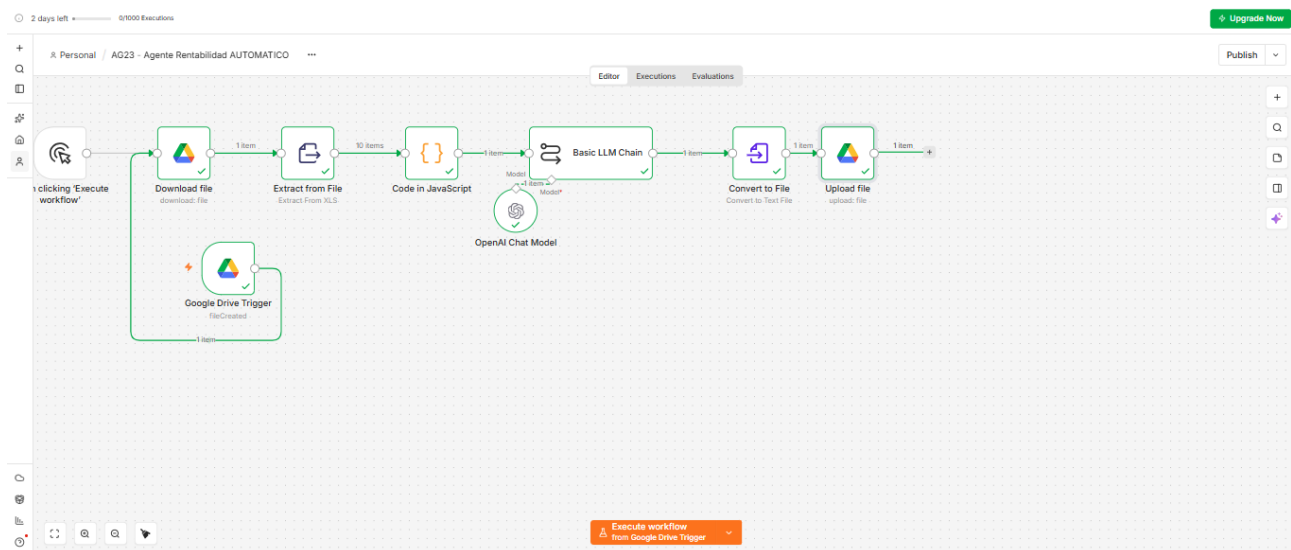


AGENTES DE IA CON n8n + JAVASCRIPT

AUTOMATIZACIÓN · ANÁLISIS DE DATOS · REPORTING

Caso práctico de rentabilidad vehicular
con Excel, Google Drive y OpenAI

AG23 | SOLUCIÓN DE ANÁLISIS DE RENTABILIDAD VEHICULAR



ALFONSO GALEA | EDICIÓN EN ESPAÑOL | OCTUBRE 2026

Crear · Ejecutar · Validar · Mantener

Contenido

Introducción.....	3
Objetivos de aprendizaje	3
Metodología C.E.V.M.	3
Taller . Agente de rentabilidad vehicular	4
1. Problema y producto esperado.....	4
2. Datos del ejercicio	4
3. Arquitectura implementada	4
4. C — Crear: preparar fuentes y configuraciones.....	5
5. E — Ejecutar: del disparador manual al automático	5
6. V — Validar: calidad y resultados	7
7. Comprobación de ejecución automática	8
8. M — Mantener: controles para la versión 2.0	8
Producto obtenido y conclusiones.....	9
Lista final de evidencias.....	9
Anexo práctico A. Reproducción del agente	9
A.1 Preparar los datos y las carpetas	9
A.2 Configuración de n8n.....	9
A.3 Código de referencia JavaScript.....	10
A.4 Instrucciones del modelo IA	10
A.5 Validación antes de publicar	10
Anexo práctico B. Portafolio profesional de habilidades	11
Próximas guías de la colección	11
Propósito de la mejora	12
Qué quedó demostrado	12
Arquitectura del experimento	12
Cómo reproducir el reporte generado por OpenAI.....	13
Criterios de validación y resultado	13
Evidencias del producto obtenido.....	14
Evidencia PDF · Página 1 de 3.....	14
Evidencia PDF · Página 2 de 3.....	15
Evidencia PDF · Página 3 de 3.....	16
Lecciones técnicas para el portafolio.....	17
Mantenimiento y límites de la versión 2.0	17
Producto final de la Guía	18

Introducción

Las empresas que consolidan ingresos y egresos en hojas de cálculo necesitan información confiable y oportuna para controlar sus operaciones. Esta solución integra Excel, Google Drive, n8n, JavaScript y OpenAI para recibir movimientos, verificar su calidad, calcular indicadores por vehículo y preparar informes claros para responsables administrativos y de operaciones. El documento explica la arquitectura, la implementación, los controles y los resultados obtenidos en un caso de prueba.

