Optimización del Proceso de Generación de Bases De Datos para la Medición de Experiencia de Usuario en el Área de Soluciones Inmobiliarias de una Empresa del Sector Financiero

Ricardo Andres Rivas Ramos

Asesor

Gabriel Jaime Rivera León

Universidad Nacional Abierta y a Distancia - UNAD

Escuela De Ciencias Básicas Tecnología E Ingeniería ECBTI

Medellín, Antioquia

2024

Resumen

El equipo de solución inmobiliaria del Grupo Bancolombia, se enfrentaba al desafío significativo de generar manualmente bases de datos para seleccionar clientes en encuestas de experiencia, este laborioso proceso no solo consumía tiempo, con un promedio mensual de 3 horas por auxiliar, sino que también presentaba un riesgo inherente de errores, lo que obstaculizaba la evaluación precisa de la experiencia del cliente en etapas clave del proceso inmobiliario. Los indicadores cuantificables revelaban la necesidad de mejorar este proceso, incluyendo el tiempo promedio de generación de bases y devoluciones por errores en la información. La pregunta central que motiva la propuesta de esta investigación es cómo mejorar este proceso para reducir el tiempo empleado y garantizar una selección precisa de participantes mediante un método que mejore la eficiencia operativa de manera escalable y sostenible. En ese orden de ideas, para dar solución a esta problemática se logró implementar un sistema automatizado que optimizó el proceso de generación de bases de datos. A través de la identificación de criterios de selección y el desarrollo de una herramienta automatizada que fue validada, implementada y en la que se evaluaron los resultados. El desarrollo de la herramienta se describe en varios pasos, desde entender las necesidades del equipo hasta la validación de la herramienta desarrollada. La elección de Python y SQL como pilares tecnológicos para su desarrollo no fue fortuita; se priorizó la simplicidad y eficacia en el manejo de datos, entendiendo que la accesibilidad y la robustez eran clave para el éxito de la implementación. De esta forma, los resultados obtenidos fueron contundentes: una reducción drástica del tiempo dedicado, pasando de 3 horas a apenas 10.08 minutos, liberando un 95.8% de la capacidad previamente dedicada a esta tarea. Este logro no solo impactó positivamente la operatividad del equipo, pues también fue reconocido en el

programa de hitos inmobiliarios del Grupo Bancolombia, consolidándose como un caso de éxito emblemático dentro de la organización.

Palabras clave**:** Grupo Bancolombia, Solución inmobiliaria, Eficiencia operativa, Sistema automatizado

Abstract

The Bancolombia Group's real estate solution team faced the significant challenge of manually generating databases to select clients in experience surveys. This laborious process was not only time-consuming, with a monthly average of 3 hours per assistant, but also presented an inherent risk of errors, which hindered the accurate evaluation of the customer experience at key stages of the real estate process. Quantifiable indicators revealed the need to improve this process, including the average time to generate databases and returns due to errors in the information. The central question that motivates the proposal of this research is how to improve this process to reduce the time spent and guarantee an accurate selection of participants through a method that improves operational efficiency in a scalable and sustainable way. In this order of ideas, to solve this problem, an automated system was implemented that optimized the database generation process. Through the identification of selection criteria and the development of an automated tool that was validated, implemented and in which the results were evaluated. The development of the tool is described in several steps, from understanding the team's needs to validating the developed tool. The choice of Python and SQL as technological pillars for its development was not fortuitous; simplicity and efficiency in data management were prioritized, understanding that accessibility and robustness were key to the success of the implementation. In this way, the results obtained were overwhelming: a drastic reduction in the time spent, going from 3 hours to just 10.08 minutes, freeing up 95.8% of the capacity previously dedicated to this task. This achievement not only positively impacted the team's operation, as it was also recognized in the Bancolombia Group's real estate milestones program, consolidating itself as an emblematic success story within the organization.

Keywords: Bancolombia Group's, Real estate solution, Operational efficiency, Automated system

Tabla de Contenido

Lista de Tablas

Lista de Figuras

Introducción

En el panorama actual, la automatización de procesos se ha convertido en un elemento crucial para la optimización de la eficiencia operativa y la mejora continua de los servicios ofrecidos. En este contexto, Bancolombia, uno de los líderes en el sector bancario colombiano, ha emprendido un viaje hacia la transformación digital con el objetivo de mejorar la experiencia del cliente, agilizar operaciones y fortalecer su posición competitiva en el mercado.

Este trabajo se centra en analizar y presentar un estudio de caso sobre la automatización de un proceso específico dentro de Bancolombia. Se explorará en detalle el proceso seleccionado, los desafíos enfrentados por la entidad, las soluciones implementadas mediante la automatización, y los impactos resultantes en términos de eficiencia, calidad y satisfacción del cliente. Dicha investigación profundiza en detalle y evalúa el papel de la automatización en la mejora de la precisión, la reducción de errores y la mitigación de riesgos asociados con el proceso en cuestión.

Finalmente, este estudio proporcionará una visión panorámica de los beneficios tangibles e intangibles obtenidos por Bancolombia a través de la automatización de procesos, así como recomendaciones para futuras iniciativas de transformación digital en el ámbito bancario.

Problema de investigación

En el equipo de solución inmobiliaria del Grupo Bancolombia, la generación manual de bases de datos para la selección de clientes que participan en encuestas de experiencia representa un desafío significativo en términos de tiempo y precisión. Actualmente, el proceso implica que el auxiliar encargado procese 10 insumos de Excel, consumiendo un tiempo promedio de 3 horas mensuales. La necesidad de evaluar la experiencia del cliente en etapas claves, como avalúo, escrituración y desembolso, se ve obstaculizada por la gran operatividad que demanda esta tarea.

Indicadores cuantificables incluyen el tiempo promedio actual de generación de bases (3 horas mensuales), devoluciones por errores en la información (2 mensuales) y una frecuencia mensual en el envío de evaluaciones de experiencias en etapas clave. La ejecución del proyecto aspira a ofrecer una solución que contribuya significativamente a la mejora de la eficiencia operativa y a la captura de momentos cruciales de la experiencia del cliente en el sector inmobiliario del Grupo Bancolombia.

En este contexto, la pregunta central que motiva la propuesta es: ¿Cómo podemos mejorar el proceso de generación de bases de datos para la selección de clientes en encuestas de experiencia, garantizando no solo una reducción sustancial del tiempo empleado, sino también una mayor precisión en la selección de participantes?

Planteamiento del problema

La gestión de la experiencia del cliente cada día cobra más importancia en las empresas, puesto que a menudo dependen de encuestas para conocer, evaluar y mejorar el servicio. Teniendo en cuenta este fenómeno, es importante realizar de forma apropiada la selección de clientes que permita un adecuado análisis de la satisfacción de la base de clientes actual.

Para abordar la pregunta planteada y mejorar el proceso, es fundamental tener en cuenta los indicadores cuantificables del mismo. En este caso, dos de los indicadores clave son el tiempo promedio de generación de bases y el número de devoluciones por errores en la información. Como se ha dicho antes, el tiempo promedio de generación de bases es de 3 horas mensuales, indicando una eficiencia operativa que podría ser mejorada. Asimismo, se registran 2 devoluciones mensuales debido a errores en la información, lo que sugiere la necesidad de optimizar la precisión y la calidad de los datos.

Al afrontar estos desafíos, es necesario implementar un método que tenga un impacto directo en estos ítems, permitiendo aumentar la eficiencia operativa y reducir los errores en la información. Es crucial que este método sea escalable y sostenible a largo plazo, lo que implica utilizar herramientas diseñadas específicamente para el manejo masivo de datos.

Una posible solución podría ser la implementación de un sistema automatizado de generación de bases de datos, que agilice el proceso y reduzca significativamente el tiempo requerido. Este sistema podría incluir algoritmos de validación de datos en tiempo real para minimizar los errores y garantizar la precisión de la información. Además, la adopción de herramientas de gestión de datos avanzadas, como bases de datos relacionales o plataformas de gestión de datos en la nube, podría facilitar la escalabilidad y la sostenibilidad del proceso a medida que la cantidad de datos aumente.

En resumen, para mejorar el proceso de generación de bases de datos es fundamental implementar un enfoque que tenga en cuenta los indicadores cuantificables, como el tiempo de generación y la precisión de los datos, en el que se utilicen herramientas y métodos que permitan aumentar la eficiencia operativa para garantizar la calidad de la información de manera escalable y sostenible.

Justificación

En el actual contexto de globalización y competitividad comercial en el que las organizaciones se desarrollan, la estrategia de satisfacer y fidelizar al cliente cobra cada vez más relevancia. En este sentido, Del Puerto (2023), plantea que escuchar la voz del cliente a través del estudio de su "experiencia" con la empresa se constituye en el camino para garantizar un crecimiento rentable y sostenido.

Optimizar el proceso de generación de bases de datos para la medición de experiencia del usuario en el área de soluciones inmobiliarias, representa una oportunidad estratégica para la empresa con un impacto positivo en múltiples áreas como: la liberación de FTE (Full Time Equivalent), es decir que reducen los costos laborales, porque al tener más tiempo los empleados podrán enfocarse en otras actividades productivas y eficientes para la empresa; también, habrá minimización de errores debido a que la automatización del proceso elimina la posibilidad de errores humanos, esto garantiza exactitud y confiabilidad en la recolección de datos, de esta forma, se podrá realizar la toma de decisiones, identificar rápidamente áreas de mejora, detectar tendencias emergentes y crear sus estrategias de manera oportuna, optimizando sus productos y servicios para satisfacer mejor las necesidades de sus clientes.

La implementación de tecnologías avanzadas garantiza escalabilidad en los procesos, puesto que se podrá adaptar fácilmente en un mayor volumen de encuestas y usuarios, permitiendo un crecimiento sostenido sin comprometer la eficiencia del proceso. Así como también la estandarización, ya que respalda la consistencia en la forma en que se recopilan y procesan los datos, asegurando la calidad y comparabilidad de la información. Finalmente, la puesta en marcha de estos procesos también mejora la satisfacción de los empleados, entendiendo que realizar actividades manuales y repetidas disminuye la efectividad en el trabajo.

Objetivos

Objetivo General

Implementar un sistema automatizado que permita optimizar el proceso de generación de bases de datos para la medición de experiencia de usuario en el área de soluciones inmobiliarias de una empresa del sector financiero.

Objetivos Específicos

Identificar los criterios que se deben considerar en la selección de clientes que participan en la evaluación de la experiencia de usuario.

Desarrollar una herramienta que permita automatizar el proceso de generación de bases de datos para medición de la experiencia de usuario.

Validar la herramienta desarrollada para garantizar que las bases de datos generadas manejen y generen información certificada.

Implementar la herramienta desarrollada en el proceso de generación de bases de datos para la evaluación de la experiencia de usuario.

Evaluar los resultados obtenidos en cuando a la reducción de tiempos, y aumento de la tasa de respuesta, garantizando que la información generada tenga como máximo dos días de antigüedad

Marco Teórico

La Experiencia del Cliente

De acuerdo con las dinámicas cambiantes, la industria se ha dado cuenta que mejorar únicamente el aspecto de un sistema no resulta suficiente para satisfacer las necesidades de los usuarios, y la experiencia de este influye directamente en la recepción de la transformación de las empresas frente a las demandas de estos. Por eso, se han originado numerosos conceptos que intentan reflejar dichos aspectos de cara a ofrecer servicios más atractivos en función de las necesidades y expectativas de los usuarios. Surge el concepto de experiencia de usuario (del inglés User Experience, UX), entendiéndolo según Norman (2013) como una corriente de estudio e investigación centrada en analizar la relación que se establece entre las personas, las tecnologías y los dispositivos de cualquier tipo que se utilizan para realizar diversas tareas, poniendo especial énfasis en la vertiente emocional e intelectual de estas asociaciones.

Por otro lado, Gartner (2015) plantea que la experiencia del cliente puede definirse como el conjunto de percepciones que tienen todos los individuos que interactúan con la marca, a través de los diferentes canales. En ese sentido, cuando se conoce el sentir del cliente existe la capacidad de mejorar y optimizar los procesos, al diseñar de acuerdo con las necesidades de este. De esta forma, se pueden crear organizaciones más cercanas y que tengan un crecimiento sostenible, trabajando con el propósito de satisfacer las necesidades reales que generen un beneficio para la sociedad y el mercado que se está atendiendo.

Peter Drucker, el padre del management, sintetiza con su frase el fin de los conceptos que diversos autores han planteado anteriormente *lo que no se mide no se puede mejorar*, debido a que se centra en la perspectiva en la que el motivo y ser de cualquier negocio es satisfacer las necesidades y deseos de los clientes. Esto implica que todas las funciones de la empresa deben

orientarse hacia la creación de valor para el cliente. Así mismo, se fundamenta en una de las fases del proceso administrativo, el control, porque conocer el estado del proceso que se administra, permite ejecutar planes de mejora que ayudan a alcanzar objetivos.

El interés y la preocupación por ofrecer sistemas interactivos que los usuarios describen como eficaces, eficientes y satisfactorios ha llevado a los investigadores a estudiar estos conceptos, en el que, a partir de la experiencia adquirida en el diseño y ejecución de sistemas y aplicaciones, se han desarrollado y perfeccionado numerosos métodos que persiguen su medición. Medir la experiencia del cliente es crucial para entender cómo estos perciben e interactúan con una marca, producto o servicio. Se debe resaltar que son varios los autores pioneros que han contribuido al desarrollo de estas herramientas.

Frederick F. Reichheld, en su libro *The Ultimate Question: Driving Good Profits and True Growth* (2007), creó la idea de medir la lealtad de del cliente a través de la pregunta ¿nos recomendaría con un amigo o colega? De esta forma, introdujo el Net Promoter Score (NPS) en el que se puede calcular el nivel de recomendación que tiene el cliente sobre el producto, este enfoque se basa en la idea de que los clientes promotores son cruciales para el crecimiento empresarial. Sin embargo, este mismo autor, más adelante en su libro *Winning on Purpose* (2021) propone el Earned Growth Rate (RGR,) que en términos amplios se centra en la cantidad de nuevos clientes que se generan a través de referencias positivas de los clientes existentes. Es decir, el crecimiento orgánico que tenga la empresa a partir de la capacidad para mantener y aumentar sus clientes.

Por otro lado, autores como Matthew Dixon, Karen Freeman, y Nicholas Toman en su artículo *Stop Trying to Delight Your Customers* (2010), abordaron el Customer Effort Score (CES), este es un indicador para reducir el esfuerzo del cliente. Por lo tanto, mide la facilidad

con la que el cliente puede resolver su necesidad con un producto o servicio. Mientras que Fornell, en su artículo *The American Customer Satisfaction Index: Nature, Purpose, and Findings* (1996), plantea una métrica que busca medir la satisfacción general de los clientes, este indicador es importante porque ayuda a las empresas a identificar áreas de mejora, monitorizar el desempeño a lo largo del tiempo y comparar su desempeño con el de la competencia.

También existen otras herramientas esenciales para este proceso de medición, como los KPI (Key Performance Indicators).

> Los indicadores claves de rendimiento KPI, son un conjunto de indicadores útiles en las organizaciones y en los proyectos para realizar la medición y monitoreo de variables establecidas al momento de decidir qué factores presentan gran influencia o tienen mayor impacto en una organización. (Ortiz y Pardo, 2021, p.9).

