Automatización de procesos en el programa “SOMOS” de la ESSA mediante RPA para la creación de contactos, asignación de productos financieros y actualización de saldos y transacciones

Brayan Ferney Uribe Rodríguez

Trabajo de grado para optar por el título de Ingeniero de Sistemas

Director

Carlos Adolfo Beltrán Castro

MSc. Ingeniería de Sistemas e Informática

Tutor

Román Alexis Suárez Cáliz

MSc. Ingeniería de Sistemas e Informática

Universidad Industrial de Santander

Facultad de ingenierías fisicomecánicas

Escuela de ingeniería de sistemas e informática

Bucaramanga

2025

Dedicatoria

A todos los estudiantes de la Universidad Industrial de Santander que exploren el potencial de la automatización y las tecnologías de la industria 4.0.

Agradecimientos

Agradezco profundamente a mis padres y hermano, que por su amor incondicional y sus sacrificios silenciosos esto no habría sido posible. A mis amigos, quienes hicieron de este camino una experiencia mucho más amena y llevadera, con las risas compartidas, los conocimientos intercambiados y el apoyo mutuo en los momentos de mayor presión. Gracias a la Universidad Industrial de Santander, mi alma máter, por brindarme una formación académica excelente y por ser el espacio donde crecí como profesional y como persona. A sus docentes y directivos, por compartir su conocimiento y guiarme en este proceso. A la Electrificadora De Santander por abrirme sus puertas y confiar en mi para el desarrollo de este proyecto. Finalmente, a mí, por el esfuerzo invertido, las noches de estudio, por haber superado los obstáculos y creer en mi capacidad para llevar este proyecto a su culminación.

Tabla de Contenido

Lista de Figuras

Glosario

CrediSomos Ágil / CrediSomos Tarjeta: Las dos modalidades de productos financieros ofrecidos a los clientes a través del programa "SOMOS". La automatización gestiona la asignación de estos productos a los nuevos contactos en el SAC

Automatización Robótica de Procesos (RPA): Tecnología que permite automatizar tareas repetitivas y basadas en reglas mediante la configuración de "bots" de software que imitan la interacción humana con los sistemas digitales.

SAC (Sistema de Administración Comercial): Sistema interno de ESSA donde se gestiona la información de los clientes, los productos financieros, y las transacciones y los saldos del programa "SOMOS". Es el sistema de destino principal de la automatización.

SOMOS (Programa "SOMOS"): El programa de lealtad y financiación de la ESSA, dirigido a sus clientes. El objetivo de este proyecto de grado fue automatizar los procesos operativos clave de este programa.

T88 y TSL: Nombres de los archivos estándar descargados desde la plataforma Redeban. El archivo T88 contiene el detalle de las transacciones (movimientos) realizadas, mientras que el archivo TSL contiene los saldos actualizados de las cuentas.

Thick Mode (Modo Grueso): Modalidad de conexión a la base de datos Oracle que requiere la instalación previa del software "Oracle Instant Client" en la máquina que ejecuta el proceso. Se recurre a este modo en entornos corporativos con altas restricciones de seguridad, ya que utiliza las librerías nativas de Oracle para la comunicación, brindando una mayor compatibilidad con configuraciones de red complejas y firewalls que podrían bloquear conexiones más directas.

OnCredit: Plataforma externa utilizada por la ESSA para la gestión y validación de la información de los clientes potenciales del programa “SOMOS”. El script de Python utiliza un reporte de esta plataforma para verificar la exactitud de los datos de las solicitudes.

PL/SQL (Procedural Language / Structured Query Language): Lenguaje de programación procedural de Oracle utilizado para crear procedimientos almacenados directamente en la base de datos. En este proyecto, se invocan procedimientos PL/SQL para realizar la carga masiva y la actualización de datos.

Power Automate (Cloud y Desktop): Plataforma de Microsoft para la automatización de procesos, la cual se usó en su modalidad Cloud para la orquestación del flujo de trabajo, y en su modalidad Desktop para la ejecución de tareas en la maquina local, como ejecución de scripts de Python y la interacción con el portal web de Redeban.

Python: Lenguaje de programación de alto nivel utilizado en este proyecto para realizar las tareas de procesamiento de datos más complejas, como la limpieza y validación de archivos Excel, la transformación de datos y la conexión a la base de datos Oracle.

Redeban: Plataforma externa utilizada para el procesamiento de transacciones financieras. El bot descarga los archivos T88 y TSL desde su portal para actualizar los movimientos y saldos de los clientes en el SAC.

Resumen

Título: Automatización de procesos en el programa "SOMOS" de la ESSA mediante RPA para la creación de contactos, asignación de productos financieros y actualización de saldos y transacciones.

Autores: Brayan Ferney Uribe Rodríguez

Palabras clave: RPA, Python, Microsoft Power Platform, Modelo Hibrido de Automatización, SOMOS.

Descripción:

La Electrificadora de Santander (ESSA) se enfrentaba a importantes ineficiencias operativas en su programa de fidelización "SOMOS", como consecuencia de la gestión manual y laboriosa de los procesos relacionados con la creación de contactos, la asignación de productos financieros y las actualizaciones de transacciones y saldos. Estas deficiencias en los procedimientos provocaban una lentitud en la activación del servicio, una mayor vulnerabilidad a los errores humanos y un efecto perjudicial general en la experiencia del cliente.

Este proyecto tiene como objetivo solucionar los problemas descritos mediante la implementación de una estructura hibrida de automatización robótica de procesos (RPA). La solución aprovecha las ventajas de Microsoft Power Platform (Power Automate) para coordinar la ejecución de los flujos de trabajo, y se utiliza Python para llevar a cabo el procesamiento avanzado de datos y la conexión directa con la base de datos Oracle del SAC.

Los resultados demuestran el efecto de la transformación: el tiempo de procesamiento diario se redujo en más de un 90% (de 2,5 h a 15 min), se eliminaron los errores tipográficos y la activación de los productos financieros se realizan el mismo día en que se hace la solicitud del cliente. En el aspecto financiero, el proyecto supondrá un ahorro anual de más de 16.5 millones de pesos colombianos. De este modo, el proyecto no solo agiliza uno de los procesos más críticos para el éxito del programa SOMOS, sino que también valida el modelo hibrido de automatización como un paradigma eficaz y rentable de la transformación digital en la ESSA.

---

[*] Trabajo de Grado

[**] Facultad de Ingenierías Fisicomecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Carlos Adolfo Beltrán Castro, Ph.D.

Abstract

Title: Process Automation in ESSA "SOMOS" Program through RPA for Contact Creation, Financial Product Assignment, and Balance and Transaction Updates.

Authors: Brayan Ferney Uribe Rodriguez

Keywords: RPA, Python, Microsoft Power Platform, Hybrid Automation Model, SOMOS.

Description:

The Electrificadora de Santander (ESSA) faced significant operational inefficiencies in its loyalty program "SOMOS," as a result of the manual and labor-intensive management of processes related to contact creation, financial product assignment, and transaction and balance updates. These procedural shortcomings caused delays in service activation, increased vulnerability to human error, and an overall negative impact on the customer experience.

This thesis aims to address the described problems through the implementation of a hybrid robotic process automation (RPA) structure. The solution leverages the advantages of Microsoft Power Platform (Power Automate) to coordinate workflow execution, while Python is used to perform advanced data processing and establish a direct connection with the SAC's Oracle database.

The empirical results demonstrate the impact of this transformation: daily processing time was reduced by more than 90% (from 2.5 hours to 15 minutes), typographical errors were completely eliminated, and the activation of financial products now takes place on the same day the customer submits the request. From a financial standpoint, the project will yield annual savings of more than 16.5 million Colombian pesos. Thus, the project not only streamlines one of the most critical processes for the success of the SOMOS program but also validates the hybrid automation model as an effective and cost-efficient paradigm for digital transformation at ESSA.

---

[*] Degree Work

[**] Faculty of Physics-Mechanics Engineering. School of Systems Engineering and Informatics. Advisor: Carlos Adolfo Beltrán Castro, Ph.D.

Introducción

Actualmente en el mundo empresarial, donde todo cambia rápidamente, existen dos cosas claves para que un negocio funcione bien: adaptarse rápidamente y mantener a los clientes contentos. La revolución digital (industria 4.0) brinda increíbles herramientas para que los procesos sean más eficientes, eliminando el trabajo manual aburrido, y dejando que los equipos se concentren en lo que es realmente importante en la empresa.

La Electrificadora de Santander (ESSA) entiende esto a la perfección. La empresa está continuamente buscando formas de innovar y mejorar su servicio. En este caso, el programa de fidelización "SOMOS", busca crear mejores relaciones con sus usuarios y hacer que su experiencia sea más positiva.

El programa "SOMOS" ofrece beneficios y opciones de financiamiento a los clientes de la ESSA, construyendo una relación solida con ellos. Pero hay un problema: no importa qué tan bueno sea el programa en papel, si la parte operativa falla, la experiencia del cliente sufre.

El equipo que maneja las operaciones diarias del programa enfrentaba un verdadero dolor de cabeza. Tenían que realizar manualmente varias tareas importantes: Crear nuevos contactos en el sistema, asignar productos financieros (Credisomos Tarjeta o Credisomos Agil), actualizar saldos y transacciones en el Sistema de Administración Comercial (SAC). Toda la información llegaba al final del día laboral, lo que provocaba retrasos inevitables. Si se realiza un registro hoy, se tenía que esperar hasta el día siguiente para que el producto fuese activado o para que las compras se registraran. Esta espera no solo afectaba a los clientes que querían usar sus beneficios inmediatamente, sino que también hacía que el programa no fuera tan eficiente y competitivo de

lo que podría ser.

Frente a este problema, este proyecto de grado abordó este desafío mediante el diseño e implementación de una solución de Automatización Robótica de Procesos (RPA)

La estrategia fue usar dos herramientas: Microsoft Power Automate (versión Cloud y Desktop) para la automatización robótica de procesos (RPA) y Python para manejar datos complejos, realizar validaciones y comunicarse con las bases de datos. Esta combinación es como tener lo mejor de dos mundos: la facilidad de uso de Power Automate para automatizar tareas repetitivas y el poder de Python cuando se necesita algo más sofisticado. Así se creó una solución a medida para el programa SOMOS de la ESSA, que resuelve exactamente lo que necesitaban.

1. Planteamiento del problema y justificación

El programa “SOMOS” es un programa de lealtad y relacionamiento de la Electrificadora de Santander (ESSA) en donde los usuarios de la ESSA podrán recibir descuentos, adquirir productos y servicios mediante dos modalidades de financiamiento: tarjeta de crédito (CrediSomos tarjeta) y crédito rotativo sin tarjeta (CrediSomos Ágil) (Electrificadora de Santander, s. f.).

Los encargados de la gestión en la creación de los contactos, la asignación del tipo de producto financiero (Tarjeta o Crediagil), el cargue de las transacciones y la actualización de los saldos en el Sistema de Administración Comercial (SAC), tienen la necesidad de agilizar el proceso, debido a que los asesores comerciales brindan la información de los clientes potenciales (un promedio de 20 por día, 2-3h de procesamiento manual) al final del día laboral. Por lo tanto, los encargados de esta gestión no tienen la posibilidad de realizar el proceso en ese mismo día,

retrasando la activación de los productos financieros y el cargue de las ventas (transacciones y saldos) hasta el día siguiente. Esta demora en la activación de productos genera frustración en los clientes que necesitan realizar compras para el mismo día, impactando negativamente en su satisfacción y reduciendo la competitividad del programa “SOMOS”.

Hoy en día, las empresas se encuentran en una etapa de transformación digital conocida como la revolución de la industria 4.0, en donde uno de sus objetivos es poder automatizar y lograr la eficiencia en los procesos de negocios con la ayuda de tecnologías más eficientes, reduciendo el tiempo, los errores humanos y la necesidad de mano de obra, permitiendo que las personas se dediquen a actividades productivas, estratégicas y que aporten valor a la empresa (Gómez Gonzales, 2020).

Hay muchos estudios en el ámbito de la automatización de procesos en los sectores financieros y empresariales que han demostrado que la implementación de RPA (Robotic Process Automation) puede transformar en gran manera estos procesos, reduciendo errores, tiempo, optimizando el uso de recursos y permitiendo que los empleados dediquen su tiempo en actividades que den mayor valor a la empresa.

En el mercado existen varias plataformas de automatización RPA como UIPath, Automation Anywhere, Blue Prism, etc. Estas tecnologías son reconocidas por su robustez y capacidad para gestionar procesos complejos, ofreciendo amplias funcionalidades y soporte para IA. Sin embargo, presentan algunas desventajas en términos de costos y complejidad: de manera general, requieren licencias comerciales costosas y su implementación puede demandar recursos técnicos especializados para poder realizar integraciones adecuadas con los sistemas existentes. Además, en entornos donde se manejan múltiples aplicaciones y que requieran una integración

fluida, la interoperabilidad de estas herramientas con tecnologías específicas puede no ser nativa.

En contraste, Power Automate destaca por tener integraciones de forma nativa con el ecosistema de Microsoft, siendo esta una ventaja para las organizaciones que ya poseen un licenciamiento de Microsoft 365. La integración directa con aplicaciones como Excel, OneDrive y SharePoint facilita la conexión con los sistemas de información ya existentes, de modo que reduce los tiempos para implementar y permite una mejor comunicación entre plataformas. Adicionalmente, Power Automate suele tener un costo menor en comparación con las otras soluciones, volviéndolo atractivo para organizaciones con ciertas restricciones en el presupuesto. Sin embargo, es importante reconocer que, en escenarios de alta complejidad o proyectos de gran envergadura, herramientas líderes como UIPath, ofrecen mayores capacidades de adaptación a procesos extremadamente específicos o versátiles.

Debido a que la ESSA posee licenciamiento de Microsoft 365, la empresa impone el uso de Power Automate como herramienta principal de automatización de procesos. Por lo anterior, al automatizar este proceso mediante RPA con Power Automate y potenciarlo con Python, este proyecto busca agilizar la carga de información para que se realice el mismo día en que se recibe. Esto incrementará la eficiencia operativa al acelerar el flujo de trabajo y liberar recursos, habrá una reducción de errores manuales ya que se minimizará la intervención humana en tareas repetitivas y mejorará la satisfacción del cliente al ofrecerle acceso más rápido a sus productos y beneficios. De modo que el programa "SOMOS" fortalecerá su competitividad frente a otros servicios similares.

2. Objetivos

2.1. Objetivo general

- Diseñar e implementar una solución de Automatización Robótica de Procesos (RPA) para agilizar el proceso diario de creación de contactos, asignación de productos financieros y actualización de transacciones y saldos en el Sistema de Administración Contable (SAC) del programa "Somos" de la ESSA, con el fin de incrementar la eficiencia operativa, reducir errores manuales y mejorar la satisfacción del cliente.

2.2. Objetivos específicos

- Realizar un análisis detallado del actual flujo de información en el programa SOMOS respecto a los procesos a automatizar.
- Diseñar el nuevo flujo de trabajo y la arquitectura de la solución.
- Implementar un Robotic Process Automation (RPA) con power automate utilizando flujos de nube y escritorio junto a python para la automatización de los procesos clave de creación de contacto, asignación de productos financieros y actualización de transacciones y saldos.

3. Estado del arte

En los últimos años, la Automatización de Procesos (RPA) se ha consolidado como una pieza fundamental en la transformación digital. El RPA consiste en el uso de bots, los cuales pueden ejecutar tareas manuales y rutinarias, teniendo en cuenta las reglas que se definan, y emulan las acciones humanas en aplicaciones de software. El objetivo principal es reducir los costos, errores y tiempos de ejecución en los procesos, de modo que se liberen los empleados y puedan realizar labores de mayor valor agregado. Según Impacto TIC (2024), en Colombia (y globalmente) el RPA ha cobrado bastante fuerza: se estima que cerca del 70-80% de los procesos basados en reglas típicos pueden ser automatizados con la ayuda de RPA, ya que los empleados usan entre un 10% y un 25% del tiempo en tareas repetitivas que no aportan ningún valor. Es por

eso por lo que muchas empresas colombianas de sectores energéticos, financieros, etc., están apostando por robots de software para la optimización de las operaciones. A continuación, se revisan los avances y casos de uso de RPA en ámbitos relevantes como lo son el sector financiero y los procesos de onboarding de clientes.

RPA en el sector financiero y de servicios

El sector financiero ha sido de los primeros en adoptar el RPA para la automatización de procesos operativos, ya sea en bancos o en compañías de servicios financieros. Hay varios estudios que reportan que, al combinar RPA con algoritmos de inteligencia artificial, las instituciones financieras han logrado realizar una transformación en sus operaciones centrales, haciéndolas exponencialmente más eficientes, y a su vez pueden personalizar sus servicios al cliente y mejorar la seguridad en las transacciones. En banca, el RPA se aplica exitosamente en tareas de back office y middle office que antes requerían una intensa labor manual, como conciliar cuentas, procesar pagos, verificación de cumplimiento normativo o generar informes periódicos.

Existen casos que ilustran el impacto de RPA en finanzas. Como ejemplo, el banco Barclays introdujo RPA a gran escala en varios procesos (desde cuentas por cobrar hasta cierre de cuentas fraudulentas), logrando reducir sus provisiones por incobrables en aproximadamente 225 millones de dólares anuales y eliminando el equivalente a más de 120 puestos de trabajo de tareas repetitivas. Similarmente, el grupo bancario ANZ logró recortar en más de un 30% los costos operativos en algunas funciones al automatizar más de 40 procesos con robots de software. Estos resultados demuestran el potencial del RPA para mejorar la eficiencia y reducir costos en el sector financiero; sin embargo, la literatura también advierte de algunos desafíos: algunos bancos han enfrentado dificultades para escalar sus iniciativas de automatización por la falta de estrategias

integrales, haciendo énfasis en la importancia de abordar el RPA dentro de una visión estratégica y con el apoyo coordinado de áreas como TI (McKinsey, s. f.).

RPA en procesos de onboarding de clientes

La incorporación ("onboarding") de nuevos clientes es un proceso crítico en varias industrias (banca, seguros, telecomunicaciones, etc.), y suele implicar varias etapas de recopilación, verificación y registro de datos. De manera tradicional, para dar de alta a un cliente se puede requerir desde completar formularios y recopilar documentos de identidad, hasta tener que realizar verificaciones regulatorias (como procedimientos "know your customer" o KYC) y configurar cuentas en diversos sistemas. El RPA ha emergido como una solución eficaz que ayuda a agilizar este proceso: en el ámbito bancario, se ha visto que una solución RPA puede automatizar la mayoría de estas tareas, disminuyendo de manera significativa los costos de operación, reduciendo el riesgo de errores y mejorando los tiempos necesarios para realizar una incorporación de un nuevo cliente. En lugar de semanas, el onboarding se puede completar en cuestión de días o incluso horas, debido a los robots de software que extraen datos de varias fuentes, los validan frente a bases de datos gubernamentales o crediticias y efectúan de manera automática las altas en los sistemas internos correspondientes.

