'CARGA OPERATIVOS', '3.EJECUCION CARGA', '')
('2024-07-01 12:56:05','sqlserver','INFO',1, 1, 1, 'CARGA STG', '1.INICIO CLASE', '')
('2024-07-01 12:56:05','sqlserver','INFO',1, 1, 1, 'CARGA STG', '2.CONFIGURACION DATOS', '')
('2024-07-01 12:56:06','sqlserver','INFO',1, 1, 1, 'CARGA STG', '3.INICIO EJECUTOR', '')
('2024-07-01 12:56:06','sqlserver','INFO',1, 1, 1, 'CARGA STG', '4.OBTENCION FICHEROS', '')
('2024-07-01 12:56:08','sqlserver','INFO',1, 1, 1, 'CARGA STG', 'CARGAR|CARGA FICHERO: Pte_AdmonPat_NFQ_Tributos - Registros', '')
```

Ilustración 20: Ejemplo ejecuciones carga operativos. Terminal de Python.

5.4 GENERACIÓN DE INFORMES

5.4.1 INFORMES DE CONSECUCIÓN

Los informes de consecución se generan de manera automatizada en una hoja auxiliar del fichero reportado al cliente a través de los datos en los registros.

Este informe sirve para llevar un seguimiento diario del estado de las gestiones de las ventas y se hace llegar al Servicer y los Coordinadores a través de correo electrónico. El informe se compone de las siguientes tablas:

Ventas/Postventas

Esta tabla muestra las gestiones OK y KO de cada departamento de EG, agrupando las ventas por la cartera:

- Total: Inmuebles que se venden en el mes actual.
- Pipeline: Inmuebles que se venden en el mes actual, pero aún no ha llegado la fecha de venta.
- Vendidos: Inmuebles ya vendidos en el mes actual.

Pipeline (posicionam)

Año	2024	Total		Tributos		CCPP		Consecución	
Mes	7	Activos	Importe	OK	KO	OK	KO	Tributos	CCPP
	Santander	438	45.808.018,00 €	248	190	371	67	57%	85%
	Sareb	0	0,00 €	0	0	0	0	0%	0%
	TP	169	12.425.150,00 €	121	47	66	103	72%	39%
	Total	**607**	**58.233.168,00 €**	**369**	237	**437**	170	**61%**	**72%**

Año	2024	PipeLine		Tributos		CCPP		Consecución	
Mes	7	Activos	Importe	OK	KO	OK	KO	Tributos	CCPP
	Santander	438	45.808.018,00 €	248	190	371	67	57%	85%
	Sareb	0	0,00 €	0	0	0	0	0%	0%
	TP	169	12.425.150,00 €	121	47	66	103	72%	39%
	Total	**607**	**58.233.168,00 €**	**369**	237	**437**	170	**61%**	**72%**

Año	2024	Vendidos		Tributos		CCPP		Consecución	
Mes	7	Activos	Importe	OK	KO	OK	KO	Tributos	CCPP
	Santander	0	0,00 €	0	0	0	0	0%	0%
	Sareb	0	0,00 €	0	0	0	0	0%	0%
	TP	0	0,00 €	0	0	0	0	0%	0%
	Total	**0**	**0,00 €**	**0**	0	**0**	0		

Ilustración 19: Informe Junio a fecha 02/07/2024.

Al ser datos del día 02/07/2024, se puede observar que aún no hay inmuebles vendidos con fecha de venta el mes de julio, el total es igual al pipeline. Empiezan el mes con un 61% y un 72% de gestiones realizadas en Tributos y CCPP, respectivamente. Esto quiere decir que los meses anteriores se han ido trabajando ventas posteriores.

El objetivo a fin de mes es que la Consecución total de cada departamento sea 95%. Los Coordinadores son los principales responsables de esto y por ello se manda este informe a diario, para hacer un seguimiento y que el Servicer pueda ver el avance diario.

El siguiente informe tiene el mismo enfoque que el anterior, pero se muestra el total de inmuebles que se venden o se han vendido en el año actual, esta vez sin agrupar por carteras. Cabe destacar que el número de inmuebles cambia día a día en función de las altas y bajas en el fichero unificado, no es un histórico. Normalmente, de media se venden unos 900

inmuebles al mes y poco a poco se van dando de baja en el fichero según el criterio del Servicer. Al igual que se dan de alta las nuevas ventas que van saliendo.

		Total				PipeLine				Vendidos			
Año	2024	Activos	Importe	Tributos	CCPP	Activos	Importe	Tributos	CCPP	Activos	Importe	Tributos	CCPP
Enero	1	161	15.019.390,00 €	96%	99%	0	0,00 €			161	15.019.390,00 €	96%	99%
Febrero	2	423	73.683.893,00 €	97%	100%	0	0,00 €			423	73.683.893,00 €	97%	100%
Marzo	3	372	40.395.148,00 €	92%	98%	0	0,00 €			372	40.395.148,00 €	92%	98%
Abril	4	437	37.098.778,00 €	96%	96%	0	0,00 €			437	37.098.778,00 €	96%	96%
Mayo	5	557	46.564.395,00 €	98%	95%	0	0,00 €			557	46.564.395,00 €	98%	95%
Junio	6	748	79.614.932,00 €	99%	97%	0	0,00 €			748	79.614.932,00 €	99%	97%
Julio	7	607	58.233.168,00 €	61%	72%	607	58.233.168,00 €	61%	72%	0	0,00 €		
Agosto	8	191	13.169.534,00 €	52%	3%	191	13.169.534,00 €	52%	3%	0	0,00 €		
Septiembre	9	51	12.602.081,00 €	57%	24%	51	12.602.081,00 €	57%	24%	0	0,00 €		
Octubre	10	175	11.611.999,00 €	13%	10%	175	11.611.999,00 €	13%	10%	0	0,00 €		
Noviembre	11	11	1.696.237,00 €	45%	36%	11	1.696.237,00 €	45%	36%	0	0,00 €		
Diciembre	12	83	13.278.230,00 €	58%	48%	83	13.278.230,00 €	58%	48%	0	0,00 €		

Ilustración 20: Informe Anual a fecha 02/07/2024

El siguiente informe muestra la consecución de ventas por departamento de cada día del mes, este enfoque permite a los coordinadores hacer foco en ciertos días para equilibrar la consecución total.

Al ser inicio de mes, aún no se ha registrado ninguna venta y tan solo se tienen en cuenta las ventas agendadas de cada día, mostrando al final cuántas de esas ventas tienen la gestión OK.

El número de ventas agendadas total hace referencia al número total de ventas en el primer informe de consecución, así como al número de ventas del mes de Julio en el informe Anual (607).

Año	2024
Mes	7

Dia	Agendadas	Vendidas	Consecución Agendadas TRIB	Consecución Agendadas CCPP	Consecución Vendidas TRIB	Consecución Vendidas CCPP	Tributos OK	CCPP OK
1	12	0	75,00%	75,00%	0,00%	0,00%	9	9
2	14	0	92,86%	64,29%	0,00%	0,00%	13	9
3	11	0	100,00%	63,64%	0,00%	0,00%	11	7
4	76	0	97,37%	93,42%	0,00%	0,00%	74	71
5	26	0	88,46%	65,38%	0,00%	0,00%	23	17
6	0	0	0,00%	0,00%	0,00%	0,00%	0	0
7	0	0	0,00%	0,00%	0,00%	0,00%	0	0
8	3	0	100,00%	100,00%	0,00%	0,00%	3	3
9	17	0	88,24%	88,24%	0,00%	0,00%	15	15
10	7	0	100,00%	57,14%	0,00%	0,00%	7	4
11	16	0	81,25%	62,50%	0,00%	0,00%	13	10
12	37	0	100,00%	86,49%	0,00%	0,00%	37	32
13	0	0	0,00%	0,00%	0,00%	0,00%	0	0
14	0	0	0,00%	0,00%	0,00%	0,00%	0	0
15	21	0	80,95%	66,67%	0,00%	0,00%	17	14
16	8	0	100,00%	50,00%	0,00%	0,00%	8	4
17	8	0	87,50%	75,00%	0,00%	0,00%	7	6
18	9	0	88,89%	22,22%	0,00%	0,00%	8	2
19	12	0	66,67%	25,00%	0,00%	0,00%	8	3
20	3	0	66,67%	0,00%	0,00%	0,00%	2	0
21	0	0	0,00%	0,00%	0,00%	0,00%	0	0
22	174	0	5,75%	95,40%	0,00%	0,00%	10	166
23	4	0	50,00%	75,00%	0,00%	0,00%	2	3
24	14	0	64,29%	50,00%	0,00%	0,00%	9	7
25	15	0	80,00%	73,33%	0,00%	0,00%	12	11
26	24	0	79,17%	50,00%	0,00%	0,00%	19	12
27	0	0	0,00%	0,00%	0,00%	0,00%	0	0
28	0	0	0,00%	0,00%	0,00%	0,00%	0	0
29	19	0	78,95%	68,42%	0,00%	0,00%	15	13
30	9	0	55,56%	22,22%	0,00%	0,00%	5	2
31	68	0	61,76%	25,00%	0,00%	0,00%	42	17

Ilustración 21: Informe consecución diaria a fecha 02/07/2024

5.4.2 INFORME DE FACTURACIÓN

Todos los días, el ejecutar el Job_BI_Nforce 1 7 1 @fecha_actual, se ejecutan los procedimientos para actualizar todos los informes internos y envía un correo de verificación mostrando un pequeño análisis de los ficheros mostrado en la Ilustración 18. Adjunto a este correo envía una imagen que muestra un informe de la facturación del mes actual, mostrando un desglose de lo que se está reportando al Servicer (ventas del mes actual), así como la facturación de otros tipos de ventas.

Haciendo referencia al informe del 02/07/2024, para observar que toda cuadra, se muestra la facturación del mes de Julio a esa fecha.

Total Ventas - Reporte AAM		607	
Facturación		**Tributos**	**CCPP**
Reportado y facturado meses anteriores		204	334
Reportado a facturar		4	12
Reportado: Gestiones OK no facturadas	Gestión Anterior	101	28
	No Gestión	45	49
	Sin Fecha Fact en BBDD	15	14
	Reportado OK	**369**	**437**
	Ventas a facturar mes en curso	**42**	**15**
No reportado a facturar	Ventas Normales (Fec.pos otro mes)	38	3
	Ventas Normales (Bajas)	0	0
	Ventas Especiales	0	0
	Postventas fuera de fichero	0	0

Ilustración 22: Informe Facturación Julio a fecha 02/07/2024

Se puede observar que de las 607 ventas reportadas al Servicer en el informe, se desglosa la facturación de las ventas que están OK, 369 para Tributos y 437 para CCPP, que coincide con lo mostrado en la Ilustración 21.

A continuación, se describe brevemente lo que describe cada fila del informe:

- Reportado y facturado meses anteriores: las ventas reportadas quieren decir que entran dentro de las 607 ventas con fecha de venta el mes actual, en este caso julio. Al ser facturado meses anteriores, quiere decir que son ventas que se gestionaron en meses anteriores y ya se mandaron a facturar en su día.
- Reportado a facturar: son ventas reportadas y que se han gestionado durante el mes actual.
- Reportado: Gestiones OK no facturadas.
    - Gestión Anterior: son ventas que salen a revisar (estaban de baja y se vuelven a dar de alta en el fichero unificado) y se mantiene la gestión anterior (OK), por lo que ya se han facturado previamente.
    - No Gestión: carteras que no se gestionan y por lo tanto son OK.
    - Sin Fecha Fact en BBDD: quiere decir que son ventas que están OK, pero no se tiene el registro de cuando se facturaron. Esto se debe a que la BBDD está alimentada con datos de enero de 2024 hasta hoy.

- No reportado a facturar: en este caso, son ventas que no se venden en julio, pero se han gestionado en julio. Las ventas se facturan en el momento que se termina la gestión, pero la fecha de venta puede ser de meses anteriores o posteriores.
    - Ventas Normales: ventas presentes en el fichero unificado.
    - Ventas Normales (Bajas): ventas que se dieron de baja en el fichero unificado, pero se facturan porque la gestión se realizó antes de que se diera de baja.
    - Ventas Especiales: explicado previamente.
    - Postventas fuera de fichero: explicado previamente.

5.4.3 INFORMES DE JUSTIFICACIÓN AL SERVICER

Debido a que el objetivo de consecución se mide en base a las ventas que se venden en el mes, es importante poder justificar al Servicer ciertas gestiones no realizas por razones excepcionales.

Como se ha comentado anteriormente, el cliente modifica día a día los datos del fichero unificado que hacen referencia al inmueble, el problema radica en que entre esos campos está la fecha de venta.

La actualización de este campo es lo que provoca que el número de ventas del reporte sea dinámico y cambie día a día. Se identificaron dos casuísticas problemáticas:

Ventas acumuladas a fin de mes:

El Servicer tiende a acumular muchas ventas hacia el final del mes. Esto se debe a que intentan maximizar el número de inmuebles vendidos a medida que el mes avanza. Sin embargo, llega un punto en el que seguir añadiendo ventas al final del mes resulta contraproducente, ya que no hay tiempo suficiente para gestionarlas adecuadamente, acumulándose demasiado trabajo en los últimos días.

Para ello se va a guardar un registro acumulado de las ventas que adelantan a menos de una semana, es decir, ventas que tienen una fecha de venta y la cambian a los próximos cinco días, lo cual justifica que no se realice la gestión. Así como cierta información del acumulado

de las ventas en los últimos cinco días del mes, registrando el número de ventas que entran en esta franja temporal desde el primer día del mes, diferenciando las que son adelantadas o retrasadas. También se muestra la respectiva consecución, de manera que se pueda ver si las ventas que son adelantadas y retrasadas a los últimos cinco días del mes estaban o no gestionadas en el momento del cambio.

Cambios en las fechas de posicionamiento:

Saliendo de la casuística anterior, los cambios en las fechas de venta pueden afectar negativamente la consecución mensual. Esto se debe a que diariamente hay activos que se eliminan y otros que se añaden al reporte.

Si un gestor está trabajando en una venta y de repente la fecha se cambia a otro mes, esa gestión se pierde para la consecución (aunque sí se facturaría). También puede ocurrir que ventas que originalmente estaban programadas para venderse otro mes, se adelanten al mes actual y no hayan sido gestionadas previamente, afectando así la consecución debido al poco tiempo que tendrían los gestores para realizar la gestión.

Para abordar este problema, se mantiene un registro de los activos que entran y salen del reporte de consecución mensual, así como de la consecución de estas ventas.

Nº Ventas	01/07/2024	472	Consecución Tributos	Consecución CCPP
Salidas de BI con mes fec.posicionamiento actual		8	50,00%	100,00%
Entradas en BI con mes fec.posicionamiento actual		10	90,00%	20,00%
Salidas: cambios de mes fec.posicionamiento actual a otro mes		7	85,71%	71,43%
Entradas: cambios de mes fec.posicionamiento de otro mes al actual		140	94,29%	83,57%
Nº Ventas	**02/07/2024**	**607**		

Nº Ventas en los últimos 5 días del mes actual	02/07/2024	135	Consecución Tributos	Consecución CCPP
Cambios de fec.posicionamiento adelantada a los últimos 5 días del mes actual		9	33,33%	33,33%
Cambios de fec.posicionamiento retrasada a los últimos 5 días del mes actual		14	100,00%	78,57%

Nº Ventas adelantadas a menos de una semana	14

Ilustración 23: Informe Justificación Julio a fecha 02/07/2024

Se puede observar en el informe que el día 1 del mes había 472 ventas y, al día siguiente 607, lo cual impacta de primeras. A continuación, se describe una breve definición y comentario de cada fila presente en el informe, a modo de ejemplo:

NOTA: Cabe destacar que este informe es acumulado y se van sumando los registros a medida que el mes avanza, en este caso al ser el día 2 del mes solo cuenta los cambios entre el día 1 y 2.

- Salidas de BI con mes fec.posicionamiento actual (8 ventas; Consecución Tributos 50%; Consecución CCPP 100%): Son ventas que tienen la fecha de venta en julio y se dan de baja (razones externas) en el propio mes. En este caso a CCPP le perjudica, porque son 8 ventas gestionadas que pierde de cara a la consecución del reporte.
- Entradas de BI con mes fec.posicionamiento actual (10; 90%; 20%): Son ventas que se dan de alta en el fichero unificado durante el mes de julio y tienen fecha de venta el mes de julio. Esto simplemente son altas, por lo que son nuevas ventas que se dan según avanza el mes, pero que se venden en el propio mes.
- Salidas: cambios de mes fec.posicionamiento actual a otro mes (7; 85,71%; 71,43%): Son ventas que el día 1 tenían fecha de venta en julio y para el día 2 la cambian de mes. En este caso perjudica a ambos, perdiendo gestiones realizadas para el reporte.
- Entradas: cambios de mes fec.posicionamiento de otro mes al actual (140; 94,29%; 83,57%): Son ventas que el día 1 de julio tenían la fecha de venta en un mes diferente a julio y para el día 2 lo tienen en julio. Se puede ver que se incluyen 140 ventas, lo cual llama la atención, pero la consecución es alta y no debería de causar problemas.

El resto de información presente en el informe ha sido explicado anteriormente.

Capítulo 6. RESULTADOS Y ANÁLISIS

6.1 EVALUACIÓN DE RESULTADOS

Tras un largo proceso de creación e implementación de la base de datos centralizada y la automatización de los procesos, se han generado una serie de resultados positivos medibles en términos de eficiencia operativa y calidad del servicio prestado. Además, este proyecto es el comienzo de un largo camino de optimización y automatización, ya que se puede aplicar a todos los servicios y clientes presentes en la empresa EG.

En primer lugar, la automatización de tareas, especialmente la carga y validación de datos, ha permitido reducir significativamente el tiempo necesario para completar estas actividades. Anteriormente, la carga manual de los ficheros y la revisión de errores consumían aproximadamente el 20% del tiempo de los coordinadores (1,5 h/día). Con la implementación de scripts en Python y el uso de procedimientos almacenados en SQL Server, este tiempo se ha reducido a un 1% (5 min/día), ya que la única tarea será corregir los errores presentes en el fichero a última hora de la tarde, esto permite que los coordinadores se enfoquen en actividades de mayor valor añadido y puedan dedicar más tiempo a la gestión de los demás servicios y no solo estar centrados en ventas.

Los scripts de carga y las validaciones implementadas en los procedimientos almacenados de la BBDD han reducido drásticamente los errores de datos. Antes de la automatización, la tasa de errores en los datos ingresados en el fichero era muy elevada, ni siquiera cuadraban las oportunidades de venta entre un fichero y otro. Con el nuevo proceso de carga y extracción, esta tasa se ha reducido a un 0%, ya que en el momento que no cuadra algo por un error humano, el propio proceso mandará un correo para que se revise la BBDD y el problema se solvente cuanto antes (en dos meses con el sistema implantado, nunca ha ocurrido).

El registro detallado de todas las gestiones en la BBDD ha mejorado la transparencia y trazabilidad de las operaciones. Ahora es posible realizar auditorías internas de manera más eficiente, identificando rápidamente cualquier discrepancia y corrigiéndola en tiempo real. Esto también ha fortalecido la confianza entre la Empresa Gestora (EG) y el Servicer, ya que se pueden justificar las gestiones realizadas con datos precisos y actualizados, así como justificar las no gestionadas por cuestiones de tiempo de gestión.

Los coordinadores además podrán realizar las tareas de análisis de manera mucho más sencilla, ya que tendrán acceso a un PowerBI donde podrán hacer uso de todas las gráficas y tablas mostradas según los datos presentes en la BBDD. Estas tablas para mostrar dicha información de valor, se crean en el procedimiento almacenado de generación de informes de la BBDD.

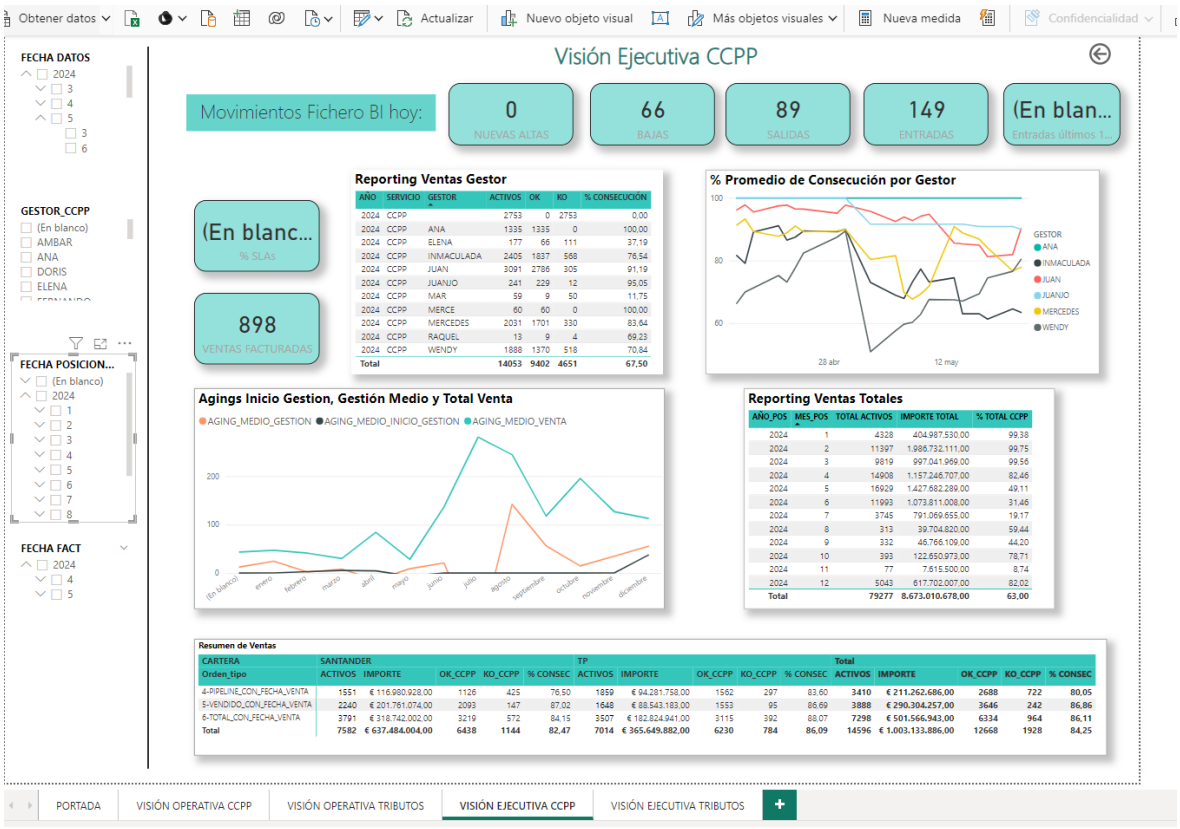

Ilustración 24: Visión ejecutiva CCPP en fecha 03/07/2024

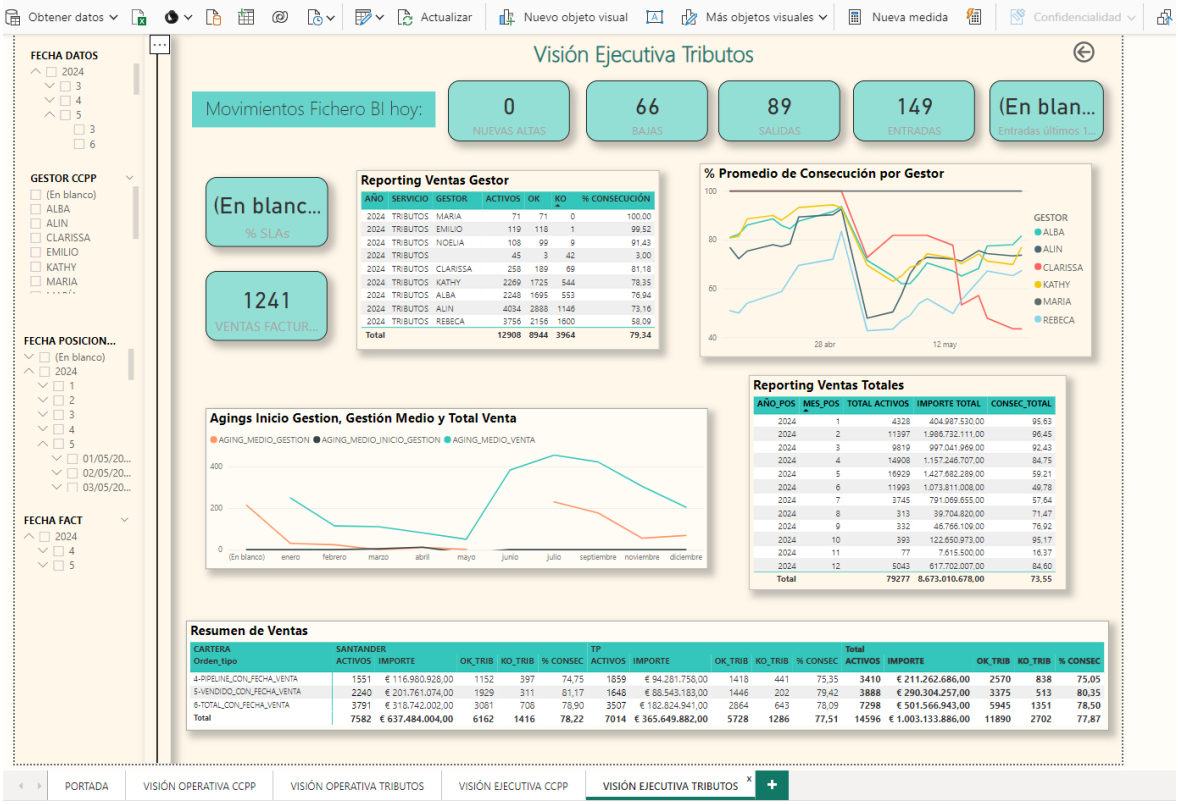

Ilustración 25: Visión ejecutiva Tributos en fecha 03/07/2024

Como se puede observar, estas hojas del BI muestran la visión ejecutiva de ambos departamentos (CCPP y Tributos). Los coordinadores pueden filtrar por fechas de venta, de facturación y por gestor, de esta manera puede analizar el trabajo de cada uno y lo que facturan de manera individualizada. Así como los tiempos medios de gestión y la consecución de las ventas. También muestra los movimientos en el fichero unificado, indicando el número de altas (nuevas ventas), bajas (ventas dadas de baja), salidas (cambio en la fecha de venta) y entradas (cambio en la fecha de venta) en el mes actual del fichero.

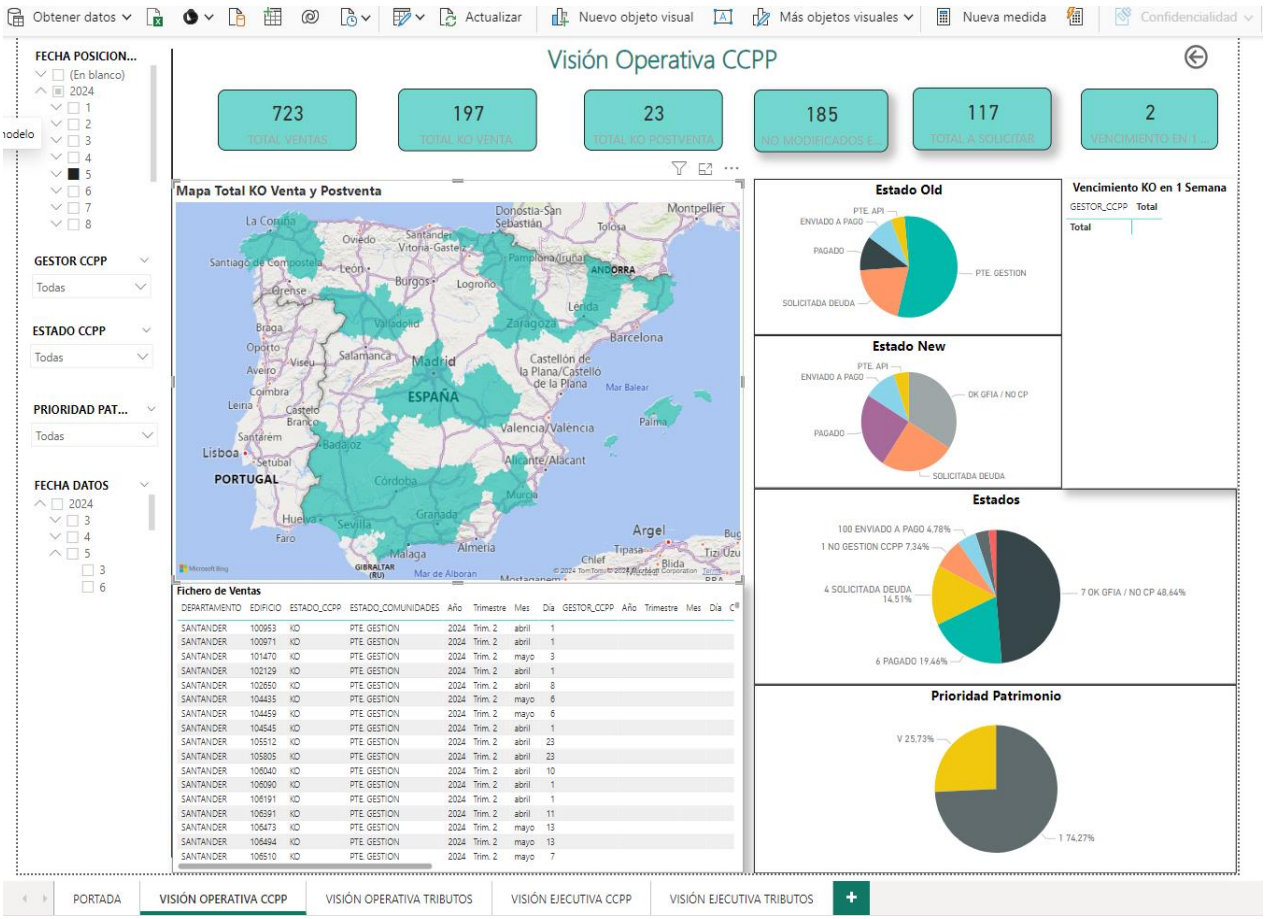

Ilustración 26: Visión operativa CCPP en fecha 03/07/2024

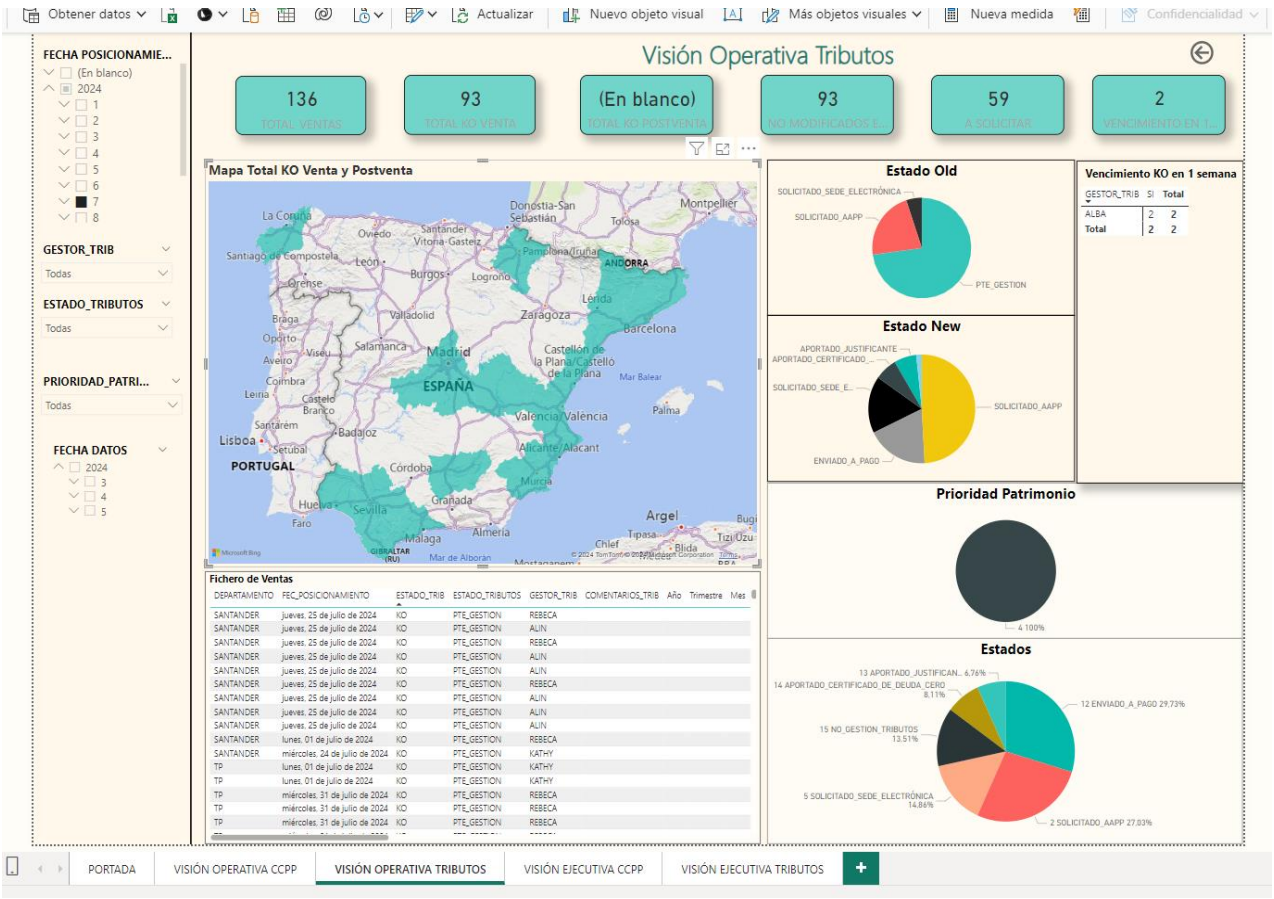

Ilustración 27: Visión operativa Tributos en fecha 03/07/2024

También se muestra una visión operativa para ambos departamentos, en ella se puede analizar el avance de las ventas filtrando por gestor, fecha de venta, estados de gestión y prioridad. Las gráficas de estado old y estado new muestra los cambios de un día para otro de los estados. También se muestran las ventas que vencen en menos de una semana y son KO, las cuales tienen que ser prioritarias frente al resto. El mapa sirve para identificar las localidades donde hay más volumen de activos, es importante sobre todo para Tributos, ya que trabajan en contacto con las administraciones públicas y diputaciones.

6.2 CONCLUSIONES Y PRÓXIMOS PASOS

Este proyecto ha permitido a EG operar de una manera mucho más sólida, eficiente y profesional. Esto no solo mejora el servicio ofrecido, sino que también abre la puerta para

atraer nuevos clientes. El tiempo ahorrado en la gestión de datos y la reducción de errores ha optimizado el uso de los recursos, beneficiando tanto a la empresa como al Servicer, que ahora recibe información más precisa y rápida.

Aunque la implementación ha comenzado con el área de ventas, el objetivo es integrar todos los diferentes datos (ficheros) manejados en los distintos servicios de EG en la BBDD. Esto no sólo optimizará el trabajo diario, sino que también permitirá implementar lógicas más complejas y personalizadas, sentando las bases para una mejora continua.

El próximo proyecto por implementar será conectar la BBDD con la de Tamías, una herramienta de gestión de inmuebles utilizada a diario por los gestores y el Servicer. Se pretende optimizar los datos y sacar información precisa de Tamías para crear un fichero de control de cartera en Tributos y CCPP. Esto permitirá llevar un seguimiento de la deuda de todos los activos gestionados en cartera, mostrando información de interés y alertas/validaciones según el departamento.

EG ofrece muchos más servicios como Gestión Catastral, Requerimientos, Formalización, Facturación, entre otros. A largo plazo se pretende incluir toda la información en la BBDD para optimizar los recursos y llevar un seguimiento preciso y eficiente.

Se pretende ampliar la red de clientes presentando los proyectos con el sistema implementado y que los gestores sigan trabajando con un solo fichero con toda la información, el objetivo es facilitar el trabajo y eficientes los servicios.

Capítulo 7. BIBLIOGRAFÍA

1. Bendezu, K. “Stored Procedures in SQL Server: An Overview”. SQL Data Insights, April 2021. http://sqldatainsights.com/2021/04/stored-procedures-in-sql-server.

2. Herrero Alcántara, T. “Working with OpenPyxl for Excel Automation in Python”. Think Big, November 2019. http://blogthinkbig.com/openpyxl-excel-automation-python.

3. Loeffler, B. “Understanding Data Staging, ODS, and Data Warehousing”, Microsoft Technet Magazine, June 2020. https://technet.microsoft.com/en-us/magazine/data-staging-ods-data-warehousing-june-2020.

4. Vlassis, N.A.; Papakonstantinou, G.; Tsanakas, P. Dynamic sensory probabilistic maps for mobile robot localization. Source: Proceedings. 1998 IEEE / RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190) New York, NY, USA: IEEE, 1998.p.718-23 vol.2 of 3 vol. xliv+2010 pp. 11.

5. Bendezu, K. “Advanced Techniques in SQL Server Stored Procedures”. SQL Data Insights, August 2022. http://sqldatainsights.com/2022/08/advanced-techniques-in-sql-server-stored-procedures.

6. Herrero Alcántara, T. “Web Scraping with Selenium in Python”. Think Big, September 2020. http://blogthinkbig.com/web-scraping-selenium-python.

7. Loeffler, B. “Data Lifecycle: From Staging to Historical Databases”, Microsoft Technet Magazine, March 2021. https://technet.microsoft.com/en-us/magazine/data-lifecycle-staging-historical-databases-mar-2021.

8. Bendezu, K. “Efficient Data Processing with Pandas in Python”. SQL Data Insights, December 2021. http://sqldatainsights.com/2021/12/efficient-data-processing-pandas-python.

9. Herrero Alcántara, T. “Automating Data Workflows with Python and Pandas”. Think Big, February 2022. http://blogthinkbig.com/automating-data-workflows-python-pandas.

ANEXOS

ANEXO 1: Procedimientos almacenados SQL Server:

HOM.HOMOGENEIZACION

```
USE [AAM_SERVICIO_CCPP_TRIBUTOS]
GO
/****** Object:  StoredProcedure [HOM].[HOMOGENEIZACION]    Script Date: 18/06/2024
16:40:22 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
/*********************************************/
-- Autor: Ángel Guzmán
-- Fecha: 20240213
-- Descripción: Proceso que homogeiniza las tablas de entrada de STG en base a una
tabla paramétrica
--             Inputs:
--                    STG.AAM_BI_VENTAS
--                    STG.AAM_OPE_VENTAS
--             Paramétrica:
--                    COD.PARAM_HOMOGENEIZACION_CAMPOS
--             Outputs:
--                    HOM.AAM_BI_VENTAS
--                    HOM.AAM_OPE_VENTAS
--
-- Versiones:
--             V1: Versión inicial
/*********************************************/


ALTER PROCEDURE [HOM].[HOMOGENEIZACION] (
    @Cliente int,
       @Servicio int,
       @Hito int
)
AS
BEGIN
    BEGIN TRY

              --#################################
              --#    0.DECLARACION DE VARIABLES   #
              --#################################
        EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION', 'INICIO
EJECUCION', NULL

        DECLARE @totalcolumnas INT = 0,
                @contador INT = 1,
                @message NVARCHAR(500),
                @fecha DATETIME = GETDATE(),
                @error NVARCHAR(500),
```

```
		@SqlUpdate NVARCHAR(MAX),
		@SqlInsert NVARCHAR(MAX),
				@SqlInsert1 NVARCHAR(MAX),
				@SqlTruncate NVARCHAR(MAX),
				@ID_Unico NVARCHAR(255),
				@Tabla_Origen NVARCHAR(100),
				@Tabla_Destino NVARCHAR(100),
				@tablename NVARCHAR(100) = NULL,
				@columnname NVARCHAR(100) = NULL,
				@transformation NVARCHAR(MAX) = NULL,
				@error_msg NVARCHAR(255) = NULL,
				@query NVARCHAR(MAX) = NULL,
				@dotPosition1 INT,
				@schemaOrigen NVARCHAR(20),
				@tableOrigen NVARCHAR(50),
				@campo_stg NVARCHAR(255),
				@schemaDestino NVARCHAR(MAX),
				@tableDestino NVARCHAR(MAX),
				@concatSQL NVARCHAR(MAX),
				@sql NVARCHAR(MAX)


	--########################################
	--# 1.OBTENCIÓN TABLAS ORIGEN Y DESTINO   #
	--########################################
	EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION',
'1.OBTENCION TABLAS ORIGEN Y DESTINO', NULL

	--1.1. Declaración de cursor
	EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION',
'1.1.DECLARACION DE CURSOR', NULL
	IF CURSOR_STATUS('global','tablas_origen_destino')>=-1
	BEGIN
		DEALLOCATE tablas_origen_destino
	END

	DECLARE tablas_origen_destino CURSOR FOR
	SELECT DISTINCT
		TABLA_ORIGEN,
		TABLA_DESTINO
	FROM COD.PARAM_HOMOGENEIZACION_CAMPOS
	WHERE ID_CLIENTE = @Cliente AND ID_SERVICIO = @Servicio AND ID_HITO =
@Hito

	OPEN tablas_origen_destino;
	FETCH NEXT FROM tablas_origen_destino INTO @Tabla_Origen,
@Tabla_Destino;

	--########################
	--#    2.HOMOGENEIZACION    #
	--########################
	EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION',
'2.INICIO PROCESO INTERNO HOMOGENEIZACION', NULL
	WHILE @@FETCH_STATUS = 0
	BEGIN
		--2.1. Comprobación de que las tablas existen
```

```
                    IF @Tabla_Origen IS NULL OR @Tabla_Destino IS NULL
                    BEGIN
                           EXEC DQ.LOG 'ERROR', @Cliente,@Servicio,@Hito,
'HOMOGENEIZACION', 'ERROR', '2.1.NO SE ENCONTRARON LAS TABLAS DE ORIGEN Y/O DESTINO';
                           EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
'HOMOGENEIZACION', '99.FIN';
                           CLOSE tablas_origen_destino;
                           DEALLOCATE tablas_origen_destino;
                           RETURN;
                    END;
                    --2.2.Borrado tablas temporales necesarias
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION',
'2.2.BORRADO TABLAS TEMPORALES', NULL
                    DROP TABLE IF EXISTS #TMP_RELACION_CAMPOS
                    DROP TABLE IF EXISTS ##TMP_ORIGEN

                    SET @SqlTruncate = 'TRUNCATE TABLE ' + @Tabla_Destino + ';'
                    EXEC sp_executesql @SqlTruncate

                    DECLARE @sqlUpdateConcat NVARCHAR(MAX),
                                  @sqlUpdateEstado NVARCHAR(MAX),
                                  @columnExistsEstado INT,
                                  @sqlCheckColumnEstado NVARCHAR(MAX),
                                  @columns NVARCHAR(MAX),
                                  @checksum NVARCHAR(MAX) = NULL

                    --2.3. Separación nombre y esquema
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION',
'2.3.SEPARACION NOMBRE Y ESQUEMA', NULL
                    SET @dotPosition1= CHARINDEX('.', @Tabla_Origen);
                    SET @schemaOrigen = LEFT(@Tabla_Origen, @dotPosition1 - 1);
                    SET @tableOrigen= RIGHT(@Tabla_Origen, LEN(@Tabla_Origen) -
@dotPosition1);
                    SET @schemaDestino = PARSENAME(@Tabla_Destino, 2); -- El segundo
fragmento es el esquema
                    SET @tableDestino = PARSENAME(@Tabla_Destino, 1); -- El primer
fragmento es el nombre de la tabla
                    --2.4. Seleccion de campos a homogeneizar
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION',
'2.4.SELECCION CAMPOS A HOMOGENEIZAR', NULL
                    SELECT
                           CR.TABLA_ORIGEN AS TABLE_NAME,
                           CR.CAMPO AS COLUMN_NAME,
                           CR.TABLA_DESTINO AS TABLE_DEST,
                           CR.TRANSFORMACION AS TRANSFORMACION,
                           ROW_NUMBER() OVER(ORDER BY CR.TABLA_ORIGEN, CR.CAMPO,
CR.TRANSFORMACION) AS RN
                    INTO #TMP_RELACION_CAMPOS
                    FROM [COD].[PARAM_HOMOGENEIZACION_CAMPOS] CR
                    WHERE CR.TABLA_ORIGEN=@Tabla_Origen
                    --2.5. Comprobacion de que hay campos que homogeneizar
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION',
'2.5.COMPROBACION EXISTENCIA CAMPOS A HOMOGENEIZAR', NULL
                    SELECT @totalcolumnas = COUNT(*) FROM #TMP_RELACION_CAMPOS
                    IF @totalcolumnas = 0
                    BEGIN
```

```
                        EXEC DQ.LOG 'ERROR', @Cliente,@Servicio,@Hito,
'HOMOGENEIZACION', 'ERROR', '2.5.REVISA LOS VALORES DE LOS PARAMETROS INTRODUCIDOS'
                        EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
'HOMOGENEIZACION', '99.FIN';
                        RETURN;
                END
                --2.6. Creacion de tabla temporal para realizar el proceso de
homogeneización
                EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION',
'2.6.CREACION TABLA TEMPORAL STG';
                SET @SqlInsert = 'SELECT *
                INTO ##TMP_ORIGEN
                FROM ' + @Tabla_Origen + '
                ;'

                EXEC sp_executesql @SqlInsert
                --2.7. Inicio de transformaciones de dato
                EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION',
'2.7.INICIO TRANSFORMACION DEL DATO';
                WHILE @contador <= @totalcolumnas
                BEGIN
                        BEGIN TRY
                                --2.7.1 Inicializacion campos a transformar
                                EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
'HOMOGENEIZACION', '2.7.1.INICIALIZACION CAMPOS A TRANSFORMAR';
                                SELECT
                                        @tablename = TABLE_NAME,
                                        @columnname = COLUMN_NAME,
                                        @transformation = TRANSFORMACION
                                FROM #TMP_RELACION_CAMPOS
                                WHERE RN = @contador
                                --2.7.2 Registro en log inicio de ejecución a
nivel de campo
                                SET @message = LEFT('2.7.2.INICIO HOMOGENEIZACION
TABLA ' + @tablename + ' CAMPO ' + @columnname, 100)
                                EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
'HOMOGENEIZACION', @message

                                SET @query = 'UPDATE TABLA SET ' + @columnname + '
= ' + @transformation + ' FROM ##TMP_ORIGEN AS TABLA;'
                                EXEC sp_executesql @query
                                SET @contador = @contador + 1
                        END TRY
                        BEGIN CATCH
                                --2.7.3 Insertar en tabla de log las excepciones
                                SET @message = '2.7.3.ERROR HOMOGENEIZACION ' +
@tablename + ' - ' + @columnname
                                SET @error = ERROR_MESSAGE()
                                EXEC DQ.LOG 'ERROR', @Cliente,@Servicio,@Hito,
'HOMOGENEIZACION', @message, @error
                                SET @contador = @contador + 1
                        END CATCH
                END

                --2.8. Volcado de validaciones
```

```
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION',
'2.8.VOLCADO DE VALIDACIONES';

                    --2.8.1 Seleccion del campo que mostrara el error de la
validación
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION',
'2.8.1.SELECCION CAMPO A MOSTRAR EN ERROR';
                    SELECT @ID_Unico = CAMPO_MOSTRAR_ERROR
                    FROM [COD].[PARAM_IDENTIFICADORES_TABLAS]
                    WHERE TABLA = @tableOrigen
                    --2.8.2. Inercion de datos en tabla de errores
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION',
'2.8.2.INSERCION DE DATOS EN TABLA ERRORES';

                    DECLARE campos_stg CURSOR FOR
                    SELECT  CONCAT('[',COLUMN_NAME,']')
                    FROM INFORMATION_SCHEMA.COLUMNS
                    WHERE TABLE_NAME = @tableOrigen AND TABLE_SCHEMA = @schemaOrigen
AND COLUMN_NAME NOT IN ('CHECKSUM', 'FEC_FICHERO');
                    OPEN campos_stg;
                    FETCH NEXT FROM campos_stg INTO @campo_stg;
                    DELETE FROM DQ.VAL_CAMPOS_HOM
                           WHERE TABLA_ORIGEN=@Tabla_Origen
                    WHILE @@FETCH_STATUS = 0
                    BEGIN
                           BEGIN TRY
                           SET @message = '2.8.2.1. INICIO RELLENO DE LA TABLA DE
ERRORES - ' + @campo_stg
                           EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
'HOMOGENEIZACION', @message
                           DECLARE @sqlQuery NVARCHAR(MAX);
                           SET @sqlQuery = '
                                  INSERT INTO DQ.VAL_CAMPOS_HOM
                                  SELECT
                                         '+CAST(@CLIENTE AS VARCHAR)+' AS
ID_CLIENTE,
                                         '+CAST(@SERVICIO AS VARCHAR)+' AS
ID_SERVICIO,
                                         '+CAST(@HITO AS VARCHAR)+' AS ID_HITO,
                                         ''' + @Tabla_Origen + ''' AS TABLA_ORIGEN,
                                         ''' + @Tabla_Destino + ''' AS TABLA_DEST,
                                         A.' + @ID_Unico + ' AS VALOR_ID,
                                         ''' + @campo_stg + ''' AS CAMPO,
                                         A.' + @campo_stg + ' AS VALOR_ORIGEN,
                                         B.' + @campo_stg + ' AS VALOR_DEST,
                                         GETDATE() AS FCH_DATOS
                                  FROM
                                         ' + @Tabla_Origen + ' A
                                  LEFT JOIN
                                         ##TMP_ORIGEN B ON A.' + @ID_Unico + ' = B.'
+ @ID_Unico + '
                                  WHERE
                                         A.' + @campo_stg + ' IS NOT NULL
                                         AND B.' + @campo_stg + ' IS NULL;'
                                  EXEC sp_executesql @sqlQuery
                           END TRY
```

```
                        BEGIN CATCH
                                SET @message = '2.8.2.2.ERROR HOMOGENEIZACION
RELLENO EN LA TABLA DE ERRORES - ' + @campo_stg
                                SET @error = ERROR_MESSAGE()
                                EXEC DQ.LOG 'ERROR', @Cliente,@Servicio,@Hito,
'HOMOGENEIZACION', @message, @error
                        END CATCH
                        FETCH NEXT FROM campos_stg INTO @campo_stg;
                END
                CLOSE campos_stg
                DEALLOCATE campos_stg

                --2.9. Completitud tabla HOM
                EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION',
'2.9.INICIO VOLCADO A TABLA HOM';
                --2.9.1 Inclusion checksum y concat_sql
                EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION',
'2.9.1.INCLUSION CHECKSUM Y CONCAT_SQL';
                ALTER TABLE ##TMP_ORIGEN
                ADD CHECKSUM NVARCHAR(MAX),CONCAT_SQL NVARCHAR(MAX)
                --2.9.2 Volcado a HOM
                EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION',
'2.9.2.VOLCADO A TABLA HOM';
                SET @SqlInsert1 = 'INSERT INTO ' + @Tabla_Destino + ' ' +'SELECT
* FROM ##TMP_ORIGEN;'
                EXEC sp_executesql @SqlInsert1
                --2.9.3 Completitud CONCAT_SQL en HOM
                EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION',
'2.9.3.COMPLETITUD CONCAT_SQL EN HOM';
                BEGIN TRY
                        SELECT @concatSQL=CONCAT_SQL
                        FROM COD.PARAM_CONCAT_SQL
                        WHERE ID_CLIENTE=@Cliente AND ID_SERVICIO=@SERVICIO AND
ID_HITO=@HITO AND TABLA=@Tabla_Destino
                        SET @sqlUpdateConcat = '
                                UPDATE ' + @Tabla_Destino + '
                                SET CONCAT_SQL = '+ @concatSQL +';'
                        EXEC sp_executesql @sqlUpdateConcat;
                END TRY
                BEGIN CATCH
                        SET @message = '2.9.3.ERROR EN EL RELLENO DE CONCAT_SQL
DE LA TABLA ' + @Tabla_Destino
                        SET @error = ERROR_MESSAGE()
                        EXEC DQ.LOG 'ERROR', @Cliente,@Servicio,@Hito,
'HOMOGENEIZACION', @message, @error
                END CATCH

                --2.9.4 Completitud Estados
                IF @Tabla_Destino='HOM.AAM_BI_VENTAS'
                BEGIN TRY
                        --2.9.5.1. Actualización SOC_CLIENTE
                        EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
'HOMOGENEIZACION', '2.9.5.1.ACTUALIZACION SOC_CLIENTE'
                        UPDATE HOM.AAM_BI_VENTAS
                        SET  SOC_CLIENTE =
                                CASE
```

```
                                        WHEN SOC_CLIENTE IS NULL OR SOC_CLIENTE =
'A definir' THEN TIPO_CLIENTE
                                        ELSE SOC_CLIENTE
                                END
                        EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
'HOMOGENEIZACION', '2.9.4.COMPLETITUD ESTADOS EN HOM'
                        SET @sqlUpdateEstado = '
                                UPDATE ' + @Tabla_Destino + '
                                SET ESTADO_COMUNIDADES =
                                                CASE
                                                        WHEN B.APLICA_CCPP = 1 AND
A.ESTADO_COMUNIDADES IS NULL THEN ''PTE. GESTION''
                                                        WHEN B.APLICA_CCPP = 0 AND
A.ESTADO_COMUNIDADES IS NULL THEN ''NO GESTION CCPP''
                                                        ELSE ''PTE. GESTION''
                                                END,
                                        ESTADO_TRIBUTOS =
                                                CASE
                                                        WHEN B.APLICA_TRIB = 1 AND
A.ESTADO_TRIBUTOS IS NULL THEN ''PTE_GESTION''
                                                        WHEN B.APLICA_TRIB = 0 AND
A.ESTADO_TRIBUTOS IS NULL THEN ''NO_GESTION_TRIBUTOS''
                                                        ELSE ''PTE_GESTION''
                                                END
                                FROM '+@Tabla_Destino+' A
                                LEFT JOIN COD.CARTERAS D ON D.CARTERA=
A.SOC_CLIENTE
                                LEFT JOIN COD.PARAM_CARTERAS B ON D.ID_CARTERA =
B.ID_CARTERA
                                WHERE (A.ESTADO_COMUNIDADES IS NULL OR
A.ESTADO_TRIBUTOS IS NULL)'

                        EXEC sp_executesql @sqlUpdateEstado;