En ese sentido, Robert S. Kaplan y David P. Norton, establecen en su libro *The Balanced Scorecard: Translating Strategy into Action* (1996) un cuadro de mando integral que permite medir el desempeño de una organización desde cuatro perspectivas: financiera, del cliente, de procesos internos y de aprendizaje y crecimiento. En síntesis, busca alinear sus actividades con la visión de la empresa para mejorar el desempeño de este. Ajustar estos KPI al propósito y las metas del negocio es indispensable para lograr tener una relación sana con los clientes y poder generar valor.

Proceso de Gestión de Solicitudes de Crédito Hipotecario y Leasing Habitacional

De acuerdo con el artículo 51 de la constitución política de Colombia de 1991:

> Todos los colombianos tienen derecho a vivienda digna. El estado fijará las condiciones necesarias para hacer efectivo este derecho y promoverá planes de vivienda de interés social, sistemas adecuados de financiación a largo plazo y formas

> asociativas de ejecución de estos programas de vivienda. (Constitución Política de Colombia, 1991)

Sin embargo, el acceso a una vivienda en Colombia es un tema complejo, debido a que involucra varios factores, ya sea de tipo económico, social o político. El periódico el Portafolio en el año 2024, afirmó que "en el 2023, según la encuesta nacional de calidad de vida del DANE, en Colombia había 52,3 millones de personas que vivía en un total de 18 millones de hogares, representando un promedio de 2,90 personas por casa". Aunque se han venido creando proyectos que benefician a la ciudanía para la adquisición de una casa, existe aún una brecha muy alta para que las personas acceder a una vivienda propia.

A raíz de esta situación se han creado varias alternativas que permiten tener un inmueble a través del financiamiento, entendiendo este concepto como:

> El proceso en el que se proporciona capital a una empresa o persona para utilizar en un proyecto o negocio, es decir, recursos como dinero y crédito para que pueda ejecutar sus planes. En el caso de las compañías, suelen ser préstamos bancarios o recursos aportados por sus inversionistas. (BBVA, 2024).

Dentro de esas alternativas aparece el crédito hipotecario y el leasing habitacional; el primero, es un préstamo a mediano o largo plazo que se otorga para la compra, ampliación, reparación o construcción de una vivienda, compra de sitios, oficinas o locales comerciales, el cual queda a nombre del cliente, pero hipotecado a favor de la entidad financiera que ofrece el producto. Mientras que el segundo; hace referencia al pago mensual, que está compuesto por capital e intereses, sobre un inmueble a manera de arrendamiento por un tiempo determinado acordado con la entidad financiera que ofrece el producto, donde está la posibilidad de hacer opción de compra del mismo inmueble al final del contrato, o antes si lo prefiere, por un

porcentaje sobre el valor inicial (Banco Colpatria, s.f; Comisión para el Mercado Financiero, s.f). Ambos productos son regidos por la normativa de vivienda de cada país, en Colombia, la ley que dicta las disposiciones en materia de vivienda y hábitat es la 2079 de 2021.

El proceso para adquirir un crédito de vivienda o un leasing habitacional es el siguiente: encontrar la vivienda que desea comprar; acercarse a una entidad financiera para recibir asesoría; se realiza el estudio de crédito por parte de la entidad financiera; avalúo del inmueble; la escrituración; el desembolso y finalmente el seguimiento o monitoreo del proceso. Ahora bien, existen particularidades que pueden variar según el país y la entidad financiera, como el porcentaje de financiamiento máximo, el tipo de vivienda permitido, la cantidad de participantes o los ingresos mínimos requeridos. Por último, hay que mencionar que en el amplio y competitivo mercado financiero, la eficacia y eficiencia en las solicitudes de estos procesos, son fundamentales para satisfacer las necesidades y expectativas de los clientes, así como en la operabilidad de las entidades financieras.

Generación de Bases de Datos a Partir de la Fuente Corporativa Oficial

La generación de bases de datos a partir de fuentes corporativas oficiales es un proceso crítico que permite a las organizaciones consolidar, gestionar y utilizar su información de manera efectiva para mejorar la eficiencia operativa y la toma de decisiones. En el caso de Bancolombia, cuenta con un sistema de Big Data, que le permite tanto a equipos analíticos como no analíticos hacer uso de más de 2.000 tablas que provienen de aplicaciones Bancolombia y de otras externas al Banco. Para ellos, es un paso clave para las empresas que quieren entender las necesidades y los comportamientos de sus clientes, anticipándose a sus decisiones y optimizando sus presupuestos de inversión. Estas tablas se encuentran publicadas en la Zona de Datos Crudos (ZDC), que son los datos tal como se recopilan originalmente, sin procesamiento ni análisis. Es

decir, son los datos que se obtienen directamente de las fuentes de datos, sin haber sido procesados, limpiados o transformados de ninguna manera, y son el insumo para crear las Fuente Corporativas Oficial (FCO). Como práctica común un alto porcentaje de las consultas que se lanzan sobre el sistema son hechas a las tablas que se encuentran en ZDC.

Dentro de las operaciones más comunes que se hacen sobre estas tablas se encuentra en primera instancia, el filtro de los subconjuntos de datos por año, mes y día de partición o año, mes y día de ingestión; en segunda instancia, el filtro a nivel de registros mediante el uso de reglas lógicas del negocio y mediante subconsultas que derivan en subconjuntos de datos para luego ser comparados y seleccionar los que cumplan la condición de igualdad dada; como tercero, la unión de varias tablas mediante uso de joins para extender más campos de otras tablas necesarias para el resultado esperado; cuarto, el uso de sentencias IF para reemplazar valores obtenidos y que requieren ser normalizados a un valor predeterminado; quinto, el uso de sentencias CASE para comparar valores obtenidos y homologar a valores estándar lógicos de cada negocio o proceso en evaluación; sexto, uso de funciones para tratamiento de datos numéricos, alfanuméricos, lógicos y de fecha que permiten hacer conversiones de los datos a valores deseados por el diseñador de la consulta; y finalmente, la transformación de valores de variables codificados a valores descriptivos mediante el uso de tablas catálogos para dar un significado de negocio a los datos.

Ahora bien, existen varios tipos de datos en ZDC, dentro de los cuales están:

Catálogos: valores que dan significados a los datos, son los que permiten traducir los datos a lenguaje de negocio para un entendimiento común.

Transaccionales: corresponde a las operaciones que se efectúan sobre los datos maestros, son hechos y eventos que registran mayor número de registros que los datos maestros y catálogos.

Maestros: se refiere a las entidades que reflejan los objetos del negocio o procesos de unidad de negocio.

Diseño Modelo de Datos Zona de Resultado Para la Publicación de una FCO

Se entiende como el proceso que permite definir las estructuras de tablas y campos que se crearán en la zona de resultados como FCO's. Además, permite realizar el mapeo entre las tablas y campos. Hay que tener en cuenta algunos elementos para el diseño:

En primer lugar; el objetivo de las FCO's es responder a las necesidades de los diferentes consumidores que requieren información para sus diferentes procesos. Los diseños deben ser orientados a la simplicidad y baja complejidad de tal forma que quien consuma la información tenga que hacer el menor número de consultas y uniones de tablas para adquirir los datos.

En segundo lugar, se deben identificar las entidades, tablas del negocio o del dominio de información. Conocer cuál es la entidad padre que sin la presencia de esta no pueden existir aquellas entidades con las que se relaciona. Usualmente es una entidad principal de tipo dato maestro. En tercer lugar, para aquellas entidades identificadas, se debe entender cómo se referencian o relacionan. Esto significa aquellos campos de una entidad A que hacen referencia a otros campos de una entidad B y que por medio de estos hay coincidencia y se pueden unir.

En cuarto lugar, los diseños no deben contemplar datos que pertenezcan a otras FCO's solamente las referencias cruzadas hacia esas otras FCO's. Para dar claridad si se está diseñando una FCO de cartera esta pieza de datos no debe tener en su diseño campos relacionados con la

FCO de clientes y únicamente debe tener la forma de relacionarse con esta mediante una llave como puede ser Llave MDM o Llave Nombre de CIF.

Como quinto, los campos del tipo código o valor que hacen referencia a tablas de catálogos, dentro del diseño de la estructura FCO se debe incluir dentro de la entidad, el código y la descripción y hacerlos parte de la nueva entidad. Con esto se busca desnormalizar el modelo y entregarle al usuario unas entidades simplificadas con un alto nivel de traducción de la información a lenguaje de negocio. En sexto lugar, el nivel de granularidad para tener en cuenta debe ser aquel que entregue el mayor detalle posible. Las FCO's no se deben convertir en resúmenes de información, o piezas de segundo nivel. Una FCO no debería ser un subconjunto de datos del universo de datos, debe entregar al máximo lo disponible en el universo de datos.

Hay otros puntos que son fundamentales para la FCO como es el caso de las sábanas de datos; tabla que unifica el contenido de varias entidades relacionadas mediante la selección de los campos que cada una puede aportar. Para esto es importante comprender la relación de cardinalidad que existe entre las tablas que se van a unir. La cardinalidad 1 a 1, hace referencia a que por cada registro de la Entidad A existe un registro que se relaciona en la Entidad B. Para aquellas que su relación de cardinalidad es 1 a 1 se debe validar si todas las tuplas tienen correspondencia o algunas no tienen y así determinar el tipo de unión que se debe efectuar para evitar la pérdida de información. La cardinalidad 1 a N, por cada registro de la Entidad A existen N registros que relacionan en la Entidad B. Para aquellas que su relación de cardinalidad es 1 a N, se debe evaluar si aplica el aplanamiento a cardinalidad 1 a 1 mediante la transposición de los registros en columnas e igualmente evaluar el tipo de unión a efectuar. Es importante tener en cuenta el tipo de N, que pueden ser:

N Fijo: para N registros fijos el número de campos a transponer seria la multiplicación de N Registros (determinar si existe alguna excepción de registros) x M Campos (determinar que campos se requieren para nueva tabla) a transponer en la nueva tabla aplanada.

N variable: para un N variable se puede evaluar si es posible aplanar la estructura mediante la selección de un número específico de registros que lo pueda llevar a una cardinalidad con N fijo y así poder hacer la transposición. En caso de que no se posible se recomienda mantenerlo por separado en otra entidad de las fuentes crudas que poblaran las nuevas FCO's.

Hay que mencionar también que, los datos dejan de ser considerados como dato de fuente corporativa oficial cuando ya no cumple con los criterios de precisión, actualidad, confiabilidad y relevancia que son requeridos para mantener su estatus oficial. Esto puede ocurrir ya sea por desactualización, errores, cambios y demás. Al suceder esto, deja de ser considerado como fuente oficial y puede ser retirado de los registros corporativos.

Ahora bien, en el caso de las FCO para que una fuente pueda ser considerada, se debe tener en cuenta que el resultado derivado de esta se convierte en insumo necesario de cara a procesos del negocio o de la operación. Es decir, sin esta se generaría afectaciones que impedirían una adecuada operación del negocio; su ejecución es periódica la cual es definida acorde a las necesidades de los distintos procesos que requieren de esta información en los tiempos definidos; no es exhaustiva en el manejo exacto de las cifras. Esto significa que para procesos que requieran información con un grado de precisión muy alto no se debe crear rutinas en la Landing Zone (LZ) y se debe buscar otra alternativa. Más del 90% de la información que requiere como entrada la rutina para generar sus resultados se encuentra concentrado en la LZ en datos crudos. Además, debe tener en cuenta la creación de la rutina la versión de los

componentes que se encuentran en el ambiente de producción con el fin de evitar reprocesos a la hora de entrega al equipo de Tecnología. Se debe realizar la validación de las horas de ingestión de aquellos insumos iniciales que son prerrequisito de la rutina para definir el horario en el cual se programará su ejecución.

En ese orden de ideas, el programa orquestador de los scripts SQL debe ser Python y en ningún momento se debe trabajar con procesos paralelizados para no generar indisponibilidad a otras rutinas que estén a la par. Los resultados de dicha rutina se publicarán en la Zona de Resultados del área de conocimiento del responsable y cumpliendo los lineamientos definidos para publicarla. También, cada nuevo control de cambios a la versión de la rutina tendrá como tiempo de estabilización una semana por medio de una ejecución controlada por parte del equipo de desarrollo de TI (Célula LZ), una vez se cumpla el tiempo sin errores será promovida para que la ejecución se realice en malla operativa de tecnología. Dichas rutinas son responsabilidad del área que la desarrolló, sus líderes deben garantizar el conocimiento de esta, para que cualquier persona del equipo este en capacidad de soportarla y debe tener un aval funcional del PO, líder analítico, líder funcional o quien ejecute el rol de la necesidad.

Manejo de Bases de Datos en SQL

En el mundo moderno los usuarios cada vez demandan más recursos en cuanto a tecnología, por lo tanto, es necesario realizar evoluciones en las bases de datos, pues son indispensables en los sistemas de información debido a que estas se utilizan en todas las áreas profesionales como la investigación, tecnología, arte, educación, sistemas médicos, programas de ingeniería, programas de desarrollo, de diseño, sistemas de información geográfica, entre otros.

“Una base de datos es un conjunto de datos almacenados en memoria externa que están organizados mediante una estructura de datos.” Marqués-Andrés, M. (2011). En otras palabras,

es un sistema electrónico que permite almacenar grandes cantidades de información de forma organizada para que sea fácil de acceder, manejar y modificar según sea necesario. Generalmente para gestionar la información contenida se utilizan sistemas de administración de bases de datos o DBMS por sus siglas en inglés, el cual funciona como un ecosistema para el tratamiento de datos y contiene los siguientes elementos: datos, esquema de datos, motor de la base de datos.

Datos

Al adentrarse más en el concepto de bases de datos es fundamental entender qué es un dato e información debido a que estos elementos son importantes para el desarrollo de estas. Para Juárez (2006) “es un conjunto de caracteres con algún significado, pueden ser numéricos, alfabéticos, o alfanuméricos, este es la unidad mínima de información. Un dato dentro de una base de datos responde a la función (objeto, atributo, valor)” (p.45). Los datos son la razón de ser de un DBMS y contienen toda la información del proceso, proyecto o negocio que se quiere analizar. Estos pueden provenir de diferentes fuentes, ya sea aplicativos, insumos manuales o generados por el mismos DBMS.

Esquema de datos

De acuerdo con la International Business Machines Corporation (IBM) “un esquema de base de datos define cómo se organizan los datos dentro de una base de datos relacional; esto incluye restricciones lógicas, como nombres de tablas, campos, tipos de datos y las relaciones entre estas entidades”. El esquema de datos proporciona una vista abstracta y estructurada de cómo se organizará la información dentro de la base de datos. Este diseño es la base tanto para el desarrollo inicial de la base de datos como para su mantenimiento y evolución a lo largo del tiempo porque permite establecer conexiones, parámetros y formas de optimizar el flujo de los datos en mi DBMS.

Motor de bases de datos

> Para la Agencia de Marketing Digital ToGrow
>
> Un motor de base de datos es el componente de software que permite a una base de datos almacenar, modificar, extraer y buscar información de manera eficiente. Actúa como el intermediario entre la base de datos física y las aplicaciones que solicitan datos, asegurando que las consultas y transacciones se ejecuten de manera efectiva. ToGrow. (s.f.)