Hay un caso de estudio que ilustra los beneficios del RPA en onboarding durante un escenario crítico: en un banco de Bermudas durante la pandemia de COVID-19, se volvió problemática el alta de nuevos clientes gracias al cierre de sucursales y la limitada disponibilidad del personal. Esta institución implementó un bot RPA integrado con su chatbot bancario para así automatizar la verificación de documentos KYC y el ingreso de datos de nuevos clientes, de modo que se eliminó la necesidad de las validaciones manuales. Esto trajo como resultado una reducción

drástica en la carga de trabajo del call center y se aceleró la incorporación de clientes pese a las restricciones sanitarias del momento, mejorando así la experiencia inicial del usuario (TechnePLus, s. f.). Con este ejemplo, se pone de manifiesto cómo el RPA puede aportar resiliencia y eficiencia en los procesos de onboarding, asegurando que la incorporación de clientes sea ágil, precisa y conforme a las normativas, incluso bajo condiciones operativas adversas.

Con los casos vistos, se confirma que la RPA es una tecnología madura y de gran impacto, con aplicaciones probadas en la automatización de procesos financieros y de onboarding. Los casos de estudio de empresas como Barclays demuestran que es posible lograr reducciones drásticas en tiempos de procesamiento y tasas de error, liberando al personal para que realicen tareas de mayor valor.

Sin embargo, en la literatura también se sugiere que las implementaciones más exitosas frecuentemente requieren ir más allá de las capacidades nativas de las plataformas RPA. En el caso de tareas que involucren una lógica de negocio compleja, validación de datos contra varias fuentes y la integración con bases de datos legadas, un enfoque híbrido que mezcle una plataforma low-code (como Power Automate) con el poder de scripting de lenguajes como Python se convierte en una práctica recomendada. Con este enfoque, es posible orquestar el flujo de trabajo de manera más fácil mientras las operaciones de datos pesadas se delegan a un lenguaje más robusto y eficiente.

Es por eso por lo que este proyecto se sitúa en esa intersección, aplicando los principios de RPA, validados en la literatura a un problema crítico en el programa "SOMOS", y adoptando un enfoque híbrido y pragmático para resolver las limitaciones de una solución netamente low-code.

4. Marco de referencia

4.1. Automatización Robótica de Procesos (RPA)

La automatización robótica de procesos (RPA) es una tecnología basada en software que imita las acciones humanas para hacer tareas rutinarias de forma automatizada (Mirabete et al., 2024). Básicamente, un "robot" de software replica las acciones que un humano haría en las aplicaciones, pero con una mayor rapidez y consistencia. Una característica importante de la RPA es el uso de bots no invasivos, los cuales interactúan directamente en las interfaces gráficas, de modo que no se requieran cambios en los sistemas subyacentes (Mirabete et al., 2024). Esto quiere decir que la RPA puede superponerse sobre las aplicaciones que son heredadas o sistemas actuales sin necesidad de integraciones complejas, únicamente haciendo uso de las mismas pantallas o API disponibles.

La tecnología RPA posee beneficios que han sido ampliamente documentados en varios sectores (Mirabete et al., 2024). Principalmente, la automatización de procesos manuales reduce en gran manera los tiempos de procesamiento y mejora la eficacia operativa. De igual forma, cuando se elimina la intervención humana en las tareas repetitivas, disminuyen los errores operativos causados por descuidos o inconsistencias. Financieramente hablando, genera un ahorro de costos al reducir la mano de obra en tareas de bajo valor añadido. Además, se ha observado que hay mejoras en la experiencia del cliente, ya que la RPA brinda respuestas más rápidas y servicios más consistentes (Mirabete et al., 2024). Estos beneficios son importantes para una empresa como la ESSA en su programa SOMOS, donde es importante la rápida creación de registros de clientes, actualización de saldos y asignaciones de productos financieros.

Es importante destacar que la RPA tiene dos tipos de modalidad: asistida y no asistida. La RPA asistida es donde el robot es un ayudante del usuario, por ejemplo, desencadenándose bajo demanda de un empleado, mientras que en la RPA desasistida los robots funcionan de manera

automática sin la intervención humana (Mirabete et al., 2024). En este proyecto, se hizo la implementación con automatización desasistida, de modo que los procesos de asignación de productos, actualización de saldos y creación de contactos se ejecuten en segundo plano de forma automática.

4.2. Microsoft Power Automate: Automatización de procesos en la nube y en el escritorio

Microsoft Power Automate es una plataforma de automatización de Microsoft que permite implementar flujos de trabajo automatizados integrando múltiples sistemas, combinando capacidades de automatización de flujos de trabajo en la nube con RPA de escritorio. Esta es una herramienta de bajo código, la cual habilita a las organizaciones a combinar la automatización de procesos robóticos (RPA), la inteligencia artificial y el análisis para lograr una mayor eficacia (Microsoft, s. f.). Para este proyecto, Power Automate actúa como la columna vertebral de la solución: coordina las tareas, conecta las aplicaciones involucradas y ejecuta la lógica necesaria tanto en entornos cloud como en workspaces locales.

Los flujos de nube de Power Automate permiten orquestar procesos empresariales que se inician automáticamente por eventos, acciones manuales o siguiendo una programación definida (Microsoft, s. f.). Gracias a la amplia gama de conectores predefinidos, Power Automate puede realizar conexiones con servicios en la nube (bases de datos, CRM, correo electrónico). Por ejemplo, la llegada de un nuevo correo electrónico puede desencadenar un flujo que actualice un archivo en una ruta de OneDrive de acuerdo con la información del correo. Estos flujos de nube manejan la lógica de negocio de alto nivel y las integraciones por medio de API, asegurando que los sistemas corporativos queden sincronizados sin intervención manual.

De manera complementaria, Power Automate ofrece flujos de escritorio para automatizar

tareas en aplicaciones que no cuenten con API accesibles o sistemas heredados. Los flujos de escritorio permiten automatizar todos los procesos repetitivos del escritorio a través de una interfaz intuitiva que da acciones predefinidas de arrastrar y soltar, o grabar las acciones del usuario. Un bot de Power Automate Desktop puede, por ejemplo, abrir una aplicación de legado de una empresa, ingresar datos, extraer información, etc. También puede interactuar con aplicaciones modernas o antiguas, como emuladores de terminal, aplicaciones web, aplicaciones de escritorio, u hojas de cálculo de Excel, todo gracias a que simula las acciones del usuario en la interfaz gráfica (clics, teclado) o incluso reconocimiento de imágenes (Microsoft, s. f.).

Un aspecto de Power Automate es la integración que hay entre flujos de nube y flujos de escritorio. Es posible que un flujo de nube desencadene la ejecución de un flujo RPA de escritorio. Por ejemplo, se tiene un flujo de nube que recopila información de un correo electrónico, y este invoca un flujo de escritorio que abre una página web y sube la información recopilada. En ese proceso, básicamente, los valores se transfieren del flujo de nube al bot de escritorio como variables de entrada, y una vez se complete, el bot retorna resultados mediante variables de salida al flujo de nube (Microsoft, s. f.).

Finalmente, a pesar de que Power Automate es una plataforma de bajo código, esta admite una extensibilidad por medio de código cuando es necesario. Particularmente, posee acciones para ejecutar scripts de Python dentro de flujos de escritorio (Microsoft, s.f.). Esto permite aprovechar código personalizado para tareas especializadas, que es el enfoque planteado en este proyecto al incorporar Python.

4.3. Python

En las soluciones de automatización de procesos, comúnmente es necesario manipular y

trasladar datos entre diferentes sistemas, realizando las transformaciones necesarias. Es por eso por lo que entra en juego los procesos de extracción, transformación y carga de datos junto con la potencia de Python. La idea es que se extraigan los datos de las fuentes, se hagan las transformaciones necesarias (limpieza, reformateo, combinaciones, cálculos, etc.) y se carguen a los sistemas de destino pertinentes. Gracias a la simplicidad, legibilidad y una amplia colección de bibliotecas, Python se ha convertido en la herramienta predilecta.

En este proyecto, Python complementa a Power Automate encargándose de las operaciones de datos que exceden las capacidades de los conectores estándar o que requieran una lógica más elaborada. Python es muy útil para la integración y transformación gracias a librerías como Pandas (manipulación de dataframes), Requests (llamadas a servicios web), y Openpyxl (lectura/escritura de Excel), entre otras.

Hay muchas experiencias que respaldan lo bueno que es Python en la automatización de procesos de datos. Por ejemplo, hay un estudio de caso del sector bancario colombiano donde implementaron scripts de Python para automatizar un flujo ETL (Extracción, Transformación y Carga de datos) interno en donde se redujo el tiempo de ejecución de 9 minutos a solo 1,5 minutos, dando resultados más rápidos y eficientes (Agudelo, 2023). Se puede observar cómo el scripting con Python acelera procesos que manualmente tomarían mucho más tiempo. Además de que Python, al ser un lenguaje multipropósito, permite incorporar lógica condicional, manejo de excepciones, y acceso a prácticamente cualquier sistema mediante API o drivers, ampliando los horizontes de la automatización más allá de lo que ofrece la interfaz de una herramienta RPA por sí sola.

5. Metodología

Para el desarrollo de este proyecto se empleó un enfoque híbrido y pragmático, basado en las prácticas del Ciclo de Vida del Desarrollo de Software (SDLC – Software Development Life Cycle), adaptado para la RPA.

Este enfoque consta de cinco fases principales: Planeación y Requerimientos, Diseño, Desarrollo, Pruebas y Despliegue, y como apoyo se usaron herramientas especializadas como los diagramas BPMN (Business Process Model and Notation).

5.1. Planeación y requerimientos

En esta fase, se aseguró la comprensión clara del problema y se definieron con precisión los requisitos necesarios para la implementación de la automatización.

Se realizó una sesión inicial con los usuarios responsables del programa SOMOS para recolectar la información que se tiene sobre el proceso actual, documentar las inconsistencias y entender de manera más clara los objetivos esperados de la automatización. De igual modo, se elaboraron los diagramas BPMN detallados del flujo actual (As-Is), en donde se identificaron las actividades manuales que generaban retrasos significativos en la creación de contactos, asignación de productos financieros y actualización de saldos en el SAC. Por último, se establecieron los requerimientos técnicos y los permisos necesarios para la ejecución del bot, los cuales son:

- Acceso a la base de datos Oracle del SAC.
- Credenciales para ingresar al SAC.
- Permisos de lectura/escritura en archivos almacenados en OneDrive.
- Autorización para la ejecución de scripts Python de manera local (mediante CMD).

5.2. Diseño

En esta fase, se diseñó el nuevo proceso automatizado, estableciendo la arquitectura técnica y funcional para la solución.

Entre las actividades realizadas en esta fase, está la realización del documento PDD en el que se describieron las actividades del proceso actual, las tareas a automatizar, las reglas específicas del proceso, y los criterios técnicos de la implementación. Además de que se detalló de manera clara el flujo futuro optimizado (To-Be) con la ayuda de los diagramas BPMN. De igual forma, se hizo la definición de la arquitectura de integración tecnológica, mostrando cómo Power Automate (Cloud y Desktop), Python, etc., interactuarían para lograr la automatización de los procesos.

5.3. Desarrollo

En esta fase se implementaron las automatizaciones descritas en el diseño, haciendo uso de herramientas de Microsoft Power Automate y scripts en Python.

Para la preparación del entorno de desarrollo, se realizó la instalación y configuración de Power Automate Desktop y Cloud en el equipo provisto por la ESSA. Además, se hizo la instalación y configuración del entorno de Python con bibliotecas clave (Pandas, Numpy, oracledb, Openpyxl, msal) para realizar transformaciones avanzadas de datos, conexión a la base de datos Oracle y manejo de documentos en la nube.

Para el caso de los flujos en Power Automate, se diseñaron los flujos automatizados específicos que permiten recibir archivos de Excel enviados por correo electrónico, almacenarlos automáticamente en OneDrive y disparar procesos adicionales según las reglas preestablecidas. Para los flujos de escritorio, se configuraron para automatizar el ingreso a la página de Redeban en donde se van a descargar los archivos T88 y TSL, necesarios para el cargue de transacciones y

actualización de saldos mediante interacciones automatizadas con su interfaz gráfica.

En cuanto a Python, se desarrollaron los scripts especializados para la validación de la información recibida, identificación de errores o inconsistencias, realizar transformaciones complejas sobre los datos, y hacer la carga final en la base de datos Oracle. Estos scripts se integraron junto a los flujos de Power Automate, permitiendo que sea una automatización coordinada y robusta.

5.4. Pruebas

En esta fase se validará integralmente la solución desarrollada mediante pruebas unitarias, pruebas de integración y pruebas finales con el usuario.

Cada script y cada flujo de Power Automate fueron probados de forma aislada, asegurando que cumplían de manera individual con los requisitos técnicos y funcionales definidos. Del mismo modo, se realizaron pruebas de integración para comprobar que la interacción entre Python y Power Automate ocurría sin errores. Finalmente, se llevaron a cabo sesiones con usuarios finales del programa SOMOS, en donde se validó que la automatización cumplía con sus expectativas operativas.

5.5. Despliegue y documentación

En esta última fase, se puso la solución automatizada en producción, se generó la documentación técnica y se realizaron las capacitaciones necesarias para asegurar la sostenibilidad de la solución en el tiempo.

Se hizo la preparación de los artefactos finales (flujos de Power Automate Cloud y Desktop, scripts de Python) y se instalaron en el entorno productivo de la ESSA. Además, se hizo

un manual técnico detallado, en donde se describe el paso a paso del funcionamiento de cada flujo, los scripts y procedimientos para gestión y soporte. Finalmente, se realizaron las sesiones necesarias para capacitar a los usuarios involucrados, de modo que entendieran claramente cómo operar y gestionar la solución implementada.

6. Desarrollo

Teniendo en cuenta las fases llevadas a cabo para el cumplimiento de los objetivos, en esta sección se explicará a detalle cómo se realizó la ejecución práctica del proyecto. A continuación, se presentarán las actividades realizadas para alcanzar exitosamente los objetivos establecidos.

6.1. Fase de planeación y requerimientos

En esta fase inicial, como ya se explicó anteriormente, el propósito principal fue obtener una vista clara sobre el proceso actual relacionado con la creación de contactos, asignación de productos financieros, cargue de transacciones y actualización de saldos en el Sistema de Administración Comercial (SAC).

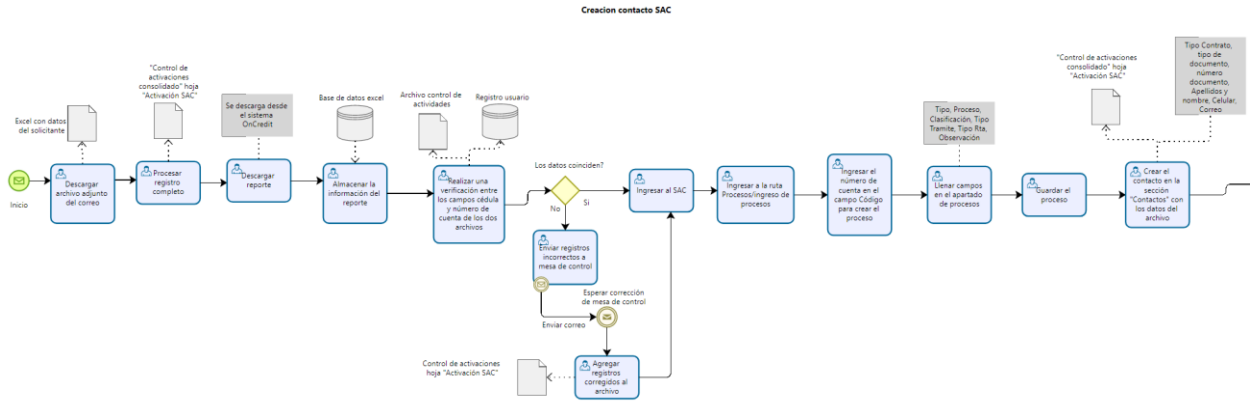
Inicialmente, se hizo una reunión con los responsables operativos del área de gestión comercial de la ESSA, específicamente con el equipo de mercadeo y ofertas. En esta reunión se tuvo la noción general del proceso y se logró identificar de manera clara los cuellos de botella operativos. Se logró detectar que, diariamente, alrededor de 20 clientes potenciales debían de ser procesados de manera manual, esto implica entre 2 a 3 horas de trabajo repetitivo por parte del personal encargado.

Una vez realizada la reunión, se procedió a elaborar los diagramas BPMN del proceso actual (As-Is), gracias a ese diagrama se logró identificar las actividades manuales susceptibles de

automatización.

Figura 1

Diagrama BPMN As-Is para el proceso creación de contacto

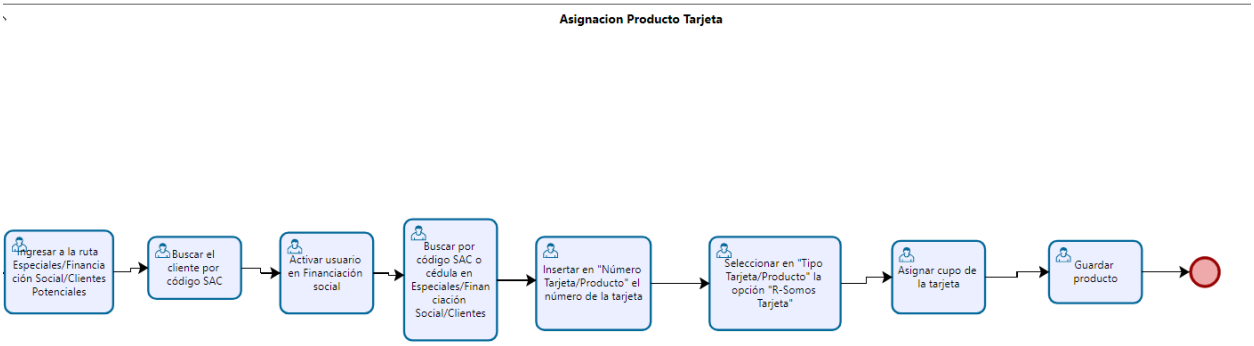


Nota: Figura hecha por el autor. Representa el proceso actual de creación de contacto sin la automatización.

Como se puede observar en la figura anterior (Figura 1), el proceso de creación de contacto es un proceso manual que empieza desde que un encargado del programa SOMOS envía un correo con un archivo adjunto (Excel), el cual es un consolidado de todos los nuevos usuarios del programa, y termina con la creación del contacto en el SAC. Este proceso es igual tanto para los usuarios de tarjeta como para los de Crediágil, con la diferencia de que el archivo en donde se procesa el registro para los usuarios de Crediágil es otro.

Figura 2

Diagrama As-Is para el proceso asignación de producto financiero.