La automatización no convierte cualquier flujo en un agente autónomo que razona y decide libremente. El proyecto AG23 corresponde a un flujo de análisis asistido por IA: JavaScript realiza los cálculos definidos y el modelo de OpenAI interpreta los resultados. Esa separación facilita el control de calidad.

Necesidad empresarial y alcance de la solución

Cuando los registros se procesan manualmente, es posible duplicar transacciones, omitir gastos o entregar informes con demora. AG23 propone un proceso repetible que detecta nuevos archivos en una carpeta de entrada, identifica incidencias, concilia totales y presenta indicadores útiles para supervisar la rentabilidad de una flota. El caso se prueba con datos ficticios y no representa una implantación contratada ni un entorno productivo.

La solución puede adaptarse, tras revisar la estructura y seguridad de cada fuente, a flotas, alquiler de equipos, ventas o unidades operativas. La separación entre cálculos verificables en JavaScript e interpretación de resultados con OpenAI permite mantener trazabilidad y controles antes de comunicar cifras a la dirección. Los beneficios de ahorro de tiempo o reducción de errores deberán medirse en cada implementación empresarial.

Objetivos operativos de la solución

- Diseñar el flujo de un sistema de análisis asistido por IA.
- Conectar archivos Excel de Google Drive con n8n.
- Procesar, validar e interpretar información financiera.
- Ejecutar un flujo de forma manual y después automáticamente.
- Definir comprobaciones, alertas y mantenimiento para asegurar resultados confiables.

Metodología C.E.V.M.

Fase	Propósito	Producto verificable
C • Crear	Objetivo, fuentes, reglas, herramientas e instrucciones	Flujo configurado
E • Ejecutar	Probar nodos, activar eventos y ejecutar el flujo	Historial de ejecución
V • Validar	Comprobar duplicados, cálculos y resultados	Tabla reconciliada e informe
M • Mantener	Supervisar errores, formatos, versiones y costos	Plan de control y mejoras

Regla transversal: un resultado no se declara validado únicamente por el estado “Succeeded”; se verifica también su contenido.

Caso práctico · Agente de rentabilidad vehicular

1. Problema y producto esperado

Una empresa de alquiler de vehículos recibe registros de ingresos y egresos en Excel. Repetir manualmente la limpieza, agregación y preparación de un reporte supone tiempo y riesgo de errores. La solución es un flujo automático que recibe un archivo, calcula indicadores y guarda un informe.

Producto obtenido: un flujo de rentabilidad que primero entregó un TXT automático; posteriormente se desarrollaron dos alternativas HTML (plantilla JavaScript y diseño generado por OpenAI) y se obtuvo un PDF mediante la impresión del HTML en Chrome. La versión experimental no automatizó el PDF.

2. Datos del ejercicio

Archivo: Taller_1_Agente_IA_Datos.xlsx. Contiene diez filas de movimientos ficticios de tres vehículos. Una transacción (TX008) se duplicó deliberadamente para comprobar su detección.

Campo	Uso
ID / Fecha	Identificación del movimiento
Vehículo / Canal	Dimensión de análisis
Tipo / Categoría	Clasificación de ingreso o egreso
Monto USD	Importe numérico a agregar

3. Arquitectura implementada

Google Drive Trigger → Download file → Extract from File → Code in JavaScript → Basic LLM Chain + OpenAI Chat Model → Convert to File → Upload file.