                        --2.9.5 Completitud Estados y Cambios Luis
                        EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
'HOMOGENEIZACION', '2.9.5.CAMBIOS RELLENO SOC_CLIENTE, INSERCIÓN DE NUEVAS CARTERAS Y
CAMBIOS EN FEC_POSICIONAMIENTO'



                        --2.9.5.2. Insercion carteras
                        EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
'HOMOGENEIZACION', '2.9.5.2.INSERCION CARTERAS EN COD_CARTERAS'
                        INSERT INTO COD.CARTERAS (CARTERA,ORIGEN)
                        SELECT
                                SOC_CLIENTE,
                                'BI' AS ORIGEN
                        FROM HOM.AAM_BI_VENTAS V
                        LEFT JOIN COD.CARTERAS C
                        ON V.SOC_CLIENTE=C.CARTERA
                        WHERE C.CARTERA IS NULL


                        --2.9.5.3. Verificacion insercion carteras
```

```
                              EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
'HOMOGENEIZACION', '2.9.5.3.VERIFICACION NUEVAS CARTERAS'
                              IF @@ROWCOUNT > 0
                                     BEGIN
                                            EXEC DQ.LOG 'INFO',
@Cliente,@Servicio,@Hito, 'HOMOGENEIZACION', '2.9.5.3.SE INSERTARON NUEVAS CARTERAS EN
COD.CARTERAS'
                                     END
                       END TRY
                       BEGIN CATCH
                              SET @message = 'ERROR EN EL RELLENO SOC_CLIENTE DE LA
TABLA ' + @Tabla_Destino
                              SET @error = ERROR_MESSAGE()
                              EXEC DQ.LOG 'ERROR', @Cliente,@Servicio,@Hito,
'HOMOGENEIZACION', @message, @error
                       END CATCH
                       IF @Hito=1
                       BEGIN
                              SET @Sql = 'DELETE FROM ' + @Tabla_Destino + ' WHERE
REF_UE IS NULL AND ID_OPORTUNIDAD IS NULL;'
                              EXEC sp_executesql @Sql
                       END

                       --2.9.6 Completitud Checksum
                       EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION',
'2.9.6.COMPLETITUD CHECKSUM EN HOM'
                       BEGIN TRY
                              SELECT @columns = STRING_AGG('[' + COLUMN_NAME + ']', ',
')
                              FROM INFORMATION_SCHEMA.COLUMNS
                              WHERE TABLE_NAME = @tableDestino AND TABLE_SCHEMA=
@schemaDestino AND COLUMN_NAME NOT IN('FEC_FICHERO','CHECKSUM')
                              SET @checksum = 'UPDATE ' + @Tabla_Destino + ' ' + 'SET
CHECKSUM = CHECKSUM(' + @columns + ');'
                              EXEC sp_executesql @checksum
                       END TRY
                       BEGIN CATCH
                              SET @message = '2.9.6.ERROR EN LA COMPLETITUD CHECKSUM EN
HOM ' + @Tabla_Destino
                              SET @error = ERROR_MESSAGE()
                              EXEC DQ.LOG 'ERROR', @Cliente,@Servicio,@Hito,
'HOMOGENEIZACION', @message, @error
                       END CATCH
                       FETCH NEXT FROM tablas_origen_destino INTO @Tabla_Origen,
@Tabla_Destino;
               END
               CLOSE tablas_origen_destino;
               DEALLOCATE tablas_origen_destino;
        END TRY
        BEGIN CATCH
               IF CURSOR_STATUS('global','tablas_origen_destino')>=-1
               BEGIN
                      DEALLOCATE tablas_origen_destino
               END
               --8 Insertar en tabla de log las excepciones
               SET @message = '99.ERROR ' + @Tabla_Origen
```

```
		SET @error = ERROR_MESSAGE()
		EXEC DQ.LOG 'ERROR', @Cliente,@Servicio,@Hito, 'HOMOGENEIZACION',
@message, @error
	END CATCH
END
```

HIS.HISTORIFICACION

```
USE [AAM_SERVICIO_CCPP_TRIBUTOS]
GO
/****** Object:  StoredProcedure [HIS].[HISTORIFICACION]    Script Date: 18/06/2024
16:39:48 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

/***********************************************/
/* Autor: Ángel Guzmán*/
/* Fecha: 20231212 */
/* Descripción:*/
/* Historificador genérico de STG a HIS. Variables de entrada:
		1. TablaHOM a historificar
		2. TablaHIS donde se historificar*/
/* Versiones:*/
/*	V1: Versión inicial)*/
/***********************************************/
ALTER PROCEDURE [HIS].[HISTORIFICACION]
    @Cliente INT,
    @Servicio INT,
	@Hito INT
AS
BEGIN
	SET NOCOUNT ON;
	DECLARE @trancount int;
	--SET @trancount = @@trancount;
	BEGIN TRY
		--IF @trancount = 0
		--	BEGIN TRANSACTION
		--ELSE
		--	SAVE TRANSACTION TR_HISTORIFICACION

		--#############################
		-- 1.DECLARACION DE VARIABLES #
		--#############################
	 EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'HISTORIFICACION', '1.INICIO
HISTORIFICACION'
	 DECLARE @logMessage NVARCHAR(500)
	 DECLARE @logtitulo NVARCHAR(255)
		DECLARE @count INT
		DECLARE @Tabla NVARCHAR(255)
		DECLARE @Tabla_Destino NVARCHAR(255)
		DECLARE @fchFichero DATE
		DECLARE @sql NVARCHAR(MAX)
		DECLARE @id_unico NVARCHAR(255)
		DECLARE @error NVARCHAR(MAX)
```

```
		SET @logtitulo='HISTORIFICACION'

		--#####################################
		-- 2.OBTENCION TABLAS A HISTORIFICAR #
		--#####################################
		EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logtitulo, '2.OBTENCION
TABLAS A HISTORIFICAR'
		IF CURSOR_STATUS('global', 'tablas')>=-1
		BEGIN
			DEALLOCATE tablas
		END

		DECLARE tablas CURSOR FOR
		SELECT DISTINCT TABLA_DESTINO
		FROM COD.PARAM_HOMOGENEIZACION_CAMPOS
		WHERE ID_CLIENTE = @Cliente AND ID_SERVICIO = @Servicio AND ID_HITO =
@Hito

		OPEN tablas
		FETCH NEXT FROM tablas INTO @Tabla_Destino

		WHILE @@FETCH_STATUS = 0
		BEGIN
			SET @Tabla = PARSENAME(@Tabla_Destino, 1); -- El primer
fragmento es el nombre de la tabla
			SET @logMessage = 'HISTORIFICACION HOM.' + @Tabla;
			SET @count=1
			--#####################################
			-- 3.OBTENCION NOMBRE IDENTIFICADOR #
			--#####################################
			EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
@logtitulo,'3.OBTENCION NOMBRE IDENTIFICADOR'
			SET @id_unico = (
				SELECT TOP(1) CAMPO
				FROM [COD].[PARAM_IDENTIFICADORES_TABLAS]
				WHERE TABLA = @Tabla
			);
			--############################
			-- 4.OBTENCION FCH_FICHERO #
			--############################
			EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
@logtitulo,'4.OBTENCION FCH_FICHERO'
			SET @sql='SELECT DISTINCT @fchFichero=FEC_FICHERO from
'+@Tabla_Destino+''
			EXEC sp_executesql @sql, N'@fchFichero DATE OUTPUT', @fchFichero
OUTPUT;

			IF @fchFichero is null
				SET @fchFichero=CAST(GETDATE() AS DATE)
			--################################
			-- 5.BORRADO TABLAS TEMPORALES #
			--################################
			EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logtitulo,
'5.BORRADO TABLAS TEMPORALES'
```

```
                    EXEC('DELETE FROM HIS.'+@tabla+' WHERE
FEC_FICHERO>='''+@fchFichero+''''')
                    DROP TABLE IF EXISTS ##FOTO_ACTUAL
                    DROP TABLE IF EXISTS ##FOTO_ACTUAL_BAJAS
                    DROP TABLE IF EXISTS ##BAJAS
                    --############################
                    -- 5.OBTENCION FOTO ACTUAL #
                    --############################
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
@logtitulo,'5.OBTENCION FOTO ACTUAL'
                    EXEC('SELECT *
                              INTO ##FOTO_ACTUAL
                              FROM (
                                        SELECT *,
                                                  ROW_NUMBER() OVER(PARTITION BY
'+@id_unico+' ORDER BY FEC_FICHERO DESC) AS R_N
                                        FROM HIS.'+@Tabla+'
                                   ) A
                             WHERE R_N=1 AND ESTADO<>''BAJA''
                             ')

                    EXEC('SELECT *
                              INTO ##FOTO_ACTUAL_BAJAS
                              FROM (
                                        SELECT *,
                                                  ROW_NUMBER() OVER(PARTITION BY
'+@id_unico+' ORDER BY FEC_FICHERO DESC) AS R_N
                                        FROM HIS.'+@Tabla+'
                                   ) A
                             WHERE R_N=1 AND ESTADO=''BAJA''
                             ')
                    ALTER TABLE ##FOTO_ACTUAL DROP COLUMN R_N
                    --#####################
                    -- 6.HISTORIFICACION #
                    --#####################
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
@logtitulo,'6.HISTORIFICACION'
                    --**************************
                    --6.1.Historificacion bajas *
                    ---**************************
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
@logtitulo,'6.1.HISTORIFICACION BAJAS'
                    --6.1.1.Seleccion bajas
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
@logtitulo,'6.1.1.SELECCION BAJAS'
                    EXEC('SELECT HIS.*
                              INTO ##BAJAS
                              FROM ##FOTO_ACTUAL AS HIS
                              LEFT JOIN HOM.' + @Tabla + ' HOM ON HIS.'+@id_unico+' =
HOM.'+@id_unico+'
                              WHERE HOM.'+@id_unico+' IS NULL;')
                    --6.1.2. Actualizacion FEC_FICHERO
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
@logtitulo,'6.1.2.ACTUALIZACION FECHA FICHERO'
                    UPDATE ##BAJAS
                    SET FEC_FICHERO=@fchFichero,
```

```
                        ESTADO='BAJA',
                        FEC_SISTEMA=DATEADD(HH, 1, GETDATE())

                    --6.1.3.Insercion en tabla histórica
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
@logtitulo,'6.1.3.INSERCION EN TABLA HISTORICA'
                    EXEC('INSERT INTO HIS.' + @Tabla + ' SELECT * FROM ##BAJAS');

                    --**************************
                    --6.2.Historificacion ALTAS *
                    ---*************************
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
@logtitulo,'6.2.HISTORIFICACION ALTAS'
                    --6.2.1.Insercion altas
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
@logtitulo,'6.2.1.INSERCION ALTAS'

                    EXEC('INSERT INTO HIS.' + @Tabla + '
                             SELECT HOM.*,
                                       ''ALTA'' AS ESTADO,
                                          DATEADD(HH, 1, GETDATE()) AS FEC_SISTEMA
                             FROM HOM.' + @Tabla + ' HOM
                             LEFT JOIN HIS.' + @Tabla + ' HIS ON HOM.'+@id_unico+' =
HIS.'+@id_unico+'
                             WHERE HIS.'+@id_unico+' IS NULL;
                            ');
                    --**************************************
                    --6.3.Historificacion MODIFICACIONES      *
                    ---*************************************
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
@logtitulo,'6.3.HISTORIFICACION MODIFICACIONES'
                    --6.3.1.Seleccion modificaciones
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
@logtitulo,'6.3.1.INSERCION MODIFICACIONES'
                    EXEC('
                           INSERT INTO HIS.' + @Tabla + '
                           SELECT HOM.*, ''MODIFICACION'' AS ESTADO, DATEADD(HH, 1,
GETDATE()) AS FEC_SISTEMA
                           FROM ##FOTO_ACTUAL HIS
                           INNER JOIN HOM.' + @Tabla + ' HOM ON HIS.'+@id_unico+' =
HOM.'+@id_unico+'
                           WHERE HOM.CHECKSUM <> HIS.CHECKSUM;
                    ');
                    --*******************************
                    --6.4.Historificacion REALTAS      *
                    ---******************************
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
@logtitulo,'6.4.HISTORIFICACION REALTAS'
                    --6.4.1.Seleccion modificaciones
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
@logtitulo,'6.4.1.INSERCION REALTAS'
                    EXEC('
                                 INSERT INTO HIS.' + @Tabla + '
                                 SELECT HOM.*, ''REALTA'' AS ESTADO, DATEADD(HH, 1,
GETDATE()) AS FEC_SISTEMA
                                 FROM HOM.' + @Tabla + ' HOM
```

```
                                    INNER JOIN ##FOTO_ACTUAL_BAJAS ACT
                                    ON HOM.'+@id_unico+' = ACT.'+@id_unico+'')

                SET @logMessage = '7.FIN HISTORIFICACION ' + @Tabla;
                   EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logtitulo,
@logMessage

                        FETCH NEXT FROM tablas INTO @Tabla_Destino;
               END
               CLOSE tablas
               DEALLOCATE tablas;

               --IF @trancount = 0
               --     COMMIT;
    END TRY
    BEGIN CATCH
               IF CURSOR_STATUS('global', 'tablas')>=-1
               BEGIN
                      DEALLOCATE tablas
               END
               --DECLARE @xstate int;
               SET @logMessage = '99.ERROR EN LA HISTORIFICACIÓN ' + @Tabla;
               SET @error=ERROR_MESSAGE()

               --SELECT @xstate = XACT_STATE();

        EXEC DQ.LOG 'ERROR', @Cliente,@Servicio,@Hito, @logtitulo, '99.ERROR',
@logMessage;
           --IF @xstate = -1
                -- ROLLBACK;
               --IF @xstate = 1 AND @trancount = 0
               --     ROLLBACK
               --IF @xstate = 1 AND @trancount > 0
               --     ROLLBACK TRANSACTION TR_HISTORIFICACION;
               RAISERROR (@error, 16, 1);
    END CATCH;
END
```

CALC.AAM_MAESTRO_VENTAS

```
USE [AAM_SERVICIO_CCPP_TRIBUTOS]
GO
/****** Object:  StoredProcedure [CALC].[AAM_MAESTRO_VENTAS]    Script Date:
18/06/2024 16:38:42 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

/*************************************************/
 --Autor: Ángel Guzmán
 --Fecha: 20240213
 --Descripción: Proceso que genera el maestro de ventas de AAM
       --     Inputs:
       --            HIS.AAM_BI_VENTAS
       --            HIS.AAM_OPE_VENTAS (TRIB Y CCPP)
```

```
	--	Outputs:
	--		CALC.AAM_MAESTRO_CCPP_VENTAS
	--		CALC.AAM_MAESTRO_TRIB_VENTAS
 --Versiones:
	--	V1: Versión inicial
/*********************************************/
ALTER PROCEDURE [CALC].[AAM_MAESTRO_VENTAS] (
	@cliente int,
	@servicio int,
	@hito int,
	@FCH_DATOS DATE
)
AS
BEGIN
	SET NOCOUNT ON;
	--###############################
	--#    1.DECLARACION DE VARIABLES  #
	--###############################
	--1.1. Obtencion variables
	DECLARE @message NVARCHAR(500),
			@error NVARCHAR(500),
			@logMessage NVARCHAR(500),
			@logtitulo NVARCHAR(255),
			@serv nvarchar(4),
			@sql NVARCHAR(MAX),
			@FCH_DATOS_1 DATE,
			@servicio_nmb nvarchar(20),
			@Previo DATE;

	SET @logtitulo='MAESTRO AAM'

	EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, '1.INICIO
EJECUCION'
	EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, '1.1.OBTENCION
VARIABLES'

	--1.2. Obtencion variables
	EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, '1.2.OBTENCION
VARIABLE SERVICIO Y VARIABLE @FCH_DATOS_1'
	SELECT @SERV=LEFT(SERVICIO,4)
	FROM COD.SERVICIOS
	WHERE ID_SERVICIO=@SERVICIO
	IF @SERVICIO = 1
		SET @servicio_nmb='TRIBUTOS'
	ELSE
		SET @servicio_nmb='COMUNIDADES'

	-- Asignar @FCH_DATOS_1
	SET @Previo = DATEADD(DAY, -1, @FCH_DATOS); -- Comienza con el día anterior
	-- Retrocede mientras sea festivo o fin de semana
	WHILE (
		DATENAME(WEEKDAY, @Previo) = 'Saturday' -- Sábado
		OR DATENAME(WEEKDAY, @Previo) = 'Sunday' -- Domingo
		OR EXISTS (SELECT 1 FROM COD.CALENDARIO_FESTIVOS WHERE FECHA = @Previo)
-- Festivo
	)
```

```
	BEGIN
		SET @Previo = DATEADD(DAY, -1, @Previo);
	END
	SET @FCH_DATOS_1 = @Previo;

	--##############################
	--#    2.RESETEO MAESTRO POR FECHA #
	--##############################
	EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, '2.RESETEO MAESTRO
POR FECHA'
	--2.1. Borrado de todo lo que ha entrado al maestro después
	EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, '2.1.BORRADO
MAESTRO CON FECHA ENTRADA POSTERIOR A LA FECHA'
	Set @sql='DELETE FROM CALC.AAM_MAESTRO_'+@SERV+'_VENTAS WHERE
			(FEC_ENTRADA_FC>='''+ CAST(@FCH_DATOS_1 AS VARCHAR)+''' AND
FEC_ENTRADA_BI>='''+CAST(@FCH_DATOS AS VARCHAR)+''') OR
			(FEC_ENTRADA_FC IS NULL AND FEC_ENTRADA_BI>='''+CAST(@FCH_DATOS
AS VARCHAR)+''') OR
			(FEC_ENTRADA_FC>='''+ CAST(@FCH_DATOS_1 AS VARCHAR)+''' AND
FEC_ENTRADA_BI IS NULL)
		'
	EXEC sp_executesql @sql
	--2.2.Actualización del maestro con los cambios hasta la fecha
	EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, '2.2.ACTUALIZACION
DEL MAESTRO CON LOS CAMBIOS HASTA LA FECHA'
	Set @sql='
		UPDATE CALC.AAM_MAESTRO_'+@SERV+'_VENTAS
			SET
				ESTADO=
					CASE
						WHEN ESTADO=''BAJA'' AND (FEC_SALIDA_FC>=
'''+ CAST(@FCH_DATOS_1 AS VARCHAR)+''' OR FEC_SALIDA_BI>= '''+ CAST(@FCH_DATOS AS
VARCHAR)+''' ) THEN ''MODIFICACION''
						ELSE ESTADO
					END
				,ORIGEN=
					CASE
						WHEN ESTADO<>''BAJA'' THEN NULL
						ELSE ORIGEN
					END
				,FEC_ENTRADA_FC =
					CASE
						WHEN FEC_ENTRADA_FC >= '''+
CAST(@FCH_DATOS_1 AS VARCHAR)+''' THEN  NULL
						WHEN FEC_ENTRADA_FC < '''+
CAST(@FCH_DATOS_1 AS VARCHAR)+''' THEN FEC_ENTRADA_FC
						ELSE NULL
					END
				,FEC_SALIDA_FC=
					CASE
						WHEN FEC_SALIDA_FC >= '''+
CAST(@FCH_DATOS_1 AS VARCHAR)+''' THEN  NULL
						WHEN FEC_SALIDA_FC < '''+ CAST(@FCH_DATOS_1
AS VARCHAR)+''' THEN FEC_SALIDA_FC
						ELSE NULL
					END
```

```
                              ,FEC_ENTRADA_BI=
                                     CASE
                                            WHEN FEC_ENTRADA_BI >= '''+ CAST(@FCH_DATOS
AS VARCHAR)+''' THEN  NULL
                                            WHEN FEC_ENTRADA_BI < '''+ CAST(@FCH_DATOS
AS VARCHAR)+''' THEN FEC_ENTRADA_BI
                                            ELSE NULL
                                     END
                              ,FEC_SALIDA_BI=
                                     CASE
                                            WHEN FEC_SALIDA_BI >= '''+ CAST(@FCH_DATOS
AS VARCHAR)+''' THEN  NULL
                                            WHEN FEC_SALIDA_BI < '''+ CAST(@FCH_DATOS
AS VARCHAR)+''' THEN FEC_SALIDA_BI
                                            ELSE NULL
                                     END
                              ,FEC_FACT_'+@SERV+'=
                                     CASE
                                            WHEN FEC_FACT_'+@SERV+'>='''+
CAST(@FCH_DATOS AS VARCHAR)+''' THEN  NULL
                                            WHEN FEC_FACT_'+@SERV+' < '''+
CAST(@FCH_DATOS AS VARCHAR)+''' THEN FEC_FACT_'+@SERV+'
                                            ELSE NULL
                                     END
                              ,FACT_'+@SERV+'=
                                     CASE
                                            WHEN FEC_FACT_'+@SERV+'>='''+
CAST(@FCH_DATOS AS VARCHAR)+''' THEN  NULL
                                            WHEN FEC_FACT_'+@SERV+' < '''+
CAST(@FCH_DATOS AS VARCHAR)+''' THEN FACT_'+@SERV+'
                                            ELSE COALESCE(FACT_'+@SERV+', NULL)
                                     END
                     '
       EXEC sp_executesql @sql
       --2.3.Borrado tablas posteriores a maestro
       EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, '2.3.BORRADO TABLAS
POSTERIORES A MAESTRO'
       DELETE FROM CALC.VAL_FEC_POSICIONAMIENTO WHERE ID_CLIENTE=@CLIENTE AND
ID_SERVICIO=@SERVICIO AND ID_HITO=@HITO AND FEC_CAMBIO>@FCH_DATOS
       DELETE FROM CALC.VAL_ESTADOS_VENTAS WHERE ID_CLIENTE=@CLIENTE AND
ID_SERVICIO=@SERVICIO AND ID_HITO=@HITO AND FEC_DATOS>@FCH_DATOS
       DELETE FROM CALC.EVOLUCION_VENTAS WHERE ID_CLIENTE=@CLIENTE AND
ID_SERVICIO=@SERVICIO AND ID_HITO=@HITO AND FEC_DATOS>=@FCH_DATOS

       --2.4.Borrado tablas temporales auxiliares
       EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, '2.4.BORRADO TABLAS
TEMPORALES AUXILIARES'
       DROP TABLE IF EXISTS ##FOTO_ACTUAL_BI
       DROP TABLE IF EXISTS ##FOTO_ACTUAL_BI_BAJAS
       DROP TABLE IF EXISTS ##FOTO_ACTUAL_SERVICIO
       DROP TABLE IF EXISTS ##FOTO_ACTUAL_SERVICIO_BAJAS

       --##########################################
       -- 3.OBTENCION TABLAS DEL DIA DEL HISTORICO #
       --##########################################
```

```
	EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito, @logtitulo, '3.INICIO CAMBIOS EN
LOS REGISTROS'
	--3.1.Obtencion perimetro vivo BI
	EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, '3.1.OBTENCION VIVO
BI'
	SET @sql='SELECT *
			INTO ##FOTO_ACTUAL_BI
			FROM (
					SELECT *,
				ROW_NUMBER() OVER (PARTITION BY CONCAT_SQL,
ID_OPORTUNIDAD ORDER BY FEC_FICHERO DESC) AS RN
					FROM HIS.AAM_BI_VENTAS
					WHERE FEC_FICHERO<='''+CAST(@FCH_DATOS AS
VARCHAR)+'''
				) AS HIS_BI
			WHERE HIS_BI.RN = 1
			AND HIS_BI.ESTADO <> ''BAJA''
		'
	EXEC sp_executesql @sql

	--3.2.Obtencion perimetro vivo servicio
	EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, '3.2.OBTENCION VIVO
SERVICIO'

	SET @sql='SELECT *
			INTO ##FOTO_ACTUAL_SERVICIO
			FROM (
					SELECT *,
				ROW_NUMBER() OVER (PARTITION BY CONCAT_SQL ORDER BY
FEC_FICHERO DESC) AS RN
					FROM HIS.AAM_OPE_'+@SERV+'_VENTAS
					WHERE FEC_FICHERO<='''+(CAST((@FCH_DATOS_1) AS
VARCHAR))+'''
				) AS HIS_BI
			WHERE HIS_BI.RN = 1
			AND HIS_BI.ESTADO <> ''BAJA''
		'
	EXEC sp_executesql @sql

	--3.3.Activos perimetro bajas en el BI (Hoy)
	EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, '3.3.OBTENCION BI
BAJAS DIARIO'
	SET @sql='SELECT *
			INTO ##FOTO_ACTUAL_BI_BAJAS
			FROM (
					SELECT *, ROW_NUMBER() OVER (PARTITION BY
CONCAT_SQL, ID_OPORTUNIDAD ORDER BY FEC_FICHERO DESC) AS RN
					FROM HIS.AAM_BI_VENTAS
					WHERE FEC_FICHERO='''+CAST(@FCH_DATOS AS
VARCHAR)+'''
				) A
			WHERE A.RN = 1 AND ESTADO = ''BAJA''';
	EXEC sp_executesql @sql

	--3.4.Activos perimetro bajas en el operativo CCPP (Hoy) - Ventas Especiales y
Postventas fuera de fichero
```

```
	EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, '3.4.OBTENCION
SERVICIO BAJAS DIARIO'
	SET @sql='SELECT *
			INTO ##FOTO_ACTUAL_SERVICIO_BAJAS
			FROM (
					SELECT *, ROW_NUMBER() OVER (PARTITION BY
CONCAT_SQL ORDER BY FEC_FICHERO DESC) AS RN
					FROM HIS.AAM_OPE_'+@SERV+'_VENTAS
					WHERE FEC_FICHERO='''+CAST(@FCH_DATOS_1 AS
VARCHAR)+'''
				) A
				WHERE A.RN = 1 AND ESTADO = ''BAJA'' AND
(VENTA_ESPECIAL=''SI'' OR POSTVENTA=''SI, NO PRESENTE EN EL BI'')'
	EXEC sp_executesql @sql

	--#################################
	--#    4.CAMBIOS EN LOS REGISTROS   #
	--#################################
	EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito, @logtitulo, '4.INICIO CAMBIOS EN
LOS REGISTROS'

	--4.1.Cambios en la fecha de posicionamiento
	EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito, @logtitulo, '4.1.REGISTRO
CAMBIOS FEC_POSICIONAMIENTO'
	SET @sql='INSERT INTO CALC.VAL_FEC_POSICIONAMIENTO
(ID_CLIENTE,ID_SERVICIO,ID_HITO,CONCAT_SQL, ID_OPORTUNIDAD, FEC_POSICIONAMIENTO_NEW,
PRIORIDAD_PATRIMONIO_NEW, FEC_POSICIONAMIENTO_OLD, PRIORIDAD_PATRIMONIO_OLD,
FEC_CAMBIO)
		SELECT '+CAST(@CLIENTE AS VARCHAR)+' AS ID_CLIENTE,
				'+CAST(@SERVICIO AS VARCHAR)+' AS ID_SERVICIO,
				'+CAST(@HITO AS VARCHAR)+' AS ID_HITO,
				   CALC.CONCAT_SQL,
				   CALC.ID_OPORTUNIDAD,
				   HIS.FEC_POSICIONAMIENTO AS
FEC_POSICIONAMIENTO_NEW,
				   HIS.PRIORIDAD_PATRIMONIO AS
PRIORIDAD_PATRIMONIO_NEW,
				   CALC.FEC_POSICIONAMIENTO AS
FEC_POSICIONAMIENTO_OLD,
				   CALC.PRIORIDAD_PATRIMONIO AS
PRIORIDAD_PATRIMONIO_OLD,
				   '''+CAST(@FCH_DATOS AS NVARCHAR)+''' AS
FEC_CAMBIO
			FROM ##FOTO_ACTUAL_BI HIS
			INNER JOIN CALC.AAM_MAESTRO_'+@SERV+'_VENTAS AS CALC
			ON HIS.ID_OPORTUNIDAD = CALC.ID_OPORTUNIDAD
			AND HIS.CONCAT_SQL = CALC.CONCAT_SQL
			WHERE (CALC.FEC_POSICIONAMIENTO <> HIS.FEC_POSICIONAMIENTO
				OR (CALC.FEC_POSICIONAMIENTO IS NULL AND
HIS.FEC_POSICIONAMIENTO IS NOT NULL)
				OR (CALC.FEC_POSICIONAMIENTO IS NOT NULL AND
HIS.FEC_POSICIONAMIENTO IS NULL))
				AND CALC.ESTADO<>''BAJA'';'
		EXEC sp_executesql @sql

	--#####################################
```

```
	--# 4.2.REGISTRO DE CAMBIOS EN ESTADOS #
	--#####################################
	EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito, @logtitulo, '4.2.REGISTRO
CAMBIOS ESTADOS'
	SET @sql='INSERT INTO CALC.VAL_ESTADOS_VENTAS
(ID_CLIENTE,ID_SERVICIO,ID_HITO,CONCAT_SQL, ID_OPORTUNIDAD, GESTOR, ESTADO_NEW,
ESTADO_OLD, FEC_DATOS)
				SELECT '+CAST(@CLIENTE AS VARCHAR)+' AS ID_CLIENTE,
					'+CAST(@SERVICIO AS VARCHAR)+' AS ID_SERVICIO,
					'+CAST(@HITO AS VARCHAR)+' AS ID_HITO,
					CALC.CONCAT_SQL,
					CALC.ID_OPORTUNIDAD,
					HIS.GESTOR_'+@SERV+' AS GESTOR,
					HIS.ESTADO_'+@servicio_nmb+' AS ESTADO_NEW,
					CALC.ESTADO_'+@servicio_nmb+' AS ESTADO_OLD,
					'''+CAST(@FCH_DATOS AS NVARCHAR)+''' AS FEC_DATOS
				FROM ##FOTO_ACTUAL_SERVICIO HIS
				INNER JOIN CALC.AAM_MAESTRO_'+@SERV+'_VENTAS CALC
				ON COALESCE(HIS.ID_OPORTUNIDAD,1) =
COALESCE(CALC.ID_OPORTUNIDAD,1)
				AND HIS.CONCAT_SQL = CALC.CONCAT_SQL
				WHERE CALC.ESTADO_'+@servicio_nmb+' <>
HIS.ESTADO_'+@servicio_nmb+'
				AND CALC.ESTADO<>''BAJA'';'
		EXEC sp_executesql @sql

	--#################################
	--#    5.INICIO GENERACIÓN DE MAESTRO #
	--#################################
	EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito, @logtitulo, '5.INICIO GENERACION
DE MAESTRO'
	BEGIN TRY
		IF @SERVICIO=1
			BEGIN TRY
				--5.1.1.Actualizacion activos que pasan de OK a KO. Damos
de baja el OK del maestro para el posterior insert a "revisar, ha pasado de ok a ko"
				UPDATE M
				SET
			M.ESTADO='BAJA'
		FROM CALC.AAM_MAESTRO_TRIB_VENTAS M
				INNER JOIN ##FOTO_ACTUAL_SERVICIO OPE ON
M.CONCAT_SQL=OPE.CONCAT_SQL
		WHERE M.ESTADO<>'BAJA' AND OPE.ESTADO_TRIBUTOS IN (SELECT ESTADO FROM
COD.ESTADO WHERE OK_KO='KO')
				AND M.ESTADO_TRIBUTOS IN (SELECT ESTADO FROM COD.ESTADO
WHERE OK_KO='OK')
				--
************************************************************
				--5.1.2.Altas Ventas Especiales y activos que pasan de KO
a OK   *
				--
************************************************************
				EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.1.ALTA VENTAS ESPECIALES Y ACTIVOS QUE PASAN DE OK A KO'
				INSERT INTO CALC.AAM_MAESTRO_TRIB_VENTAS
				SELECT
```

```
                                    OPE.FEC_FICHERO
                                    ,OPE.ID_OPORTUNIDAD
                                    ,OPE.CONCATENADO
                                    ,OPE.REF_UE
                                    ,OPE.UE
                                    ,OPE.EDIFICIO
                                    ,OPE.DEPARTAMENTO
                                    ,CAST(NULL AS NVARCHAR(50)) AS REF_ORIGEN
                                    ,OPE.SOC_CLIENTE
                                    ,CAST(NULL AS NVARCHAR(50)) AS TIPO_CLIENTE
                                    ,CAST(NULL AS NVARCHAR(50)) AS SOC_PROPIETARIA
                                    ,CAST(NULL AS NVARCHAR(50)) AS CM
                                    ,OPE.FEC_POSICIONAMIENTO
                                    ,CAST(NULL AS NVARCHAR(50)) AS ESTADO_DFA
                                    ,CAST(NULL AS DATE) AS FEC_PREVISTA_DFA
                                    ,OPE.PRIORIDAD_PATRIMONIO
                                    ,CAST(NULL AS NVARCHAR(50)) AS MAYORISTA_MINORISTA
                                    ,CAST(NULL AS NVARCHAR(50)) AS
GESTOR_FORMALIZACION
                                    ,CAST(NULL AS NVARCHAR(50)) AS DELEGACION
                                    ,OPE.MUNICIPIO
                                    ,OPE.PROVINCIA
                                    ,OPE.GESTOR_NORMALIZADO
                                    ,CAST(NULL AS NVARCHAR(50)) AS TIPO_INMUEBLE
                                    ,OPE.SUBTIPOLOGIA
                                    ,OPE.REF_CATASTRAL
                                    ,OPE.NUMERO_RC
                                    ,OPE.RC_PTE
                                    ,OPE.NMB_COMPRADOR
                                    ,CAST(NULL AS DATE) AS FEC_RESERVA
                                    ,OPE.FEC_PREVISTA_FIRMA
                                    ,CAST(NULL AS DATE) FP_AÑO
                                    ,OPE.FR_AÑO
                                    ,OPE.PAO
                                    ,OPE.VBC
                                    ,OPE.ESTADO_TRIBUTOS
                                    ,OPE.FEC_CARGA_IBI
                                    ,OPE.DIAS_PTE_PAGO_IBI
                                    ,OPE.IMP_TASAS_ANUALIZADO
                                    ,OPE.[IMP_DEUDA/PRECIO_VENTA]
                                    ,OPE.IMP_IBI_ANUALIZADO
                                    ,OPE.IMP_PRORRATA_VEND
                                    ,OPE.IMP_PRORRATA_COMP
                                    ,OPE.FEC_INI_GESTION_TRIB
                                    ,OPE.FEC_ULT_GESTION_TRIB
                                    ,OPE.FEC_FIN_GESTION_TRIB
                                    ,OPE.FEC_ENVIO_PAGO_TRIB
                                    ,OPE.NUM_REMESA_TRIB
                                    ,OPE.PROVEEDOR
                                    ,OPE.GESTOR_TRIB
                                    ,OPE.COMENTARIOS_TRIB
                                    ,DAY(OPE.FEC_POSICIONAMIENTO) AS DIA_POS
                                    ,MONTH(OPE.FEC_POSICIONAMIENTO) AS MES_POS
                                    ,YEAR(OPE.FEC_POSICIONAMIENTO) AS AÑO_POS
                                    ,OPE.ESTADO_TRIB
```

```
                                    ,CASE WHEN M.ESTADO='BAJA' AND M.ESTADO_TRIBUTOS
IN (SELECT ESTADO FROM COD.ESTADO WHERE OK_KO='OK') AND OPE.ESTADO_TRIBUTOS IN (SELECT
ESTADO FROM COD.ESTADO WHERE OK_KO='KO') THEN 'REVISAR, HA PASADO DE OK A KO'
                                          ELSE OPE.REVISAR_TRIB END AS REVISAR_TRIB
                                    ,OPE.VENTA_ESPECIAL
                                    ,OPE.POSTVENTA
                                    ,CASE
                                          WHEN LEN(OPE.REF_CATASTRAL) != 20 AND
OPE.PROVINCIA NOT IN ('ÁLAVA', 'VIZCAYA', 'GUIPÚZCOA', 'NAVARRA') THEN 'INCORRECTA'
                                          WHEN OPE.REF_CATASTRAL IS NULL THEN 'NO_RC'
                                          ELSE 'OK'
                                    END AS RC_CORRECTA
                                    ,OPE.CONCAT_SQL
                                    ,OPE.CHECKSUM
                                    ,OPE.ESTADO
                                    ,OPE.FEC_SISTEMA
                                    ,'OPERATIVO' AS ORIGEN
                                    ,CAST(NULL AS DATE) AS FEC_ENTRADA_FC
                                    ,CAST(NULL AS DATE) AS FEC_SALIDA_FC
                                    ,CAST(NULL AS DATE) AS FEC_ENTRADA_BI
                                    ,CAST(NULL AS DATE) AS FEC_SALIDA_BI
                                    ,CAST(NULL AS DATE) AS FEC_FACT_TRIB
                                    ,NULL AS FACT_TRIB
                                    ,NULL AS AGING_MEDIO_VENTA
                                    ,NULL AS AGING_MEDIO_GESTION
                                    ,NULL AS AGING_MEDIO_INICIO_GESTION
                              FROM ##FOTO_ACTUAL_SERVICIO OPE
                              LEFT JOIN (SELECT*FROM(
                                    SELECT *,
                                          ROW_NUMBER()OVER(PARTITION BY CONCAT_SQL
ORDER BY FEC_FICHERO DESC) AS RN
                                    FROM CALC.AAM_MAESTRO_TRIB_VENTAS)
                              X WHERE RN=1) M ON OPE.CONCAT_SQL=M.CONCAT_SQL
                              WHERE (M.CONCAT_SQL IS NULL AND OPE.VENTA_ESPECIAL='SI')
                                    OR (M.CONCAT_SQL IS NOT NULL AND M.ESTADO='BAJA')

                              --*******************************
                              --5.2.Actualizacion facturacion    *
                              --*******************************
                              EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.2.ACTUALIZACION FACTURACION'
                              --5.2.1.Añadir las columnas de facturación del maestro
sobre el operativo
                              EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.2.1.ADICION COLUMNAS FACTURACION A LA FOTO ACTUAL DEL SERVICIO'
                              ALTER TABLE ##FOTO_ACTUAL_SERVICIO
                              ADD FEC_FACT_TRIB DATE,
                                    FACT_TRIB INT;
                              --5.2.2.Copiar sobre la temporal las columnas de
facturación
                              EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.2.2.IDENTIFICACION DE ACTIVOS YA FACTURADOS'
                              UPDATE O
                              SET O.FEC_FACT_TRIB = M.FEC_FACT_TRIB,
                                    O.FACT_TRIB = M.FACT_TRIB
                              FROM ##FOTO_ACTUAL_SERVICIO AS O
```

```
                              INNER JOIN CALC.AAM_MAESTRO_TRIB_VENTAS AS M ON
O.CONCAT_SQL = M.CONCAT_SQL
                              WHERE M.ESTADO<>'BAJA';

                              --5.2.4.Facturar los activos que pasan de KO a OK
                              EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.2.3.ACTUALIZACION FECHA FACTURACION PARA LO FACTURABLE'
                              UPDATE OPE
                              SET OPE.FACT_TRIB= 1,
                                     OPE.FEC_FACT_TRIB = @FCH_DATOS_1
                              FROM ##FOTO_ACTUAL_SERVICIO OPE
                              LEFT JOIN COD.ESTADO E
                              ON OPE.ESTADO_TRIBUTOS = E.ESTADO
                              WHERE E.OK_KO='OK' AND E.FACT=1
                              AND (OPE.REVISAR_TRIB IS NULL OR
OPE.REVISAR_TRIB='NUEVA_GESTION')
                              AND OPE.FEC_FACT_TRIB IS NULL
                              AND OPE.FACT_TRIB IS NULL
                              AND EXISTS (
                                           SELECT 1
                                           FROM CALC.AAM_MAESTRO_TRIB_VENTAS M
                                           LEFT JOIN COD.ESTADO E2 ON
M.ESTADO_TRIBUTOS=E2.ESTADO
                                           WHERE OPE.CONCAT_SQL=M.CONCAT_SQL
                                                  AND E2.OK_KO='KO' AND
M.ESTADO<>'BAJA'
                              )

                              --*******************************
                              --5.3.Actualizacion operativo      *
                              --*******************************
                              EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.3.ACTUALIZACION OPERATIVO'
                              UPDATE M
                              SET  M.[ID_OPORTUNIDAD] = OPE.ID_OPORTUNIDAD
                                     ,M.[CONCATENADO] = OPE.CONCATENADO
                                     ,M.[REF_UE] = OPE.REF_UE
                                     ,M.[UE] = OPE.UE
                                     ,M.[EDIFICIO] = OPE.EDIFICIO
                                     ,M.[DEPARTAMENTO] = OPE.DEPARTAMENTO
                                     ,M.[SOC_CLIENTE] = OPE.SOC_CLIENTE
                                     ,M.[FEC_POSICIONAMIENTO] = OPE.FEC_POSICIONAMIENTO
                                     ,M.[PRIORIDAD_PATRIMONIO] =
OPE.PRIORIDAD_PATRIMONIO
                                     ,M.[MUNICIPIO] = OPE.MUNICIPIO
                                     ,M.[PROVINCIA] = OPE.PROVINCIA
                                     ,M.[GESTOR_NORMALIZADO] = OPE.GESTOR_NORMALIZADO
                                     ,M.[SUBTIPOLOGIA] = OPE.SUBTIPOLOGIA
                                     ,M.[REF_CATASTRAL] = OPE.REF_CATASTRAL
                                     ,M.[NUMERO_RC] = OPE.NUMERO_RC
                                     ,M.[RC_PTE] = OPE.RC_PTE
                                     ,M.[NMB_COMPRADOR] = OPE.NMB_COMPRADOR
                                     ,M.[FEC_PREVISTA_FIRMA] = OPE.FEC_PREVISTA_FIRMA
                                     ,M.[FR_AÑO] = OPE.FR_AÑO
                                     ,M.[PAO] = OPE.PAO
                                     ,M.VBC=OPE.VBC
```

```
                                ,M.[ESTADO_TRIBUTOS] = OPE.ESTADO_TRIBUTOS
                                ,M.[FEC_CARGA_IBI] = OPE.FEC_CARGA_IBI
                                ,M.[DIAS_PTE_PAGO_IBI] = OPE.DIAS_PTE_PAGO_IBI
                                ,M.[IMP_TASAS_ANUALIZADO] =
OPE.IMP_TASAS_ANUALIZADO
                                ,M.[IMP_DEUDA/PRECIO_VENTA] =
OPE.[IMP_DEUDA/PRECIO_VENTA]
                                ,M.[IMP_IBI_ANUALIZADO] = OPE.IMP_IBI_ANUALIZADO
                                ,M.[IMP_PRORRATA_VEND] = OPE.IMP_PRORRATA_VEND
                                ,M.[IMP_PRORRATA_COMP] = OPE.IMP_PRORRATA_COMP
                                ,M.[FEC_INI_GESTION_TRIB] =
OPE.FEC_INI_GESTION_TRIB
                                ,M.[FEC_ULT_GESTION_TRIB] =
OPE.FEC_ULT_GESTION_TRIB
                                ,M.[FEC_FIN_GESTION_TRIB] =
OPE.FEC_FIN_GESTION_TRIB
                                ,M.[FEC_ENVIO_PAGO_TRIB] = OPE.FEC_ENVIO_PAGO_TRIB
                                ,M.[NUM_REMESA_TRIB] = OPE.NUM_REMESA_TRIB
                                ,M.[PROVEEDOR] = OPE.PROVEEDOR
                                ,M.[GESTOR_TRIB] = OPE.GESTOR_TRIB
                                ,M.[COMENTARIOS_TRIB] = OPE.COMENTARIOS_TRIB
                                ,M.[DIA_POS] = DAY(OPE.FEC_POSICIONAMIENTO)
                                ,M.[MES_POS] = MONTH(OPE.FEC_POSICIONAMIENTO)
                                ,M.[AÑO_POS] = YEAR(OPE.FEC_POSICIONAMIENTO)
                                ,M.[ESTADO_TRIB] = OPE.ESTADO_TRIB
                                ,M.[REVISAR_TRIB] = CASE WHEN
M.REVISAR_TRIB='REVISAR, HA PASADO DE OK A KO' AND M.FEC_ENTRADA_BI IS NULL AND
M.FEC_ENTRADA_FC IS NULL THEN M.REVISAR_TRIB ELSE OPE.REVISAR_TRIB END
                                ,M.[VENTA_ESPECIAL] = OPE.VENTA_ESPECIAL
                                ,M.[POSTVENTA] = OPE.POSTVENTA
                                ,M.[RC_CORRECTA]=
                                        CASE
                                                WHEN LEN(OPE.REF_CATASTRAL) != 20
AND OPE.PROVINCIA NOT IN ('ÁLAVA', 'VIZCAYA', 'GUIPÚZCOA', 'NAVARRA') THEN
'INCORRECTA'
                                                WHEN OPE.REF_CATASTRAL IS NULL THEN
'NO_RC'
                                                ELSE 'OK'
                                        END
                                ,M.[FEC_FICHERO] = OPE.FEC_FICHERO
                                ,M.[CONCAT_SQL] = OPE.CONCAT_SQL
                                ,M.[CHECKSUM] = OPE.CHECKSUM
                                ,M.[ESTADO] = OPE.ESTADO
                                ,M.[FEC_SISTEMA] = OPE.FEC_SISTEMA
                                ,M.FEC_FACT_TRIB= OPE.FEC_FACT_TRIB
                                ,M.FACT_TRIB = OPE.FACT_TRIB
                                ,M.ORIGEN='OPERATIVO'
                        FROM CALC.AAM_MAESTRO_TRIB_VENTAS M
                        INNER JOIN ##FOTO_ACTUAL_SERVICIO OPE
                        ON OPE.CONCAT_SQL=M.CONCAT_SQL
                        WHERE M.ESTADO<>'BAJA'

                        --**************************
                        --5.4.Actualizacion bajas    *
                        --**************************
```

```
                              EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.4.ACTUALIZACION BAJAS'
                              --5.4.1.Bajas operativo (ventas especiales y postventas
fuera de fichero)
                              EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.4.1.ACTUALIZACION BAJAS OPERATIVO'
                              UPDATE M
                              SET
                                     M.ESTADO=OPE.ESTADO
                              FROM CALC.AAM_MAESTRO_TRIB_VENTAS M
                              INNER JOIN ##FOTO_ACTUAL_SERVICIO_BAJAS OPE
                              ON OPE.CONCAT_SQL=M.CONCAT_SQL
                              WHERE M.ESTADO<>'BAJA'

                              --5.4.2.Bajas BI
                              EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.4.2.ACTUALIZACION BAJAS BI'
                              UPDATE M
                              SET
                                     M.ESTADO=BI.ESTADO
                              FROM CALC.AAM_MAESTRO_TRIB_VENTAS M
                              INNER JOIN ##FOTO_ACTUAL_BI_BAJAS BI
                              ON BI.CONCAT_SQL=M.CONCAT_SQL
                              AND BI.ID_OPORTUNIDAD=M.ID_OPORTUNIDAD
                              WHERE M.ESTADO<>'BAJA'

                              --**************************************
                              --5.6.Actualizacion activos a revisar     *
                              --**************************************
                              EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.6.ACTUALIZACION ACTIVOS A REVISAR'
                              --5.6.1.Añadir sobre la temporal del BI las columnas de
gestión que faltan de OPE
                              EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.6.1.ADICION DE COLUMNAS DE GESTION A FOTO ACTUAL TEMPORAL'
                              ALTER TABLE ##FOTO_ACTUAL_BI
                              ADD REVISAR_TRIB NVARCHAR(50);
                              -- 5.6.2.Dar de baja lo que sale como Alta en el BI y ya
existía en Maestro (para luego poner a revisar)
                              UPDATE M
                              SET M.ESTADO='BAJA'
                              FROM CALC.AAM_MAESTRO_TRIB_VENTAS M
                              INNER JOIN ##FOTO_ACTUAL_BI BI ON
M.CONCAT_SQL=BI.CONCAT_SQL
                              WHERE (BI.ESTADO = 'ALTA' OR BI.ESTADO='REALTA')
                                     AND BI.FEC_FICHERO = @FCH_DATOS AND
M.ESTADO<>'BAJA'
                              --5.6.2.Copiar las columnas de gestión sobre las altas
del BI que ya existían en el maestro (cualquier tipo de venta dada de baja)
                              EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.6.2.ACTUALIZACION DE COLUMNAS DE GESTION'
                              UPDATE BI
                              SET
                                     BI.VBC = T.VBC,
                                     BI.ESTADO_TRIBUTOS = T.ESTADO_TRIBUTOS,
                                     BI.FEC_CARGA_IBI = T.FEC_CARGA_IBI,
```

```
                                    BI.DIAS_PTE_PAGO_IBI = T.DIAS_PTE_PAGO_IBI,
                                    BI.IMP_TASAS_ANUALIZADO = T.IMP_TASAS_ANUALIZADO,
                                    BI.[IMP_DEUDA/PRECIO_VENTA] =
T.[IMP_DEUDA/PRECIO_VENTA],
                                    BI.IMP_IBI_ANUALIZADO = T.IMP_IBI_ANUALIZADO,
                                    BI.IMP_PRORRATA_VEND = T.IMP_PRORRATA_VEND,
                                    BI.IMP_PRORRATA_COMP = T.IMP_PRORRATA_COMP,
                                    BI.FEC_INI_GESTION_TRIB = T.FEC_INI_GESTION_TRIB,
                                    BI.FEC_ULT_GESTION_TRIB = T.FEC_ULT_GESTION_TRIB,
                                    BI.FEC_FIN_GESTION_TRIB = T.FEC_FIN_GESTION_TRIB,
                                    BI.FEC_ENVIO_PAGO_TRIB = T.FEC_ENVIO_PAGO_TRIB,
                                    BI.NUM_REMESA_TRIB = T.NUM_REMESA_TRIB,
                                    BI.PROVEEDOR = T.PROVEEDOR,
                                    BI.GESTOR_TRIB = T.GESTOR_TRIB,
                                    BI.COMENTARIOS_TRIB = T.COMENTARIOS_TRIB,
                                    BI.REVISAR_TRIB = 'REVISAR'
                             FROM ##FOTO_ACTUAL_BI BI
                       INNER JOIN (SELECT *
                                          FROM (
                                                 SELECT M.CONCAT_SQL
                                                        ,M.VBC
                                                        ,M.ESTADO_TRIBUTOS
                                                        ,M.FEC_CARGA_IBI
                                                        ,M.DIAS_PTE_PAGO_IBI
                                                        ,M.IMP_TASAS_ANUALIZADO
                                                        ,M.[IMP_DEUDA/PRECIO_VENTA]
                                                        ,M.IMP_IBI_ANUALIZADO
                                                        ,M.IMP_PRORRATA_VEND
                                                        ,M.IMP_PRORRATA_COMP
                                                        ,M.FEC_INI_GESTION_TRIB
                                                        ,M.FEC_ULT_GESTION_TRIB
                                                        ,M.FEC_FIN_GESTION_TRIB
                                                        ,M.FEC_ENVIO_PAGO_TRIB
                                                        ,M.NUM_REMESA_TRIB
                                                        ,M.PROVEEDOR
                                                        ,M.GESTOR_TRIB
                                                        ,M.COMENTARIOS_TRIB
                                                        ,M.ESTADO
                                                        ,ROW_NUMBER() OVER (PARTITION
BY M.CONCAT_SQL ORDER BY M.FEC_FICHERO DESC) AS RN
                                                 FROM CALC.AAM_MAESTRO_TRIB_VENTAS M
                                                 WHERE M.CONCAT_SQL IS NOT NULL
                                          ) AS T WHERE T.RN=1 AND T.ESTADO='BAJA') T
                                    ON BI.CONCAT_SQL = T.CONCAT_SQL
                                    WHERE (BI.ESTADO = 'ALTA' OR BI.ESTADO='REALTA')
                                    AND BI.FEC_FICHERO = @FCH_DATOS

                             --*******************************************************
                             --5.7.Actualizacion altas BI dadas de baja en maestro    *
                             --*******************************************************
                             EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.7.ACTUALIZACION ALTAS BI DADAS DE BAJA EN MAESTRO'
                             INSERT INTO CALC.AAM_MAESTRO_TRIB_VENTAS
                             SELECT
                                    BI.FEC_FICHERO
                                    ,BI.ID_OPORTUNIDAD
```

```
,BI.CONCATENADO
,BI.REF_UE
,BI.UE
,BI.EDIFICIO
,NULL AS DEPARTAMENTO
,BI.REF_ORIGEN
,BI.SOC_CLIENTE
,BI.TIPO_CLIENTE
,BI.SOC_PROPIETARIA
,BI.CM
,BI.FEC_POSICIONAMIENTO
,BI.ESTADO_DFA
,BI.FEC_PREVISTA_DFA
,BI.PRIORIDAD_PATRIMONIO
,BI.MAYORISTA_MINORISTA
,BI.GESTOR_FORMALIZACION
,BI.DELEGACION
,BI.MUNICIPIO
,BI.PROVINCIA
,BI.GESTOR_NORMALIZADO
,BI.TIPO_INMUEBLE
,BI.SUBTIPOLOGIA
,BI.REF_CATASTRAL
,NULL AS NUMERO_RC
,NULL AS RC_PTE
,BI.NMB_COMPRADOR
,BI.FEC_RESERVA
,BI.FEC_PREVISTA_FIRMA
,BI.FP_AÑO
,BI.FR_AÑO
,BI.PAO
,BI.VBC
,BI.ESTADO_TRIBUTOS
,BI.FEC_CARGA_IBI
,BI.DIAS_PTE_PAGO_IBI
,BI.IMP_TASAS_ANUALIZADO
,BI.[IMP_DEUDA/PRECIO_VENTA]
,BI.IMP_IBI_ANUALIZADO
,BI.IMP_PRORRATA_VEND
,BI.IMP_PRORRATA_COMP
,BI.FEC_INI_GESTION_TRIB
,BI.FEC_ULT_GESTION_TRIB
,BI.FEC_FIN_GESTION_TRIB
,BI.FEC_ENVIO_PAGO_TRIB
,BI.NUM_REMESA_TRIB
,BI.PROVEEDOR
,BI.GESTOR_TRIB
,BI.COMENTARIOS_TRIB
,DAY(BI.FEC_POSICIONAMIENTO) AS DIA_POS
,MONTH(BI.FEC_POSICIONAMIENTO) AS MES_POS
,YEAR(BI.FEC_POSICIONAMIENTO) AS AÑIO_POS
,NULL AS ESTADO_TRIB
,BI.REVISAR_TRIB
,'NO' VENTA_ESPECIAL
,NULL AS POSTVENTA
,CASE
```