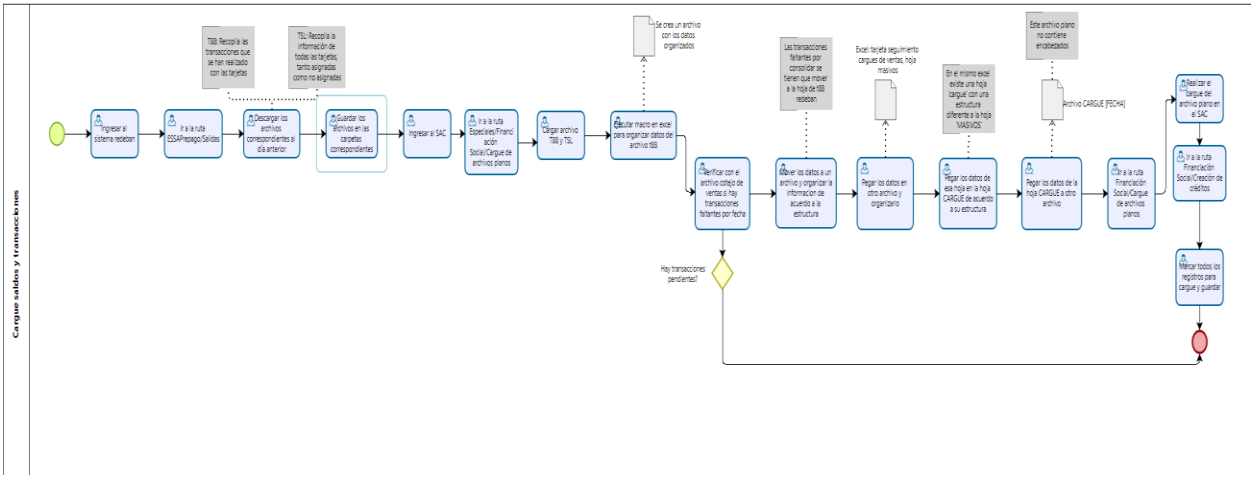


Nota: Figura hecha por el autor. Representa el proceso actual de asignación de producto financiero sin la automatización.

En la figura anterior se puede evidenciar el proceso manual que se realiza para la asignación de productos financieros. Ese proceso se hace después de realizar la creación del contacto. Para el caso de Crediágil es el mismo proceso, pero cambian los inputs, es decir, en lugar del número de la tarjeta, se ingresa el número de cédula, y en el tipo de tarjeta/producto se ingresa "S-Somos Ágil".

Figura 3

Diagrama As-Is para el proceso cargue de transacciones y actualización de saldos.



Nota: Figura hecha por el autor. Representa el proceso actual de cargue de transacciones y actualización de saldos sin la automatización.

En la Figura 3 se puede ver cómo es el proceso manual para realizar el cargue de transacciones y actualizar saldos. En este proceso, se descargan dos archivos fundamentales desde la página de Redeban; el T88 posee los movimientos realizados de las tarjetas y el TSL tiene los saldos actualizados, todo esto a fecha del día anterior a la realización del proceso. El procedimiento finaliza con el cargue de los archivos al SAC, y un archivo adicional el cual extrae solo la información importante del T88.

Con los diagramas As-Is realizados, se procedió a solicitar los permisos y credenciales del SAC, la base de datos Oracle, Redeban, y se solicitó la licencia para Power Automate.

De esta forma, finalizó la primera fase, con los diagramas As-Is y los requerimientos solicitados, se da paso a la fase del diseño.

6.2. Fase de diseño

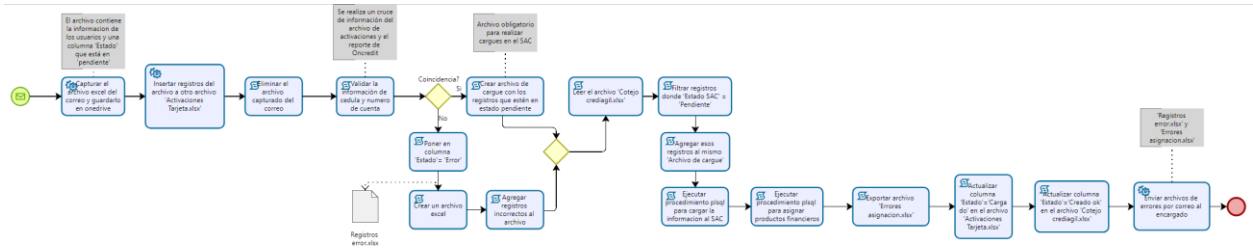
En esta fase se realizó la definición técnica detallada para resolver los problemas que se

identificaron en la anterior fase gracias a los diagramas As-Is.

En primer lugar, se realizó el modelado optimizado del nuevo proceso (To-Be), en donde se contemplaron los puntos más importantes a automatizar, como lo son la recepción automática de archivos de Excel por correo electrónico, y validar y procesar de manera automática los datos con la ayuda de Python.

Figura 4

Diagrama To-be para el proceso de creación de contactos y asignación de productos financieros



Nota: Figura hecha por el autor.

Como se puede observar en la Figura 4, se tiene el diagrama BPMN que corresponde al nuevo proceso automatizado para la creación de contactos y asignación de productos financieros del programa SOMOS.

El flujo inicia con la llegada de un correo electrónico, el cual contiene un archivo de Excel con la información de los usuarios que requieren el producto financiero de tarjeta, por lo que el estado inicial de esos usuarios es "Pendiente".

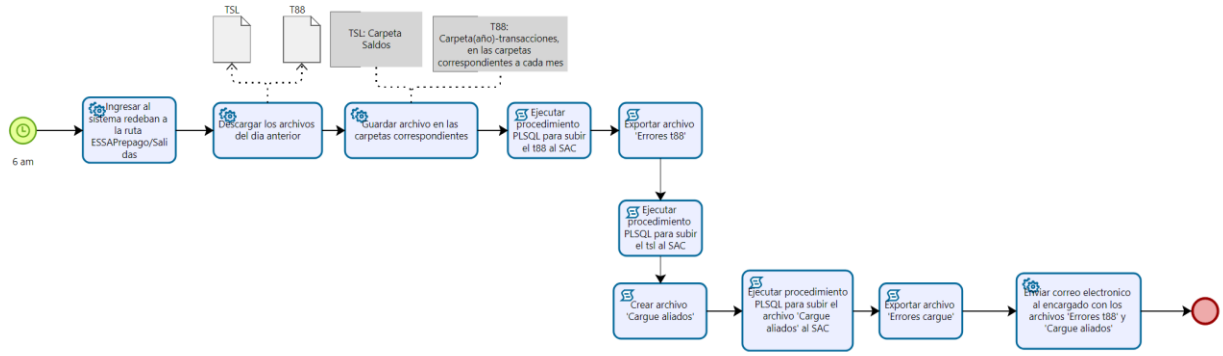
- Recepción del archivo: Se contempla la captura automatizada del archivo adjunto del correo y su almacenamiento en una carpeta en la nube (OneDrive)
- Consolidación inicial de los datos: Cuando se capture el archivo, los registros se transfieren de manera automática a un archivo maestro llamado “Activaciones Tarjeta.xlsx”. Luego, el archivo original que se recibió por correo se elimina.

- Validación y clasificación de registros: Debido a que la plataforma OnCredit (plataforma donde se descargan los reportes con toda la información de los clientes) contiene un sistema de Captcha, no es posible automatizar la descarga de los reportes, de modo que un encargado del programa realizará la alimentación del archivo de reporte de manera diaria. Teniendo en cuenta lo anterior, los datos son validados de manera automática contra el reporte de OnCredit, en donde habrá dos rutas posibles:
    - Si los registros coinciden de manera correcta con OnCredit, seguirán con el estado "Pendiente" y serán transferidos a un nuevo archivo denominado "Archivo de cargue", el cual se usará posteriormente para cargar los datos en el SAC.
    - Si no coinciden, se etiquetan como "Error" y se almacenan en un archivo separado, llamado "Registros error.xlsx".
- Revisión complementaria (Archivo Crediagil): Este archivo es diligenciado diariamente por otro encargado del programa, por lo que de manera automática se revisará el archivo "Cotejo crediagil.xlsx", el cual está en una carpeta compartida. De dicho archivo se extraen los registros en donde su estado sea "Pendiente", y se agregan al mismo "Archivo de cargue" generado en el paso anterior.
- Carga y asignación mediante procedimientos PLSQL: Cuando ya se tenga listo el archivo definitivo de cargue, se ejecutan dos procedimientos almacenados en la base de datos Oracle del SAC:
    - El primer procedimiento plsql realiza la carga de los registros de contactos al SAC.
    - El segundo procedimiento asigna los productos financieros correspondientes (tarjeta o crediagil) y genera un archivo de errores llamado "Errores_asignacion.xlsx" en caso de inconsistencias.

- Actualización del estado final: Después de la ejecución de los procedimientos, se actualiza de manera automática la columna “Estado” en el archivo “Activaciones Tarjeta.xlsx” a “Cargado”, al igual que en el archivo “Cotejo crediagil.xlsx” se actualiza la columna “Estado” a “Creado ok”.
- Gestión automatizada de errores: Para culminar esta parte de la automatización, se revisa los archivos generados durante el proceso (Registros_error.xlsx, Errores_asignacion.xlsx) y se envía un correo electrónico al encargado, avisándole sobre los errores presentados para su revisión manual.

Figura 5

Diagrama To-Be para el proceso de cargue de transacciones y actualización de saldos



Nota: Figura hecha por el autor.

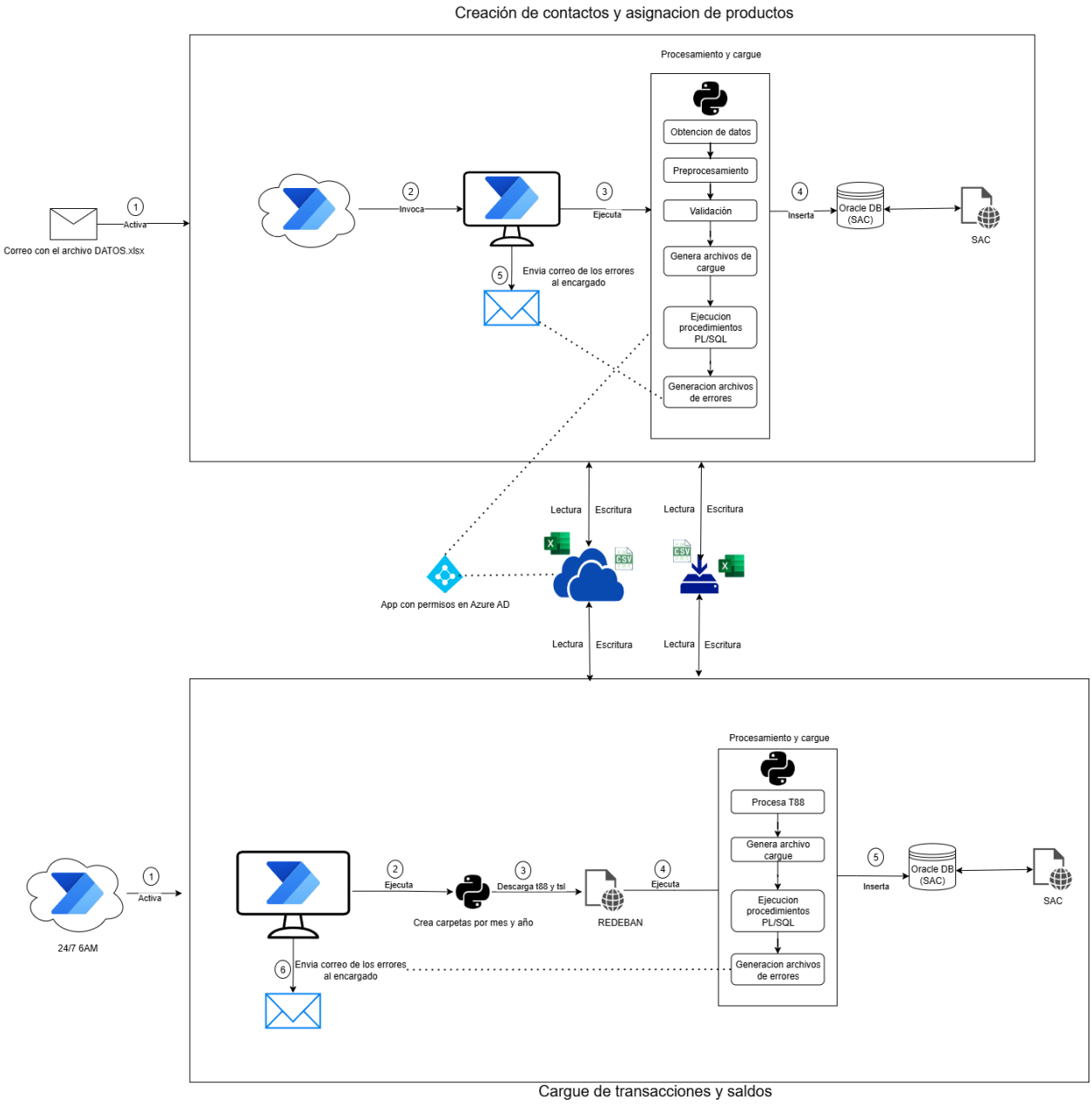
Como se puede observar en la Figura 5, el proceso da inicio de forma automática diariamente a las 6 a.m., esto debido a que en el intervalo de las 3 a 5 a.m. el sistema de Redeban genera los archivos T88 y TSL del día anterior.

- Acceso automatizado al sistema fuente (Redeban): La solución inicia con un bot que automáticamente ingresa al sistema redeban y accede a la ruta especifica ESSAPrepago/Salidas, donde diariamente se generan los archivos necesarios.

- Descarga y almacenamiento automático de archivos: Cuando el bot accede a la ruta indicada, realiza la descarga de los archivos T88 y TSL. Ambos archivos son almacenados automáticamente en las carpetas correspondientes de acuerdo con una estructura previamente configurada.
- Carga automática del archivo T88 al SAC: Una vez almacenado, se ejecuta un procedimiento PL/SQL especializado para cargar el archivo T88 en la base de datos del SAC. Si existen errores durante la carga, se exportarán en un archivo denominado “Errores t88.csv”, de modo que se pueda realizar una revisión.
- Carga automática del archivo TSL al SAC: Una vez se hace el cargue del archivo T88, el flujo continúa automáticamente ejecutando el procedimiento PL/SQL encargado de subir el archivo TSL al SAC.
- Creación y cargue del archivo aliados: Se genera un archivo llamado “Cargue aliados.csv”, el cual es importante ya que posee la información de los aliados comerciales en donde se hicieron las transacciones de cada uno de los usuarios. Este archivo se crea en base al t88, en donde solo se toma la información relevante de dicho archivo. Posteriormente se realiza el cargue al SAC con la ayuda de un procedimiento PL/SQL y genera el archivo “Errores cargue.csv” en caso de que existan errores durante el procedimiento.
- Notificación por correo electrónico: Finalmente, se envía de forma automática un correo electrónico al encargado del proceso. Este correo adjunta los archivos de errores (‘Errores t88.csv’ y ‘Errores cargue.csv’).

Figura 6.

Arquitectura de la automatización.



Nota: Figura realizada por el autor.

En la figura anterior está la arquitectura de la automatización. Está dividido en dos procesos: el primero, que es la creación de contactos y asignación de productos, inicia con la llegada de un correo, en donde dispara un flujo de Power Automate Cloud, el cual inicializa un flujo de Power Automate Desktop cuyo objetivo es ejecutar un script de Python que realiza las

acciones de obtención de datos de diferentes archivos, preprocesamiento de esos datos, validación de los datos, generación de archivo de cargue, ejecución de procedimientos PL/SQL y generación de archivos con registros erróneos. Los procedimientos PL/SQL realizan la inserción a la base de datos del SAC y la información se puede ver reflejada en la página del SAC. Finalmente, el flujo de escritorio realiza el envío de un correo electrónico al encargado, en caso de que se hayan creado archivos con registros erróneos.

El segundo proceso es el cargue de transacciones y actualización de saldos, el cual inicia con un flujo de nube que es disparado todos los días a las 6 a.m., dicho flujo activará un flujo de escritorio que ejecutará un script de Python, el cual creará carpetas por mes y año. Luego de esa ejecución, el flujo de escritorio se dirigirá a la página de Redeban a realizar la descarga de los archivos T88 y TSL y los guardará en las carpetas previamente creadas. Continuando, el flujo ejecutará otro script que procesará el archivo T88, generará un archivo de cargue, ejecutará procedimientos PL/SQL para el T88, TSL y el archivo de cargue, y si surgen errores, generará archivos con los registros con error. Con los procedimientos PL/SQL se subirá la información al SAC, la cual se verá reflejada en la página web. Finalmente, el flujo enviará un correo electrónico en caso de que se hayan creado archivos con registros erróneos.

6.3. Fase de desarrollo

En esta fase se materializó el diseño previamente definido, poniendo en práctica la automatización propuesta. Se desarrollaron y configuraron cada uno de los componentes técnicos establecidos en la anterior fase. A continuación, se describirán de manera detallada las actividades realizadas durante la fase.

6.3.1. Preparación del entorno y herramientas

Como primer paso, se configuró el entorno de desarrollo. Se solicitó la instalación de Power Automate Desktop en la estación de trabajo proporcionada por la ESSA, configurándolo de manera correcta para la ejecución estable de los flujos. Asimismo, se configuraron los siguientes elementos técnicos:

- Se instaló y configuró Python en la versión (3.11), haciendo uso de las siguientes bibliotecas:
    - pandas: Para el manejo eficiente de los archivos Excel, csv, en estructuras tipo Dataframe.
    - openpyxl: lectura y escritura de archivos Excel (xlsx), creando y modificando celdas y estilos, manteniendo el formato directamente desde Python.
    - logging: sistema de logs para generar un histórico de la ejecución y facilitar la depuración.
    - os: realiza operaciones sobre el sistema de archivo y rutas (crea carpetas, comprueba la existencia de archivos, une rutas).
    - datetime: trabaja con fechas y horas (obtiene fecha actual, suma intervalors, formatea tiemstamps).
    - requests: envío de peticiones HTTP (GET, POST, etc.), útil para descargar archivos desde una API o servicio web y subir datos de manera remota.
    - time: funciones de pausa (sleep) y medición de tiempos transcurridos; es ideal para controlar retrasos en llamadas externas o medir performance
    - Fernet (cryptography.fernet): cifra simétricamente los datos; asegura que la información sensible como contraseñas o tokens quede protegida y se pueda

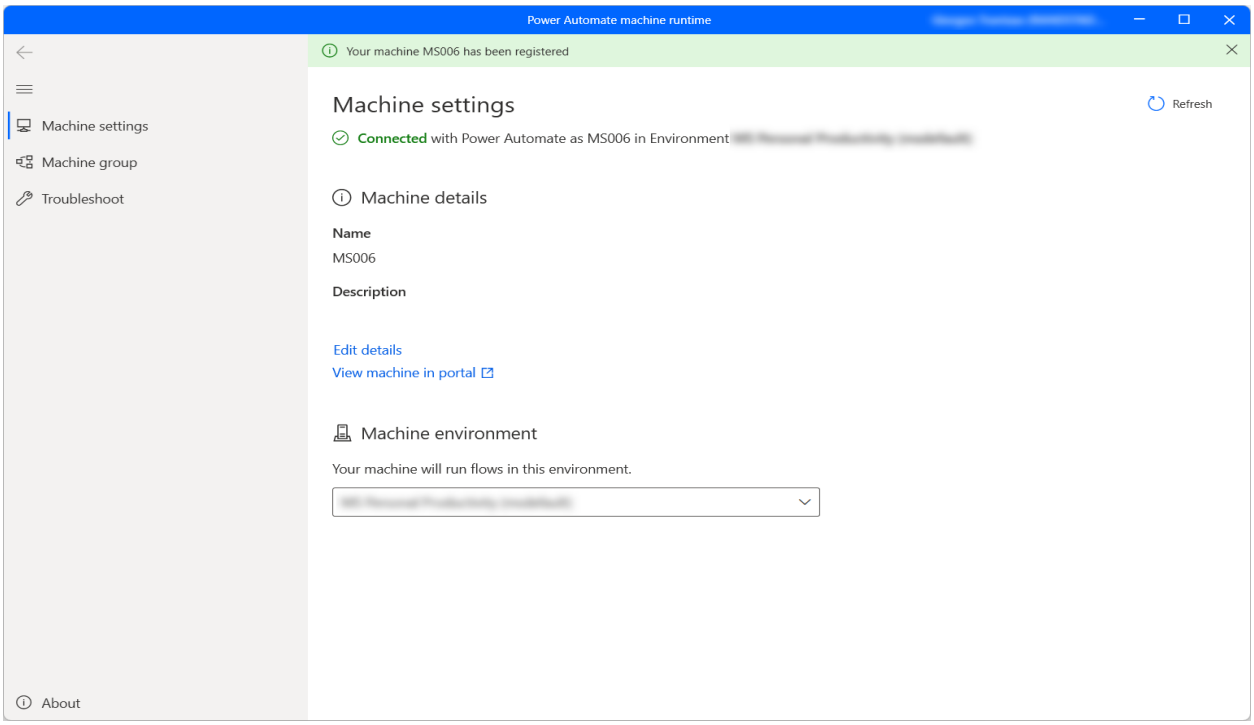
descifrar con la clave correcta.