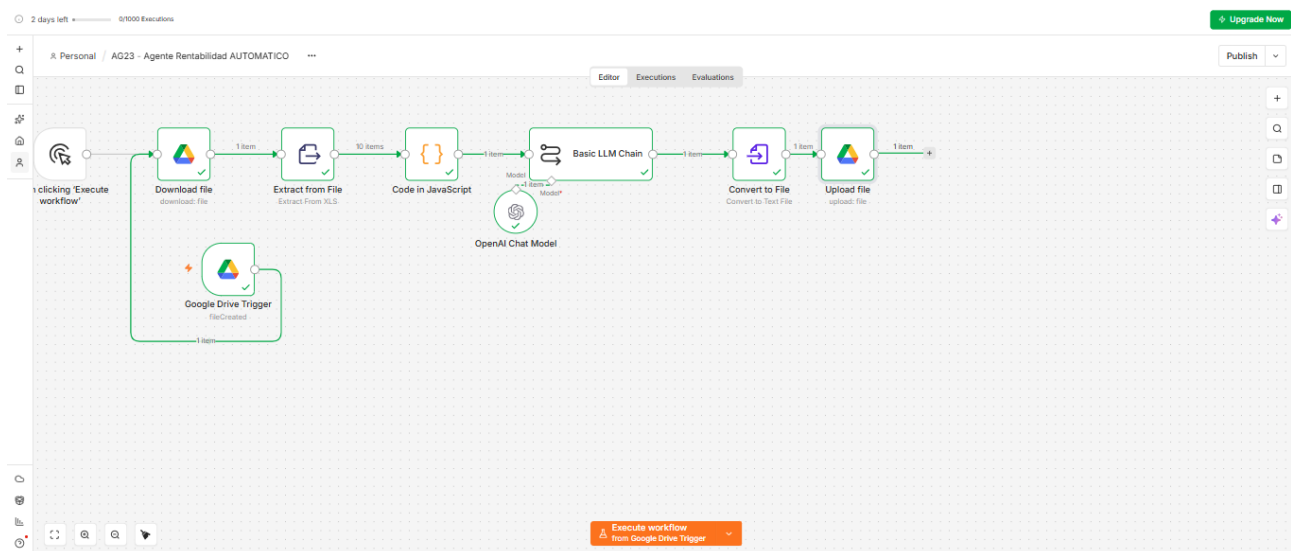


Figura 1. Arquitectura real del flujo AG23 en n8n.

4. C — Crear: preparar fuentes y configuraciones

En Google Drive se creó una carpeta de proyecto con dos subcarpetas: ENTRADA_EXCEL, destinada a los archivos entrantes, e INFORMES_GENERADOS, destinada a los informes producidos.

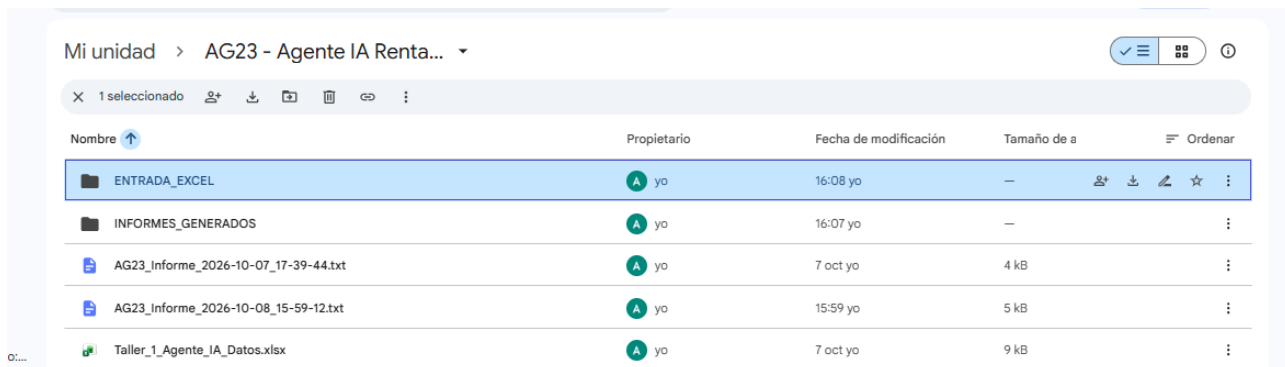


Figura 2. Organización original de las carpetas del proyecto en Google Drive.

Los nodos se configuraron para descargar el archivo, extraer XLSX, analizar con JavaScript y OpenAI y subir un TXT. El campo “File” del nodo Download file se cambió a “By ID”, usando la expresión `{{ $json.id }}` para descargar el archivo recibido desde el disparador.

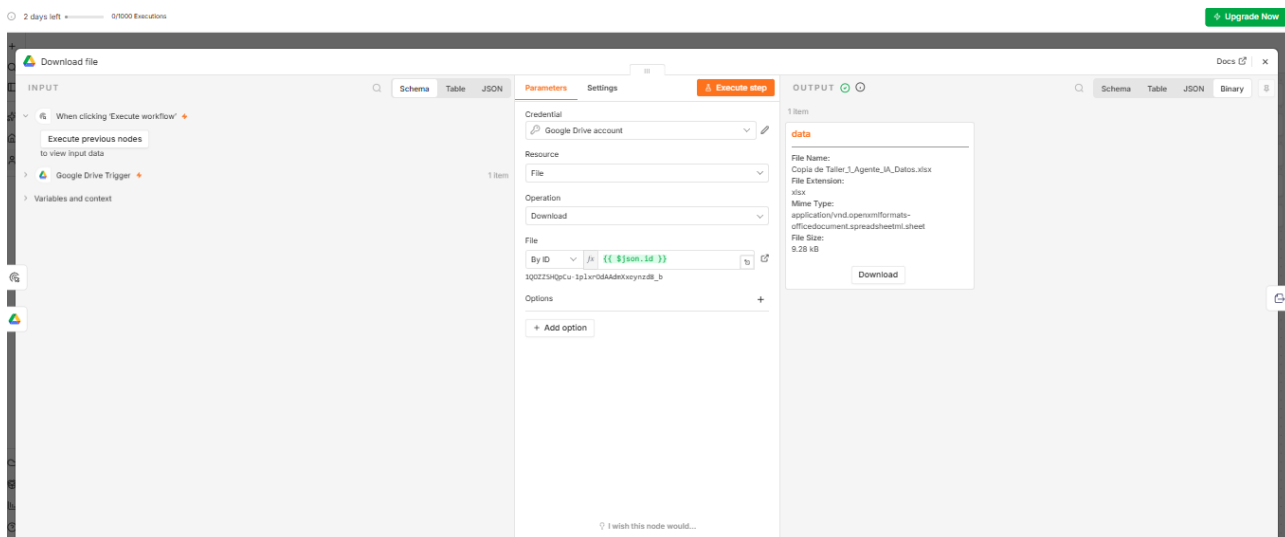


Figura 3. Descarga dinámica del Excel recibido mediante el identificador del evento.