```
                                        WHEN LEN(BI.REF_CATASTRAL) != 20 AND
BI.PROVINCIA NOT IN ('ÁLAVA', 'VIZCAYA', 'GUIPÚZCOA', 'NAVARRA') THEN 'INCORRECTA'
                                        WHEN BI.REF_CATASTRAL IS NULL THEN 'NO_RC'
                                        ELSE 'OK'
                                END AS RC_CORRECTA
                                ,BI.CONCAT_SQL
                                ,BI.CHECKSUM
                                ,BI.ESTADO
                                ,BI.FEC_SISTEMA
                                ,'BI' AS ORIGEN
                                ,CAST(NULL AS DATE) AS FEC_ENTRADA_FC
                                ,CAST(NULL AS DATE) AS FEC_SALIDA_FC
                                ,CAST(NULL AS DATE) AS FEC_ENTRADA_BI
                                ,CAST(NULL AS DATE) AS FEC_SALIDA_BI
                                ,CAST(NULL AS DATE) AS FEC_FACT_TRIB
                                ,NULL AS FACT_TRIB
                                ,NULL AS AGING_MEDIO_VENTA
                                ,NULL AS AGING_MEDIO_GESTION
                                ,NULL AS AGING_MEDIO_INICIO_GESTION
                        FROM ##FOTO_ACTUAL_BI BI INNER JOIN
CALC.AAM_MAESTRO_TRIB_VENTAS M ON BI.CONCAT_SQL=M.CONCAT_SQL
                        WHERE (BI.ESTADO = 'ALTA' OR BI.ESTADO='REALTA') AND
BI.FEC_FICHERO=@FCH_DATOS
                                AND M.ESTADO='BAJA' AND M.CONCAT_SQL IS NOT NULL

                        --************************************************
                        --5.8.Insercion altas BI no existentes en Maestro *
                        --************************************************
                        EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.8.INSERCION ALTAS BI NO EXISTENTES EN MAESTRO'
                        INSERT INTO CALC.AAM_MAESTRO_TRIB_VENTAS
                        SELECT
                                BI.FEC_FICHERO
                                ,BI.ID_OPORTUNIDAD
                                ,BI.CONCATENADO
                                ,BI.REF_UE
                                ,BI.UE
                                ,BI.EDIFICIO
                                ,NULL AS DEPARTAMENTO
                                ,BI.REF_ORIGEN
                                ,BI.SOC_CLIENTE
                                ,BI.TIPO_CLIENTE
                                ,BI.SOC_PROPIETARIA
                                ,BI.CM
                                ,BI.FEC_POSICIONAMIENTO
                                ,BI.ESTADO_DFA
                                ,BI.FEC_PREVISTA_DFA
                                ,BI.PRIORIDAD_PATRIMONIO
                                ,BI.MAYORISTA_MINORISTA
                                ,BI.GESTOR_FORMALIZACION
                                ,BI.DELEGACION
                                ,BI.MUNICIPIO
                                ,BI.PROVINCIA
                                ,BI.GESTOR_NORMALIZADO
                                ,BI.TIPO_INMUEBLE
                                ,BI.SUBTIPOLOGIA
```

```
                                    ,BI.REF_CATASTRAL
                                    ,NULL AS NUMERO_RC
                                    ,NULL AS RC_PTE
                                    ,BI.NMB_COMPRADOR
                                    ,BI.FEC_RESERVA
                                    ,BI.FEC_PREVISTA_FIRMA
                                    ,BI.FP_AÑO
                                    ,BI.FR_AÑO
                                    ,BI.PAO
                                    ,BI.VBC
                                    ,BI.ESTADO_TRIBUTOS
                                    ,BI.FEC_CARGA_IBI
                                    ,BI.DIAS_PTE_PAGO_IBI
                                    ,BI.IMP_TASAS_ANUALIZADO
                                    ,BI.[IMP_DEUDA/PRECIO_VENTA]
                                    ,BI.IMP_IBI_ANUALIZADO
                                    ,BI.IMP_PRORRATA_VEND
                                    ,BI.IMP_PRORRATA_COMP
                                    ,BI.FEC_INI_GESTION_TRIB
                                    ,BI.FEC_ULT_GESTION_TRIB
                                    ,BI.FEC_FIN_GESTION_TRIB
                                    ,BI.FEC_ENVIO_PAGO_TRIB
                                    ,BI.NUM_REMESA_TRIB
                                    ,BI.PROVEEDOR
                                    ,BI.GESTOR_TRIB
                                    ,BI.COMENTARIOS_TRIB
                                    ,DAY(BI.FEC_POSICIONAMIENTO) AS DIA_POS
                                    ,MONTH(BI.FEC_POSICIONAMIENTO) AS MES_POS
                                    ,YEAR(BI.FEC_POSICIONAMIENTO) AS AÑO_POS
                                    ,NULL AS ESTADO_TRIB
                                    ,NULL AS REVISAR_TRIB
                                    ,'NO' AS VENTA_ESPECIAL
                                    ,NULL AS POSTVENTA
                                    ,CASE
                                            WHEN LEN(BI.REF_CATASTRAL) != 20 AND
BI.PROVINCIA NOT IN ('ÁLAVA', 'VIZCAYA', 'GUIPÚZCOA', 'NAVARRA') THEN 'INCORRECTA'
                                            WHEN BI.REF_CATASTRAL IS NULL THEN 'NO_RC'
                                            ELSE 'OK'
                                    END AS RC_CORRECTA
                                    ,BI.CONCAT_SQL
                                    ,BI.CHECKSUM
                                    ,BI.ESTADO
                                    ,BI.FEC_SISTEMA
                                    ,'BI' AS ORIGEN
                                    ,CAST(NULL AS DATE) AS FEC_ENTRADA_FC
                                    ,CAST(NULL AS DATE) AS FEC_SALIDA_FC
                                    ,CAST(NULL AS DATE) AS FEC_ENTRADA_BI
                                    ,CAST(NULL AS DATE) AS FEC_SALIDA_BI
                                    ,CAST(NULL AS DATE) AS FEC_FACT_TRIB
                                    ,NULL AS FACT_TRIB
                                    ,NULL AS AGING_MEDIO_VENTA
                                    ,NULL AS AGING_MEDIO_GESTION
                                    ,NULL AS AGING_MEDIO_INICIO_GESTION
                            FROM ##FOTO_ACTUAL_BI BI
                            LEFT JOIN CALC.AAM_MAESTRO_TRIB_VENTAS M
                            ON BI.CONCAT_SQL=M.CONCAT_SQL
```

```
                        WHERE M.CONCAT_SQL IS NULL

                        --*******************************
                        --5.9.Actualizacion campos BI      *
                        --*******************************
                        EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.9.ACTUALIZACION CAMPOS BI'
                        UPDATE M
                        SET
                                M.ID_OPORTUNIDAD=BI.ID_OPORTUNIDAD
                               ,M.CONCATENADO=BI.CONCATENADO
                               ,M.REF_UE=BI.REF_UE
                               ,M.UE=BI.UE
                               ,M.EDIFICIO=BI.EDIFICIO
                               ,M.REF_ORIGEN=BI.REF_ORIGEN
                               ,M.SOC_CLIENTE=BI.SOC_CLIENTE
                               ,M.TIPO_CLIENTE=BI.TIPO_CLIENTE
                               ,M.SOC_PROPIETARIA=BI.SOC_PROPIETARIA
                               ,M.CM=BI.CM
                               ,M.FEC_POSICIONAMIENTO=BI.FEC_POSICIONAMIENTO
                               ,M.ESTADO_DFA=BI.ESTADO_DFA
                               ,M.FEC_PREVISTA_DFA=BI.FEC_PREVISTA_DFA
                               ,M.PRIORIDAD_PATRIMONIO=BI.PRIORIDAD_PATRIMONIO
                               ,M.MAYORISTA_MINORISTA=BI.MAYORISTA_MINORISTA
                               ,M.GESTOR_FORMALIZACION=BI.GESTOR_FORMALIZACION
                               ,M.DELEGACION=BI.DELEGACION
                               ,M.MUNICIPIO=BI.MUNICIPIO
                               ,M.PROVINCIA=BI.PROVINCIA
                               ,M.GESTOR_NORMALIZADO=BI.GESTOR_NORMALIZADO
                               ,M.TIPO_INMUEBLE=BI.TIPO_INMUEBLE
                               ,M.SUBTIPOLOGIA=BI.SUBTIPOLOGIA
                               ,M.REF_CATASTRAL=CASE WHEN M.RC_CORRECTA =
'INCORRECTA' OR M.RC_CORRECTA='NO_RC' THEN BI.REF_CATASTRAL ELSE M.REF_CATASTRAL END
                               ,M.NMB_COMPRADOR=BI.NMB_COMPRADOR
                               ,M.FEC_RESERVA=BI.FEC_RESERVA
                               ,M.FEC_PREVISTA_FIRMA=BI.FEC_PREVISTA_FIRMA
                               ,M.FP_AÑO=BI.FP_AÑO
                               ,M.FR_AÑO=BI.FR_AÑO
                               ,M.PAO=BI.PAO
                               ,M.DIA_POS=DAY(BI.FEC_POSICIONAMIENTO)
                               ,M.MES_POS=MONTH(BI.FEC_POSICIONAMIENTO)
                               ,M.AÑO_POS=YEAR(BI.FEC_POSICIONAMIENTO)
                        FROM CALC.AAM_MAESTRO_TRIB_VENTAS M
                        INNER JOIN ##FOTO_ACTUAL_BI BI
                        ON M.CONCAT_SQL=BI.CONCAT_SQL AND
M.ID_OPORTUNIDAD=BI.ID_OPORTUNIDAD

                        --*******************************
                        --5.10.Actualizacion postventas    *
                        --*******************************
                        EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.10.ACTUALIZACION POSTVENTAS'
                        --5.10.1.Postventa presente en el BI
                        EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.10.1.ACTUALIZACION POSTVENTAS PRESENTES EN BI'
                        UPDATE CALC.AAM_MAESTRO_TRIB_VENTAS
```

```
                            SET POSTVENTA=NULL
                            WHERE (PRIORIDAD_PATRIMONIO<>'V' OR PRIORIDAD_PATRIMONIO
IS NULL)
                            UPDATE CALC.AAM_MAESTRO_TRIB_VENTAS
                            SET POSTVENTA='SI'
                            WHERE PRIORIDAD_PATRIMONIO='V'
                            AND (POSTVENTA IS NULL OR POSTVENTA<>'SI, NO PRESENTE EN
EL BI') ;
                            --5.10.2.Postventa no presente en el BI
                            EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.10.2.ACTUALIZACION POSTVENTAS NO PRESENTES EN BI'
                            UPDATE A
                            SET A.POSTVENTA='SI, NO PRESENTE EN EL BI',
                                   A.ESTADO='ALTA'
                            FROM CALC.AAM_MAESTRO_TRIB_VENTAS A
                            WHERE A.PRIORIDAD_PATRIMONIO='V'
                                   AND A.ESTADO_TRIBUTOS IN (
                                   SELECT ESTADO
                                   FROM COD.ESTADO
                                   WHERE OK_KO = 'KO' AND SERVICIO='TRIBUTOS'
                                   )
                                   AND A.CONCAT_SQL IN(
                                          SELECT
                                                 CONCAT_SQL
                                          FROM ##FOTO_ACTUAL_BI_BAJAS
                                   );

                            --**************************************
                            --5.11.Actualizacion fecha de entrada      *
                            --**************************************
                            EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.11.ACTUALIZACION FECHA ENTRADA'
                            UPDATE CALC.AAM_MAESTRO_TRIB_VENTAS
                            SET FEC_ENTRADA_FC = CASE WHEN (VENTA_ESPECIAL='SI' OR
POSTVENTA='SI, NO PRESENTE EN EL BI') AND FEC_ENTRADA_FC IS NULL THEN @FCH_DATOS_1
                                                                                ELSE
FEC_ENTRADA_FC END,
                                   FEC_ENTRADA_BI = CASE WHEN (VENTA_ESPECIAL='NO'
AND (POSTVENTA IS NULL OR POSTVENTA ='SI')) AND FEC_ENTRADA_BI IS NULL THEN @FCH_DATOS
                                                                                ELSE
FEC_ENTRADA_BI END,
                                   FEC_SALIDA_FC = CASE WHEN (VENTA_ESPECIAL='SI' OR
POSTVENTA='SI, NO PRESENTE EN EL BI') AND FEC_SALIDA_FC IS NULL AND ESTADO='BAJA' THEN
@FCH_DATOS_1
                                                                                ELSE
FEC_SALIDA_FC END,
                                   FEC_SALIDA_BI = CASE WHEN (VENTA_ESPECIAL='NO' AND
(POSTVENTA IS NULL OR POSTVENTA ='SI')) AND FEC_SALIDA_BI IS NULL AND ESTADO='BAJA'
THEN @FCH_DATOS
                                                                                ELSE
FEC_SALIDA_BI END

                            UPDATE CALC.AAM_MAESTRO_TRIB_VENTAS
                            SET FEC_SALIDA_FC = CASE WHEN
FEC_ENTRADA_FC<FEC_ENTRADA_BI THEN FEC_ENTRADA_BI
                                                                                ELSE NULL END
```

```
                              WHERE FEC_SALIDA_FC IS NULL AND FEC_ENTRADA_BI IS NOT
NULL AND FEC_ENTRADA_FC IS NOT NULL

                              --
************************************************************
                              --5.12.Actualizacion facturacion bajas y gestiones
anteriores    *
                              --
************************************************************
                              EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.12.ACTUALIZACION FACTURACION BAJAS Y GESTIONES ANTERIORES'
                              --5.12.1. Actualizacion facturacion bajas
                              EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.12.1.ACTUALIZACION FACTURACION BAJAS'
                              UPDATE A
                              SET A.FACT_TRIB= CASE WHEN A.ESTADO_TRIBUTOS IN (SELECT
Estado FROM COD.ESTADO WHERE FACT=1 AND SERVICIO='TRIBUTOS')
                                                                    THEN 1
                                                                    --WHEN
A.ESTADO_TRIBUTOS IN (SELECT Estado FROM COD.ESTADO WHERE FACT=2 AND
SERVICIO='TRIBUTOS')
                                                                    --THEN 2
                                                                    ELSE NULL
                                                              END,
                                    A.FEC_FACT_TRIB = CASE WHEN A.ESTADO_TRIBUTOS IN
(SELECT Estado FROM COD.ESTADO WHERE FACT=1 AND SERVICIO='TRIBUTOS')
                                                                    THEN @FCH_DATOS_1
                                                                    ELSE NULL
                                                              END
                              FROM CALC.AAM_MAESTRO_TRIB_VENTAS A
                                    WHERE A.ESTADO='BAJA' AND A.FACT_TRIB IS NULL AND
A.FEC_FACT_TRIB IS NULL AND (A.REVISAR_TRIB IS NULL OR A.REVISAR_TRIB NOT IN
('GESTION_ANTERIOR','REVISAR','REVISAR, HA PASADO DE KO A OK'))

                              --5.12.2 Actualización facturación nueva_gestión
terminada en el día (al no pasar de ko a ok no se factura, hay que hacerlo por
separado)
                              UPDATE A
                              SET FACT_TRIB=CASE WHEN A.ESTADO_TRIBUTOS IN (SELECT
Estado FROM COD.ESTADO WHERE FACT=1 AND SERVICIO='TRIBUTOS')
                                                                    THEN 1
                                                                    ELSE NULL
                                                              END,
                                    FEC_FACT_TRIB=CASE WHEN A.ESTADO_TRIBUTOS IN
(SELECT Estado FROM COD.ESTADO WHERE FACT=1 AND SERVICIO='TRIBUTOS')
                                                                    THEN @FCH_DATOS_1
                                                                    ELSE NULL
                                                              END
                              FROM CALC.AAM_MAESTRO_TRIB_VENTAS A
                              WHERE REVISAR_TRIB='NUEVA_GESTION'
                                    AND A.FACT_TRIB IS NULL AND A.FEC_FACT_TRIB IS
NULL

                              ---5.12.2.Facturación GESTION_ANTERIOR (BAJAS). 2 casos:
Previamente FACT=2 o FACT=0
```

```
                            EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.12.2.ACTUALIZACION FACTURACION BAJAS GESTIONES ANTERIORES'
                            --5.12.2.1.CASO 1
                            --UPDATE A
                            --SET A.FACT_TRIB = CASE WHEN A.ESTADO_TRIBUTOS IN
(SELECT Estado FROM COD.ESTADO WHERE FACT=1 AND SERVICIO='TRIBUTOS')
                            --                                      THEN 3 --Se factura la
parte de solicitado a OK
                            --                                      WHEN A.ESTADO_TRIBUTOS
IN (SELECT Estado FROM COD.ESTADO WHERE FACT=2 AND SERVICIO='TRIBUTOS')
                            --                                      THEN 0
                            --                                      ELSE 0
                            --                                END,
                            --     A.FEC_FACT_TRIB = @FCH_DATOS_1
                            --FROM CALC.AAM_MAESTRO_TRIB_VENTAS AS A
                            --WHERE A.ESTADO = 'BAJA'
                            --     AND (A.FACT_TRIB IS NULL OR A.FACT_TRIB = 0)
                            --     AND A.FEC_FACT_TRIB IS NULL
                            --     AND A.REVISAR_TRIB = 'GESTION_ANTERIOR'
                            --     AND EXISTS (
                            --             SELECT 1
                            --             FROM CALC.AAM_MAESTRO_TRIB_VENTAS AS B
                            --             WHERE A.CONCAT_SQL = B.CONCAT_SQL
                            --                    AND B.FACT_TRIB = 2
                            --                    AND B.FEC_SISTEMA < A.FEC_SISTEMA
                            --             ORDER BY B.FEC_SISTEMA DESC
                            --             OFFSET 0 ROWS
                            --             FETCH NEXT 1 ROWS ONLY
                            --     );
                            --5.12.2.2.CASO 2

                            --5.12.3 Facturación GESTION_ANTERIOR que pasa de KO a
OK. Los mismos 2 casos
                            --EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.12.3.ACTUALIZACION FACTURACION GESTIONES ANTERIORES NO BAJAS'
                            --5.12.3.1.CASO 1
                            --UPDATE A
                            --SET A.FACT_TRIB = 3,
                            --     A.FEC_FACT_TRIB =@FCH_DATOS_1
                            --FROM CALC.AAM_MAESTRO_TRIB_VENTAS A
                            --LEFT JOIN COD.ESTADO E ON A.ESTADO_TRIBUTOS = E.ESTADO
                            --WHERE (A.FACT_TRIB IS NULL OR A.FACT_TRIB = 0)
                            --     AND A.FEC_FACT_TRIB IS NULL
                            --     AND A.REVISAR_TRIB = 'GESTION_ANTERIOR'
                            --     AND E.OK_KO = 'OK'
                            --     AND EXISTS (
                            --             SELECT 1
                            --             FROM CALC.AAM_MAESTRO_TRIB_VENTAS AS B
                            --             WHERE B.CONCAT_SQL = A.CONCAT_SQL
                            --                    AND B.FACT_TRIB = 2
                            --                    AND B.FEC_SISTEMA < A.FEC_SISTEMA
                            --             ORDER BY B.FEC_SISTEMA DESC
                            --             OFFSET 0 ROWS
                            --             FETCH NEXT 1 ROWS ONLY
                            --     );
                            --5.12.3.2.CASO 2
```

```
                        --UPDATE A
                        --SET A.FACT_TRIB = 1,
                        --     A.FEC_FACT_TRIB = @FCH_DATOS_1
                        --FROM CALC.AAM_MAESTRO_TRIB_VENTAS A
                        --LEFT JOIN COD.ESTADO E ON A.ESTADO_TRIBUTOS = E.ESTADO
                        --WHERE A.FACT_TRIB IS NULL
                        --     AND A.FEC_FACT_TRIB IS NULL
                        --     AND A.REVISAR_TRIB = 'GESTION_ANTERIOR'
                        --     AND E.OK_KO = 'OK'
                        --     AND EXISTS (
                        --            SELECT 1
                        --            FROM CALC.AAM_MAESTRO_TRIB_VENTAS AS B
                        --            WHERE B.CONCAT_SQL = A.CONCAT_SQL
                        --                   AND B.FACT_TRIB IS NULL
                        --                   AND B.FEC_FICHERO < A.FEC_FICHERO
                        --            ORDER BY B.FEC_FICHERO DESC
                        --            OFFSET 0 ROWS
                        --            FETCH NEXT 1 ROWS ONLY
                        --     );


                        --
**********************************************************************
                        --5.13.Actualizacion columnas departamento, tiempos
medios y estado      *
                        --
**********************************************************************
                        EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.13.ACTUALIZACION DEPARTAMENTO, TIEMPOS MEDIOS Y ESTADOS'
                        --3.14.1.Rellenar la columna DEPARTAMENTO
                        UPDATE M
                               SET M.DEPARTAMENTO=D.DEPARTAMENTO
                        FROM CALC.AAM_MAESTRO_TRIB_VENTAS M
                        LEFT JOIN COD.CARTERAS B ON M.SOC_CLIENTE = B.CARTERA
                        LEFT JOIN COD.PARAM_CARTERAS C ON
B.ID_CARTERA=C.ID_CARTERA
                        LEFT JOIN COD.DEPARTAMENTOS D ON
C.ID_DEPARTAMENTO=D.ID_DEPARTAMENTO
                        WHERE M.DEPARTAMENTO IS NULL
                        --3.14.2.Rellenar columna Estado_trib
                        UPDATE A
                        SET ESTADO_TRIB=B.OK_KO
                        FROM CALC.AAM_MAESTRO_TRIB_VENTAS A
                        LEFT JOIN COD.ESTADO B ON A.ESTADO_TRIBUTOS=B.Estado
                        --3.14.3.Rellenar tiempos medios
                        UPDATE A
                        SET AGING_MEDIO_VENTA=
                               CASE
                                      WHEN FEC_ENTRADA_BI IS NOT NULL AND
FEC_ENTRADA_FC IS NOT NULL THEN DATEDIFF(DAY, IIF(FEC_ENTRADA_BI < FEC_ENTRADA_FC,
CAST(FEC_ENTRADA_BI AS DATE), CAST(FEC_ENTRADA_FC AS DATE)),FEC_POSICIONAMIENTO)
                                      WHEN FEC_ENTRADA_BI IS NOT NULL AND
FEC_ENTRADA_FC IS NULL THEN DATEDIFF(DAY, CAST(FEC_ENTRADA_BI AS
DATE),FEC_POSICIONAMIENTO)
```

```
                                        WHEN FEC_ENTRADA_FC IS NOT NULL AND
FEC_ENTRADA_BI IS NULL THEN DATEDIFF(DAY, CAST(FEC_ENTRADA_FC AS
DATE),FEC_POSICIONAMIENTO)
                                        ELSE NULL
                                END,
                                AGING_MEDIO_GESTION =
                                CASE
                                        WHEN FEC_INI_GESTION_TRIB IS NOT NULL AND
FEC_FACT_TRIB IS NOT NULL THEN DATEDIFF(DAY, FEC_INI_GESTION_TRIB,FEC_FACT_TRIB)
                                        ELSE NULL
                                END,
                                AGING_MEDIO_INICIO_GESTION =
                                CASE
                                        WHEN FEC_ENTRADA_BI IS NOT NULL AND
FEC_ENTRADA_FC IS NOT NULL THEN DATEDIFF(DAY, IIF(FEC_ENTRADA_BI < FEC_ENTRADA_FC,
CAST(FEC_ENTRADA_BI AS DATE), CAST(FEC_ENTRADA_FC AS DATE)),FEC_INI_GESTION_TRIB)
                                        WHEN FEC_ENTRADA_BI IS NOT NULL AND
FEC_ENTRADA_FC IS NULL THEN DATEDIFF(DAY, CAST(FEC_ENTRADA_BI AS
DATE),FEC_INI_GESTION_TRIB)
                                        WHEN FEC_ENTRADA_FC IS NOT NULL AND
FEC_ENTRADA_BI IS NULL THEN DATEDIFF(DAY, CAST(FEC_ENTRADA_FC AS
DATE),FEC_INI_GESTION_TRIB)
                                        ELSE NULL
                                END
                        FROM CALC.AAM_MAESTRO_TRIB_VENTAS A

                        --*********************************************
                        --5.14.Actualizacion operaciones de limpieza     *
                        --*********************************************
                        EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.14.ACTUALIZACION OPERACIONES DE LIMPIEZA'
                        ;
                        WITH CTE AS (
                                SELECT *, ROW_NUMBER() OVER (PARTITION BY
CONCAT_SQL, COALESCE(ID_OPORTUNIDAD,1) ORDER BY FEC_FICHERO DESC) AS RN
                                FROM CALC.AAM_MAESTRO_TRIB_VENTAS WHERE ESTADO <>
'BAJA' AND FEC_FICHERO<=@FCH_DATOS
                        )
                        DELETE FROM CTE
                        WHERE RN > 1;

                        UPDATE CALC.AAM_MAESTRO_TRIB_VENTAS
                        SET ESTADO_TRIBUTOS='PTE_GESTION'
                        WHERE ESTADO_TRIBUTOS='PTE_GESTIÓN'

                        EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '98.FIN GENERACIÓN MAESTRO TRIBUTOS'
                END TRY
                BEGIN CATCH
                        SET @message = '99.ERROR EN LA GENERACIÓN DEL MAESTRO
TRIB'
                                SET @error = ERROR_MESSAGE()
                                EXEC DQ.LOG 'ERROR', @cliente, @servicio,@hito,
@logtitulo, @message, @error
                END CATCH
        ELSE IF @SERVICIO=2
```

```
                    BEGIN TRY
                            --5.1.1.Actualizacion activos que pasan de OK a KO. Damos
de baja el OK del maestro para el posterior insert a "revisar, ha pasado de ok a ko"
                            UPDATE M
                            SET
                        M.ESTADO='BAJA'
                  FROM CALC.AAM_MAESTRO_CCPP_VENTAS M
                            INNER JOIN ##FOTO_ACTUAL_SERVICIO OPE ON
M.CONCAT_SQL=OPE.CONCAT_SQL
                  WHERE M.ESTADO<>'BAJA' AND OPE.ESTADO_COMUNIDADES IN (SELECT ESTADO
FROM COD.ESTADO WHERE OK_KO='KO')
                            AND M.ESTADO_COMUNIDADES IN (SELECT ESTADO FROM
COD.ESTADO WHERE OK_KO='OK')
                            --********************************
                            --5.1.2.Altas Ventas Especiales    *
                            --********************************
                            EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.1.ALTA VENTAS ESPECIALES Y ACTIVOS QUE PASAN DE OK A KO'
                            INSERT INTO CALC.AAM_MAESTRO_CCPP_VENTAS
                            SELECT
                                  OPE.FEC_FICHERO
                                  ,OPE.ID_OPORTUNIDAD
                                  ,OPE.CONCATENADO
                                  ,OPE.REF_UE
                                  ,OPE.UE
                                  ,OPE.EDIFICIO
                                  ,CAST(NULL AS NVARCHAR(50)) AS REF_ORIGEN
                                  ,CAST(NULL AS NVARCHAR(50)) AS SOC_CLIENTE
                                  ,OPE.TIPO_CLIENTE
                                  ,OPE.SOC_PROPIETARIA
                                  ,OPE.CM
                                  ,OPE.FEC_POSICIONAMIENTO
                                  ,CAST(NULL AS NVARCHAR(50)) AS ESTADO_DFA
                                  ,CAST(NULL AS DATE) AS FEC_PREVISTA_DFA
                                  ,OPE.PRIORIDAD_PATRIMONIO
                                  ,OPE.MAYORISTA_MINORISTA
                                  ,OPE.GESTOR_FORMALIZACION
                                  ,CAST(NULL AS NVARCHAR(50)) AS DELEGACION
                                  ,OPE.MUNICIPIO
                                  ,OPE.PROVINCIA
                                  ,OPE.GESTOR_NORMALIZADO
                                  ,CAST(NULL AS NVARCHAR(50)) AS TIPO_INMUEBLE
                                  ,OPE.SUBTIPOLOGIA
                                  ,OPE.REF_CATASTRAL
                                  ,OPE.NMB_COMPRADOR
                                  ,CAST(NULL AS DATE) AS FEC_RESERVA
                                  ,OPE.FEC_PREVISTA_FIRMA
                                  ,OPE.FP_AÑO
                                  ,OPE.FR_AÑO
                                  ,CAST(NULL AS NUMERIC(19,2)) AS PAO
                                  ,OPE.ESTADO_COMUNIDADES
                                  ,OPE.COD_PROMOCION
                                  ,OPE.CIF
                                  ,CASE WHEN OPE.CIF = 'NO CP' THEN 0 ELSE
LEN(OPE.CIF)-LEN(REPLACE(OPE.CIF, '/', ''))+1 END AS NUMERO_COMUNIDADES
                                  ,OPE.FEC_ENVIO_REMESA
```

```
                                ,OPE.ETI_REMESA
                                ,OPE.FEC_INI_GESTION_CCPP
                                ,OPE.FEC_ULT_GESTION_CCPP
                                ,OPE.GESTOR_CCPP
                                ,OPE.FP_MAIL
                                ,OPE.COMENTARIOS_CCPP
                                ,OPE.MIX_PAO_VBC_CCPP
                                ,OPE.MOTIVO_NO_FIRMA_CCPP
                                ,OPE.ALTA_PDTE
                                ,OPE.FEC_ALTA_PDTE
                                ,OPE.FEC_ALTA_FPF
                                ,OPE.ESTADO_CCPP
                                ,OPE.LOTE
                                ,OPE.FEC_REAL_CCPP
                                ,OPE.VENTA_BR_ID_DIR_CAR
                                ,OPE.PROMO_ACTIVO
                                ,CASE WHEN M.ESTADO='BAJA' AND
M.ESTADO_COMUNIDADES IN (SELECT ESTADO FROM COD.ESTADO WHERE OK_KO='OK') AND
OPE.ESTADO_COMUNIDADES IN (SELECT ESTADO FROM COD.ESTADO WHERE OK_KO='KO') THEN
'REVISAR, HA PASADO DE OK A KO'
                                      ELSE OPE.REVISAR_CCPP END AS REVISAR_CCPP
                                ,OPE.VENTA_ESPECIAL
                                ,OPE.POSTVENTA
                                ,OPE.CONCAT_SQL
                                ,OPE.CHECKSUM
                                ,OPE.ESTADO
                                ,OPE.FEC_SISTEMA
                                ,'OPERATIVO' AS ORIGEN
                                ,CAST(NULL AS DATE) AS FEC_ENTRADA_FC
                                ,CAST(NULL AS DATE) AS FEC_SALIDA_FC
                                ,CAST(NULL AS DATE) AS FEC_ENTRADA_BI
                                ,CAST(NULL AS DATE) AS FEC_SALIDA_BI
                                ,DAY(OPE.FEC_POSICIONAMIENTO) AS DIA_POS
                                ,MONTH(OPE.FEC_POSICIONAMIENTO) AS MES_POS
                                ,YEAR(OPE.FEC_POSICIONAMIENTO) AS AÑO_POS
                                ,CAST(NULL AS DATE) AS FEC_FACT_CCPP
                                ,NULL AS FACT_CCPP
                                ,NULL AS DEPARTAMENTO
                                ,NULL AS AGING_MEDIO_VENTA
                                ,NULL AS AGING_MEDIO_GESTION
                                ,NULL AS AGING_MEDIO_INICIO_GESTION
                        FROM ##FOTO_ACTUAL_SERVICIO OPE
                        LEFT JOIN (SELECT*FROM(
                                SELECT *,
                                      ROW_NUMBER()OVER(PARTITION BY CONCAT_SQL
ORDER BY FEC_FICHERO DESC) AS RN
                                FROM CALC.AAM_MAESTRO_CCPP_VENTAS)
                        X WHERE RN=1) M ON OPE.CONCAT_SQL=M.CONCAT_SQL
                        WHERE (M.CONCAT_SQL IS NULL AND OPE.VENTA_ESPECIAL='SI')
                                OR (M.CONCAT_SQL IS NOT NULL AND M.ESTADO='BAJA')
                        --*******************************
                        --5.2.Actualizacion facturacion    *
                        --*******************************
                        EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.2.ACTUALIZACION FACTURACION'
```

```
                              --5.2.1.Añadir las columnas de facturación del maestro
sobre el operativo
                              EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.2.1.ADICION COLUMNAS FACTURACION A LA FOTO ACTUAL DEL SERVICIO'
                              ALTER TABLE ##FOTO_ACTUAL_SERVICIO
                              ADD FEC_FACT_CCPP DATETIME,
                                     FACT_CCPP INT;
                              --5.2.2.Copiar sobre la temporal las columnas de
facturación
                              EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.2.2.IDENTIFICACION DE ACTIVOS YA FACTURADOS'
                              UPDATE O
                              SET O.FEC_FACT_CCPP = M.FEC_FACT_CCPP,
                                     O.FACT_CCPP = M.FACT_CCPP
                              FROM ##FOTO_ACTUAL_SERVICIO AS O
                              INNER JOIN CALC.AAM_MAESTRO_CCPP_VENTAS AS M ON
O.CONCAT_SQL = M.CONCAT_SQL
                              WHERE M.ESTADO<>'BAJA';

                              EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.2.3.ACTUALIZACION FACTURACION'
                              --5.2.3.Actualizacion activos que pasan de KO a OK
                              UPDATE OPE
                              SET OPE.FACT_CCPP= 1,
                                     OPE.FEC_FACT_CCPP = @FCH_DATOS_1
                              FROM ##FOTO_ACTUAL_SERVICIO OPE
                              LEFT JOIN COD.ESTADO E
                              ON OPE.ESTADO_COMUNIDADES = E.ESTADO
                              WHERE E.OK_KO='OK' AND E.FACT=1
                              AND (OPE.REVISAR_CCPP IS NULL OR
OPE.REVISAR_CCPP='NUEVA_GESTION')
                              AND OPE.FEC_FACT_CCPP IS NULL
                              AND OPE.FACT_CCPP IS NULL
                              AND EXISTS (
                                           SELECT 1
                                           FROM CALC.AAM_MAESTRO_CCPP_VENTAS M
                                           LEFT JOIN COD.ESTADO E2 ON
M.ESTADO_COMUNIDADES=E2.ESTADO
                                           WHERE OPE.CONCAT_SQL=M.CONCAT_SQL
                                                  AND E2.OK_KO='KO' AND
M.ESTADO<>'BAJA'
                              )

                              --*******************************
                              --5.3.Actualizacion operativo      *
                              --*******************************
                              EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.3.ACTUALIZACION OPERATIVO'
                              UPDATE M
                              SET  M.[ID_OPORTUNIDAD] = OPE.ID_OPORTUNIDAD
                                     ,M.[CONCATENADO] = OPE.CONCATENADO
                                     ,M.[REF_UE] = OPE.REF_UE
                                     ,M.[UE] = OPE.UE
                                     ,M.[EDIFICIO] = OPE.EDIFICIO
                                     ,M.[TIPO_CLIENTE] = OPE.TIPO_CLIENTE
                                     ,M.[SOC_PROPIETARIA] = OPE.SOC_PROPIETARIA
```

```
                                    ,M.[CM] = OPE.CM
                                    ,M.[FEC_POSICIONAMIENTO] = OPE.FEC_POSICIONAMIENTO
                                    ,M.[PRIORIDAD_PATRIMONIO] =
OPE.PRIORIDAD_PATRIMONIO
                                    ,M.[MAYORISTA_MINORISTA] = OPE.MAYORISTA_MINORISTA
                                    ,M.[GESTOR_FORMALIZACION] =
OPE.GESTOR_FORMALIZACION
                                    ,M.[MUNICIPIO] = OPE.MUNICIPIO
                                    ,M.[PROVINCIA] = OPE.PROVINCIA
                                    ,M.[GESTOR_NORMALIZADO] = OPE.GESTOR_NORMALIZADO
                                    ,M.[SUBTIPOLOGIA] = OPE.SUBTIPOLOGIA
                                    ,M.[REF_CATASTRAL] = OPE.REF_CATASTRAL
                                    ,M.[NMB_COMPRADOR] = OPE.NMB_COMPRADOR
                                    ,M.[FEC_PREVISTA_FIRMA] = OPE.FEC_PREVISTA_FIRMA
                                    ,M.[FP_AÑO] = OPE.FP_AÑO
                                    ,M.[FR_AÑO] = OPE.FR_AÑO
                                    ,M.[ESTADO_COMUNIDADES] = OPE.[ESTADO_COMUNIDADES]
                                    ,M.[COD_PROMOCION] = OPE.[COD_PROMOCION]
                                    ,M.[CIF] = OPE.[CIF]
                                    ,M.[FEC_ENVIO_REMESA] = OPE.[FEC_ENVIO_REMESA]
                                    ,M.[ETI_REMESA] = OPE.[ETI_REMESA]
                                    ,M.[FEC_INI_GESTION_CCPP] =
OPE.[FEC_INI_GESTION_CCPP]
                                    ,M.[FEC_ULT_GESTION_CCPP]
=OPE.[FEC_ULT_GESTION_CCPP]
                                    ,M.[GESTOR_CCPP] =OPE.[GESTOR_CCPP]
                                    ,M.[FP_MAIL] = OPE.[FP_MAIL]
                                    ,M.[COMENTARIOS_CCPP] = OPE.[COMENTARIOS_CCPP]
                                    ,M.[MIX_PAO_VBC_CCPP] = OPE.[MIX_PAO_VBC_CCPP]
                                    ,M.[MOTIVO_NO_FIRMA_CCPP] =
OPE.[MOTIVO_NO_FIRMA_CCPP]
                                    ,M.ALTA_PDTE=OPE.ALTA_PDTE
                                    ,M.FEC_ALTA_PDTE=OPE.FEC_ALTA_PDTE
                                    ,M.FEC_ALTA_FPF=OPE.FEC_ALTA_FPF
                                    ,M.ESTADO_CCPP=OPE.ESTADO_CCPP
                                    ,M.LOTE=OPE.LOTE
                                    ,M.FEC_REAL_CCPP=OPE.FEC_REAL_CCPP

       ,M.VENTA_BR_ID_DIRECCION_CARTERA=OPE.VENTA_BR_ID_DIR_CAR
                                    ,M.PROMO_ACTIVO=OPE.PROMO_ACTIVO
                                    ,M.REVISAR_CCPP=CASE WHEN M.REVISAR_CCPP='REVISAR,
HA PASADO DE OK A KO' AND M.FEC_ENTRADA_BI IS NULL AND M.FEC_ENTRADA_FC IS NULL THEN
M.REVISAR_CCPP ELSE OPE.REVISAR_CCPP END
                                    ,M.VENTA_ESPECIAL=OPE.VENTA_ESPECIAL
                                    ,M.POSTVENTA=OPE.POSTVENTA
                                    ,M.FEC_FICHERO=OPE.FEC_FICHERO
                                    ,M.CHECKSUM=OPE.CHECKSUM
                                    ,M.[ESTADO] = OPE.[ESTADO]
                                    ,M.FEC_SISTEMA=OPE.FEC_SISTEMA
                                    ,M.FEC_FACT_CCPP= OPE.FEC_FACT_CCPP
                                    ,M.FACT_CCPP = OPE.FACT_CCPP
                                    ,M.ORIGEN='OPERATIVO'
                               FROM CALC.AAM_MAESTRO_CCPP_VENTAS M
                               INNER JOIN ##FOTO_ACTUAL_SERVICIO OPE
                               ON OPE.CONCAT_SQL=M.CONCAT_SQL
                               WHERE M.ESTADO<>'BAJA'
```

```
                        --*************************
                        --5.4.Actualizacion bajas    *
                        --*************************
                        EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.4.ACTUALIZACION BAJAS'
                        --5.4.1.Bajas operativo (ventas especiales y postventas
fuera de fichero)
                        EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.4.1.ACTUALIZACION BAJAS OPERATIVO'
                        UPDATE M
                        SET
                               M.ESTADO=OPE.ESTADO
                        FROM CALC.AAM_MAESTRO_CCPP_VENTAS M
                        INNER JOIN ##FOTO_ACTUAL_SERVICIO_BAJAS OPE
                        ON OPE.CONCAT_SQL=M.CONCAT_SQL
                        WHERE M.ESTADO<>'BAJA'

                        --5.4.2.Bajas BI
                        EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.4.2.ACTUALIZACION BAJAS BI'
                        UPDATE M
                        SET
                               M.ESTADO=BI.ESTADO
                        FROM CALC.AAM_MAESTRO_CCPP_VENTAS M
                        INNER JOIN ##FOTO_ACTUAL_BI_BAJAS BI
                        ON BI.CONCAT_SQL=M.CONCAT_SQL
                        AND BI.ID_OPORTUNIDAD=M.ID_OPORTUNIDAD
                        WHERE M.ESTADO<>'BAJA'

                        --*************************************
                        --5.6.Actualizacion activos a revisar     *
                        --*************************************
                        EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.6.ACTUALIZACION ACTIVOS A REVISAR'
                        --5.6.1.Añadir sobre la temporal del BI las columnas de
gestión que faltan de OPE
                        EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.6.1.ADICION DE COLUMNAS DE GESTION A FOTO ACTUAL TEMPORAL'
                        ALTER TABLE ##FOTO_ACTUAL_BI
                        ADD NUMERO_COMUNIDADES INT,
                               REVISAR_CCPP NVARCHAR(50);
                                                    -- 5.6.2.Dar de baja lo que
sale como Alta en el BI y ya existía en Maestro (para luego poner a revisar)
                        UPDATE M
                        SET M.ESTADO='BAJA'
                        FROM CALC.AAM_MAESTRO_CCPP_VENTAS M
                        INNER JOIN ##FOTO_ACTUAL_BI BI ON
M.CONCAT_SQL=BI.CONCAT_SQL
                        WHERE (BI.ESTADO = 'ALTA' OR BI.ESTADO='REALTA')
                               AND BI.FEC_FICHERO = @FCH_DATOS AND
M.ESTADO<>'BAJA'
                        --5.6.2.Copiar las columnas de gestión sobre las altas
del BI que ya existían en el maestro (cualquier tipo de venta dada de baja)
                        EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.6.2.ACTUALIZACION DE COLUMNAS DE GESTION'
```

```
                    UPDATE BI
                SET
                    BI.ESTADO_COMUNIDADES = T.ESTADO_COMUNIDADES,
                    BI.COD_PROMOCION = T.COD_PROMOCION,
                    BI.CIF = T.CIF,
                    BI.NUMERO_COMUNIDADES = T.NUMERO_COMUNIDADES,
                    BI.FEC_ENVIO_REMESA = T.FEC_ENVIO_REMESA,
                    BI.ETI_REMESA = T.ETI_REMESA,
                    BI.FEC_INI_GESTION_CCPP = T.FEC_INI_GESTION_CCPP,
                    BI.FEC_ULT_GESTION_CCPP = T.FEC_ULT_GESTION_CCPP,
                    BI.GESTOR_CCPP = T.GESTOR_CCPP,
                    BI.FP_MAIL = T.FP_MAIL,
                    BI.COMENTARIOS_CCPP = T.COMENTARIOS_CCPP,
                    BI.MIX_PAO_VBC_CCPP = T.MIX_PAO_VBC_CCPP,
                    BI.MOTIVO_NO_FIRMA_CCPP = T.MOTIVO_NO_FIRMA_CCPP,
                    BI.REVISAR_CCPP = 'REVISAR'
                FROM ##FOTO_ACTUAL_BI BI
                INNER JOIN (SELECT *
                            FROM (
                                SELECT M.CONCAT_SQL,
                                    M.ESTADO_COMUNIDADES,
                                    M.COD_PROMOCION,
                                    M.CIF,
                                    M.NUMERO_COMUNIDADES,
                                    M.FEC_ENVIO_REMESA,
                                    M.ETI_REMESA,
                                    M.FEC_INI_GESTION_CCPP,
                                    M.FEC_ULT_GESTION_CCPP,
                                    M.GESTOR_CCPP,
                                    M.FP_MAIL,
                                    M.COMENTARIOS_CCPP,
                                    M.MIX_PAO_VBC_CCPP,
                                    M.MOTIVO_NO_FIRMA_CCPP,
                                                                M.ESTADO,
                                    ROW_NUMBER() OVER (PARTITION BY M.CONCAT_SQL ORDER
BY M.FEC_FICHERO DESC) AS RN
                                FROM CALC.AAM_MAESTRO_CCPP_VENTAS M
                                WHERE M.CONCAT_SQL IS NOT NULL
                            ) AS T WHERE T.RN=1 AND T.ESTADO='BAJA') T
                ON BI.CONCAT_SQL = T.CONCAT_SQL
                WHERE (BI.ESTADO = 'ALTA' OR BI.ESTADO='REALTA')
                AND BI.FEC_FICHERO = @FCH_DATOS

                            --*******************************************************
                            --5.7.Actualizacion altas BI dadas de baja en maestro    *
                            --*******************************************************
                            EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.7.ACTUALIZACION ALTAS BI DADAS DE BAJA EN MAESTRO'
                            INSERT INTO CALC.AAM_MAESTRO_CCPP_VENTAS
                            SELECT
                                   BI.FEC_FICHERO
                                   ,BI.ID_OPORTUNIDAD
                                   ,BI.CONCATENADO
                                   ,BI.REF_UE
                                   ,BI.UE
                                   ,BI.EDIFICIO
```

```
                                    ,BI.REF_ORIGEN
                                    ,BI.SOC_CLIENTE
                                    ,BI.TIPO_CLIENTE
                                    ,BI.SOC_PROPIETARIA
                                    ,BI.CM
                                    ,BI.FEC_POSICIONAMIENTO
                                    ,BI.ESTADO_DFA
                                    ,BI.FEC_PREVISTA_DFA
                                    ,BI.PRIORIDAD_PATRIMONIO
                                    ,BI.MAYORISTA_MINORISTA
                                    ,BI.GESTOR_FORMALIZACION
                                    ,BI.DELEGACION
                                    ,BI.MUNICIPIO
                                    ,BI.PROVINCIA
                                    ,BI.GESTOR_NORMALIZADO
                                    ,BI.TIPO_INMUEBLE
                                    ,BI.SUBTIPOLOGIA
                                    ,BI.REF_CATASTRAL
                                    ,BI.NMB_COMPRADOR
                                    ,BI.FEC_RESERVA
                                    ,BI.FEC_PREVISTA_FIRMA
                                    ,BI.FP_AÑO
                                    ,BI.FR_AÑO
                                    ,BI.PAO
                                    ,BI.ESTADO_COMUNIDADES
                                    ,BI.COD_PROMOCION
                                    ,BI.CIF
                                    ,BI.NUMERO_COMUNIDADES
                                    ,BI.FEC_ENVIO_REMESA
                                    ,BI.ETI_REMESA
                                    ,BI.FEC_INI_GESTION_CCPP
                                    ,BI.FEC_ULT_GESTION_CCPP
                                    ,BI.GESTOR_CCPP
                                    ,BI.FP_MAIL
                                    ,BI.COMENTARIOS_CCPP
                                    ,BI.MIX_PAO_VBC_CCPP
                                    ,BI.MOTIVO_NO_FIRMA_CCPP
                                    ,BI.ALTA_PDTE
                                    ,BI.FEC_ALTA_PDTE
                                    ,BI.FEC_ALTA_FPF
                                    ,NULL AS ESTADO_CCPP
                                    ,NULL AS LOTE
                                    ,COALESCE(BI.FP_MAIL, BI.FEC_POSICIONAMIENTO,
BI.FR_AÑO, BI.FP_AÑO) AS FEC_REAL_CCPP
                                    ,NULL AS VENTA_BR_ID_DIR_CAR
                                    ,NULL AS PROMO_ACTIVO
                                    ,BI.REVISAR_CCPP
                                    ,'NO' VENTA_ESPECIAL
                                    ,NULL AS POSTVENTA
                                    ,BI.CONCAT_SQL
                                    ,BI.CHECKSUM
                                    ,BI.ESTADO
                                    ,BI.FEC_SISTEMA
                                    ,'BI' AS ORIGEN
                                    ,CAST(NULL AS DATE) AS FEC_ENTRADA_FC
                                    ,CAST(NULL AS DATE) AS FEC_SALIDA_FC
```

```
                                ,CAST(NULL AS DATE) AS FEC_ENTRADA_BI
                                ,CAST(NULL AS DATE) AS FEC_SALIDA_BI
                                ,DAY(BI.FEC_POSICIONAMIENTO) AS DIA_POS
                                ,MONTH(BI.FEC_POSICIONAMIENTO) AS MES_POS
                                ,YEAR(BI.FEC_POSICIONAMIENTO) AS AÑO_POS
                                ,CAST(NULL AS DATE) AS FEC_FACT_CCPP
                                ,NULL AS FACT_CCPP
                                ,NULL AS DEPARTAMENTO
                                ,NULL AS AGING_MEDIO_VENTA
                                ,NULL AS AGING_MEDIO_GESTION
                                ,NULL AS AGING_MEDIO_INICIO_GESTION
                        FROM ##FOTO_ACTUAL_BI BI INNER JOIN
CALC.AAM_MAESTRO_CCPP_VENTAS M ON BI.CONCAT_SQL=M.CONCAT_SQL
                        WHERE (BI.ESTADO = 'ALTA' OR BI.ESTADO='REALTA') AND
BI.FEC_FICHERO=@FCH_DATOS
                                AND M.ESTADO='BAJA' AND M.CONCAT_SQL IS NOT NULL

                        --*************************************************
                        --5.8.Insercion altas BI no existentes en Maestro *
                        --*************************************************
                        EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.8.INSERCION ALTAS BI NO EXISTENTES EN MAESTRO'
                        INSERT INTO CALC.AAM_MAESTRO_CCPP_VENTAS
                        SELECT
                                BI.FEC_FICHERO
                                ,BI.ID_OPORTUNIDAD
                                ,BI.CONCATENADO
                                ,BI.REF_UE
                                ,BI.UE
                                ,BI.EDIFICIO
                                ,BI.REF_ORIGEN
                                ,BI.SOC_CLIENTE
                                ,BI.TIPO_CLIENTE
                                ,BI.SOC_PROPIETARIA
                                ,BI.CM
                                ,BI.FEC_POSICIONAMIENTO
                                ,BI.ESTADO_DFA
                                ,BI.FEC_PREVISTA_DFA
                                ,BI.PRIORIDAD_PATRIMONIO
                                ,BI.MAYORISTA_MINORISTA
                                ,BI.GESTOR_FORMALIZACION
                                ,BI.DELEGACION
                                ,BI.MUNICIPIO
                                ,BI.PROVINCIA
                                ,BI.GESTOR_NORMALIZADO
                                ,BI.TIPO_INMUEBLE
                                ,BI.SUBTIPOLOGIA
                                ,BI.REF_CATASTRAL
                                ,BI.NMB_COMPRADOR
                                ,BI.FEC_RESERVA
                                ,BI.FEC_PREVISTA_FIRMA
                                ,BI.FP_AÑO
                                ,BI.FR_AÑO
                                ,BI.PAO
                                ,BI.ESTADO_COMUNIDADES
                                ,BI.COD_PROMOCION
```

```
                                    ,BI.CIF
                                    ,CASE WHEN BI.CIF = 'NO CP' OR BI.CIF='0' THEN 0
ELSE LEN(BI.CIF)-LEN(REPLACE(BI.CIF, '/', ''))+1 END AS NUMERO_COMUNIDADES
                                    ,BI.FEC_ENVIO_REMESA
                                    ,BI.ETI_REMESA
                                    ,BI.FEC_INI_GESTION_CCPP
                                    ,BI.FEC_ULT_GESTION_CCPP
                                    ,BI.GESTOR_CCPP
                                    ,BI.FP_MAIL
                                    ,BI.COMENTARIOS_CCPP
                                    ,BI.MIX_PAO_VBC_CCPP
                                    ,BI.MOTIVO_NO_FIRMA_CCPP
                                    ,BI.ALTA_PDTE
                                    ,BI.FEC_ALTA_PDTE
                                    ,BI.FEC_ALTA_FPF
                                    ,NULL AS ESTADO_CCPP
                                    ,NULL AS LOTE
                                    ,COALESCE(BI.FP_MAIL, BI.FEC_POSICIONAMIENTO,
BI.FR_AÑO, BI.FP_AÑO) AS FEC_REAL_CCPP
                                    ,NULL AS VENTA_BR_ID_DIR_CAR
                                    ,NULL AS PROMO_ACTIVO
                                    ,NULL AS REVISAR_CCPP
                                    ,'NO' AS VENTA_ESPECIAL
                                    ,NULL AS POSTVENTA
                                    ,BI.CONCAT_SQL
                                    ,BI.CHECKSUM
                                    ,BI.ESTADO
                                    ,BI.FEC_SISTEMA
                                    ,'BI' AS ORIGEN
                                    ,CAST(NULL AS DATE) AS FEC_ENTRADA_FC
                                    ,CAST(NULL AS DATE) AS FEC_SALIDA_FC
                                    ,CAST(NULL AS DATE) AS FEC_ENTRADA_BI
                                    ,CAST(NULL AS DATE) AS FEC_SALIDA_BI
                                    ,DAY(BI.FEC_POSICIONAMIENTO) AS DIA_POS
                                    ,MONTH(BI.FEC_POSICIONAMIENTO) AS MES_POS
                                    ,YEAR(BI.FEC_POSICIONAMIENTO) AS AÑO_POS
                                    ,CAST(NULL AS DATE) AS FEC_FACT_CCPP
                                    ,NULL AS FACT_CCPP
                                    ,NULL AS DEPARTAMENTO
                                    ,NULL AS AGING_MEDIO_VENTA
                                    ,NULL AS AGING_MEDIO_GESTION
                                    ,NULL AS AGING_MEDIO_INICIO_GESTION
                            FROM ##FOTO_ACTUAL_BI BI
                            LEFT JOIN CALC.AAM_MAESTRO_CCPP_VENTAS M
                            ON BI.CONCAT_SQL=M.CONCAT_SQL
                            WHERE M.CONCAT_SQL IS NULL

                            --********************************
                            --5.9.Actualizacion campos BI      *
                            --********************************
                            EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.9.ACTUALIZACION CAMPOS BI'
                            UPDATE M
                            SET
                                    M.ID_OPORTUNIDAD=BI.ID_OPORTUNIDAD
                                    ,M.CONCATENADO=BI.CONCATENADO
```