- oracledb: cliente oficial de Oracle para conectar y ejecutar consultas PL/SQL contra base de datos Oracle.
- sqlalchemy: capa de abstracción de bases de datos (ORM y ejecución de SQL); facilita la construcción de consultas y la gestión de conexiones de forma independiente del motor.
- io: manejo de flujos de datos en memoria (StringIO, BytesIO); permite el procesamiento de archivos o buffers sin necesidad de almacenarlos en disco.
- re: expresiones regulares para búsqueda, validación y reemplazo de patrones de texto (limpiar cadenas o extraer subcadenas especificas).
- msal (Microsoft Authentication Library): gestiona la autenticación y obtención de tokens de acceso contra Azure AD y Microsoft Identity Platform; permite la implementación de flujos de autenticación (cliente confidencial, autorizado por usuario, etc…) para consumir APIs seguras de Microsoft (Graph, SharePoint, OneDrive, etc).

- Se realizó la correcta configuración para que Power Automate Cloud y Power Automate Desktop puedan comunicarse, enlazando la maquina con el mismo entorno que se tiene en power automate cloud.

Figura 7

Conexión de la maquina con el entorno.

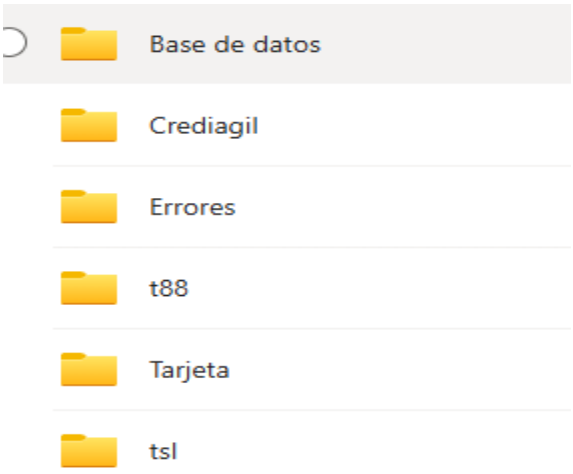


Nota: Imagen sacada de Microsoft. (2025, 21 de junio). *Registered machine* [Captura de pantalla]. Microsoft Learn. *Administrar máquinas - Power Automate | Microsoft Learn*

- Se creó la siguiente estructura de carpetas en el OneDrive compartido con los encargados del proceso.

Figura 8.

Estructura carpetas del OneDrive.



Nota: Captura tomada por el autor.

En la carpeta **Base de datos** está el archivo **Reporte de OnCredit**, que posee todos los datos de los clientes del programa SOMOS; este archivo se actualiza diariamente por un encargado del programa. En la carpeta **Crediagil** está el archivo **Cotejo crediagil**, que posee la información de los clientes que solicitaron el producto financiero Crediágil; este archivo es actualizado por una persona encargada diariamente. En la carpeta **Tarjeta** está el archivo **Activaciones Tarjeta**, que es el archivo madre en donde se guardan los registros de los usuarios que solicitan el producto financiero de tarjeta; el otro archivo que se guarda es el archivo que llega por correo electrónico con la información de los usuarios a registrar en el SAC. En la carpeta de **errores** se almacenarán los archivos CSV que se generarán si existen registros con errores durante el proceso de creación de contactos y asignación de productos financieros (Registros_error.xlsx, Errores_asignacion.xlsx). En la carpeta **t88** se guardan los archivos que se descargan desde Redeban, de la misma manera con la carpeta **tsl**.

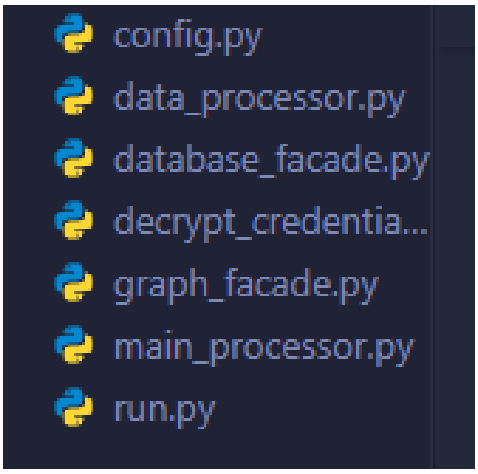
6.3.2. Desarrollo de scripts de Python

El objetivo principal del desarrollo de los scripts en Python es garantizar la calidad de los datos recibidos y facilitar la inserción segura en la base de datos Oracle del SAC.

Con el fin de que el proceso tenga un fácil mantenimiento, se fundamentó en el principio de separación de responsabilidades, en donde cada módulo tiene un propósito claramente definido.

Figura 9

Estructura del proyecto en Python (Proceso creación de contactos y asignación de productos financieros)



Nota: Figura elaborada por el autor. Captura de pantalla del entorno Visual Studio Code.

- config.py: El propósito de este módulo es centralizar todos los parámetros variables y sensibles que se van a usar en la solución para que funcione correctamente. Aquí se almacenan los identificadores para la autenticación con la API de Microsoft Graph y las credenciales de acceso a la base de datos Oracle. Hay que tener en cuenta que este módulo solo define la estructura para recibir dichas credenciales, pero no las almacena en texto plano. También, centraliza todas las rutas de los archivos en OneDrive y en el sistema de archivos local, de este modo se puede modificar la ubicación de un recurso en un único punto sin la necesidad de alterar el código en múltiples lugares.
- decrypt_credentials.py: Este módulo garantiza que las credenciales sensibles, como secretos de API y contraseñas de bases de datos, no se almacenen en texto plano dentro del código. Con la ayuda de la biblioteca cryptography y el algoritmo de Fernet, realiza un cifrado simétrico robusto de un archivo JSON que contiene los datos sensibles. Opera con una clave.key, elemento necesario para el descifrado que debe ser gestionado de manera segura.
- data_processor.py: Este módulo se encarga de la limpieza, validación y transformación de

los datos. Es una caja de herramientas en donde se encapsula toda la lógica repetitiva y compleja asociada al preprocesamiento de los datos.

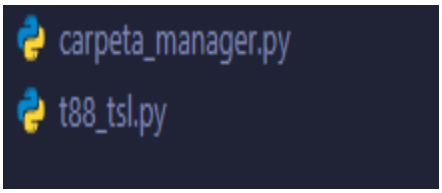
- graph_facade.py: Este módulo gestiona el ciclo de vida del token de autenticación, abstrae las llamadas HTTP para realizar descargas, subidas y eliminar archivos de OneDrive.
- database_facade.py: Este módulo administra la conexión a la base de datos Oracle haciendo uso de SQLAlchemy como ORM para una gestión de conexiones estandarizada y oracledb como el driver de conexión. Durante el desarrollo se encontró un desafío técnico relacionado con la conexión a la base de datos del SAC. La librería oracledb para Python puede operar en dos modos: "Thin Mode" (delgado), que es un modo por defecto, ligero y basado puramente en Python, y el "Thick Mode" (grueso), que requiere la instalación local de las librerías del cliente de Oracle (Oracle Instant Client). Debido a las políticas de seguridad de red y las configuraciones de firewall de la ESSA, los intentos de conexión directa usando el "Thin Mode" eran bloqueados, impidiendo que haya comunicación con la base de datos. Es por eso por lo que, para superar esa restricción, se tomó la decisión de implementar la conexión en "Thick Mode". Con esto, se realizó la instalación del cliente en la máquina de ejecución y la configuración explícita del script de Python para que usara estas librerías nativas.
- run.py: Este módulo es el punto de entrada. Su responsabilidad es inicializar y ensamblar todos los componentes del sistema.
- main_processor.py: En este módulo reside la lógica de negocio y la orquestación del flujo de trabajo completo. Se hace uso de los demás componentes como herramientas para ejecutar el proceso de principio a fin. Su flujo de proceso es el siguiente:
    1. Adquisición: Utiliza GraphFacade para descargar los archivos de entrada (Archivo

Madre, Consolidado, etc.) desde OneDrive.

2. Preprocesamiento: Emplea DataProcessor para limpiar, normalizar y estandarizar los datos leídos en dataframes de pandas.
3. Validación: Se cruza la información de las solicitudes con la base de datos OneCredit para validad la existencia de los registros, además de enriquecerlos con datos importantes como contacto y dirección.
4. Segregación: Separa los registros en dos grupos: válidos (continúan en el proceso) y erróneos (que son reportados en un archivo de errores).
5. Carga de datos: Con los registros válidos, genera un archivo de cargue en formato csv y lo guarda en una ruta local accesible por la base de datos.
6. Ejecución en base de datos: Invoca DatabaseFacade para ejecutar procedimientos PL/SQL que procesa el archivo csv, y actualiza los contactos en el sistema central.
7. Activación de servicios: Ejecuta una serie de procedimientos PL/SQL para realizar la activación de los productos (Tarjeta o Crediagil) para los clientes cuyos datos se cargaron exitosamente.
8. Actualización y reporte: Actualiza los archivos maestros en OneDrive (Archivo Madre, archivo Crediagil) con el estado final de cada solicitud (“Cargado”, “Error”) y sube los reportes de errores generados.

Figura 10

Estructura del proyecto en Python (Proceso cargue de transacciones y actualización de saldos).



Nota: Figura elaborada por el autor. Captura de pantalla del entorno Visual Studio Code.

Para el proceso de cargue de transacciones y actualización de saldos se usarán dos módulos.

- carpeta_manager.py: Este módulo se encarga de la creación de carpetas por año y mes dentro de la ruta definida. Así se facilita la organización de los archivos T88.
- t88_tsl.py: Este módulo localiza los archivos del día anterior (T88), extrae y formatea sus datos, genera los archivos de cargue de aliados y ejecuta procedimientos PL/SQL para el cargue en el SAC. Para el caso del archivo TSL, solo se realiza el cargue al SAC con la ejecución del procedimiento PL/SQL.

6.3.3. Ejecución de procedimientos PL/SQL

Como ya se ha mencionado, en este proyecto se implementaron algunos procedimientos PL/SQL para realizar el cargue de la información directamente a la base de datos. Se decidió realizar de este modo, debido a que hacerlo por medio de la interfaz gráfica del SAC con ayuda de Power Automate Desktop a futuro no es adecuado a nivel de soporte, ya que el SAC, al estar en constante actualización en su interfaz gráfica, los selectores de Power Automate Desktop fallarían cada vez que se realicen los cambios.

Se diseñaron las especificaciones de los procedimientos PL/SQL, los cuales fueron implementados por el equipo de TI de ESSA, para garantizar la integridad y seguridad de la base de datos. Se realizaron los siguientes tres procedimientos PL/SQL:

- Procedimiento PL/SQL 1: Carga de archivos.

  Este es un bloque PL/SQL anónimo el cual delega a la base de datos la tarea de leer un

archivo plano del sistema de archivos y procesar los registros. Este procedimiento posee un parámetro el cual es modificable de acuerdo con el proceso que se requiera realizar. En el caso de este proceso se usa para: cargar el archivo plano con la información de los contactos, cargar el archivo plano con la información del T88, cargar el archivo plano con la información del TSL y cargar el archivo plano con la información de los aliados.

- Procedimiento PL/SQL 2: Activación de cliente.

  Activa el cliente tomando todos los datos de contacto que el script de Python recopila y los inserta en las tablas principales del sistema de financiación social, esto se hace ya que antes de activar el cliente, el estado es "Cliente potencial", que una vez activado pasa a ser "Cliente".

- Procedimiento PL/SQL 3: Asignación de productos.

  Asigna el producto financiero correspondiente. Una vez que el cliente está activado, este procedimiento crea el registro del producto asociado (Tarjeta o crediagil). Toma como parámetros la cedula del cliente, el tipo de producto ('R' para tarjeta, 'S' para crediagil), el numero de la tarjeta (en caso de ser crediagil se pasa la cedula), y el monto.

6.3.4. Desarrollo de flujos en Power Automate

Una vez listo el entorno y los scripts desarrollados, se hicieron los flujos automatizados que permiten la orquestación y ejecución de los procesos del programa SOMOS. Para lograrlo, se usaron dos tipos de flujos: Flujos de Nube y Flujos de Escritorio.

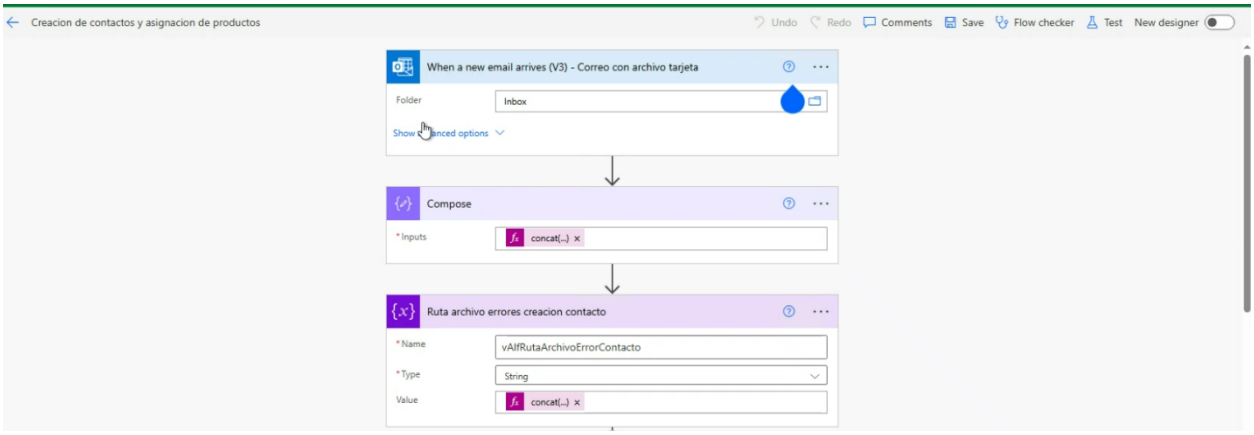
Flujos de nube:

Los flujos de nube se encargan de la orquestación general de las tareas, tales como recibir los correos electrónicos, desencadenar flujos de escritorio y el control de ejecución diaria programada. Para este proyecto se hicieron dos flujos de nube: un flujo para el proceso relacionado

a la creación de contactos y activación de productos financieros; y el otro flujo para el proceso de cargue de transacciones y actualización de saldos.

Figura 11

Flujo de nube 1. Creación de contactos y activación de productos financieros.

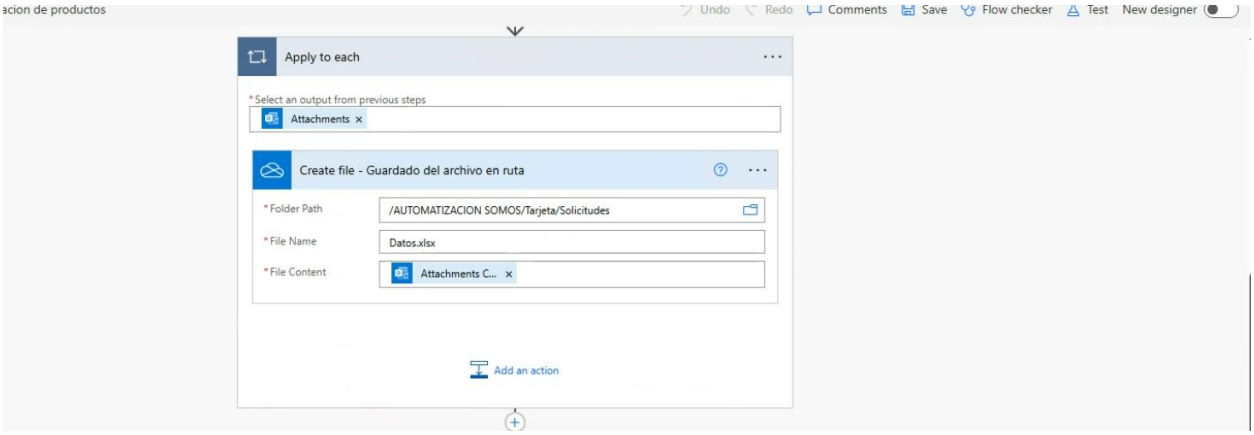


Nota: Figura elaborada por el autor. Captura de pantalla #1 del flujo de nube 1.

Como se muestra en la Figura 11, el proceso comienza con un evento disparador de correo electrónico que monitorea continuamente una bandeja de entrada específica. El disparador está configurado para activarse únicamente cuando se recibe un correo con un asunto predefinido, garantizando así que solo las solicitudes válidas inicien el proceso. Posteriormente, se definen variables dinámicas para especificar las ubicaciones de almacenamiento de los archivos de error, permitiendo un esquema flexible de nomenclatura basado en la fecha de ejecución.

Figura 12

Flujo de nube 1. Creación de contactos y activación de productos financieros.

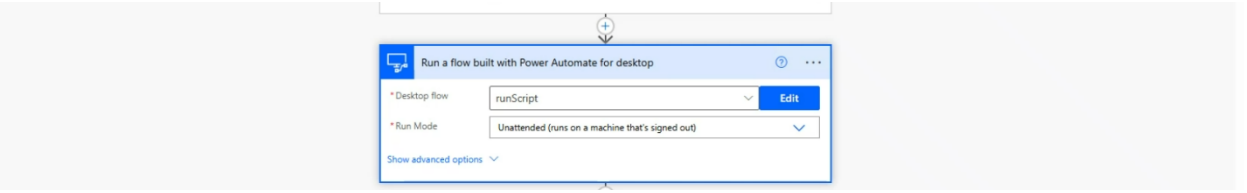


Nota: Figura elaborada por el autor. Captura de pantalla #2 del flujo de nube 1.

En la Figura 12, el flujo realiza el *staging* de los datos. Toma el archivo Excel adjunto del correo entrante y lo inserta en un repositorio central en OneDrive con un nombre estandarizado asignado ("Datos.xlsx"). Esta es una etapa muy importante porque separa la ingesta de datos de su procesamiento posterior, preparando así el archivo para ser procesado por el siguiente componente de la arquitectura.