5. E — Ejecutar: del disparador manual al automático

Primero se comprobó el funcionamiento del flujo mediante ejecución manual. Después se agregó Google Drive Trigger, configurado para vigilar ENTRADA_EXCEL y detectar el evento File Created. Durante el taller se estableció una frecuencia de revisión de un minuto, apropiada para la prueba, no necesariamente para producción.

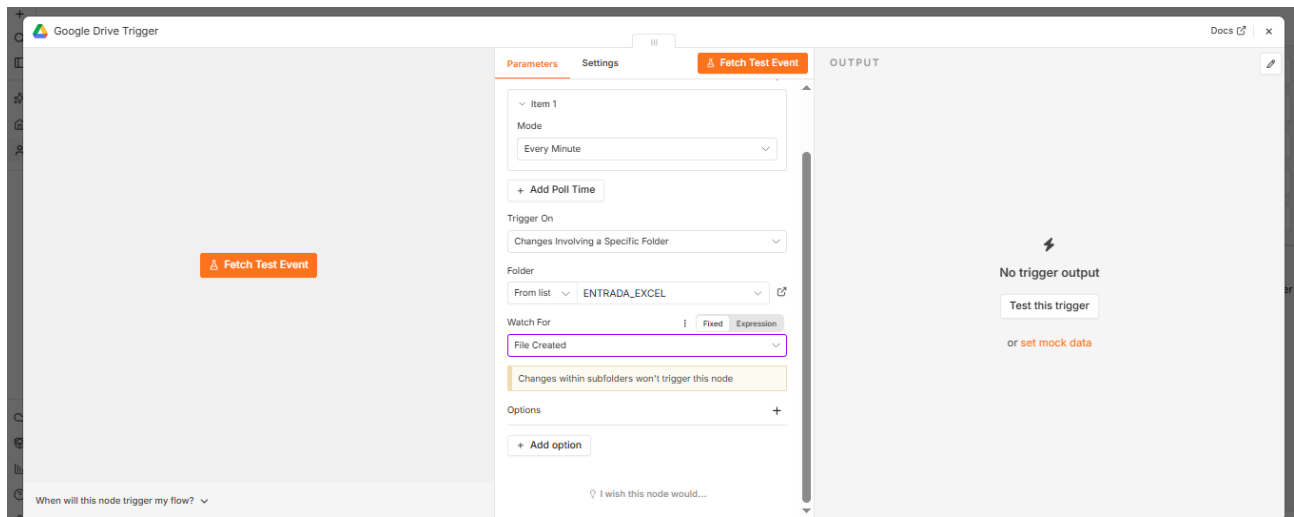


Figura 4. Configuración del disparador: carpeta de entrada y evento File Created.

El nodo Upload file quedó dirigido a INFORMES_GENERADOS, con nombre de informe que incorpora fecha y hora, evitando confundir entradas y salidas.