En otras palabras, es el corazón del DBMS, debido a que actúa esencialmente como una capa intermedia entre los usuarios y los datos almacenados, facilitando diversas operaciones como la creación de tablas, consultas, inserción de datos, actualización, eliminación y mantenimiento general de la base de datos.

Para poder generar cada una de estas operaciones es necesario utilizar un lenguaje informático que permita dar instrucciones al motor, este lenguaje es llamado SQL, por sus siglas en inglés significa Lenguaje de Consulta Estructurada (Structured Query Language). Ahora bien, dentro de los motores de bases de datos más comunes se encuentra:

SQL Server

MySQL

SQLite

Oracle database

PostgreSQL

Todos estos motores trabajan bajo el mismo método de consulta SQL, sin embargo, puede que existan particularidades que hagan variar la cantidad de funciones disponible en cada sistema.

Query o Consultas a Bases de Datos

Las consultas o Querys como son llamadas en el mundo de la informática, son instrucciones o comandos escritos en un lenguaje especifico, que son indispensables para cualquier proyecto tecnológico, porque facilita recopilar, recuperar, manipular y gestionar información de diversas fuentes para procesarla y tomar decisiones que ayuden a apalancar procesos, proyectos o negocios.

CRUD (Create, Read, Update, Delete)

Las operaciones CRUD (Create, Read, Update, Delete) son fundamentales en el manejo de bases de datos, permitiendo a los usuarios interactuar efectivamente con los datos:

Crear (Create): "la operación de creación permite insertar nuevos registros en la base de datos" (Connolly & Begg, 2014, p. 263).

Leer (Read): "la operación de lectura es fundamental para recuperar información almacenada en la base de datos" (Ramakrishnan & Gehrke, 2003, p. 86).

Actualizar (Update): "actualizar permite modificar registros existentes en la base de datos según las necesidades del usuario" (Elmasri & Navathe, 2016, p. 414).

Eliminar (Delete): "eliminar registros es una operación crítica que debe realizarse con precaución para evitar pérdidas de datos irreversibles" (Coronel & Morris, 2016, p. 368).

ETL (Extraction, Transformation, Loading)

ETL en español (extraer, transformar y cargar datos), es un proceso crucial e integral en la gestión y análisis de datos que se utiliza para mover y consolidar datos desde múltiples fuentes hacia una base de dato centralizada; consta de tres etapas principales:

Extracción (Extraction): "la extracción de datos implica recoger información desde diversas fuentes como bases de datos operacionales y archivos planos" (Kimball et al., 2008, p. 151).

Transformación (Transformation): "durante la etapa de transformación, los datos son limpiados, estructurados y preparados para su integración en el sistema de destino" (Inmon, 2002, p. 97).

Carga (Loading): "la carga de datos es el proceso final donde los datos transformados se insertan en el almacén de datos o base de datos de destino" (Kimball et al., 2008, p. 175).

Python

Python es un lenguaje de programación creado por Guido van Rossum a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses “Monty Python”. Se trata de un lenguaje interpretado o de script, con tipado dinámico, fuertemente tipado, multiplataforma y orientado a objetos. González Duque, R. (2015).

Es un lenguaje de programación multipropósito, es usado en campos como la automatización, programación web, ciencia de datos, juegos, aplicaciones de escritorio, etc. Python es muy recomendado para las personas que desean iniciar en el mundo de la programación debido a que su sintaxis es muy similar al lenguaje natural, además que tiene una comunidad gigante que día a día genera recursos para facilitar aún más el desarrollo en esta plataforma.

Algunos autores como Marzal Varó y Gracia Luengo (2009) destacan a Python por sus ventajas en el mundo de la programación, puesto que resaltan que tiene un lenguaje muy expresivo, es muy compacto, Python es muy legible, su sintaxis es muy elegante y permite la

escritura de programas cuya lectura resulta más fácil que si utilizaran otros lenguajes de programación.

Al mejorar y agilizar el proceso de desarrollo de software, Python dispone de una serie de librerías y framework tanto propios y desarrollados por terceros. Para poder entender cómo podría mejorar esto la programación debemos partir del concepto de librería y de framework.

Librerias

Las librerías de Python son conjuntos de módulos y funciones predefinidos que facilitan el desarrollo de software al proporcionar funcionalidades específicas que pueden ser reutilizadas en diferentes proyectos.

> Estas son una parte esencial del ecosistema del lenguaje, ya que permiten a los desarrolladores acceder a un conjunto amplio de funcionalidades específicas, como manipulación de cadenas, operaciones matemáticas, acceso a bases de datos, manipulación de archivos, creación de interfaces gráficas, procesamiento de datos científicos, creación de sitios web, entre muchas otras. (Aurora, 2023)

Para este proyecto investigativo se utilizaron específicamente tres de las librerías de Python

Datatime

PYTZ

Openpyxl

La primera, Datetime, es un “módulo que proporciona clases para manipular fechas y horas.” (Datetime — Basic Date And Time Types, s. f.). Este ofrece funciones como NOW() que permiten obtener la fecha y hora actuales del sistema. Esta funcionalidad es fundamental en aplicaciones que requieren registrar eventos en tiempo real, calcular intervalos de tiempo o

simplemente para mostrar la hora actual en una interfaz de usuario. Además de NOW(), Datetime permite la creación de objetos de fecha y hora específicos, así como la manipulación de estos objetos mediante diversas operaciones como la suma y resta de intervalos temporales, el ajuste de zonas horarias y la conversión entre formatos de fecha y hora. Dichas capacidades hacen que la librería sea extremadamente útil en el desarrollo de aplicaciones que dependen de la gestión precisa del tiempo, como sistemas de gestión de eventos, cronogramas de tareas, registros de actividad y análisis de datos temporales.

Como segundo, está PYTZ, también conocida como Python Timezone (PYTZ) “incorpora la base de datos Olson tz a Python. Esta biblioteca permite cálculos precisos y multiplataforma de zona horaria utilizando Python 2.4 o superior” (PYTZ, 2024). En este trabajo, se utilizó esta librería para obtener la zona horaria de Colombia, debido a que es ampliamente utilizada para gestionar zonas horarias en aplicaciones que requieren precisión en la representación y conversión de horarios en diferentes ubicaciones geográficas. Para la investigación, es crucial para las aplicaciones que necesitan mostrar o registrar eventos en hora local colombiana. Esta funcionalidad facilita asegurar que los horarios sean correctos y consistentes, independientemente de la ubicación geográfica del usuario o del servidor de la aplicación.

Además de la obtención de zonas horarias específicas como la de Colombia, PYTZ soporta ajustes avanzados como la conversión entre diferentes zonas horarias, el manejo de horarios de verano (DST), y la representación de fechas y horas con precisión hasta el nivel de segundo. Esto la convierte en una herramienta esencial para el desarrollo de aplicaciones globales que necesitan gestionar datos temporales de manera robusta y confiable.

En tercer lugar, está Openpyxl; “es una biblioteca de Python para leer/escribir archivos xlsx/xlsm/xltx/xltm de Excel” (Eric Gazoni, Charlie Clark, 2024). Esta herramienta es indispensable para aplicaciones que necesitan manipular datos estructurados en hojas de cálculo de Excel de manera programática y eficiente.

Una de las funcionalidades destacadas de Openpyxl es su integración con pandas, donde sirve como motor para exportar dataframes directamente a archivos de Excel. Esto facilita la creación de informes, análisis de datos y la generación de documentos con formato específico compatible con Excel, aprovechando las capacidades de ambas herramientas. Además de la exportación de dataframes, Openpyxl permite realizar operaciones avanzadas como la creación de nuevas hojas de cálculo, la edición de celdas y formatos, la inserción de gráficos y la aplicación de fórmulas complejas. Estas características hacen de Openpyxl una opción versátil y poderosa para trabajar con datos en formato Excel dentro del entorno de desarrollo de Python.

Framework

“Un framework es un esquema o marco de trabajo que ofrece una estructura base para elaborar un proyecto con objetivos específicos, una especie de plantilla que sirve como punto de partida para la organización y desarrollo de software” (Unir, 2023). Estas, facilitan el desarrollo, la implementación y la gestión de aplicaciones al proporcionar una estructura predefinida y un conjunto de patrones de diseño comunes. Los framework que usaron dentro del proyecto fueron:

Pandas

Orquestador2

Por un lado, Pandas es “una herramienta de manipulación y análisis de datos de código abierto, rápida, potente, flexible y fácil de usar, construida sobre el lenguaje de programación Python.” (Pandas - Python Data Analysis Library, s. f.). Se especializa en el manejo de datos

estructurados, especialmente en formato tabular conocido como dataframe. Esta estructura de datos es central para Pandas y permite realizar operaciones complejas de forma intuitiva, como filtrado, agrupamiento, manipulación de índices y fusión de datos.

En aplicaciones prácticas, Pandas ofrece funciones clave como to_excel, que facilita la exportación de un dataframe directamente a archivos de Excel, y read_excel, que permite la lectura de datos desde archivos de Excel hacia un dataframe en Python. Estas funcionalidades son esenciales para la interoperabilidad entre datos estructurados almacenados en hojas de cálculo y el entorno de análisis de datos en Python.

Además de la exportación e importación de datos desde y hacia Excel, Pandas soporta una amplia gama de operaciones de manipulación y transformación de datos, lo que la convierte en una herramienta imprescindible para científicos de datos, analistas y desarrolladores que trabajan con datos complejos y grandes volúmenes de información.

Por otro lado, se encuentra Orquestador2 que es un framework desarrollado por Bancolombia, diseñado específicamente para facilitar la ejecución ordenada y secuencial de una serie de pasos dentro de la plataforma analítica de la organización. Este framework es fundamental para coordinar y automatizar tareas complejas que involucran múltiples acciones sobre datos y sistemas.

Una de las características principales de Orquestador2 es su capacidad para proporcionar funciones de ayuda que permiten configurar y controlar el comportamiento de la ejecución. Esto incluye la gestión de logs, que son registros detallados de las acciones realizadas durante la ejecución, y helpers, que son utilidades o herramientas adicionales necesarias para llevar a cabo las actividades planificadas en la Zona de Aterrizaje (LZ, por sus siglas en inglés).

El framework asegura que cada paso en el proceso esté correctamente definido y que las dependencias entre ellos se manejen de manera eficiente. Esto garantiza la integridad y la precisión en la ejecución de flujos de trabajo complejos dentro del entorno analítico de Bancolombia. Además de su función principal de orquestación, Orquestador2 incluye capacidades para la gestión de errores, la monitorización del progreso de las tareas y la escalabilidad para manejar grandes volúmenes de datos. Estas características hacen que Orquestador2 sea una herramienta poderosa y central en el ecosistema analítico de Bancolombia, facilitando la automatización de procesos críticos y mejorando la eficiencia operativa en el tratamiento y análisis de datos.

Para finalizar, al desarrollar proyectos de manera organizada y escalable en Python, es fundamental trabajar dentro de entornos virtuales, puesto que es una herramienta que permite crear y gestionar entornos de desarrollo independientes y aislados para proyectos específicos. Además, son útiles porque permiten a los desarrolladores trabajar en proyectos con dependencias y versiones de paquetes diferentes sin interferencias entre ellos. Además, proporcionan una forma de aislar las dependencias específicas de cada proyecto, lo cual es crucial para evitar conflictos entre diferentes versiones de bibliotecas y garantizar la reproducibilidad del entorno de desarrollo.

Automatización de Procesos Usando Script de Python

Automatizar un proceso requiere de muchas validaciones que son ajenas al proceso de desarrollo, pero que son indispensables para su consecución, es importante conocer la necesidad de cada uno de los implicados para que la solución planteada sea ganadora para todas las partes. El conocimiento sobre el proceso también es vital, puede que la solución al problema no requiera una intervención tecnológica y que por lo contrario esta sea un obstáculo que deteriore aún más

el estado de salud de ese proceso, también puede pasar que los habilitadores que hay actualmente no cumplan con los requerimientos necesarios para dar una solución óptima. Cuando está la claridad de cada uno de los criterios y se considera que el camino correcto es la automatización se pueden utilizar herramientas como Python para mejorar el proceso a través de programas o scripts.

De forma literal la palabra Script traduce guion, cuando se automatiza un proceso con Python, lo que se hace es dar ese paso a paso que debe cumplir el programa para poder solucionar el problema por el cual fue creado. Puede que ese paso a paso no sea precisamente el que realizaba la persona de forma manual, sino que requiera de herramientas de integración como bases de datos y de esta forma, aunque se difiera en el proceso podamos lograr los resultados esperados.

La automatización tiene muchos beneficios importantes, entre ellos la reducción de costos operativos, mejoras en la calidad, velocidad en los procesos y mejora continua. Es importante contabilizar cada uno de los factores y que sirvan de indicadores para evaluar los proyectos de automatización llevados a cabo para establecer caso de negocio y conocer las ganancias de este.

Este proyecto utilizó un tipo de automatización muy utilizado en el mercado para la analítica de datos y son los Pipelines de datos. Estos “se caracterizan por definir el conjunto de pasos o fases y las tecnologías involucradas en un proceso de movimiento o procesamiento de datos” (Fernandez, 2024). Este tipo de tecnología genera disponibilidad de la información en los tiempos y plataformas necesarios para su consumo.

Generalidades de la Empresa y sus Procesos

Grupo Bancolombia

El grupo Bancolombia tiene como propósito general, promover el desarrollo sostenible para lograr el bienestar de todos, su contribución es realizada mediante una amplia gama de productos y servicios financieros que permiten apalancar los sueños y deseos de sus clientes como lo son cuentas, depósitos, servicios transaccionales, créditos de consumo, comerciales, de vivienda, microcréditos y plataformas como A la Mano y Nequi. Grupo Bancolombia. (2024).

El grupo Bancolombia tiene presencia en Colombia, Panamá, Guatemala y el salvador, cuenta con más de 34.000 empleados incluido el comité directivo que está encabezado por Juan Carlos Mora Uribe (presidente del grupo), Aimeé Sentmat de Grimaldo (presidenta Banistmo), Rafael Barraza Domínguez (presidente Bancoagrícola), Federico Bolaños Coloma (presidente Bam). Grupo Bancolombia (2024).

Seguidamente están las vicepresidencias que atienen los diferentes puntos de interés del banco, las de Colombia responden directamente al presidente del grupo y las de otros países al presidente asignado. La encargada de desarrollar y mantener un portafolio de productos y servicios del banco es la vicepresidencia encabezada por Liliana Vásquez Uribe, uno de los productos que tiene bajo su gobierno es el inmobiliario, y es en este equipo donde se diseña e implementa todo el proceso de gestión de solicitudes de Crédito Hipotecario y Leasing Habitacional. Grupo Bancolombia. (2024).

Proceso de Gestión de Solicitudes de Crédito Hipotecario y Leasing Habitacional en Bancolombia

El proceso de solicitudes de crédito hipotecario y leasing habitacional tiene por objetivo atender las solicitudes de crédito u operaciones de solución inmobiliaria que ingresen al banco,

desde la radicación hasta su desembolso. Para que un crédito hipotecario o leasing habitacional pueda ser desembolsado, deben pasar por las siguientes fases: asesoría y venta; gestión documental; radicación de solicitudes; estudio y decisión; informar inmueble; seguros: validación de garantías: estudios jurídicos y avalúo; ratificación, escrituración; y finalmente, desembolso.

