Operación Baas y Open Finance

**Documento Manual Técnico**

***Nombre de Desarrollo:*** *Herramienta de cruces de la conciliación Bpay*

|  |  |  |  |
| --- | --- | --- | --- |
| **Control cambios** | | | |
| **Versión** | **Fecha** | **Observación** | **Responsable** |
| **1.0** | **03/02/2025** | **Versión inicial** | **Juan Pablo Jaramillo** |

Contenido

[1. Descripción 2](#_Toc168992042)

[2. Introducción 3](#_Toc168992043)

[3. Alcance 3](#_Toc168992044)

[4. Proceso 4](#_Toc168992045)

[5. Ciclos ACH 5](#_Toc168992046)

[6. Pasos de la rutina 6](#_Toc168992047)

[7. Componentes de la rutina 7](#_Toc168992048)

# Descripción

|  |  |
| --- | --- |
| Código del proceso en la cadena de valor |  |
| Nombre del proceso |  |
| Nombre del Desarrollo | modelo de propensión de pago |
| Código de Intervención del Asistente en Procesos |  |
| Objetivo del asistente en procesos |  |
| Vicepresidencia | Vicepresidencia de Ecosistemas |
| Dirección |  |
| Gerencia dueña del proceso | Operación Ecosistemas |
| Nombre del Gerente dueño del proceso | gabriel andres gallego castro |
| Nombre del jefe de sección dueño del proceso | Juan David Rivera |
| Nombre analista(s) configurador(es) | Juan Pablo Jaramillo Castrillón |

# Introducción

Bancolombia cuenta con un gran portafolio de productos de crédito para los diferentes segmentos de clientes; Estos son otorgados mes a mes con ciertas obligaciones o compromisos bancarios, con cuotas que por lo general son mensuales. La gran mayoría de los clientes que adquieren una obligación, cumplen con sus promesas de pago, sin embargo, hay un porcentaje de clientes que mes a mes no lo hacen, y se catalogan como "obligaciones morosas".

La gestión de esa cartera que entra en mora se empieza a realizar desde el día 1 de mora, bajo diferentes estrategias. Una de ellas, es ofrecerles a los clientes ciertas opciones de pago, para que su altura de mora o días de mora no se incrementen, y por lo tanto, no sea una cartera que termine en procesos de judicialización o castigo, que implicaría amplios gastos de gestión, y hasta perdida del capital expuesto al cliente.

Las opciones de pago son un conjunto de diferentes estrategias que ayudan a aliviar la carga financiera que los clientes en mora están viviendo. Algunos ejemplos de opciones de pago son:

* Ampliaciones de plazo de la deuda
* Reducciones de cuota de la deuda
* Renegociación de tasa de interés
* Reestructuraciones de crédito

Ahora, esas opciones de pago se manejan como un producto preaprobado, y no todos los clientes tienen acceso a ellas, e incluso solo se les es preaprobado como máximo 3 opciones de pago, por obligación del cliente, por mes. Las aplicaciones de estas opciones de pago son limitadas, es decir, una vez a la obligación se le haya aplicado una opción de pago, esta debe esperar por lo menos de 3 a 4 meses para poder aplicar a otra opción de pago (este tiempo varía dependiendo de la opción de pago aplicada).