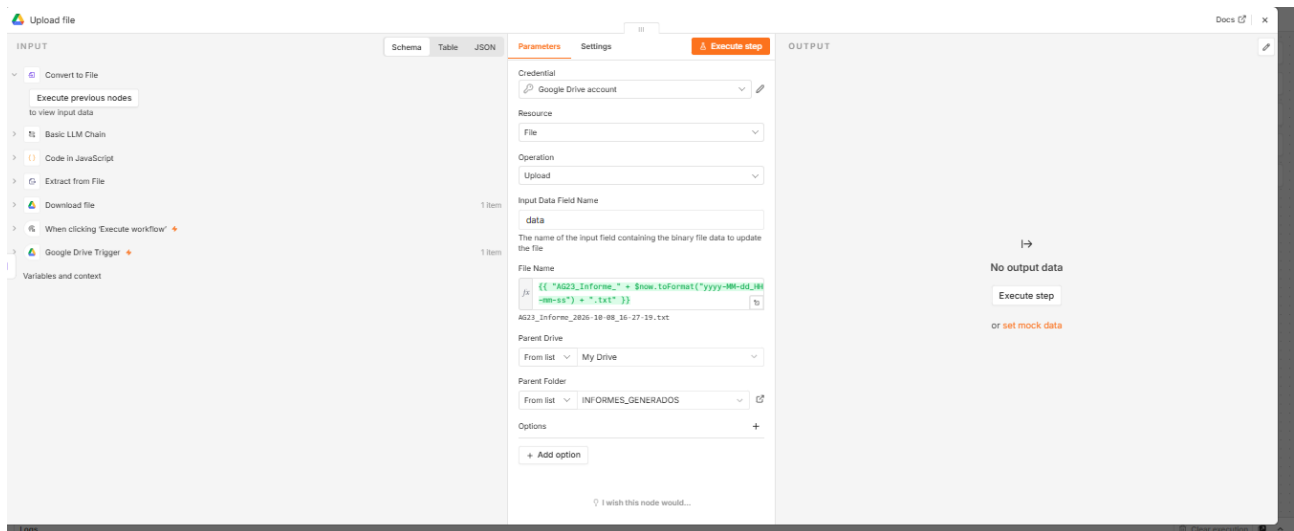


Figura 5. Configuración del destino de los informes generados.

La versión AG23 - Agente Rentabilidad v1.0 se publicó. n8n confirmó que revisaría Google Drive periódicamente; luego se subió PRUEBA_AUTOMATICA_AG23.xlsx sin pulsar Execute workflow.

Figura 7. Fragmento del informe real de análisis y de la tabla de indicadores.

7. Comprobación de ejecución automática

La pestaña Executions registró la ejecución automática del 8 de octubre de 2026 a las 16:39:10, con estado Succeeded y una duración de 39.578 segundos. El historial mostró los nodos del procesamiento en verde, incluida la carga final del informe.

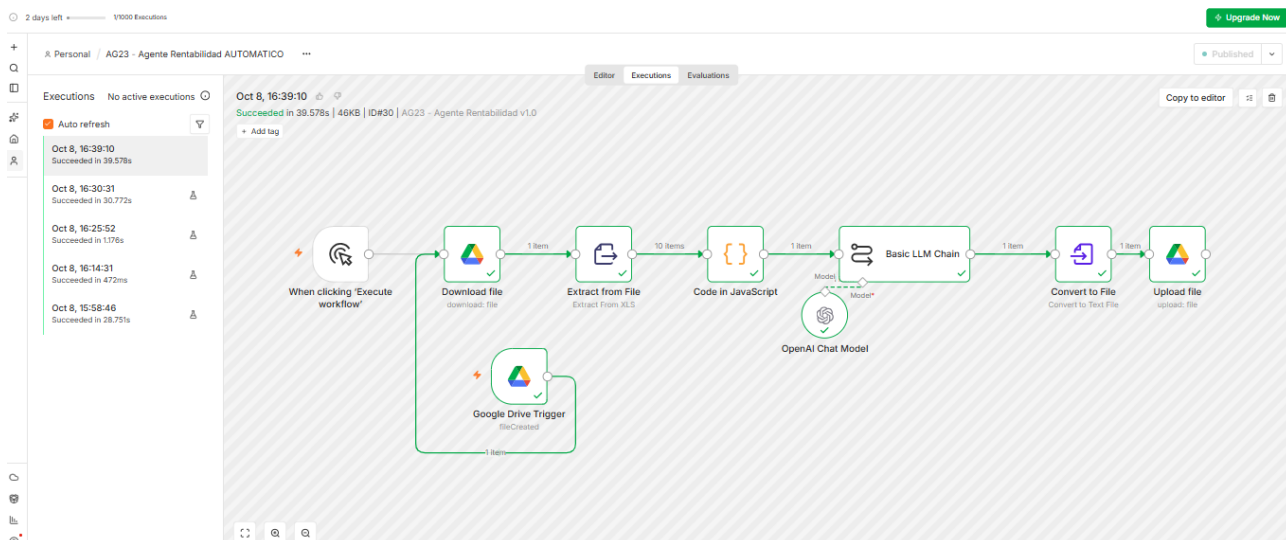


Figura 8. Evidencia principal: ejecución automática exitosa iniciada por Google Drive Trigger.

En una prueba previa también se confirmó visualmente la creación de un TXT en la carpeta INFORMES_GENERADOS. Para la ejecución de las 16:39, la captura del historial confirma el éxito del nodo Upload file; la apertura y revisión del TXT específico de esa ejecución queda como verificación adicional recomendada.

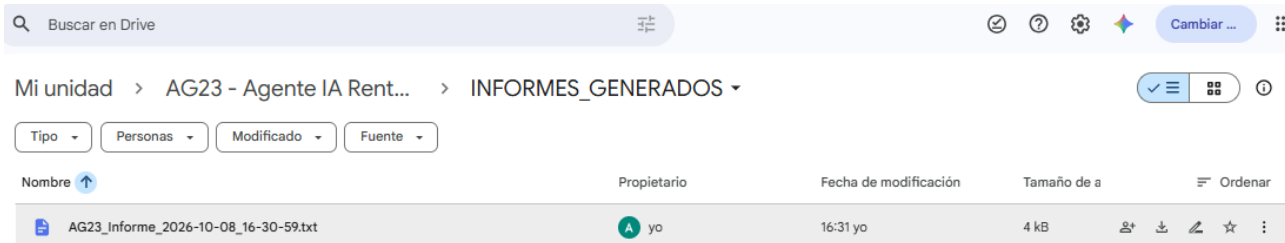


Figura 9. Archivo de informe visible en INFORMES_GENERADOS durante las pruebas.