```
                                    ,M.REF_UE=BI.REF_UE
                                    ,M.UE=BI.UE
                                    ,M.EDIFICIO=BI.EDIFICIO
                                    ,M.REF_ORIGEN=BI.REF_ORIGEN
                                    ,M.SOC_CLIENTE=BI.SOC_CLIENTE
                                    ,M.TIPO_CLIENTE=BI.TIPO_CLIENTE
                                    ,M.SOC_PROPIETARIA=BI.SOC_PROPIETARIA
                                    ,M.CM=BI.CM
                                    ,M.FEC_POSICIONAMIENTO=BI.FEC_POSICIONAMIENTO
                                    ,M.ESTADO_DFA=BI.ESTADO_DFA
                                    ,M.FEC_PREVISTA_DFA=BI.FEC_PREVISTA_DFA
                                    ,M.PRIORIDAD_PATRIMONIO=BI.PRIORIDAD_PATRIMONIO
                                    ,M.MAYORISTA_MINORISTA=BI.MAYORISTA_MINORISTA
                                    ,M.GESTOR_FORMALIZACION=BI.GESTOR_FORMALIZACION
                                    ,M.DELEGACION=BI.DELEGACION
                                    ,M.MUNICIPIO=BI.MUNICIPIO
                                    ,M.PROVINCIA=BI.PROVINCIA
                                    ,M.GESTOR_NORMALIZADO=BI.GESTOR_NORMALIZADO
                                    ,M.TIPO_INMUEBLE=BI.TIPO_INMUEBLE
                                    ,M.SUBTIPOLOGIA=BI.SUBTIPOLOGIA
                                    ,M.REF_CATASTRAL=BI.REF_CATASTRAL
                                    ,M.NMB_COMPRADOR=BI.NMB_COMPRADOR
                                    ,M.FEC_RESERVA=BI.FEC_RESERVA
                                    ,M.FEC_PREVISTA_FIRMA=BI.FEC_PREVISTA_FIRMA
                                    ,M.FP_AÑO=BI.FP_AÑO
                                    ,M.FR_AÑO=BI.FR_AÑO
                                    ,M.PAO=BI.PAO
                                    ,M.ALTA_PDTE=BI.ALTA_PDTE
                                    ,M.FEC_ALTA_PDTE=BI.FEC_ALTA_PDTE
                                    ,M.FEC_ALTA_FPF=BI.FEC_ALTA_FPF
                                    ,M.DIA_POS=DAY(BI.FEC_POSICIONAMIENTO)
                                    ,M.MES_POS=MONTH(BI.FEC_POSICIONAMIENTO)
                                    ,M.AÑO_POS=YEAR(BI.FEC_POSICIONAMIENTO)
                                FROM CALC.AAM_MAESTRO_CCPP_VENTAS M
                                INNER JOIN ##FOTO_ACTUAL_BI BI
                                ON M.CONCAT_SQL=BI.CONCAT_SQL AND
M.ID_OPORTUNIDAD=BI.ID_OPORTUNIDAD

                                --*******************************
                                --5.10.Actualizacion postventas    *
                                --*******************************
                                EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.10.ACTUALIZACION POSTVENTAS'
                                --5.10.1.Postventa presente en el BI
                                EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.10.1.ACTUALIZACION POSTVENTAS PRESENTES EN BI'
                                UPDATE CALC.AAM_MAESTRO_CCPP_VENTAS
                                SET POSTVENTA=NULL
                                WHERE (PRIORIDAD_PATRIMONIO<>'V' OR PRIORIDAD_PATRIMONIO
IS NULL)
                                UPDATE CALC.AAM_MAESTRO_CCPP_VENTAS
                                SET POSTVENTA='SI'
                                WHERE PRIORIDAD_PATRIMONIO='V'
                                AND (POSTVENTA IS NULL OR POSTVENTA<>'SI, NO PRESENTE EN
EL BI') ;
                                --5.10.2.Postventa no presente en el BI
```

```
                              EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.10.2.ACTUALIZACION POSTVENTAS NO PRESENTES EN BI'
                              UPDATE A
                              SET A.POSTVENTA='SI, NO PRESENTE EN EL BI',
                                     A.ESTADO='ALTA'
                              FROM CALC.AAM_MAESTRO_CCPP_VENTAS A
                              WHERE A.PRIORIDAD_PATRIMONIO='V'
                                     AND A.ESTADO_COMUNIDADES IN (
                                     SELECT ESTADO
                                     FROM COD.ESTADO
                                     WHERE OK_KO = 'KO' AND SERVICIO='COMUNIDADES'
                                     )
                                     AND A.CONCAT_SQL IN(
                                            SELECT
                                                   CONCAT_SQL
                                            FROM ##FOTO_ACTUAL_BI_BAJAS
                                     );



                              --*********************************************
                              --5.11.Actualizacion fecha de entrada y salida    *
                              --*********************************************
                              EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.11.ACTUALIZACION FECHA ENTRADA Y SALIDA'
                              UPDATE CALC.AAM_MAESTRO_CCPP_VENTAS
                              SET FEC_ENTRADA_FC = CASE WHEN (VENTA_ESPECIAL='SI' OR
POSTVENTA='SI, NO PRESENTE EN EL BI') AND FEC_ENTRADA_FC IS NULL THEN @FCH_DATOS_1
                                                                                 ELSE
FEC_ENTRADA_FC END,
                                     FEC_ENTRADA_BI = CASE WHEN (VENTA_ESPECIAL='NO'
AND (POSTVENTA IS NULL OR POSTVENTA ='SI')) AND FEC_ENTRADA_BI IS NULL THEN @FCH_DATOS
                                                                                 ELSE
FEC_ENTRADA_BI END,
                                     FEC_SALIDA_FC = CASE WHEN (VENTA_ESPECIAL='SI' OR
POSTVENTA='SI, NO PRESENTE EN EL BI') AND FEC_SALIDA_FC IS NULL AND ESTADO='BAJA' THEN
@FCH_DATOS_1
                                                                                 ELSE
FEC_SALIDA_FC END,
                                     FEC_SALIDA_BI = CASE WHEN (VENTA_ESPECIAL='NO' AND
(POSTVENTA IS NULL OR POSTVENTA ='SI')) AND FEC_SALIDA_BI IS NULL AND ESTADO='BAJA'
THEN @FCH_DATOS
                                                                                 ELSE
FEC_SALIDA_BI END

                              UPDATE CALC.AAM_MAESTRO_CCPP_VENTAS
                              SET FEC_SALIDA_FC = CASE WHEN
FEC_ENTRADA_FC<FEC_ENTRADA_BI THEN FEC_ENTRADA_BI
                                                                                 ELSE NULL END
                              WHERE FEC_SALIDA_FC IS NULL AND FEC_ENTRADA_BI IS NOT
NULL AND FEC_ENTRADA_FC IS NOT NULL


                              --
************************************************************
```

```
                              --5.12.Actualizacion facturacion bajas y gestiones
anteriores    *
                              --
************************************************************
                              EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.12.ACTUALIZACION FACTURACION BAJAS Y GESTIONES ANTERIORES'
                              --5.12.1. Actualizacion facturacion bajas
                              EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.12.1.ACTUALIZACION FACTURACION BAJAS'
                              UPDATE A
                              SET A.FACT_CCPP= CASE WHEN A.ESTADO_COMUNIDADES IN
(SELECT Estado FROM COD.ESTADO WHERE FACT=1 AND SERVICIO='CCPP')
                                                                    THEN 1
                                                                    --WHEN
A.ESTADO_COMUNIDADES IN (SELECT Estado FROM COD.ESTADO WHERE FACT=2 AND
SERVICIO='CCPP')  --TOCAR SI SE QUIERE AÑADIR FACTURACION PARCIAL
                                                                    --THEN 2
                                                                    ELSE NULL
                                                              END,
                                    A.FEC_FACT_CCPP = CASE WHEN A.ESTADO_COMUNIDADES
IN (SELECT Estado FROM COD.ESTADO WHERE FACT=1 AND SERVICIO='CCPP')
                                                                    THEN @FCH_DATOS_1
                                                                    ELSE NULL
                                                              END
                              FROM CALC.AAM_MAESTRO_CCPP_VENTAS A
                              WHERE A.ESTADO='BAJA'
                                    AND A.FACT_CCPP IS NULL AND A.FEC_FACT_CCPP IS
NULL
                                    AND (A.REVISAR_CCPP IS NULL OR A.REVISAR_CCPP NOT
IN ('GESTION_ANTERIOR','REVISAR','REVISAR, HA PASADO DE KO A OK'))

                              --5.12.2 Actualización facturación nueva_gestión
terminada en el día (al no pasar de ko a ok no se factura, hay que hacerlo por
separado)
                              UPDATE A
                              SET FACT_CCPP=CASE WHEN A.ESTADO_COMUNIDADES IN (SELECT
Estado FROM COD.ESTADO WHERE FACT=1 AND SERVICIO='CCPP')
                                                                    THEN 1
                                                                    ELSE NULL
                                                              END,
                                    FEC_FACT_CCPP=CASE WHEN A.ESTADO_COMUNIDADES IN
(SELECT Estado FROM COD.ESTADO WHERE FACT=1 AND SERVICIO='CCPP')
                                                                    THEN @FCH_DATOS_1
                                                                    ELSE NULL
                                                              END
                              FROM CALC.AAM_MAESTRO_CCPP_VENTAS A
                              WHERE REVISAR_CCPP='NUEVA_GESTION'
                                    AND A.FACT_CCPP IS NULL AND A.FEC_FACT_CCPP IS
NULL

                              --5.12.2.Facturación GESTION_ANTERIOR (BAJAS). 2 casos:
Previamente FACT=2 o FACT=0
                              EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.12.2.ACTUALIZACION FACTURACION BAJAS GESTIONES ANTERIORES'
                              --5.12.2.1.CASO 1
                              --UPDATE A
```

```
                              --SET A.FACT_CCPP = CASE WHEN A.ESTADO_COMUNIDADES IN
(SELECT Estado FROM COD.ESTADO WHERE FACT=1 AND SERVICIO='CCPP')
                              --                                    THEN 3--Se factura la
facturacion al completo (3 si se quiere facturar parcial)
                              --                                    WHEN
A.ESTADO_COMUNIDADES IN (SELECT Estado FROM COD.ESTADO WHERE FACT=2 AND
SERVICIO='CCPP')
                              --                                    THEN 0
                              --                                    ELSE 0
                              --                               END,
                              --     A.FEC_FACT_CCPP = @FCH_DATOS_1
                              --FROM CALC.AAM_MAESTRO_CCPP_VENTAS AS A
                              --WHERE A.ESTADO = 'BAJA'
                              --     AND (A.FACT_CCPP IS NULL OR A.FACT_CCPP = 0)
                              --     AND A.FEC_FACT_CCPP IS NULL
                              --     AND A.REVISAR_CCPP = 'GESTION_ANTERIOR'
                              --     AND EXISTS (
                              --            SELECT 1
                              --            FROM CALC.AAM_MAESTRO_CCPP_VENTAS AS B
                              --            WHERE A.CONCAT_SQL = B.CONCAT_SQL
                              --                   AND B.FACT_CCPP = 2 -- En el momento
que se da de alta se copia el estado anterior y es FACT=2
                              --                   AND B.FEC_SISTEMA < A.FEC_SISTEMA
                              --            ORDER BY B.FEC_SISTEMA DESC
                              --            OFFSET 0 ROWS
                              --            FETCH NEXT 1 ROWS ONLY
                              --     );

                              --5.12.2.2.CASO 2


                              --5.12.3 Facturación GESTION_ANTERIOR que pasa de KO a
OK. Los mismos 2 casos
                              --EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.12.3.ACTUALIZACION FACTURACION GESTIONES ANTERIORES NO BAJAS'
                              --5.12.3.1.CASO 1
                              --UPDATE A
                              --SET A.FACT_CCPP = 3,
                              --     A.FEC_FACT_CCPP = @FCH_DATOS_1
                              --FROM CALC.AAM_MAESTRO_CCPP_VENTAS A
                              --LEFT JOIN COD.ESTADO E ON A.ESTADO_COMUNIDADES =
E.ESTADO
                              --WHERE (A.FACT_CCPP IS NULL OR A.FACT_CCPP = 0)
                              --     AND A.FEC_FACT_CCPP IS NULL
                              --     AND A.REVISAR_CCPP = 'GESTION_ANTERIOR'
                              --     AND E.OK_KO = 'OK'
                              --     AND EXISTS (
                              --            SELECT 1
                              --            FROM CALC.AAM_MAESTRO_CCPP_VENTAS AS B
                              --            WHERE B.CONCAT_SQL = A.CONCAT_SQL
                              --                   AND B.FACT_CCPP = 2
                              --                   AND B.FEC_SISTEMA < A.FEC_SISTEMA
                              --            ORDER BY B.FEC_SISTEMA DESC
                              --            OFFSET 0 ROWS
                              --            FETCH NEXT 1 ROWS ONLY
                              --     );
```

```
                            --5.12.3.2.CASO 2
                            --UPDATE A
                            --SET A.FACT_CCPP = 1,
                            --     A.FEC_FACT_CCPP = @FCH_DATOS_1
                            --FROM CALC.AAM_MAESTRO_CCPP_VENTAS A
                            --LEFT JOIN COD.ESTADO E ON A.ESTADO_COMUNIDADES =
E.ESTADO
                            --WHERE A.FACT_CCPP IS NULL
                            --     AND A.FEC_FACT_CCPP IS NULL
                            --     AND A.REVISAR_CCPP = 'GESTION_ANTERIOR'
                            --     AND E.OK_KO = 'OK'
                            --     AND EXISTS (
                            --             SELECT 1
                            --             FROM CALC.AAM_MAESTRO_CCPP_VENTAS AS B
                            --             WHERE B.CONCAT_SQL = A.CONCAT_SQL
                            --                    AND B.FACT_CCPP IS NULL
                            --                    AND B.FEC_FICHERO < A.FEC_FICHERO
                            --             ORDER BY B.FEC_FICHERO DESC
                            --             OFFSET 0 ROWS
                            --             FETCH NEXT 1 ROWS ONLY
                            --     );

                            --
**********************************************************************
                            --5.13.Actualizacion columnas departamento, tiempos
medios y estado      *
                            --
**********************************************************************
                            EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.13.ACTUALIZACION DEPARTAMENTO, TIEMPOS MEDIOS Y ESTADOS'

                            UPDATE M
                            SET M.SOC_CLIENTE='SANTANDER'
                            FROM CALC.AAM_MAESTRO_CCPP_VENTAS M
                            WHERE (TIPO_CLIENTE IN ('BSAN','SANTANDER')
                                   OR SOC_PROPIETARIA IN ('SANTANDER', 'ALTAMIRA'))
                            AND SOC_CLIENTE IS NULL

                            UPDATE M
                            SET M.SOC_CLIENTE=BI.SOC_CLIENTE
                            FROM CALC.AAM_MAESTRO_CCPP_VENTAS M
                            INNER JOIN (SELECT
                                                DISTINCT TIPO_CLIENTE,
                                                SOC_CLIENTE
                                        FROM HIS.AAM_BI_VENTAS) BI
                            ON M.TIPO_CLIENTE=BI.TIPO_CLIENTE
                            WHERE M.SOC_CLIENTE IS NULL

                            UPDATE M
                            SET M.DEPARTAMENTO=D.DEPARTAMENTO
                            FROM CALC.AAM_MAESTRO_CCPP_VENTAS M
                            LEFT JOIN COD.CARTERAS B ON M.SOC_CLIENTE = B.CARTERA
                            LEFT JOIN COD.PARAM_CARTERAS C ON
B.ID_CARTERA=C.ID_CARTERA
                            LEFT JOIN COD.DEPARTAMENTOS D ON
C.ID_DEPARTAMENTO=D.ID_DEPARTAMENTO
```

```
                        WHERE M.DEPARTAMENTO IS NULL

                        UPDATE M
                        SET M.DEPARTAMENTO='TP'
                        FROM CALC.AAM_MAESTRO_CCPP_VENTAS M
                        WHERE M.DEPARTAMENTO IS NULL
                        AND (M.SOC_PROPIETARIA='TP' OR M.TIPO_CLIENTE='TP')

                        UPDATE A
                        SET A.AGING_MEDIO_VENTA =
                                CASE
                                        WHEN FEC_ENTRADA_BI IS NOT NULL AND
FEC_ENTRADA_FC IS NOT NULL THEN DATEDIFF(DAY, IIF(FEC_ENTRADA_BI < FEC_ENTRADA_FC,
CAST(FEC_ENTRADA_BI AS DATE), CAST(FEC_ENTRADA_FC AS DATE)),FEC_POSICIONAMIENTO)
                                        WHEN FEC_ENTRADA_BI IS NOT NULL AND
FEC_ENTRADA_FC IS NULL THEN DATEDIFF(DAY, CAST(FEC_ENTRADA_BI AS
DATE),FEC_POSICIONAMIENTO)
                                        WHEN FEC_ENTRADA_FC IS NOT NULL AND
FEC_ENTRADA_BI IS NULL THEN DATEDIFF(DAY, CAST(FEC_ENTRADA_FC AS
DATE),FEC_POSICIONAMIENTO)
                                        ELSE NULL
                                END,
                            A.AGING_MEDIO_GESTION =
                                CASE
                                        WHEN FEC_INI_GESTION_CCPP IS NOT
NULL AND FEC_FACT_CCPP IS NOT NULL THEN DATEDIFF(DAY,
FEC_INI_GESTION_CCPP,FEC_FACT_CCPP)
                                        ELSE NULL
                                END,
                            A.AGING_MEDIO_INICIO_GESTION =
                                CASE
                                        WHEN FEC_ENTRADA_BI IS NOT NULL AND
FEC_ENTRADA_FC IS NOT NULL THEN DATEDIFF(DAY, IIF(FEC_ENTRADA_BI < FEC_ENTRADA_FC,
CAST(FEC_ENTRADA_BI AS DATE), CAST(FEC_ENTRADA_FC AS DATE)),FEC_INI_GESTION_CCPP)
                                        WHEN FEC_ENTRADA_BI IS NOT NULL AND
FEC_ENTRADA_FC IS NULL THEN DATEDIFF(DAY, CAST(FEC_ENTRADA_BI AS
DATE),FEC_INI_GESTION_CCPP)
                                        WHEN FEC_ENTRADA_FC IS NOT NULL AND
FEC_ENTRADA_BI IS NULL THEN DATEDIFF(DAY, CAST(FEC_ENTRADA_FC AS
DATE),FEC_INI_GESTION_CCPP)
                                        ELSE NULL
                                END
                        FROM CALC.AAM_MAESTRO_CCPP_VENTAS A;

                        UPDATE A
                        SET ESTADO_CCPP=B.OK_KO
                        FROM CALC.AAM_MAESTRO_CCPP_VENTAS A
                        LEFT JOIN COD.ESTADO B ON A.ESTADO_COMUNIDADES=B.Estado

                        --*********************************************
                        --5.14.Actualizacion operaciones de limpieza     *
                        --*********************************************
                        EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '5.14.ACTUALIZACION OPERACIONES DE LIMPIEZA'
                        UPDATE CALC.AAM_MAESTRO_CCPP_VENTAS
                        SET CIF=NULL, NUMERO_COMUNIDADES=NULL
```

```
                              WHERE CIF='0'
                              UPDATE CALC.AAM_MAESTRO_CCPP_VENTAS
                              SET COD_PROMOCION=NULL
                              WHERE COD_PROMOCION='0'

                              ;WITH CTE2 AS (
                                     SELECT *, ROW_NUMBER() OVER (PARTITION BY
CONCAT_SQL, COALESCE(ID_OPORTUNIDAD,1) ORDER BY FEC_FICHERO DESC) AS RN
                                     FROM CALC.AAM_MAESTRO_CCPP_VENTAS WHERE ESTADO <>
'BAJA' AND FEC_FICHERO<=@FCH_DATOS
                              )
                              DELETE FROM CTE2
                              WHERE RN > 1;

                              EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito,
@logtitulo, '98.FIN GENERACION DE MAESTRO VENTAS CCPP'
                       END TRY
                       BEGIN CATCH
                              SET @message = '99.ERROR EN LA GENERACIÓN DEL MAESTRO
CCPP'
                              SET @error = ERROR_MESSAGE()
                              EXEC DQ.LOG 'ERROR', @cliente, @servicio,@hito,
@logtitulo, @message, @error
                       END CATCH
       END TRY
       BEGIN CATCH
              SET @message = '99.ERROR EN LA GENERACION DEL MAESTRO'
              SET @error = ERROR_MESSAGE()
              EXEC DQ.LOG 'ERROR', @cliente, @servicio,@hito, @logtitulo, @message,
@error
              DROP TABLE IF EXISTS ##FOTO_ACTUAL_BI
              DROP TABLE IF EXISTS ##FOTO_ACTUAL_BI_BAJAS
              DROP TABLE IF EXISTS ##FOTO_ACTUAL_SERVICIO
              DROP TABLE IF EXISTS ##FOTO_ACTUAL_SERVICIO_BAJAS
       END CATCH

       --#####################################################
       --#    6.EVOLUCION DE VENTAS: ALTAS, BAJAS, PERMANECEN #
       --#####################################################
       EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito, @logtitulo, '6.RELLENO TABLA
EVOLUCION VENTAS'

       --Cambiar, las bajas y altas de ventas especiales y postventas fuera de fichero
se registran en @fch_datos_1
       SET @sql='INSERT INTO CALC.EVOLUCION_VENTAS (ID_CLIENTE,ID_SERVICIO,ID_HITO,
FEC_DATOS,ENTRADAS,SALIDAS,PERMANECEN)
                     SELECT '+CAST(@CLIENTE AS VARCHAR)+' AS ID_CLIENTE,
                              '+CAST(@SERVICIO AS VARCHAR)+' AS ID_SERVICIO,
                                '+CAST(@HITO AS VARCHAR)+' AS ID_HITO,
                                '''+CAST(@FCH_DATOS AS NVARCHAR)+''' AS
FEC_DATOS,
                                SUM(CASE WHEN ISNULL(FEC_ENTRADA_BI,''19000101'')
= '''+CAST(@FCH_DATOS AS VARCHAR)+''' THEN 1 ELSE 0 END) AS ENTRADAS,
                                SUM(CASE WHEN ISNULL(FEC_SALIDA_BI,''19000101'')
= '''+CAST(@FCH_DATOS AS VARCHAR)+''' THEN 1 ELSE 0 END) AS SALIDAS,
```

```
                                    SUM(CASE WHEN ESTADO<>''BAJA'' AND
VENTA_ESPECIAL=''NO'' AND (POSTVENTA IS NULL OR POSTVENTA=''SI'') THEN 1 ELSE 0 END)-
SUM(CASE WHEN ISNULL(FEC_ENTRADA_BI,''19000101'') = '''+CAST(@FCH_DATOS AS
VARCHAR)+''' THEN 1 ELSE 0 END) AS PERMANECEN
                    FROM CALC.AAM_MAESTRO_'+@SERV+'_VENTAS'

       EXEC sp_executesql @sql

       --#####################################################
       --#    7.ACTUALIZACION TABLA REP.AAM_FACTURACION_VENTAS #
       --#####################################################
       EXEC DQ.LOG 'INFO', @cliente, @servicio,@hito, @logtitulo, '7.ACTUALIZACION
TABLA REP.AAM_FACTURACION_VENTAS'
       EXEC REP.ACTUALIZAR_FACTURACION_VENTAS @cliente, @servicio, @hito, @FCH_DATOS_1
END
```

DQ.ORQUESTADOR_VALIDACIONES

```
USE [AAM_SERVICIO_CCPP_TRIBUTOS]
GO
/****** Object:  StoredProcedure [DQ].[ORQUESTADOR_VALIDACIONES]    Script Date:
18/06/2024 16:45:32 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
/*************************************************/
-- Autor: Ángel Guzmán
-- Fecha: 20240213
-- Descripción: Proceso que valida diferentes campos, orquestador para ejecutar los
diferentes procedimientos de validación
--             Tipo de validaciones:
--                    Negocio
--                    Campos Obligatorios
--             Resultado de validaciones:
--                    DQ.VAL_CAMPOS_OBLIGATORIOS
--                    DQ.VAL_NEGOCIO
--
-- Versiones:
--             V1: Versión inicial
/*************************************************/
ALTER PROCEDURE [DQ].[ORQUESTADOR_VALIDACIONES] (
       @Cliente int,
       @Servicio int,
       @Hito int
)
AS
BEGIN
       BEGIN TRY
              DECLARE @logMessage NVARCHAR(500);
        DECLARE @logtitulo NVARCHAR(255);

              SET @logtitulo='VALIDACION'
              --############################
              -- 0.DECLARACION DE VARIABLES #
              --############################
```

```
        EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logtitulo, '0.INICIO
VALIDACION'

              --#########################
              -- 1.VALIDACION DUPLICADOS #
              --#########################
              EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo,
'1.VALIDACION DUPLICADOS'
              EXEC DQ.VALIDACION_DUPLICADOS @Cliente, @Servicio, @Hito

              --######################
              -- 2.VALIDACION ESTADO #
              --######################
              EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo,
'2.VALIDACION ESTADO'
              IF @Cliente=1 and @Hito=1
              BEGIN
                     EXEC DQ.VALIDACION_ESTADOS @Cliente, @Servicio, @Hito
              END

              --#########################
              -- 3.VALIDACION TIPO_VENTA #
              --#########################
              EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo,
'3.VALIDACION TIPO VENTA'
              IF @Cliente=1 and @Hito=1
              BEGIN
                     EXEC DQ.VALIDACION_TIPO_VENTA @Cliente, @Servicio, @Hito
              END
              --###################################
              -- 4.VALIDACION CAMPOS OBLIGATORIOS #
              --###################################
              EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo,
'4.VALIDACION CAMPOS OBLIGATORIOS'
              EXEC DQ.VALIDACION_CAMPOS_OBLIGATORIOS @Cliente, @Servicio, @Hito

              --########################
              -- 5.VALIDACION NEGOCIO #
              --########################
              EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo,
'5.VALIDACION NEGOCIO'
              EXEC DQ.VALIDACION_NEGOCIO @Cliente, @Servicio, @Hito

              --###############################
              -- 6.VALIDACION NUMERO ACTIVOS #
              --###############################
              EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo,
'6.VALIDACION NUMERO ACTIVOS'
              EXEC DQ.VALIDACION_NUMERO_ACTIVOS @Cliente, @Servicio, @Hito

              --###############################
              -- 7.VALIDACION CAMPO POSTVENTA #
              --###############################
              EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo,
'7.VALIDACION CAMPO POSTVENTA'
              IF @Cliente=1 and @Hito=1
```

```
            BEGIN
                  EXEC DQ.VALIDACION_CAMPO_POSTVENTA @Cliente, @Servicio, @Hito
            END

        EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logtitulo, '98.FIN EJECUCION'
    END TRY
    BEGIN CATCH
            SET @logMessage=error_message()
            EXEC DQ.LOG 'ERROR', @Cliente, @Servicio, @Hito,
@logtitulo,'99.ERROR',@logMessage
            RAISERROR (@logMessage, 16, 1);
    END CATCH;
END
```

DQ.VALALIDACION_CAMPO_POSTVENTA

```
USE [AAM_SERVICIO_CCPP_TRIBUTOS]
GO
/****** Object:  StoredProcedure [DQ].[VALIDACION_CAMPO_POSTVENTA]    Script Date:
18/06/2024 16:51:36 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO


/*************************************************/
-- Autor: Ángel Guzmán
-- Fecha: 20240213
-- Descripción: Proceso de validación de estados
--            Inputs:
--                  Cliente
--                  Servicio
--                  Hito
--            Resultado de validaciones:
--                  DQ.VAL_CAMPOS_DUPLICADOS
--
-- Versiones:
--            V1: Versión inicial
/*************************************************/
ALTER PROCEDURE [DQ].[VALIDACION_CAMPO_POSTVENTA] (
       @Cliente int,
       @Servicio int,
       @Hito int
)
AS
BEGIN
       BEGIN TRY
        --#############################
            -- 1.DECLARACION DE VARIABLES #
            --#############################
            DECLARE @schema NVARCHAR(10),
                        @message NVARCHAR(MAX),
                        @Tabla_Destino NVARCHAR(100),
                        @column_id nvarchar(255),
                        @column_rep nvarchar(255),
```

```
                        @error int,
                        @sqlCheck NVARCHAR(MAX),
                        @sqlTipoVenta NVARCHAR(MAX),
                        @count int,
                        @logMessage NVARCHAR(255)='VALIDACION CAMPO POSTVENTA'
            EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logMessage, '1.INICIO
VALIDACIÓN'

            --##############################
            -- 2.TRUNCADO TABLA VALIDACION #
            --##############################
            EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logMessage, '2.TRUNCADO
TABLA DQ.VAL_CAMPO_POSTVENTA'
            DELETE FROM DQ.VAL_CAMPO_POSTVENTA WHERE ID_CLIENTE=@CLIENTE AND
ID_SERVICIO=@SERVICIO AND ID_HITO=@HITO

            --##############################
            -- 3.OBTENCIÓN TABLA A VALIDAR #
            --##############################
            EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logMessage,
'3.OBTENCIÓN TABLA A VALIDAR'
            IF CURSOR_STATUS('global','tablas')>=-1
            BEGIN
                  DEALLOCATE tablas
            END

            DECLARE tablas CURSOR FOR
            SELECT DISTINCT
                  TABLA,
                  ESQUEMA
            FROM COD.PARAM_VALIDACION_CAMPOS
            WHERE ID_CLIENTE = @Cliente AND ID_SERVICIO = @Servicio AND ID_HITO =
@Hito

            OPEN tablas
            FETCH NEXT FROM tablas INTO @Tabla_Destino, @schema

            --##############################
            -- 4.VALIDACION CAMPO POSTVENTA #
            --##############################
            EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logMessage, '4.INICIO
VALIDACION CAMPO POSTVENTA'
            WHILE @@FETCH_STATUS = 0
            BEGIN
                  --4.1.Comprobacion de parametros vacios
                  EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logMessage,
'4.1.COMPROBACION DE PARAMETROS VACIOS'
                  IF @Tabla_Destino IS NULL OR @schema IS NULL
                  BEGIN
                              SET @message = '98.REVISE LOS PARAMETROS
INTRODUCIDOS EN EL ORQUESTADOR'
                              SET @error = ERROR_NUMBER()
                              EXEC DQ.LOG 'ERROR', @Cliente, @Servicio, @Hito,
@logMessage, '99.ERROR',@message
                              DEALLOCATE tablas;
                              RAISERROR (@message, 16, 1);
```

```
                    END

                    --4.2.Borrado tablas temporales
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logMessage,
'4.2.BORRADO TABLAS TEMPORALES'
                    DROP TABLE IF EXISTS #TMP_CAMPO_POSTVENTA

                    --4.3.Creacion de tabla temporal de campo
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logMessage,
'4.3.CREACION TABLA TEMPORAL CAMPOS'
                    CREATE TABLE #TMP_CAMPO_POSTVENTA (
                           ID_CLIENTE INT,
                           ID_SERVICIO INT,
                           ID_HITO INT,
                           CAMPO_ID NVARCHAR(100),
                           VALOR_ID NVARCHAR(100)
                    )

                    --4.4.Ejecucion de la validacion
                    EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logMessage,
'4.4.EJECUCIÓN DE LA VALIDACIÓN'
                    SELECT @COLUMN_ID= CAMPO, @COLUMN_REP=CAMPO_MOSTRAR_ERROR FROM
COD.PARAM_IDENTIFICADORES_TABLAS WHERE TABLA=@Tabla_Destino
                    SET @sqlTipoVenta = 'INSERT INTO #TMP_CAMPO_POSTVENTA
                                                SELECT '+CAST(@CLIENTE AS
VARCHAR)+' AS ID_CLIENTE,
                                                      '+CAST(@SERVICIO AS
VARCHAR)+' AS ID_SERVICIO,
                                                      '+CAST(@HITO AS
VARCHAR)+' AS ID_HITO,
                                                      ''POSTVENTA'' AS
CAMPO_ID,
                                                      '+@COLUMN_ID+' AS
VALOR_ID
                                                FROM HOM.'+@Tabla_Destino+'
                                                WHERE (POSTVENTA IS NOT NULL
AND POSTVENTA NOT IN (''SI'',''SI, NO PRESENTE EN EL BI''))'
                    EXEC sp_executesql @sqlTipoVenta

                    --4.5.Insercion de validaciones
                    EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logMessage,
'4.5.INSERCION DE VALIDACION'
                    INSERT INTO DQ.VAL_CAMPO_POSTVENTA
                    SELECT
ID_CLIENTE,ID_SERVICIO,ID_HITO,@Tabla_Destino,CAMPO_ID,VALOR_ID
                    FROM #TMP_CAMPO_POSTVENTA

                    FETCH NEXT FROM tablas INTO @Tabla_Destino, @schema;
             END
             CLOSE tablas;
             DEALLOCATE tablas;

             --5.Comprobacion Estados
             EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logMessage,
'5.COMPROBACION CAMPO POSTVENTA INCORRECTOS'
             SET @sqlCheck =
```

```
                              'SELECT @count=COUNT(*)
                               FROM DQ.VAL_CAMPO_POSTVENTA
                               WHERE ID_CLIENTE=' +CAST(@cliente AS VARCHAR)+' AND
ID_SERVICIO='+CAST(@servicio AS VARCHAR)+' AND ID_HITO='+ CAST(@hito AS VARCHAR)+'
                               AND TABLA = ''' +@Tabla_Destino+''''
              EXEC sp_executesql @sqlCheck, N'@count INT OUTPUT', @count OUTPUT;
              IF @count =0
                     BEGIN
                            EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
@logMessage, '98.FIN VALIDACION CAMPO POSTVENTA';
                     END
              ELSE
                     BEGIN
                            SET @message = '99.CAMPO POSTVENTA ERRÓNEO' +
@Tabla_Destino;
                            EXEC DQ.LOG 'ERROR', @Cliente, @Servicio, @Hito,
@logMessage,'99.ERROR', @message;
                     END
       END TRY
       BEGIN CATCH
              IF CURSOR_STATUS('global','tablas')>=-1
              BEGIN
                     DEALLOCATE tablas
              END
              SET @message = '99.ERROR EN LA VALIDACION DE CAMPO POSTVENTA EN LA TABLA
'+@Tabla_Destino
              SET @error = ERROR_MESSAGE();
              EXEC DQ.LOG 'ERROR', @Cliente,@Servicio,@Hito, @logMessage,
'99.ERROR',@error
       END CATCH
END
```

DQ.VALIDACION_CAMPOS_OBLIGATORIOS

```
USE [AAM_SERVICIO_CCPP_TRIBUTOS]
GO
/****** Object:  StoredProcedure [DQ].[VALIDACION_CAMPOS_OBLIGATORIOS]    Script Date:
18/06/2024 16:51:10 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
/*********************************************/
-- Autor: Ángel Guzmán
-- Fecha: 20240213
-- Descripción: Proceso de validación campos obligatorios
--              Inputs:
--                     Cliente
--                     Servicio
--                     Hito
--              Resultado de validaciones:
--                     DQ.VAL_CAMPOS_OBLIGATORIOS
--
-- Versiones:
--              V1: Versión inicial
/*********************************************/
```

```
ALTER PROCEDURE [DQ].[VALIDACION_CAMPOS_OBLIGATORIOS] (
	@Cliente int,
	@Servicio int,
	@Hito int
)
AS
BEGIN
	BEGIN TRY
		--##############################
		-- 1.DECLARACION DE VARIABLES #
		--##############################
		EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, 'VALIDACION CAMPOS
OBLIGATORIOS', '1.INICIO VALIDACIÓN'
		DECLARE @totalcolumnas INT = 0,
				@contador INT = 1,
				@schema NVARCHAR(10),
				@Tabla_Destino NVARCHAR(100),
				@message NVARCHAR(500),
				@fecha DATE = GETDATE(),
				@error NVARCHAR(500),
				@columnas_origen NVARCHAR(100),
				@columna_id NVARCHAR(100),
				@sql NVARCHAR(MAX),
				@sqlcheck NVARCHAR(MAX),
				@count int,
				@fecha2 date
		SET @fecha2= CAST(GETDATE() AS DATE)
		--##############################
		-- 2.OBTENCIÓN TABLA A VALIDAR #
		--##############################
		EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION CAMPOS
OBLIGATORIOS', '2.DECLARACION DE CURSOR'
		IF CURSOR_STATUS('global','tablas')>=-1
		BEGIN
			DEALLOCATE tablas
		END

		DECLARE tablas CURSOR FOR
		SELECT DISTINCT
			TABLA,
			ESQUEMA
		FROM COD.PARAM_VALIDACION_CAMPOS
		WHERE ID_CLIENTE = @Cliente AND ID_SERVICIO = @Servicio AND ID_HITO =
@Hito

		OPEN tablas
		FETCH NEXT FROM tablas INTO @Tabla_Destino, @schema;

		--###################################
		-- 3.VALIDACION CAMPOS OBLIGATORIOS #
		--###################################
		EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION CAMPOS
OBLIGATORIOS', '3.INICIO VALIDACION CAMPOS OBLIGATORIOS'
		WHILE @@FETCH_STATUS = 0
		BEGIN
			--3.1.Comprobacion de parametros vacios
```

```
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION CAMPOS
OBLIGATORIOS', '3.1.COMPROBACION DE PARAMETROS VACIOS'
                    IF @Tabla_Destino IS NULL OR @schema IS NULL
                    BEGIN
                           SET @message = '98.REVISE LOS PARAMETROS INTRODUCIDOS EN
EL ORQUESTADOR'
                           EXEC DQ.LOG 'ERROR', @Cliente, @Servicio, @Hito,
'VALIDACION CAMPOS OBLIGATORIOS', '99.ERROR','99.REVISE PARAMETROS INTRODUCIDOS EN EL
ORQUESTADOS';
                           EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito,
'VALIDACION CAMPOS OBLIGATORIOS', '98.FIN';
                           CLOSE tablas;
                           DEALLOCATE tablas;
                           RAISERROR (@message, 16, 1);
                    END;

                    --3.2.Borrado tablas temporales necesarias
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION CAMPOS
OBLIGATORIOS', '3.2.BORRADO TABLAS TEMPORALES'
                    DROP TABLE IF EXISTS #TMP_RELACION_CAMPOS
                    DROP TABLE IF EXISTS ##TMP_ORIGEN
                    DROP TABLE IF EXISTS ##TMP_CAMPO
                    SET @sql='DELETE FROM DQ.VAL_CAMPOS_OBLIGATORIOS WHERE TABLA
='''+@Tabla_Destino+''''
                    EXEC sp_executesql @sql


                    --3.3.Creacion de tabla temporal de campo
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION CAMPOS
OBLIGATORIOS', '3.3.CREACION TABLA TEMPORAL CAMPOS'
                    CREATE TABLE ##TMP_CAMPO (
                           TABLA NVARCHAR(100),
                           CAMPO NVARCHAR(100),
                           VALOR NVARCHAR(100),
                           RESULTADO_VALIDACION BIT,
                           CAMPO_ID NVARCHAR(100),
                           VALOR_ID NVARCHAR(100),
                           FCH_DATOS DATETIME
                    )

                    --3.4. Insertar en tabla temporal #TMP_FORMATO_CAMPOS los datos
de todos los campos para iterar en bucle por cada tabla a validar
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION CAMPOS
OBLIGATORIOS', '3.4.INSERCION EN TABLA TEMPORAL PARA ITERAR'
                    SELECT
                           TABLA
                           ,CAMPO
                           ,ROW_NUMBER() OVER(ORDER BY TABLA, CAMPO) AS RN
                    INTO #TMP_CAMPOS_OBLIGATORIOS
                    FROM COD.PARAM_VALIDACION_CAMPOS
                    WHERE ESQUEMA = @schema
                    AND TABLA = @Tabla_Destino
                    AND OBLIGATORIO = 1
                    SELECT @totalcolumnas = COUNT(*) FROM #TMP_CAMPOS_OBLIGATORIOS
```

```
                    SELECT @columna_id = CAMPO_MOSTRAR_ERROR FROM
COD.PARAM_IDENTIFICADORES_TABLAS WHERE TABLA = @Tabla_Destino

                    -- 3.5 Si la query no devuelve resultados es que los parámetros
no cuadran con el contenido de la tabla
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION CAMPOS
OBLIGATORIOS', '3.5.COMPROBACION QUERY DEVUELVE RESULTADOS'
                    IF @totalcolumnas = 0
                    BEGIN
                           SET @message = '99.REVISE LOS VALORES DE LOS PARAMETROS
INTRODUCIDOS'

                           EXEC DQ.LOG 'ERROR', @Cliente, @Servicio, @Hito,
'VALIDACION ESTADOS','99.ERROR',@message
                    END

                    --3.6. Procedimiento para ejecutar las validaciones
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION CAMPOS
OBLIGATORIOS', '3.6.INICIO VALIDACION CAMPOS INDIVIDUAL'
                    WHILE @contador <= @totalcolumnas
                    BEGIN
                           BEGIN TRY
                                  DECLARE @columnname NVARCHAR(100) = NULL, @query
NVARCHAR(MAX) = NULL

                                  --3.6.1 Inicializar variables con datos de campo
                                  SELECT @columnname = CAMPO
                                  FROM #TMP_CAMPOS_OBLIGATORIOS
                                  WHERE RN = @contador

                                  SET @message = left('3.6.1.INICIO VALIDACION TABLA
'+@schema+'.' + @Tabla_Destino + ' CAMPO ' + @columnname,100)
                                  EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito,
'VALIDACION CAMPOS OBLIGATORIOS', @message

                                  IF @columnname = @columna_id SET @columnas_origen
= @columna_id ELSE SET @columnas_origen = @columnname + ',' + @columna_id

                                  --3.6.2 Insertar datos del campo completo en tabla
tempora #TMP_CAMPO ejecutando las validaciones
                                  EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
'VALIDACION CAMPOS OBLIGATORIOS', '3.6.2.INSERCION DATOS DEL CAMPO EN TABLA TEMPORAL'
                                  SET @query = '
                                         INSERT INTO ##TMP_CAMPO
                                         SELECT *
                                         FROM (
                                                SELECT ''' + @Tabla_Destino + ''' AS
TABLA,
                                                       ''' + @columnname + ''' AS
CAMPO,
                                                       LEFT(' + @columnname + ',
255) AS VALOR,
                                                       CASE
                                                              WHEN ' + @columnname +
' IS NULL THEN 0
```

```
                                                                WHEN LTRIM(RTRIM(' +
@columnname + ')) = '''' THEN 0
                                                                ELSE 1
                                                        END AS RESULTADO_VALIDACION,
                                                        ''''+@columna_id+''' AS
CAMPO_ID,
                                                        '+@columna_id+' AS VALOR_ID,
                                                        ''' + CAST(@fecha AS
NVARCHAR) + ''' AS FCH_DATOS
                                                FROM (
                                                        SELECT ' + @columnas_origen +
'
                                                        FROM '+@schema +'.'
+@Tabla_Destino + ' WITH (NOLOCK)
                                                        GROUP BY ' + @columnas_origen
+ '
                                                ) AS B
                                        ) AS A
                                        WHERE A.RESULTADO_VALIDACION = 0;'
                                EXEC sp_executesql @query
                                SET @message = left('3.6.2.FIN VALIDACION TABLA
'+@schema+'.' + @Tabla_Destino + ' CAMPO ' + @columnname,100)
                                EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito,
'VALIDACION CAMPOS OBLIGATORIOS', @message

                                SET @contador = @contador + 1
                        END TRY
                        BEGIN CATCH
                                -- 5.6 Insertar en tabla de log las excepciones
                                SET @error=error_message()
                                EXEC DQ.LOG 'ERROR', @Cliente, @Servicio, @Hito,
'VALIDACION CAMPOS OBLIGATORIOS','99.ERROR',@error
                                SET @contador = @contador + 1
                        END CATCH
                END

                --3.7. Inserción de resultados en tabla DQ.VAL_FORMATO_CAMPOS
                EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION CAMPOS
OBLIGATORIOS', '3.7.INSERCION EN TABLA VAL_CAMPOS_OBLIGATORIOS'
                INSERT INTO DQ.VAL_CAMPOS_OBLIGATORIOS
(ID_CLIENTE,ID_SERVICIO,ID_HITO,TABLA, CAMPO, VALOR, RESULTADO_VALIDACION, CAMPO_ID,
VALOR_ID, FCH_DATOS)
                SELECT @CLIENTE,@SERVICIO,@HITO, * FROM ##TMP_CAMPO


                FETCH NEXT FROM tablas INTO @Tabla_Destino, @schema;
        END
        CLOSE tablas
        DEALLOCATE tablas
        --4.Comprobacion campos obligatorios
        EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION CAMPOS
OBLIGATORIOS', '4.COMPROBACION EXISTENCIA CAMPOS OBLIGATORIOS'
        SET @sqlcheck='SELECT @count=COUNT(*)
                        FROM DQ.VAL_CAMPOS_HOM H
                        LEFT JOIN DQ.VAL_VALIDACION_BLOQUEANTE B
                        ON H.TABLA_DEST = B.TABLA AND H.CAMPO=B.CAMPO
```

```
                                        WHERE ID_CLIENTE=' +CAST(@cliente AS VARCHAR)+'
AND ID_SERVICIO='+CAST(@servicio AS VARCHAR)+' AND ID_HITO='+ CAST(@hito AS VARCHAR)+'
                                        AND B.TABLA = ''HOM.' +@Tabla_Destino+''''
              EXEC sp_executesql @sqlcheck, N'@count INT OUTPUT', @count OUTPUT;
              IF @count =0
                     BEGIN
                            EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION
CAMPOS OBLIGATORIOS', '98.FIN VALIDACION CAMPOS OBLIGATORIOS';
                     END
              ELSE
                     BEGIN
                            SET @error=error_message()
                            EXEC DQ.LOG 'ERROR', @Cliente, @Servicio, @Hito,
'VALIDACION CAMPOS OBLIGATORIOS','99.ERROR', 'VALIDACION BLOQUEANTE';
                     END
              EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, 'VALIDACION CAMPOS
OBLIGATORIOS', '98.FIN VALIDACIÓN'
       END TRY
       BEGIN CATCH
              IF CURSOR_STATUS('global','tablas')>=-1
              BEGIN
                     DEALLOCATE tablas
              END
              SET @error=error_message()
              EXEC DQ.LOG 'ERROR', @Cliente, @Servicio, @Hito, 'VALIDACION CAMPOS
OBLIGATORIOS','99.ERROR',@error
       END CATCH
END
```

DQ.VALIDACION_DUPLICADOS

```
USE [AAM_SERVICIO_CCPP_TRIBUTOS]
GO
/****** Object:  StoredProcedure [DQ].[VALIDACION_DUPLICADOS]    Script Date:
18/06/2024 16:50:45 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

/*********************************************/
-- Autor: Ángel Guzmán
-- Fecha: 20240213
-- Descripción: Proceso de validación de estados
--              Inputs:
--                     Cliente
--                     Servicio
--                     Hito
--              Resultado de validaciones:
--                     DQ.VAL_CAMPOS_DUPLICADOS
--
-- Versiones:
--              V1: Versión inicial
/*********************************************/
ALTER PROCEDURE [DQ].[VALIDACION_DUPLICADOS] (
       @Cliente int,
```

```
	@Servicio int,
	@Hito int
)
AS
BEGIN
	BEGIN TRY
	 --##############################
		-- 1.DECLARACION DE VARIABLES #
		--##############################
		EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, 'VALIDACION DUPLICADOS',
'1.INICIO VALIDACIÓN'
		DECLARE @schema NVARCHAR(10),
				@message NVARCHAR(MAX),
				@Tabla_Destino NVARCHAR(100),
				@column_id nvarchar(255),
				@column_rep nvarchar(255),
				@error int,
				@sqlCheck NVARCHAR(MAX),
				@sqlDuplicados NVARCHAR(MAX),
				@count int

		--##############################
		-- 2.TRUNCADO TABLA VALIDACION #
		--##############################
		EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, 'VALIDACION DUPLICADOS',
'2.TRUNCADO TABLA DQ.VAL_DUPLICADOS'
		DELETE FROM DQ.VAL_DUPLICADOS WHERE ID_CLIENTE=@CLIENTE AND
ID_SERVICIO=@SERVICIO AND ID_HITO=@HITO

		--##############################
		-- 3.OBTENCIÓN TABLA A VALIDAR #
		--##############################
		EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, 'VALIDACION DUPLICADOS',
'3.OBTENCIÓN TABLA A VALIDAR'
		IF CURSOR_STATUS('global','tablas')>=-1
		BEGIN
			DEALLOCATE tablas
		END
		DECLARE tablas CURSOR FOR
		SELECT DISTINCT
			TABLA,
			ESQUEMA
		FROM COD.PARAM_VALIDACION_CAMPOS
		WHERE ID_CLIENTE = @Cliente AND ID_SERVICIO = @Servicio AND ID_HITO =
@Hito

		OPEN tablas;
		FETCH NEXT FROM tablas INTO @Tabla_Destino, @schema;

		--############################
		-- 4.VALIDACION DUPLICADOS #
		--############################
		EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION DUPLICADOS',
'4.INICIO VALIDACION DUPLICADOS'

		WHILE @@FETCH_STATUS = 0
```

```
            BEGIN
                  --4.1.Comprobacion de parametros vacios
                  EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION
DUPLICADOS', '4.1.COMPROBACION DE PARAMETROS VACIOS'
                  IF @Tabla_Destino IS NULL OR @schema IS NULL
                  BEGIN
                                    SET @message = '98.REVISE LOS PARAMETROS
INTRODUCIDOS EN EL ORQUESTADOR'
                                    SET @error = ERROR_NUMBER()
                                    EXEC DQ.LOG 'ERROR', @Cliente, @Servicio, @Hito,
'VALIDACION DUPLICADOS', '99.ERROR',@message
                                    CLOSE tablas;
                                    DEALLOCATE tablas;
                                    RAISERROR (@message, 16, 1);
                  END

                  --4.2.Borrado tablas temporales
                  EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION
DUPLICADOS', '4.2.BORRADO TABLAS TEMPORALES'
                  DROP TABLE IF EXISTS #TMP_DUPLICADOS

                  --4.3.Creacion de tabla temporal de campo
                  EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION
DUPLICADOS', '4.3.CREACION TABLA TEMPORAL CAMPOS'
                  CREATE TABLE #TMP_DUPLICADOS (
                        ID_CLIENTE INT,
                        ID_SERVICIO INT,
                        ID_HITO INT,
                        TABLA NVARCHAR(255),
                        CAMPO_ID NVARCHAR(100),
                        VALOR_ID NVARCHAR(100)
                  )
                  --4.4.Ejecucion de la validacion
                  EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, 'VALIDACION
DUPLICADOS', '4.4.EJECUCIÓN DE LA VALIDACIÓN'
                  SELECT @COLUMN_ID= CAMPO, @COLUMN_REP=CAMPO_MOSTRAR_ERROR FROM
COD.PARAM_IDENTIFICADORES_TABLAS WHERE TABLA=@Tabla_Destino
                  SET @sqlDuplicados = 'INSERT INTO #TMP_DUPLICADOS
                                                SELECT '+CAST(@CLIENTE AS
VARCHAR)+' AS ID_CLIENTE,

'+CAST(@SERVICIO AS VARCHAR)+' AS ID_SERVICIO,
                                                      '+CAST(@HITO
AS VARCHAR)+' AS ID_HITO,

'''+@Tabla_Destino+''' AS TABLA,

'''+@column_rep+''' AS CAMPO_ID,
                                                      '+@COLUMN_ID+'
AS VALOR_ID
                                                FROM HOM.'+@Tabla_Destino+'
                                                GROUP BY '+@column_id+'
                                                HAVING COUNT(*) > 1'
                  EXEC sp_executesql @sqlDuplicados

                  --4.5.Insercion de validaciones
```

```
                        EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, 'VALIDACION
DUPLICADOS', '4.5.INSERCION DE VALIDACION'
                        INSERT INTO DQ.VAL_DUPLICADOS
                        SELECT
ID_CLIENTE,ID_SERVICIO,ID_HITO,@Tabla_Destino,CAMPO_ID,VALOR_ID
                        FROM #TMP_DUPLICADOS


                        FETCH NEXT FROM tablas INTO @Tabla_Destino, @schema;
                END
                CLOSE tablas;
                DEALLOCATE tablas;

                --5.Comprobacion valores duplicados
                EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION DUPLICADOS',
'5.COMPROBACION EXISTENCIA DUPLICADOS'
                SET     @sqlcheck='SELECT @count=COUNT(*)
                                        FROM DQ.VAL_DUPLICADOS
                                        WHERE ID_CLIENTE=' +CAST(@cliente AS VARCHAR)+'
AND ID_SERVICIO='+CAST(@servicio AS VARCHAR)+' AND ID_HITO='+ CAST(@hito AS VARCHAR)+'
                                        AND TABLA = ''' +@Tabla_Destino+''''
                EXEC sp_executesql @sqlcheck, N'@count INT OUTPUT', @count OUTPUT;
                IF @count =0
                        BEGIN
                                EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION
DUPLICADOS', '98.FIN VALIDACION DUPLICADOS';
                        END
                ELSE
                        BEGIN
                                SET @message = '99.EXISTEN DUPLICADOS EN LA TABLA ' +
@Tabla_Destino;
                                EXEC DQ.LOG 'ERROR', @Cliente, @Servicio, @Hito,
'VALIDACION DUPLICADOS','99.ERROR', @message;
                        END
        END TRY
        BEGIN CATCH
                IF CURSOR_STATUS('global','tablas')>=-1
                BEGIN
                        DEALLOCATE tablas
                END
                SET @ERROR=ERROR_MESSAGE()
                SET @message = '99.ERROR EN LA COMPROBACIÓN DE DUPLICADOS EN LA TABLA
'+@Tabla_Destino
                EXEC DQ.LOG 'ERROR', @Cliente,@Servicio,@Hito, 'VALIDACION DUPLICADOS',
'99.ERROR',@ERROR
        END CATCH
END
```

DQ.VALIDACION_ESTADOS

```
USE [AAM_SERVICIO_CCPP_TRIBUTOS]
GO
/****** Object:  StoredProcedure [DQ].[VALIDACION_ESTADOS]    Script Date: 18/06/2024
16:50:21 ******/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO

/***********************************************/
-- Autor: Ángel Guzmán
-- Fecha: 20240213
-- Descripción: Proceso de validación de estados
--            Inputs:
--                   Cliente
--                   Servicio
--                   Hito
--            Resultado de validaciones:
--                   DQ.VAL_ESTADOS
--
-- Versiones:
--            V1: Versión inicial
/***********************************************/
ALTER PROCEDURE [DQ].[VALIDACION_ESTADOS] (
       @Cliente int,
       @Servicio int,
       @Hito int
)
AS
BEGIN
       BEGIN TRY