Figura 1

Etapas del proceso de gestión de solicitudes de crédito hipotecario y leasing habitacional en Bancolombia.

Fuente. Bancolombia, 2024.

En el proceso participan constructoras, inmobiliarias, agencias de seguros, abogados externos, firmas valuadoras y notarias. El banco debe velar porque cada uno de los actores del proceso pueda ejecutar su labor de manera que para el cliente sea fácil, amigable y rápido. Con esto se garantiza una experiencia ganadora.

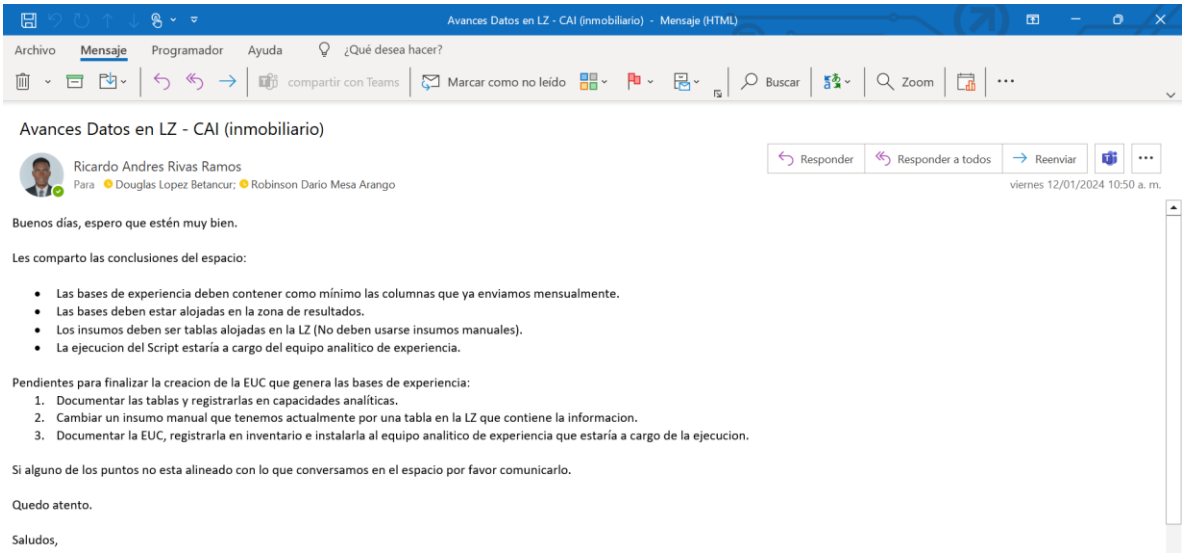
Desarrollo de la Herramienta para la Automatización del Proceso de Generación de Bases de Datos Para Medición de la Experiencia de Usuario.

Conocer la necesidad de la información del equipo de experiencia de clientes del grupo Bancolombia:

Separe un espacio con el equipo para analizar porque se comparte la información y que implicaciones tiene un cambio en el proceso.

Figura 2

Avances Datos en LZ - CAI



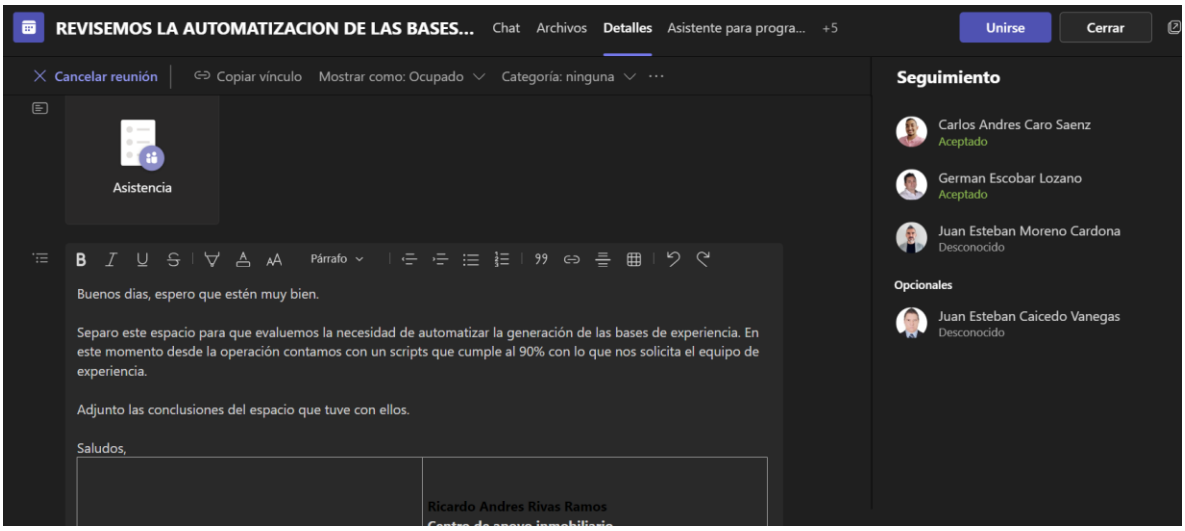
Nota. La ilustración muestra las conclusiones del espacio.

Identificación detallada de los requisitos específicos del equipo de solución inmobiliaria del Grupo Bancolombia:

Separe un espacio con el equipo de evolución del servicio para revisar si el prototipo que había planteado inicialmente era viable y los posibles ajustes que se debían realizar al proyecto.

Figura 3

Revisión de la automatización de las bases de experiencia

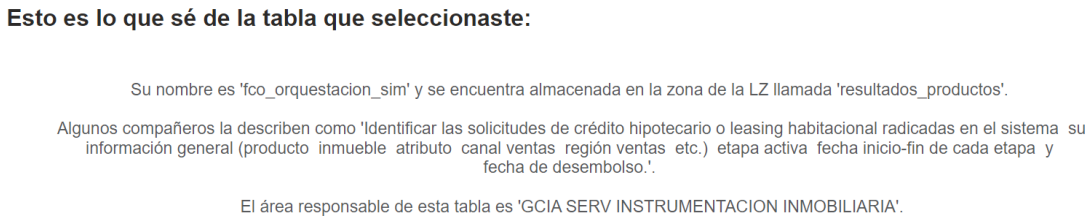


Nota. La ilustración detalla la solución del quipo inmobiliario Bancolombia (2024)

Identificar la FCO (Fuente Corporativa Oficial) que contiene la información necesaria para seleccionar a los clientes que se les enviaran las encuestas de experiencia.

Figura 4

FCO que contiene información



Fuente. Tablero de tablas lz powerbi Bancolombia, 2024

Desarrollo de la arquitectura del sistema, considerando las tecnologías y herramientas más adecuadas.

Para el desarrollo de la herramienta se escogió Python por ser un lenguaje de programación simple y potente en el manejo de datos, SQL para las consultas en bases de datos y Word para el manual de usuario.

Archivos y su Funcionalidad

setup.cfg: configura varias opciones de setuptools, incluyendo el versionamiento del paquete utilizando versioneer.

Figura 5

Setup.cfg

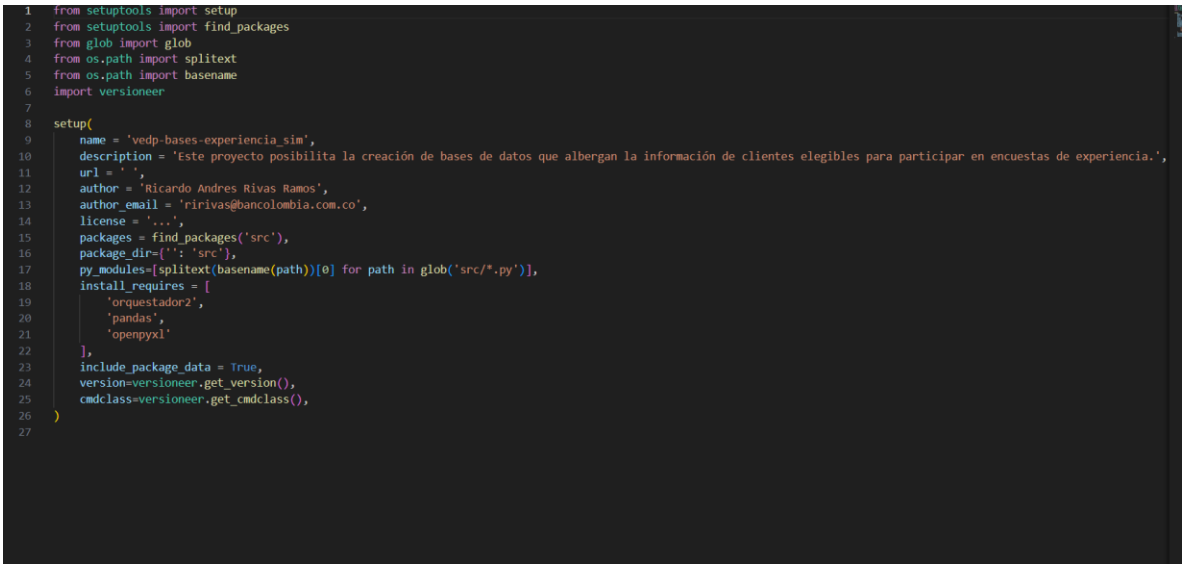
```
[versioneer]
VCS = git
style = pep440
versionfile_source = src/generador_bases_exp/_version.py
versionfile_build = generador_bases_exp/_version.py
tag_prefix = ''

```

Fuente. Archivo Setup.cfg, 2024

setup.py: es el archivo de configuración principal del paquete. Define metadata del paquete como el nombre, descripción, autor, dependencias, y la versión que se extrae utilizando versioneer.

Figura 6

Setup.py



```python
from setuptools import setup
from setuptools import find_packages
from glob import glob
from os.path import splitext
from os.path import basename
import versioneer

setup(
    name = 'vedp-bases-experiencia_sim',
    description = 'Este proyecto posibilita la creación de bases de datos que albergan la información de clientes elegibles para participar en encuestas de experiencia.',
    url = ' ',
    author = 'Ricardo Andres Rivas Ramos',
    author_email = 'ririvas@bancolombia.com.co',
    license = '...',
    packages = find_packages('src'),
    package_dir={'': 'src'},
    py_modules=[splitext(basename(path))[0] for path in glob('src/*.py')],
    install_requires = [
        'orquestador2',
        'pandas',
        'openpyxl'
    ],
    include_package_data = True,
    version=versioneer.get_version(),
    cmdclass=versioneer.get_cmdclass(),
)

```

Fuente. Archivo setup.py, 2024

Manejo de Versiones con versioneer para asegurar un control adecuado sobre las versiones del paquete, se utiliza versioneer. Este esquema de versionamiento se basa en tags de Git y sigue el estándar major.minor.patch. A continuación, se describe cómo se integra versioneer en el proyecto a través del archivo setup.py:

Figura 7

Integración versioneer en el proyecto

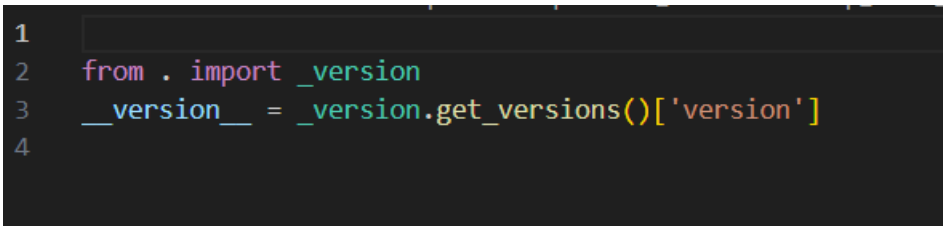
```python
from setuptools import setup
from setuptools import find_packages
from glob import glob
from os.path import splitext
from os.path import basename
import versioneer

setup(
    name = 'vedp-bases-experiencia_sim',
    description = 'Este proyecto posibilita la creación de bases de datos que albergan la información de clientes elegibles para participar en encuestas de experiencia.',
    url = ' ',
    author = 'Ricardo Andres Rivas Ramos',
    author_email = 'ririvas@bancolombia.com.co',
    license = '...',
    packages = find_packages('src'),
    package_dir={'': 'src'},
    py_modules=[splitext(basename(path))[0] for path in glob('src/*.py')],
    install_requires = [
        'orquestador2',
        'pandas',
        'openpyxl'
    ],
    include_package_data = True,
    version=versioneer.get_version(),
    cmdclass=versioneer.get_cmdclass(),
)
```

Nota. Fuente. Archivo setup.py, 2024

__init__.py: Un archivo __init__.py en un directorio de Python indica que el directorio debe considerarse como un paquete Python, en este caso su uso también implica la primera ejecución que haría el paquete cuando se instala, obtener la versión.

Figura 8

Archivo_init_py



```python
from . import _version
__version__ = _version.get_versions()['version']
```

Fuente. Archivo _init_.py (2024)

_version.py: se utiliza para almacenar información sobre la versión de un paquete Python de manera específica y estructurada.

Módulos.py: contiene todas las funciones del proyecto que serán ejecutadas de forma organizada por el orquestador.

Static: la carpeta static está diseñada para contener archivos estáticos que no requieren ser generados dinámicamente por el servidor de aplicaciones en cada solicitud.

Config.json: se utiliza típicamente para almacenar configuraciones específicas de la aplicación de manera estructurada y fácilmente accesible. Este archivo es especialmente útil cuando se necesitan parametrizar ciertos aspectos de la aplicación que pueden cambiar entre diferentes entornos (desarrollo, prueba, producción) o entre diferentes instalaciones del mismo software.

Sql: la carpeta sql contiene todas las consultas que el programa realiza al motor de bases de datos.

Excel: la carpeta Excel contiene los insumos de Excel que se cargan al programa para su ejecución.

1. Creación de módulos y consultas necesarias para el correcto funcionamiento de la herramienta.

Después de haber generado la estructura correcta para un proyecto de analítica, es necesario implementar toda la lógica del negocio necesaria para poder obtener los resultados esperados. Para esto es importante modificar una parte esencial del proyecto y son las consultas a las bases de datos, a continuación, se explica cómo fue el desarrollo de cada una de estas y como contribuye cada una a los objetivos del proyecto:

000_ESQUEMA

El script SQL crea dos tablas dentro de un esquema o base de datos (`{zona_resultados_r2}`). A continuación, se desglosa cada parte del script y se explica cómo contribuye al desarrollo de un proyecto

Explicación del Script SQL

Primera Tabla: `base_experiencia_desembolsos`

```sql
CREATE TABLE IF NOT EXISTS {zona_resultados_r2}.base_experiencia_desembolsos (
    numero_radicado varchar(25),
    ult_f_terminacion timestamp,
    instrumentacion varchar(100),
    producto_insumo varchar(20),
    idp_tipodocumento string,
    tipo_doc varchar(50),
    num_doc int,
    nombrecompleto varchar(100),
    tipo_inmueble varchar(20),
    producto_origen varchar(20),
    constructoragerenciada varchar(20),
    producto varchar(100),
    atributo_producto varchar(50),
    segm varchar(200),
    colomex string,
```

```
    email1 string,
    email1_valido string,
    email2 string,
    email2_valido string,
    valida_cruce varchar(50),
    fecha_corte timestamp,
    year int,
    month int,
    ingestion_day int
);
```

CREATE TABLE IF NOT EXISTS`: esta parte del comando crea una tabla si no existe ya en la base de datos `{zona_resultados_r2}`. Este enfoque previene errores si la tabla ya está definida.

base_experiencia_desembolsos`: es el nombre de la tabla que se crea dentro del esquema `{zona_resultados_r2}`.