8. M — Mantener: controles para la versión 2.0

- Verificar tipo MIME y extensión XLSX antes del procesamiento.
- Validar columnas obligatorias y rechazar archivos incompletos.
- Evitar volver a analizar el mismo identificador de archivo.
- Registrar errores por nodo y generar notificaciones.
- Registrar totales de control, archivo origen y fecha de ejecución.
- Revisar la frecuencia de sondeo, consumo de OpenAI y límites de n8n.
- Guardar versiones del flujo y repetir pruebas luego de cada cambio.

Estos controles se proponen como mejoras; no deben confundirse con funcionalidades ya verificadas de la primera versión.

Producto obtenido y conclusiones

Se construyó y publicó un flujo de análisis de rentabilidad vehicular que detecta archivos Excel nuevos en Google Drive, ejecuta cálculos mediante JavaScript, genera un informe interpretado por OpenAI y lo guarda en Drive. La ejecución automática registrada confirma la viabilidad técnica del procesamiento integrado en las condiciones de la prueba.

El taller confirma tres principios: (1) la automatización requiere datos de entrada bien definidos; (2) los cálculos críticos deben ser reproducibles y separados de la redacción generativa; y (3) el mantenimiento y la verificación forman parte del diseño, no son actividades opcionales posteriores.

Lista final de evidencias

Evidencia	Verificación
Excel y error intencional	10 registros; TX008 duplicado
Análisis financiero	Ingresos \$1,200; egresos \$430; utilidad \$770
Flujo n8n	Nodos conectados y ejecutados
Publicación	AG23 v1.0 publicado
Ejecución automática	16:39:10; Succeeded; 39.578 s
Informe guardado	Confirmado en prueba previa; revisión final por archivo recomendada

Anexo práctico A. Reproducción del agente

Este anexo convierte el caso desarrollado en un procedimiento que otra persona puede seguir. Se necesita acceso a n8n, Google Drive, una cuenta de OpenAI con credenciales configuradas y el archivo de datos de entrenamiento. No publique las credenciales ni la información privada de la cuenta.

A.1 Preparar los datos y las carpetas

- Crear la carpeta AG23 - Agente IA Rentabilidad en Drive.
- Crear ENTRADA_EXCEL e INFORMES_GENERADOS dentro de la carpeta principal.
- Preparar Taller_1_Agente_IA_Datos.xlsx (10 movimientos ficticios).
- Mantener el duplicado TX008 para verificar que el agente lo detecta.

A.2 Configuración de n8n

Nodo	Ajuste principal
Google Drive Trigger	Changes Involving a Specific Folder; Folder: ENTRADA_EXCEL; Watch For: File Created
Download file	File → Download; By ID; expresión {{ \$json.id }}
Extract from File	Extract From XLS; campo binario data
Code in JavaScript	Procesamiento de registros: duplicados, ingresos, egresos y márgenes

Basic LLM Chain	Mensaje de usuario: {{ JSON.stringify(\$json) }}; OpenAI Chat Model conectado
Convert to File	Convert to Text File; salida binaria data
Upload file	File → Upload; Input data; Folder: INFORMES_GENERADOS

A.3 Código de referencia JavaScript

El siguiente ejemplo es didáctico y reproduce las reglas del ejercicio. No constituye una transcripción verificada del código original del nodo usado durante la demostración. Debe ejecutarse con los nombres de columnas indicados y comprobarse antes de utilizarlo en otros archivos.

```
const filas = $input.all().map(x => x.json);
const vistos = new Map(), incidencias = [], totales = {};
for (let i = 0; i < filas.length; i++) {
  const r = filas[i];
  const id = String(r.ID ?? '').trim();
  const vehiculo = String(r['Vehículo'] ?? '').trim();
  const tipo = String(r.Tipo ?? '').trim();
  const monto = Number(r['Monto USD']);
  if (!id || !vehiculo || !['Ingreso', 'Egreso'].includes(tipo) || !Number.isFinite(monto) || monto < 0) {
    incidencias.push({fila: i+2, id, problema: 'Invalido'});
    continue;
  }
  if (vistos.has(id)) {
    const exacto = JSON.stringify(vistos.get(id)) === JSON.stringify(r);
    incidencias.push({fila: i+2, id, problema: exacto ? 'Duplicado exacto' : 'ID con diferencias'});
    continue;
  }
  vistos.set(id, r);
  totales[vehiculo] ??= {ingresos: 0, egresos: 0};
  if (tipo === 'Ingreso') totales[vehiculo].ingresos += monto;
  else totales[vehiculo].egresos += monto;
}
const rentabilidad = Object.entries(totales).map(([vehiculo,v]) => ({
  vehiculo, ...v, utilidad: v.ingresos-v.egresos,
  margen: v.ingresos ? 100*(v.ingresos-v.egresos)/v.ingresos : null
}));
return [{json:{diagnostico:{originales:filas.length,
  validos:vistos.size, incidencias}, rentabilidad}}];
```