              --##############################
              -- 1.DECLARACION DE VARIABLES #
              --##############################
              EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, 'VALIDACION ESTADOS',
'1.INICIO VALIDACIÓN'
              DECLARE @schema NVARCHAR(10),
                           @message NVARCHAR(MAX),
                           @Tabla_Destino NVARCHAR(100),
                           @column_id nvarchar(255),
                           @column_rep nvarchar(255),
                           @error NVARCHAR(500),
                           @sqlCheck NVARCHAR(MAX),
                           @sqlEstados NVARCHAR(MAX),
                           @count int,
                           @serv nvarchar(4),
                           @Estado NVARCHAR(MAX);


              --##############################
              -- 2.TRUNCADO TABLA VALIDACION #
              --##############################
              EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, 'VALIDACION ESTADOS',
'2.TRUNCAR TABLA DQ.VAL_ESTADOS'
              DELETE FROM DQ.VAL_ESTADOS WHERE ID_CLIENTE=@CLIENTE AND
ID_SERVICIO=@SERVICIO AND ID_HITO=@HITO
              --##############################
              -- 3.OBTENCIÓN TABLA A VALIDAR #
              --##############################
              EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, 'VALIDACION ESTADOS',
'3.OBTENCIÓN TABLA A VALIDAR'
```

```
		IF CURSOR_STATUS('global','tablas')>=-1
		BEGIN
			DEALLOCATE tablas
		END
		DECLARE tablas CURSOR FOR
		SELECT DISTINCT
			TABLA,
			ESQUEMA
		FROM COD.PARAM_VALIDACION_CAMPOS
		WHERE ID_CLIENTE = @Cliente AND ID_SERVICIO = @Servicio AND ID_HITO =
@Hito

		OPEN tablas;
		FETCH NEXT FROM tablas INTO @Tabla_Destino, @schema;

		--###################################
		-- 4.ASIGNACION ESTADO SEGUN TABLA #
		--###################################
		--4. Asignación variable @Estado según la tabla a validar
		EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, 'VALIDACION ESTADOS',
'4.OBTENCIÓN DE LA COLUMNA A VALIDAR SEGÚN LA TABLA'
		IF @Servicio=1 BEGIN SET @Estado ='ESTADO_TRIBUTOS' END
		IF @Servicio=2 BEGIN SET @Estado='ESTADO_COMUNIDADES' END
		IF @Servicio=1 BEGIN SET @serv ='TRIB' END
		IF @Servicio=2 BEGIN SET @serv='CCPP' END

		IF @Tabla_Destino IS NULL OR @schema IS NULL
			BEGIN
				SET @message = '98.REVISE LOS PARAMETROS INTRODUCIDOS EN
EL ORQUESTADOR'
				SET @error = ERROR_NUMBER()
				EXEC DQ.LOG 'ERROR', @Cliente, @Servicio, @Hito,
'VALIDACION ESTADOS', '99.ERROR',@message
			END;
		--########################
		-- 5.VALIDACION ESTADOS #
		--########################
		EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION ESTADOS',
'5.INICIO VALIDACION DUPLICADOS'
		WHILE @@FETCH_STATUS = 0
		BEGIN
			--5.1.Comprobacion de parametros vacios
			EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION
ESTADOS', '5.1.COMPROBACION DE PARAMETROS VACIOS'
			IF @Tabla_Destino IS NULL OR @schema IS NULL
			BEGIN
					SET @message = '98.REVISE LOS PARAMETROS
INTRODUCIDOS EN EL ORQUESTADOR'
					SET @error = ERROR_NUMBER()
					EXEC DQ.LOG 'ERROR', @Cliente, @Servicio, @Hito,
'VALIDACION ESTADOS', '99.ERROR',@message
					CLOSE tablas;
					DEALLOCATE tablas;
					RAISERROR (@message, 16, 1);
			END
```

```
                    --5.2.Borrado tablas temporales
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION
ESTADOS', '5.2.BORRADO TABLAS TEMPORALES'
                    DROP TABLE IF EXISTS #TMP_ESTADOS

                    --5.3.Creacion de tabla temporal de campo
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION
ESTADOS', '5.3.CREACION TABLA TEMPORAL CAMPOS'
                    CREATE TABLE #TMP_ESTADOS (
                           ID_CLIENTE INT,
                           ID_SERVICIO INT,
                           ID_HITO INT,
                           TABLA  NVARCHAR(255),
                           CAMPO_ID NVARCHAR(100),
                           VALOR_ID NVARCHAR(100)
                    )

                    --5.4.Ejecucion de la validacion
                    EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, 'VALIDACION
ESTADOS', '5.4.EJECUCIÓN DE LA VALIDACIÓN'
                    SELECT @COLUMN_ID= CAMPO, @COLUMN_REP=CAMPO_MOSTRAR_ERROR FROM
COD.PARAM_IDENTIFICADORES_TABLAS WHERE TABLA=@Tabla_Destino
                    SET @sqlEstados = 'INSERT INTO #TMP_ESTADOS
                                                      SELECT '+CAST(@CLIENTE AS
VARCHAR)+' AS ID_CLIENTE,
                                                             '+CAST(@SERVICIO AS
VARCHAR)+' AS ID_SERVICIO,
                                                             '+CAST(@HITO AS
VARCHAR)+' AS ID_HITO,
                                                             '''+@Tabla_Destino
+''' AS TABLA,
                                                             '''+@Estado+''' AS
CAMPO_ID,
                                                             '+@COLUMN_ID+' AS
VALOR_ID
                                                      FROM HOM.'+@Tabla_Destino+'
                                                      WHERE '+@Estado+' NOT IN
(SELECT estado FROM COD.ESTADO)'
                    EXEC sp_executesql @sqlEstados

                    --5.5.Insercion de validaciones
                    EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, 'VALIDACION
ESTADOS', '5.5.INSERCION DE VALIDACION'
                    INSERT INTO DQ.VAL_ESTADOS
                    SELECT
ID_CLIENTE,ID_SERVICIO,ID_HITO,@Tabla_Destino,CAMPO_ID,VALOR_ID
                    FROM #TMP_ESTADOS

                    FETCH NEXT FROM tablas INTO @Tabla_Destino, @schema;
             END
             CLOSE tablas;
             DEALLOCATE tablas;


             --6.Comprobacion Estados
```

```
		EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION ESTADOS',
'6.COMPROBACION ESTADOS INCORRECTOS'
		SET	@sqlcheck='SELECT @count=COUNT(*)
					FROM DQ.VAL_ESTADOS
					WHERE ID_CLIENTE=' +CAST(@cliente AS VARCHAR)+'
AND ID_SERVICIO='+CAST(@servicio AS VARCHAR)+' AND ID_HITO='+ CAST(@hito AS VARCHAR)+'
					AND TABLA = ''' +@Tabla_Destino+''''
		EXEC sp_executesql @sqlcheck, N'@count INT OUTPUT', @count OUTPUT;
		IF @COUNT = 0
			BEGIN
				EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION
ESTADOS', '6.1.FIN COMPROBACIÓN DE ESTADOS';
			END
		ELSE
			BEGIN
				SET @message = '99.ERROR EN LA COMPROBACIÓN DE ESTADOS '
+ @Tabla_Destino
				EXEC DQ.LOG 'ERROR', @Cliente, @Servicio, @Hito,
'VALIDACION ESTADOS', '99.ERROR','99.EXISTEN VALORES DE ESTADOS NO VÁLIDOS. FIN
COMPROBACIÓN DE ESTADOS';
			END
	END TRY
	BEGIN CATCH
		--7. Insertar en tabla de log las excepciones
		SET @message = '99.ERROR EN LA COMPROBACIÓN DE ESTADOS ' +
@Tabla_Destino
		SET @error = ERROR_NUMBER()
		EXEC DQ.LOG 'ERROR', @Cliente, @Servicio, @Hito, 'VALIDACION ESTADOS',
'99.ERROR','99.EXISTEN VALORES NO VÁLIDOS. FIN COMPROBACIÓN DE ESTADOS';
	END CATCH
END
```

DQ.VALIDACION_NEGOCIO

```
USE [AAM_SERVICIO_CCPP_TRIBUTOS]
GO
/****** Object:  StoredProcedure [DQ].[VALIDACION_NEGOCIO]    Script Date: 18/06/2024
16:47:33 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
/*************************************************/
-- Autor: Ángel Guzmán
-- Fecha: 20240213
-- Descripción: Proceso de validación campos negocio
--		Inputs:
--			Cliente
--			Servicio
--			Hito
--		Resultado de validaciones:
--			DQ.VAL_NEGOCIO
--
-- Versiones:
--		V1: Versión inicial
/*************************************************/
```

```
ALTER PROCEDURE [DQ].[VALIDACION_NEGOCIO] (
	@Cliente int,
	@Servicio int,
	@Hito int
)
AS
BEGIN
	BEGIN TRY
		--##############################
		-- 1.DECLARACION DE VARIABLES #
		--##############################
		EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, 'VALIDACION CAMPOS
NEGOCIO', '1.INICIO VALIDACIÓN'
		DECLARE @totalcolumnas INT = 0,
				@contador INT = 1,
				@schema VARCHAR(10),
				@Tabla_Destino VARCHAR(100),
				@message nvarchar(500),
				@fecha DATETIME = GETDATE(),
				@error nvarchar(500),
				@columnas_origen NVARCHAR(100),
				@columna_id NVARCHAR(100),
				@mensaje NVARCHAR(100)

		--##############################
		-- 2.OBTENCIÓN TABLA A VALIDAR #
		--##############################
		EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION CAMPOS
NEGOCIO', '2.DECLARACION DE CURSOR'
		IF CURSOR_STATUS('global','tablas')>=-1
		BEGIN
			DEALLOCATE tablas
		END

		DECLARE tablas CURSOR FOR
		SELECT DISTINCT
			TABLA,
			ESQUEMA
		FROM COD.PARAM_VALIDACION_CAMPOS
		WHERE ID_CLIENTE = @Cliente AND ID_SERVICIO = @Servicio AND ID_HITO =
@Hito

		OPEN tablas;
		FETCH NEXT FROM tablas INTO @Tabla_Destino, @schema;

		--##############################
		-- 3.VALIDACION CAMPOS NEGOCIO #
		--##############################
		EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION CAMPOS
OBLIGATORIOS', '3.INICIO VALIDACION CAMPOS NEGOCIO'
		WHILE @@FETCH_STATUS = 0
		BEGIN
			--3.1.Comprobacion de parametros vacios
			EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION CAMPOS
NEGOCIO', '3.1.COMPROBACION DE PARAMETROS VACIOS'
			IF @Tabla_Destino IS NULL OR @schema IS NULL
```

```
                    BEGIN
                           EXEC DQ.LOG 'ERROR', @Cliente, @Servicio, @Hito,
'VALIDACION CAMPOS NEGOCIO', '99.ERROR','99.REVISE PARAMETROS INTRODUCIDOS EN EL
ORQUESTADOS';
                           EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito,
'VALIDACION CAMPOS NEGOCIO', '98.FIN';
                           CLOSE tablas;
                           DEALLOCATE tablas;
                           RAISERROR (@message, 16, 1);
                    END;

                    --3.2.Borrado tablas temporales necesarias
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION CAMPOS
NEGOCIO', '3.2.BORRADO TABLAS TEMPORALES'
                    DROP TABLE IF EXISTS #TMP_RELACION_CAMPOS
                    DROP TABLE IF EXISTS ##TMP_ORIGEN
                    DELETE FROM DQ.VAL_NEGOCIO WHERE TABLA = @Tabla_Destino

                    --3.3.Creacion de tabla temporal de campo
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION CAMPOS
NEGOCIO', '3.3.CREACION TABLA TEMPORAL CAMPOS'
                    CREATE TABLE #TMP_CAMPO (
                           TABLA NVARCHAR(100),
                           CAMPO NVARCHAR(100),
                           VALOR NVARCHAR(100),
                           RESULTADO_VALIDACION BIT,
                           CAMPO_ID NVARCHAR(100),
                           VALOR_ID NVARCHAR(100),
                           MENSAJE_ERROR NVARCHAR(500),
                           FCH_DATOS DATETIME
                    )

                    --3.4. Insertar en tabla temporal #TMP_FORMATO_CAMPOS los datos
de todos los campos para iterar en bucle por cada tabla a validar
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION CAMPOS
NEGOCIO', '3.4.INSERCION EN TABLA TEMPORAL PARA ITERAR'
                    SELECT
                           CR.TABLA AS TABLE_NAME,
                           CR.ESQUEMA AS TABLE_SCHEMA,
                           CR.CAMPO AS COLUMN_NAME,
                           CR.VALIDACION AS VALIDACION,
                           CR.MSG_ERROR AS MENSAJE_ERROR,
                           ROW_NUMBER() OVER(ORDER BY CR.TABLA, CR.CAMPO,
CR.VALIDACION) AS RN
                    INTO #TMP_RELACION_CAMPOS
                    FROM [COD].[PARAM_CAMPOS_NEGOCIO] CR
                    WHERE CR.TABLA = @Tabla_Destino
                    AND CR.ESQUEMA = @schema

                    SELECT @totalcolumnas = COUNT(*) FROM #TMP_RELACION_CAMPOS

                    SELECT @columna_id = CAMPO_MOSTRAR_ERROR FROM
COD.PARAM_IDENTIFICADORES_TABLAS WHERE TABLA = @Tabla_Destino

                    -- 3.5 Si la query no devuelve resultados es que los parámetros
no cuadran con el contenido de la tabla
```

```
                        EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION CAMPOS
NEGOCIO', '3.5.COMPROBACION QUERY DEVUELVE RESULTADOS'
                        IF @totalcolumnas = 0
                        BEGIN
                                SET @message = '99.REVISE LOS VALORES DE LOS PARAMETROS
INTRODUCIDOS '+@Tabla_Destino
                                EXEC DQ.LOG 'ERROR', @Cliente, @Servicio, @Hito,
'VALIDACION ESTADOS','99.ERROR',@message
                        END

                        --3.6. Procedimiento para ejecutar las validaciones
                        EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION CAMPOS
OBLIGATORIOS', '3.6.INICIO VALIDACION CAMPOS INDIVIDUAL'
                        WHILE @contador <= @totalcolumnas
                        BEGIN
                                BEGIN TRY
                                        DECLARE @columnname VARCHAR(100) = NULL,
@validation VARCHAR(MAX) = NULL, @error_msg VARCHAR(255) = NULL, @query NVARCHAR(MAX)
= NULL

                                        --3.6.1 Inicializar variables con datos de campo
                                        SELECT
                                                @columnname = COLUMN_NAME,
                                                @validation = VALIDACION,
                                                @error_msg = MENSAJE_ERROR
                                        FROM #TMP_RELACION_CAMPOS
                                        WHERE RN = @contador

                                        SET @message = left('3.6.1.INICIO VALIDACION TABLA
'+@schema+'.' + @Tabla_Destino + ' CAMPO ' + @columnname,100)
                                        EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito,
'VALIDACION CAMPOS NEGOCIO', @message, NULL

                                        IF @columnname = @columna_id SET @columnas_origen
= @columna_id ELSE SET @columnas_origen = @columnname + ',' + @columna_id

                                        --3.6.2 Insertar datos del campo completo en tabla
tempora #TMP_CAMPO ejecutando las validaciones
                                        EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
'VALIDACION CAMPOS NEGOCIO', '3.6.2.INSERCION DATOS DEL CAMPO EN TABLA TEMPORAL'
                                        SET @query = 'INSERT INTO #TMP_CAMPO
                                          SELECT *
                                          FROM (
                                                SELECT ''' + @Tabla_Destino + ''' AS
TABLA,
                                                        ''' + @columnname + ''' AS CAMPO,
                                                        ' + ISNULL('' + @columnname + '',
'''') + ' AS VALOR_CAMPO,
                                                        CASE
                                                                WHEN ' + ISNULL(@columnname,
'NULL') + ' IS NULL THEN 1
                                                                WHEN ' + ISNULL(@columnname,
'NULL') + ' = '''' THEN 0
                                                                --WHEN CHARINDEX(CHAR(10), '
+ ISNULL(@columnname, 'NULL') + ') > 0 OR CHARINDEX(CHAR(13), ' + ISNULL(@columnname,
'NULL') + ') > 0 THEN 0
```

```
                                                        WHEN ' + ISNULL(@validation,
'NULL') + ' THEN 0
                                                        ELSE 1
                                                END AS RESULTADO_VALIDACION,
                                                '''+@columna_id+''' AS CAMPO_ID,
                                                '+@columna_id+' AS VALOR_ID,
                                                '''+ @error_msg + ''' AS
MENSAJE_ERROR,
                                                '''+ CAST(@fecha AS NVARCHAR) + '''
AS FCH_DATOS
                                        FROM (SELECT * FROM HOM.' +
ISNULL(@Tabla_Destino, 'NULL') + ' WITH (NOLOCK)) AS B
                                        --WHERE NOT EXISTS (SELECT 1 FROM
STG.DQ.VAL_FROMATO_CAMPOS T2 WHERE T2.TABLA = ''' + @Tabla_Destino + ''' AND T2.CAMPO
= ''' + @columnname + ''')
                                  ) AS A WHERE A.RESULTADO_VALIDACION = 0;'
                                EXEC sp_executesql @query


                                SET @message = left('3.6.2.FIN VALIDACION TABLA
'+@schema+ '.' + @Tabla_Destino + ' CAMPO ' + @columnname,100)
                                EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito,
'VALIDACION CAMPOS NEGOCIO', @message

                                SET @contador = @contador + 1
                        END TRY
                        BEGIN CATCH
                                --5.6 Insertar en tabla de log las excepciones
                                SET @message = '99.ERROR VALIDACIÓN ' +
@Tabla_Destino + ' - ' + @columnname
                                EXEC DQ.LOG 'ERROR', @Cliente, @Servicio, @Hito,
'VALIDACION CAMPOS NEGOCIO','99.ERROR', @message
                                SET @contador = @contador + 1
                        END CATCH
                END

                --3.7. Inserción de resultados en tabla DQ.VAL_FORMATO_CAMPOS
                EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'VALIDACION CAMPOS
NEGOCIO', '3.7.INSERCION EN TABLA VAL_NEGOCIO', NULL
                INSERT INTO DQ.VAL_NEGOCIO (ID_CLIENTE, ID_SERVICIO,
ID_HITO,TABLA, CAMPO, VALOR_CAMPO, RESULTADO_VALIDACION,CAMPO_ID, VALOR_ID,
MENSAJE_ERROR, FCH_DATOS)
                SELECT @CLIENTE,@SERVICIO,@HITO, * FROM #TMP_CAMPO

                FETCH NEXT FROM tablas INTO @Tabla_Destino, @schema;
        END
        CLOSE tablas
        DEALLOCATE tablas
        EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, 'VALIDACION CAMPOS
NEGOCIO', '98.FIN VALIDACIÓN'
    END TRY
    BEGIN CATCH
        IF CURSOR_STATUS('global','tablas')>=-1
        BEGIN
                DEALLOCATE tablas
        END
```

```
            SET @message = '99.ERROR VALIDACIÓN ' + @Tabla_Destino
            EXEC DQ.LOG 'ERROR', @Cliente, @Servicio, @Hito, 'VALIDACION CAMPOS
NEGOCIO', '99.ERROR',@message
      END CATCH
END
```

DQ.VALIDACION_NUMERO_ACTIVOS

```
USE [AAM_SERVICIO_CCPP_TRIBUTOS]
GO
/****** Object:  StoredProcedure [DQ].[VALIDACION_NUMERO_ACTIVOS]    Script Date:
18/06/2024 16:49:36 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO


/*********************************************/
-- Autor: Ángel Guzmán
-- Fecha: 20240213
-- Descripción: Proceso de validación de estados
--            Inputs:
--                  Cliente
--                  Servicio
--                  Hito
--            Resultado de validaciones:
--                  DQ.VAL_CAMPOS_DUPLICADOS
--
-- Versiones:
--            V1: Versión inicial
/*********************************************/
ALTER PROCEDURE [DQ].[VALIDACION_NUMERO_ACTIVOS] (
      @Cliente int,
      @Servicio int,
      @Hito int
)
AS
BEGIN
      BEGIN TRY
       --############################
            -- 1.DECLARACION DE VARIABLES #
            --############################
            DECLARE @schema NVARCHAR(10),
                        @message NVARCHAR(MAX),
                        @Tabla_Destino NVARCHAR(100),
                        @column_id nvarchar(255),
                        @column_rep nvarchar(255),
                        @error int,
                        @sqlCheck NVARCHAR(MAX),
                        @sqlTipoVenta NVARCHAR(MAX),
                        @count int,
                        @serv nvarchar(4),
                        @logMessage NVARCHAR(255)='VALIDACION NUMERO ACTIVOS'
            EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logMessage, '1.INICIO
VALIDACIÓN'
```

```
		IF @Servicio=1 BEGIN SET @serv ='TRIB' END
		IF @Servicio=2 BEGIN SET @serv='CCPP' END
		--##############################
		-- 2.TRUNCADO TABLA VALIDACION #
		--##############################
		EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logMessage, '2.TRUNCADO
TABLA DQ.VAL_NUMERO_ACTIVOS'
		DELETE FROM DQ.VAL_NUMERO_ACTIVOS WHERE ID_CLIENTE=@CLIENTE AND
ID_SERVICIO=@SERVICIO AND ID_HITO=@HITO

		--##############################
		-- 3.OBTENCIÓN TABLA A VALIDAR #
		--##############################
		EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logMessage,
'3.OBTENCIÓN TABLA A VALIDAR'
		IF CURSOR_STATUS('global','tablas')>=-1
		BEGIN
			DEALLOCATE tablas
		END

		DECLARE tablas CURSOR FOR
		SELECT DISTINCT
			TABLA,
			ESQUEMA
		FROM COD.PARAM_VALIDACION_CAMPOS
		WHERE ID_CLIENTE = @Cliente AND ID_SERVICIO = @Servicio AND ID_HITO =
@Hito

		OPEN tablas
		FETCH NEXT FROM tablas INTO @Tabla_Destino, @schema

		--##############################
		-- 4.VALIDACION NUMERO ACTIVOS #
		--##############################
		EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logMessage, '4.INICIO
VALIDACION NUMERO ACTIVOS'
		WHILE @@FETCH_STATUS = 0
		BEGIN
			--4.1.Comprobacion de parametros vacios
			EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logMessage,
'4.1.COMPROBACION DE PARAMETROS VACIOS'
			IF @Tabla_Destino IS NULL OR @schema IS NULL
			BEGIN
					SET @message = '98.REVISE LOS PARAMETROS
INTRODUCIDOS EN EL ORQUESTADOR'
					SET @error = ERROR_NUMBER()
					EXEC DQ.LOG 'ERROR', @Cliente, @Servicio, @Hito,
@logMessage, '99.ERROR',@message
					DEALLOCATE tablas;
					RAISERROR (@message, 16, 1);
			END

			--4.2.Borrado tablas temporales
			EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logMessage,
'4.2.BORRADO TABLAS TEMPORALES'
			DROP TABLE IF EXISTS #TMP_NUMERO_BI
```

```
                    DROP TABLE IF EXISTS #TMP_NUMERO_OPE

                    --4.3.Creacion de tabla temporal de campo
                    EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logMessage,
'4.3.CREACION TABLA TEMPORAL CAMPOS'
                    CREATE TABLE #TMP_NUMERO_BI (
                           ID_CLIENTE INT,
                           ID_SERVICIO INT,
                           ID_HITO INT,
                           CAMPO_ID NVARCHAR(100),
                           VALOR_ID NVARCHAR(100)
                    )
                    CREATE TABLE #TMP_NUMERO_OPE (
                           ID_CLIENTE INT,
                           ID_SERVICIO INT,
                           ID_HITO INT,
                           CAMPO_ID NVARCHAR(100),
                           VALOR_ID NVARCHAR(100)
                    )

                    --4.4.Ejecucion de la validacion
                    EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logMessage,
'4.4.EJECUCIÓN DE LA VALIDACIÓN'
                    SELECT @COLUMN_ID= CAMPO, @COLUMN_REP=CAMPO_MOSTRAR_ERROR FROM
COD.PARAM_IDENTIFICADORES_TABLAS WHERE TABLA=@Tabla_Destino
                    SET @sqlTipoVenta = 'INSERT INTO #TMP_NUMERO_OPE
                                                        SELECT '+CAST(@CLIENTE AS
VARCHAR)+' AS ID_CLIENTE,
                                                                '+CAST(@SERVICIO AS
VARCHAR)+' AS ID_SERVICIO,
                                                                '+CAST(@HITO AS
VARCHAR)+' AS ID_HITO,
                                                                ''DIFIERE DEL
FICHERO DE BI'' AS CAMPO_ID,
                                                                B.'+@COLUMN_REP+'
AS VALOR_ID
                                                        FROM
REP.OPE_VENTAS_'+@serv+' B LEFT JOIN HOM.'+@Tabla_Destino+' A ON
A.'+@COLUMN_REP+'=B.'+@COLUMN_REP+'
                                                        WHERE (A.VENTA_ESPECIAL IS
NULL OR A.VENTA_ESPECIAL =''SI'') AND (A.POSTVENTA IS NULL OR A.POSTVENTA=''SI, NO
PRESENTE EN EL BI'')
                                                        AND B.VENTA_ESPECIAL=''NO''
AND (B.POSTVENTA=''SI'' OR B.POSTVENTA IS NULL)
                                                        AND A.'+@COLUMN_REP+' IS
NULL'
                    EXEC sp_executesql @sqlTipoVenta

                    --4.5.Insercion de validaciones
                    EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logMessage,
'4.5.INSERCION DE VALIDACION'
                    INSERT INTO DQ.VAL_NUMERO_ACTIVOS
                    SELECT
ID_CLIENTE,ID_SERVICIO,ID_HITO,@Tabla_Destino,CAMPO_ID,VALOR_ID
                    FROM #TMP_NUMERO_OPE
                    FETCH NEXT FROM tablas INTO @Tabla_Destino, @schema;
```

```
		END
		CLOSE tablas;
		DEALLOCATE tablas;

		--5.Comprobacion Estados
		EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logMessage,
'5.COMPROBACION ACTIVOS INCORRECTOS'
		SET @sqlCheck =
				'SELECT @count=COUNT(*)
				 FROM DQ.VAL_NUMERO_ACTIVOS
				 WHERE ID_CLIENTE=' +CAST(@cliente AS VARCHAR)+' AND
ID_SERVICIO='+CAST(@servicio AS VARCHAR)+' AND ID_HITO='+ CAST(@hito AS VARCHAR)+'
				 AND TABLA = ''' +@Tabla_Destino+''''
		EXEC sp_executesql @sqlCheck, N'@count INT OUTPUT', @count OUTPUT;
		IF @count =0
			BEGIN
				EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
@logMessage, '98.FIN VALIDACION NUMERO ACTIVOS';
			END
		ELSE
			BEGIN
				SET @message = '99.NUMERO DE ACTIVOS INCOHERENTE ' +
@Tabla_Destino;
				EXEC DQ.LOG 'ERROR', @Cliente, @Servicio, @Hito,
@logMessage,'99.ERROR', @message;
			END
	END TRY
	BEGIN CATCH
		IF CURSOR_STATUS('global','tablas')>=-1
		BEGIN
			DEALLOCATE tablas
		END
		SET @message = '99.ERROR EN LA COMPROBACIÓN DEL NUMERO DE ACTIVOS EN LA
TABLA '+@Tabla_Destino
		SET @error = ERROR_MESSAGE();
		EXEC DQ.LOG 'ERROR', @Cliente,@Servicio,@Hito, @logMessage,
'99.ERROR',@error
	END CATCH
END
```

DQ.VALIDACION_TIPO_VENTA

```
USE [AAM_SERVICIO_CCPP_TRIBUTOS]
GO
/****** Object:  StoredProcedure [DQ].[VALIDACION_TIPO_VENTA]    Script Date:
18/06/2024 16:49:09 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO


/*************************************************/
-- Autor: Ángel Guzmán
-- Fecha: 20240213
-- Descripción: Proceso de validación de estados
```

```
--            Inputs:
--                  Cliente
--                  Servicio
--                  Hito
--            Resultado de validaciones:
--                  DQ.VAL_CAMPOS_DUPLICADOS
--
-- Versiones:
--            V1: Versión inicial
/*********************************************/
ALTER PROCEDURE [DQ].[VALIDACION_TIPO_VENTA] (
       @Cliente int,
       @Servicio int,
       @Hito int
)
AS
BEGIN
       BEGIN TRY
        --##############################
              -- 1.DECLARACION DE VARIABLES #
              --##############################
              DECLARE @schema NVARCHAR(10),
                           @message NVARCHAR(MAX),
                           @Tabla_Destino NVARCHAR(100),
                           @column_id nvarchar(255),
                           @column_rep nvarchar(255),
                           @error int,
                           @sqlCheck NVARCHAR(MAX),
                           @sqlTipoVenta NVARCHAR(MAX),
                           @count int,
                           @logMessage NVARCHAR(255)='VALIDACION TIPO VENTA'
              EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logMessage, '1.INICIO
VALIDACIÓN'

              --##############################
              -- 2.TRUNCADO TABLA VALIDACION #
              --##############################
              EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logMessage, '2.TRUNCADO
TABLA DQ.VAL_TIPO_VENTA'
              DELETE FROM DQ.VAL_TIPO_VENTA WHERE ID_CLIENTE=@CLIENTE AND
ID_SERVICIO=@SERVICIO AND ID_HITO=@HITO

              --##############################
              -- 3.OBTENCIÓN TABLA A VALIDAR #
              --##############################
              EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logMessage,
'3.OBTENCIÓN TABLA A VALIDAR'
              IF CURSOR_STATUS('global','tablas')>=-1
              BEGIN
                     DEALLOCATE tablas
              END

              DECLARE tablas CURSOR FOR
              SELECT DISTINCT
                     TABLA,
                     ESQUEMA
```

```
		FROM COD.PARAM_VALIDACION_CAMPOS
		WHERE ID_CLIENTE = @Cliente AND ID_SERVICIO = @Servicio AND ID_HITO =
@Hito

		OPEN tablas
		FETCH NEXT FROM tablas INTO @Tabla_Destino, @schema

		--##########################
		-- 4.VALIDACION TIPO VENTA #
		--##########################
		EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logMessage, '4.INICIO
VALIDACION TIPO VENTA'
		WHILE @@FETCH_STATUS = 0
		BEGIN
			--4.1.Comprobacion de parametros vacios
			EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logMessage,
'4.1.COMPROBACION DE PARAMETROS VACIOS'
			IF @Tabla_Destino IS NULL OR @schema IS NULL
			BEGIN
					SET @message = '98.REVISE LOS PARAMETROS
INTRODUCIDOS EN EL ORQUESTADOR'
					SET @error = ERROR_NUMBER()
					EXEC DQ.LOG 'ERROR', @Cliente, @Servicio, @Hito,
@logMessage, '99.ERROR',@message
					DEALLOCATE tablas;
					RAISERROR (@message, 16, 1);
			END

			--4.2.Borrado tablas temporales
			EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logMessage,
'4.2.BORRADO TABLAS TEMPORALES'
			DROP TABLE IF EXISTS #TMP_TIPO_VENTA

			--4.3.Creacion de tabla temporal de campo
			EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logMessage,
'4.3.CREACION TABLA TEMPORAL CAMPOS'
			CREATE TABLE #TMP_TIPO_VENTA (
				ID_CLIENTE INT,
				ID_SERVICIO INT,
				ID_HITO INT,
				CAMPO_ID NVARCHAR(100),
				VALOR_ID NVARCHAR(100)
			)

			--4.4.Ejecucion de la validacion
			EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logMessage,
'4.4.EJECUCIÓN DE LA VALIDACIÓN'
			SELECT @COLUMN_ID= CAMPO, @COLUMN_REP=CAMPO_MOSTRAR_ERROR FROM
COD.PARAM_IDENTIFICADORES_TABLAS WHERE TABLA=@Tabla_Destino
			SET @sqlTipoVenta = 'INSERT INTO #TMP_TIPO_VENTA
								SELECT '+CAST(@CLIENTE AS
VARCHAR)+' AS ID_CLIENTE,
									'+CAST(@SERVICIO AS
VARCHAR)+' AS ID_SERVICIO,
									'+CAST(@HITO AS
VARCHAR)+' AS ID_HITO,
```

```
                                                                ''VENTA_ESPECIAL''
AS CAMPO_ID,
                                                                '+@COLUMN_ID+' AS
VALOR_ID
                                                        FROM HOM.'+@Tabla_Destino+'
                                                        WHERE (VENTA_ESPECIAL='''' OR
VENTA_ESPECIAL IS NULL OR VENTA_ESPECIAL NOT IN (''NO'',''SI''))'
                        EXEC sp_executesql @sqlTipoVenta

                        --4.5.Insercion de validaciones
                        EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logMessage,
'4.5.INSERCION DE VALIDACION'
                        INSERT INTO DQ.VAL_TIPO_VENTA
                        SELECT
ID_CLIENTE,ID_SERVICIO,ID_HITO,@Tabla_Destino,CAMPO_ID,VALOR_ID
                        FROM #TMP_TIPO_VENTA

                        FETCH NEXT FROM tablas INTO @Tabla_Destino, @schema;
                END
                CLOSE tablas;
                DEALLOCATE tablas;

                --5.Comprobacion Estados
                EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logMessage,
'5.COMPROBACION TIPOS DE VENTA INCORRECTOS'
                SET @sqlCheck =
                                'SELECT @count=COUNT(*)
                                 FROM DQ.VAL_TIPO_VENTA
                                 WHERE ID_CLIENTE=' +CAST(@cliente AS VARCHAR)+' AND
ID_SERVICIO='+CAST(@servicio AS VARCHAR)+' AND ID_HITO='+ CAST(@hito AS VARCHAR)+'
                                 AND TABLA = ''' +@Tabla_Destino+''''
                EXEC sp_executesql @sqlCheck, N'@count INT OUTPUT', @count OUTPUT;
                IF @count =0
                        BEGIN
                                EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,
@logMessage, '98.FIN VALIDACION TIPO DE VENTA';
                        END
                ELSE
                        BEGIN
                                SET @message = '99.TIPO DE VENTA VACIO EN LA TABLA' +
@Tabla_Destino;
                                EXEC DQ.LOG 'ERROR', @Cliente, @Servicio, @Hito,
@logMessage,'99.ERROR', @message;
                        END
        END TRY
        BEGIN CATCH
                IF CURSOR_STATUS('global','tablas')>=-1
                BEGIN
                        DEALLOCATE tablas
                END
                SET @message = '99.ERROR EN LA COMPROBACIÓN DEL TIPO DE VENTA EN LA
TABLA '+@Tabla_Destino
                SET @error = ERROR_MESSAGE();
                EXEC DQ.LOG 'ERROR', @Cliente,@Servicio,@Hito, @logMessage,
'99.ERROR',@error
        END CATCH
```

```
END
```

DQ.REP_VALIDACIONES

```
USE [AAM_SERVICIO_CCPP_TRIBUTOS]
GO
/****** Object:  StoredProcedure [DQ].[REP_VALIDACIONES]    Script Date: 18/06/2024
16:44:38 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

/*********************************************/
-- Autor: Jairo Jimenez
-- Fecha: 20240323
-- Descripción: Proceso que recoge todas las validaciones y las vuelca en una tabla
resumen
--     Tablas Inputs:
--             * DQ.VAL_ESTADOS
--             * DQ.VAL_DUPLICADOS
--             * DQ.VAL_CAMPOS_OBLIGATORIOS
--             * DQ.VAL_CAMPOS_HOM
--             * DQ.VAL_NEGOCIO
--             * DQ.VAL_TIPO_VENTA
--             Resultado de validaciones:
--                    DQ.VALIDACIONES
-- Versiones:
--             V1: Versión inicial
/*********************************************/
ALTER PROCEDURE [DQ].[REP_VALIDACIONES] (
       @cliente int,
       @servicio int,
       @hito int
)
AS
BEGIN
       BEGIN TRY
              DECLARE @logMessage NVARCHAR(500)
        DECLARE @logtitulo NVARCHAR(255)
              DECLARE @SERV NVARCHAR(4)
              DECLARE @sql NVARCHAR(MAX)

              SET @logtitulo='REPORTE VALIDACIONES'
              --##############################
              -- 0.DECLARACION DE VARIABLES #
              --##############################
        EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, @logtitulo, '0.INICIO REPORTE'

              --############################################
              -- 1.BORRADO PREVIO TABLA REPORTE VALIDACION #
              --############################################
              EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, '1.REPORTE
VALIDACIONES'
              DELETE FROM DQ.VALIDACIONES
              WHERE ID_CLIENTE=@CLIENTE AND ID_SERVICIO=@SERVICIO AND ID_HITO=@HITO
```

```
		--###################################
		-- 2.OBTENCION DE VARIABLE SERVICIO #
		--###################################
		EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, '2.OBTENCION
VARIABLE SERVICIO'
		SELECT @SERV=LEFT(SERVICIO,4)
		FROM COD.SERVICIOS
		WHERE ID_SERVICIO=@SERVICIO

		--######################
		-- 3.OBTENCION REPORTE #
		--######################
		EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, '3.OBTENCION
REPORTE'
		SET @sql='INSERT INTO DQ.VALIDACIONES
				SELECT '+CAST(@CLIENTE AS VARCHAR)+' AS ID_CLIENTE,
					'+CAST(@SERVICIO AS VARCHAR)+' AS
ID_SERVICION,
					'+CAST(@HITO AS VARCHAR)+' AS ID_HITO,
					A.*,
					X.ORIGEN AS ORIGEN
				FROM (
					--NEGOCIO
					SELECT DISTINCT C.SERVICIO,
							A.VALOR_ID,
							B.GESTOR_'+@serv+' AS GESTOR,
							''NEGOCIO'' AS TIPO_VAL,
							A.CAMPO,
							A.MENSAJE_ERROR
					FROM DQ.VAL_NEGOCIO AS A
					INNER JOIN REP.OPE_VENTAS_'+@serv+' AS B
					ON TRY_CAST(A.VALOR_ID AS NVARCHAR(255)) =
B.REF_UE
					INNER JOIN COD.SERVICIOS C
					ON A.ID_SERVICIO = C.ID_SERVICIO
					WHERE A.ID_CLIENTE='+CAST(@CLIENTE AS VARCHAR)+'
AND A.ID_SERVICIO='+CAST(@SERVICIO AS VARCHAR)+' AND A.ID_HITO='+CAST(@HITO AS
VARCHAR)+'
					UNION ALL
					--CAMPOS OBLIGATORIOS
					SELECT DISTINCT C.SERVICIO,
							A.VALOR_ID,
							B.GESTOR_'+@serv+' AS GESTOR,
							''CAMPOS OBLIGATORIOS'' AS TIPO_VAL,
							A.CAMPO,
							''VALOR NO INFORMADO'' AS
MENSAJE_ERROR
					FROM DQ.VAL_CAMPOS_OBLIGATORIOS AS A
					INNER JOIN REP.OPE_VENTAS_'+@serv+' AS B
					ON TRY_CAST(A.VALOR_ID AS NVARCHAR(255)) =
B.REF_UE
					INNER JOIN COD.SERVICIOS C
					ON A.ID_SERVICIO = C.ID_SERVICIO
```

```
                                    WHERE A.ID_CLIENTE='+CAST(@CLIENTE AS VARCHAR)+'
AND A.ID_SERVICIO='+CAST(@SERVICIO AS VARCHAR)+' AND A.ID_HITO='+CAST(@HITO AS
VARCHAR)+'
                                    UNION ALL
                                    --DUPLICADOS
                                    SELECT DISTINCT C.SERVICIO,
                                                A.VALOR_ID,
                                                B.GESTOR_'+@serv+' AS GESTOR,
                                                ''DUPLICADOS'' AS TIPO_VAL,
                                                A.CAMPO_ID AS CAMPO,
                                                ''VALOR DUPLICADO'' AS MENSAJE_ERROR
                                    FROM DQ.VAL_DUPLICADOS AS A
                                    FULL OUTER JOIN REP.OPE_VENTAS_'+@serv+' AS B
                                    ON TRY_CAST(A.VALOR_ID AS NVARCHAR(255)) =
B.REF_UE
                                    INNER JOIN COD.SERVICIOS C
                                    ON A.ID_SERVICIO = C.ID_SERVICIO
                                    WHERE A.ID_CLIENTE='+CAST(@CLIENTE AS VARCHAR)+'
AND A.ID_SERVICIO='+CAST(@SERVICIO AS VARCHAR)+' AND A.ID_HITO='+CAST(@HITO AS
VARCHAR)+'
                                    UNION ALL
                                    --ESTADOS
                                    SELECT DISTINCT C.SERVICIO,
                                                A.VALOR_ID,
                                                B.GESTOR_'+@serv+' AS GESTOR,
                                                ''ESTADOS'' AS TIPO_VAL,
                                                A.CAMPO_ID AS CAMPO,
                                                ''ESTADO INCORRECTO'' AS
MENSAJE_ERROR
                                    FROM DQ.VAL_ESTADOS AS A
                                    INNER JOIN REP.OPE_VENTAS_'+@serv+' AS B
                                    ON TRY_CAST(A.VALOR_ID AS NVARCHAR(255)) =
B.REF_UE
                                    INNER JOIN COD.SERVICIOS C
                                    ON A.ID_SERVICIO = C.ID_SERVICIO
                                    WHERE A.ID_CLIENTE='+CAST(@CLIENTE AS VARCHAR)+'
AND A.ID_SERVICIO='+CAST(@SERVICIO AS VARCHAR)+' AND A.ID_HITO='+CAST(@HITO AS
VARCHAR)+'
                                    UNION ALL
                                    --ESTADOS OK/KO
                                    SELECT DISTINCT C.SERVICIO,
                                                A.VALOR_ID,
                                                B.GESTOR_'+@serv+' AS GESTOR,
                                                ''ESTADOS'' AS TIPO_VAL,
                                                A.CAMPO_ID AS CAMPO,
                                                ''ESTADO OK/KO INCORRECTO'' AS
MENSAJE_ERROR
                                    FROM DQ.VAL_ESTADOS_OK_KO AS A
                                    INNER JOIN REP.OPE_VENTAS_'+@serv+' AS B
                                    ON TRY_CAST(A.VALOR_ID AS NVARCHAR(255)) =
B.REF_UE
                                    INNER JOIN COD.SERVICIOS C
                                    ON A.ID_SERVICIO = C.ID_SERVICIO
                                    WHERE A.ID_CLIENTE='+CAST(@CLIENTE AS VARCHAR)+'
AND A.ID_SERVICIO='+CAST(@SERVICIO AS VARCHAR)+' AND A.ID_HITO='+CAST(@HITO AS
VARCHAR)+'
```

```
                                UNION ALL
                                --TIPO VENTA
                                SELECT DISTINCT C.SERVICIO,
                                             A.VALOR_ID,
                                             B.GESTOR_'+@serv+' AS GESTOR,
                                             ''TIPO VENTA'' AS TIPO_VAL,
                                             A.CAMPO_ID AS CAMPO,
                                             ''TIPO VENTA NO INFORMADO'' AS
MENSAJE_ERROR
                                FROM DQ.VAL_TIPO_VENTA AS A
                                FULL OUTER JOIN REP.OPE_VENTAS_'+@serv+' AS B
                                ON TRY_CAST(A.VALOR_ID AS NVARCHAR(255)) =
B.REF_UE
                                INNER JOIN COD.SERVICIOS C
                                ON A.ID_SERVICIO = C.ID_SERVICIO
                                WHERE A.ID_CLIENTE='+CAST(@CLIENTE AS VARCHAR)+'
AND A.ID_SERVICIO='+CAST(@SERVICIO AS VARCHAR)+' AND A.ID_HITO='+CAST(@HITO AS
VARCHAR)+'
                                UNION ALL
                                --FORMATO
                                SELECT DISTINCT D.SERVICIO,
                                             A.VALOR_ID,
                                             B.GESTOR_'+@serv+' AS GESTOR,
                                             ''FORMATO'' AS TIPO_VAL,
                                             A.CAMPO AS CAMPO,
                                             ''FORMATO INCORRECTO'' AS
MENSAJE_ERROR
                                FROM DQ.VAL_CAMPOS_HOM AS A
                                INNER JOIN REP.OPE_VENTAS_'+@serv+' AS B
                                ON TRY_CAST(A.VALOR_ID AS NVARCHAR(255)) =
B.REF_UE
                                INNER JOIN COD.SERVICIOS D
                                ON A.ID_SERVICIO = D.ID_SERVICIO
                                WHERE A.ID_CLIENTE='+CAST(@CLIENTE AS VARCHAR)+'
AND A.ID_SERVICIO='+CAST(@SERVICIO AS VARCHAR)+' AND A.ID_HITO='+CAST(@HITO AS
VARCHAR)+'
                           ) A LEFT JOIN COD.PARAM_VALIDACION_CAMPOS X ON
A.CAMPO=X.CAMPO
                           AND X.ID_CLIENTE='+CAST(@CLIENTE AS VARCHAR)+' AND
X.ID_SERVICIO='+CAST(@SERVICIO AS VARCHAR)+' AND X.ID_HITO='+CAST(@HITO AS VARCHAR)+''
              EXEC (@sql)
              -- NUMERO ACTIVOS, ESTÁ A PARTE PORQUE NO LO PUEDO METER EN LA MISMA
QUERY (QUIZA ES DEMASIADO LARGA)
              SET @sql='INSERT INTO DQ.VALIDACIONES
                                --NUMERO ACTIVOS
                                SELECT '+CAST(@CLIENTE AS VARCHAR)+' AS
ID_CLIENTE,
                                       '+CAST(@SERVICIO AS VARCHAR)+' AS
ID_SERVICION,
                                       '+CAST(@HITO AS VARCHAR)+' AS ID_HITO,
                                       A.*,
                                       X.ORIGEN AS ORIGEN
                                FROM (
                                       SELECT DISTINCT C.SERVICIO,
                                       A.VALOR_ID,
                                       B.GESTOR_'+@serv+' AS GESTOR,
```

```
                                        ''NUMERO ACTIVOS'' AS TIPO_VAL,
                                        A.CAMPO_ID AS CAMPO,
                                        ''REVISAR SI EL ACTIVO ESTÁ PRESENTE EN EL
FICHERO, EN CASO DE ESTAR, COMPROBAR VENTA ESPECIAL ES "NO" Y POSTVENTA DISTINTO DE
"SI, NO PRESENTE EN EL BI"'' AS MENSAJE_ERROR
                                    FROM DQ.VAL_NUMERO_ACTIVOS AS A
                                    FULL OUTER JOIN REP.OPE_VENTAS_'+@serv+' AS B
                                    ON TRY_CAST(A.VALOR_ID AS NVARCHAR(255)) =
B.REF_UE
                                    INNER JOIN COD.SERVICIOS C
                                    ON A.ID_SERVICIO = C.ID_SERVICIO
                                    WHERE A.ID_CLIENTE='+CAST(@CLIENTE AS VARCHAR)+'
AND A.ID_SERVICIO='+CAST(@SERVICIO AS VARCHAR)+' AND A.ID_HITO='+CAST(@HITO AS
VARCHAR)+'
                                    UNION ALL
                                    --CAMPO POSTVENTA
                                    SELECT DISTINCT C.SERVICIO,
                                                A.VALOR_ID,
                                                B.GESTOR_'+@serv+' AS GESTOR,
                                                ''CAMPOS ERRONEOS'' AS TIPO_VAL,
                                                A.CAMPO_ID AS CAMPO,
                                                ''VALOR DE POSTVENTA INCORRECTO,
TIENE QUE SER NULO, "SI" O "SI, NO PRESENTE EN EL BI"'' AS MENSAJE_ERROR
                                    FROM DQ.VAL_CAMPO_POSTVENTA AS A
                                    FULL OUTER JOIN REP.OPE_VENTAS_'+@serv+' AS B
                                    ON TRY_CAST(A.VALOR_ID AS NVARCHAR(255)) =
B.REF_UE
                                    INNER JOIN COD.SERVICIOS C
                                    ON A.ID_SERVICIO = C.ID_SERVICIO
                                    WHERE A.ID_CLIENTE='+CAST(@CLIENTE AS VARCHAR)+'
AND A.ID_SERVICIO='+CAST(@SERVICIO AS VARCHAR)+' AND A.ID_HITO='+CAST(@HITO AS
VARCHAR)+'
                                    ) A
                                    LEFT JOIN COD.PARAM_VALIDACION_CAMPOS X ON
A.CAMPO=X.CAMPO
                                    AND X.ID_CLIENTE='+CAST(@CLIENTE AS VARCHAR)+' AND
X.ID_SERVICIO='+CAST(@SERVICIO AS VARCHAR)+' AND X.ID_HITO='+CAST(@HITO AS VARCHAR)+''
                EXEC (@sql)
                --3.1.Establecer Origen param los campos
                UPDATE DQ.VALIDACIONES
                SET ORIGEN ='NFORCE'
                WHERE ORIGEN IS NULL
                --3.2.Establecer Origen para los ID_OPORTUNIDAD que sean ventas normales
                SET @sql='UPDATE V
                                    SET ORIGEN=''NFORCE''
                                    FROM DQ.VALIDACIONES V
                                    LEFT JOIN REP.OPE_VENTAS_'+@serv+' B
                                    ON TRY_CAST(V.VALOR_ID AS NVARCHAR(255))=B.REF_UE
                                    INNER JOIN COD.SERVICIOS D
                                    ON V.ID_SERVICIO = D.ID_SERVICIO
                                    WHERE B.VENTA_ESPECIAL=''NO''
                                    AND (POSTVENTA IS NULL OR POSTVENTA=''SI'')
                                    AND V.CAMPO=''ID_OPORTUNIDAD''
                                    AND V.ID_CLIENTE='+CAST(@CLIENTE AS VARCHAR)+' AND
V.ID_SERVICIO='+CAST(@SERVICIO AS VARCHAR)+' AND V.ID_HITO='+CAST(@HITO AS VARCHAR)+''
                EXEC sp_executesql @sql
```

```
	END TRY
	BEGIN CATCH
		SET @logMessage=error_message()
		EXEC DQ.LOG 'ERROR', @Cliente, @Servicio, @Hito,
@logtitulo,'99.ERROR',@logMessage
		RAISERROR (@logMessage, 16, 1);
	END CATCH
END
```

REP.GENERACION_REP_VENTAS_OPE

```
USE [AAM_SERVICIO_CCPP_TRIBUTOS]
GO
/****** Object:  StoredProcedure [REP].[GENERACION_REP_VENTAS_OPE]    Script Date:
18/06/2024 16:42:31 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

/***********************************************/
/* Autor: Ángel Guzmán*/
/* Fecha: 20231212 */
/* Descripción:*/
/* Generación ficheros operativos a partir de las tablas históricas de ventas y lotes:
	1. REP.OPE_VENTAS_TRIB
	2. REP.OPE_VENTAS_CCPP*/
/* Versiones:*/
/*	V1: Versión inicial)*/
/***********************************************/
ALTER PROCEDURE [REP].[GENERACION_REP_VENTAS_OPE] (
	@Cliente int,
	@Servicio int,
	@Hito int,
	@tipo int
)
AS
BEGIN
	BEGIN TRY
		--################################
		--#    1.DECLARACION DE VARIABLES  #
		--################################
	 EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'GENERACION REPORTE VENTAS
OPERATIVO', '1.INICIO EJECUCION'
		DECLARE
			@message nvarchar(500),
			@error nvarchar(500)

		--############################
		--#    2.BORRADO TABLAS PREVIO     #
		--############################
		-- 1. Truncar tablas de reporte
		EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'GENERACION REPORTE VENTAS
OPERATIVO', '2.BORRADO PREVIO DE TABLAS'
		IF @CLIENTE=1
```

```
                    BEGIN
                           IF @SERVICIO=1
                           BEGIN
                                        TRUNCATE TABLE REP.OPE_VENTAS_TRIB
                                        DROP TABLE IF EXISTS
REP.AAM_MAESTRO_TRIB_VENTAS
                           END
                           IF @SERVICIO=2
                           BEGIN
                                        TRUNCATE TABLE REP.OPE_VENTAS_CCPP
                                        DROP TABLE IF EXISTS
REP.AAM_MAESTRO_CCPP_VENTAS
                           END

                    END

             --#############################################
             --#    3.GENERACION REPORTE VENTAS OPERATIVO     #
             --#############################################

             -- SI @tipo=1, el reporte tira del maestro, sino de HOM
             EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'GENERACION REPORTE VENTAS
OPERATIVO', '3.GENERACION REPORTE VENTAS OPERATIVO', NULL

             IF @Cliente=1
                    BEGIN
                           IF @servicio=1
                                  BEGIN
                                        IF @tipo=1
                                               BEGIN
                                                      --3.1. Reporte tributos de
Maestro
                                                      INSERT INTO
REP.OPE_VENTAS_TRIB
                                                      SELECT
                                                             A.ID_OPORTUNIDAD,
                                                             A.CONCATENADO,
                                                             A.REF_UE,
                                                             A.UE,
                                                             A.EDIFICIO,
                                                             A.DEPARTAMENTO,
                                                             A.SOC_CLIENTE,
                                                             A.FEC_POSICIONAMIENTO,

       A.PRIORIDAD_PATRIMONIO,
                                                             A.MUNICIPIO,
                                                             A.PROVINCIA,
                                                             A.GESTOR_NORMALIZADO,
                                                             A.SUBTIPOLOGIA,
                                                             A.REF_CATASTRAL,
                                                             A.NUMERO_RC,
                                                             A.RC_PTE,
                                                             A.NMB_COMPRADOR,
                                                             A.FEC_PREVISTA_FIRMA,
                                                             A.FR_AÑO,
                                                             A.PAO,
```

```
                                                                A.VBC,
                                                                A.ESTADO_TRIBUTOS,
                                                                A.FEC_CARGA_IBI,
                                                                A.DIAS_PTE_PAGO_IBI,

        A.IMP_TASAS_ANUALIZADO,

        A.[IMP_DEUDA/PRECIO_VENTA],
                                                                A.IMP_IBI_ANUALIZADO,
                                                                A.IMP_PRORRATA_VEND,
                                                                A.IMP_PRORRATA_COMP,

        A.FEC_INI_GESTION_TRIB,

        A.FEC_ULT_GESTION_TRIB,

        A.FEC_FIN_GESTION_TRIB,
                                                                A.FEC_ENVIO_PAGO_TRIB,
                                                                A.NUM_REMESA_TRIB,
                                                                A.PROVEEDOR,
                                                                A.GESTOR_TRIB,
                                                                A.COMENTARIOS_TRIB,
                                                                IIF(A.ESTADO_TRIBUTOS
IN (SELECT Estado FROM COD.ESTADO WHERE SERVICIO='TRIBUTOS' AND OK_KO='OK'), 'OK',
'KO') AS OK_KO_TRIB,