Los campos que siguen (`numero_radicado`, `ult_f_terminacion`, `instrumentacion`, etc.) definen la estructura de la tabla:

Cada campo tiene un tipo de datos específico (`varchar`, `timestamp`, `int`, etc.) que define el tipo de datos que puede almacenar (por ejemplo, cadenas de caracteres, fechas, números enteros).

La longitud de los campos `varchar` (como `varchar(25)`, `varchar(100)`) especifica la máxima cantidad de caracteres que puede contener cada valor.

Segunda Tabla: `base_experiencia_abogados`

```sql
CREATE TABLE IF NOT EXISTS {zona_resultados_r2}.base_experiencia_abogados (
    numero_radicado varchar(25),
    instrumentacion varchar(100),
    nombre_abogado varchar(100),
    ult_f_terminacion timestamp,
    idp_tipodocumento string,
    tipo_doc varchar(50),
    num_doc int,
    nombrecompleto varchar(100),
    tipo_inmueble varchar(20),
    producto_origen varchar(20),
    segm varchar(200),
    tipo_doc_abogado int,
    doc_abogado int,
    producto varchar(100),
    constructoragerenciada varchar(20),
    atributo_producto varchar(50),
    colomex string,
    compracartera varchar(20),
    email1 string,
```

```
    email1_valido string,
    email2 string,
    email2_valido string,
    valida_cruce varchar(50),
    fecha_corte timestamp,
    year int,
    month int,
    ingestion_day int
);
```

- Este comando también utiliza `CREATE TABLE IF NOT EXISTS` para crear la tabla `base_experiencia_abogados` en el esquema `{zona_resultados_r2}`, si no existe previamente.

- Los campos definidos (`numero_radicado`, `instrumentacion`, `nombre_abogado`, etc.) tienen tipos de datos similares a los de la tabla anterior. Sin embargo, hay algunos campos adicionales específicos para esta tabla, como `tipo_doc_abogado`, `doc_abogado`, y `compracartera`.

Contribución del Esquema al Desarrollo del Proyecto

Definición de Estructura de Datos

Establece la estructura y tipos de datos que se utilizarán para almacenar información relacionada con desembolsos (`base_experiencia_desembolsos`) y abogados (`base_experiencia_abogados`).

Consistencia en la Información

Asegura que los datos almacenados en las tablas sigan un formato y tipo específicos, facilitando la consulta y manipulación de datos de manera coherente a lo largo del desarrollo del proyecto.

Facilita la Integración con la Aplicación

Permite que la aplicación Python acceda y manipule los datos almacenados en estas tablas mediante consultas SQL, proporcionando una capa de almacenamiento estructurado para la información crítica del proyecto.

Soporte para Operaciones CRUD

Facilita la creación (`INSERT`), lectura (`SELECT`), actualización (`UPDATE`) y eliminación (`DELETE`) de datos, lo cual es fundamental para cualquier aplicación que maneje información transaccional o de gestión.

Escalabilidad y Mantenimiento

La definición de estas tablas en un script SQL centralizado y versionado facilita la gestión de cambios en la estructura de la base de datos a medida que el proyecto evoluciona y se amplía.

En resumen, el uso de scripts SQL como este en proyectos Python proporciona una base sólida y organizada para el almacenamiento y gestión de datos, promoviendo una estructura de desarrollo ordenada y eficiente.

001_EXTRAER_INFO

El script SQL realiza dos operaciones principales: crea tablas en una base de datos (`{zona_procesos}`) y pobla esas tablas utilizando datos provenientes de consultas CTE (Common Table Expressions) desde otras fuentes de datos

(`{zona_resultados}.fco_orquestacion_sim` y `{master_data}`). A continuación, se desglosa cada parte del script y explicaré su función en el contexto del desarrollo del proyecto:

Creación de Tablas

1. Tabla `base_experiencia_abogados`

```sql
DROP TABLE IF EXISTS {zona_procesos}.base_experiencia_abogados PURGE;
CREATE TABLE IF NOT EXISTS {zona_procesos}.base_experiencia_abogados AS
WITH orq AS (
    SELECT
        numero_radicado,
        f_fin_avaluo,
        CAST(f_fin_escrituracion AS TIMESTAMP) AS f_fin_escrituracion,
        fechadedesembolsoinicial,
        tipo_doc,
        celula_estudio,
        CAST(num_doc AS INT) AS num_doc,
        nombrecompleto,
        tipo_inmueble,
        producto_origen,
        tipo_proyecto,
        producto,
```

```
        atributo_producto,
        instrumentacion,
        compracartera,
        nombre_abogado,
        constructoragerenciada
    FROM
        {zona_resultados}.fco_orquestacion_sim
    WHERE
        year = {ing_orquestacion[year]}
        AND month = {ing_orquestacion[month]}
        AND ingestion_day = {ing_orquestacion[day]}
),
mcd AS (
    SELECT
        CAST(num_doc AS INT) AS num_doc,
        cod_tipo_doc,
        segm,
        email1,
        email1_valido,
        email2,
        email2_valido
    FROM
        {master_data}
```

```
    WHERE
        year = {ing_master_data[year]}
        AND month = {ing_master_data[month]}
        AND ingestion_day = {ing_master_data[day]}
)
SELECT
    orq.numero_radicado,
    orq.instrumentacion,
    orq.nombre_abogado,
    orq.f_fin_escrituracion AS ult_f_terminacion,
    mcd.cod_tipo_doc AS idp_tipodocumento,
    orq.tipo_doc,
    orq.num_doc,
    orq.nombrecompleto,
    orq.tipo_inmueble,
    orq.producto_origen,
    mcd.segm,
    CAST(NULL AS INT) AS tipo_doc_abogado,
    CAST(NULL AS INT) AS doc_abogado,
    orq.producto,
    orq.constructoragerenciada,
    orq.atributo_producto,
    CASE
```

```
            WHEN orq.celula_estudio = 'CELULA PARA ESTUDIO DE CREDITO
COLOMBIANOS EN EL EXTERIOR' THEN 'Colomex'
            ELSE NULL
        END AS colomex,
        orq.compracartera,
        mcd.email1,
        mcd.email1_valido,
        mcd.email2,
        mcd.email2_valido,
        CAST('Cruzo con MCD' AS VARCHAR(50)) AS valida_cruce,
        CAST(current_timestamp() AS TIMESTAMP) AS fecha_corte,
        CAST(year(current_date()) AS INT) AS year,
        CAST(month(current_date()) AS INT) AS month,
        CAST(day(current_date()) AS INT) AS ingestion_day
    FROM
        orq
    LEFT JOIN mcd ON orq.num_doc = mcd.num_doc
    WHERE
        orq.tipo_proyecto IN ('USADO', 'PROYECTO NO GERENCIADO NO
FINANCIADO')
        AND orq.atributo_producto NOT IN ('DACION EN PAGO', 'CESION DE
DERECHOS LEASING')
        AND tipo_inmueble NOT IN ('SIN DESCRIPCION')
```

```
    AND orq.compracartera NOT IN ('SI')
    AND NOT (mcd.email1 IS NULL AND mcd.email2 IS NULL)
    AND NOT (mcd.email1_valido = 'N' AND mcd.email2_valido = 'N')
    AND year(orq.f_fin_escrituracion) = (SELECT year(add_months(current_timestamp(),
-1)))
    AND month(orq.f_fin_escrituracion) = (SELECT
month(add_months(current_timestamp(), -1)));
```

2. Tabla `base_experiencia_desembolsos`

```sql
DROP TABLE IF EXISTS {zona_procesos}.base_experiencia_desembolsos PURGE;
CREATE TABLE IF NOT EXISTS {zona_procesos}.base_experiencia_desembolsos AS
WITH orq AS (
  SELECT
    numero_radicado,
    f_fin_avaluo,
    CAST(f_fin_escrituracion AS TIMESTAMP) AS f_fin_escrituracion,
    cast(fechadedesembolsoinicial AS TIMESTAMP) AS fechadedesembolsoinicial,
    tipo_doc,
    celula_estudio,
    CAST(num_doc AS INT) AS num_doc,
```

```
        nombrecompleto,
        tipo_inmueble,
        producto_origen,
        tipo_proyecto,
        producto,
        atributo_producto,
        instrumentacion,
        compracartera,
        nombre_abogado,
        constructoragerenciada
    FROM
        {zona_resultados}.fco_orquestacion_sim
    WHERE
        year = {ing_orquestacion[year]}
        AND month = {ing_orquestacion[month]}
        AND ingestion_day = {ing_orquestacion[day]}
),
mcd AS (
    SELECT
        CAST(num_doc AS INT) AS num_doc,
        cod_tipo_doc,
        segm,
        email1,
```

```
        email1_valido,
        email2,
        email2_valido
    FROM
        {master_data}
    WHERE
        year = {ing_master_data[year]}
        AND month = {ing_master_data[month]}
        AND ingestion_day = {ing_master_data[day]}
)
SELECT
    orq.numero_radicado,
    orq.fechadedesembolsoinicial AS ult_f_terminacion,
    orq.instrumentacion,
    orq.producto_origen AS producto_insumo,
    mcd.cod_tipo_doc AS idp_tipodocumento,
    orq.tipo_doc,
    orq.num_doc,
    orq.nombrecompleto,
    orq.tipo_inmueble,
    orq.producto_origen,
    orq.constructoragerenciada,
    orq.producto,
```

```
        orq.atributo_producto,
        mcd.segm,
        CASE
            WHEN orq.celula_estudio = 'CELULA PARA ESTUDIO DE CREDITO
COLOMBIANOS EN EL EXTERIOR' THEN 'Colomex'
            ELSE NULL
        END AS colomex,
        mcd.email1,
        mcd.email1_valido,
        mcd.email2,
        mcd.email2_valido,
        CAST('Cruzo con MCD' AS VARCHAR(50)) AS valida_cruce,
        CAST(current_timestamp() AS TIMESTAMP) AS fecha_corte,
        CAST(year(current_date()) AS INT) AS year,
        CAST(month(current_date()) AS INT) AS month,
        CAST(day(current_date()) AS INT) AS ingestion_day
    FROM
        orq
    LEFT JOIN mcd ON orq.num_doc = mcd.num_doc
    WHERE
        tipo_inmueble NOT IN ('SIN DESCRIPCION')
        AND NOT (mcd.email1 IS NULL AND mcd.email2 IS NULL)
        AND NOT (mcd.email1_valido = 'N' AND mcd.email2_valido = 'N')
```

```
        AND year(fechadedesembolsoinicial) = (SELECT
year(add_months(current_timestamp(), -1)))
        AND month(fechadedesembolsoinicial) = (SELECT
month(add_months(current_timestamp(), -1)));
```

Explicación de Cada Comando y Funcionalidad

`DROP TABLE IF EXISTS {zona_procesos}.base_experiencia_abogados PURGE;`

Este comando elimina la tabla `base_experiencia_abogados` si existe previamente en el esquema `{zona_procesos}`. El uso de `PURGE` asegura que cualquier objeto relacionado también se elimine definitivamente.

`CREATE TABLE IF NOT EXISTS {zona_procesos}.base_experiencia_abogados AS ...`

Crea una nueva tabla `base_experiencia_abogados` en el esquema `{zona_procesos}` si no existe previamente.

Utiliza una consulta CTE (Common Table Expression) para obtener datos de la tabla `{zona_resultados}.fco_orquestacion_sim` y `{master_data}`.

Los datos seleccionados se transforman y se insertan en la nueva tabla `base_experiencia_abogados`, asegurando que los tipos de datos y las transformaciones necesarias se realicen correctamente.

Consulta CTE `orq` y `mcd`:

La consulta `orq` selecciona datos de `{zona_resultados}.fco_orquestacion_sim`, aplicando conversiones de tipos (`CAST`) y condiciones (`WHERE`) para filtrar los datos según criterios específicos como el año, mes y día de ingestión.

La consulta `mcd` selecciona datos de `{master_data}` con conversiones y condiciones similares.

. `SELECT ...

FROM orq LEFT JOIN mcd ON ...`

Combina los resultados de las consultas `orq` y `mcd` utilizando un `LEFT JOIN` en `num_doc`, asegurando que los datos de `mcd` se incorporen cuando estén disponibles.

Condiciones `WHERE`:

Las condiciones filtran los datos según criterios específicos como tipo de proyecto, atributo de producto, tipo de inmueble, compracartera y validez de correos electrónicos (`email1`, `email2`).

Columnas Calculadas:

Las columnas como `valida_cruce`, `fecha_corte`, `year`, `month` y `ingestion_day` son calculadas con funciones como `CAST`, `current_timestamp()`, `year(current_date())`, `month(current_date())`, y `day(current_date())`, proporcionando valores estáticos o derivados de la fecha y hora actuales.

Contribución al Desarrollo del Proyecto

Primero, la automatización y procesamiento de datos: el script automatiza la creación de tablas y la carga de datos, facilitando el procesamiento y la transformación de datos complejos desde múltiples fuentes.

Segundo, la integración de datos: utiliza consultas CTE para integrar datos de varias fuentes (`{zona_resultados}.fco_orquestacion_sim` y `{master_data}`), asegurando que los datos estén disponibles y estructurados adecuadamente para análisis posteriores.

Tercero, mantenimiento de datos históricos: las condiciones `WHERE` que filtran por fechas (como `year()` y `month()`) aseguran que solo se incluyan datos relevantes para el período anterior al actual, ayudando en la gestión de datos históricos.

Cuarto, aseguramiento de calidad de datos: las condiciones `WHERE` también garantizan que solo se procesen datos válidos según criterios específicos, como la presencia de correos electrónicos válidos y la exclusión de ciertos tipos de proyectos o productos.

Quinto, eficiencia y rendimiento: el uso de funciones de fecha y consultas optimizadas (`LEFT JOIN`, condiciones `WHERE` específicas) mejora el rendimiento del procesamiento de datos, asegurando que las operaciones sean eficientes incluso con grandes volúmenes de datos.

En resumen, este script SQL no solo crea estructuras de tabla y carga datos, sino que también facilita la integración y procesamiento eficiente de datos complejos, contribuyendo significativamente al desarrollo y mantenimiento de un proyecto de análisis de datos o de sistemas de información en un entorno empresarial.