Figura 13

Flujo de nube 1. Creación de contactos y activación de productos financieros

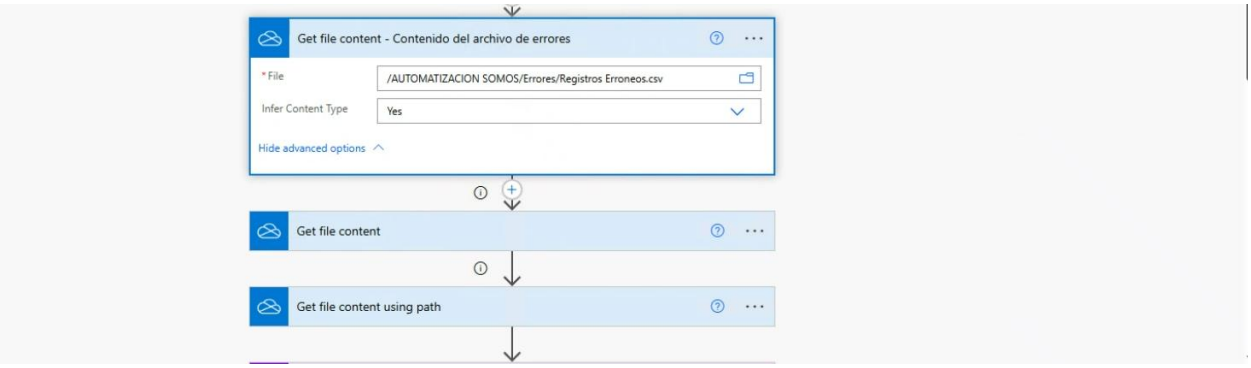


Nota: Figura elaborada por el autor. Captura de pantalla #3 del flujo de nube 1.

La Figura 13 describe otro componente importante de la arquitectura híbrida: la delegación del procesamiento. El flujo en la nube invoca al flujo de escritorio llamado *runScript* en modo desatendido, transfiriendo así el control a la máquina local. Allí se ejecuta un script de Python que encapsula lógica de negocio compleja que incluye validación, transformación y carga a la base de datos.

Figura 14

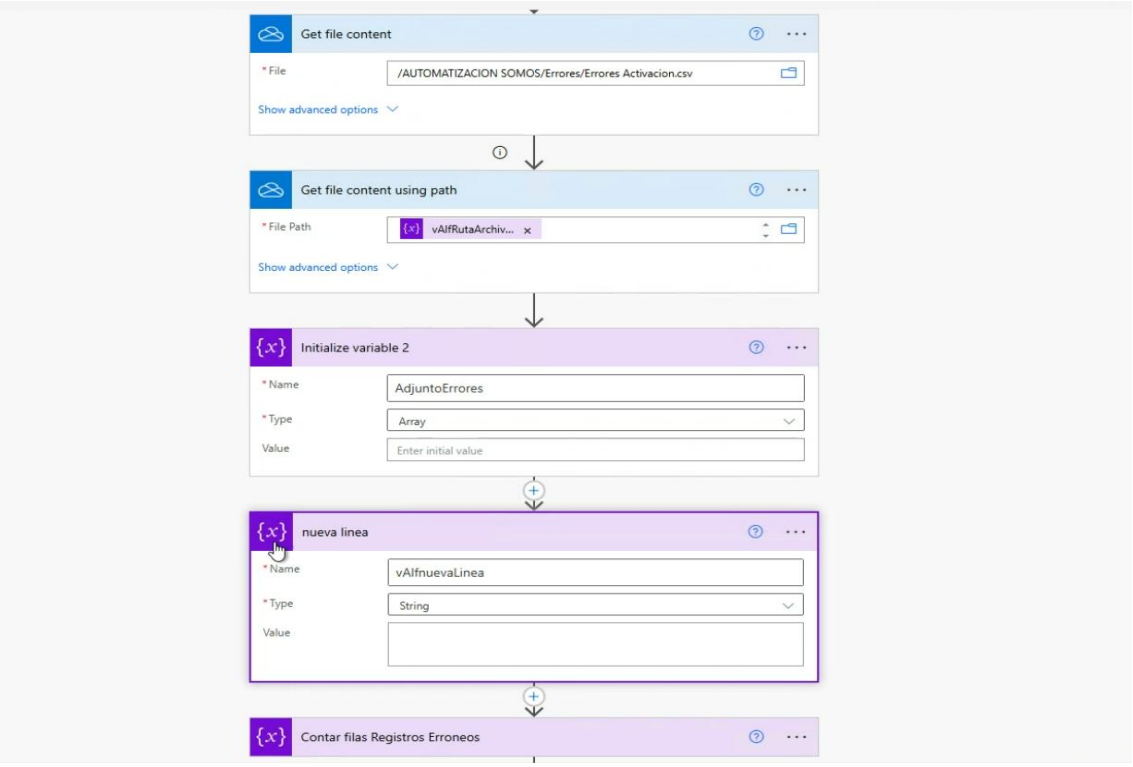
Flujo de nube 1. Creación de contactos y activación de productos financieros.



Nota: Figura elaborada por el autor. Captura de pantalla #4 del flujo de nube 1.

Figura 15

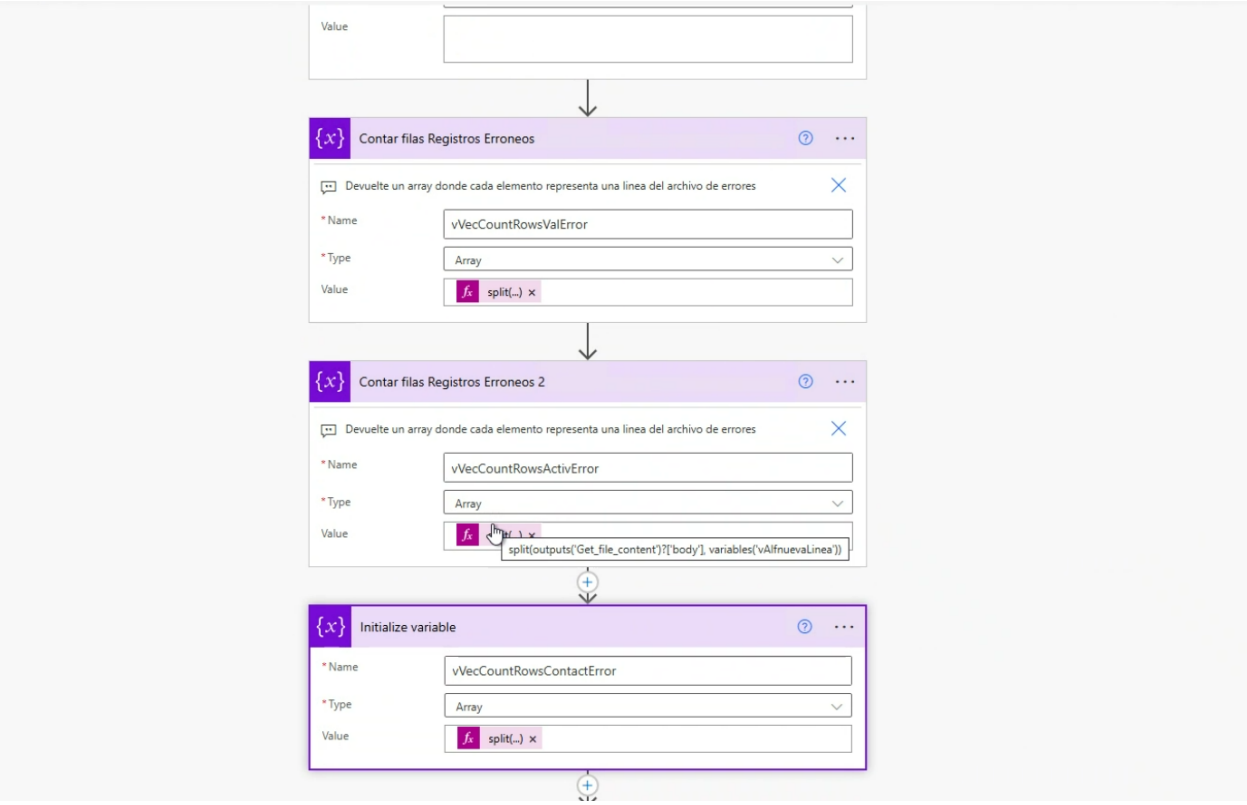
Flujo de nube 1. Creación de contactos y activación de productos financieros.



Nota: Figura elaborada por el autor. Captura de pantalla #5 del flujo de nube 1.

Figura 16

Flujo de nube 1. Creación de contactos y activación de productos financieros.

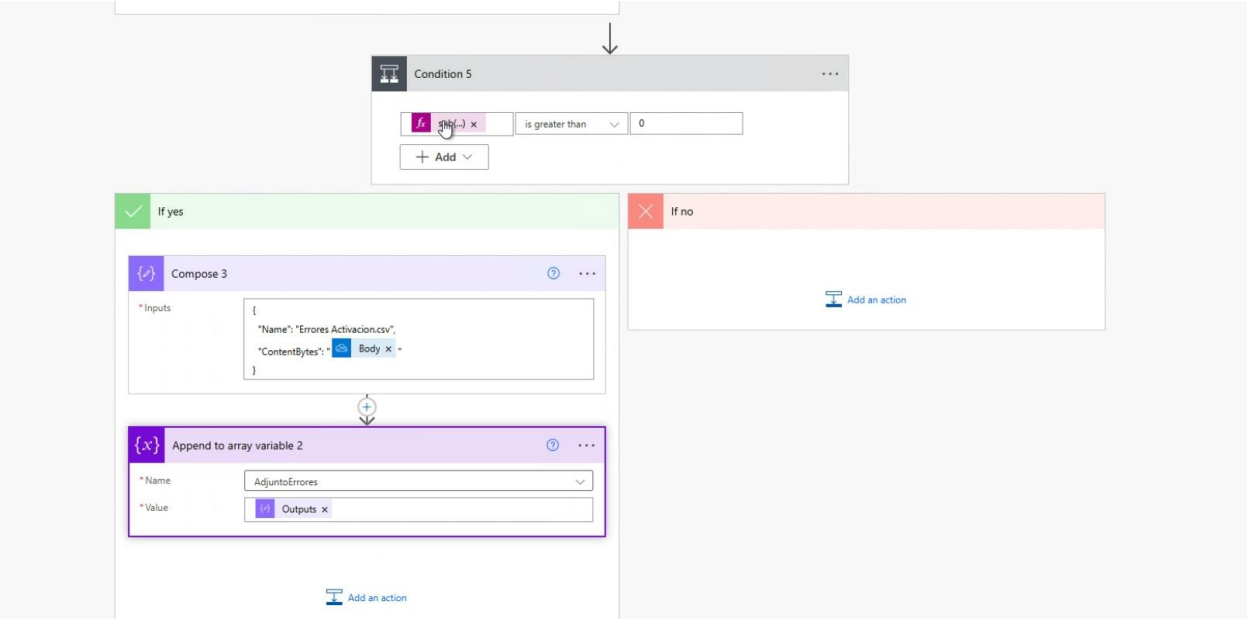


Nota: Figura elaborada por el autor. Captura de pantalla #6 del flujo de nube 1.

Posteriormente, cuando el flujo de escritorio finaliza, el control regresa nuevamente al flujo en la nube para la gestión de errores. Como se indica en las figuras 14, 15 y 16, el flujo recupera el contenido de los posibles archivos de error generados por el script de Python. Utiliza variables y operaciones de texto para analizar dicho contenido y determinar si se reportaron anomalías durante la ejecución.

Figura 17

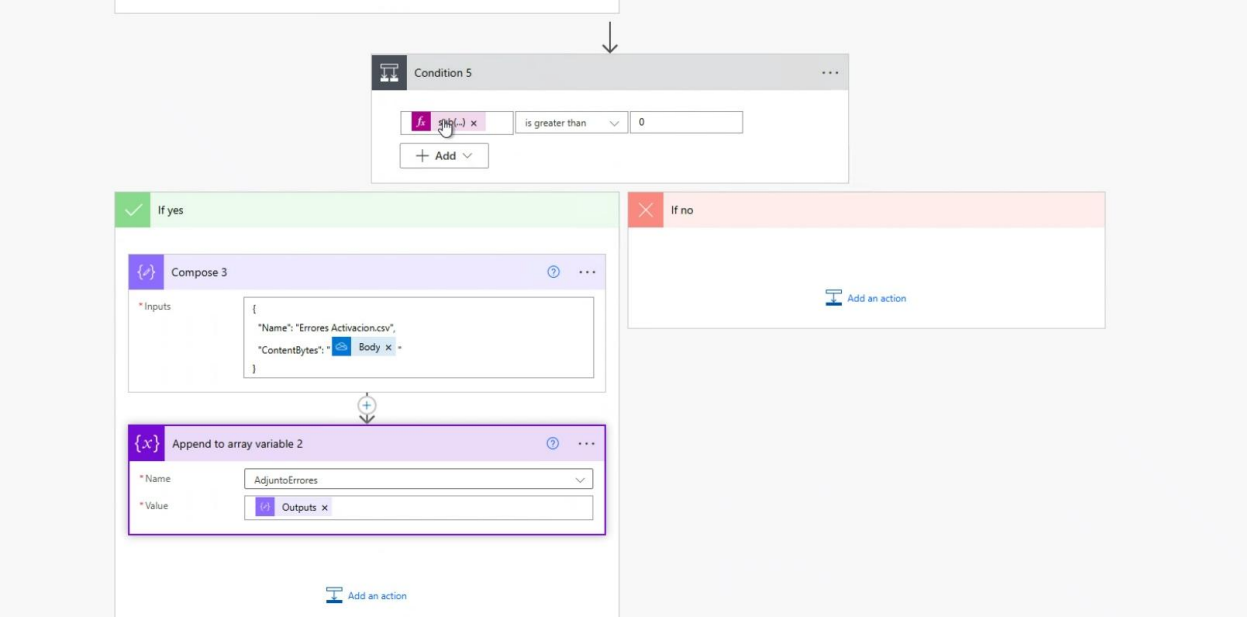
Flujo de nube 1. Creación de contactos y activación de productos financieros.



Nota: Figura elaborada por el autor. Captura de pantalla #7 del flujo de nube 1.

Figura 18

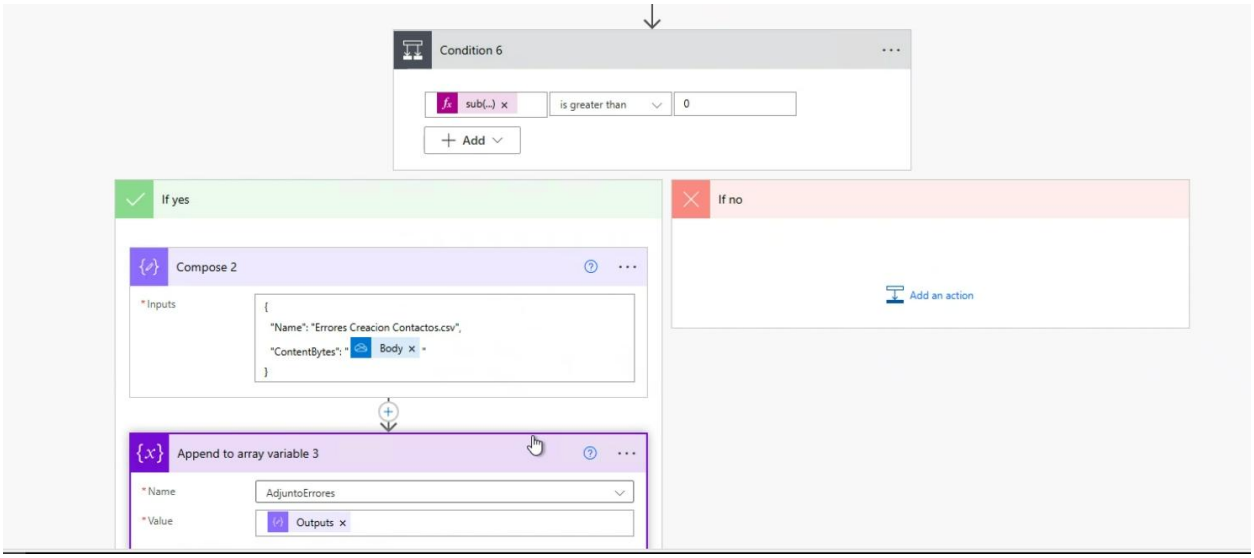
Flujo de nube 1. Creación de contactos y activación de productos financieros.



Nota: Figura elaborada por el autor. Captura de pantalla #8 del flujo de nube 1.

Figura 19

Flujo de nube 1. Creación de contactos y activación de productos financieros.

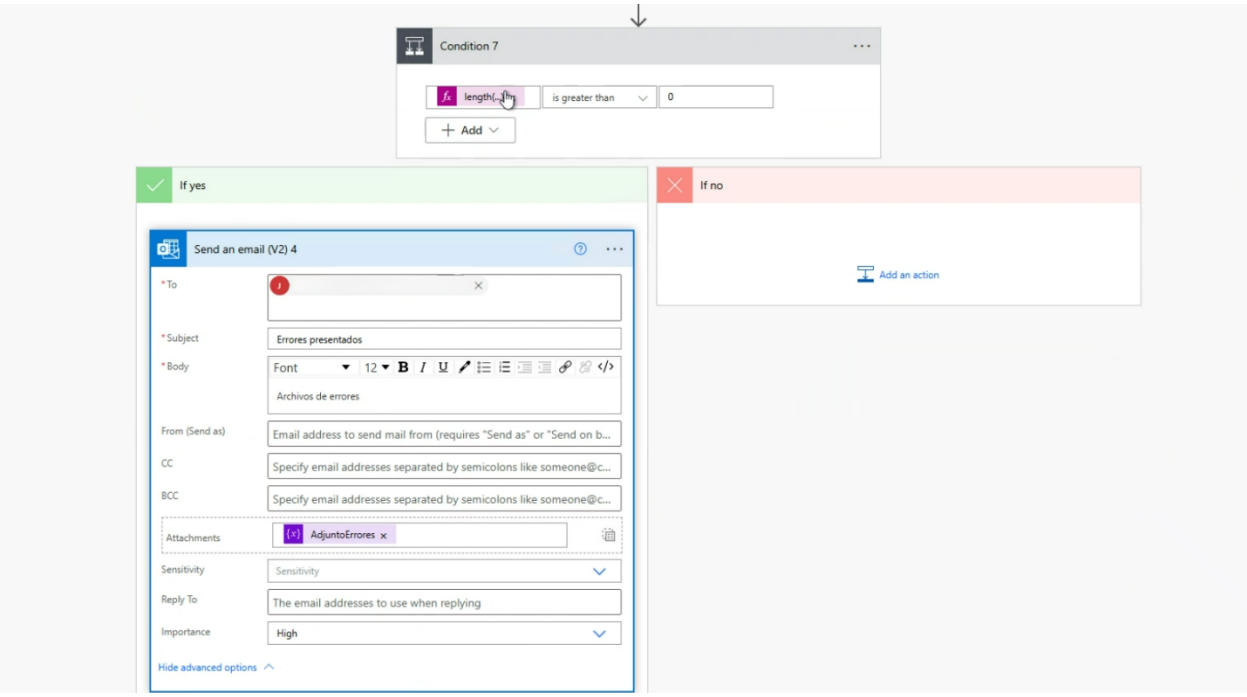


Nota: Figura elaborada por el autor. Captura de pantalla #9 del flujo de nube 1.