        DAY(A.FEC_POSICIONAMIENTO) AS DIA_FEC_VENTA,

        MONTH(A.FEC_POSICIONAMIENTO) AS MES_FEC_VENTA,

        YEAR(A.FEC_POSICIONAMIENTO) AS AÑO_FEC_VENTA,
                                                                A.VENTA_ESPECIAL AS
VENTA_ESPECIAL,
                                                                A.REVISAR_TRIB,
                                                                A.POSTVENTA,
                                                                CASE
                                                                        WHEN
LEN(A.REF_CATASTRAL) != 20 AND A.PROVINCIA NOT IN ('ÁLAVA', 'VIZCAYA', 'GUIPÚZCOA',
'NAVARRA') THEN 'INCORRECTA'
                                                                        WHEN
A.REF_CATASTRAL IS NULL THEN 'NO_RC'
                                                                        ELSE 'OK'
                                                                END AS RC_CORRECTA,
                                                                A.ORIGEN,
                                                                CASE
                                                                        WHEN EXISTS (
                                                                                SELECT 1
                                                                                FROM
REP.VAL_FEC_POSICIONAMIENTO V
                                                                                WHERE
V.CONCAT_SQL = A.REF_UE
                                                                                  AND
V.ID_OPORTUNIDAD = A.ID_OPORTUNIDAD
                                                                                  AND
(V.FEC_POSICIONAMIENTO_OLD > CAST(GETDATE() + 7 AS DATE) OR V.FEC_POSICIONAMIENTO_OLD
IS NULL)
```

```
                                                                                AND
V.FEC_POSICIONAMIENTO_NEW <= CAST(GETDATE() + 7 AS DATE)
                                                                        )
                                                                        THEN
'ADELANTADA A MENOS DE UNA SEMANA'
                                                                        ELSE NULL
                                                                END AS
VENTA_ADELANTADA
                                                        FROM
CALC.AAM_MAESTRO_TRIB_VENTAS A
                                                        WHERE A.ESTADO <> 'BAJA'

                                                        --
#############################################################
                                                        --#     4.ELIMINACIÓN
POSTVENTAS Y VENTAS ESPECIALES FINALIZADAS #
                                                        --
#############################################################

                                                        EXEC DQ.LOG 'INFO', '1',
NULL,'1', 'GENERACION REPORTE VENTAS OPERATIVO', '4.ELIMINACIÓN DE POSTVENTAS
FINALIZADAS', NULL
                                                        DELETE FROM
REP.OPE_VENTAS_TRIB
                                                        WHERE POSTVENTA = 'SI, NO
PRESENTE EN EL BI'
                                                        AND ESTADO_TRIBUTOS IN (
                                                                SELECT ESTADO
                                                                FROM COD.ESTADO
                                                                WHERE OK_KO = 'OK'
                                                        ) AND
ESTADO_TRIBUTOS!='ENVIADO_A_PAGO';
                                                        EXEC DQ.LOG 'INFO', '1',
NULL,'1', 'GENERACION REPORTE VENTAS OPERATIVO', '4.ELIMINACIÓN DE VENTAS ESPECIALES
FINALIZADAS', NULL
                                                        DELETE FROM
REP.OPE_VENTAS_TRIB
                                                        WHERE VENTA_ESPECIAL = 'SI'
                                                        AND ESTADO_TRIBUTOS IN (
                                                                SELECT ESTADO
                                                                FROM COD.ESTADO
                                                                WHERE OK_KO = 'OK'
                                                        ) AND
ESTADO_TRIBUTOS!='ENVIADO_A_PAGO'
                                                        ;
                                                        SELECT *
                                                        INTO
REP.AAM_MAESTRO_TRIB_VENTAS
                                                        FROM
CALC.AAM_MAESTRO_TRIB_VENTAS
                                                END
                                        ELSE
                                                BEGIN
                                                        --3.2. Reporte tributos de
HOM
```

```
                                                    INSERT INTO
REP.OPE_VENTAS_TRIB
                                                    SELECT
                                                           A.ID_OPORTUNIDAD,
                                                           A.CONCATENADO,
                                                           A.REF_UE,
                                                           A.UE,
                                                           A.EDIFICIO,
                                                           A.DEPARTAMENTO,
                                                           A.SOC_CLIENTE,
                                                           A.FEC_POSICIONAMIENTO,

       A.PRIORIDAD_PATRIMONIO,
                                                           A.MUNICIPIO,
                                                           A.PROVINCIA,
                                                           A.GESTOR_NORMALIZADO,
                                                           A.SUBTIPOLOGIA,
                                                           A.REF_CATASTRAL,
                                                           A.NUMERO_RC,
                                                           A.RC_PTE,
                                                           A.NMB_COMPRADOR,
                                                           A.FEC_PREVISTA_FIRMA,
                                                           A.FR_AÑO,
                                                           A.PAO,
                                                           A.VBC,
                                                           A.ESTADO_TRIBUTOS,
                                                           A.FEC_CARGA_IBI,
                                                           A.DIAS_PTE_PAGO_IBI,

       A.IMP_TASAS_ANUALIZADO,

       A.[IMP_DEUDA/PRECIO_VENTA],
                                                           A.IMP_IBI_ANUALIZADO,
                                                           A.IMP_PRORRATA_VEND,
                                                           A.IMP_PRORRATA_COMP,

       A.FEC_INI_GESTION_TRIB,

       A.FEC_ULT_GESTION_TRIB,

       A.FEC_FIN_GESTION_TRIB,
                                                           A.FEC_ENVIO_PAGO_TRIB,
                                                           A.NUM_REMESA_TRIB,
                                                           A.PROVEEDOR,
                                                           A.GESTOR_TRIB,
                                                           A.COMENTARIOS_TRIB,
                                                           IIF(A.ESTADO_TRIBUTOS
IN (SELECT Estado FROM COD.ESTADO WHERE SERVICIO='TRIBUTOS' AND OK_KO='OK'), 'OK',
'KO') AS OK_KO_TRIB,

       DAY(A.FEC_POSICIONAMIENTO) AS DIA_FEC_VENTA,

       MONTH(A.FEC_POSICIONAMIENTO) AS MES_FEC_VENTA,

       YEAR(A.FEC_POSICIONAMIENTO) AS AÑO_FEC_VENTA,
```

```
                                                                A.VENTA_ESPECIAL AS
VENTA_ESPECIAL,
                                                                A.REVISAR_TRIB,
                                                                A.POSTVENTA,
                                                                CASE
                                                                        WHEN
LEN(A.REF_CATASTRAL) != 20 AND A.PROVINCIA NOT IN ('ÁLAVA', 'VIZCAYA', 'GUIPÚZCOA',
'NAVARRA') THEN 'INCORRECTA'
                                                                        WHEN
A.REF_CATASTRAL IS NULL THEN 'NO_RC'
                                                                        ELSE 'OK'
                                                                END AS RC_CORRECTA,
                                                                'OPERATIVO' AS ORIGEN,
                                                                CASE
                                                                        WHEN EXISTS (
                                                                                SELECT 1
                                                                                FROM
REP.VAL_FEC_POSICIONAMIENTO V
                                                                                WHERE
V.CONCAT_SQL = A.REF_UE
                                                                                  AND
V.ID_OPORTUNIDAD = A.ID_OPORTUNIDAD
                                                                                  AND
(V.FEC_POSICIONAMIENTO_OLD > CAST(GETDATE() + 7 AS DATE) OR V.FEC_POSICIONAMIENTO_OLD
IS NULL)
                                                                                  AND
V.FEC_POSICIONAMIENTO_NEW <= CAST(GETDATE() + 7 AS DATE)
                                                                        )
                                                                        THEN
'ADELANTADA A MENOS DE UNA SEMANA'
                                                                        ELSE NULL
                                                                END AS
VENTA_ADELANTADA
                                                        FROM HOM.AAM_OPE_TRIB_VENTAS
A

                                                        DROP TABLE IF EXISTS
REP.AAM_MAESTRO_TRIB_VENTAS
                                                        SELECT *
                                                        INTO
REP.AAM_MAESTRO_TRIB_VENTAS
                                                        FROM
CALC.AAM_MAESTRO_TRIB_VENTAS
                                                END
                                        END
                                ELSE IF @Servicio=2
                                        BEGIN
                                                IF @tipo=1
                                                        BEGIN
                                                        --3.3. Reporte ccpp de
Maestro
                                                        INSERT INTO
REP.OPE_VENTAS_CCPP
                                                        SELECT
```

```
                                                        A.ID_OPORTUNIDAD,

                                                        A.CONCATENADO,
                                                        A.REF_UE,
                                                        A.UE,
                                                        A.EDIFICIO,
                                                        A.TIPO_CLIENTE,
                                                        A.SOC_PROPIETARIA,
                                                        A.CM,
                                                        A.FEC_POSICIONAMIENTO,
       A.PRIORIDAD_PATRIMONIO,
                                                        A.MAYORISTA_MINORISTA,
       A.GESTOR_FORMALIZACION,
                                                        A.MUNICIPIO,
                                                        A.PROVINCIA,
                                                        A.GESTOR_NORMALIZADO,
                                                        A.SUBTIPOLOGIA,
                                                        A.REF_CATASTRAL,
                                                        A.NMB_COMPRADOR,
                                                        A.FEC_PREVISTA_FIRMA,
                                                        A.FP_AÑO,
                                                        A.FR_AÑO,
                                                        A.ESTADO_COMUNIDADES,
                                                        A.COD_PROMOCION AS
COD_PROMOCION,
                                                        A.CIF AS CIF,
                                                        CASE WHEN A.CIF = 'NO
CP' THEN 0 ELSE LEN(A.CIF)-LEN(REPLACE(A.CIF, '/', ''))+1 END AS NUMERO_COMUNIDADES,
                                                        A.FEC_ENVIO_REMESA,
                                                        A.ETI_REMESA,
       A.FEC_INI_GESTION_CCPP,
       A.FEC_ULT_GESTION_CCPP,
                                                        A.GESTOR_CCPP,
                                                        A.FP_MAIL,
                                                        A.COMENTARIOS_CCPP,
                                                        A.MIX_PAO_VBC_CCPP,
       A.MOTIVO_NO_FIRMA_CCPP,
                                                        A.ALTA_PDTE,
                                                        A.FEC_ALTA_PDTE,
                                                        A.FEC_ALTA_FPF,
       IIF(A.ESTADO_COMUNIDADES IN (SELECT Estado FROM COD.ESTADO WHERE
SERVICIO='CCPP' AND OK_KO='OK'), 'OK', 'KO') AS OK_KO_CCPP,
                                                        A.VENTA_ESPECIAL AS
VENTA_ESPECIAL,
                                                        COALESCE(A.FP_MAIL,
A.FEC_POSICIONAMIENTO, A.FR_AÑO, A.FP_AÑO) AS FEC_REAL,
                                                        A.LOTE AS LOTE,
                                                        NULL AS
VENTA_BR_ID_DIR_CAR,
```

```
                                                                NULL AS
PROMO_O_ACTIVO,
                                                                A.REVISAR_CCPP,
                                                                A.POSTVENTA,
                                                                CASE
                                                                   WHEN
FEC_ULT_GESTION_CCPP < DATEADD(MONTH, -1, GETDATE()) AND ESTADO_COMUNIDADES IN ('OK
GFIA / NO CP') AND PRIORIDAD_PATRIMONIO <> 'V' THEN 'COMPROBAR ESTADO GESTIÓN'
                                                                   ELSE NULL
                                                           END AS CAMBIO_MES,
                                                           CASE
                                                                     WHEN EXISTS (
                                                                            SELECT 1
                                                                            FROM
REP.VAL_FEC_POSICIONAMIENTO V
                                                                            WHERE
V.CONCAT_SQL = A.REF_UE
                                                                              AND
V.ID_OPORTUNIDAD = A.ID_OPORTUNIDAD
                                                                              AND
(V.FEC_POSICIONAMIENTO_OLD > CAST(GETDATE() + 7 AS DATE) OR V.FEC_POSICIONAMIENTO_OLD
IS NULL)
                                                                              AND
V.FEC_POSICIONAMIENTO_NEW <= CAST(GETDATE() + 7 AS DATE)
                                                                     )
                                                                     THEN
'ADELANTADA A MENOS DE UNA SEMANA'
                                                                     ELSE NULL
                                                           END AS
VENTA_ADELANTADA
                                                    FROM
CALC.AAM_MAESTRO_CCPP_VENTAS A
                                                    --LEFT JOIN (
                                                    --     SELECT
                                                    --            CONCAT_SQL,
                                                    --
       MAX(CODIGO_PROMOCION) AS CODIGO_PROMOCION,
                                                    --
       MAX([CIF_NIF_COM]) AS [CIF_NIF_COM]
                                                    --     FROM STG.TAMIAS_CCPP
                                                    --     GROUP BY CONCAT_SQL
                                                    --) AS B ON A.CONCAT_SQL =
B.CONCAT_SQL
                                                    WHERE A.ESTADO <> 'BAJA'

                                                    --
##################################################################
                                                    --#    4.ELIMINACIÓN
POSTVENTAS Y VENTAS ESPECIALES FINALIZADAS #
                                                    --
##################################################################

                                                    EXEC DQ.LOG 'INFO', '1',
NULL,'1', 'GENERACION REPORTE VENTAS OPERATIVO', '4.ELIMINACIÓN DE POSTVENTAS
FINALIZADAS', NULL
```

```
                                                        DELETE FROM
REP.OPE_VENTAS_CCPP
                                                        WHERE POSTVENTA = 'SI, NO
PRESENTE EN EL BI'
                                                        AND ESTADO_COMUNIDADES ='OK
GFIA / NO CP'
                                                        EXEC DQ.LOG 'INFO', '1',
NULL,'1', 'GENERACION REPORTE VENTAS OPERATIVO', '4.ELIMINACIÓN DE VENTAS ESPECIALES
FINALIZADAS', NULL
                                                        DELETE FROM
REP.OPE_VENTAS_CCPP
                                                        WHERE VENTA_ESPECIAL = 'SI'
                                                        AND (POSTVENTA='SI' OR
PRIORIDAD_PATRIMONIO='V' OR FEC_POSICIONAMIENTO<CAST(GETDATE() AS DATE))
                                                        AND ESTADO_COMUNIDADES ='OK
GFIA / NO CP'

                                                        DROP TABLE IF EXISTS
REP.AAM_MAESTRO_CCPP_VENTAS
                                                        SELECT *
                                                        INTO
REP.AAM_MAESTRO_CCPP_VENTAS
                                                        FROM
CALC.AAM_MAESTRO_CCPP_VENTAS

                                                END
                                        ELSE
                                                BEGIN
                                                        --3.4. Reporte ccpp de HOM
                                                        INSERT INTO
REP.OPE_VENTAS_CCPP
                                                        SELECT
                                                                A.ID_OPORTUNIDAD,


                                                                A.CONCATENADO,
                                                                A.REF_UE,
                                                                A.UE,
                                                                A.EDIFICIO,
                                                                A.TIPO_CLIENTE,
                                                                A.SOC_PROPIETARIA,
                                                                A.CM,
                                                                A.FEC_POSICIONAMIENTO,

        A.PRIORIDAD_PATRIMONIO,
                                                                A.MAYORISTA_MINORISTA,

        A.GESTOR_FORMALIZACION,
                                                                A.MUNICIPIO,
                                                                A.PROVINCIA,
                                                                A.GESTOR_NORMALIZADO,
                                                                A.SUBTIPOLOGIA,
                                                                A.REF_CATASTRAL,
                                                                A.NMB_COMPRADOR,
                                                                A.FEC_PREVISTA_FIRMA,
                                                                A.FP_AÑO,
```

```
                                                                A.FR_AÑO,
                                                                A.ESTADO_COMUNIDADES,
                                                                A.COD_PROMOCION AS
COD_PROMOCION,
                                                                A.CIF AS CIF,
                                                                CASE WHEN A.CIF = 'NO
CP' THEN 0 ELSE LEN(A.CIF)-LEN(REPLACE(A.CIF, '/', ''))+1 END AS NUMERO_COMUNIDADES,
                                                                A.FEC_ENVIO_REMESA,
                                                                A.ETI_REMESA,

       A.FEC_INI_GESTION_CCPP,

       A.FEC_ULT_GESTION_CCPP,
                                                                A.GESTOR_CCPP,
                                                                A.FP_MAIL,
                                                                A.COMENTARIOS_CCPP,
                                                                A.MIX_PAO_VBC_CCPP,

       A.MOTIVO_NO_FIRMA_CCPP,
                                                                A.ALTA_PDTE,
                                                                A.FEC_ALTA_PDTE,
                                                                A.FEC_ALTA_FPF,

       IIF(A.ESTADO_COMUNIDADES IN (SELECT Estado FROM COD.ESTADO WHERE
SERVICIO='CCPP' AND OK_KO='OK'), 'OK', 'KO') AS OK_KO_CCPP,
                                                                A.VENTA_ESPECIAL AS
VENTA_ESPECIAL,
                                                                COALESCE(A.FP_MAIL,
A.FEC_POSICIONAMIENTO, A.FR_AÑO, A.FP_AÑO) AS FEC_REAL,
                                                                A.LOTE AS LOTE,
                                                                NULL AS
VENTA_BR_ID_DIR_CAR,
                                                                NULL AS
PROMO_O_ACTIVO,
                                                                A.REVISAR_CCPP,
                                                                A.POSTVENTA,
                                                                CASE
                                                                   WHEN
FEC_ULT_GESTION_CCPP < DATEADD(MONTH, -1, GETDATE()) AND ESTADO_COMUNIDADES IN ('OK
GFIA / NO CP') AND PRIORIDAD_PATRIMONIO <> 'V' THEN 'COMPROBAR ESTADO GESTIÓN'
                                                                   ELSE NULL
                                                        END AS CAMBIO_MES,
                                                        CASE
                                                                WHEN EXISTS (
                                                                       SELECT 1
                                                                       FROM
REP.VAL_FEC_POSICIONAMIENTO V
                                                                       WHERE
V.CONCAT_SQL = A.REF_UE
                                                                         AND
V.ID_OPORTUNIDAD = A.ID_OPORTUNIDAD
                                                                         AND
(V.FEC_POSICIONAMIENTO_OLD > CAST(GETDATE() + 7 AS DATE) OR V.FEC_POSICIONAMIENTO_OLD
IS NULL)
                                                                         AND
V.FEC_POSICIONAMIENTO_NEW <= CAST(GETDATE() + 7 AS DATE)
```

```
                                                                    )
                                                                    THEN
'ADELANTADA A MENOS DE UNA SEMANA'
                                                                    ELSE NULL
                                                            END AS
VENTA_ADELANTADA
                                                    FROM HOM.AAM_OPE_CCPP_VENTAS
A
                                                    --LEFT JOIN (
                                                    --      SELECT
                                                    --              CONCAT_SQL,
                                                    --
       MAX(CODIGO_PROMOCION) AS CODIGO_PROMOCION,
                                                    --
       MAX([CIF_NIF_COM]) AS [CIF_NIF_COM]
                                                    --      FROM STG.TAMIAS_CCPP
                                                    --      GROUP BY CONCAT_SQL
                                                    --) AS B ON A.CONCAT_SQL =
B.CONCAT_SQL

                                                    DROP TABLE IF EXISTS
REP.AAM_MAESTRO_CCPP_VENTAS
                                                    SELECT *
                                                    INTO
REP.AAM_MAESTRO_CCPP_VENTAS
                                                    FROM
CALC.AAM_MAESTRO_CCPP_VENTAS

                                                END
                                    END
                        END
       END TRY
       BEGIN CATCH
              SET @message = '99.ERROR GENERACIÓN REPORTE VENTAS OPERATIVO'
              SET @error = ERROR_MESSAGE()
              EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito,'GENERACION REPORTE VENTAS
OPERATIVO', @message, @error
       END CATCH
END
```

REP.GENERACION_REP_VENTAS_AAM

```
USE [AAM_SERVICIO_CCPP_TRIBUTOS]
GO
/****** Object:  StoredProcedure [REP].[GENERACION_REP_VENTAS_AAM]    Script Date:
18/06/2024 16:43:11 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

/*********************************************/
/* Autor: Ángel Guzmán*/
/* Fecha: 20231212 */
/* Descripción:*/
```

```
/* Una vez se cargan los ficheros operativos se generan tres tablas para reportar a
Altamira:
       1. REP.CLI_VENTAS
       2. REP.CLI_VENTAS_CCPP
       3. REP.CLI_VENTAS_TRIB*/
/* Versiones:*/
/*     V1: Versión inicial)*/
/*********************************************/
ALTER PROCEDURE [REP].[GENERACION_REP_VENTAS_AAM](
       @Cliente int,
       @Servicio int,
       @Hito int
)
AS
BEGIN
       BEGIN TRY

              --#############################
              -- 1.DECLARACION DE VARIABLES #
              --#############################

              DECLARE
                     @message nvarchar(500),
                     @error nvarchar(500)

              EXEC DQ.LOG 'INFO', @Cliente,@Servicio,@Hito, 'GENERACION REPORTES AAM',
'1.INICIO GENERACIÓN', NULL

              IF @cliente=1
                     BEGIN
                            IF @servicio=7
                                   BEGIN
                                          IF @hito=1
                                                 BEGIN

                                                        EXEC DQ.LOG 'INFO',
@Cliente,@Servicio,@Hito, 'GENERACION REPORTE VENTAS AAM', '1.INICIO GENERACIÓN', NULL

                                                        SET NOCOUNT ON;

                                                        --
############################
                                                        -- 2.BORRAR TABLAS TEMPORALES
#
                                                        --
############################

                                                        EXEC DQ.LOG
'INFO',@Cliente,@Servicio,@Hito, 'GENERACION REPORTE VENTAS AAM', '2.BORRAR TABLAS
TEMPORALES', NULL

                                                        DROP TABLE IF EXISTS
#TMP_REP_TRIB
                                                        DROP TABLE IF EXISTS
#TMP_REP_CCPP
```

```
                                                        --
##############################
                                                        -- 3.RELLENO TABLAS
TEMPORALES #
                                                        --
##############################

                                                        EXEC DQ.LOG
'INFO',@Cliente,@Servicio,@Hito, 'GENERACION REPORTE VENTAS AAM', '3.RELLENO TABLAS
TEMPORALES', NULL

                                                        --3.1.Registros del operativo
de tributos provenientes de BI
                                                        SELECT
                                                               *
                                                        INTO #TMP_REP_TRIB
                                                        FROM (
                                                               SELECT
                                                                      *
                                                               FROM
CALC.AAM_MAESTRO_TRIB_VENTAS A
                                                        ) B
                                                        WHERE (B.POSTVENTA IS NULL OR
B.POSTVENTA='SI') AND B.ESTADO!='BAJA' AND VENTA_ESPECIAL='NO'
                                                        --3.2.Registros del operativo
de comunidades provenientes de BI
                                                        SELECT
                                                               *
                                                        INTO #TMP_REP_CCPP
                                                        FROM (
                                                               SELECT
                                                                      *
                                                               FROM
CALC.AAM_MAESTRO_CCPP_VENTAS A
                                                        ) B
                                                        WHERE (B.POSTVENTA IS NULL OR
B.POSTVENTA='SI') AND B.ESTADO!='BAJA' AND VENTA_ESPECIAL='NO'

                                                        --############################
                                                        --#    4.BORRADO TABLAS
PREVIO #
                                                        --############################
                                                        -- 1. Truncar tablas de
reporte

                                                        EXEC DQ.LOG 'INFO',
@Cliente,@Servicio,@Hito, 'GENERACION REPORTE VENTAS AAM', '4.BORRADO PREVIO DE
TABLAS'

                                                        TRUNCATE TABLE REP.CLI_VENTAS
                                                        TRUNCATE TABLE
REP.CLI_VENTAS_BI

                                                        --############################
                                                        -- 5.RELLENO TABLA REPORTE  #
                                                        --############################
```

```
                                                    EXEC DQ.LOG
'INFO',@Cliente,@Servicio,@Hito, 'GENERACION REPORTE VENTAS AAM', '5.RELLENO TABLA
REPORTE GRAFICAS', NULL

                                                    INSERT INTO REP.CLI_VENTAS
                                                    SELECT

       COALESCE(A.ID_OPORTUNIDAD, B.ID_OPORTUNIDAD) AS ID_OPORTUNIDAD,

       COALESCE(A.CONCATENADO, B.CONCATENADO) AS CONCATENADO,
                                                                COALESCE(A.REF_UE,
B.REF_UE) AS REF_UE,
                                                                COALESCE(A.UE, B.UE)
AS UE,
                                                                COALESCE(A.EDIFICIO,
B.EDIFICIO) AS EDIFICIO,
                                                                COALESCE(A.REF_ORIGEN,
B.REF_ORIGEN) AS REF_ORIGEN,

       COALESCE(A.SOC_CLIENTE, B.SOC_CLIENTE) AS SOC_CLIENTE,

       COALESCE(A.TIPO_CLIENTE, B.TIPO_CLIENTE) AS TIPO_CLIENTE,

       COALESCE(A.SOC_PROPIETARIA, B.SOC_PROPIETARIA) AS SOC_PROPIETARIA,
                                                                COALESCE(A.CM, B.CM)
AS CM,

       COALESCE(A.FEC_POSICIONAMIENTO, B.FEC_POSICIONAMIENTO) AS FEC_POSICIONAMIENTO,
                                                                COALESCE(A.ESTADO_DFA,
B.ESTADO_DFA) AS ESTADO_DFA,

       COALESCE(A.FEC_PREVISTA_DFA, B.FEC_PREVISTA_DFA) AS FEC_PREVISTA_DFA,

       COALESCE(A.PRIORIDAD_PATRIMONIO, B.PRIORIDAD_PATRIMONIO) AS
PRIORIDAD_PATRIMONIO,

       COALESCE(A.MAYORISTA_MINORISTA, B.MAYORISTA_MINORISTA) AS MAYORISTA_MINORISTA,

       COALESCE(A.GESTOR_FORMALIZACION, B.GESTOR_FORMALIZACION) AS
GESTOR_FORMALIZACION,
                                                                COALESCE(A.DELEGACION,
B.DELEGACION) AS DELEGACION,
                                                                COALESCE(A.MUNICIPIO,
B.MUNICIPIO) AS MUNICIPIO,
                                                                COALESCE(A.PROVINCIA,
B.PROVINCIA) AS PROVINCIA,

       COALESCE(A.GESTOR_NORMALIZADO, B.GESTOR_NORMALIZADO) AS GESTOR_NORMALIZADO,

       COALESCE(A.TIPO_INMUEBLE, B.TIPO_INMUEBLE) AS TIPO_INMUEBLE,

       COALESCE(A.SUBTIPOLOGIA, B.SUBTIPOLOGIA) AS SUBTIPOLOGIA,

       COALESCE(A.REF_CATASTRAL, B.REF_CATASTRAL) AS REF_CATASTRAL,
```

```
       COALESCE(A.NMB_COMPRADOR, B.NMB_COMPRADOR) AS NMB_COMPRADOR,

       COALESCE(A.FEC_RESERVA, B.FEC_RESERVA) AS FEC_RESERVA,

       COALESCE(A.FEC_PREVISTA_FIRMA, B.FEC_PREVISTA_FIRMA) AS FEC_PREVISTA_FIRMA,
                                                              COALESCE(A.FP_AÑO,
B.FP_AÑO) AS FP_AÑO,
                                                              COALESCE(A.FR_AÑO,
B.FR_AÑO) AS FR_AÑO,
                                                              COALESCE(A.PAO, B.PAO)
AS PAO,
                                                              B.VBC AS VBC,
                                                              B.ESTADO_TRIBUTOS AS
ESTADO_TRIBUTOS,
                                                              B.FEC_CARGA_IBI AS
FEC_CARGA_IBI,
                                                              B.DIAS_PTE_PAGO_IBI AS
DIAS_PTE_PAGO_IBI,
                                                              B.IMP_TASAS_ANUALIZADO
AS IMP_TASAS_ANUALIZADO,

       B.[IMP_DEUDA/PRECIO_VENTA] AS [IMP_DEUDA/PRECIO_VENTA],
                                                              B.IMP_IBI_ANUALIZADO
AS IMP_IBI_ANUALIZADO,
                                                              B.IMP_PRORRATA_VEND AS
IMP_PRORRATA_VEND,
                                                              B.IMP_PRORRATA_COMP AS
IMP_PRORRATA_COMP,
                                                              B.FEC_INI_GESTION_TRIB
AS FEC_INI_GESTION_TRIB,
                                                              B.FEC_ULT_GESTION_TRIB
AS FEC_ULT_GESTION_TRIB,
                                                              B.FEC_FIN_GESTION_TRIB
AS FEC_FIN_GESTION_TRIB,
                                                              B.FEC_ENVIO_PAGO_TRIB
AS FEC_ENVIO_PAGO_TRIB,
                                                              B.NUM_REMESA_TRIB AS
NUM_REMESA_TRIB,
                                                              B.PROVEEDOR AS
PROVEEDOR,
                                                              B.GESTOR_TRIB AS
GESTOR_TRIB,
                                                              B.COMENTARIOS_TRIB AS
COMENTARIOS_TRIB,
                                                              A.ESTADO_COMUNIDADES
AS ESTADO_COMUNIDADES,
                                                              A.COD_PROMOCION AS
COD_PROMOCION,
                                                              A.CIF AS CIF,
                                                              A.FEC_ENVIO_REMESA AS
FEC_ENVIO_REMESA,
                                                              A.ETI_REMESA AS
ETI_REMESA,
                                                              A.FEC_INI_GESTION_CCPP
AS FEC_INI_GESTION_CCPP,
```

```
                                                                A.FEC_ULT_GESTION_CCPP
AS FEC_ULT_GESTION_CCPP,
                                                                A.GESTOR_CCPP AS
GESTOR_CCPP,
                                                                A.FP_MAIL AS FP_MAIL,
                                                                A.COMENTARIOS_CCPP AS
COMENTARIOS_CCPP,
                                                                A.MIX_PAO_VBC_CCPP AS
MIX_PAO_VBC_CCPP,
                                                                A.MOTIVO_NO_FIRMA_CCPP
AS MOTIVO_NO_FIRMA_CCPP,
                                                                A.ALTA_PDTE AS
ALTA_PDTE,
                                                                A.FEC_ALTA_PDTE AS
FEC_ALTA_PDTE,
                                                                A.FEC_ALTA_FPF AS
FEC_ALTA_FPF,

       COALESCE(A.DEPARTAMENTO,B.DEPARTAMENTO) AS CARTERA,
                                                                B.ESTADO_TRIB AS
OK_KO_TRIB,
                                                                A.ESTADO_CCPP AS
OK_KO_CCPP,
                                                                CASE
                                                                       WHEN
B.PRIORIDAD_PATRIMONIO = 'V' THEN DAY(B.FR_AÑO)
                                                                       ELSE
                                                                              CASE

       WHEN B.FEC_PREVISTA_FIRMA IS NOT NULL THEN DAY(B.FEC_PREVISTA_FIRMA)

       ELSE DAY(B.FP_AÑO)
                                                                              END
                                                                END AS DIA_REAL,
                                                                CASE
                                                                       WHEN
B.PRIORIDAD_PATRIMONIO = 'V' THEN MONTH(B.FR_AÑO)
                                                                       ELSE
                                                                              CASE

       WHEN B.FEC_PREVISTA_FIRMA IS NOT NULL THEN MONTH(B.FEC_PREVISTA_FIRMA)

       ELSE MONTH(B.FP_AÑO)
                                                                              END
                                                                END AS MES_REAL,
                                                                CASE
                                                                       WHEN
B.PRIORIDAD_PATRIMONIO = 'V' THEN YEAR(B.FR_AÑO)
                                                                       ELSE
                                                                              CASE

       WHEN B.FEC_PREVISTA_FIRMA IS NOT NULL THEN YEAR(B.FEC_PREVISTA_FIRMA)

       ELSE YEAR(B.FP_AÑO)
                                                                              END
                                                                END AS AÑO_REAL,
```

```
	DAY(COALESCE(A.FEC_POSICIONAMIENTO, B.FEC_POSICIONAMIENTO)) AS DIA_POS,

	MONTH(COALESCE(A.FEC_POSICIONAMIENTO, B.FEC_POSICIONAMIENTO)) AS MES_POS,

	YEAR(COALESCE(A.FEC_POSICIONAMIENTO, B.FEC_POSICIONAMIENTO)) AS AÑO_POS,
                                                                CASE
                                                                        WHEN
COALESCE(A.FEC_PREVISTA_FIRMA, B.FEC_PREVISTA_FIRMA) IS NOT NULL THEN 'SI'
                                                                        ELSE 'NO'
                                                                END AS
EXISTE_FEC_FIRMA
                                                        FROM #TMP_REP_CCPP A
                                                        FULL OUTER JOIN #TMP_REP_TRIB
B ON

	A.CONCAT_SQL=B.CONCAT_SQL

                                                        --
############################
                                                        -- 6.RELLENO TABLA REPORTE 2
#
                                                        --
############################

                                                        EXEC DQ.LOG
'INFO',@Cliente,@Servicio,@Hito, 'GENERACION REPORTE VENTAS AAM', '6.RELLENO TABLA
REPORTE BI', NULL

                                                        INSERT INTO REP.CLI_VENTAS_BI
                                                        SELECT

	COALESCE(A.ID_OPORTUNIDAD, B.ID_OPORTUNIDAD) AS ID_OPORTUNIDAD,

	COALESCE(A.CONCATENADO, B.CONCATENADO) AS CONCATENADO,
                                                                COALESCE(A.REF_UE,
B.REF_UE) AS REF_UE,
                                                                COALESCE(A.UE, B.UE)
AS UE,
                                                                COALESCE(A.EDIFICIO,
B.EDIFICIO) AS EDIFICIO,
                                                                COALESCE(A.REF_ORIGEN,
B.REF_ORIGEN) AS REF_ORIGEN,

	COALESCE(A.SOC_CLIENTE, B.SOC_CLIENTE) AS SOC_CLIENTE,

	COALESCE(A.TIPO_CLIENTE, B.TIPO_CLIENTE) AS TIPO_CLIENTE,

	COALESCE(A.SOC_PROPIETARIA, B.SOC_PROPIETARIA) AS SOC_PROPIETARIA,
                                                                COALESCE(A.CM, B.CM)
AS CM,

	COALESCE(A.FEC_POSICIONAMIENTO, B.FEC_POSICIONAMIENTO) AS FEC_POSICIONAMIENTO,
                                                                COALESCE(A.ESTADO_DFA,
B.ESTADO_DFA) AS ESTADO_DFA,
```

```
       COALESCE(A.FEC_PREVISTA_DFA, B.FEC_PREVISTA_DFA) AS FEC_PREVISTA_DFA,

       COALESCE(A.PRIORIDAD_PATRIMONIO, B.PRIORIDAD_PATRIMONIO) AS
PRIORIDAD_PATRIMONIO,

       COALESCE(A.MAYORISTA_MINORISTA, B.MAYORISTA_MINORISTA) AS MAYORISTA_MINORISTA,

       COALESCE(A.GESTOR_FORMALIZACION, B.GESTOR_FORMALIZACION) AS
GESTOR_FORMALIZACION,
                                                              COALESCE(A.DELEGACION,
B.DELEGACION) AS DELEGACION,
                                                              COALESCE(A.MUNICIPIO,
B.MUNICIPIO) AS MUNICIPIO,
                                                              COALESCE(A.PROVINCIA,
B.PROVINCIA) AS PROVINCIA,

       COALESCE(A.GESTOR_NORMALIZADO, B.GESTOR_NORMALIZADO) AS GESTOR_NORMALIZADO,

       COALESCE(A.TIPO_INMUEBLE, B.TIPO_INMUEBLE) AS TIPO_INMUEBLE,

       COALESCE(A.SUBTIPOLOGIA, B.SUBTIPOLOGIA) AS SUBTIPOLOGIA,

       COALESCE(A.REF_CATASTRAL, B.REF_CATASTRAL) AS REF_CATASTRAL,

       COALESCE(A.NMB_COMPRADOR, B.NMB_COMPRADOR) AS NMB_COMPRADOR,

       COALESCE(A.FEC_RESERVA, B.FEC_RESERVA) AS FEC_RESERVA,

       COALESCE(A.FEC_PREVISTA_FIRMA, B.FEC_PREVISTA_FIRMA) AS FEC_PREVISTA_FIRMA,
                                                              COALESCE(A.FP_AÑO,
B.FP_AÑO) AS FP_AÑO,
                                                              COALESCE(A.FR_AÑO,
B.FR_AÑO) AS FR_AÑO,
                                                              COALESCE(A.PAO, B.PAO)
AS PAO,
                                                              B.VBC AS VBC,
                                                              B.ESTADO_TRIBUTOS AS
ESTADO_TRIBUTOS,
                                                              B.FEC_CARGA_IBI AS
FEC_CARGA_IBI,
                                                              B.DIAS_PTE_PAGO_IBI AS
DIAS_PTE_PAGO_IBI,
                                                              B.IMP_TASAS_ANUALIZADO
AS IMP_TASAS_ANUALIZADO,

       B.[IMP_DEUDA/PRECIO_VENTA] AS [IMP_DEUDA/PRECIO_VENTA],
                                                              B.IMP_IBI_ANUALIZADO
AS IMP_IBI_ANUALIZADO,
                                                              B.IMP_PRORRATA_VEND AS
IMP_PRORRATA_VEND,
                                                              B.IMP_PRORRATA_COMP AS
IMP_PRORRATA_COMP,
                                                              B.FEC_INI_GESTION_TRIB
AS FEC_INI_GESTION_TRIB,
```

```
                                                                B.FEC_ULT_GESTION_TRIB
AS FEC_ULT_GESTION_TRIB,
                                                                B.FEC_FIN_GESTION_TRIB
AS FEC_FIN_GESTION_TRIB,
                                                                B.FEC_ENVIO_PAGO_TRIB
AS FEC_ENVIO_PAGO_TRIB,
                                                                B.NUM_REMESA_TRIB AS
NUM_REMESA_TRIB,
                                                                B.PROVEEDOR AS
PROVEEDOR,
                                                                B.GESTOR_TRIB AS
GESTOR_TRIB,
                                                                B.COMENTARIOS_TRIB AS
COMENTARIOS_TRIB,
                                                                A.ESTADO_COMUNIDADES
AS ESTADO_COMUNIDADES,
                                                                A.COD_PROMOCION AS
COD_PROMOCION,

                                                                A.CIF AS CIF,
                                                                A.FEC_ENVIO_REMESA AS
FEC_ENVIO_REMESA,
                                                                A.ETI_REMESA AS
ETI_REMESA,
                                                                A.FEC_INI_GESTION_CCPP
AS FEC_INI_GESTION_CCPP,
                                                                A.FEC_ULT_GESTION_CCPP
AS FEC_ULT_GESTION_CCPP,
                                                                A.GESTOR_CCPP AS
GESTOR_CCPP,

                                                                A.FP_MAIL AS FP_MAIL,
                                                                A.COMENTARIOS_CCPP AS
COMENTARIOS_CCPP,
                                                                A.MIX_PAO_VBC_CCPP AS
MIX_PAO_VBC_CCPP,
                                                                A.MOTIVO_NO_FIRMA_CCPP
AS MOTIVO_NO_FIRMA_CCPP,
                                                                A.ALTA_PDTE AS
ALTA_PDTE,
                                                                A.FEC_ALTA_PDTE AS
FEC_ALTA_PDTE,
                                                                A.FEC_ALTA_FPF AS
FEC_ALTA_FPF
                                                        FROM #TMP_REP_CCPP A
                                                        FULL OUTER JOIN #TMP_REP_TRIB
B ON

       A.CONCAT_SQL=B.CONCAT_SQL

                                                        --
############################################################
                                                        -- 7.RELLENO TABLAS REPORTE
GRÁFICAS: FEC_POS, ALTAS, BAJAS  #
                                                        --
############################################################
```

```
                                                    EXEC DQ.LOG
'INFO',@Cliente,@Servicio,@Hito, 'GENERACION REPORTE VENTAS AAM', '7.RELLENO REPORTES
BAJAS, ALTAS Y CAMBIOS FEC_POSICIONAMIENTO DEL MES', NULL

                                                    EXEC DQ.LOG
'INFO',@Cliente,@Servicio,@Hito, 'GENERACION REPORTE VENTAS AAM', '7.1.BORRADO TABLAS
TEMPORALES', NULL

                                                    DROP TABLE IF EXISTS
REP.VAL_FEC_POSICIONAMIENTO;
                                                    DROP TABLE IF EXISTS
REP.AAM_BAJAS_MAESTRO_VENTAS
                                                    DROP TABLE IF EXISTS
#CTE_Maestro_Trib_Ventas;
                                                    DROP TABLE IF EXISTS
#CTE_Maestro_CCPP_Ventas;
                                                    DROP TABLE IF EXISTS
REP.AAM_ALTAS_MAESTRO_VENTAS

                                                    EXEC DQ.LOG
'INFO',@Cliente,@Servicio,@Hito, 'GENERACION REPORTE VENTAS AAM', '7.2.OBTENCION
ÚLTIMOS REGISTROS DE LOS MAESTROS', NULL

                                                    SELECT
                                                           *,
                                                           ROW_NUMBER() OVER
(PARTITION BY REF_UE, ID_OPORTUNIDAD ORDER BY FEC_FICHERO DESC) AS rn
                                                    INTO #CTE_Maestro_Trib_Ventas
                                                    FROM

       CALC.AAM_MAESTRO_TRIB_VENTAS

                                                    SELECT
                                                           *,
                                                           ROW_NUMBER() OVER
(PARTITION BY REF_UE, ID_OPORTUNIDAD ORDER BY FEC_FICHERO DESC) AS rn
                                                    INTO #CTE_Maestro_CCPP_Ventas
                                                    FROM

       CALC.AAM_MAESTRO_CCPP_VENTAS

                                                    EXEC DQ.LOG
'INFO',@Cliente,@Servicio,@Hito, 'GENERACION REPORTE VENTAS AAM', '7.3.RELLENO
REP.VAL_FEC_POSICIONAMIENTO', NULL

                                                    SELECT
                                                           A.CONCAT_SQL,
                                                           A.ID_OPORTUNIDAD,

       DAY(A.FEC_POSICIONAMIENTO_NEW) AS DIA_POS_NEW,

       MONTH(A.FEC_POSICIONAMIENTO_NEW) AS MES_POS_NEW,

       YEAR(A.FEC_POSICIONAMIENTO_NEW) AS AÑO_POS_NEW,

       DAY(A.FEC_POSICIONAMIENTO_OLD) AS DIA_POS_OLD,
```

```
	MONTH(A.FEC_POSICIONAMIENTO_OLD) AS MES_POS_OLD,

	YEAR(A.FEC_POSICIONAMIENTO_OLD) AS AÑO_POS_OLD,
		DAY(A.FEC_CAMBIO) AS
DIA_CAMBIO,
		MONTH(A.FEC_CAMBIO) AS
MES_CAMBIO,
		YEAR(A.FEC_CAMBIO) AS
AÑO_CAMBIO,

	A.FEC_POSICIONAMIENTO_NEW,

	A.FEC_POSICIONAMIENTO_OLD,
		B.ESTADO_TRIB,
		C.ESTADO_CCPP
	INTO
REP.VAL_FEC_POSICIONAMIENTO
	FROM

	CALC.VAL_FEC_POSICIONAMIENTO A
	LEFT JOIN

	#CTE_Maestro_Trib_Ventas B ON A.CONCAT_SQL = B.REF_UE AND A.ID_OPORTUNIDAD =
B.ID_OPORTUNIDAD AND B.rn = 1
	LEFT JOIN

	#CTE_Maestro_CCPP_Ventas C ON A.CONCAT_SQL = C.REF_UE AND A.ID_OPORTUNIDAD =
C.ID_OPORTUNIDAD AND C.rn = 1
	WHERE
		A.ID_SERVICIO = 1
		AND
MONTH(A.FEC_CAMBIO) = MONTH(CAST(GETDATE() AS DATE))
		AND YEAR(A.FEC_CAMBIO)
= YEAR(CAST(GETDATE() AS DATE));

	EXEC DQ.LOG
'INFO',@Cliente,@Servicio,@Hito, 'GENERACION REPORTE VENTAS AAM', '7.4.RELLENO
REP.AAM_ALTAS_MAESTRO_VENTAS', NULL

	SELECT DISTINCT
		A.ID_OPORTUNIDAD,
		A.REF_UE,

	DAY(A.FEC_POSICIONAMIENTO) AS DIA_POS,

	MONTH(A.FEC_POSICIONAMIENTO) AS MES_POS,

	YEAR(A.FEC_POSICIONAMIENTO) AS AÑO_POS,
		DAY(A.FEC_FICHERO) AS
DIA_SALIDA,
		MONTH(A.FEC_FICHERO)
AS MES_SALIDA,
		YEAR(A.FEC_FICHERO) AS
AÑO_SALIDA,
		B.ESTADO_TRIB,
```

```
                                                        C.ESTADO_CCPP
                                                    INTO
REP.AAM_ALTAS_MAESTRO_VENTAS
                                                    FROM
                                                        HIS.AAM_BI_VENTAS A
                                                    LEFT JOIN

        #CTE_Maestro_Trib_Ventas B ON A.CONCAT_SQL = B.REF_UE AND A.ID_OPORTUNIDAD =
B.ID_OPORTUNIDAD AND B.rn = 1
                                                    LEFT JOIN

        #CTE_Maestro_CCPP_Ventas C ON A.CONCAT_SQL = C.REF_UE AND A.ID_OPORTUNIDAD =
C.ID_OPORTUNIDAD AND C.rn = 1
                                                    WHERE
                                                        A.ESTADO IN
('ALTA','REALTA')
                                                        AND
MONTH(A.FEC_FICHERO) = MONTH(CAST(GETDATE() AS DATE))
                                                        AND
YEAR(A.FEC_FICHERO) = YEAR(CAST(GETDATE() AS DATE));

                                                    EXEC DQ.LOG
'INFO',@Cliente,@Servicio,@Hito, 'GENERACION REPORTE VENTAS AAM', '7.5.RELLENO
REP.AAM_BAJAS_MAESTRO_VENTAS', NULL

                                                    SELECT DISTINCT
                                                        A.ID_OPORTUNIDAD,
                                                        A.REF_UE,

        DAY(A.FEC_POSICIONAMIENTO) AS DIA_POS,

        MONTH(A.FEC_POSICIONAMIENTO) AS MES_POS,

        YEAR(A.FEC_POSICIONAMIENTO) AS AÑO_POS,
                                                        DAY(A.FEC_FICHERO) AS
DIA_SALIDA,
                                                        MONTH(A.FEC_FICHERO)
AS MES_SALIDA,
                                                        YEAR(A.FEC_FICHERO) AS
AÑO_SALIDA,
                                                        B.ESTADO_TRIB,
                                                        C.ESTADO_CCPP
                                                    INTO
REP.AAM_BAJAS_MAESTRO_VENTAS
                                                    FROM HIS.AAM_BI_VENTAS A
                                                    LEFT JOIN

        #CTE_Maestro_Trib_Ventas B ON A.CONCAT_SQL = B.REF_UE AND A.ID_OPORTUNIDAD =
B.ID_OPORTUNIDAD AND B.rn = 1
                                                    LEFT JOIN

        #CTE_Maestro_CCPP_Ventas C ON A.CONCAT_SQL = C.REF_UE AND A.ID_OPORTUNIDAD =
C.ID_OPORTUNIDAD AND C.rn = 1
                                                    WHERE
                                                        A.ESTADO='BAJA'
```

```
                                                                    AND
MONTH(A.FEC_FICHERO) =MONTH(CAST(GETDATE() AS DATE))
                                                                    AND
YEAR(A.FEC_FICHERO)=YEAR(CAST(GETDATE() AS DATE))
                                                        END
                                        END
                        END
        END TRY
        BEGIN CATCH
                SET @message = '99.ERROR GENERACION REPORTE'
                SET @error = ERROR_MESSAGE()
                EXEC DQ.LOG 'ERROR',@Cliente,@Servicio,@Hito, 'GENERACIÓN REPORTES AAM',
@message, @error
        END CATCH
END
```

REP.ACTUALIZAR_FACURACION_VENTAS

```
USE [AAM_SERVICIO_CCPP_TRIBUTOS]
GO
/****** Object:  StoredProcedure [REP].[ACTUALIZAR_FACTURACION_VENTAS]    Script Date:
18/06/2024 16:43:43 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
/*************************************************/
 --Autor: Ángel Guzmán
 --Fecha: 20240416
 --Descripción: Proceso que rellena la tabla REP.AAM_FACTURACION_VENTAS
 --Versiones:
        --      V1: Versión inicial
/*************************************************/
ALTER PROCEDURE [REP].[ACTUALIZAR_FACTURACION_VENTAS] (
        -- Add the parameters for the stored procedure here
        @cliente int,
        @servicio int,
        @hito int,
        @FCH_DATOS_1 DATE
)
AS
BEGIN

        SET NOCOUNT ON;

        --###############################
        --#     1.DECLARACION DE VARIABLES  #
        --###############################
        --1.1. Obtencion variables
        DECLARE @logtitulo NVARCHAR(255),
                @serv NVARCHAR(255),
                @servicio_nmb NVARCHAR(255),
                @sql NVARCHAR(MAX),
                @FCH_DATOS DATE = DATEADD(DAY, 1, @FCH_DATOS_1)

        SET @logtitulo='FACTURACION VENTAS'
```

```
	EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, '1.INICIO
EJECUCION'
	EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, '1.2.OBTENCION
VARIABLE SERVICIO'
	--1.2. Obtencion variables
	SELECT @SERV=LEFT(SERVICIO,4)
	FROM COD.SERVICIOS
	WHERE ID_SERVICIO=@SERVICIO
	IF @SERVICIO = 1
		SET @servicio_nmb='TRIBUTOS'
	ELSE
		SET @servicio_nmb='COMUNIDADES'

	--##############################################
	-- 2.RELLENAR TABLA REP.AAM_FACTURACION_VENTAS #
	--##############################################
	--2.1.Borrado Previo
	DELETE FROM REP.AAM_FACTURACION_VENTAS WHERE FEC_FACT>=@FCH_DATOS_1 AND
ID_CLIENTE=@cliente AND ID_SERVICIO=@servicio AND ID_HITO=@hito
	--2.2.Relleno tabla
	EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, '2.RELLENO TABLA
REP.AAM_FACTURACION_VENTAS CON LO FACTURADO EN EL DIA'
	SET @sql='INSERT INTO REP.AAM_FACTURACION_VENTAS
			SELECT
				'+CAST(@CLIENTE AS VARCHAR)+' AS ID_CLIENTE,
			'+CAST(@SERVICIO AS VARCHAR)+' AS ID_SERVICIO,
			'+CAST(@HITO AS VARCHAR)+' AS ID_HITO,
				ID_OPORTUNIDAD,
				CONCAT_SQL AS REF_UE,
				DEPARTAMENTO,
				SOC_CLIENTE,
				TIPO_CLIENTE,
				ESTADO_'+@servicio_nmb+' AS ESTADO,
				FEC_POSICIONAMIENTO,
				PRIORIDAD_PATRIMONIO,
				FEC_FACT_'+@serv+' AS FEC_FACT,
				VENTA_ESPECIAL,
				POSTVENTA,
				NULL AS ESTADO_FINAL
			FROM CALC.AAM_MAESTRO_'+@serv+'_VENTAS
			WHERE FEC_FACT_'+@serv+'='''+CAST(@FCH_DATOS_1 AS
NVARCHAR(50))+'''
				'
	EXEC sp_executesql @sql

	--#####################################################################
	-- 3. ACTUALIZACION CAMBIOS FEC_POSICIONAMIENTO, ESTADO, PRIORIDAD #
	--#####################################################################
	EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, '3.ACTUALIZACION
CAMBIOS FEC_POSICIONAMIENTO'
	SET @sql='UPDATE A
			SET A.ESTADO = M.ESTADO_'+@servicio_nmb+',
				A.PRIORIDAD_PATRIMONIO=M.PRIORIDAD_PATRIMONIO,
				A.FEC_POSICIONAMIENTO=M.FEC_POSICIONAMIENTO
			FROM REP.AAM_FACTURACION_VENTAS A
			INNER JOIN CALC.AAM_MAESTRO_'+@serv+'_VENTAS M
```

```
                        ON A.ID_OPORTUNIDAD=M.ID_OPORTUNIDAD
                               AND A.REF_UE=M.REF_UE
                        WHERE A.FEC_FACT=M.FEC_FACT_'+@serv+'
                               AND A.ID_SERVICIO='+CAST(@SERVICIO AS
NVARCHAR(20))+'
        '
        EXEC sp_executesql @sql

        --################################
        -- 4. ACTUALIZACION ESTADO_FINAL #
        --################################
        EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, ' 4.ACTUALIZACION
ESTADO_FINAL'
        SET @sql='UPDATE R
        SET
               ESTADO_FINAL=
                      CASE
                        WHEN R.VENTA_ESPECIAL=''SI'' THEN ''VENTA FUERA DE
FICHERO''
                        WHEN R.VENTA_ESPECIAL=''NO'' AND M.ESTADO=''BAJA'' AND
(R.POSTVENTA IS NULL OR R.POSTVENTA<>''SI, NO PRESENTE EN EL BI'') THEN ''SALIDAS DE
FICHERO (VENTAS EN ESTADO OK)''
                        WHEN R.POSTVENTA=''SI, NO PRESENTE EN EL BI'' OR
(R.PRIORIDAD_PATRIMONIO=''V'' AND ((YEAR(R.FEC_POSICIONAMIENTO)=YEAR(GETDATE()) AND
MONTH(R.FEC_POSICIONAMIENTO)<MONTH(GETDATE())) OR
(YEAR(R.FEC_POSICIONAMIENTO)<YEAR(GETDATE())))) THEN ''POSTVENTA DE MESES ANTERIORES''
                        WHEN R.PRIORIDAD_PATRIMONIO=''V'' AND
MONTH(R.FEC_POSICIONAMIENTO)=MONTH(GETDATE()) AND
YEAR(R.FEC_POSICIONAMIENTO)=YEAR(GETDATE()) THEN ''VENTA/POSTVENTA MES EN CURSO
(VENDIDOS)''
                        ELSE ''VENTAS GESTIONADAS MES EN CURSO''
                      END
        FROM REP.AAM_FACTURACION_VENTAS R
        LEFT JOIN CALC.AAM_MAESTRO_'+@serv+'_VENTAS M
        ON R.REF_UE=M.REF_UE
               AND R.ID_OPORTUNIDAD=M.ID_OPORTUNIDAD
               AND R.FEC_FACT=M.FEC_FACT_'+@serv+'
               AND R.ID_SERVICIO='+CAST(@SERVICIO AS NVARCHAR(20))+'
        WHERE MONTH(R.FEC_FACT)=MONTH(GETDATE())
        '
        EXEC sp_executesql @sql

