

Extracción resiliente de documentos en aplicaciones SaaS heredadas mediante RPA paralela: estudio técnico de un proyecto Taleo

Joaquín Díaz, Sebastián Beltrán y William Betancourt
Oracle Cloud Infrastructure y automatización de extracción

Resumen—Este artículo documenta la evolución técnica de un proyecto de recuperación documental en Oracle Taleo, una aplicación SaaS en proceso de retiro. Los enfoques iniciales—exportación por Taleo Connect Client (TCC) y extracción relacional para consulta mediante BI—produjeron datos estructurados, pero no lograron el resultado operativo requerido: recuperar representaciones documentales consultables y asociadas a sus entidades de negocio. A partir del análisis del front end se diseñó una automatización en Python que reprodujo de manera controlada la navegación, la generación de documentos y la descarga de resultados. La solución se ejecutó de forma paralela sobre máquinas virtuales de Oracle Cloud Infrastructure (OCI), con dimensionamiento iterativo de memoria y CPU. Se describe el modelo de capacidad, la observación de que la memoria fue el recurso limitante, el escalamiento hasta 68 GB de RAM y 8 OCPU, las limitaciones de latencia de la aplicación fuente y un esquema de validación basado en inventario y conciliación. El trabajo muestra que, en sistemas SaaS heredados, la automatización de la interfaz puede ser un mecanismo de recuperación técnicamente defendible cuando los canales estructurados no preservan la representación documental necesaria.

Palabras clave—automatización robótica de procesos, RPA, Python, Oracle Cloud Infrastructure, Taleo, extracción documental, capacidad.

I. INTRODUCCIÓN

La retirada de una aplicación SaaS plantea un problema que no se resuelve únicamente con copiar tablas. El valor operativo puede residir en documentos, imágenes, vistas renderizadas y relaciones que el esquema relacional no expresa de forma directamente consumible. Este estudio presenta, sin identificar a la organización usuaria, la trayectoria técnica de una recuperación de información en Taleo. El propósito no es afirmar completitud absoluta, sino describir un método reproducible, sus límites y la evidencia con la que se controló su ejecución.

La contribución principal fue cambiar la unidad de extracción: de registros aislados a representaciones documentales generadas por la propia interfaz. Ese cambio se derivó del análisis del front end y no de una suposición de que la capa de datos fuera incorrecta. El resultado fue una automatización Python paralelizada, con trazabilidad de cada ítem y capacidad ajustada empíricamente.

II. CONTEXTO Y PLANTEAMIENTO DEL PROBLEMA

El ofrecimiento estándar de terminación contempla, entre otras modalidades, un archivo de base de datos y extractos estructurados [1]. Dichos mecanismos son adecuados para conservar datos relacionales, pero su consumo exige montaje, interpretación del modelo y trabajo adicional de integración. En el caso estudiado coexistían personalizaciones y referencias circulares; por ello, la consulta de entidades relacionadas no se reducía a un conjunto estable de uniones.

El criterio de éxito fue operativo: obtener artefactos que conservaran la representación visible y permitieran localizarla por identificadores de negocio. Se incluyeron cuatro familias: requisiciones, candidatos, postulaciones y correspondencia. Esta definición evita equiparar “dato exportado” con “documento disponible”, dos propiedades que deben medirse por separado.

La iniciativa no fue lineal. Comenzó con una solicitud de extracción en noviembre de 2022 y una entrega estructurada complementaria en 2023. En 2025 se activó una nueva fase: se estudió el diagrama entidad–relación, se construyeron consultas de exploración y se evaluó TCC. El diagnóstico acumulado no descartó la información estructurada; identificó precisamente qué parte de la experiencia histórica no podía reconstruirse de forma consistente solo con ella. La fase de automatización sobre la interfaz surgió de esa evidencia.

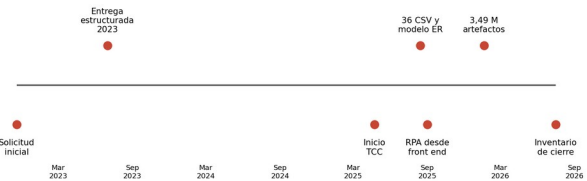


Fig. 1. Cronología técnica del proyecto. Los hitos muestran cambios de método y de evidencia disponible; no representan una serie de rendimiento.