Nota: el número de fila suma 2 por la fila de cabecera en Excel. En producción hay que normalizar tipos y columnas, verificar transacciones con mismo ID pero contenido diferente y gestionar formatos incompatibles.

A.4 Instrucciones del modelo IA

System prompt sugerido: «Eres un analista de datos empresariales. Trabaja exclusivamente con el JSON recibido. No inventes cifras. Redacta: diagnóstico de calidad; tabla de ingresos, egresos, utilidad y margen por vehículo; empatados; tres recomendaciones; fórmulas y límites de la información. No presentes utilidad operativa como ganancia neta completa».

User prompt en Basic LLM Chain: {{ JSON.stringify(\$json) }}

A.5 Validación antes de publicar

- Verificar 10 movimientos originales, 9 válidos y duplicado TX008.

- Comprobar ingresos \$1,200, egresos \$430 y utilidad \$770.
- Corroborar empate de utilidad entre Corolla y Santa Fe (\$320 cada uno).
- Realizar prueba del Trigger y confirmar que la descarga usa el ID de ese evento.
- Publicar el workflow; subir un archivo de prueba nuevo sin pulsar Execute workflow.
- Revisar Executions y abrir el informe correspondiente a esa ejecución.

Anexo práctico B. Documentación técnica y puesta en servicio

Para adaptar esta solución a una empresa, conviene disponer del flujo n8n exportable, el código JavaScript, un conjunto de datos de prueba, los resultados de control y las instrucciones de instalación. Esta documentación facilita revisar la lógica, repetir las pruebas y mantener el sistema cuando cambian las fuentes de información.

Recurso	Propósito operativo
README.md	Descripción de la solución, requisitos de uso, arquitectura y reproducción
workflow/AG23_Rentabilidad.json	Configuración transportable de la automatización sin credenciales
src/calculo_rentabilidad.js	Reglas de cálculo y transformación para mantenimiento
data/Taller_1_Agente_IA_Datos.xlsx	Conjunto de prueba para validar la detección de incidencias
evidencias/	Resultados de pruebas, ejecución y conciliación
resultados/	Informes y controles para revisión administrativa
tests/	Casos de prueba de duplicados, valores vacíos y conciliación

Estado técnico: el flujo y una ejecución automática se comprobaron con datos ficticios. La exportación JSON del workflow, los archivos de código separados y las pruebas automatizadas se identifican como documentación adicional por completar antes de un despliegue empresarial. Las capturas y los ejemplos públicos deben ocultar credenciales, correos e identificadores privados.

Posibilidades de integración empresarial

La arquitectura integra n8n, JavaScript, OpenAI y Google Drive. Según las necesidades del negocio, puede ampliarse en proyectos posteriores con consultas SQL, análisis en Python o R y paneles de Power BI. Estas extensiones no forman parte de las funciones verificadas del caso AG23 y requieren diseño y pruebas independientes.

ACTUALIZACIÓN · Reportes profesionales HTML y PDF

Ampliación del caso AG23 · Experimento HTML y PDF · 8 de octubre de 2026

Propósito de la mejora

El agente inicial entregaba un informe de texto. En esta ampliación se experimentó con dos formas de generar un informe HTML visual: (A) una plantilla programada con JavaScript y (B) un documento HTML compuesto por OpenAI a partir del JSON previamente validado. El archivo HTML de la opción B se convirtió posteriormente a PDF mediante la función Imprimir de Google Chrome. No se automatizó la conversión a PDF en n8n.

Resultados comprobados de la ampliación

Generación programada	JavaScript generó el archivo AG23_Reporte_Rentabilidad.html ; se visualizó en Chrome y se guardó en Drive.
Generación asistida por IA	Basic LLM Chain1 con OpenAI produjo código HTML completo desde los indicadores del primer JavaScript.
Archivo HTML final de la prueba	Convert to File2 produjo AG23_Reporte_OpenAI.html; se descargó y abrió en Chrome.
Archivo PDF	El usuario convirtió ese HTML en PDF mediante Chrome. Se revisó un documento de 3 páginas.
Pendiente / fuera del alcance	La generación y carga automática del PDF desde n8n no se implementó.