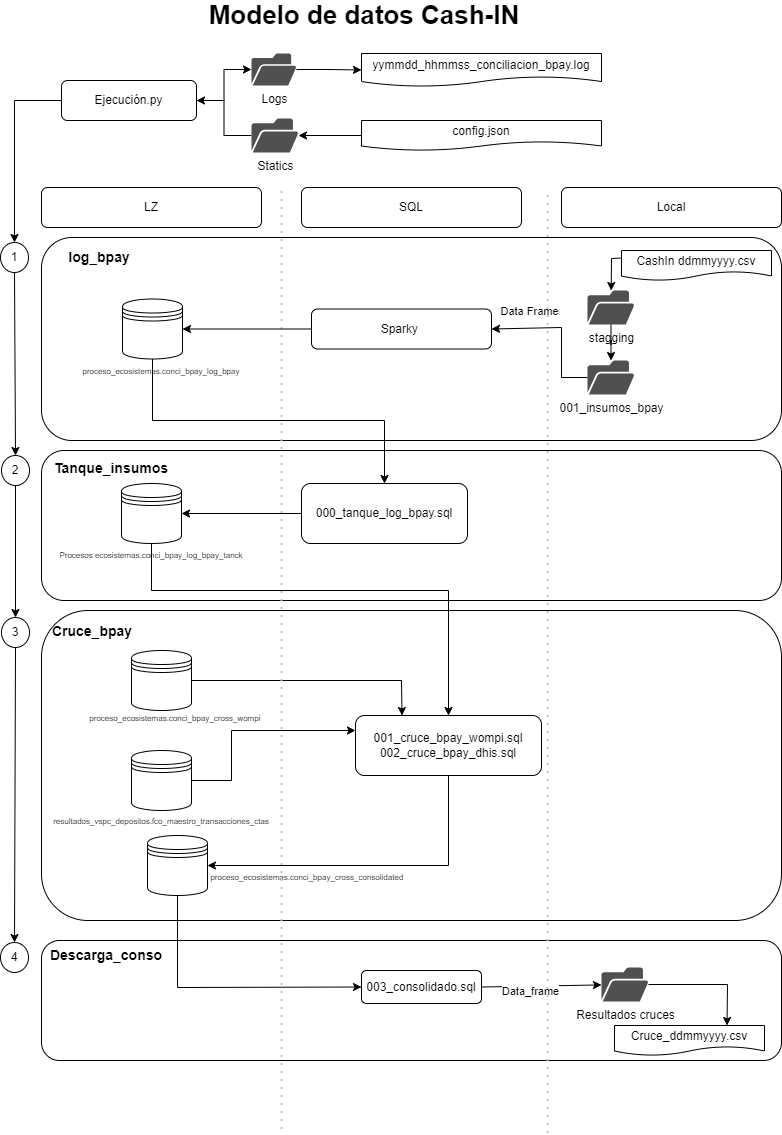
Debido a la gran cantidad de clientes que el banco maneja, y que se encuentran en mora, existen diferentes mecanismos de gestión para cada cliente y obligación. Actualmente se cuenta con un sistema de priorización por lotes, basado en la exposición de la deuda y la probabilidad de pago, sin embargo, al negocio le gustaría conocer de forma anticipada (un mes) que tan probable es que un cliente acepte una de las opciones de pago, para que esta sea incorporada como una variable más, dentro de la priorización por lotes que actualmente existe, y de esta forma tener una eficiencia mayor, y una gestión más intensa sobre los clientes con alta probabilidad de aceptar una opción de pago, y por consiguiente ser más eficientes, y contribuir a que el índice de cartera vencida del banco se ralentice.

# Alcance

* Predecir si un cliente en mora aceptará o no una opción de pago preaprobada en el siguiente mes, con el fin de mejorar la gestión de cartera vencida y la eficiencia en la recuperación de deudas.
* Cobertura del Modelo
* Clientes en mora (desde el día 1 de mora).
* Segmento de gestión directa y aliados.
* Opciones de pago preaprobadas (máximo 3 por cliente/mes).
* Ventana de predicción de 1 mes
* El modelo permitirá:
* Priorizar clientes con alta probabilidad de aceptación, optimizando los recursos de gestión.
* Reducir el índice de cartera vencida, al enfocar esfuerzos en los clientes más propensos a pagar.
* Evitar judicialización y castigos, disminuyendo costos y pérdidas de capital
* Enfoque analítico
* **Tipo de modelo:** Clasificación binaria (¿acepta o no acepta?).
* **Entradas clave:** Historial de mora, uso de opciones de pago previas, perfil financiero del cliente, etc.
* **Salida:** Probabilidad de aceptación de una opción de pago en el siguiente mes.