III. MÉTODOS PREVIOS Y SUS LIMITACIONES

A. Extracción por TCC

La vía TCC generó 36 archivos CSV con 33 787 733 registros y más de 1 GB de datos estructurados. Por tanto, no fue una ausencia de datos ni una falla de transferencia. Sin embargo, no fue exitosa para el objetivo de recuperación documental: los CSV no producían por sí mismos las vistas, los adjuntos ni las asociaciones de negocio necesarias para una consulta operativa. La reconstrucción requería combinar entidades con referencias circulares y campos personalizados, lo que incrementó la ambigüedad de las uniones y el costo de validación.

B. Extracción relacional y consulta BI

También se construyó una capa de consulta sobre la extracción relacional, con consultas y visualizaciones. El enfoque permitió inspeccionar subconjuntos de datos, pero no fue exitoso como mecanismo final de recuperación de documentos. La causa no fue la herramienta de BI: era una desalineación entre la semántica de las tablas y el resultado que debía preservarse. Un modelo útil para análisis no garantiza que reproduzca la experiencia documental de una transacción histórica.

C. Lección de diseño

Ambos intentos aportaron conocimiento: el modelo físico, los identificadores y las familias de objetos. Ese conocimiento permitió formular la hipótesis posterior: si la interfaz puede construir una representación que satisface la necesidad documental, entonces la interfaz es un punto de extracción válido, sujeto a controles de rendimiento, trazabilidad y cumplimiento aplicables.

IV. DISEÑO DE LA AUTOMATIZACIÓN

La solución nació al analizar el front end de Taleo: se identificaron rutas de navegación, parámetros de búsqueda, eventos que generan la representación y la descarga asociada. La idea inicial recordaba herramientas históricas de grabación de acciones de ratón, que capturan una secuencia y la repiten como macro. Ese antecedente fue útil como intuición, pero insuficiente como implementación: una macro basada solo en coordenadas es frágil ante latencia, cambios de foco y variaciones de contenido.

Python permitió sustituir la repetición ciega por un flujo con estado. Cada bot recibió una partición de identificadores, abrió una sesión controlada, ejecutó la búsqueda, esperó condiciones observables, solicitó la generación del artefacto, descargó el resultado y registró su desenlace. La automatización no dependió exclusivamente del movimiento del ratón; combinó acciones de navegador con comprobaciones de disponibilidad, reintentos acotados, tiempos de espera y registros de error.

El análisis del front end fue una actividad de ingeniería inversa funcional, no de extracción de código. Se observaron los elementos de navegación expuestos a un usuario autorizado, los identificadores requeridos por cada búsqueda y el punto de transición entre la visualización y la descarga. Esta distinción importa: el bot replicó un flujo permitido de uso de la aplicación y registró cada resultado, en lugar de intentar inferir documentos a partir de estructuras internas incompletas.

Fig. 1. Flujo lógico de cada bot: partición de identificadores → navegación y búsqueda → generación en front end → descarga → inventario y reconciliación.

A. Control de estado y tolerancia a fallas

La unidad mínima de trabajo fue un identificador de negocio y un tipo documental. Cada intento generó un estado terminal: descargado, no localizado, fallido por tiempo de espera o pendiente de reproceso. El registro de estados evitó que un reinicio se interpretara como una ejecución nueva y permitió asignar solo pendientes a los reintentos. Esta propiedad es esencial para evitar duplicados y para distinguir un problema de origen de una falla transitoria del bot.

B. Organización de salida

Los resultados se almacenaron con una estructura que preserva la familia documental y el identificador, lo que permite conciliación por carpeta y subcarpeta. La nomenclatura funciona como índice operativo: favorece la búsqueda, sostiene el conteo automatizado y hace auditable la relación entre una solicitud y su resultado.

V. ARQUITECTURA DE EJECUCIÓN EN OCI

La ejecución paralela se realizó en máquinas virtuales OCI. La arquitectura separó el plano de control—listas de trabajo, parámetros y registros—del plano de ejecución, compuesto por múltiples bots independientes. Cada instancia consumía sesiones de navegador, procesos de automatización, búferes de descarga y espacio temporal. La capacidad no se fijó por intuición; se ajustó mediante observación durante campañas controladas.

La elección de una configuración flexible fue coherente con la naturaleza del problema. OCI define una forma de cómputo como la plantilla que asigna OCPU, memoria y otros recursos; sus formas flexibles permiten seleccionar OCPU y memoria al crear o redimensionar una VM [5]. En consecuencia, el diseño no trató RAM y CPU como un único “tamaño de servidor”: cada recurso se ajustó en función del comportamiento de las sesiones de navegador y de la aplicación origen.