002_DROP_DUPLICATE

El script SQL realiza varias operaciones para gestionar duplicados y asegurar la integridad de los datos en las tablas `base_experiencia_desembolsos` y `base_experiencia_abogados`. A continuación, se desglosa cada sección y explicaré su propósito:

Sección 1: Eliminación de Duplicados en `base_experiencia_desembolsos`

```sql
WITH desem_t1 AS (
    SELECT
        numero_radicado,
        ult_f_terminacion,
```

instrumentacion,

producto_insumo,

idp_tipodocumento,

tipo_doc,

num_doc,

nombrecompleto,

tipo_inmueble,

producto_origen,

constructoragerenciada,

producto,

atributo_producto,

segm,

colomex,

email1,

email1_valido,

email2,

email2_valido,

valida_cruce,

fecha_corte,

year,

month,

ingestion_day,

```
            ROW_NUMBER() OVER(PARTITION BY numero_radicado ORDER BY
ult_f_terminacion DESC) AS rn
        FROM
            {zona_procesos}.base_experiencia_desembolsos
    )
    INSERT OVERWRITE TABLE {zona_procesos}.base_experiencia_desembolsos
    SELECT
        numero_radicado,
        ult_f_terminacion,
        instrumentacion,
        producto_insumo,
        idp_tipodocumento,
        tipo_doc,
        num_doc,
        nombrecompleto,
        tipo_inmueble,
        producto_origen,
        constructorageerenciada,
        producto,
        atributo_producto,
        segm,
        colomex,
        email1,
```

```
    email1_valido,
    email2,
    email2_valido,
    valida_cruce,
    fecha_corte,
    year,
    month,
    ingestion_day
FROM
    desem_t1
WHERE
    rn = 1;
```

Propósito: eliminar duplicados en la tabla `base_experiencia_desembolsos` basados en el campo `numero_radicado`. Utiliza una CTE (`desem_t1`) que asigna un número de fila (`rn`) para cada conjunto de registros agrupados por `numero_radicado`, ordenados por `ult_f_terminacion` en orden descendente. Luego, inserta los registros de `desem_t1` de nuevo en la tabla `base_experiencia_desembolsos`, seleccionando solo aquellos donde `rn = 1`, es decir, el registro más reciente (basado en `ult_f_terminacion`) para cada `numero_radicado`.

Sección 2: Eliminación de Duplicados en `base_experiencia_abogados`

```sql
WITH Abo_t1 AS (
```

```
SELECT
    numero_radicado,
    instrumentacion,
    nombre_abogado,
    ult_f_terminacion,
    idp_tipodocumento,
    tipo_doc,
    num_doc,
    nombrecompleto,
    tipo_inmueble,
    producto_origen,
    segm,
    tipo_doc_abogado,
    doc_abogado,
    producto,
    constructoragerenciada,
    atributo_producto,
    colomex,
    compracartera,
    email1,
    email1_valido,
    email2,
    email2_valido,
```

```
            valida_cruce,
            fecha_corte,
            year,
            month,
            ingestion_day,
            ROW_NUMBER() OVER(PARTITION BY numero_radicado ORDER BY
ult_f_terminacion DESC) AS rn
        FROM
            {zona_procesos}.base_experiencia_abogados
    )
    INSERT OVERWRITE TABLE {zona_procesos}.base_experiencia_abogados
    SELECT
        numero_radicado,
        instrumentacion,
        nombre_abogado,
        ult_f_terminacion,
        idp_tipodocumento,
        tipo_doc,
        num_doc,
        nombrecompleto,
        tipo_inmueble,
        producto_origen,
        segm,
```

```
    tipo_doc_abogado,
    doc_abogado,
    producto,
    constructoragerenciada,
    atributo_producto,
    colomex,
    compracartera,
    email1,
    email1_valido,
    email2,
    email2_valido,
    valida_cruce,
    fecha_corte,
    year,
    month,
    ingestion_day
FROM
    Abo_t1
WHERE
    rn = 1;
```

Propósito: eliminar duplicados en la tabla `base_experiencia_abogados` basados en el campo `numero_radicado`. Similar a la sección anterior, utiliza una CTE (`Abo_t1`) que asigna

un número de fila (`rn`) para cada conjunto de registros agrupados por `numero_radicado`, ordenados por `ult_f_terminacion` en orden descendente. Inserta los registros de `Abo_t1` de nuevo en la tabla `base_experiencia_abogados`, seleccionando solo aquellos donde `rn = 1`, es decir, el registro más reciente (basado en `ult_f_terminacion`) para cada `numero_radicado`.

Sección 3: Eliminación de Duplicados entre Bases

```sql
WITH duplic_t1 AS (
    SELECT
        t1.numero_radicado,
        t1.ult_f_terminacion,
        t1.instrumentacion,
        t1.producto_insumo,
        t1.idp_tipodocumento,
        t1.tipo_doc,
        t1.num_doc,
        t1.nombrecompleto,
        t1.tipo_inmueble,
        t1.producto_origen,
        t1.constructoragerenciada,
        t1.producto,
        t1.atributo_producto,
        t1.segm,
        t1.colomex,
```

```
        t1.email1,
        t1.email1_valido,
        t1.email2,
        t1.email2_valido,
        t1.valida_cruce,
        t1.fecha_corte,
        t1.year,
        t1.month,
        t1.ingestion_day,
        t2.numero_radicado AS base_abo
    FROM
        {zona_procesos}.base_experiencia_desembolsos t1
    LEFT JOIN
        {zona_procesos}.base_experiencia_abogados t2
    ON
        t1.numero_radicado = t2.numero_radicado
)
INSERT OVERWRITE TABLE {zona_procesos}.base_experiencia_desembolsos
SELECT
    numero_radicado,
    ult_f_terminacion,
    instrumentacion,
    producto_insumo,
```

```
    idp_tipodocumento,
    tipo_doc,
    num_doc,
    nombrecompleto,
    tipo_inmueble,
    producto_origen,
    constructoragerenciada,
    producto,
    atributo_producto,
    segm,
    colomex,
    email1,
    email1_valido,
    email2,
    email2_valido,
    valida_cruce,
    fecha_corte,
    year,
    month,
    ingestion_day
FROM
    duplic_t1
WHERE
```

base_abo IS NULL;

```

Propósito: eliminar duplicados entre las tablas `base_experiencia_desembolsos` y `base_experiencia_abogados` basados en el campo `numero_radicado`. Utiliza una CTE (`duplic_t1`) para unir las tablas `base_experiencia_desembolsos` (`t1`) y `base_experiencia_abogados` (`t2`) por `numero_radicado`. Inserta los registros de `duplic_t1` en `base_experiencia_desembolsos`, seleccionando solo aquellos donde `base_abo IS NULL`, lo que significa que no hay un registro correspondiente en `base_experiencia_abogados`.

En conclusión, estas secciones de código SQL, son parte de un proceso ETL (Extract, Transform, Load) que asegura la integridad de los datos, eliminando duplicados en las tablas `base_experiencia_desembolsos` y `base_experiencia_abogados`, así como entre ambas. Cada sección utiliza `ROW_NUMBER()` para identificar y retener solo los registros más recientes basados en ciertos criterios, contribuyendo así a mantener la consistencia de los datos y a optimizar su uso en análisis posteriores.

003_HISTORICO_FINAL

El código SQL que has proporcionado tiene como objetivo insertar registros nuevos en las tablas `base_experiencia_desembolsos` y `base_experiencia_abogados` en el esquema `{zona_resultados_r2}`, asegurándose de que no haya duplicados en relación con las tablas históricas (`base_experiencia_desembolsos_hist` y `base_experiencia_abogados` en `{zona_resultados_r2}`). Aquí está el desglose y explicación de cada parte:

Sección 1: Insertar nuevos registros en `base_experiencia_desembolsos`

```sql
WITH des_t1 AS (

```
SELECT
    t1.numero_radicado,
    t1.ult_f_terminacion,
    t1.instrumentacion,
    t1.producto_insumo,
    t1.idp_tipodocumento,
    t1.tipo_doc,
    t1.num_doc,
    t1.nombrecompleto,
    t1.tipo_inmueble,
    t1.producto_origen,
    t1.constructoragerenciada,
    t1.producto,
    t1.atributo_producto,
    t1.segm,
    t1.colomex,
    t1.email1,
    t1.email1_valido,
    t1.email2,
    t1.email2_valido,
    t1.valida_cruce,
    t1.fecha_corte,
    t1.year,
```

```
        t1.month,
        t1.ingestion_day,
        t2.numero_radicado AS hist
    FROM
        {zona_procesos}.base_experiencia_desembolsos t1
    LEFT JOIN
        {zona_resultados_r2}.base_experiencia_desembolsos_hist t2
    ON
        t1.numero_radicado = t2.numero_radicado
)
INSERT INTO TABLE {zona_resultados_r2}.base_experiencia_desembolsos
SELECT
    numero_radicado,
    ult_f_terminacion,
    instrumentacion,
    producto_insumo,
    idp_tipodocumento,
    tipo_doc,
    num_doc,
    nombrecompleto,
    tipo_inmueble,
    producto_origen,
    constructoragerenciada,
```

```
    producto,
    atributo_producto,
    segm,
    colomex,
    email1,
    email1_valido,
    email2,
    email2_valido,
    valida_cruce,
    fecha_corte,
    year,
    month,
    ingestion_day
FROM
    des_t1
WHERE
    hist IS NULL;
```

Propósito: insertar nuevos registros en la tabla `base_experiencia_desembolsos` en el esquema `{zona_resultados_r2}`, evitando duplicados en la tabla histórica `base_experiencia_desembolsos_hist`. Utiliza una CTE (`des_t1`) para unir `base_experiencia_desembolsos` (`t1`) con `base_experiencia_desembolsos_hist` (`t2`) por

`numero_radicado`. Inserta en `base_experiencia_desembolsos` aquellos registros de `des_t1` donde `hist IS NULL`, lo que indica que no hay un registro correspondiente en la tabla histórica.

Sección 2: Insertar nuevos registros en `base_experiencia_abogados`

```sql
WITH abo_t1 AS (
    SELECT
        t1.numero_radicado,
        t1.instrumentacion,
        t1.nombre_abogado,
        t1.ult_f_terminacion,
        t1.idp_tipodocumento,
        t1.tipo_doc,
        t1.num_doc,
        t1.nombrecompleto,
        t1.tipo_inmueble,
        t1.producto_origen,
        t1.segm,
        t1.tipo_doc_abogado,
        t1.doc_abogado,
        t1.producto,
        t1.constructoragerenciada,
        t1.atributo_producto,
        t1.colomex,
```

```
        t1.compracartera,
        t1.email1,
        t1.email1_valido,
        t1.email2,
        t1.email2_valido,
        t1.valida_cruce,
        t1.fecha_corte,
        t1.year,
        t1.month,
        t1.ingestion_day,
        t2.numero_radicado AS hist
    FROM
        {zona_procesos}.base_experiencia_abogados t1
    LEFT JOIN
        {zona_resultados_r2}.base_experiencia_abogados t2
    ON
        t1.numero_radicado = t2.numero_radicado
)
INSERT INTO TABLE {zona_resultados_r2}.base_experiencia_abogados
SELECT
    numero_radicado,
    instrumentacion,
    nombre_abogado,
```

ult_f_terminacion,

idp_tipodocumento,

tipo_doc,

num_doc,

nombrecompleto,

tipo_inmueble,

producto_origen,

segm,

tipo_doc_abogado,

doc_abogado,

producto,

constructoragerenciada,

atributo_producto,

colomex,

compracartera,

email1,

email1_valido,

email2,

email2_valido,

valida_cruce,

fecha_corte,

year,

month,

```
    ingestion_day
FROM
    abo_t1
WHERE
    hist IS NULL;
```

Propósito: insertar nuevos registros en la tabla `base_experiencia_abogados` en el esquema `{zona_resultados_r2}`, evitando duplicados en la tabla histórica `base_experiencia_abogados`. Utiliza una CTE (`abo_t1`) para unir `base_experiencia_abogados` (`t1`) con `base_experiencia_abogados` (`t2`) por `numero_radicado`. Inserta en `base_experiencia_abogados` aquellos registros de `abo_t1` donde `hist IS NULL`, lo que indica que no hay un registro correspondiente en la tabla histórica.

Explicación General

Estas consultas SQL, utilizan joins izquierdos (`LEFT JOIN`) para comparar los registros entre las tablas de origen (`base_experiencia_desembolsos` y `base_experiencia_abogados` en `{zona_procesos}`) y las tablas históricas (`base_experiencia_desembolsos_hist` y `base_experiencia_abogados` en `{zona_resultados_r2}`). Luego, insertan solo los registros nuevos (que no tienen un equivalente en la tabla histórica) en las tablas de resultados (`base_experiencia_desembolsos` y `base_experiencia_abogados` en `{zona_resultados_r2}`).

Esto asegura que las tablas de resultados estén actualizadas con la información más reciente y sin duplicados, optimizando así la integridad y la eficiencia del proceso ETL.

Para que estos scripts SQL puedan ser orquestados de forma correcta y ordenada en el motor de bases de datos, es necesario que se den las instrucciones adecuadas a través de los

módulos de Python estos módulos a su vez se alimentan del archivo config para poder ejecutarse en los entornos requeridos por el usuario de la aplicación.

Config.json

“{

"global": {

"dsn": "IMPALA_PROD",

"username": "ririvas"

},

"UltIngestions": {

"tablas_ingestions": {

"ing_master_data": "resultados_vspc_clientes.master_customer_data",

"ing_orquestacion":"resultados_productos.fco_orquestacion_sim"

},

"zona_procesos": "proceso_productos",

"zona_resultados": "resultados_productos",

"zona_resultados_r2": "proceso_productos",

"master_data": "resultados_vspc_clientes.master_customer_data",

"orquestacion":"resultados_productos.fco_orquestacion_sim"},

"EjeSQL1": {

"archivos": [

"000_ESQUEMA.sql","001_EXTRAER_INFO.sql","002_DROP_DUPLICATE.sql","003_HISTORICO_FINAL.sql"

]

},

"EXPOR_BASES": {

}

}”

Definición de las variables del JSON:

global.dsn: nombre de la conexión de Impala.

global.username: nombre de usuario para la conexión.

UltIngestions.tablas_ingestions: definición de las tablas de ingestión.

UltIngestions.zona_procesos,UltIngestions.zona_resultados, UltIngestions.zona_resultados_r2: tsquemas o zonas utilizadas en las consultas a la BD.

UltIngestions.master_data, UltIngestions.orquestacion: tablas específicas para master data y orquestación.

EjeSQL1.archivos: lista de archivos SQL a ejecutar.

Construcción de las Consultas SQL

Se utilizarán las tablas definidas en UltIngestions.tablas_ingestions para las operaciones de ingestión. Los esquemas UltIngestions.zona_procesos y UltIngestions.zona_resultados_r2

serán los objetivos para insertar los datos procesados. Las consultas SQL se basarán en los archivos especificados en EjeSQL1.archivos.

Implementación en SQL

Los archivos SQL especificados en EjeSQL1.archivos, contienen las operaciones descritas previamente (extracción, eliminación de duplicados, manejo de históricos, etc.) y las variables que se encuentran entre corchetes ({}) en estos archivos son las que se remplazarían por los parámetros configurados en este archivo. json.