# Proceso

# Pasos de la rutina



### **Componentes de la rutina**

**ejecucion.py**

El archivo es el punto de entrada del ejecutable que coordina y orquesta el proceso. Esto involucra a “etl.py” y utiliza la configuración proporcionada en el archivo “config.json”. Además, se encarga de configurar y gestionar la generación de archivos de registros (logs) en la carpeta “logs”.

**config.json**

Se utiliza para almacenar parámetros y configuraciones necesarias para la ejecución de la rutina.

**etl.py**

Contiene un conjunto de funciones diseñadas para facilitar la carga de insumos manuales y la ejecución recursiva de scripts SQL almacenados en una carpeta especifica.

**fecha\_conciliacion:**

La clase fecha\_conciliacion es responsable de solicitar al usuario la fecha de conciliación necesaria para el procesamiento posterior de datos. Esta clase hereda de Step y define un único método ejecutar. Al ser ejecutado, el método solicita al usuario que ingrese una fecha en el formato AAAAMMDD. Esta entrada es convertida a un entero, eliminando cualquier espacio en blanco, y se guarda en los parámetros globales bajo la clave f\_efectiva\_trn. Finalmente, estos parámetros actualizados y se almacenan utilizando el método setPayload. Esta fecha será utilizada en pasos posteriores del flujo de procesamiento de datos.

**Log Bpay:**

El módulo log\_bpay es una clase diseñada para automatizar el procesamiento y transferencia de archivos CSV dentro de un flujo de trabajo específico. Este módulo gestiona archivos ubicados en la carpeta stagging, que está dentro de la carpeta de insumos 001\_insumo\_bpay. El proceso comienza con la creación de las carpetas necesarias (stagging e ingested) si no existen. Luego, el módulo busca archivos CSV en stagging y, si encuentra alguno, lee el primer archivo, transforma los datos relevantes (convirtiendo la columna amount a tipo entero y renombrando la columna date a date\_trx), y sube estos datos transformados a una zona de procesos (LZ: proceso\_ecosistemas.conci\_bpay\_log\_bpay) utilizando el método subir\_df de sparky . Una vez que los datos han sido subidos, el archivo CSV procesado se mueve a la carpeta ingested, consolidando así los archivos que han sido subidos a la LZ. Si no se encuentran archivos CSV en la carpeta stagging, el módulo notifica al usuario. Este proceso asegura que los datos se procesen y transfieran de manera ordenada y eficiente, manteniendo un registro de los archivos procesados.

**Tanque Insumo:**

La clase “Tanque\_insumos” automatiza el proceso de respaldo y actualización de datos en el sistema, garantizando que los registros de la tabla “conci\_bpay\_log\_bpay\_tanck” estén correctamente gestionados y optimizados para su procesamiento continuo dentro del sistema de gestión de insumos en la zona de procesos. El query comienza con la creación de una tabla de respaldo denominada “conci\_bpay\_log\_bpay\_tanck\_backup”, replicando la estructura de la tabla original “conci\_bpay\_log\_bpay\_tanck” y organizando los datos en formato PARQUET con particiones por año, mes y día. A continuación, se realiza la carga de datos desde “conci\_bpay\_log\_bpay\_tanck” hacia “conci\_bpay\_log\_bpay\_tanck\_backup”, con transformaciones como el ajuste de tipos de datos y la extracción de la fecha de transacción para convertirla en el campo numérico “f\_trx”. Posteriormente, la tabla original conci\_bpay\_log\_bpay\_tanck se elimina para potencialmente ser reemplazada o actualizada con nuevos datos. Luego, se re-crea y carga “conci\_bpay\_log\_bpay\_tanck” utilizando los datos previamente respaldados desde “conci\_bpay\_log\_bpay\_tanck\_backup”. Finalmente, se realizan cálculos estadísticos sobre “conci\_bpay\_log\_bpay\_tanck” para mejorar su rendimiento y usabilidad futura. Este proceso garantiza la gestión eficiente y organizada de los datos dentro del sistema de gestión de insumos en la zona de procesos.