El dimensionamiento comenzó con mediciones de consumo por bot y observación de CPU, memoria, estabilidad de sesión y tasa de errores. Para una instancia con M GB de RAM y consumo estable m por bot, el límite de memoria se estimó como $N_{\text{mem}} = \text{floor}((M -$

R)/m), donde R es la reserva para sistema operativo, navegador base, telemetría y picos. El límite de CPU se estimó en paralelo como $N_{\text{cpu}} = \text{floor}(C/u)$, donde C representa la capacidad disponible y u el consumo medio por bot. El número aplicable de bots fue $\min(N_{\text{mem}}, N_{\text{cpu}}, N_{\text{fuente}})$, siendo N_{fuente} el límite impuesto por la aplicación origen.

Configuración y criterio de capacidad

Memoria — Hasta 68 GB RAM. Recurso más valioso; determinó la cantidad segura de sesiones concurrentes

Cómputo — Hasta 8 OCPU. Suficiente para coordinar procesos, pero no eliminó la espera de la aplicación origen

Concurrencia — Flota de bots particionada. Se aumentó gradualmente según telemetría y tasa de error

Almacenamiento — Salida e inventarios persistentes. Necesario para descargas, reintentos y conciliación

La observación decisiva fue que la RAM, no la CPU, restringía el escalamiento antes de alcanzar 8 OCPU. Las sesiones de navegador y sus procesos auxiliares mantuvieron memoria residente mientras esperaban respuestas remotas. Incrementar CPU sin ampliar memoria no elevaba de forma proporcional el número estable de bots. Esta relación debe verificarse por carga de trabajo; no constituye una regla universal para toda automatización web.

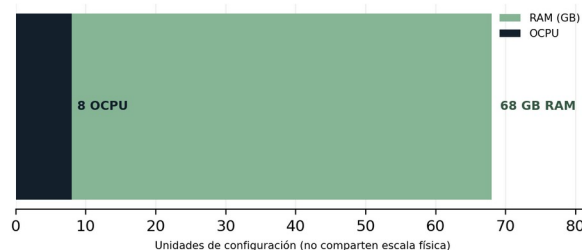


Fig. 2. Configuración máxima observada durante la campaña. La figura comunica dos recursos diferentes; no compara sus unidades ni representa utilización porcentual.

VI. LENTITUD DEL ORIGEN Y ESTRATEGIA DE PARALELISMO

Taleo presentó latencia variable en navegación, renderizado y generación de resultados. En términos de teoría de colas, el proceso fue predominantemente I/O-bound: durante una fracción relevante del ciclo, el bot esperaba al servicio remoto y no consumía CPU de manera continua. Por ello, un grado de concurrencia moderado podía aumentar el rendimiento agregado, siempre que no superara la memoria disponible ni indujera errores o degradación en el origen.

La política de escalamiento fue conservadora: (1) ejecutar una muestra; (2) medir tiempo por ítem, memoria por bot y tasa de fallas; (3) incrementar el número de bots; (4) detener el aumento si la tasa de errores o el tiempo mediano se deterioraban; y (5) reprocesar únicamente estados recuperables. No se trató de “hacer clic más rápido”; los tiempos de espera fueron parte del protocolo porque la sincronización con el origen condicionaba la validez de cada descarga.

La tasa de producción puede expresarse como $T = N/(t_{\text{nav}} + t_{\text{gen}} + t_{\text{down}} + t_{\text{wait}})$, para N bots estables y tiempos medios de navegación, generación, descarga y espera. Esta expresión hace visible que, cuando t_{wait} domina, añadir CPU tiene bajo retorno marginal; la ganancia proviene de concurrencia controlada y de memoria suficiente para sostener las sesiones.

VII. RESULTADOS Y VALIDACIÓN

Una campaña documentada reportó 3 487 563 artefactos recuperados desde el front end, incluyendo PDF, capturas y un archivo de roles. El inventario de cierre posterior registró 9 436 415 archivos, 3 284 894 directorios y subdirectorios recorridos, y 930,87 GB analizados. Estos valores describen etapas y unidades de medida distintas; no deben sumarse ni usarse como una única métrica de cobertura.

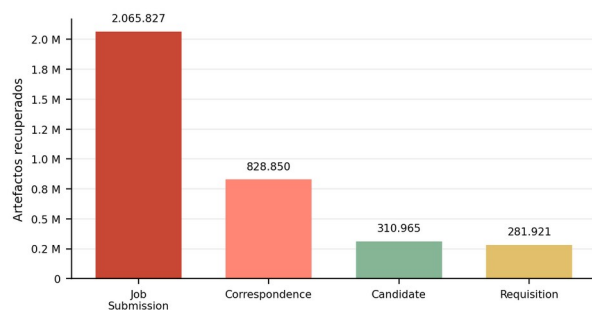


Fig. 3. Composición de la campaña automatizada documentada (3 487 563 artefactos). Las categorías suman el total de esa campaña específica.