Módulos

```
“# -*- coding: utf-8 -*-
"""
Created on Mon Feb  15 11:45:06 2024

@author: ririvas
"""

from orquestador2.step import Step
import pandas as pd
from datetime import datetime , timedelta
from pytz import timezone

class UltIngestions(Step):
```

```
        def ejecutar(self):
            """ Busca la ultima ingestión de cada tabla proporcionada en la lista
'tablas_ingestions' de la configuración del Step """
            log = self.getLog()
            helper = self.getHelper()
            cfg = self.getStepConfig()
            timeZone = timezone("America/Bogota")
            now = datetime.now(timeZone)
            params = {}
            params.update(cfg)
            params.update({tabla:helper.obtener_ultima_ingestion(ruta) for tabla,ruta in
params["tablas_ingestions"].items() if tabla not in params})
            params['f_corte'] = now.strftime('%Y%m%d')
           #params['f_corte_ant'] = params['f_corte']-1
            params['anho'] = int(now.strftime('%Y'))
            params['mes'] = int(now.strftime('%m'))
            params['dia'] = int(now.strftime('%d'))
            self.setPayload(params)


    class EjeSQL1(Step): #ejecuta la validación de ingestiones y la extracción de datos
        def ejecutar(self):
            log = self.getLog()
            cfg = self.getStepConfig()
```

```
        params = self.getPayload()
        helper = self.getHelper()
        lista_sql = cfg["archivos"]
        params['cargar']='Si'
#         #log.info("Parametros calculados: {0}".format(params))
        if params['ing_orquestacion']['month'] == params['mes']:
            for file in lista_sql:
                titulo=log._line()+'\n'+log._titulo(file)+'\n'+log._line()
                log.print(titulo)
                archivo = self.getSQLPath() + file

                helper.ejecutar_archivo(archivo, params)
        else:
            params['cargar']='No'

        self.setPayload(params)
    class EXPOR_BASES(Step): #ejecuta la validación de ingestiones y la extracción de
datos
        def ejecutar(self):
            log = self.getLog()
            cfg = self.getStepConfig()
            params = self.getPayload()
            helper = self.getHelper()
```

```
            self.setPayload(params)
            if params['cargar']=='No':
                print("La fuente actual no contiene los la informacion para generar las bases")
            else:
                log.info("Exportar resultados")
                abo=helper.obtener_dataframe(f"SELECT * FROM
proceso_productos.base_experiencia_abogados where year={params['anho']} and
month={params['mes']} and ingestion_day={params['dia']}")
                abo.to_excel("EXPABOG_BASE.xlsx", index=False)
                abo.rename(columns={"instrumentacion":"INSTRUMENTACION INTERNA"})
                des=helper.obtener_dataframe(f"SELECT * FROM
proceso_productos.base_experiencia_desembolsos where year={params['anho']} and
month={params['mes']} and ingestion_day={params['dia']}")
                des.to_excel("EXPDESEMB_BASE.xlsx", index=False)
            #ava=helper.obtener_dataframe(f"SELECT * FROM
proceso_productos.base_experiencia_avaluo where year={params['anho']} and
month={params['mes']} and ingestion_day={params['dia']}")
            #ava.rename(columns={"instrumentacion":"Equipo instrumentacion",
"firma":"Firma Valuadora"})
            #ava.to_excel("EXPAVALUO_BASE.xlsx", index=False)


        class EjeSparky(Step):
            def ejecutar(self):
```

```
spark = self.getSparky()
helper = self.getHelper()

ruta = self.getFolderPath() + "excel/consolidado de asignaciones.xlsx"

helper.ejecutar_consulta("drop table if exists proceso_productos.consolidado_asignaciones_avaluo")
df=pd.read_excel(ruta,usecols=["CASO","FIRMA"])
spark.subir_df(df, "consolidado_asignaciones_avaluo", zona='proceso_productos', modo="overwrite")”
```

1. Módulo `UltIngestions.py`

Este módulo define la clase `UltIngestions`, que es un paso en el flujo de trabajo de ETL. Aquí se busca la última ingestión de cada tabla especificada en la configuración del paso y se preparan los parámetros para pasos posteriores.

Propósito: encontrar la última ingestión de datos para cada tabla listada en `tablas_ingestions`.

Conexión con otros módulos:

- Conexión lógica: los parámetros calculados (como la fecha de corte actual) se pasan al siguiente paso del flujo.
- Módulo de configuración (`config.json`): usa la configuración proporcionada para determinar las tablas y rutas de ingestión.
- Helper (`helper.py`): utiliza métodos del helper para obtener la última ingestión de cada tabla.

2. Modulo `EjeSQL1.py`

Define la clase `EjeSQL1`, que ejecuta validaciones de ingestión y extracción de datos a través de módulos SQL especificados en la configuración.

Propósito: validar si las tablas tienen ingestiones recientes y ejecutar Módulos SQL para extraer datos si es necesario.

Conexión con otros Módulos:

- Conexión lógica: verifica la fecha de ingestión (`ing_orquestacion['month']`) y ejecuta los Modulos SQL si corresponde.
- Módulos SQL (`000_ESQUEMA.sql`, etc.): Se ejecutan estos Módulos SQL para extraer datos de las fuentes relevantes.
- Helper (`helper.py`): Se utiliza para ejecutar Módulos SQL y pasar parámetros entre pasos.

3. Módulo `EXPOR_BASES.py`

Define la clase `EXPOR_BASES`, que exporta los resultados de las bases de datos a módulos Excel si los datos están disponibles para el mes y día actuales.

Propósito: exportar los resultados de las tablas `base_experiencia_abogados` y `base_experiencia_desembolsos` si la fecha de ingestión coincide con el mes actual.

Conexión con otros módulos:

- Conexión lógica: verifica si `params['cargar']` es `'Si'` para decidir si exportar los datos.
- Módulos Excel (`EXPABOG_BASE.xlsx`, `EXPDESEMB_BASE.xlsx`, etc.): Se generan estos módulos Excel con los datos extraídos.

- Helper (`helper.py`): Se utiliza para obtener los datos de las tablas específicas basadas en los parámetros de fecha.

4. Módulo `EjeSparky.py` (no proporcionado en el código, pero mencionado en el contexto)

Define la clase `EjeSparky`, que parece estar diseñada para interactuar con Spark y manejar datos de manera distribuida.

Propósito: cargar datos desde un módulo Excel a un entorno Spark y realizar operaciones como eliminar una tabla existente antes de cargar datos nuevos.

Conexión con otros módulos:

- Conexión lógica: utiliza Spark para cargar datos desde módulos locales o sistemas de almacenamiento a Spark DataFrames.
- Módulos Excel y Spark: conecta datos almacenados localmente en módulos Excel con un entorno Spark para su procesamiento distribuido.
- Helper (`helper.py`): se utiliza para ejecutar consultas y operaciones en Spark.

Conexiones entre módulos:

- Uso de configuración (`config.json`): todos los módulos acceden a la configuración a través de métodos como `getStepConfig()` para obtener detalles como las tablas de ingestión, módulos SQL, etc.
- Helper (`helper.py`): este módulo contiene métodos auxiliares que son compartidos entre todos los módulos para tareas comunes como ejecutar Módulos SQL, obtener datos de bases de datos, escribir módulos Excel, interactuar con Spark, etc.

- Clases Step: todas las clases (`UltIngestions`, `EjeSQL1`, `EXPOR_BASES`, y `EjeSparky`) heredan de una clase base `Step`, que proporciona métodos y funcionalidades compartidas, como la gestión de logs, acceso a configuraciones y helpers.

Flujo General del Proceso

1. UltIngestions:

- Determina la última ingestión para las tablas especificadas.

- Calcula la fecha de corte actual.

- Prepara los parámetros para el siguiente paso.

2. EjeSQL1:

- Verifica si las tablas necesitan ser actualizadas (basado en la fecha de ingestión).

- Ejecuta Módulos SQL para extraer datos si es necesario.

3. EXPOR_BASES:

Exporta datos de las tablas especificadas a módulos Excel si los datos están disponibles para la fecha actual.

4. EjeSparky (si se incluye):

- Carga datos desde Modulos Excel a un entorno Spark para procesamiento distribuido.

Cada Módulo y clase en el flujo de trabajo ETL tiene un propósito específico y utiliza métodos compartidos para acceder a la configuración, manejar errores, y realizar tareas comunes como la extracción, transformación y carga de datos. La estructura modular y la compartición de métodos a través del helper facilitan el mantenimiento y la escalabilidad del código en un entorno de producción.

Se empezó con la construcción de los querys(consultas) en sql que traerían la información de la base de datos banco.

Script de Ejecución "script_ejecucion"

Es un script de orquestación que coordina diferentes módulos y pasos dentro de un flujo de trabajo ETL (Extract, Transform, Load). Vamos a desglosar cada parte y explicar cómo interactúan entre sí:

"import os

from src.vedp_bases_experiencia_sim.Modulos import EXPOR_BASES,EjeSparky,EjeSQL1,UltIngestions

import argparse

import pkg_resources

from orquestador2.orquestador2 import Orchestrator

if not os.path.exists("logs/"):

print("Directorio de log no creado, Creándolo .....")

os.mkdir("logs/")

###################################

Generación Logs - Compilación

###################################

parser = argparse.ArgumentParser(description='vedp-bases-experiencia_sim')

```
        # Parámetro para determinar tipo de log
        parser.add_argument('-lt',
                        '--log_type',
                        type = str,
                        help = 'Tipo de log: normal, compilación o estabilidad',
                        default = '') #Dejar vacío al finalizar las ejecuciones (las opciones son 'cmp' o
'est' )

        # Parámetro para determinar porcentaje de datos
        parser.add_argument('-pl',
                        '--porcentaje_limit',
                        type = int,
                        help = 'Porcentaje para el LIMIT en los logs de compilación',
                        default = 100) #Dejar vacío al finalizar las ejecuciones
        args = parser.parse_args()
        kw = {k:w for k,w in vars(args).items() if w}
        #steps = [EjeSparky(**kw),UltIngestions(**kw),
EjeSQL1(**kw),EXPOR_BASES(**kw) ]
        steps = [UltIngestions(**kw), EjeSQL1(**kw),EXPOR_BASES(**kw) ]
        orquestador = Orchestrator('vedp-bases-experiencia_sim', steps,**kw)
        orquestador.ejecutar()”
```

- Importaciones

```python
import os
from src.vedp_bases_experiencia_sim.Modulos import EXPOR_BASES, EjeSparky, EjeSQL1, UltIngestions
import argparse
from orquestador2.orquestador2 import Orchestrator
```

- `os`: Librería estándar de Python para operaciones del sistema operativo.
- `EXPOR_BASES, EjeSparky, EjeSQL1, UltIngestions`: módulos importados desde el paquete `src.vedp_bases_experiencia_sim.Modulos`. Estos módulos contienen las clases y funciones que realizan las tareas específicas de extracción, transformación y carga de datos.
- `argparse`: Librería estándar para analizar argumentos de línea de comandos.
- `Orchestrator`: Clase o función importada desde `orquestador2.orquestador2`, encargada de coordinar los pasos definidos en el flujo de trabajo.

- Creación del Directorio de Logs

```python
if not os.path.exists("logs/"):
    print("Directorio de log no creado, Creándolo .....")
    os.mkdir("logs/")
```

- Se verifica si el directorio `logs/` existe. Si no existe, se crea. Esto es útil para almacenar registros (logs) del proceso de ejecución.

- Parámetros de Entrada

```python
parser = argparse.ArgumentParser(description='vedp-bases-experiencia_sim')


parser.add_argument('-lt', '--log_type',
                    type=str,
                    help='Tipo de log: normal, compilación o estabilidad',
                    default='')   Dejar vacío al finalizar las ejecuciones (las opciones son 'cmp' o 'est' )


parser.add_argument('-pl', '--porcentaje_limit',
                    type=int,
                    help='Porcentaje para el LIMIT en los logs de compilación',
                    default=100)   Dejar vacío al finalizar las ejecuciones


args = parser.parse_args()
kw = {k: w for k, w in vars(args).items() if w}
```

- Argumentos de línea de comandos: se configura un parser (`argparse.ArgumentParser`) para aceptar argumentos como `-lt` para el tipo de log y `-pl` para el porcentaje de límite en los logs de compilación.

- `kw`: se crea un diccionario `kw` que contiene todos los argumentos analizados por `argparse`. Este diccionario se pasa como parámetro a las instancias de las clases que se van a crear más adelante.

- Definición de Pasos (Steps)

```python
steps = [UltIngestions(kw), EjeSQL1(kw), EXPOR_BASES(kw)]
```

- `UltIngestions(kw), EjeSQL1(kw), EXPOR_BASES(kw)`: se crean instancias de las clases `UltIngestions`, `EjeSQL1` y `EXPOR_BASES` pasando los argumentos analizados como kwargs (`kw`). Cada una de estas clases hereda de una clase base común y ejecuta diferentes partes del proceso ETL.

- Ejecución del Orquestador

```python
orquestador = Orchestrator('vedp-bases-experiencia_sim', steps, kw)
orquestador.ejecutar()
```

- `Orchestrator`: se instancia la clase `Orchestrator`, es una clase definida en `orquestador2` que se encarga de ejecutar los pasos definidos en `steps`.

- `orquestador.ejecutar()`: se llama al método `ejecutar()` del objeto `orquestador`, que ejecuta el flujo de trabajo definido por los pasos (`steps`).

Este script de orquestación es típico en entornos de ETL donde múltiples pasos deben coordinarse y ejecutarse en un orden específico. Utiliza argumentos de línea de comandos para

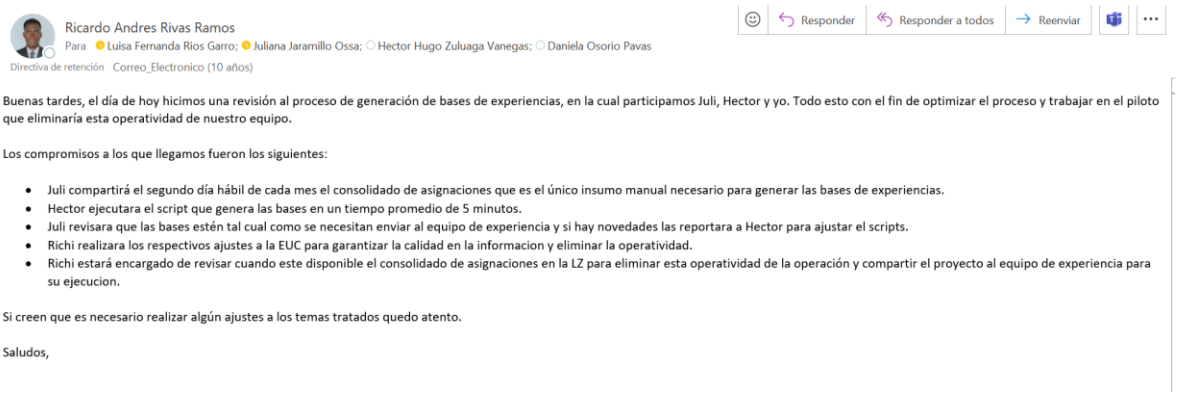
configurar el comportamiento y ejecuta tareas encapsuladas en clases específicas (`UltIngestions`, `EjeSQL1`, `EXPOR_BASES`) que son parte de un sistema modular más grande. La utilización de un orquestador (`Orchestrator`) centraliza la ejecución de estos pasos y facilita la gestión del flujo de trabajo completo desde un único punto.

Validación de la Herramienta de Automatización Desarrollada

Se generó un piloto y realizó las respectivas mejoras al proyecto para poder ajustarlo correctamente, los resultados de las pruebas fueron los siguientes:

Figura 9

Revisión al proceso de generación de bases de experiencias



Nota. Comunicado sobre la revisión del piloto del proyecto (2024)

La ejecución se sigue realizando de forma mensual, y se tiene pronosticado que, finalizando el mes de mayo de 2024 se compartiría al equipo de experiencia de clientes del grupo Bancolombia, las fuentes necesarias para que puedan consultar ellos mismos la información con la periodicidad que definan.