|  |  |
| --- | --- |
| **INPUT** | **OUTPUT** |
| proceso\_ecosistemas.conci\_bpay\_log\_bpay | proceso\_ecosistemas.conci\_bpay\_log\_bpay\_tanck |

**Cruce Bpay**

La clase Cruce\_bpay realiza un proceso integral de cruce y consolidación de datos utilizando varios tipos de joins para integrar información clave de diferentes fuentes de datos. Primero, utiliza un FULL OUTER JOIN entre las tablas bpay y wompi para combinar datos de transacciones de BPAY y Wompi. Este tipo de join asegura que todas las filas de ambas tablas se incluyan en el resultado, incluso si no hay una coincidencia en las claves de unión (en este caso, los números de seguimiento).

En el segundo query, se emplea nuevamente un FULL OUTER JOIN entre las tablas bpay y dhis. Aquí, los datos de transacciones de BPAY se comparan con los datos de transacciones bancarias de DHIS usando el número de seguimiento como clave de unión. Este tipo de join garantiza que todas las transacciones de ambas fuentes se consideren en el resultado final, permitiendo identificar y comparar transacciones coincidentes o ausentes entre ambas plataformas.

El proceso culmina con la creación de la tabla conci\_bpay\_cross\_consolidated, donde se integran los resultados de los joins anteriores utilizando un FULL OUTER JOIN entre las tablas wompi\_table y dhis\_table. Aquí, se combinan y consolidan detalles importantes como tipos de documentos, números de cuenta, valores de transacción, referencias, fechas y estados de las transacciones de ambas fuentes, asegurando un registro completo y validado de las operaciones financieras realizadas, con el fin de facilitar análisis posteriores y asegurar la integridad de los datos cruzados.

Tabla 1 Cruce Bpay vs Wompi

|  |  |
| --- | --- |
| **001\_cruce\_bpay\_wompi.sql** | |
| **INPUT** | **OUTPUT** |
| * resultados\_wompi.wompi\_transactions * proceso\_ecosistemas.conci\_bpay\_log\_bpay\_tanck | * proceso\_ecosistemas.conci\_bpay\_cross\_wompi |

Tabla 2 Cruce Bpay vs Depositos

|  |  |
| --- | --- |
| **002\_cruce\_bpay\_dhis.sql** | |
| **INPUT** | **OUTPUT** |
| * resultados\_vspc\_depositos.fco\_maestro\_transacciones\_ctas * proceso\_ecosistemas.conci\_bpay\_log\_bpay\_tanck | * proceso\_ecosistemas.conci\_bpay\_cross\_dhis |

Tabla 3 Cruce Bpay vs Depositos vs Wompi

|  |  |
| --- | --- |
| **003\_consolidado.sql** | |
| **INPUT** | **OUTPUT** |
| * proceso\_ecosistemas.conci\_bpay\_cross\_wompi * proceso\_ecosistemas.conci\_bpay\_cross\_dhis | * proceso\_ecosistemas.conci\_bpay\_cross\_consolidated |

**Descarga Conso**

se encarga de descargar y guardar los resultados de una consulta SQL en un archivo CSV. El método ejecutar obtiene los parámetros necesarios desde el payload, utiliza un helper para ejecutar una consulta SQL y obtener los resultados en un DataFrame, y luego guarda este DataFrame en un archivo CSV en la ruta especificada, nombrado según la fecha efectiva de la transacción. En resumen, descarga\_conso automatiza la extracción de datos de una base de datos y los transforma en un formato CSV para facilitar su análisis y almacenamiento.

Log y manejo de errores

La rutina genera en la carpeta logs:

* Un archivo .log, el cuál toma el nombre de “yyyymmdd\_hhmmss\_.log”. Este archivo contiene las sentencias que se van ejecutando de la rutina, registrando la hora con la que inicia la rutina y los tiempos que toma cada ejecución.