        --#######################################
        -- 5. RELLENO TABLAS A REPORTAR A DIARIO #
        --#######################################

        EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, ' 5.1.RELLENO TABLA
A REPORTAR FACTURACIÓN VENTAS'

        DROP TABLE IF EXISTS REP.AAM_FACTURACION_VENTAS_DIARIO
        DROP TABLE IF EXISTS REP.AAM_FACTURACION_VENTAS_REPORTE

        SELECT
               CASE WHEN ID_SERVICIO=1 THEN 'TRIBUTOS' ELSE 'CCPP' END AS SERVICIO,
               ID_OPORTUNIDAD,
               REF_UE,
```

```
		DEPARTAMENTO,
		TIPO_CLIENTE,
		ESTADO,
		FEC_POSICIONAMIENTO,
		PRIORIDAD_PATRIMONIO,
		FEC_FACT,
		VENTA_ESPECIAL,
		POSTVENTA,
		ESTADO_FINAL
	INTO REP.AAM_FACTURACION_VENTAS_DIARIO
	FROM REP.AAM_FACTURACION_VENTAS
	WHERE YEAR(FEC_FACT)=YEAR(@FCH_DATOS_1) AND MONTH(FEC_FACT)=MONTH(@FCH_DATOS_1)

	EXEC DQ.LOG 'INFO', @Cliente, @Servicio, @Hito, @logtitulo, ' 5.2.RELLENO TABLA
INFORME FACTURACION - REPORTE AAM'
	;
	-- Ventas presentes en BI
	WITH CTE_MAESTRO_TRIB AS (
		SELECT*FROM(
			SELECT
				*,
				ROW_NUMBER() OVER (PARTITION BY CONCAT_SQL ORDER BY
FEC_FICHERO DESC) AS RN
			FROM
				CALC.AAM_MAESTRO_TRIB_VENTAS
			WHERE
				VENTA_ESPECIAL='NO' AND (POSTVENTA IS NULL OR
POSTVENTA='SI') AND ESTADO<>'BAJA'
		) A WHERE RN=1
	),
	CTE_MAESTRO_CCPP AS (
		SELECT*FROM(
			SELECT
				*,
				ROW_NUMBER() OVER (PARTITION BY CONCAT_SQL ORDER BY
FEC_FICHERO DESC) AS RN
			FROM
				CALC.AAM_MAESTRO_CCPP_VENTAS
			WHERE
				VENTA_ESPECIAL='NO' AND (POSTVENTA IS NULL OR
POSTVENTA='SI') AND ESTADO<>'BAJA'
		) A WHERE RN=1
	)
	SELECT
		COALESCE(A.ID_OPORTUNIDAD, B.ID_OPORTUNIDAD) AS ID_OPORTUNIDAD,
		COALESCE(A.REF_UE, B.REF_UE) AS REF_UE,
		A.ESTADO_TRIBUTOS,
		B.ESTADO_COMUNIDADES,
		COALESCE(A.FEC_POSICIONAMIENTO,B.FEC_POSICIONAMIENTO) AS
FEC_POSICIONAMIENTO,
		COALESCE(DAY(A.FEC_POSICIONAMIENTO), DAY(B.FEC_POSICIONAMIENTO)) AS
DIA_POS,
		COALESCE(MONTH(A.FEC_POSICIONAMIENTO), MONTH(B.FEC_POSICIONAMIENTO)) AS
MES_POS,
		COALESCE(YEAR(A.FEC_POSICIONAMIENTO), YEAR(B.FEC_POSICIONAMIENTO)) AS
AÑO_POS,
```

```
		FEC_FACT_TRIB,
		DAY(FEC_FACT_TRIB) AS DIA_FACT_TRIB,
		MONTH(FEC_FACT_TRIB) AS MES_FACT_TRIB,
		YEAR(FEC_FACT_TRIB) AS AÑO_FACT_TRIB,
		FEC_FACT_CCPP,
		DAY(FEC_FACT_CCPP) AS DIA_FACT_CCPP,
		MONTH(FEC_FACT_CCPP) AS MES_FACT_CCPP,
		YEAR(FEC_FACT_CCPP) AS AÑO_FACT_CCPP,
		A.VENTA_ESPECIAL AS VENTA_ESPECIAL_TRIB,
		A.POSTVENTA AS POSTVENTA_TRIB,
		A.ESTADO_TRIB AS OK_KO_TRIB,
		A.REVISAR_TRIB,
		B.VENTA_ESPECIAL AS VENTA_ESPECIAL_CCPP,
		B.POSTVENTA AS POSTVENTA_CCPP,
		B.ESTADO_CCPP AS OK_KO_CCPP,
		B.REVISAR_CCPP,
		COALESCE(A.ESTADO,B.ESTADO) AS ESTADO
	INTO REP.AAM_FACTURACION_VENTAS_REPORTE
	FROM CTE_MAESTRO_TRIB A
	FULL OUTER JOIN CTE_MAESTRO_CCPP  B
	ON A.REF_UE = B.REF_UE AND A.ID_OPORTUNIDAD = B.ID_OPORTUNIDAD

	UNION ALL
	--Ventas normales facturadas TRIBUTOS dadas de baja
	SELECT
		ID_OPORTUNIDAD,
		REF_UE,
		ESTADO_TRIBUTOS,
		NULL AS ESTADO_COMUNIDADES,
		FEC_POSICIONAMIENTO,
		DAY(FEC_POSICIONAMIENTO) AS DIA_POS,
		MONTH(FEC_POSICIONAMIENTO) AS MES_POS,
		YEAR(FEC_POSICIONAMIENTO) AS AÑO_POS,
		FEC_FACT_TRIB,
		DAY(FEC_FACT_TRIB) AS DIA_FACT_TRIB,
		MONTH(FEC_FACT_TRIB) AS MES_FACT_TRIB,
		YEAR(FEC_FACT_TRIB) AS AÑO_FACT_TRIB,
		NULL AS FEC_FACT_CCPP,
		NULL AS DIA_FACT_CCPP,
		NULL AS MES_FACT_CCPP,
		NULL AS AÑO_FACT_CCPP,
		VENTA_ESPECIAL AS VENTA_ESPECIAL_TRIB,
		POSTVENTA AS POSTVENTA_TRIB,
		ESTADO_TRIB AS OK_KO_TRIB,
		REVISAR_TRIB,
		NULL AS VENTA_ESPECIAL_CCPP,
		NULL AS POSTVENTA_CCPP,
		NULL AS OK_KO_CCPP,
		NULL AS REVISAR_CCPP,
		ESTADO
	FROM CALC.AAM_MAESTRO_TRIB_VENTAS
	WHERE VENTA_ESPECIAL='NO' AND (POSTVENTA='SI' OR POSTVENTA IS NULL) AND
ESTADO='BAJA'
		AND MONTH(FEC_FACT_TRIB)=MONTH(GETDATE()) AND
YEAR(FEC_FACT_TRIB)=YEAR(GETDATE())
```

```
	UNION ALL
	--Ventas especial y postventas TRIBUTOS facturadas en el mes
	SELECT
		ID_OPORTUNIDAD,
		REF_UE,
		ESTADO_TRIBUTOS,
		NULL AS ESTADO_COMUNIDADES,
		FEC_POSICIONAMIENTO,
		DAY(FEC_POSICIONAMIENTO) AS DIA_POS,
		MONTH(FEC_POSICIONAMIENTO) AS MES_POS,
		YEAR(FEC_POSICIONAMIENTO) AS AÑO_POS,
		FEC_FACT_TRIB,
		DAY(FEC_FACT_TRIB) AS DIA_FACT_TRIB,
		MONTH(FEC_FACT_TRIB) AS MES_FACT_TRIB,
		YEAR(FEC_FACT_TRIB) AS AÑO_FACT_TRIB,
		NULL AS FEC_FACT_CCPP,
		NULL AS DIA_FACT_CCPP,
		NULL AS MES_FACT_CCPP,
		NULL AS AÑO_FACT_CCPP,
		VENTA_ESPECIAL AS VENTA_ESPECIAL_TRIB,
		POSTVENTA AS POSTVENTA_TRIB,
		ESTADO_TRIB AS OK_KO_TRIB,
		REVISAR_TRIB,
		NULL AS VENTA_ESPECIAL_CCPP,
		NULL AS POSTVENTA_CCPP,
		NULL AS OK_KO_CCPP,
		NULL AS REVISAR_CCPP,
		ESTADO
	FROM CALC.AAM_MAESTRO_TRIB_VENTAS
	WHERE (VENTA_ESPECIAL='SI' OR POSTVENTA='SI, NO PRESENTE EN EL BI')
		AND MONTH(FEC_FACT_TRIB)=MONTH(GETDATE()) AND
YEAR(FEC_FACT_TRIB)=YEAR(GETDATE())

	UNION ALL
	--Ventas normales facturadas CCPP dadas de baja
	SELECT
		ID_OPORTUNIDAD,
		REF_UE,
		NULL AS ESTADO_TRIBUTOS,
		ESTADO_COMUNIDADES,
		FEC_POSICIONAMIENTO,
		DAY(FEC_POSICIONAMIENTO) AS DIA_POS,
		MONTH(FEC_POSICIONAMIENTO) AS MES_POS,
		YEAR(FEC_POSICIONAMIENTO) AS AÑO_POS,
		NULL AS FEC_FACT_TRIB,
		NULL AS DIA_FACT_TRIB,
		NULL AS MES_FACT_TRIB,
		NULL AS  AÑO_FACT_TRIB,
		FEC_FACT_CCPP,
		DAY(FEC_FACT_CCPP) AS DIA_FACT_CCPP,
		MONTH(FEC_FACT_CCPP)AS  MES_FACT_CCPP,
		YEAR(FEC_FACT_CCPP)AS AÑO_FACT_CCPP,
		NULL AS VENTA_ESPECIAL_TRIB,
		NULL AS POSTVENTA_TRIB,
		NULL AS OK_KO_TRIB,
		NULL AS REVISAR_TRIB,
```

```
                VENTA_ESPECIAL AS VENTA_ESPECIAL_CCPP,
                POSTVENTA AS POSTVENTA_CCPP,
                ESTADO_CCPP AS OK_KO_CCPP,
                REVISAR_CCPP,
                ESTADO
        FROM CALC.AAM_MAESTRO_CCPP_VENTAS
        WHERE VENTA_ESPECIAL='NO' AND (POSTVENTA='SI' OR POSTVENTA IS NULL) AND
ESTADO='BAJA'
                AND MONTH(FEC_FACT_CCPP)=MONTH(GETDATE()) AND
YEAR(FEC_FACT_CCPP)=YEAR(GETDATE())

        UNION ALL
        --Ventas especiales y postventas CCPP facturadas en el mes
        SELECT
                ID_OPORTUNIDAD,
                REF_UE,
                NULL AS ESTADO_TRIBUTOS,
                ESTADO_COMUNIDADES,
                FEC_POSICIONAMIENTO,
                DAY(FEC_POSICIONAMIENTO) AS DIA_POS,
                MONTH(FEC_POSICIONAMIENTO) AS MES_POS,
                YEAR(FEC_POSICIONAMIENTO) AS AÑO_POS,
                NULL AS FEC_FACT_TRIB,
                NULL AS DIA_FACT_TRIB,
                NULL AS MES_FACT_TRIB,
                NULL AS  AÑO_FACT_TRIB,
                FEC_FACT_CCPP,
                DAY(FEC_FACT_CCPP) AS DIA_FACT_CCPP,
                MONTH(FEC_FACT_CCPP)AS  MES_FACT_CCPP,
                YEAR(FEC_FACT_CCPP)AS AÑO_FACT_CCPP,
                NULL AS VENTA_ESPECIAL_TRIB,
                NULL AS POSTVENTA_TRIB,
                NULL AS OK_KO_TRIB,
                NULL AS REVISAR_TRIB,
                VENTA_ESPECIAL AS VENTA_ESPECIAL_CCPP,
                POSTVENTA AS POSTVENTA_CCPP,
                ESTADO_CCPP AS OK_KO_CCPP,
                REVISAR_CCPP,
                ESTADO
        FROM CALC.AAM_MAESTRO_CCPP_VENTAS
        WHERE (VENTA_ESPECIAL='SI' OR POSTVENTA='SI, NO PRESENTE EN EL BI')
                AND MONTH(FEC_FACT_CCPP)=MONTH(GETDATE()) AND
YEAR(FEC_FACT_CCPP)=YEAR(GETDATE())
END
```

ANEXO 2: Scripts de Python

Clases:

Carga_BI.py

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
```

```
from selenium.webdriver.chrome.options import Options
import time
import os
import shutil
from datetime import datetime
from eventLogger import EventLogger
import configparser
import sys

class extractorBI():

    # Configuración del navegador y URL
    def __init__(self,id_cliente,id_servicio,id_hito):
        self.id_cliente=id_cliente
        self.id_servicio=id_servicio
        self.id_hito=id_hito
        self.log=EventLogger()
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"EXTRACCION BI","1.INICIO EJECUCION","")
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"EXTRACCION BI","2.CONFIGURACION
DATOS","")
        config_obj = configparser.ConfigParser()
        config_obj.read("Config.ini")
        param_carga_bi = config_obj["carga_BI"]
        self.url = param_carga_bi["url"]
        self.driver_path=param_carga_bi["driver_path"]
        self.microsoft_login=param_carga_bi["microsoft_login_url"]
        self.powerbi_login = param_carga_bi["powerbi_login"]
        self.usuario=param_carga_bi["usuario"]
        self.contra=param_carga_bi["contra"]
        self.ruta_destino=param_carga_bi["ruta_destino"]
        self.ruta_descarga=param_carga_bi["ruta_descarga"]

    def archivo_descargado_completo(self,ruta):
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"EXTRACCION BI","14.1.COMPROBACION
DESCARGA FICHERO","")
        try:
            return os.path.exists(ruta) and os.path.getsize(ruta) > 0
        except Exception as e:
            self.log.writeLog("ERROR", self.id_cliente, self.id_servicio, self.id_hito,"EXTRACCION BI","14.1.COMPROBACION
DESCARGA FICHERO","ERROR EN LA COMPROBACION DE LA DESCARGA DEL FICHERO|ERROR:" + str(e)+"")

    # Inicializar el navegador
    def extraccion_bi(self):
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"EXTRACCION BI","3.INICIO
EXTRACCION","")
        try:
            self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"EXTRACCION BI","4.CONFIGURACION
DEL DRIVER","")
            chrome_options = Options()
            ancho_pantalla = 1280
            alto_pantalla = 720
            chrome_options.add_argument(f"--window-size={ancho_pantalla},{alto_pantalla}")
            #chrome_options.add_argument('--headless')
            chrome_options.add_argument('--disable-gpu')
            chrome_options.add_experimental_option("prefs", {
```

```
            #"download.default_directory": self.ruta_descarga,  # Ruta de descarga
            "download.prompt_for_download": False,
            "download.directory_upgrade": True,
            "safebrowsing_for_trusted_sources_enabled": False,
            "safebrowsing.enabled": False
        })
        driver = webdriver.Chrome(options=chrome_options, keep_alive=True)
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"EXTRACCION BI","5.APERTURA URL
BI","")
        driver.get(self.url)
        current_url = driver.current_url
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"EXTRACCION BI","6.LOGIN
ALTERNATIVO","")
        if self.microsoft_login in current_url:
            driver.implicitly_wait(25)
            # Introducir nombre de usuario
            username_field = driver.find_element(By.ID, "i0116")
            username_field.send_keys(self.usuario)
            username_field.send_keys(Keys.ENTER)
            # Introducir contraseña
            password_field = driver.find_element(By.ID, "i0118")
            password_field.send_keys(self.contra)
            login_button = WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.XPATH,
"//input[@type='submit' and @value='Iniciar sesión']")))
            login_button.click()
            #Mantener la sesion iniciada: no
            sesion_ini = WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.ID, "idBtn_Back")))
            sesion_ini.click()
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"7.EXTRACCION BI","LOGIN","")
        if  self.powerbi_login in current_url:
            driver.implicitly_wait(25)
            # Introducir nombre de usuario
            username_field = driver.find_element(By.ID, "email")
            username_field.send_keys(self.usuario)
            username_field.send_keys(Keys.ENTER)
            time.sleep(3)
            # Introducir contraseña
            password_field = driver.find_element(By.ID, "i0118")
            password_field.send_keys(self.contra)
            login_button = WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.ID, "idSIButton9")))
            login_button.click()
            #Mantener la sesion iniciada: no
            sesion_ini = WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.ID, "idBtn_Back")))
            sesion_ini.click()

        time.sleep(15)
        action = webdriver.ActionChains(driver=driver)
        # 1280x559 Pantalla ordenador
        # 1920x919 Pantalla grande VM
        action = webdriver.ActionChains(driver=driver)
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"EXTRACCION BI","8.PDTE ADMON","")
        action.move_by_offset(0.578906*ancho_pantalla, 0.2494812*alto_pantalla).perform()
        time.sleep(2)
        action.click().perform()
```

```
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"EXTRACCION BI","9.SELECCION TABLA","")
        action.move_by_offset(-0.57890*ancho_pantalla, -0.249481*alto_pantalla).perform()
        time.sleep(2)
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"EXTRACCION BI","10.SELECCION TABLA","")
        action.move_by_offset(0.4375*ancho_pantalla, 0.70787*alto_pantalla).perform()
        time.sleep(1)
        action.double_click().perform()
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"EXTRACCION BI","11.CLICK TRES PUNTOS TABLA","")
        action.move_by_offset(0.31640625*ancho_pantalla, -0.12838998*alto_pantalla).perform()
        time.sleep(2)
        action.click().perform()
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"EXTRACCION BI","12.CLICK EXPORTAR TABLA","")
        action.move_by_offset(0.08*ancho_pantalla, -0.20833631*alto_pantalla).perform()
        time.sleep(2)
        action.click().perform()
        time.sleep(2)
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"EXTRACCION BI","13.CLICK EXPORTAR","")
        action.move_by_offset(-0.2*ancho_pantalla, 0.37*alto_pantalla).perform()
        time.sleep(2)
        action.click().perform()
        # Fichero descargado
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"EXTRACCION BI","14.DESCARGA FICHERO","")
        ruta_descargas = os.path.expanduser(self.ruta_descarga)
        archivo = "data.xlsx"
        ruta_inicial = os.path.join(ruta_descargas, archivo)
        wait = WebDriverWait(driver, 40)  # espera hasta 60 segundos
        wait.until(lambda x: self.archivo_descargado_completo(ruta_inicial)if x else False)
        # Obtener la fecha actual en el formato YYYYMMDD
        fecha_actual = datetime.now().strftime("%Y%m%d")
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"EXTRACCION BI","15.RENOMBRADO FICHERO","")
        # Nuevo nombre del archivo
        nuevo_nombre = f"{fecha_actual}_Pte_AdmonPat_NFQ.xlsx"
        # Ruta de destino
        ruta_destino_completa = os.path.join(self.ruta_destino, nuevo_nombre)
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"EXTRACCION BI","16.MOVIMIENTO FICHERO A RUTA DESTINO","")
        # Cambiar el nombre del archivo y moverlo a la carpeta de destino
        shutil.move(ruta_inicial, ruta_destino_completa)
        driver.quit()
    except Exception as e:
        self.log.writeLog("ERROR", self.id_cliente, self.id_servicio, self.id_hito,"EXTRACCION BI","1.EXTRACCION","ERROR AL OBTENER FICHERO BI|ERROR:" + str(e)+"")
        driver.quit()
        sys.exit(1)
```

Carga.py

```
import pandas as pd
```

```
from sqlConnector import MS_DB
from sqlalchemy import text
from datetime import datetime
import os
import re
import shutil
from eventLogger import EventLogger
import configparser
import sys

class ejecutorCarga():

    def __init__(self, id_cliente, id_servicio, id_hito,id_tipoFichero) :
        self.log=EventLogger()
        self.ms_db=MS_DB()
        self.id_cliente=id_cliente
        self.id_servicio=id_servicio
        self.id_hito=id_hito
        self.id_tipoFichero=id_tipoFichero
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"CARGA STG","1.INICIO CLASE","")
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"CARGA STG","2.CONFIGURACION
DATOS","")
        config_obj = configparser.ConfigParser()
        config_obj.read("Config.ini")
        param_carga = config_obj["carga"]
        self.pathInputs = param_carga["rutaInputs"]

    def tratarFicheros(self,datos,tipologia_complementaria):
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"CARGA STG","6.TRATAMIENTO FILAS
FICHERO","")
        try:
            if tipologia_complementaria=='AAM_BI_VENTAS':
                datos = datos.iloc[:-3]
            return datos
        except Exception as e:
            self.log.writeLog("ERROR", self.id_cliente, self.id_servicio, self.id_hito,"CARGA STG","6.TRATAMIENTO FILAS
FICHERO","ERROR EN EL TRATAMIENTO DE LAS FILAS DEL FICHERO|ERROR:"+ str(e)+"")
            sys.exit()

    def obtenerFicheros(self):
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"CARGA STG","4.OBTENCION FICHEROS","")
        try:
            with self.ms_db.engine.begin() as conn:
                query=f"""SELECT A.NOMBRE, A.HEADER, A.HOJA, A.COLUMNAS, A.TIPOLOGIA_COMPLEMENTARIA,
A.RUTA FROM COD.FICHEROS A LEFT JOIN COD.CLIENTES B ON A.ID_CLIENTE=B.ID_CLIENTE LEFT JOIN
COD.SERVICIOS C ON A.ID_SERVICIO=C.ID_SERVICIO LEFT JOIN COD.HITOS D ON A.ID_HITO=D.ID_HITO
WHERE A.ID_TIPO_FICHERO=1 AND A.ID_CLIENTE='{self.id_cliente}' AND A.ID_SERVICIO='{self.id_servicio}' AND
A.ID_HITO='{self.id_hito}' AND A.ID_TIPO_FICHERO='{self.id_tipoFichero}' ORDER BY
A.TIPOLOGIA_COMPLEMENTARIA"""
                result=conn.execute(text(query)).fetchall()
                if result:
                    return result
                else:
                    self.log.writeLog("ERROR", self.id_cliente, self.id_servicio, self.id_hito,"CARGA STG","4.OBTENCION
FICHEROS","EL RESULTADO DE BUSQUEDA DE FICHEROS NO DEVUELVE REGISTROS")
```

```
                sys.exit()
        except Exception as e:
            self.log.writeLog("ERROR", self.id_cliente, self.id_servicio, self.id_hito,"CARGA STG","4.OBTENCION
FICHEROS","ERROR AL OBTENER FICHEROS|ERROR:"+ str(e)+"")
            sys.exit()

    def regularizarFechas(self, fecha_entrada):
        try:
            date_patterns = [
                r'(^\d{4})[-/.](\d{2})[-/.](\d{2}$)',  # YYYY[-/.]MM[-/.]DD
                r'(^\d{2})[-/.](\d{2})[-/.](\d{4}$)',  # MM[-/.]DD[-/.]YYYY O DD[-/.]MM[-/.]YYYY
            ]
            # Matcheo por patrones
            for pattern in date_patterns:
                match = re.search(pattern, fecha_entrada)
                if match:
                    groups = match.groups()
                    # Primer dia es Mes o Dia
                    if len(groups[0]) == 4:
                        year, month, day = groups
                    else:
                        if len(groups[1]) == 4:
                            day, year, month = groups
                        else:
                            first_number = int(groups[0])
                            second_number = int(groups[0])
                            if first_number <= 12 and second_number > 12:  # Si el primer numero es menor que 12 y el segundo mayor
que 12 entonces MM-DD-YYYY
                                month, day, year = groups
                            else:
                                day, month, year = groups

                    # Convertir a YYYY-MM-DD
                    transformed_date = f"{day.zfill(2)}/{month.zfill(2)}/{year}"
                    return transformed_date
                else:
                    # Si no se puede matchear nos quedamos con la de entrada
                    return fecha_entrada
        except Exception as e:
            self.log.writeLog("ERROR", self.id_cliente, self.id_servicio, self.id_hito,"CARGA STG","9.1.REGULARIZACION
FECHAS","ERROR AL REGULARIZAR FECHAS|ERROR:"+ str(e)+"")
            sys.exit()

    def obtener_nombres_columnas_sql(self, tableSTG):
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"CARGA STG","8.OBTENCION NOMBRES
COLUMNAS","")
        try:
            # Obtiene los nombres reales de las columnas de la tabla en SQL
            with self.ms_db.engine.begin() as conn:
                result = conn.execute(text(f"SELECT TOP 1 * FROM {tableSTG}"))
                nombres_columnas_sql = result.keys()
                return nombres_columnas_sql
        except Exception as e:
            self.log.writeLog("ERROR", self.id_cliente, self.id_servicio, self.id_hito,"CARGA STG","8.OBTENCION NOMBRES
COLUMNAS","ERROR AL OBTENER LOS NOMBRES DE LAS COLUMNAS|ERROR:"+ str(e)+"")
```

```
            sys.exit()

    def excelToDataframe(self, excel,hoja,header, columnas, doctype):
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"CARGA STG","5.EXCEL TO DATAFRAME","")
        # Crea el dataframe (las columnas checksum y fecha_fichero estarán vacías)
        df=None
        try:
            if doctype=='xlsx':
                df=pd.read_excel(excel, usecols=columnas, header=header, dtype ='str', engine='calamine',
sheet_name=hoja,index_col=False, na_values=["", "nan", "NaT", "--Sin Valor--"])
            elif doctype=='csv':
                df=pd.read_csv(excel, usecols=columnas, dtype ='str', sep=';', engine="python", header=header,index_col=False,
na_values=["", "nan", "NaT", "--Sin Valor--"], encoding = "ISO-8859-1",on_bad_lines='skip')
            elif doctype=='txt':
                with open(excel,'r') as file:
                    lines=file.readlines()
                    for index, line in enumerate(lines):
                        lines[index] = line.replace('\n','')
                    df=pd.DataFrame(lines)
            df = df.where(pd.notna(df), None)
            return df
        except Exception as e:
            self.log.writeLog("ERROR", self.id_cliente, self.id_servicio, self.id_hito,"CARGA STG","5.EXCEL TO
DATAFRAME","ERROR AL CONVERTIR FICHERO A DATAFRAME|ERROR:"+ str(e)+"")
            sys.exit()

    def borrado_previo(self, tableSTG,FEC_FICHERO):
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"CARGA STG","7.BORRADO PREVIO","")
        try:
            # truncar la tabla
            with self.ms_db.engine.begin() as conn:
                if self.id_cliente=='1' and self.id_servicio=='3' and self.id_hito=='16':
                    conn.execute(text(f"DELETE FROM " + tableSTG +" WHERE FEC_FICHERO <= CONVERT(DATE,'"
+FEC_FICHERO +"')"))
                else:
                    conn.execute(text(f"TRUNCATE TABLE " + tableSTG))
        except Exception as e:
            # Si la excepción indica que la tabla no existe, imprime un mensaje
            if "does not exist" in str(e):
                self.log.writeLog("ERROR", self.id_cliente, self.id_servicio, self.id_hito,"CARGA STG","7.BORRADO
PREVIO","LA TABLA " +str(tableSTG)+" NO EXISTE.")
                sys.exit()
            else:
                # Si es otra excepción, imprime el mensaje de error
                self.log.writeLog("ERROR", self.id_cliente, self.id_servicio, self.id_hito,"CARGA STG","7.BORRADO
PREVIO","NO SE HA PODIDO TRUNCAR LA TABLA " +str(tableSTG)+ "|ERROR:" +str(e)+"")
                sys.exit()

    def cargar(self,excel_path,ruta,archivos_encontrados,nombre,df, tipologia_complementaria, FEC_FICHERO,
nombres_columnas_sql):
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"CARGA STG","9.CARGAR","")
        try:
            # Rellenar la columna fecha_fichero
            df.insert(0,'FEC_FICHERO',FEC_FICHERO)
            if self.id_hito=='21':
```

```
            df.insert(2,'ID',None)
            df.insert(3,'NOMBRE_FIC',archivos_encontrados.split(".")[0])
            identificadores = []
            for i in range(len(df)):
                identificadores.append(i + 1)
            df.columns = nombres_columnas_sql
            df['ID'] = identificadores
        else:
            df.columns = nombres_columnas_sql
        #Convierte los valores nulos en None
        df.replace({pd.NaT: None}, inplace=True)
        df = df.where(pd.notna(df), None)
        #Convierte el tipo de valor a cadena (compatible con nvarchar) y los None se quedan así para que SQL los detecte como NULL
        df = df.map(lambda x: str(x) if pd.notna(x) else None)
        #Debido a que se obtienen todos los campos como strings y para que sea aplicable a todo tipo de tablas
        #se debe recorrer todos los valores del dataframe y transformar aquellos que son fechas
        if 'BAU' not in tipologia_complementaria and 'AAM_FACT' not in tipologia_complementaria:
            self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"CARGA STG","9.CARGAR|REGULARIZACION DE FECHAS","")
            for i in range(df.shape[0]):
                for j in range(df.shape[1]):
                    if df.iloc[i].iloc[j] is not None:
                        df.iloc[i].iloc[j] = self.regularizarFechas(df.iloc[i].iloc[j])

        batch_size = 1000
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"CARGA STG","9.CARGAR|INICIO CARGA","")
        j=0
        for i in range(0, len(df), batch_size):
            j=j+1
            batch = df.iloc[i:i + batch_size]
            self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"CARGA STG","9.CARGAR|BATCH"+str(j)+"","")
            batch.to_sql(tipologia_complementaria,schema= "STG", con=self.ms_db.engine.connect(), if_exists='append', index=False, chunksize=1000)
            self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"CARGA STG","9.CARGAR|BATCH"+str(j)+" FINALIZADO","")
        if nombre!= 'Control cartera BS' and nombre!='Control cartera TP':
            shutil.move(excel_path, os.path.join(self.pathInputs, ruta, 'OLD', archivos_encontrados))
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"CARGA STG","9.CARGAR|CARGA FINALIZADA","")
    except Exception as e:
        self.log.writeLog("ERROR", self.id_cliente, self.id_servicio, self.id_hito,"CARGA STG","9.CARGAR","EL FICHERO NO SE HA CARGADO CORRECTAMENTE|ERROR:"+ str(e)+"")
        sys.exit()

def ejecutor(self):
    #1.Log
    self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"CARGA STG","3.INICIO EJECUTOR","")

    #3.Obtención de ficheros asociados a parametros
    tip_complementaria=None
    ficheros=self.obtenerFicheros()
    if ficheros:
```

```
        for i in ficheros:
            nombre, header, hoja, columnas, tipologia_complementaria, ruta = i
            doctype = nombre.split('.')[-1]
            nombre = nombre.split('.')[0]
            excel_path=os.path.join(self.pathInputs, ruta)
            archivos = os.listdir(excel_path)
            archivos_encontrados = [file for file in archivos if nombre in file and '~$' not in file]
            self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"CARGA STG",f"CARGAR|CARGA
FICHERO: {str(nombre)} - {str(hoja)}","")
            if len(archivos_encontrados)>=1:
                for i in range(len(archivos_encontrados)):
                    excel_path = os.path.join(self.pathInputs, ruta, archivos_encontrados[i])
                    try:
                        dato = archivos_encontrados[0].split('_')[0]
                        FEC_FICHERO = datetime.strptime(dato, '%Y%m%d')
                        FEC_FICHERO = FEC_FICHERO.strftime('%Y%m%d')
                    except Exception as e:
                        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"CARGA STG","3.EJECUTOR","EL
FICHERO NO TIENE FECHA, SE INCLUYE LA FECHA DE HOY|ERROR:"+ str(e)+"")
                        FEC_FICHERO = datetime.now().strftime("%Y%m%d")

                    tableSTG="STG."+tipologia_complementaria.upper()
                    if doctype=='csv':
                        columnas = columnas.split(', ')

                    datos=self.excelToDataframe(excel_path,hoja,header,columnas,doctype)
                    #2.Elimino últimas filas del data frame (AAM_BI_VENTAS)
                    datos=self.tratarFicheros(datos,tipologia_complementaria)
                    #3.Borrado previo
                    if tip_complementaria!=tipologia_complementaria:
                        self.borrado_previo(tableSTG, FEC_FICHERO)
                        tip_complementaria=tipologia_complementaria
                    else:
                        pass
                    #4.Obtención columnas
                    nombres_columnas_sql = self.obtener_nombres_columnas_sql(tableSTG)
                    #5.Carga
                    self.cargar(excel_path,ruta,archivos_encontrados[i],nombre,datos, tipologia_complementaria, FEC_FICHERO,
nombres_columnas_sql)
        else:
                self.log.writeLog("ERROR",self.id_cliente, self.id_servicio, self.id_hito,"CARGA STG","3.EJECUTOR","NO SE
HAN ENCONTRADO FICHEROS A CARGAR")

        else:
            self.log.writeLog("ERROR",self.id_cliente, self.id_servicio, self.id_hito,"CARGA STG","3.EJECUTOR","ERROR AL
CARGAR EL FICHERO")
            sys.exit(1)
```

eventLogger.py

```
import pandas as pd
import pyodbc as sqlserver
import configparser
from datetime import datetime
```

```
class EventLogger():
    def __init__(self):
        config_obj = configparser.ConfigParser()
        config_obj.read("Config.ini")

        event_logger = config_obj["eventlogger"]
        dbparam = config_obj["sqlserver"]

        self.myuserid=dbparam["user"]
        self.mypassword=dbparam["password"]
        self.mydatabase=dbparam["db"]
        self.myserver=dbparam["host"]
        self.connectionString = f'DRIVER={{SQL
Server}};SERVER={self.myserver};DATABASE={self.mydatabase};UID={self.myuserid};PWD={self.mypassword}'

        self.log_table = event_logger["log_table"]

    #Utiliza la conexión propia del script GeneradorFicheros.py
    def writeLog(self, level, id_cliente, id_servicio, id_hito, proceso , mensaje, error):
        err=error.replace("'","")
        mensaje=mensaje.replace("'","")
        query = """INSERT INTO {} VALUES ('{}','{}','{}',{}, {}, {}, '{}', '{}', '{}')""".format(self.log_table,
datetime.now().strftime('%Y-%m-%d %H:%M:%S'), str(self.myuserid), str(level), id_cliente, id_servicio, id_hito, str(proceso),
str(mensaje), str(err))
        print(query)
        self.conn = sqlserver.connect(self.connectionString)
        self.cur = self.conn.cursor()
        self.cur.execute(query)
        self.cur.commit()
        self.cur.close()
        self.conn.close()
```

generadorFicheros.py

```
from sqlConnector import MS_DB
from eventLogger import EventLogger
import pandas as pd
from sqlalchemy import text,create_engine
import configparser
from shutil import copyfile
import os
from datetime import datetime
from openpyxl import load_workbook
from openpyxl.worksheet.datavalidation import DataValidation
from openpyxl.utils import get_column_letter
from openpyxl.styles import NamedStyle
from openpyxl.styles import named_styles


class generadorFicheros():
    def __init__(self,id_cliente,id_servicio,id_hito,id_tipo):
        #0.parametros
        self.id_cliente=id_cliente
        self.id_servicio=id_servicio
        self.id_hito=id_hito
        self.id_tipo=id_tipo
```

```
        #1.Log
        self.log=EventLogger()
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION FICHEROS","1.INICIO EJECUCION","")
        #2.Conexion
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION FICHEROS","2.CONEXION A BASE DE DATOS","")
        self.conexion=MS_DB()
        #3.Inicializacion variables
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION FICHEROS","3.CONFIGURACION DATOS","")
        config_obj = configparser.ConfigParser()
        config_obj.read("Config.ini")
        param_carga = config_obj["rutaRaizFicheros"]
        self.rutaMatriz = param_carga["rutaMatriz"]

    ################################################
    #   OBTENCION PARAMETROS FICHEROS A REPORTAR   #
    ################################################
    def obtencionParametrosFicheros(self):
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION FICHEROS","4.OBTENCION PARAMETROS FICHEROS","")
        try:
            with self.conexion.engine.begin() as conn:
                query=f"""SELECT B.CLIENTE,C.SERVICIO,A.NOMBRE,A.HOJA,A.ID_TIPO_FICHERO,A.HEADER,A.TIPOLOGIA_COMPLEMENTARIA,A.COLUMNAS_FECHA,A.COLUMNAS_IMP,A.RUTA,A.COLUMNAS_CONV_NUM FROM COD.FICHEROS A LEFT JOIN COD.CLIENTES B ON A.ID_CLIENTE=B.ID_CLIENTE LEFT JOIN COD.SERVICIOS C ON A.ID_SERVICIO=C.ID_SERVICIO WHERE APLICA=1 AND A.ID_CLIENTE= {self.id_cliente} AND A.ID_SERVICIO= {self.id_servicio} AND A.ID_HITO= {self.id_hito} AND ID_TIPO_FICHERO={self.id_tipo} ORDER BY A.NOMBRE"""
                result=conn.execute(text(query))
                df = pd.DataFrame(result.fetchall())
                return df
        except Exception as e:
            self.log.writeLog("ERROR", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION FICHEROS","4.OBTENCION PARAMETROS FICHEROS","ERROR AL OBTENER LOS PARAMETROS DE LOS FICHEROS|ERROR:"+ str(e) +"")
            return None

    ########################
    #   OBTENCION  RUTAS   #
    ########################
    def obtencionRutas(self,cliente,nmbFichero,ruta):
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION FICHEROS","5.OBTENCION PLANTILLA","")
        try:
            pathOrigenPlantilla=self.rutaMatriz+"Plantillas\\"+cliente+"\\"+nmbFichero
            pathDestinoPlantilla=self.rutaMatriz+"Outputs\\"+ruta+"\\"+datetime.today().strftime('%Y%m%d')+"_"+nmbFichero #Cambiar por la fecha del fichero a generar
            return [pathOrigenPlantilla,pathDestinoPlantilla]
        except Exception as e:
            self.log.writeLog("ERROR", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION FICHEROS","5.OBTENCION PLANTILLA","ERROR AL OBTENER LA PLANTILLA|ERROR:"+ str(e) +"")
            return None
```

```
    ##########################
    #   COPIADO  PLANTILLAS   #
    ##########################
    def copiadoPlantilla(self,rutaOrigenPlantilla,rutaDestinoPlantilla):
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION FICHEROS","6.COPIADO
PLANTILLA","")
        try:
            # Check if file already exists
            if os.path.exists(rutaDestinoPlantilla):
                self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION
FICHEROS","6.1.BORRADO PREVIO FICHERO","")
                try:
                    os.remove(rutaDestinoPlantilla)
                except Exception:
                    pass
            self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION FICHEROS","6.2.MOVER
PLANTILLA A RUTA DESTINO","")
            copyfile(rutaOrigenPlantilla, rutaDestinoPlantilla)
        except Exception as e:
            self.log.writeLog("ERROR", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION FICHEROS","6.COPIADO
PLANTILLA","ERROR AL COPIAR LA PLANTILLA|ERROR:"+ str(e) +"")
            return None

    ##########################
    #   FORMATEAR IMPORTES   #
    ##########################
    def formatearImportes(self,importes,df):
        try:
            if importes is not None:
                self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION FICHEROS","7.2.ASIGNAR
TIPO DE DATO IMPORTES SOBRE DF","")
                importes_columnas = importes.split(',')
                for i in importes_columnas:
                    df.iloc[:,int(i)-1]=pd.to_numeric(df.iloc[:,int(i)-1])
            else:
                self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION FICHEROS","7.2.NO ES
NECESARIO ASIGNAR TIPO DE DATO IMPORTES SOBRE DF","")

        except Exception as e:
            self.log.writeLog("ERROR", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION
FICHEROS","7.2.FORMATEAR IMPORTES","ERROR AL FORMATEAR IMPORTES EN EL FICHERO|ERROR:"+ str(e)
+"")

    ##########################
    #   COMPROBAR NUMEROS   #
    ##########################
    def es_numero(self,valor):
        try:
            float(valor)
            return True
        except (ValueError, TypeError):
            return False

    ##########################
    #   CONVERTIR A NUMERO   #
```

```
    ##########################
    def convertirNumero(self,numero,ws):
        try:
            if numero is not None:
                self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION
FICHEROS","7.6.CONVERTIR CELDAS A NUMERO","")
                col_numero=numero.split(',')
                for col in col_numero:
                    colf=get_column_letter(int(col))
                    for cell in ws[colf]:
                        if self.es_numero(cell.value)==True:
                            cell.value=float(cell.value)
            else:
                self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION FICHEROS","7.6.NO ES
NECESARIO CONVERTIR CELDAS A NUMERO","")
        except Exception as e:
            self.log.writeLog("ERROR", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION
FICHEROS","7.6.CONVERTIR CELDAS A NUMERO","ERROR AL CONVERTIR CELDAS A NUMERO|ERROR:"+
str(e) +"")

    ########################
    #   FORMATEAR FECHAS   #
    ########################
    def formatearFechas(self,fechas,df,num,ws, wb):
        try:
            #Definir tipo de dato sobre df
            if fechas is not None:
                if num=='1':
                    self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION
FICHEROS","7.1.ASIGNAR TIPO DE DATO FECHAS SOBRE DF","")
                    fechas_columnas = fechas.split(',')
                    for i in fechas_columnas:
                        df[df.columns[int(i)-1]]=pd.to_datetime(df[df.columns[int(i)-1]])
                elif num=='2':
                    self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION
FICHEROS","7.5.FORMATEAR FECHAS SOBRE EXCEL","")
                    if 'date_style' not in wb.named_styles:
                        date_style = NamedStyle(name='date_style', number_format='DD/MM/YYYY')
                    else:
                        index = wb.named_styles.index('date_style')
                        date_style = wb.named_styles[index]
                    fechas_columnas = fechas.split(',')
                    for col in fechas_columnas:
                        colf=get_column_letter(int(col))
                        for cell in ws[colf]:
                            if isinstance(cell.value, datetime):
                                cell.style = date_style
                else:
                    pass
            else:
                if num=='1':
                    self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION FICHEROS","7.1.NO ES
NECESARIO ASIGNAR TIPO DE DATO FECHAS SOBRE DF","")
                elif num=='2':
```

```
            self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION FICHEROS","7.5.NO ES NECESARIO FORMATEAR FECHAS SOBRE EXCEL","")
        else:
            pass
    except Exception as e:
        self.log.writeLog("ERROR", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION FICHEROS","7.1.FORMATEAR FECHAS","ERROR AL FORMATEAR LAS FECHAS EN EL FICHERO|ERROR:"+ str(e) +"")

  ########################
  #  CREAR VALIDACION   #
  ########################
  def create_validations(self,sheet_val, column):
    try:
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION FICHEROS","7.3.1.CREAR VALIDACION","")
        validations = ''
        for cell in sheet_val[column]:
            if cell.row>1:
                if cell.value is not None and cell.row==2:
                    validations="$"+get_column_letter(cell.column)+"$"+str(cell.row)+":"
                elif cell.value is None or (sheet_val[column][-1]==cell):
                    if sheet_val[column][-1]==cell:
                        validations=validations+"$"+get_column_letter(cell.column)+"$"+str((cell.row))
                    else:
                        validations=validations+"$"+get_column_letter(cell.column)+"$"+str((cell.row)-1)
                    break
        return validations
    except Exception as e:
        self.log.writeLog("ERROR", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION FICHEROS","7.3.1.CREAR VALIDACION","ERROR AL CREAR LA VALIDACION DE DATOS|ERROR:"+ str(e) +"")

  ########################
  #  VALIDACIÓN DATOS   #
  ########################
  def validaciondatos(self,RutaDestinoPlantilla,max_row,hojaFichero):
    try:
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION FICHEROS","7.3.VALIDACION DE DATOS","")
        book = load_workbook(RutaDestinoPlantilla)
        if 'Aux' not in book.sheetnames or hojaFichero=='Validaciones':
            self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION FICHEROS","7.3.NO ES NECESARIA LA VALIDACION DE DATOS","")
            return
        sheet_val = book['Aux']
        sheet_registros=book['Registros']
        i=0
        dv = []
        for column_val in sheet_val.iter_cols():
            if column_val[0].internal_value is not None:
                datavalidations = None
                datavalidations = self.create_validations(sheet_val, column_val[0].column_letter)
                dv.append(DataValidation(type="list", formula1="Aux!{}".format(datavalidations), allow_blank=True))
                sheet_registros.add_data_validation(dv[i])
                for column in sheet_registros.iter_cols():
```

```
                if column_val[0].internal_value==column[0].internal_value:
                    letter=column[0].column_letter
                    dv[i].add('{}2:{}{}'.format(letter, letter, max_row ))
                    break
                else:
                    pass
            i=i+1
        book.save(RutaDestinoPlantilla)
    except Exception as e:
        self.log.writeLog("ERROR", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION
FICHEROS","7.3.VALIDACION DE DATOS","ERROR AL INCLUIR LA VALIDACION DE DATOS|ERROR:"+ str(e) +"")


  ##################################
  #   INSERTAR DATOS EN FICHERO   #
  ##################################
  def insertarDatos(self,nmbTabla,hojaFichero,rutaDestinoPlantilla,fechas,importes,numero,header):
    nombre_fichero = rutaDestinoPlantilla.split('\\')[-1]
    mensaje = f"7.INSERCION DE DATOS EN FICHERO: {nombre_fichero} - HOJA: {hojaFichero}"
    self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito, "GENERACION FICHEROS", mensaje, "")
    try:
        #1.Obtención de datos de la tabla de reporte
        with self.conexion.engine.begin() as conn:
            if hojaFichero=="Validaciones":
                query=f"""SELECT VALOR_ID, GESTOR, TIPO_VAL, CAMPO, MENSAJE_ERROR, ORIGEN FROM
DQ.VALIDACIONES WHERE ID_CLIENTE={self.id_cliente} AND ID_SERVICIO={self.id_servicio} AND
ID_HITO={self.id_hito}"""
            else:
                query=f"""SELECT * FROM REP.{nmbTabla}"""
            result=conn.execute(text(query))
            #Indicar si la tabla está vacía
            if result.rowcount == 0:
                self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION FICHEROS","7.NO
EXISTEN DATOS, TABLA VACIA","")
                return
            df = pd.DataFrame(result.fetchall())
        #Asignar tipos de datos al df (fechas e importes)
        self.formatearFechas(fechas,df,'1',None,None)
        self.formatearImportes(importes,df)
        #Asignar desplegables en el Excel
        max_row=df.shape[0]+1
        self.validaciondatos(rutaDestinoPlantilla, max_row,hojaFichero)
        #5.Inserción de datos en fichero correspondiente
        with pd.ExcelWriter(rutaDestinoPlantilla,engine="openpyxl",mode='a',if_sheet_exists='overlay')  as writer:
            prim_fila=header+1
            df.to_excel(writer,sheet_name=hojaFichero, index=False,header=False, startrow=prim_fila,startcol=0)
            wb=load_workbook(rutaDestinoPlantilla)
            ws = writer.sheets[hojaFichero]
            self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION FICHEROS","7.4.INCLUIR
FILTROS EN ENCABEZADOS","")
            if self.id_servicio=="2" and self.id_hito=="3":
                end_col = ws.max_column
                end_col_letter = ws.cell(row=4, column=end_col).column_letter
                ws.auto_filter.ref = f"A4:{end_col_letter}4"
            else:
```

```
            ws.auto_filter.ref = ws.dimensions
            #Formatear Fechas en excel
            self.formatearFechas(fechas,df,'2', ws, wb)
            #Convertir a número en la hoja excel
            self.convertirNumero(numero,ws)
            load_workbook(rutaDestinoPlantilla).save
        except Exception as e:
            self.log.writeLog("ERROR", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION FICHEROS","7.INSERCION DATOS","ERROR AL INSERTAR DATOS EN EL REPORTE|ERROR:"+ str(e) +"")
            return None

    ############################
    #   GENERACION FICHEROS    #
    ############################
    def generacionFicheros(self):
        #1.Obtención parametros de los ficheros
        parametrosFicheros=self.obtencionParametrosFicheros()
        nmbFic=None
        for i in range(len(parametrosFicheros)):
            cliente=parametrosFicheros.iloc[i]['CLIENTE']
            servicio=parametrosFicheros.iloc[i]['SERVICIO']
            nmbFichero=parametrosFicheros.iloc[i]['NOMBRE']
            hojaFichero=parametrosFicheros.iloc[i]['HOJA']
            header=parametrosFicheros.iloc[i]['HEADER']
            nmbTabla=parametrosFicheros.iloc[i]['TIPOLOGIA_COMPLEMENTARIA']
            rutaDestino=parametrosFicheros.iloc[i]['RUTA']
            tipoFichero=parametrosFicheros.iloc[i]['ID_TIPO_FICHERO']
            fechas=parametrosFicheros.iloc[i]['COLUMNAS_FECHA']
            importes=parametrosFicheros.iloc[i]['COLUMNAS_IMP']
            numero=parametrosFicheros.iloc[i]['COLUMNAS_CONV_NUM']
            if nmbFic!=nmbFichero:
                #2.Obtención de rutas
                rutas=self.obtencionRutas(cliente,nmbFichero,rutaDestino)
                rutaOrigenPlantilla=rutas[0]
                rutaDestinoPlantilla=rutas[1]
                #3.Copiado de ficheros
                self.copiadoPlantilla(rutaOrigenPlantilla,rutaDestinoPlantilla)
                nmbFic=nmbFichero
            #5.Inserción de datos
            self.insertarDatos(nmbTabla,hojaFichero,rutaDestinoPlantilla,fechas,importes,numero,header)
            self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"GENERACION FICHEROS","98.GENERACION FINALIZADA","")
```

SenderEmail.py

```
from warnings import filterwarnings
filterwarnings('ignore')
from google.oauth2.credentials import Credentials
from googleapiclient.discovery import build
from google.auth.transport.requests import Request
from google_auth_oauthlib.flow import InstalledAppFlow
import base64
from eventLogger import EventLogger
import sys
import base64
import os
```

```
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.image import MIMEImage
from email.mime.audio import MIMEAudio
from email.mime.application import MIMEApplication
from email.mime.base import MIMEBase
import mimetypes

SCOPES = ["https://www.googleapis.com/auth/gmail.readonly","https://www.googleapis.com/auth/gmail.send"]

class Sender():
    def __init__(self, id_cliente,id_servicio, id_hito):
        #0.Variables clave
        self.id_cliente=id_cliente
        self.id_servicio=id_servicio
        self.id_hito=id_hito
        #1.Log
        self.log=EventLogger()
        #2.Obtencion credenciales
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"ENVIO MAIL
AUTOMATICO","1.INICIALIZACION SENDER","")
        creds = None
        if os.path.exists("token.json"):
            creds = Credentials.from_authorized_user_file("token.json", SCOPES)
        if not creds or not creds.valid:
            if creds and creds.expired and creds.refresh_token:
                creds.refresh(Request())
            else:
                flow = InstalledAppFlow.from_client_secrets_file("credentials.json", SCOPES)
                creds = flow.run_local_server(port=0)

            with open("token.json", "w") as token:
                token.write(creds.to_json())

        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"ENVIO MAIL AUTOMATICO","2.OBTENCION
CREDENCIALES","")
        creds = Credentials.from_authorized_user_file('token.json', SCOPES)
        #3.Inicializacion servicio
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"ENVIO MAIL
AUTOMATICO","3.INICIALIZACION SERVICIO","")
        self.service = build('gmail', 'v1', credentials=creds)

    def __addBody(self,body,ruta_imagen,ruta_imagen2):
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"ENVIO MAIL AUTOMATICO","6.ADICION
BODY","")
        try:
            if ruta_imagen is not None:
                image = MIMEImage(open(ruta_imagen, 'rb').read())
                image.add_header('Content-ID', '<image1>')
                if ruta_imagen2 is not None:
                    image.add_header('Content-Disposition', 'inline', filename="ResumenVentas.png")
                else:
                    image.add_header('Content-Disposition', 'inline', filename="FacturacionReporteAAM.png")
                self.msg.attach(image)
                if ruta_imagen2 is not None:
```

```
            image2 = MIMEImage(open(ruta_imagen2, 'rb').read())
            image2.add_header('Content-ID', '<image2>')
            image2.add_header('Content-Disposition', 'inline', filename="CambiosRegistros.png")
            self.msg.attach(image2)
            mensaje=MIMEText(body,'html')
            self.msg.attach(mensaje)
            return
         mensaje=MIMEText(body)
         self.msg.attach(mensaje)
      else:
         mensaje=MIMEText(body)
         self.msg.attach(mensaje)
   except Exception as e:
      self.log.writeLog("ERROR", self.id_cliente, self.id_servicio, self.id_hito,"ENVIO MAIL
AUTOMATICO","6.ADICION BODY","ERROR AL AÑADIR MENSAJE AL MAIL|ERROR:"+ str(e)+"")
      sys.exit()

 def __addAttachment(self,attachments):
   self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"ENVIO MAIL AUTOMATICO","7.ADICION
ATACCHMENT","")
   try:
      for attachment in attachments:
         content_type, encoding = mimetypes.guess_type(attachment)
         if content_type is None or encoding is not None:
            content_type = 'application/octet-stream'
         main_type, sub_type = content_type.split('/', 1)
         if main_type == 'text':
            fp = open(attachment, 'rb')
            message = MIMEText(fp.read().decode("utf-8"), _subtype=sub_type)
            fp.close()
         elif main_type == 'image':
            fp = open(attachment, 'rb')
            message = MIMEImage(fp.read(), _subtype=sub_type)
            fp.close()
         elif main_type == 'audio':
            fp = open(attachment, 'rb')
            message = MIMEAudio(fp.read(), _subtype=sub_type)
            fp.close()
         elif main_type=="application":
            fp = open(attachment, 'rb')
            message = MIMEApplication(fp.read(), _subtype=sub_type)
            fp.close()
         else:
            fp = open(attachment, 'rb')
            message = MIMEBase(fp.read(), _subtype=sub_type)
            fp.close()

         filename = os.path.basename(attachment)
         message.add_header('Content-Disposition', 'attachment', filename=filename)
         self.msg.attach(message)
   except Exception as e:
      self.log.writeLog("ERROR", self.id_cliente, self.id_servicio, self.id_hito,"ENVIO MAIL
AUTOMATICO","7.ADICION ATACCHMENT","ERROR AL AÑADIR OBJETO AL MAIL|ERROR:"+ str(e)+"")
      sys.exit()
```

```
    def sendMail(self,toperson, asunto,body,ruta_imagen,ruta_imagen2,attachments,cc):
        #4.Inicializacion mensaje
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"ENVIO MAIL
AUTOMATICO","4.INICIALIZACION MENSAJE","")
        self.msg= MIMEMultipart()
        #5.Variables del mensaje
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"ENVIO MAIL
AUTOMATICO","5.INICIALIZACION VARIABLES MENSAJE","")
        self.msg['To'] = toperson
        self.msg['CC']= cc
        self.msg['Subject'] = asunto
        #6.Obtencion mensaje
        self.__addBody(body,ruta_imagen,ruta_imagen2)
        #6.2.Firma
        with open('FirmaAngel.txt', 'r', encoding='utf-8') as file:
            firma_html = file.read()
        self.msg.attach(MIMEText(firma_html, 'html'))
        #7.Adicion archivos
        if attachments:
            self.__addAttachment(attachments)
        #7.Envio mail
        self.log.writeLog("INFO", self.id_cliente, self.id_servicio, self.id_hito,"ENVIO MAIL AUTOMATICO","8.ENVIO
MAIL","")
        try:
            encoded_message = base64.urlsafe_b64encode(self.msg.as_bytes()).decode()
            mensaje = {'raw': encoded_message}
            self.service.users().messages().send(userId="me", body=mensaje).execute()
        except Exception as e:
            self.log.writeLog("ERROR", self.id_cliente, self.id_servicio, self.id_hito,"ENVIO MAIL AUTOMATICO","3.5.ENVIO
MAIL","ERROR AL ENVIAR MAIL|ERROR:"+ str(e)+"")
            sys.exit()
```

sqlConnector.py

```
from contextlib import contextmanager
import configparser
import sys
import sqlalchemy
from sqlalchemy.sql import text
from urllib.parse import quote_plus
import pyodbc

class MS_DB():

    def __init__(self):
        # 1. Lectura de fichero de configuración
        config_obj = configparser.ConfigParser()
        config_obj.read("Config.ini")
        dbparam = config_obj["sqlserver"]
        # 2. Set de parametros de conexión
        driver = dbparam["driver"]
        user = dbparam["user"]
        password = dbparam["password"]
        host = dbparam["host"]
        dbase = dbparam["db"]
        self.driver = driver
```

```
        self.username = user
        self._password = password
        self.host = host
        self.db = dbase
        connectionString =
f'DRIVER={{{self.driver}}};SERVER={self.host};DATABASE={self.db};UID={self.username};PWD={self._password};Tru
sted_Connection=no;MARS_Connection=Yes'
        quoted = quote_plus(connectionString)
        new_con = f'mssql+pyodbc:///?odbc_connect={quoted}'
        self.engine = sqlalchemy.create_engine(new_con, use_setinputsizes=False)

    def __repr__(self):
        return f"MS-SQLServer('{self.username}', <password hidden>, '{self.host}', '{self.db}')"

    def __str__(self):
        return f"MS-SQLServer Module for STP on {self.host}"

    def __del__(self):
        self.engine.dispose()
        print("Connection closed.")

    @contextmanager
    def open_db_connection(self):
        connection = self.engine.connect()
        try:
            yield connection
        except sqlalchemy.exc.DatabaseError as err:
            sys.stderr.write(str(err))
            connection.rollback()
            raise err
        else:
            connection.commit()
        finally:
            connection.close()

    def execute_query(self, query):
        with self.open_db_connection() as connection:
            result = connection.execute(text(query))
            return result.fetchall()
```