Manual de Usuario

Objetivo: Describir el paso a paso del funcionamiento de la EUC que genera la información para las evaluaciones de las bases de experiencia.

Tabla 1

Control de versión

Versión	Fecha de Modificación DD/MM/AAAA	Acción C: Creación M: Modificación	Nombre del responsable del cambio	Descripción breve del cambio	Aprobado por
1	21/02/2024	C	Ricardo Andrés Rivas Ramos	Creación del manual de funcionamiento de la EUC	Jefe de seccion centro de apoyo inmobiliario

Nota. Paso a paso para el funcionamiento de la EUC. Fuente. Rivas, 2024

Este proyecto posibilita la creación de bases de datos que albergan la información de los clientes elegibles para participar en encuestas de experiencia.

1. Requisitos previos.

Python 3.7 en adelante

Estar en la red Bancolombia ya sea dentro del banco o por VPN.

Licencia de LZ

Controlador de impala

2. Ejecución.

a. Se ingresa a la carpeta de Base de experiencias

Figura 10

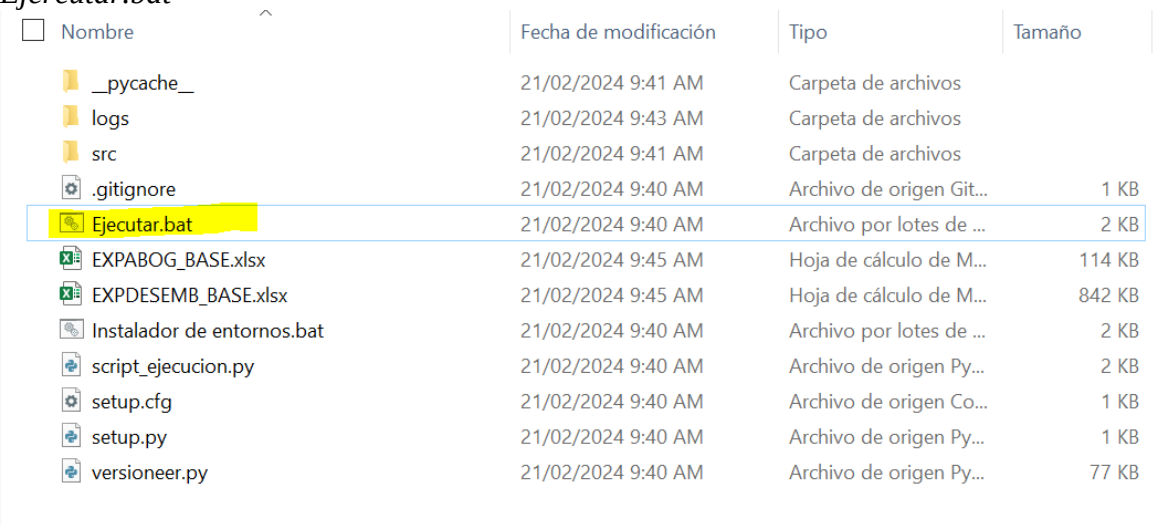
Carpeta base de experiencias

Nota. Pasos para la ejecución del proyecto, Fuente. Rivas (2024)

b. Al ingresar se despliega la siguiente información y le damos doble clic al botón de Ejecutar.bat

Figura 11

Ejercutar.bat

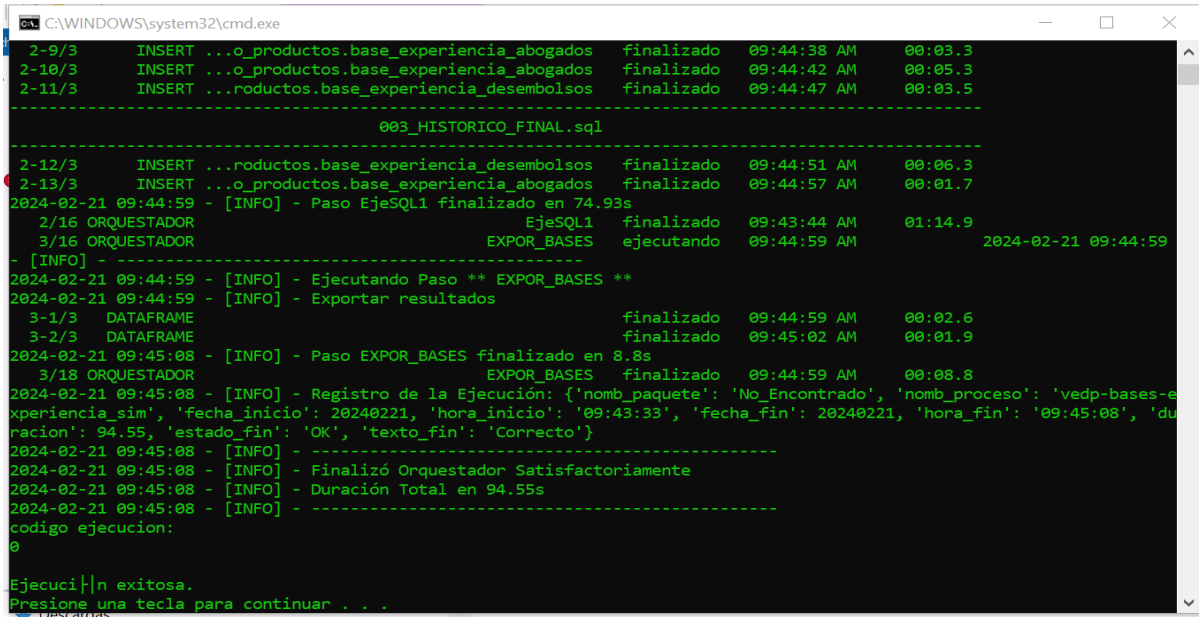


Nota. Pasos para la ejecución del proyecto. Fuente. Rivas, 2024.

c. Cuando el proceso termine se debera ver de la siguiente manera y en color verde, lo cual indicará que se ejecutó de manera correcta y la información llego a su destino.

Figura 12

Correcta ejecución del proceso



Nota. Pasos para la ejecución del proyecto. Fuente. Rivas (2024)

Si la pantalla aparece en color rojo es por que el proceso no se ejecuto de manera correcta, por lo cual debes comunicarte con el analista de datos y auxiliar de automatizaciones para validar el error y gestionan una pronta solucion.Terminado el proceso la informacion quedara consignada en las zonas de la LZ que fueron parametrisadas en la instalacion del proyecto.

Tabla experiencia abogados: base_experiencia_abogados

Tabla experiencia desembolso: base_experiencia_desembolsos

Resultados

A lo largo de este trabajo investigativo, se ha argumentado la importancia y ventajas de implementar un sistema automatizado. Por lo tanto, al hacer una síntesis de todo lo que se ha dicho anteriormente, se puede concluir que, la automatización en la generación de bases de datos permite que las empresas puedan ofrecer una experiencia para el cliente más rápida y eficiente, porque procesa volúmenes de información en el que el error humano es menor; mejora el tipo de atención, puesto que se personaliza la atención a partir de la comprensión y satisfacción de las necesidades del cliente; reduce costos, mejora la eficiencia operativa y todo esto se traduce en una mayor satisfacción del cliente y una ventaja competitiva en el mercado.

Bancolombia ha sido un pionero en la adopción de tecnologías financieras en la región. La implementación de soluciones digitales, como la banca en línea y las aplicaciones móviles, ha mejorado el acceso y la eficiencia de los servicios bancarios, lo que contribuye a la inclusión financiera y a una mejor experiencia para los clientes. Esto, evidencia el compromiso que tiene la empresa en la continua evolución para mejorar los procesos que hoy por hoy, consolidan a Bancolombia como uno de los bancos más importantes de Colombia y Latinoamérica

En lo que respecta a la investigación, generó resultados satisfactorios y una ganancia muy importante para el proceso, particularmente en la reducción de tiempo y eficiencia, ya que paso de tardar 3 horas a 10.08 minutos, liberando el 95.8% de la capacidad dedicada a esta tarea. Además, este mejoramiento fue reconocido en el programa de hitos inmobiliarios del grupo Bancolombia.

Figura 13

Comunicado Hitos Inmobiliarios EVC inmobiliario

Fuente. EVC negocio inmobiliario, 2024

Conclusiones y Recomendaciones

La implementación de un sistema automatizado para la generación de bases de datos ha llevado a una mejora significativa en la eficiencia operativa, ha conducido a una reducción de costos laborales al liberar tiempo valioso de los empleados. Esta liberación de recursos humanos permite su reasignación a tareas más estratégicas y de mayor valor agregado, por lo que a su vez mejora la productividad del equipo.

La eliminación de tareas manuales ha minimizado los errores en la información, garantizando la precisión y confiabilidad de los datos recolectados. Esto permite una toma de decisiones más informada y estratégica, basada en información precisa y actualizada. Así mismo, la automatización ha permitido una recopilación más oportuna de datos, lo que ha facilitado el envío de encuestas de experiencia con mayor frecuencia. Esto ha llevado a una mejora continua en la experiencia del cliente al identificar y abordar sus necesidades de manera más ágil y efectiva, teniendo en cuenta que la experiencia del cliente es actualmente el principal medidor de calidad y os procesos de las empresas deben apuntar a garantizarla.

La mejora en el proceso ha sido reconocida en el programa de hitos inmobiliarios del Grupo Bancolombia, lo que resalta el impacto positivo que ha tenido en la organización y la importancia estratégica de la iniciativa.

En resumen, la implementación de un sistema automatizado para la generación de bases de datos ha generado beneficios significativos en términos de eficiencia operativa, reducción de costos, mejora en la experiencia del cliente y reconocimiento organizacional, permitiendo a la empresa obtener insights más profundos, tomar decisiones más informadas y ofrecer soluciones inmobiliarias de alta calidad que satisfagan las necesidades y expectativas de sus clientes de manera efectiva y eficiente.

Aunque se ha logrado una mejora significativa con la implementación del sistema automatizado, es importante seguir monitoreando el proceso para identificar áreas de mejora adicionales. Realizar evaluaciones periódicas ayudará a garantizar que el sistema siga siendo efectivo y eficiente a medida que cambian las necesidades del negocio.

Se tiene en cuenta la posibilidad de expandir la automatización a otros procesos dentro del Grupo Bancolombia, identificando áreas donde la implementación de herramientas automatizadas podría generar beneficios similares en términos de eficiencia operativa, reducción de costos y mejora de la experiencia del cliente.

Referencias Bibliográficas

Aurora. (2023, julio 18). ¿Qué son las librerías de Python? ID Digital School. Bootcamps. https://lc.cx/N2IV9j

Banco Colpatria. (s.f.) Leasing Habitacional o Crédito Hipotecario - Scotiabank Colpatria. https://lc.cx/PhLpNn

Bancolombia. (2019, enero 2). Modelo Procesador Consumidor Fuentes de Datos Corporativas. Dirección de Capacidades Analíticas y Gobierno de Información DICAGI.

BBVA. (2024, mayo 17). ¿Qué es un Financiamiento? BBVA México. https://lc.cx/dOcJDM

Comisión para el Mercado Financiero. (s.f.) ¿Qué es un Crédito Hipotecario? https://lc.cx/5xIZYc

Connolly, T. M., & Begg, C. E. (2014). Database Systems: A Practical Approach to Design, Implementation, and Management (6th ed.). Pearson Education.

Constitución Política de Colombia. (1991). Constitución Política de Colombia. https://www.constitucioncolombia.com

Coronel, C., & Morris, S. (2016). Database Systems: Design, Implementation, & Management (12th ed.). Cengage Learning.

Del Puerto, N. J. I. (2023). La Gestión en Relación al Cliente (CRM), como Estrategia de Negocio en la Ciudad de Pilar, Paraguay. Ciencia Latina Revista Científica Multidisciplinar. ciencialatina.org.

Díaz de Cerio Escudero, J. L. (2019). La experiencia de cliente en la era digital. El nuevo viaje del cliente.

Dixon, M., Freeman, K., & Toman, N. (2010). Stop trying to delight your customers. Harvard Business Review, 88(7/8), 116-122. https://lc.cx/xs_icO

Elmasri, R., & Navathe, S. B. (2016). Fundamentals of Database Systems (7th ed.). Pearson.

Fernández, O. (2024, febrero 23). Aprende qué es una Pipeline de Datos. Aprender BIG DATA. https://lc.cx/ut0Fh0

Fornell, C., Johnson, M. D., Anderson, E. W., Cha, J., & Bryant, B. E. (1996). The American Customer Satisfaction Index: Nature, Purpose, and Findings. Journal of Marketing, 60(4), 7-18. https://lc.cx/jyx-kb

Frederick F. Reichheld. (2007). La pregunta decisiva. Harvard Business School Press. Grupo Planeta (GBS). ISBN 8423424618.

Frederick F. Reichheld. (2021). Winning on Purpose. Harvard Business Review Press.

Función Pública. (2021). Ley 2079 de 2021. https://lc.cx/y6lp4n

Gartner (2018). The Customer Experience in 2020 September 10, 2015.

González Duque, R. (2015). Python para todos. Otros.

Grupo Bancolombia. (2024). Informe de Gestión 2024. https://lc.cx/dhCHTV

Inmon, W. H. (2002). Building the Data Warehouse (3rd ed.). Wiley.

Kimball, R., Reeves, L., Thornthwaite, W., & Mundy, J. (2008). The Data Warehouse Lifecycle Toolkit (2nd ed.). Wiley.

Kaplan, R. S., & Norton, D. P. (1996). The Balanced Scorecard: Translating Strategy into Action. Harvard Business School Press. https://lc.cx/5AEoAb

Marqués, Andrés. (2011). Bases de datos. Universitat Jaume I.

Martin, J. (2024, mayo 4). ¿Cómo están compuestos los hogares colombianos actualmente? Esto dice el Dane. Portafolio. https://lc.cx/Rd66QT

Marzal, A., y Gracia, I. (2009). Introducción a la programación con Python. Universitat Jaume I.

Norman, D. A. (2013). The design of everyday things. Basic Books.

openpyxl - A Python library to read/write Excel 2010 xlsx/xlsm files - (2024, mayo 29). openpyxl 3.1.3 documentation. https://lc.cx/K-wOZZ

Ortiz, V. & Pardo, H. F. (2021). Importancia y ventajas de los KPI (Key Performance Indicators) en los proyectos: enfoque de procesos en el sector petrolero. https://lc.cx/8wZxV_

Pandas. (2024). Python Data Analysis Library. https://pandas.pydata.org/

PYTZ. (2024, febrero 2). PyPI. https://pypi.org/project/pytz/

Python Software Foundation. (2024). Datetime-Basic date and time types. Python Documentation. https://lc.cx/lb-5FT

Python Software Foundation. (2024). Entornos virtuales y paquetes. Python Documentation. https://lc.cx/44_54y

Ramakrishnan, R., & Gehrke, J. (2003). Database Management Systems (3rd ed.). McGraw-Hill.

Schwarz Díaz, M. (2018). Reflexiones sobre la medición de la Experiencia del Cliente. Universidad de Lima, Facultad de Ciencias Empresariales y Económicas.

ToGrow. (s.f.) ¿Qué son los Motores de Bases de Datos? https://lc.cx/kkaZBq

Unir, V. (2023, septiembre 27). Framework: qué es, para qué sirve y algunos ejemplos. UNIR FP. https://lc.cx/NK-gAz