En las Figuras 17, 18 y 19, los reportes de error son consolidados en el flujo. Además, se verifica la presencia de registros en cada archivo de error mediante una serie de condiciones. En caso afirmativo, se recopila el nombre del archivo y su contenido en un Array de adjuntos.

Figura 20

Flujo de nube 1. Creación de contactos y activación de productos financieros.

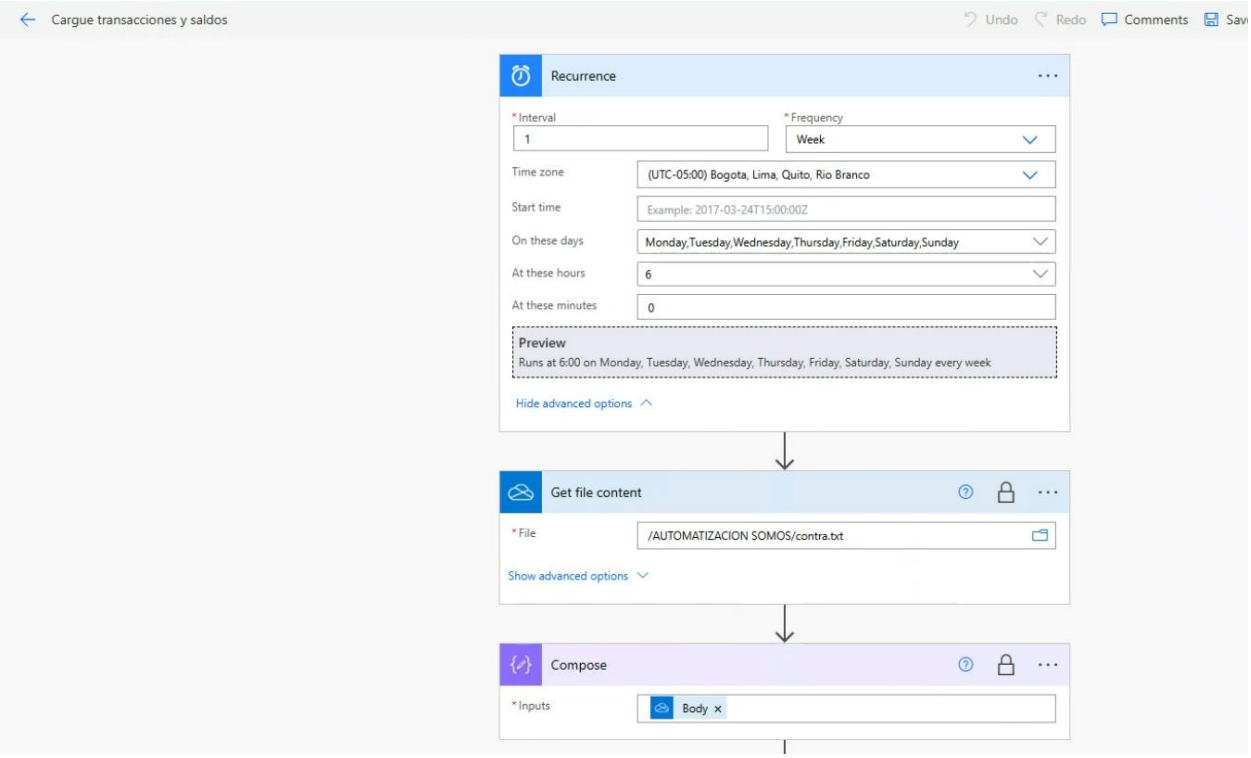


Nota: Figura elaborada por el autor. Captura de pantalla #10 del flujo de nube 1.

En la fase final (Figura 20), el flujo implementa una notificación condicional. Verifica la existencia de elementos en el Array de adjuntos, y solo en ese caso se envía un correo electrónico al usuario responsable con todos los reportes de error consolidados. Este mecanismo asegura que los usuarios solo sean notificados cuando sea necesario intervenir manualmente.

Figura 21

Flujo de nube 2. Cargue de transacciones y actualización de saldos.

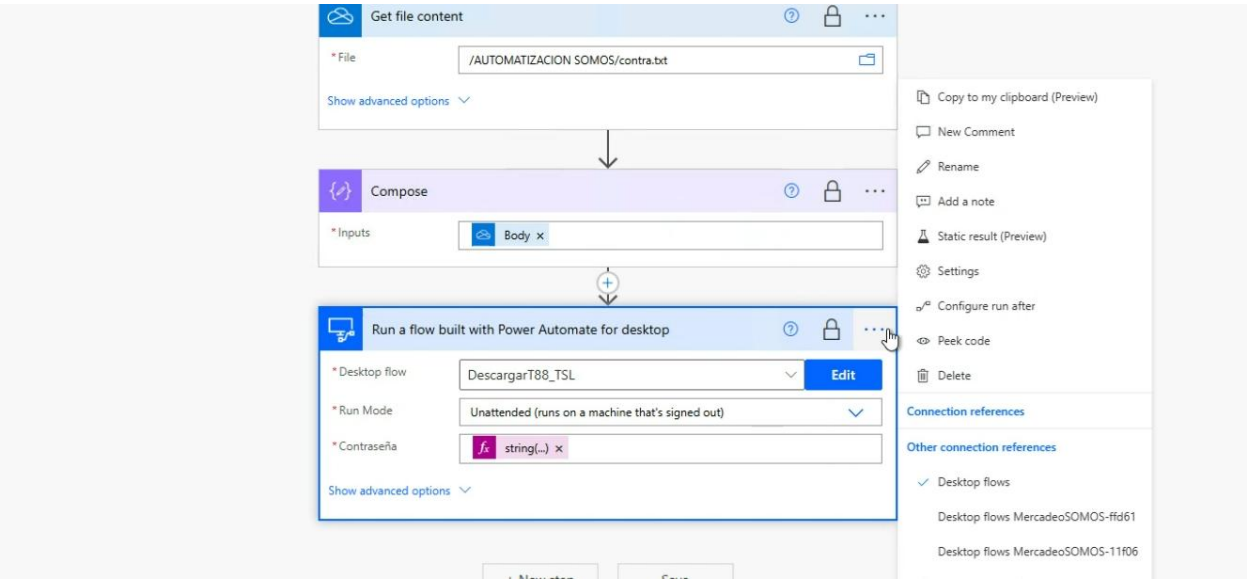


Nota: Figura elaborada por el autor. Captura de pantalla #1 del flujo de nube 2.

El proceso (Figura 21) está programado para iniciarse diariamente a las 6:00 a.m. mediante un disparador de recurrencia. Para garantizar una gestión segura de las credenciales, la contraseña del portal Redeban se recupera en el flujo desde un archivo de texto restringido en OneDrive, evitando así almacenarla directamente en el flujo.

Figura 22

Flujo de nube 2. Cargue de transacciones y actualización de saldos.



Nota: Figura elaborada por el autor. Captura de pantalla #2 del flujo de nube 2.

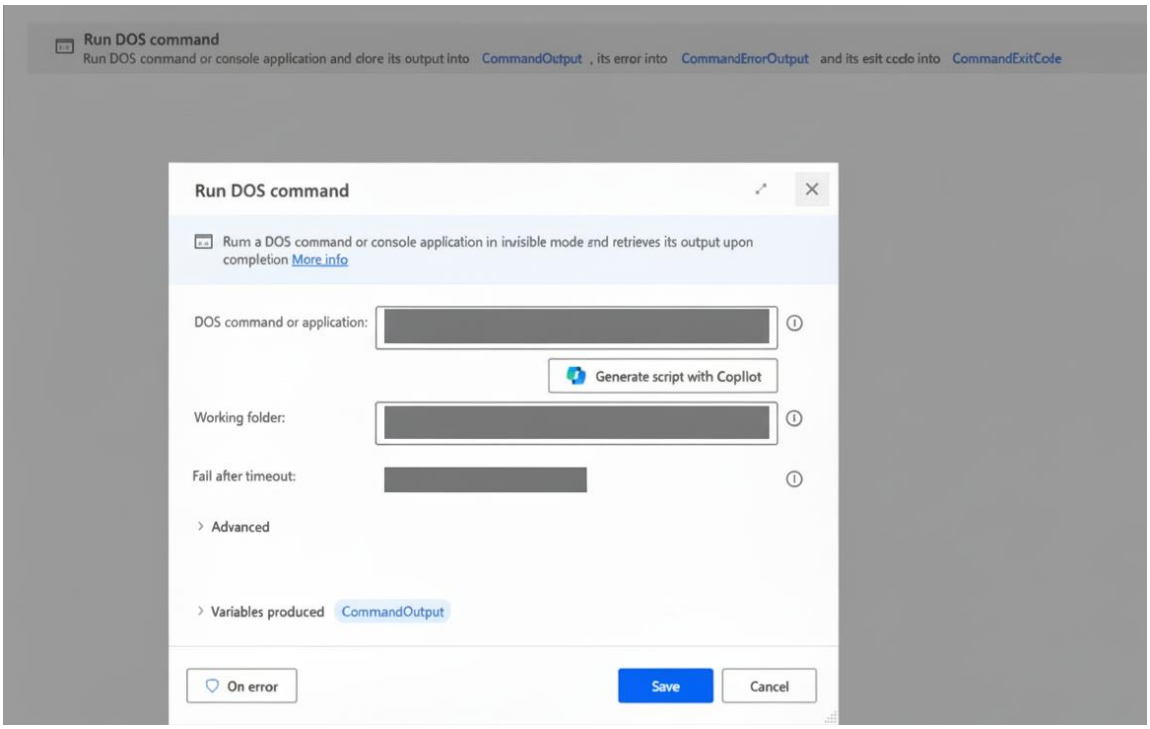
El flujo en la nube mostrado en la Figura 22 invoca al flujo de escritorio *DescargarT88_TSL* en modo desatendido. Transfiere de manera segura la contraseña recuperada como un parámetro de entrada, delegando así la ejecución de todas las tareas locales y de interacción web.

Flujos de escritorio:

Los flujos de escritorio fueron desarrollados para ejecutar acciones de manera local, tales como ejecutar scripts de Python e interactuar con el sistema Redeban. De la misma manera que con los flujos de nube, se realizaron dos flujos de escritorio: un flujo para el proceso de creación de contactos y asignación de productos financieros, y un flujo para el proceso de cargue de transacciones y actualización de saldos.

Figura 23

Flujo de escritorio 1. Creación de contactos y asignación de productos financieros.

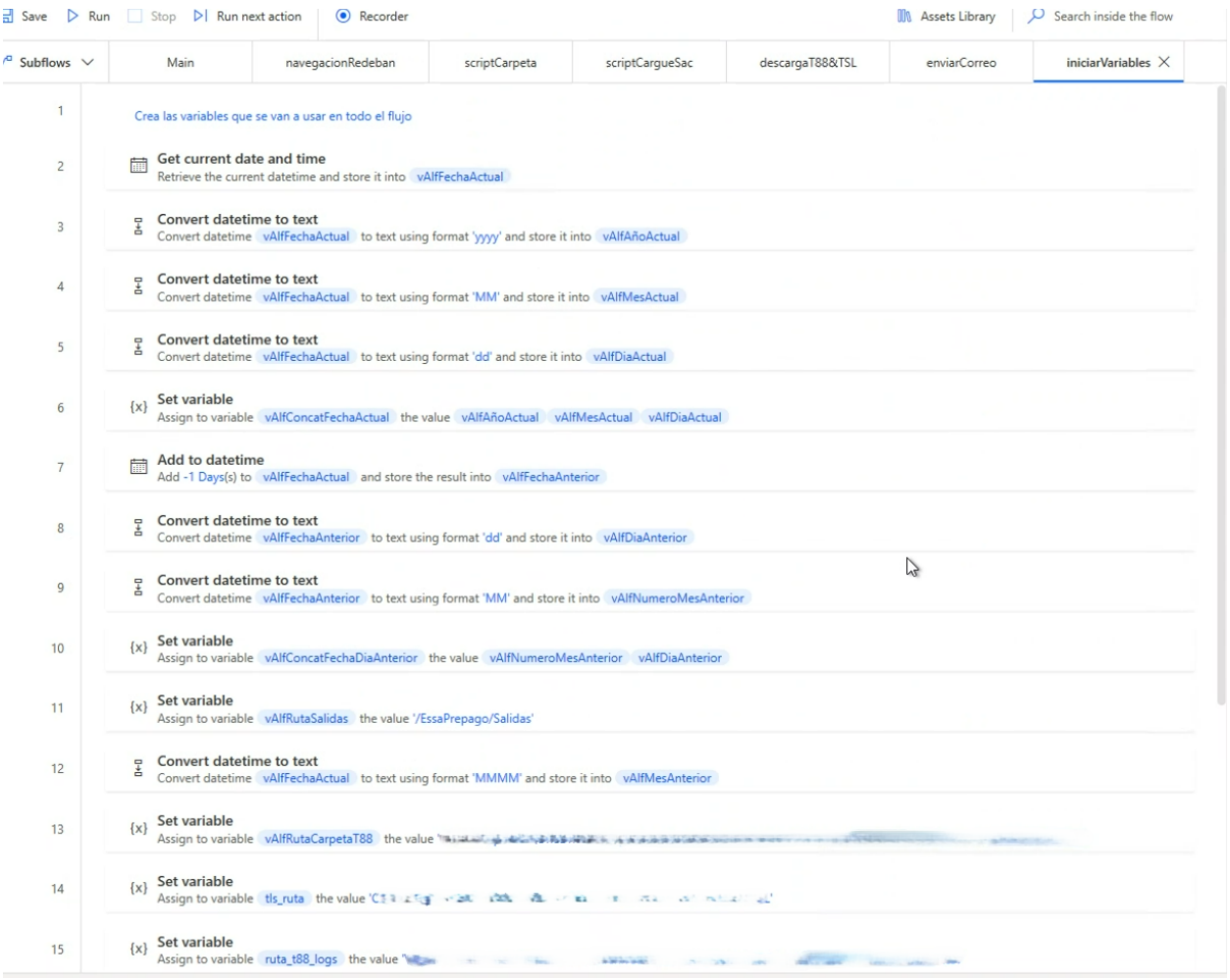


Nota: Figura elaborada por el autor. Captura de pantalla #1 del flujo de escritorio 1.

Este flujo (Figura 23) tiene un único propósito: actuar como un contenedor (*wrapper*) para el script de Python. Mediante la acción *Run DOS command*, ejecuta el script *run.py*, que contiene toda la lógica de negocio. Esta arquitectura permite desacoplar la orquestación (Power Automate) de la lógica de procesamiento (Python).

Figura 24

Flujo de escritorio 2. Cargue de transacciones y actualización de saldos.



Nota: Figura elaborada por el autor. Captura de pantalla #1 del flujo de escritorio 2.

Figura 25

Flujo de escritorio 2. Cargue de transacciones y actualización de saldos.

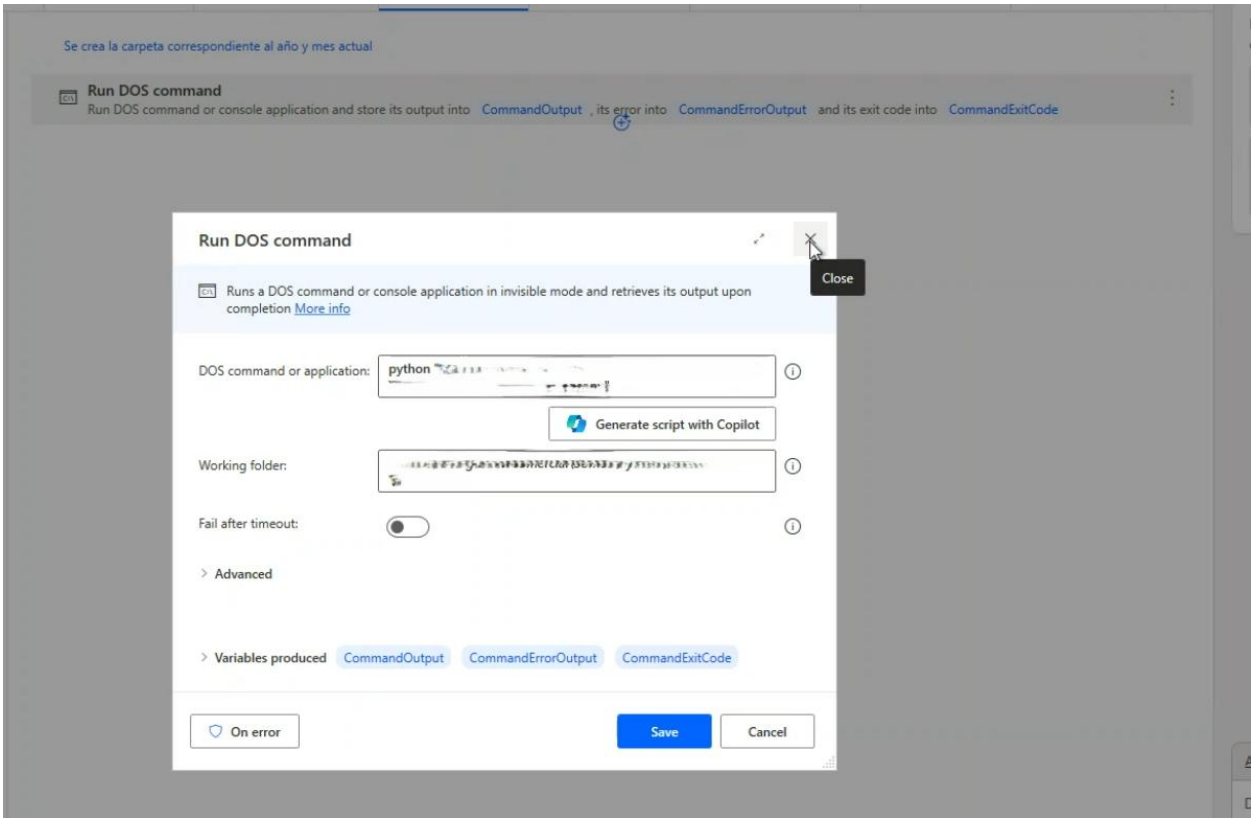
Nota: Figura elaborada por el autor. Captura de pantalla #2 del flujo de escritorio 2.

El flujo comienza inicializando su entorno (subflujo "IniciarVariables"). Como se muestra

en las figuras, calcula las fechas de operación (día actual y anterior) y establece las rutas de los archivos requeridos. Esta preparación garantiza que el bot disponga de toda la información contextual necesaria antes de iniciar las interacciones.

Figura 26

Flujo de escritorio 2. Cargue de transacciones y actualización de saldos.