Métricas de ejecución e inventario

Campaña automatizada documentada — 3 487 563 artefactos. PDF, capturas y archivo de roles recuperados desde el front end

Inventario final — 9 436 415 archivos. Conteo físico de archivos en la estructura entregada

Árbol recorrido — 3 284 894 directorios. Evidencia de recorrido y generación de inventario

Volumen analizado — 930,87 GB. Magnitud física del conjunto inventariado

Errores de inventario — 0. Sin errores de procesamiento o conectividad registrados durante el inventario

La validación combinó cuatro controles: conteo automatizado por carpeta y subcarpeta; conciliación de cantidades entre origen de trabajo y salida; revisión de estructura y nomenclatura; y muestras funcionales de apertura y contenido. Ninguno de ellos, por separado, demuestra integridad semántica absoluta. Juntos establecen evidencia verificable de qué se procesó, qué se transfirió y qué excepciones permanecieron.

VIII. DISCUSIÓN

El caso contradice una premisa frecuente: que la extracción de base de datos es siempre la forma más completa de preservar una aplicación. Para información relacional, esa vía puede ser superior. Para evidencia documental vinculada a una interfaz histórica, una representación generada por el front end puede ser más útil y más verificable. La elección correcta depende de la propiedad que se quiere conservar: estructura de datos, vista documental, relaciones de negocio o capacidad de consulta.

La automatización en Python fue mejor que una macro grabada porque incorporó idempotencia práctica, control de estados, telemetría y partición de trabajo. Sin embargo, conserva riesgos: dependencia de la interfaz, cambios visuales, límites de sesión, velocidad variable y necesidad de proteger los datos personales procesados. Su aplicación debe contar con autorización, controles de acceso, retención definida y revisión de los términos aplicables.

La experiencia aporta una recomendación metodológica para futuros retiros de SaaS: definir desde el comienzo tres objetivos de preservación independientes—datos estructurados, evidencia documental y capacidad de consulta—y elegir para cada uno un mecanismo y una validación. La exportación de datos no debe prometer implícitamente la preservación del documento renderizado; de manera equivalente, un PDF no sustituye una base relacional si se requieren análisis o relaciones transaccionales. El diseño híbrido reduce ese falso dilema.

IX. CONCLUSIONES

El proyecto mostró que un enfoque de recuperación documental basado en front end puede complementar los extractos estructurados cuando estos no alcanzan el resultado operativo requerido. TCC y la capa BI fueron fuentes importantes de conocimiento y datos, pero no resolvieron la necesidad de reconstruir documentos consultables. El análisis de la interfaz permitió diseñar una alternativa más alineada con esa necesidad.

El rendimiento estuvo condicionado principalmente por la latencia del origen y la memoria residente de las sesiones; en el punto de configuración más exigente, 68 GB de RAM fueron más determinantes que 8 OCPU para sostener bots adicionales. El escalamiento gradual, el inventario y la conciliación fueron tan importantes como el código. La lección general es que la automatización de UI no debe evaluarse por similitud con una macro, sino por su capacidad de controlar estado, evidenciar resultados y operar dentro de límites de infraestructura y del sistema fuente.

REFERENCIAS

- [1] Oracle, “Oracle Taleo Integration Termination Extract Offering,” versión 1.3, abril de 2014.
- [2] Documentación técnica interna del proyecto, “Distribución de entregables y cronología de extracción,” 2023–2026.
- [3] Documentación técnica interna del proyecto, “Actualización de extracción TCC, propuesta PDF y requisitos no funcionales,” 2025–2026.
- [4] Documentación interna de control, “Inventario de archivos y evidencias de validación,” corte de cierre del proyecto, 2026.
- [5] Oracle Cloud Infrastructure, “Compute Shapes,” documentación en línea, consultada en septiembre de 2026. [En línea]. Disponible: <https://docs.oracle.com/en-us/iaas/Content/Compute/References/computeshapes.htm>
- [6] Oracle, “Exporting E Offer Letter PDFs using Taleo Connect,” Oracle Taleo Enterprise Documentation, 2025. [En línea]. Disponible: <https://docs.oracle.com/en/cloud/saas/taleo-enterprise/25b/ottcg/c-exportingofferletterwithtcc.html>