Jobs:

Job_Carga_BI.py

```
from sqlConnector import MS_DB
from eventLogger import EventLogger
from Carga_BI import extractorBI
from Carga import ejecutorCarga
from sqlalchemy import text
import sys

############################
#   EJECUCION EXTRACCION   #
############################
def ejecutarExtaccion(id_cliente,id_servicio,id_hito,log):
    log.writeLog("INFO",id_cliente,id_servicio,id_hito,"CARGA BI","3.EJECUCION EXTRACCION","")
```

```
    try:
        extractor=extractorBI(id_cliente,id_servicio,id_hito)
        extractor.extraccion_bi()
    except Exception as e:
        log.writeLog("ERROR",id_cliente,id_servicio,id_hito,"CARGA BI","3.EJECUCION EXTRACCION","ERROR AL
EXTRAER FICHERO BI|ERROR:"+str(e)+"")
        sys.exit()

#######################
#   EJECUCION CARGA   #
#######################
def ejecutarCarga(id_cliente,id_servicio,id_hito,id_tipo_fichero_carga,log):
    log.writeLog("INFO",id_cliente,id_servicio,id_hito,"CARGA BI","4.EJECUCION CARGA","")
    try:
        cargador=ejecutorCarga(id_cliente,id_servicio,id_hito,id_tipo_fichero_carga)
        cargador.ejecutor()
    except Exception as e:
        log.writeLog("ERROR",id_cliente,id_servicio,id_hito,"CARGA BI","4.EJECUCION CARGA","ERROR AL CARGAR
FICHERO BI|ERROR:"+str(e)+"")
        sys.exit()

################################
#   EJECUCION HOMOGENEIZACION   #
################################
def ejecutarHomogeneizacion(conexion,id_cliente,id_servicio,id_hito,log):
    log.writeLog("INFO", id_cliente, id_servicio, id_hito,"CARGA BI","5.EJECUCION HOMOGENEIZACION","")
    try:
        with conexion.engine.begin() as conn:
            query=f"""SET NOCOUNT ON; EXEC HOM.HOMOGENEIZACION {id_cliente}, {id_servicio}, {id_hito} """
            conn.execute(text(query))
    except Exception as e:
        log.writeLog("ERROR", id_cliente, id_servicio, id_hito,"CARGA BI","5.EJECUCION HOMOGENEIZACION","ERROR
AL EJECUTAR LA HOMOGENEIZACION|ERROR:"+ str(e)+"")
        sys.exit()

if __name__=="__main__":
    #0.Parametros
    try:
        id_cliente=sys.argv[1]
        id_servicio=sys.argv[2]
        id_hito=sys.argv[3]
        id_tipo_fichero_carga=1
        #1.Log
        log=EventLogger()
        log.writeLog("INFO", id_cliente, id_servicio, id_hito,"CARGA BI","1.INICIO EJECUCION","")
        #2.Conexion
        log.writeLog("INFO", id_cliente, id_servicio, id_hito,"CARGA BI","2.CONEXION A BASE DE DATOS","")
        conexion=MS_DB()
        #3.Extraccion ficheros bi
        ejecutarExtaccion(id_cliente,id_servicio,id_hito,log)
        #4.Carga fichero BI
        ejecutarCarga(id_cliente,id_servicio,id_hito,id_tipo_fichero_carga,log)
        #5.Homogeneización fichero bi
        ejecutarHomogeneizacion(conexion,id_cliente,id_servicio,id_hito,log)
    except Exception:
```

```
    sys.exit()
```

Job_Carga_Operativos.py

```
from sqlConnector import MS_DB
from eventLogger import EventLogger
from Carga import ejecutorCarga
from generadorFicheros import generadorFicheros
from sqlalchemy import text
from datetime import datetime
import sys
import configparser
from SenderEmail import Sender
import glob
import os

######################
#   EJECUCION CARGA   #
######################
def ejecutarCarga(id_cliente,id_servicio,id_hito,id_tipo_fichero_carga,log):
   log.writeLog("INFO",id_cliente,id_servicio,id_hito,"CARGA OPERATIVOS","3.EJECUCION CARGA","")
   try:
      cargador=ejecutorCarga(id_cliente,id_servicio,id_hito,id_tipo_fichero_carga)
      cargador.ejecutor()
   except Exception as e:
      log.writeLog("ERROR",id_cliente,id_servicio,id_hito,"CARGA OPERATIVOS","3.EJECUCION CARGA","ERROR AL
CARGAR FICHEROS OPERATIVOS|ERROR:"+str(e)+"")
      sys.exit()

################################
#   EJECUCION HOMOGENEIZACION   #
################################
def ejecutarHomogeneizacion(conexion,id_cliente,id_servicio,id_hito,log):
   log.writeLog("INFO", id_cliente, id_servicio, id_hito,"CARGA OPERATIVOS","4.EJECUCION
HOMOGENEIZACION","")
   try:
      with conexion.engine.begin() as conn:
         query=f"""SET NOCOUNT ON; EXEC HOM.HOMOGENEIZACION {id_cliente}, {id_servicio}, {id_hito} """
         conn.execute(text(query))
   except Exception as e:
      log.writeLog("ERROR", id_cliente, id_servicio, id_hito,"CARGA OPERATIVOS","4.EJECUCION
HOMOGENEIZACION","ERROR AL EJECUTAR LA HOMOGENEIZACION|ERROR:"+ str(e)+"")
      sys.exit()

#########################
#   EJECUCION VALIDACION   #
#########################
def ejecutarValidaciones(conexion,id_cliente,id_servicio,id_hito,log):

   try:
      if id_hito == '1':
         log.writeLog("INFO", id_cliente, id_servicio, id_hito,"CARGA OPERATIVOS","5.EJECUCION
VALIDACIONES","")
         with conexion.engine.begin() as conn:
            query=f"""SET NOCOUNT ON; EXEC DQ.ORQUESTADOR_VALIDACIONES {id_cliente}, {id_servicio},
{id_hito}"""
```

```
            conn.execute(text(query))
        else:
            log.writeLog("INFO", id_cliente, id_servicio, id_hito,"CARGA OPERATIVOS","5.NO EXISTEN VALIDACIONES PARA EL FICHERO EN CARGA","")
            pass
    except Exception as e:
        log.writeLog("ERROR", id_cliente, id_servicio, id_hito,"CARGA OPERATIVOS","5.EJECUCION VALIDACIONES","ERROR AL EJECUTAR LAS VALIDACIONES|ERROR:"+ str(e)+"")
        sys.exit()

##########################
#   EJECUCION REPORTING   #
##########################
def ejecutarReporting(conexion,id_cliente, id_servicio, id_hito,id_tipo_fichero_rep,log):
    log.writeLog("INFO", id_cliente, id_servicio, id_hito,"CARGA OPERATIVOS","6.EJECUCION REPORTING","")
    try:
        count=0
        with conexion.engine.begin() as conn:
            if id_tipo_fichero_rep==4 and id_hito=='1':
                log.writeLog("INFO", id_cliente, id_servicio, id_hito,"CARGA OPERATIVOS","6.1.REPORTE VALIDACIONES","")
                query=f"""SET NOCOUNT ON; EXEC DQ.REP_VALIDACIONES {id_cliente}, {id_servicio}, {id_hito}"""
                conn.execute(text(query))
                log.writeLog("INFO", id_cliente, id_servicio, id_hito,"CARGA OPERATIVOS","6.2.COMPROBACION VALIDACIONES NFORCE","")
                query = f"""SELECT COUNT(*) FROM DQ.VALIDACIONES WHERE ID_CLIENTE={id_cliente} AND ID_SERVICIO={id_servicio} AND ID_HITO={id_hito} AND ORIGEN = 'NFORCE'"""
                result = conn.execute(text(query)).fetchone()
                count = result[0]
            if count > 0:
                log.writeLog("INFO", id_cliente, id_servicio, id_hito,"CARGA OPERATIVOS","6.3.EXISTEN VALIDACIONES DE NFORCE. EXTRACION REPORTE VENTAS OPERATIVO","")
                query=f"""SET NOCOUNT ON; EXEC REP.GENERACION_REP_VENTAS_OPE {id_cliente}, {id_servicio}, {id_hito}, 0"""
                conn.execute(text(query))
            else:
                log.writeLog("INFO", id_cliente, id_servicio, id_hito,"CARGA OPERATIVOS","98.NO EXISTEN VALIDACIONES DE NFORCE, NO ES NECESARIO EL REPORTING. FIN CARGA","")
                pass
        return count
    except Exception as e:
        log.writeLog("ERROR", id_cliente, id_servicio, id_hito,"CARGA OPERATIVOS","6.EJECUCION REPORTING","ERROR AL EJECUTAR EL REPORTING|ERROR:"+ str(e)+"")
        sys.exit()

########################
#   EJECUCION VOLCADO   #
########################
def ejecutarVolcado(id_cliente,id_servicio,id_hito,id_tipo_fichero_rep,log,count):
    try:
        if count>0:
            log.writeLog("INFO", id_cliente, id_servicio, id_hito,"CARGA OPERATIVOS","7.EJECUCION VOLCADO","")
            ejecutorFicheros=generadorFicheros(id_cliente,id_servicio,id_hito,id_tipo_fichero_rep)
            ejecutorFicheros.generacionFicheros()
        else:
```

```
            pass
    except Exception as e:
        log.writeLog("ERROR", id_cliente, id_servicio, id_hito,"CARGA OPERATIVOS","7.EJECUCION
VOLCADO","ERROR AL GENERAR LOS REPORTES|ERROR:"+str(e)+"")
        sys.exit()

##############################
#   EXTRACCION PARAMETROS    #
##############################
def __obtencionParametros(conexion,id_cliente,id_servicio,id_hito,id_tipo_fichero_rep,log):
    log.writeLog("INFO", id_cliente, id_servicio, id_hito,"REPORTING","8.1.OBTENCION PARAMETROS","")
    try:
        with conexion.engine.begin() as conn:
            query=f"""SELECT RUTA FROM COD.FICHEROS WHERE ID_CLIENTE= {id_cliente} AND ID_SERVICIO=
{id_servicio} AND ID_HITO= {id_hito} AND ID_TIPO_FICHERO={id_tipo_fichero_rep}"""
            result=conn.execute(text(query)).fetchall()
            for i in result:
                ruta=i[0]
                return ruta
            else:
                return None
    except Exception as e:
        log.writeLog("ERROR", id_cliente, id_servicio, id_hito,"REPORTING","5.1.OBTENCION PARAMETROS","ERROR
AL OBTENER LOS PARAMETROS PARA EL REPORTING|ERROR:"+str(e)+"")
        sys.exit()

###############################
#   OBTENCION RUTA FICHERO    #
###############################
def __getFile(ruta):
    log.writeLog("INFO", id_cliente, id_servicio, id_hito,"REPORTING","8.2.OBTENCION FICHERO","")
    try:
        config_obj = configparser.ConfigParser()
        config_obj.read("Config.ini")
        param_carga = config_obj["paths"]
        ruta_outputs = param_carga["out_path"]
        ruta=ruta_outputs+ruta

        list_of_files=os.listdir(ruta)
        #list_of_files = glob.glob(ruta)
        for i in list_of_files:
            max_time=0
            if os.path.getctime(os.path.join(ruta,i))>max_time and '.ini' not in i and "Maestro" not in i:
                max_time=os.path.getctime(os.path.join(ruta,i))
                ruta_fichero=[os.path.join(ruta,i)]
        return ruta_fichero
    except Exception as e:
        log.writeLog("ERROR", id_cliente, id_servicio, id_hito,"REPORTING","5.2.OBTENCION FICHERO","ERROR AL
OBTENER EL FICHERO DE SALIDA|ERROR:"+str(e)+"")
        sys.exit()

###################
#   ENVIO MAIL    #
###################
def envioMail(conexion,id_cliente,id_servicio,id_hito,id_tipo_fichero_rep, count):
```

```
    try:
        if count>0:
            log.writeLog("INFO", id_cliente, id_servicio, id_hito,"REPORTING","8.ENVIO MAIL","")
            #1.Obtencion ruta fichero
            ruta=__obtencionParametros(conexion,id_cliente,id_servicio,id_hito,id_tipo_fichero_rep,log)
            #2.Obtencion fichero de ruta
            fichero=__getFile(ruta)
            #3.Envio mail
            if id_cliente=='1' and id_hito=="1":
                if id_servicio=='1':
                    toperson='mercedes.casellas@nforceservices.es; eva.quintana@nforceservices.es'
                    cc='angel.guzman@nfq.es'
                    #toperson='angel.guzman@nfq.es'
                    #cc=''
                elif id_servicio=='2':
                    toperson='melody.sweet@nforceservices.es; ana.sanz@nforceservices.es'
                    cc='angel.guzman@nfq.es'
                    #toperson='angel.guzman@nfq.es'
                    #cc=''
                if id_tipo_fichero_rep==4:
                    asunto="Ficheros Operativos Validados"
                    body="Compartimos ficheros operativos para revisión.\n\nPor favor, corregid validaciones de NFORCE y volved a
dejar el fichero en la ruta inputs.\n\nMuchas gracias.\n"
                imagen=None
                imagen2=None
                sender=Sender(id_cliente,id_servicio,id_hito)
                sender.sendMail(toperson,asunto,body,imagen,imagen2, fichero,cc)
        else:
            if id_cliente=='1' and id_hito=="1":
                if id_servicio=='1':
                    toperson='mercedes.casellas@nforceservices.es; eva.quintana@nforceservices.es'
                    cc='angel.guzman@nfq.es'
                    #toperson='angel.guzman@nfq.es'
                    #cc=''
                elif id_servicio=='2':
                    toperson='melody.sweet@nforceservices.es; ana.sanz@nforceservices.es'
                    cc='angel.guzman@nfq.es'
                    #toperson='angel.guzman@nfq.es'
                    #cc=''
                if id_tipo_fichero_rep==4:
                    asunto="Ficheros Operativos Validados"
                    body="No existen validaciones, fichero cargado correctamente."
                imagen=None
                imagen2=None
                fichero=None
                sender=Sender(id_cliente,id_servicio,id_hito)
                sender.sendMail(toperson,asunto,body,imagen,imagen2, fichero,cc)
    except Exception as e:
        log.writeLog("ERROR", id_cliente, id_servicio, id_hito,"REPORTING","8.ENVIO MAIL","ERROR AL ENVIAR EL
MAIL|ERROR:"+str(e)+"")
        sys.exit()

def ejecutarReporteMaestro(conexion, id_cliente,id_servicio,id_hito,id_tipo_fichero_rep,log, count):
    try:
        if count>0:
```

```
        log.writeLog("INFO", id_cliente, id_servicio, id_hito,"REPORTING","9.EJECUCION REPORTING CON MAESTRO (RESET PARA VOLVER A EJECUTAR LAS VALIDACIONES AL CARGARLO DE NUEVO)","")
        with conexion.engine.begin() as conn:
            if id_tipo_fichero_rep==4 and id_hito=='1':
                log.writeLog("INFO", id_cliente, id_servicio, id_hito,"REPORTING","9.1.RESETEO TABLA REP","")
                query=f"""SET NOCOUNT ON; EXEC REP.GENERACION_REP_VENTAS_OPE {id_cliente}, {id_servicio}, {id_hito}, 1"""
                conn.execute(text(query))
            else:
                log.writeLog("INFO", id_cliente, id_servicio, id_hito,"REPORTING","98.NO ES NECESARIO RESETEAR LA TABLA REPORTING. FIN CARGA","")
                pass
        return
    except Exception as e:
        log.writeLog("ERROR", id_cliente, id_servicio, id_hito,"REPORTING","9.EJECUCION REPORTING CON MAESTRO","ERROR AL EJECUTAR EL REPORTING|ERROR:"+ str(e)+"")
        sys.exit()

if __name__=="__main__":
    #0.Parametros
    id_cliente=sys.argv[1]
    id_servicio=sys.argv[2]
    id_hito=sys.argv[3]
    id_tipo_fichero_carga=1
    id_tipo_fichero_rep=4
    #1.Log
    log=EventLogger()
    log.writeLog("INFO", id_cliente, id_servicio, id_hito,"CARGA OPERATIVOS","1.INICIO EJECUCION","")
    #2.Conexion
    log.writeLog("INFO", id_cliente, id_servicio, id_hito,"CARGA OPERATIVOS","2.CONEXION A BASE DE DATOS","")
    conexion=MS_DB()
    #3.Carga ficheros operativos
    ejecutarCarga(id_cliente,id_servicio,id_hito,id_tipo_fichero_carga,log)
    #4.Homogeneización ficheros operativos
    ejecutarHomogeneizacion(conexion,id_cliente,id_servicio,id_hito,log)
    #5.Validación ficheros operativos
    ejecutarValidaciones(conexion,id_cliente,id_servicio,id_hito,log)
    #6.Reporting Operativos
    count=ejecutarReporting(conexion,id_cliente,id_servicio,id_hito,id_tipo_fichero_rep,log)
    #7.Volcado a ficheros
    ejecutarVolcado(id_cliente,id_servicio,id_hito,id_tipo_fichero_rep,log,count)
    #8.Envío correo validaciones
    envioMail(conexion,id_cliente,id_servicio,id_hito,id_tipo_fichero_rep,count)
    #9.Resetear reportes para volver a ejecutar validaciones
    ejecutarReporteMaestro(conexion, id_cliente,id_servicio,id_hito,id_tipo_fichero_rep,log, count)
```

Job_Historificacion.py

```
from sqlConnector import MS_DB
from eventLogger import EventLogger
from sqlalchemy import text
import sys


##################################
#   EJECUCION HISTORIFICACIÓN   #
```

```
################################
def ejecutarHistorificacion(conexion,id_cliente,id_servicio,id_hito,log):
    log.writeLog("INFO", id_cliente, id_servicio, id_hito,"HISTORIFICACION","3.EJECUCION HISTORIFICACION","")
    try:
        with conexion.engine.begin() as conn:
            query=f"""SET NOCOUNT ON; EXEC HIS.HISTORIFICACION {id_cliente}, {id_servicio}, {id_hito}"""
            conn.execute(text(query))
    except Exception as e:
        log.writeLog("ERROR", id_cliente, id_servicio, id_hito,"HISTORIFICACION","3.EJECUCION
HISTORIFICACION","ERROR AL EJECUTAR LA HISTORIFICACION|ERROR:"+ str(e)+"")
        sys.exit()

if __name__=="__main__":
    #0.Parametros
    id_cliente=sys.argv[1]
    id_servicio=sys.argv[2]
    id_hito=sys.argv[3]
    #1.Log
    log=EventLogger()
    log.writeLog("INFO", id_cliente, id_servicio, id_hito,"HISTORIFICACION","1.INICIO EJECUCION","")
    #2.Conexion
    log.writeLog("INFO", id_cliente, id_servicio, id_hito,"HISTORIFICACION","2.CONEXION A BASE DE DATOS","")
    conexion=MS_DB()
    #3.Historificación
    ejecutarHistorificacion(conexion,id_cliente,id_servicio,id_hito,log)
```

Job_Maestro.py

```
from sqlConnector import MS_DB
from eventLogger import EventLogger
from sqlalchemy import text
import sys

#########################
#   EJECUCION MAESTRO   #
#########################
def ejecutarMaestro(conexion,id_cliente, id_servicio, id_hito,fch_datos,log):
    log.writeLog("INFO", id_cliente, id_servicio, id_hito,"GENERACION MAESTRO","3.EJECUCION MAESTRO","")
    try:
        with conexion.engine.begin() as conn:
            if id_hito=='1':
                query=f"""SET NOCOUNT ON; EXEC CALC.AAM_MAESTRO_VENTAS
{id_cliente},{id_servicio},{id_hito},'{fch_datos}' """
                conn.execute(text(query))
            elif id_hito=='11':
                query=f"""SET NOCOUNT ON; EXEC CALC.AAM_GENERAR_MAESTRO_FACTURAS
{id_cliente},{id_servicio},{id_hito},'{fch_datos}' """
                conn.execute(text(query))
            elif id_hito=='5':
                query=f"""SET NOCOUNT ON; EXEC CALC.AAM_GENERAR_MAESTRO_CALENDARIO_TRIBUTARIO
{id_cliente},{id_servicio},{id_hito},'{fch_datos}' """
                conn.execute(text(query))
    except Exception as e:
        log.writeLog("ERROR", id_cliente, id_servicio, id_hito,"GENERACION MAESTRO","3.EJECUCION
MAESTRO","ERROR AL EJECUTAR EL MAESTRO|ERROR:"+ str(e)+"")
        sys.exit()
```

```
if __name__=="__main__":
    #0.Parametros
    id_cliente=sys.argv[1]
    id_servicio=sys.argv[2]
    id_hito=sys.argv[3]
    fch_datos=sys.argv[4]
    #1.Log
    log=EventLogger()
    log.writeLog("INFO", id_cliente, id_servicio, id_hito,"GENERACION MAESTRO","1.INICIO EJECUCION","")
    #2.Conexion
    log.writeLog("INFO", id_cliente, id_servicio, id_hito,"GENERACION MAESTRO","2.CONEXION A BASE DE
DATOS","")
    conexion=MS_DB()
    #3.Generacion Maestro
    ejecutarMaestro(conexion,id_cliente, id_servicio, id_hito, fch_datos, log)
```

Job_Reporting.py

```
from sqlConnector import MS_DB
from eventLogger import EventLogger
from generadorFicheros import generadorFicheros
from sqlalchemy import text
import sys
from SenderEmail import Sender
import glob
import os
import configparser
import openpyxl
from PIL import Image
import win32com.client
import time
import pyautogui
import pygetwindow as gw
from openpyxl import load_workbook
import xlwings as xw
import pdf2image
from datetime import datetime

############################
#   EJECUCION REPORTING    #
############################
def ejecutarReporting(conexion,id_cliente, id_servicio, id_hito,id_tipo_fichero_rep,log):
    log.writeLog("INFO", id_cliente, id_servicio, id_hito,"REPORTING","3.EJECUCION REPORTING","")
    try:
        with conexion.engine.begin() as conn:
            if id_tipo_fichero_rep==4 and id_hito=='1':
                log.writeLog("INFO", id_cliente, id_servicio, id_hito,"REPORTING","3.1.REPORTE VENTAS OPERATIVO","")
                query=f"""SET NOCOUNT ON; EXEC REP.GENERACION_REP_VENTAS_OPE {id_cliente}, {id_servicio},
{id_hito}, 1"""
                conn.execute(text(query))
                log.writeLog("INFO", id_cliente, id_servicio, id_hito,"REPORTING","3.2.REPORTE VALIDACIONES","")
                query=f"""SET NOCOUNT ON; EXEC DQ.REP_VALIDACIONES {id_cliente}, {id_servicio}, {id_hito}"""
                conn.execute(text(query))
            if id_tipo_fichero_rep==2 and id_hito=='1':
```

```
            log.writeLog("INFO", id_cliente, id_servicio, id_hito,"REPORTING","3.1.REPORTE VENTAS CLIENTE","")
            query=f"""SET NOCOUNT ON; EXEC REP.GENERACION_REP_VENTAS_AAM {id_cliente}, {id_servicio},
{id_hito}"""
            conn.execute(text(query))
            log.writeLog("INFO", id_cliente, id_servicio, id_hito,"REPORTING","3.2.REPORTE TOTAL VENTAS
CLIENTE","")
            query=f"""SET NOCOUNT ON; EXEC REP.GENERACION_REP_TOTAL_VENTAS_AAM {id_cliente},
{id_servicio}, {id_hito}"""
            conn.execute(text(query))
         if id_tipo_fichero_rep==4 and id_hito=='11':
            log.writeLog("INFO", id_cliente, id_servicio, id_hito,"REPORTING","3.1.REPORTE FACTURAS","")
            query=f"""SET NOCOUNT ON; EXEC REP.GENERACION_REP_AAM_FACTURAS {id_cliente}, {id_servicio},
{id_hito}"""
            conn.execute(text(query))
         else:
            pass
   except Exception as e:
      log.writeLog("ERROR", id_cliente, id_servicio, id_hito,"REPORTING","3.EJECUCION REPORTING","ERROR AL
EJECUTAR EL REPORTING|ERROR:"+ str(e)+"")
      sys.exit()

#########################
#  EJECUCION VOLCADO  #
#########################
def ejecutarVolcado(id_cliente,id_servicio,id_hito,id_tipo_fichero_rep,log):
   log.writeLog("INFO", id_cliente, id_servicio, id_hito,"REPORTING","4.EJECUCION VOLCADO","")
   try:
      ejecutorFicheros=generadorFicheros(id_cliente,id_servicio,id_hito,id_tipo_fichero_rep)
      ejecutorFicheros.generacionFicheros()
   except Exception as e:
      log.writeLog("ERROR", id_cliente, id_servicio, id_hito,"REPORTING","4.EJECUCION VOLCADO","ERROR AL
GENERAR LOS REPORTES|ERROR:"+str(e)+"")
      sys.exit()

############################
#  EXTRACCION PARAMETROS  #
############################
def __obtencionParametros(conexion,id_cliente,id_servicio,id_hito,id_tipo_fichero_rep,log):
   log.writeLog("INFO", id_cliente, id_servicio, id_hito,"REPORTING","5.1.OBTENCION PARAMETROS","")
   try:
      with conexion.engine.begin() as conn:
         query=f"""SELECT RUTA FROM COD.FICHEROS WHERE ID_CLIENTE= {id_cliente} AND ID_SERVICIO=
{id_servicio} AND ID_HITO= {id_hito} AND ID_TIPO_FICHERO={id_tipo_fichero_rep}"""
         result=conn.execute(text(query)).fetchall()
         for i in result:
            ruta=i[0]
            return ruta
         else:
            return None
   except Exception as e:
      log.writeLog("ERROR", id_cliente, id_servicio, id_hito,"REPORTING","5.1.OBTENCION PARAMETROS","ERROR
AL OBTENER LOS PARAMETROS PARA EL REPORTING|ERROR:"+str(e)+"")
      sys.exit()

############################
```

```
#   OBTENCION RUTA FICHERO    #
################################
def __getFile(ruta):
    log.writeLog("INFO", id_cliente, id_servicio, id_hito,"REPORTING","5.2.OBTENCION FICHERO","")
    try:
        config_obj = configparser.ConfigParser()
        config_obj.read("Config.ini")
        param_carga = config_obj["paths"]
        ruta_outputs = param_carga["out_path"]
        ruta=ruta_outputs+ruta
        list_of_files = glob.glob(os.path.join(ruta, "*"))
        #Se necesitan 2 ficheros en el reporte de cliente (el de BI normal, Resumen ventas)
        list_of_files = [file for file in list_of_files if "Total" not in os.path.basename(file) and "Resumen" not in
os.path.basename(file) and "Facturacion" not in  os.path.basename(file) and "Cambios" not in  os.path.basename(file) and "$"
not in  os.path.basename(file) and "ini" not in  os.path.basename(file)]
        list_of_files.sort(key=os.path.getctime, reverse=True)
        latest_files = list_of_files[:2]
        return latest_files
    except Exception as e:
        log.writeLog("ERROR", id_cliente, id_servicio, id_hito,"REPORTING","5.2.OBTENCION FICHERO","ERROR AL
OBTENER EL FICHERO DE SALIDA|ERROR:"+str(e)+"")
        sys.exit()

############################
#   OBTENCION CAPTURA    #
############################
def capturaExcel(ficheros, ruta):
    log.writeLog("INFO", id_cliente, id_servicio, id_hito,"REPORTING","5.3.OBTENCION CAPTURA","")
    try:
        config_obj = configparser.ConfigParser()
        config_obj.read("Config.ini")
        param_carga = config_obj["paths"]
        ruta_outputs = param_carga["out_path"]
        ruta=ruta_outputs+ruta
        rutapdf=ruta+"\\ResumenVentas.pdf"
        rutaimg=ruta+"\\ResumenVentas.jpg"
        rutapdf2=ruta+"\\CambiosRegistros.pdf"
        rutaimg2=ruta+"\\CambiosRegistros.jpg"
        if os.path.exists(rutaimg):
            os.remove(rutaimg)
        if os.path.exists(rutapdf):
            os.remove(rutapdf)
        if os.path.exists(rutaimg2):
            os.remove(rutaimg2)
        if os.path.exists(rutapdf2):
            os.remove(rutapdf2)
        archivo_excel = next((archivo for archivo in ficheros if 'Gráficas' in archivo), None)
        if archivo_excel:
            #Guardar Resumen Ventas como pdf
            app = xw.App(visible=False)
            libro = app.books.open(archivo_excel)
            hoja = libro.sheets["Resumen Ventas"]
            hoja.api.PageSetup.PrintArea = "A1:K36"
            hoja.api.PageSetup.Zoom = False
            hoja.api.PageSetup.FitToPagesWide = 1
```

```
            hoja.api.PageSetup.FitToPagesTall = 1
            hoja.api.ExportAsFixedFormat(0, rutapdf)
            hoja2 = libro.sheets["Cambios Registros"]
            for cell in hoja2.range("C2:C9"):
                if isinstance(cell.value, datetime):
                    cell.number_format = 'DD/MM/YYYY'
                pass
            for cell in hoja2.range('F3:G11'):
                if isinstance(cell.value, (int, float)) and '%' in cell.number_format:
                    valor_porcentaje = cell.value * 100
                    cell.value = f"{valor_porcentaje:.2f}%".replace('.', ',')
                pass
            hoja2.api.PageSetup.PrintArea = "B2:G13"
            hoja2.api.PageSetup.Zoom = False
            hoja2.api.PageSetup.FitToPagesWide = 1
            hoja2.api.PageSetup.FitToPagesTall = 1
            hoja2.api.ExportAsFixedFormat(0, rutapdf2)
            libro.close()
            app.quit()
            #Pasar pdf a imagen
            imagenes = pdf2image.convert_from_path(rutapdf, 300)  # 300 DPI
            imagen=imagenes[0]
            ancho, alto = imagen.size
            imagenes2 = pdf2image.convert_from_path(rutapdf2, 300)  # 300 DPI
            imagen2=imagenes2[0]
            ancho2, alto2 = imagen2.size
            #Recortar imagen
            imagen_recortada = imagen.crop((150, 150, ancho-150, alto-1700))
            imagen_recortada.save(rutaimg, "JPEG")  # Guardar como JPG
            imagen_recortada2 = imagen2.crop((150, 150, ancho2-180, alto2-2490))
            imagen_recortada2.save(rutaimg2, "JPEG")  # Guardar como JPG
        else:
            pass
        return rutaimg, rutaimg2
    except Exception as e:
        log.writeLog("ERROR", id_cliente, id_servicio, id_hito,"REPORTING","5.3.OBTENCION FICHERO","ERROR AL
OBTENER LA CAPTURA|ERROR:"+str(e)+"")
        sys.exit()

###################
#   ENVIO MAIL    #
###################
def envioMail(conexion,id_cliente,id_servicio,id_hito,id_tipo_fichero_rep):
    try:
        if id_cliente=="1":
            log.writeLog("INFO", id_cliente, id_servicio, id_hito,"REPORTING","5.ENVIO MAIL","")
            if id_servicio=="7" and id_hito=="1":
                #1.Obtencion ruta fichero
                ruta=__obtencionParametros(conexion,id_cliente,id_servicio,id_hito,id_tipo_fichero_rep,log)
                #2.Obtencion fichero de ruta
                ficheros=__getFile(ruta)
                #3.Hacer Captura
                imagen,imagen2=capturaExcel(ficheros, ruta)
                toperson='ma.decaceres.gil@altamiraam.com; j.matarranz.bellido@dovalue.es; lg.marin.campos@altamiraam.com;
jairo.jimenez@nfq.es; melody.sweet@nforceservices.es; eva.quintana@nforceservices.es'
```

```
            cc='ana.sanz@nforceservices.es; mercedes.casellas@nforceservices.es; francisco.bernabe@nfq.es;
pedro.romero@nfq.es; emilio.parrilla@nforceservices.es; angel.guzman@nfq.es; marta.rodrigo@nfq.es'
            #4.Envio mail
            #toperson='angel.guzman@nfq.es'
            #cc=''
            if id_tipo_fichero_rep==2:
               asunto="Ficheros Seguimiento de Ventas/Postventas"
               body="""<html lang="es">
                     <head>
                        <meta charset="UTF-8">
                     </head>
                     <body>
                        <p>Buenos días,</p>
                        <p>Adjunto gráfica seguimiento de ventas:</p>
                        <img src='cid:image1' width='900' height='665'>
                        <p>Adjunto gráfica del acumulado de cambios en los registros en mes actual:</p>
                        <img src='cid:image2' width='1000' height='330'>
                     </body>
                     </html>"""
               sender=Sender(id_cliente,id_servicio,id_hito)
               sender.sendMail(toperson,asunto,body, imagen,imagen2, ficheros,cc)
         else:
            log.writeLog("INFO", id_cliente, id_servicio, id_hito,"REPORTING","5.NO ES NECESARIO ENVIAR MAIL DE
REPORTE","")
            pass
      else:
         pass
   except Exception as e:
      log.writeLog("ERROR", id_cliente, id_servicio, id_hito,"REPORTING","5.ENVIO MAIL","ERROR AL ENVIAR EL
MAIL|ERROR:"+str(e)+"")
      sys.exit()

if __name__=="__main__":
   #0.Parametros
   id_cliente=sys.argv[1]
   id_servicio=sys.argv[2]
   id_hito=sys.argv[3]
   if id_servicio=="7":
      id_tipo_fichero_rep=2
   else:
      id_tipo_fichero_rep=4
   #1.Log
   log=EventLogger()
   log.writeLog("INFO", id_cliente, id_servicio, id_hito,"REPORTING","1.INICIO EJECUCION","")
   #2.Conexion
   log.writeLog("INFO", id_cliente, id_servicio, id_hito,"REPORTING","2.CONEXION A BASE DE DATOS","")
   conexion=MS_DB()
   #3.Reporting Operativos
   ejecutarReporting(conexion,id_cliente,id_servicio,id_hito,id_tipo_fichero_rep,log)
   #4.Volcado a ficheros
   ejecutarVolcado(id_cliente,id_servicio,id_hito,id_tipo_fichero_rep,log)
   #5.Envio mail a responsables
   envioMail(conexion,id_cliente,id_servicio,id_hito,id_tipo_fichero_rep)
```

Job_BI_Nforce.py

```
from sqlConnector import MS_DB
from eventLogger import EventLogger
from sqlalchemy import text
import sys
from SenderEmail import Sender
import pandas as pd
from datetime import datetime, timedelta
import glob
import os
import configparser
import xlwings as xw
import pdf2image

#################################
#   EJECUCION REPORTES VENTAS   #
#################################
def ejecutarqueryBI(conexion,fch_datos,log):
    log.writeLog("INFO", '1', '7', '1',"GENERACION INFORMES VENTAS","3.EJECUCION SP PARA EL BI NFORCE","")
    try:
        with conexion.engine.begin() as conn:
            query=f"""SET NOCOUNT ON; EXEC REP.GENERAR_REPORTES_VENTAS '{fch_datos}' """
            conn.execute(text(query))
    except Exception as e:
        log.writeLog("ERROR", '1', '7', '1',"GENERACION INFORMES VENTAS","3.EJECUCION
REP.GENERAR_REPORTES_VENTAS","ERROR AL EJECUTAR REPORTES VENTAS|ERROR:"+ str(e)+"")
        sys.exit()

#####################
#   ENVIAR CORREO   #
#####################

def obtener_resultado(query, conn):
    if "SELECT" in query:
        result = conn.execute(text(query))
        df = pd.DataFrame(result.fetchall(), columns=result.keys())
        if not df.empty:
            return str(df.iloc[0, 0])
    else:
        return str(query)

###############################
#   OBTENCION RUTA FICHERO    #
###############################
def __getFile(ruta):
    log.writeLog("INFO", '1', '7', '1',"REPORTING","5.2.OBTENCION FICHERO","")
    try:
        config_obj = configparser.ConfigParser()
        config_obj.read("Config.ini")
        param_carga = config_obj["paths"]
        ruta_outputs = param_carga["out_path"]
        ruta=ruta_outputs+ruta
        list_of_files = glob.glob(os.path.join(ruta, "*"))
        #Se necesitan 2 ficheros en el reporte de cliente (el de BI normal, Resumen ventas)
        list_of_files = [file for file in list_of_files if "MesActual" in os.path.basename(file)]
        list_of_files.sort(key=os.path.getctime, reverse=True)
```

```
        latest_files = list_of_files[:1]
        return latest_files
    except Exception as e:
        log.writeLog("ERROR", '1', '7', '1',"REPORTING","5.2.OBTENCION FICHERO","ERROR AL OBTENER EL
FICHERO DE SALIDA|ERROR:"+str(e)+"")
        sys.exit()

#########################
#   OBTENCION CAPTURA    #
#########################
def capturaExcel(ficheros, ruta):
    log.writeLog("INFO",'1', '7', '1',"REPORTING","5.3.OBTENCION CAPTURA","")
    try:
        config_obj = configparser.ConfigParser()
        config_obj.read("Config.ini")
        param_carga = config_obj["paths"]
        ruta_outputs = param_carga["out_path"]
        ruta=ruta_outputs+ruta
        rutapdf=ruta+"\\FacturacionReporteAAM.pdf"
        rutaimg=ruta+"\\FacturacionReporteAAM.jpg"
        if os.path.exists(rutaimg):
            os.remove(rutaimg)
        if os.path.exists(rutapdf):
            os.remove(rutapdf)
        archivo_excel = next((archivo for archivo in ficheros if 'MesActual' in archivo), None)
        if archivo_excel:
            #Guardar como pdf
            app = xw.App(visible=False)
            libro = app.books.open(archivo_excel)
            hoja = libro.sheets["Gráficas"]
            hoja.api.PageSetup.PrintArea = "B3:E16"
            hoja.api.PageSetup.Zoom = False
            hoja.api.PageSetup.FitToPagesWide = 1
            hoja.api.PageSetup.FitToPagesTall = 1
            hoja.api.ExportAsFixedFormat(0, rutapdf)
            libro.close()
            app.quit()
            #Pasar pdf a imagen
            imagenes = pdf2image.convert_from_path(rutapdf, 300)  # 300 DPI
            imagen=imagenes[0]
            ancho, alto = imagen.size
            #Recortar imagen
            imagen_recortada = imagen.crop((0, 0, ancho, alto-1700))
            imagen_recortada.save(rutaimg, "JPEG")  # Guardar como JPG
        else:
            pass
        return rutaimg
    except Exception as e:
        log.writeLog("ERROR", '1', '7', '1',"REPORTING","5.3.OBTENCION CAPTURA","ERROR AL OBTENER LA
CAPTURA|ERROR:"+str(e)+"")
        sys.exit()

def enviarcorreo(conexion, log, ruta):
    log.writeLog("INFO", '1', '7', '1',"ENVIAR CORREO INFORME NFORCE","1.INICIO: DEFINIR VARIABLES","")
    try:
```

```
    asunto="Informe Diario Ventas/Postventas"
    toperson='marta.rodrigo@nfq.es; mercedes.casellas@nforceservices.es; eva.quintana@nforceservices.es;
melody.sweet@nforceservices.es; ana.sanz@nforceservices.es'
    #toperson='angel.guzman@nfq.es'
    cc='angel.guzman@nfq.es; jairo.jimenez@nfq.es'
    #cc='angel.guzman@nfq.es'
    ficheros=__getFile(ruta)
    imagen=capturaExcel(ficheros, ruta)
    imagen2=None
    id_cliente="1"
    id_servicio="7"
    id_hito="1"
    messages = []
    querys= []
    fch_datos = datetime.now().date()
    with conexion.engine.begin() as conn:
      log.writeLog("INFO", '1', '7', '1',"ENVIAR CORREO INFORME NFORCE","2.COMPROBACION CUADRE
VENTAS","")
      result = conn.execute(text("""SELECT COUNT(*) AS conteo FROM REP.OPE_VENTAS_TRIB WHERE
VENTA_ESPECIAL='NO' AND (POSTVENTA IS NULL OR POSTVENTA='SI')"""))
      conteo3 = result.scalar()
      result = conn.execute(text("""SELECT COUNT(*) AS conteo FROM REP.OPE_VENTAS_CCPP WHERE
VENTA_ESPECIAL='NO' AND (POSTVENTA IS NULL OR POSTVENTA='SI')"""))
      conteo4 = result.scalar()
      result = conn.execute(text("""SELECT COUNT(*) AS conteo FROM REP.CLI_VENTAS"""))
      conteo5 = result.scalar()
      result = conn.execute(text("""SELECT COUNT(*) AS conteo FROM HOM.AAM_BI_VENTAS"""))
      conteo6 = result.scalar()
      if conteo3 == conteo4 and conteo4 == conteo5 and conteo5 == conteo6:
        log.writeLog("INFO", '1', '7', '1',"ENVIAR CORREO INFORME NFORCE","2.1.COMPROBACION CUADRE
VENTAS: OK","")
        messages.append(str(f"Fecha del informe: {fch_datos}"))
        messages.append(f"Nº de ventas en BI: {conteo3}")
      else:
        log.writeLog("INFO", '1', '7', '1',"ENVIAR CORREO INFORME NFORCE","2.2.COMPROBACION CUADRE
VENTAS: KO","")
        messages.append(str(f"Fecha del informe: {fch_datos}"))
        messages.append("Hay discrepancias en los conteos. Avisar a Ángel Guzmán para que revise la base de datos,
gracias.")
        messages.append(f"Operativo Tributos (SOLO BI): {conteo3}")
        messages.append(f"Operativo CCPP (SOLO BI): {conteo4}")
        messages.append(f"Reporte cliente: {conteo5}")
        messages.append(f"Descarga BI: {conteo6}")
        body= "Buenos días,\n\nAdjunto Informe diario de ventas:\n"
        body = body + "\n".join(messages)
        body=body+"\n\n"
        sender=Sender(id_cliente,id_servicio,id_hito)
        sender.sendMail(toperson,asunto,body, imagen,imagen2, ficheros,cc)
        return
      log.writeLog("INFO", '1', '7', '1',"ENVIAR CORREO INFORME NFORCE","2.EJECUTAR QUERYS SQL
(CONSTRUIR BODY)","")
      querys.append("""SELECT 'Altas BI: ' + CAST(COUNT(*) AS VARCHAR(10)) AS resultado FROM
HIS.AAM_BI_VENTAS WHERE ESTADO IN ('ALTA', 'REALTA') AND FEC_FICHERO = CAST(GETDATE() AS
DATE)""")
```

```
        querys.append("""SELECT 'Bajas BI: ' + CAST(COUNT(*) AS VARCHAR(10)) AS resultado FROM
HIS.AAM_BI_VENTAS WHERE ESTADO IN ('BAJA') AND FEC_FICHERO = CAST(GETDATE() AS DATE)""")
        querys.append("""SELECT 'Operativo de Tributos: - Ventas Totales: '+CAST(COUNT(*) AS VARCHAR(10)) AS
conteo FROM REP.OPE_VENTAS_TRIB""")
        querys.append("""SELECT '      - Postventas fuera de fichero: '+ CAST(COUNT(*) AS VARCHAR(10)) AS conteo
FROM REP.OPE_VENTAS_TRIB WHERE POSTVENTA='SI, NO PRESENTE EN EL BI'""")
        querys.append("""SELECT '      - Ventas fuera de fichero: '+CAST(COUNT(*) AS VARCHAR(10)) AS conteo FROM
REP.OPE_VENTAS_TRIB WHERE VENTA_ESPECIAL='SI'""")
        querys.append("""SELECT '      - Activos a Revisar: '+CAST(COUNT(*) AS VARCHAR(10)) AS conteo FROM
REP.OPE_VENTAS_TRIB WHERE REVISAR_TRIB IN ('REVISAR','REVISAR, HA PASADO DE OK A KO')""")
        querys.append("""SELECT 'Operativo de CCPP: - Ventas Totales: '+CAST(COUNT(*) AS VARCHAR(10)) AS conteo
FROM REP.OPE_VENTAS_CCPP""")
        querys.append("""SELECT '      - Postventas fuera de fichero: '+ CAST(COUNT(*) AS VARCHAR(10)) AS conteo
FROM REP.OPE_VENTAS_CCPP WHERE POSTVENTA='SI, NO PRESENTE EN EL BI'""")
        querys.append("""SELECT '      - Ventas fuera de fichero: '+CAST(COUNT(*) AS VARCHAR(10)) AS conteo FROM
REP.OPE_VENTAS_CCPP WHERE VENTA_ESPECIAL='SI'""")
        querys.append("""SELECT '      - Activos a Revisar: '+CAST(COUNT(*) AS VARCHAR(10)) AS conteo FROM
REP.OPE_VENTAS_CCPP WHERE REVISAR_CCPP IN ('REVISAR','REVISAR, HA PASADO DE OK A KO')""")
        querys.append("""SELECT 'Facturación total mes actual Tributos: '+CAST(COUNT(*) AS VARCHAR(10)) AS conteo
FROM REP.AAM_FACTURACION_VENTAS WHERE ID_SERVICIO=1 AND
MONTH(FEC_FACT)=MONTH(GETDATE()) AND YEAR(FEC_FACT)=YEAR(GETDATE())""")
        querys.append("SELECT '       - Facturadas día anterior: ' + CAST(COUNT(*) AS VARCHAR(10)) AS conteo FROM
REP.AAM_FACTURACION_VENTAS WHERE ID_SERVICIO = 1 AND FEC_FACT = CASE WHEN
DATEPART(weekday, GETDATE()) = 2 THEN CAST(DATEADD(day, -3, GETDATE()) AS DATE) ELSE
CAST(DATEADD(day, -1, GETDATE()) AS DATE) END")
        querys.append("""SELECT 'Facturación total mes actual CCPP: '+CAST(COUNT(*) AS VARCHAR(10)) AS conteo
FROM REP.AAM_FACTURACION_VENTAS WHERE ID_SERVICIO=2 AND
MONTH(FEC_FACT)=MONTH(GETDATE()) AND YEAR(FEC_FACT)=YEAR(GETDATE())""")
        querys.append("SELECT '       - Facturadas día anterior: ' + CAST(COUNT(*) AS VARCHAR(10)) AS conteo FROM
REP.AAM_FACTURACION_VENTAS WHERE ID_SERVICIO = 2 AND FEC_FACT = CASE WHEN
DATEPART(weekday, GETDATE()) = 2 THEN CAST(DATEADD(day, -3, GETDATE()) AS DATE) ELSE
CAST(DATEADD(day, -1, GETDATE()) AS DATE) END")
        for query in querys:
            resultado = obtener_resultado(query, conn)
            messages.append(resultado)
        body= "Buenos días,\n\nAdjunto Informe diario de ventas:\n\n"
        body = body + "\n".join(messages)+"\n"
        body = body + "\nAdjunto fichero con la facturación del mes en curso\n"
        sender=Sender(id_cliente,id_servicio,id_hito)
        sender.sendMail(toperson,asunto,body, imagen,imagen2, ficheros,cc)
    except Exception as e:
        log.writeLog("ERROR", '1', '7', '1',"ENVIAR CORREO INFORME NFORCE","1.EJECUTAR","ERROR AL MANDAR
CORREO|ERROR:"+ str(e)+"")
        sys.exit()

if __name__=="__main__":
    #0.Parametros
    fch_datos=sys.argv[1]
    #1.Log
    log=EventLogger()
    log.writeLog("INFO", '1', '7', '1',"GENERACION INFORMES VENTAS","1.INICIO EJECUCION","")
    #2.Conexion
    log.writeLog("INFO", '1', '7', '1',"GENERACION INFORMES VENTAS","2.CONEXION A BASE DE DATOS","")
    conexion=MS_DB()
    #3.Generación informes ventas
```

```
    ejecutarqueryBI(conexion, fch_datos,log)
    ruta="AAM\Tributos - CCPP\Ventas"
    enviarcorreo(conexion,log,ruta)
```

Scheduler.py

```
from warnings import filterwarnings
filterwarnings('ignore')
import schedule
import time
import locale
import os
import datetime
import sys



locale.setlocale(locale.LC_TIME, 'es_ES.UTF-8')

def CargaBI():
    ejecucion=os.system(rf" python job_Carga_BI.py 1 7 1")
    if ejecucion==1:
        raise Exception

def CargaOperativos_Ventas():
    ejecucion1=os.system(rf" python job_Carga_Operativos.py 1 1 1")
    ejecucion2=os.system(rf" python job_Carga_Operativos.py 1 2 1")
    if ejecucion1 == 1 or ejecucion2 == 1:
        raise Exception
def Hist_Maestro_Reporting_Ventas():
    os.system("job_Historificacion 1 1 1")
    os.system("job_Historificacion 1 2 1")
    os.system("job_Historificacion 1 7 1")
    fecha_actual = datetime.datetime.now()
    fecha_formateada = fecha_actual.strftime("%Y%m%d")
    os.system(f"job_Maestro 1 1 1 {fecha_formateada}")
    os.system(f"job_Maestro 1 2 1 {fecha_formateada}")
    os.system(rf" python job_Reporting.py 1 1 1")
    os.system(rf" python job_Reporting.py 1 2 1")
    os.system(rf" python job_Reporting.py 1 7 1")
    os.system(f" python job_BI_Nforce.py {fecha_formateada}")

def Reporting_Facturas():
```

```
    fecha_actual = datetime.datetime.now()
    fecha_formateada = fecha_actual.strftime("%Y%m%d")
    os.system(f"job_Maestro 1 3 11 {fecha_formateada}")
    os.system(rf" python job_Reporting.py 1 3 11")

def Carga_Hist_Maestro_Reporting_Calendario():
    os.system(rf" python job_Carga_Operativos.py 1 1 5")
    os.system("job_Historificacion 1 1 5")
    fecha_actual = datetime.datetime.now()
    fecha_formateada = fecha_actual.strftime("%Y%m%d")
    os.system(f"job_Maestro 1 1 5 {fecha_formateada}")
    os.system(rf" python job_Reporting.py 1 1 5")

schedule.every().day.at("05:45").do(CargaBI)
schedule.every().day.at("05:55").do(Hist_Maestro_Reporting_Ventas)
#schedule.every().day.at("13:31").do(Carga_Hist_Maestro_Reporting_Calendario)
#schedule.every().day.at("20:54").do(CargaOperativos_Ventas)
#schedule.every().day.at("16:29").do(Reporting_Facturas)

festivos = {
    datetime.date(2024, 1, 1),
    datetime.date(2024, 3, 28),
    datetime.date(2024, 3, 29),
    datetime.date(2024, 5, 1),
    datetime.date(2024, 5, 2),
    datetime.date(2024, 5, 15),
    datetime.date(2024, 7, 25),
    datetime.date(2024, 8, 15),
    datetime.date(2024, 11, 1),
    datetime.date(2024, 12, 6),
    datetime.date(2024, 12, 25)
}

def es_dia_laboral(fecha):
    # Retorna True si no es fin de semana ni festivo
    return fecha.weekday() < 5 and fecha.date() not in festivos

while True:
    try:
        if es_dia_laboral(datetime.datetime.now()):
            schedule.run_pending()
        time.sleep(1)
    except Exception as e:
```

```
break
```