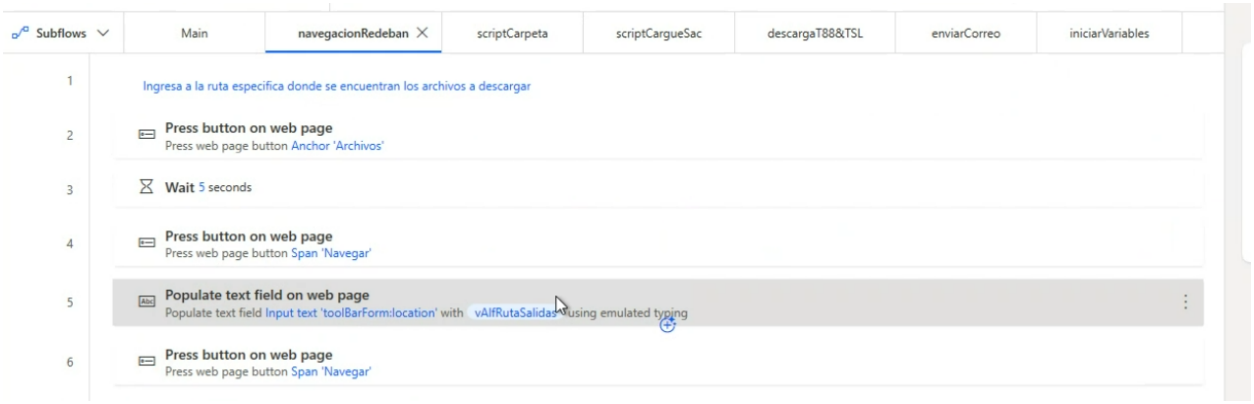


Nota: Figura elaborada por el autor. Captura de pantalla #3 del flujo de escritorio 2.

A continuación, el flujo (Figura 26) ejecuta un script auxiliar de Python (*carpeta_manager.py*) para asegurar que exista la estructura de carpetas de destino, organizada por año y mes.

Figura 27

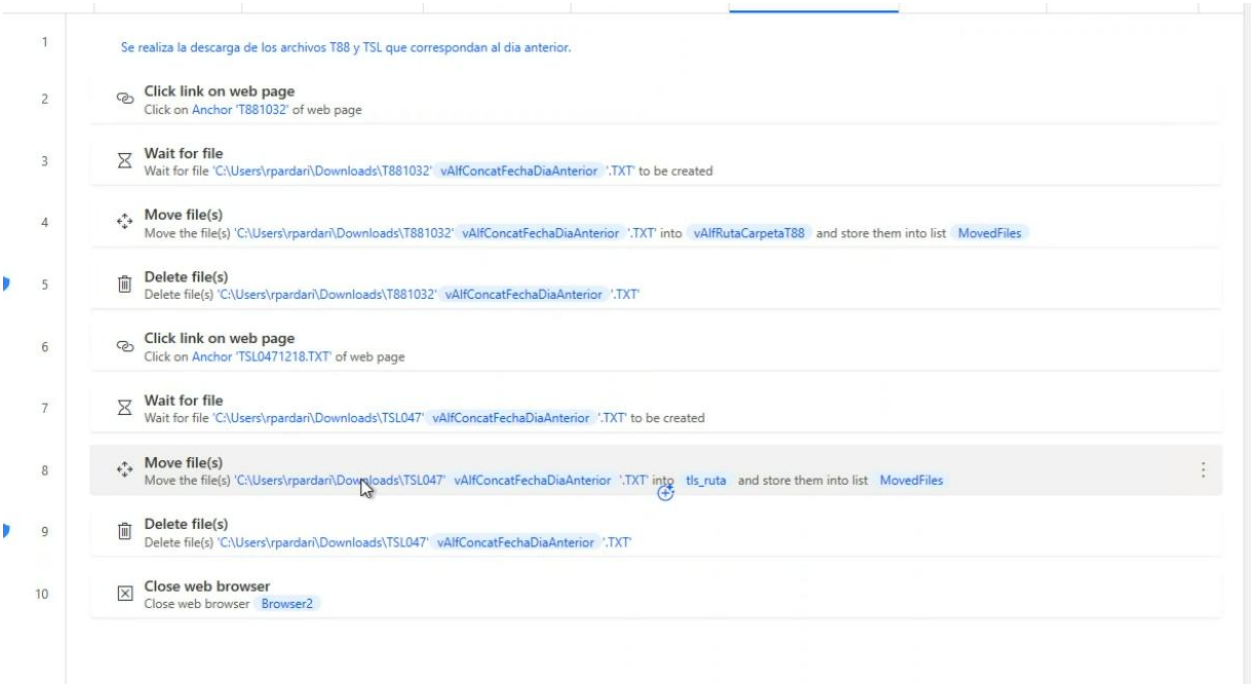
Flujo de escritorio 2. Cargue de transacciones y actualización de saldos.



Nota: Figura elaborada por el autor. Captura de pantalla #4 del flujo de escritorio 2.

Figura 28

Flujo de escritorio 2. Cargue de transacciones y actualización de saldos.



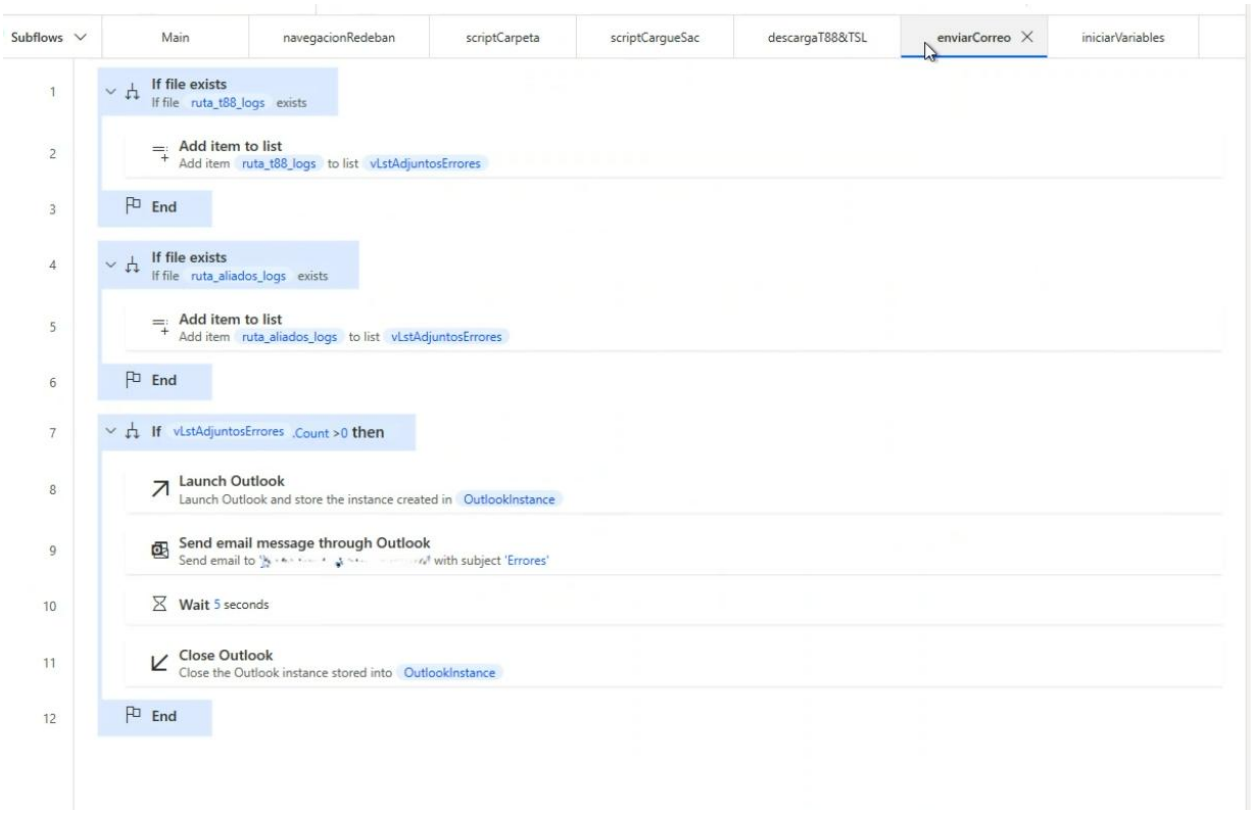
Nota: Figura elaborada por el autor. Captura de pantalla #5 del flujo de escritorio 2.

La fase de interacción con la interfaz de usuario se muestra en las Figuras 27 y 28. El bot abre un navegador, se autentica en el portal de Redeban (subflujo "navegacionRedeban") y accede a la sección de reportes. Allí identifica dinámicamente los archivos T88 y TSL correspondientes

al día anterior, los descarga y los organiza en sus carpetas de destino (subflujo "descargaT88&TSL").

Figura 29

Flujo de escritorio 2. Cargue de transacciones y actualización de saldos.

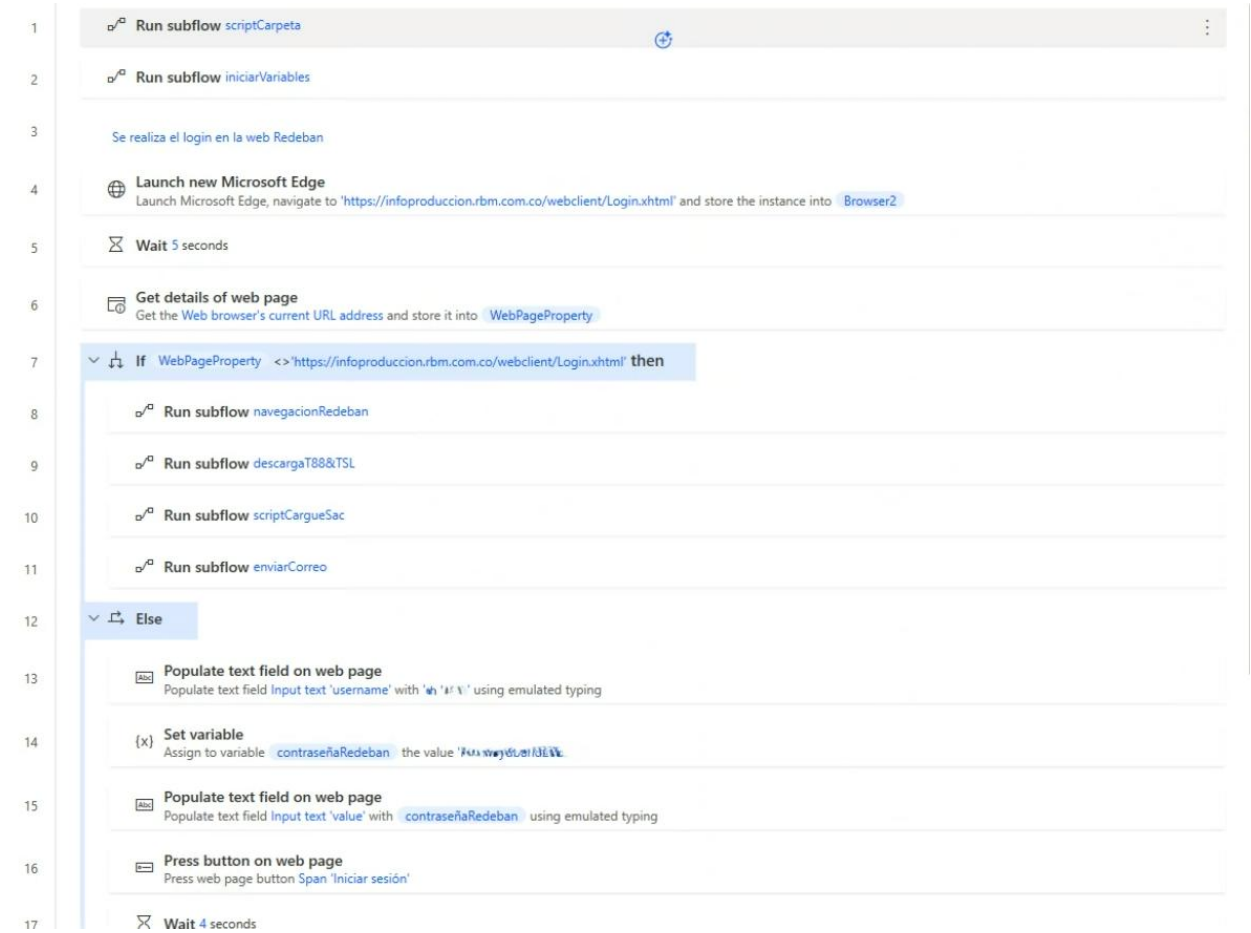


Nota: Figura elaborada por el autor. Captura de pantalla #6 del flujo de escritorio 2.

Cuando los archivos fuente están disponibles, el flujo delega nuevamente el procesamiento intensivo de datos ejecutando el script principal *t88_tsl.py*. Finalmente (Figura 29), el flujo gestiona la notificación de errores a nivel local. Evalúa si se generaron registros de error y, en caso afirmativo, automatiza el cliente de Outlook para enviar una notificación al equipo correspondiente (subflujo "enviarCorreo"). La Figura 30 muestra la ilustración de todos estos subflujos encadenados en el flujo principal.

Figura 30

Flujo de escritorio 2. Cargue de transacciones y actualización de saldos.



Nota: Figura elaborada por el autor. Captura de pantalla #7 del flujo de escritorio 2.

En la Figura 30 se puede observar el flujo final, que condensa todos los subflujos vistos previamente. En resumen, el flujo realiza lo siguiente:

1. Crea las carpetas por mes para los archivos T88 (Script de python).
2. Inicializa las variables (fechas, rutas, listas).
3. Verifica si ya se está logeado en Redeban, si no lo está se logea.
4. Navega hasta encontrar los archivos dentro del portal Redeban.
5. Descarga los archivos T88 y TSL correspondientes al día anterior, mueve los archivos a

las carpetas de destino y elimina las descargas temporales.

6. Realiza el cargue de la información al SAC ejecutando el script de python.
7. Verifica si se crearon archivos de logs con errores, y si los hay se envía un correo automáticamente con los adjuntos.

6.4. Fase de pruebas

Para garantizar que los flujos de automatización y los scripts cumplieran con los requerimientos funcionales y técnicos ya establecidos, se implementó una estrategia de pruebas multifásica que abarca desde la validación individual de los componentes hasta la verificación integral del flujo de trabajo por parte de los usuarios finales. Esta es una fase importante para identificar y corregir errores de manera temprana, asegurando la correcta interacción entre las distintas tecnologías.

6.4.1. Pruebas unitarias

El objetivo de las pruebas unitarias es verificar que cada componente de la solución funcione de manera correcta y aislada, permitiendo depurar la lógica interna de cada artefacto antes de ensamblarlos.

Validación de scripts de Python:

- Cada módulo de python (data_processor.py, database_facade.py, etc.) fue probado de forma independiente.
- Se hizo uso de archivos Excel y csv de muestra con distintos escenarios: datos correctos, datos incompletos (campos vacíos), formatos incorrectos (cedulas con letras, correos sin arroba, acentos, etc)

- Se verificó que los scripts realizaran correctamente las tareas de limpieza, transformación y validación de datos. De igual forma se confirmó la correcta generación de archivos de salida (archivos de cargue y reportes de errores) y que la conexión a la base de datos ORACLE fuera exitosa y con los permisos adecuados para la ejecución de los procedimientos PL/SQL.

Validación de flujos de Power Automate:

- Flujos de nube: Se hicieron ejecuciones manuales para probar la lógica de orquestación. Por ejemplo, para el flujo de creación de contactos, se enviaban correos de prueba a la cuenta designada para confirmar que el trigger se activaba de manera correcta, que el archivo adjunto era almacenado en la ruta de OneDrive especificada y que la llamada al flujo se iniciara sin errores.

- Flujos de escritorio: Cada flujo de escritorio se ejecutó directamente desde power automate desktop, validando que las acciones individuales, como la ejecución de comandos de DOS para los scripts de python, o la navegación y descarga de archivos en el portal Redeban, se completaran exitosamente.

6.4.2. Pruebas de integración

Cuando los componentes ya se validaron individualmente, las pruebas de integración se enfocaron en asegurar que la comunicación entre Power Automate y los scripts de Python sea correcta, simulando el flujo de trabajo completo.

Integración Power Automate Cloud – Desktop – Python:

Se ejecutaron los procesos de extremo a extremo, iniciando con el envío de un correo

electrónico para el proceso de creación de contactos y asignación de productos, y fijando un horario para el proceso de cargue de transacciones y actualización de saldos. Se validaron los siguientes puntos críticos, a modo de verificar que la cadena de ejecución no se rompiera:

- La correcta invocación del flujo de escritorio desde el flujo de nube, verificando que la conexión en la acción “Run a Flow built with power automate for desktop” corresponda a la de la maquina en donde se tiene el flujo de escritorio.
- Que el flujo de escritorio ejecute scripts de Python sin conflictos de permisos o de entorno. Debido a que la acción de ejecutar scripts de Python integrada en Power Automate Desktop a fecha de la realización del proyecto solo soporta versiones de Python 2.x y que por parte del equipo de TI tiene prohibido la ejecución de scripts por medio de PowerShell, se optó por ejecutar los scripts mediante la opción “Run DOS command”
- La habilidad del script de Python para lectura y escritura de archivos en las ubicaciones de OneDrive que power automate utiliza como intermediario.
- El retorno de la ejecución al flujo de nube para las acciones posteriores, como la recopilación de archivos de error y el envío del correo de notificación.

Integración con sistemas externos:

Se hizo la prueba de la interacción completa del sistema con las plataformas externas: la conexión a la base de datos Oracle del SAC y la autenticación y descarga de archivos desde el portal de Redeban, a modo de asegurar que los permisos y las credenciales configuradas eran suficientes y funcionaban en un entorno controlado.

6.4.3. Pruebas de aceptación del usuario

La fase final de pruebas consistió en la validación de la solución por parte de los responsables operativos del programa “SOMOS”. Estas pruebas se hicieron en un entorno de preproducción, usando datos reales para simular las condiciones de operación del día a día.

Escenarios de prueba:

Los usuarios prepararon un lote de casos de prueba con datos reales de clientes, incluyendo:

- Registros válidos para la creación de contactos y asignación de productos (“CrediSomos Tarjeta” y “CrediSomos Agil”).
- Registros con errores conocidos (ej. cédulas ya existentes, información incompleta, nombres con acentos, etc.) de modo que se verificara que el sistema los identificara y los reportara correctamente.
- El proceso completo de descarga y carga de los archivos T88 y TSL desde Redeban.

Proceso de validación:

Los usuarios ejecutaron el proceso de la misma forma en que lo harían en su día a día, o sea, enviando el correo con el archivo de validaciones para el proceso de creación de contactos y asignación de productos, y esperar hasta la madrugada para el proceso de cargue de transacciones y asignación de saldos. Posteriormente, verificaron directamente en el sistema SAC que:

- Los contactos válidos fueron correctamente creados.
- Los productos financieros se asignaron a los clientes correspondientes.
- Las transacciones y saldos se actualizaron según los archivos de Redeban (T88, TSL)

- Recibieron las notificaciones por correo electrónico con los reportes de errores adjuntos.

Finalizando esta fase, los usuarios validaron que la solución automatizada cumplía con todas las expectativas operativas, era fiable y estaba lista para realizar el despliegue en el entorno productivo. Se obtuvo feedback durante la etapa, el cual permitió realizar ajustes mínimos como mejorar el formato de los correos y mejorar la claridad con los logs de errores.

6.5. Fase de despliegue y documentación

En la última fase del proyecto, se centró en pasar la solución automatizada desde el entorno de desarrollo y pruebas hacia un entorno productivo totalmente operativo. En esta etapa, el objetivo fue asegurar una puesta en marcha controlada, transferir el conocimiento necesario a los equipos de ESSA y entregar una solución sostenible en el tiempo. Es por eso por lo que este proceso se dividió en cinco actividades clave: preparación del entorno de producción, puesta en marcha, generación de documentación, capacitación de los usuarios y la entrega formal de la solución.

6.5.1. Preparación del entorno de producción

Se preparó y configuró la infraestructura de producción para alojar la automatización. Este proceso incluyó:

- Se solicitó la asignación de una máquina virtual al equipo de TI, en donde el equipo brindó una en donde solo se ejecutan los bots de manera desatendida. En este equipo se instalaron los siguientes componentes de software necesarios:
    - Python (versión 3.11), junto con todas las librerías requeridas mencionadas anteriormente.
    - El cliente de Oracle necesario para la conexión en modo Thick, garantizando la

comunicación con la base de datos del SAC.

- Microsoft Power Automate Desktop ya estaba instalado en la máquina.
- Migración de artefactos: Se migraron las versiones finales y probadas de todos los componentes del proyecto, como los scripts de Python que fueron transferidos a una estructura de carpetas definitiva en el servidor de producción; los flujos de power automate (nube y escritorio) fueron exportados desde el entorno de desarrollo e importados en el entorno de producción.

6.5.2. Puesta en producción y monitoreo inicial

El despliegue se realizó de manera controlada a modo de minimizar cualquier impacto en la operación diaria.

- Los flujos de nube fueron activados en producción. Primeramente, se habilitó el disparador de correo electrónico para el proceso de creación de contactos y asignación de productos financieros y, posteriormente, se activó el disparador programado (Recurrence) para el proceso de carga de transacciones, alineándolo con el horario de operación (6:00 AM).
- Durante una semana de operación, el equipo del proyecto estuvo haciendo un monitoreo intensivo de cada ejecución, en donde se revisaron historiales de ejecución en power automate cloud y los archivos de log generados por los scripts de Python, a modo de validar que todos los procesos se completaban sin errores. Este acompañamiento se hizo junto a los usuarios finales para poder resolver de inmediato cualquier incidencia.

6.5.3. Generación de documentación técnica y de usuario

Para garantizar la mantenibilidad y correcta operación de la solución a largo plazo, se

hicieron dos documentos claves:

- Manual técnico: Este documento es dirigido al equipo de TI y soporte, en el cual se detalla la arquitectura y el funcionamiento interno de la solución. Incluye lo siguiente:
    - Diagramas de la arquitectura y el flujo de procesos (BPMN “To-Be”).
    - Descripción en detalle de cada script de Python, sus funciones y dependencias.
    - Explicación paso a paso de los flujos de power automate (nube y escritorio).
    - Una sección de errores comunes (como fallos en la conexión de bases de datos, o cambios en la interfaz de Redeban) y sus posibles soluciones.
- Manual de usuario: Este documento orientado a los responsables del programa “SOMOS”, el cual explica cómo interactuar con la automatización desde una perspectiva funcional. Contiene:
    - Instrucciones claras sobre el formato y la estructura que debe tener el archivo Excel de solicitudes enviado por correo.
    - Una guía para interpretar las notificaciones por correo electrónico, explicando el significado de cada archivo de error adjunto.
    - Un protocolo de escalamiento en donde se define a quien contactar en caso de que la automatización falle o se presente un comportamiento inesperado.

6.5.4. Capacitación y transferencia de conocimiento

Se llevaron a cabo sesiones de capacitación con el personal involucrado para asegurar una adopción exitosa y autónoma de la nueva herramienta.

- Se hizo una sesión para los usuarios operativos, en donde se demostró en vivo el proceso completo, desde el envío del correo hasta la recepción de la notificación de errores. Se les guio sobre cómo preparar los datos de entrada y qué hacer frente a los reportes de error generados por el sistema.
- Se hizo una sesión para el equipo de soporte de TI, en donde se les transfirió el conocimiento técnico, explicando la arquitectura de la solución, la ubicación de los scripts, como revisar los logs de ejecución y los pasos básicos para el diagnóstico de problemas descritos en el manual técnico.

6.5.5. Entrega formal

Una vez concluido el periodo de monitoreo intensivo y verificado el funcionamiento estable de la solución durante dos semanas consecutivas sin incidentes mayores, se hizo la entrega formal en donde se firmó un acta de conformidad por parte del líder del área usuaria, dándole la responsabilidad de la operación diaria al equipo del programa "SOMOS" y el soporte de primer nivel al área de TI de la ESSA. Con esto, el proyecto concluyó, dejando implementada la solución funcional, documentada y sostenible.

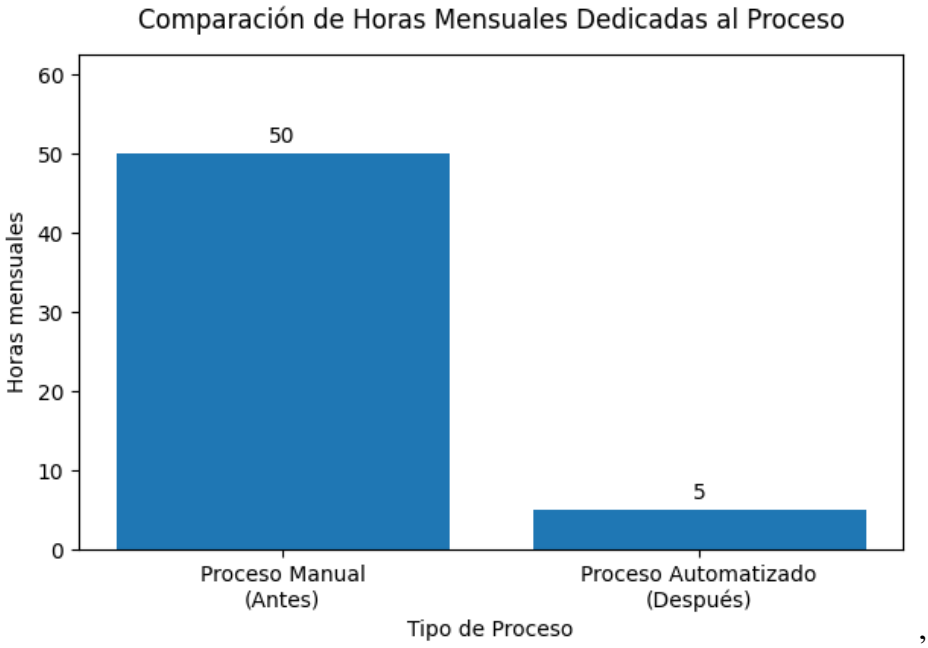
7. Resultados

Antes de la automatización, el proceso de creación de contactos, asignación de productos financieros y el cargue de transacciones y actualización de saldos era una tarea enteramente manual, intensiva en tiempo y susceptible a errores. Gracias a la implementación de la solución RPA, hubo una reducción del tiempo de procesamiento, en donde manualmente, con un promedio de 20 solicitudes diarias, se consumía entre 2 y 3 horas de trabajo; ahora se ejecuta el ciclo completo (recepción, validación, carga y notificación) en aproximadamente 15 minutos. Esto

representa una reducción de más del 90% en el tiempo de dedicación diario a esta tarea, traduciéndose a una liberación anual de más de 500 horas de trabajo manual.

Figura 31

Comparación de esfuerzo operativo mensual (Horas)



Nota. Gráfica realizada por el autor. La figura muestra la reducción del tiempo de dedicación requerido por el personal, pasando de 50 horas mensuales en el proceso manual a solo 5 horas mensuales para la supervisión del proceso automatizado.

Como se observa en la Figura 31, el esfuerzo requerido se ha minimizado. Esto no solo optimiza el uso del tiempo del personal actual, sino que también le brinda al programa "SOMOS" la capacidad de escalar y procesar un volumen de solicitudes mucho mayor sin la necesidad de aumentar los recursos humanos.

La eliminación de errores humanos ha sido uno de los principales beneficios de la

automatización. Antes, la entrada manual de datos tenía un riesgo de errores de digitación, como números de cédula incorrectos o asignación de productos equivocados. Gracias a que ahora el sistema procesa la información de manera sistemática y la valida contra las fuentes maestras, la tasa de errores se ha reducido prácticamente a cero. Los únicos errores posibles que se presentan provienen de la calidad de los datos de entrada, como, por ejemplo, cuando un cliente no existe en la base de OnCredit, los cuales son identificados y notificados para su corrección.

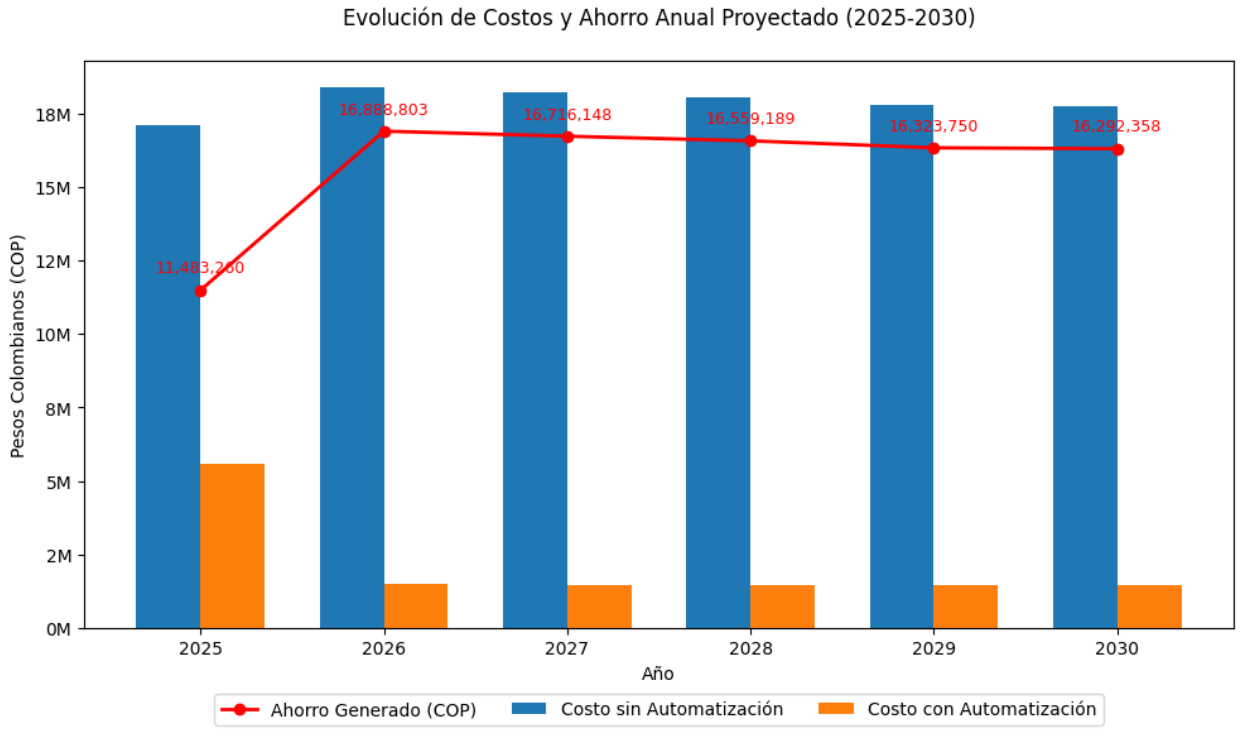
Además, la automatización ha representado una mejora significativa en la experiencia del cliente, ya que uno de los principales problemas del proceso manual era el retraso en la activación de los productos financieros, que hacía que los clientes tuvieran que esperar hasta el día siguiente para poder usarlos. Con la implementación del nuevo sistema, la activación se realiza el mismo día en que se recibe la solicitud, eliminando fricciones y elevando la satisfacción del usuario, así como su percepción de agilidad en la atención.

La métrica que se usa en la ESSA para medir la rentabilidad es el retorno sobre la inversión (ROI). En el cálculo del ROI se tiene en cuenta el costo de la tarea sin automatizar, el costo de la automatización y el costo de la tarea con automatización. En el costo de la automatización se tienen en cuenta los aspectos de costo anual de las personas que desarrollaron el bot, el costo anual de la máquina virtual usada por el bot, el costo anual del orquestador usado por el bot, el costo anual de la licencia usada por el bot y los costos de mantenimiento.

La Figura 32 ofrece una visión amplia de la viabilidad económica del proyecto, en donde se comparan los costos del proceso manual con los de la solución automatizada y el ahorro generado.

Figura 32

Evolución de costos y ahorro anual proyectado (2025-2030)



Nota: *Gráfica realizada por el autor.* La figura compara el costo anual del proceso manual ("Costo sin Automatización") con el costo total de la solución automatizada. La línea roja indica el ahorro neto generado cada año.

Se puede observar que las barras azules de la Figura 32 representan el alto costo del proceso manual (superior a los $18 millones de COP anuales), y contrastan fuertemente con las barras naranjas, las cuales muestran un costo mucho menor de la operación automatizada. Esto quiere decir que hay un ahorro neto anual que se estabiliza por encima de los $16 millones de COP y un ROI con valor estimado de 91.87% a partir del segundo año, poniendo en evidencia una optimización significativa de los recursos.

8. Conclusiones

Tras la finalización del diseño, implementación y evaluación de la solución de automatización, se dan las siguientes conclusiones:

- El objetivo general se logró con éxito mediante la implementación de una solución RPA eficaz. Este proyecto ha cambiado significativamente el proceso de gestión de solicitudes en el programa SOMOS, donde la eficiencia operativa mejoró considerablemente, como lo indica una reducción del tiempo de procesamiento de más del 90%, que se redujo a minutos. La eliminación de la introducción manual de datos eliminó los errores operativos, lo que garantizó una información de alta calidad en el SAC. Por último, aumentó el nivel de satisfacción de los clientes, lo que fue posible gracias a la posibilidad de activar productos financieros en el mismo día de la solicitud.

- La metodología basada en los objetivos específicos demostró ser una hoja de ruta efectiva para el éxito del proyecto. El análisis en detalle del flujo de trabajo "As-Is" fue muy importante para identificar los cuellos de botella y así definir los requerimientos de la automatización. El diseño de la arquitectura híbrida que fusiona Power Automate y Python demostró ser una excelente estrategia, proporcionando la facilidad de orquestación de una plataforma low-code y la potencia necesaria para el procesamiento de datos complejos. Finalmente, la implementación del RPA se realizó con éxito, entregando una solución robusta.

- Este proyecto no es solo un éxito técnico, también es un caso de negocio rentable para la ESSA. Esta solución genera un ahorro anual sostenido superior a los $16 millones de COP.

- La solución sirve como modelo probado para dar seguimiento a los proyectos de transformación digital en la organización. Este proyecto ha demostrado empíricamente el

valor de la automatización inteligente en situaciones operativas. Con más de 500 horas anuales de trabajo repetitivo liberadas, no solo hace que para la ESSA sea rentable, sino que también ayuda al personal a centrarse en actividades más relevantes desde el punto de vista estratégico, lo que crea un ambiente favorable para la innovación y la mejora constante.

9. Trabajo futuro

Aunque la solución adoptada ya ha demostrado ser sólida y cumple todos los objetivos propuestos, la naturaleza de la automatización como proceso dinámico permite una evolución y mejoras continuas. A continuación, se describen algunas áreas de trabajo futuro que pueden contribuir aún más al valor global de la solución:

- **Implementación de un Dashboard de Monitoreo y Analítica:**

  La creación de un panel de control en Microsoft Power BI que se comunique con los registros de ejecución de Power Automate y los archivos de logs generados por Python. Esto permitiría tanto al equipo de SOMOS como al departamento de TI ver el rendimiento del bot en tiempo real, como el número de casos procesados por él, el tiempo medio de procesamiento y las tasas de error, así como proporcionar medidas analíticas de las solicitudes de los clientes. Estas capacidades ayudarían a una gestión proactiva y facilitarían la toma de decisiones basada en datos.

- **Incorporación de Inteligencia Artificial para el Procesamiento de Documentos:**

  La mejora del proceso mediante la inteligencia artificial, y en especial el reconocimiento óptico de caracteres (OCR), permitiría extraer automáticamente datos en formatos no

estructurados, como plantillas de solicitudes escaneadas o fotocopias de documentos de identidad. Esto eliminaría la necesidad de copiar manualmente la información a archivos de Excel y, por lo tanto, automatizaría aún más el ciclo de vida de la solicitud.

- **Expansión de la Automatización a Otros Procesos del Programa "SOMOS":**

  La arquitectura y los componentes actuales deben utilizarse como base para la automatización de otros procesos relacionados dentro del programa SOMOS. Las posibles ampliaciones son la gestión de solicitudes, quejas y reclamos relacionadas con el programa (PQR); la creación automática de informes de gestión mensuales; y la automatización de las campañas de comunicación y marketing dirigidas a los clientes afiliados.

Referencias bibliográficas

Electrificadora de Santander (ESSA). (n.d.). *Somos*. https://www.essa.com.co/site/clientes/hogar/somos

Gómez González, L. M. (2020). *Aplicaciones de RPA en el ámbito empresarial* [Tesis de fin de grado, Universidad Politécnica de Madrid]. Archivo Digital UPM. https://oa.upm.es/58123/1/TFG_LAURA_MARIA_GOMEZ_GONZALEZ.pdf

Impacto TIC. (2024, 29 de julio). *Automatización Robótica de Procesos (RPA) multiplica eficiencia en empresas*. Impacto TIC. https://impactotic.co/innovacion/transformacion-digital/rpa-automatizacion-robotica-de-procesos/

McKinsey & Company. *The transformative power of automation in banking*. McKinsey & Company. https://www.mckinsey.com/~/media/mckinsey/industries/financial services/our insights/the transformative power of automation in banking/the-transformative-power-of-automation-in-banking.pdf

Techneplus. *Automating customer onboarding with RPA: A case study of a bank's KYC project in Bermuda*. Techneplus. https://www.techneplus.com/automating-customer-onboarding-with-rpa-a-case-study-of-a-banks-kyc-project-in-bermuda/

Mirabete, Martín & Villagra, Andrea & Pandolfi, Daniel. (2024). AUTOMATIZACIÓN ROBÓTICA DE PROCESOS: UNA REVISIÓN SISTEMÁTICA DE LA LITERATURA. https://www.researchgate.net/publication/387097792_AUTOMATIZACION_ROBOTICA_DE_PROCESOS_UNA_REVISION_SISTEMATICA_DE_LA_LITERATURA

Microsoft. (s. f.). *Power Automate*. https://www.microsoft.com/es-es/power-platform/products/power-automate#tabs-pill-bar-ocb9d4_tab3

Microsoft. (s. f.). *Introducción a Power Automate*. Microsoft Learn. https://learn.microsoft.com/es-es/power-automate/overview-cloud

Microsoft. (s. f.). *Introducción a los flujos de escritorio*. Microsoft Learn. https://learn.microsoft.com/es-es/power-automate/desktop-flows/introduction

Microsoft. (s. f.). *Disparar flujos de escritorio*. Microsoft Learn. https://learn.microsoft.com/es-es/power-automate/desktop-flows/trigger-desktop-flows

Microsoft. (s. f.). *Scripting actions: Run Python script*. Microsoft Learn. https://learn.microsoft.com/en-us/power-automate/desktop-flows/actions-reference/scripting#runpythonscript

Agudelo, S. (2023). *Automatización de los procesos de extracción, transformación, carga de sucursales y la categorización de los comentarios de los clientes del área de Inteligencia Experiencia del Cliente de Bancolombia S. A*. Universidad de Antioquia. https://bibliotecadigital.udea.edu.co/server/api/core/bitstreams/633d127b-4c33-4b4e-ad7b-bb168e4b4a78/content