Arquitectura del experimento

Excel → Google Drive / Extract from File → JavaScript (limpieza, validación, cálculo) → dos ramas:

A. OpenAI interpreta → JavaScript crea plantilla HTML → Convert to File → MIME text/html → Upload file a Google Drive.

B. Basic LLM Chain1 + OpenAI Chat Model1 genera HTML directamente → Convert to File2 → descarga del HTML → Chrome → Guardar como PDF (manual).

Ambas ramas conservan el cálculo financiero fuera del modelo de lenguaje. La rama B es un experimento de generación de presentación; requiere revisión antes de utilizarse con archivos distintos.

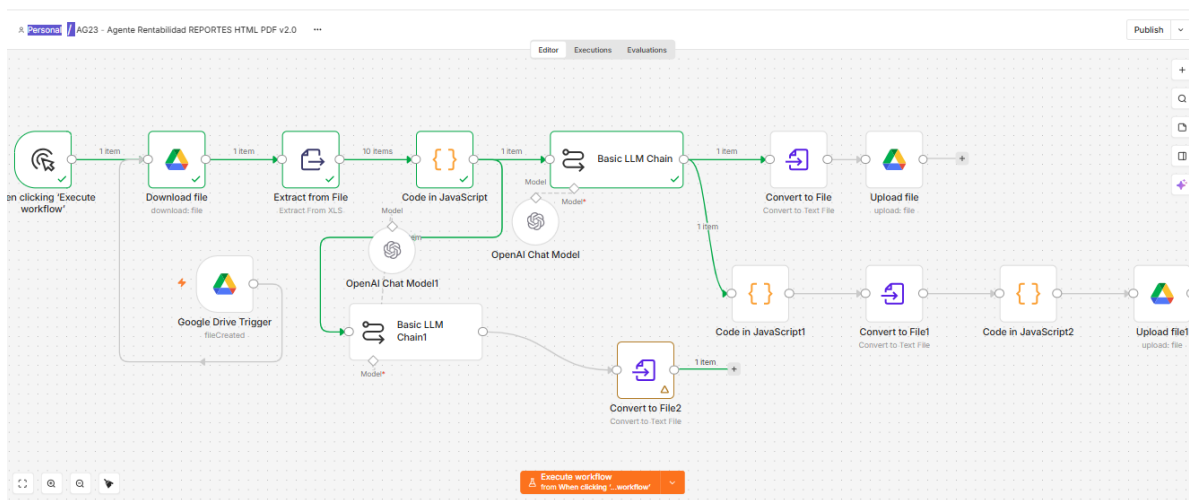


Figura 1. Ramas de comparación de informes en el editor de n8n, tras retirar PDFBolt.

Cómo reproducir el reporte generado por OpenAI

1. Partir de la salida de Code in JavaScript (un JSON con diagnóstico, rentabilidad y resumen).
2. Conectar una nueva rama Basic LLM Chain1 al nodo de cálculos, no a Extract from File.
3. Conectar OpenAI Chat Model1 a la entrada Model de Basic LLM Chain1.
4. Seleccionar Define below para el mensaje de usuario y usar la expresión `{{ JSON.stringify($json) }}`.
5. Añadir instrucciones System: generar HTML5 completo con CSS interno; tarjetas, barras, tabla, incidencias, conciliación y recomendaciones; solo con datos suministrados.
6. Solicitar USD con separador de miles y dos decimales, porcentajes con dos decimales, fecha no inventada, eliminación de etiquetas genéricas y aviso de datos ficticios.
7. Ejecutar Basic LLM Chain1 y confirmar que el campo text comienza por `<!DOCTYPE html>`.
8. Conectar Convert to File2; elegir Convert to Text File, Text Input Field = text, salida binaria = data y nombre AG23_Reporte_OpenAI.html.
9. Ejecutar Convert to File2; descargar el HTML y abrirlo en Chrome. Verificar todos los indicadores y secciones.
10. Para obtener el PDF manualmente: Chrome → Ctrl+P → Destino: Guardar como PDF → activar Gráficos de fondo → opcionalmente desactivar Encabezados y pies de página → guardar.

Criterios de validación y resultado

Indicador	Esperado	Verificado en PDF
Ingresos	\$1,200.00	\$1,200.00
Egresos	\$430.00	\$430.00
Utilidad operativa	\$770.00	\$770.00
Margen sobre ingresos	64.17 %	64.17 %
Originales / válidos / excluidos	10 / 9 / 1	10 / 9 / 1
Duplicado	TX008	TX008

Comprobaciones adicionales: la tabla por vehículo coincide con los cálculos; Corolla 2020 y Santa Fe 2018 empatan en utilidad (\$320.00), mientras Santa Fe 2018 tiene mayor ingreso (\$500.00). El informe declara explícitamente que los datos son ficticios.

Evidencias del producto obtenido

Las imágenes siguientes provienen del PDF entregado al finalizar el experimento, no de material de stock.

Evidencia PDF · Página 1 de 3

Resumen ejecutivo y gráfico de ingresos

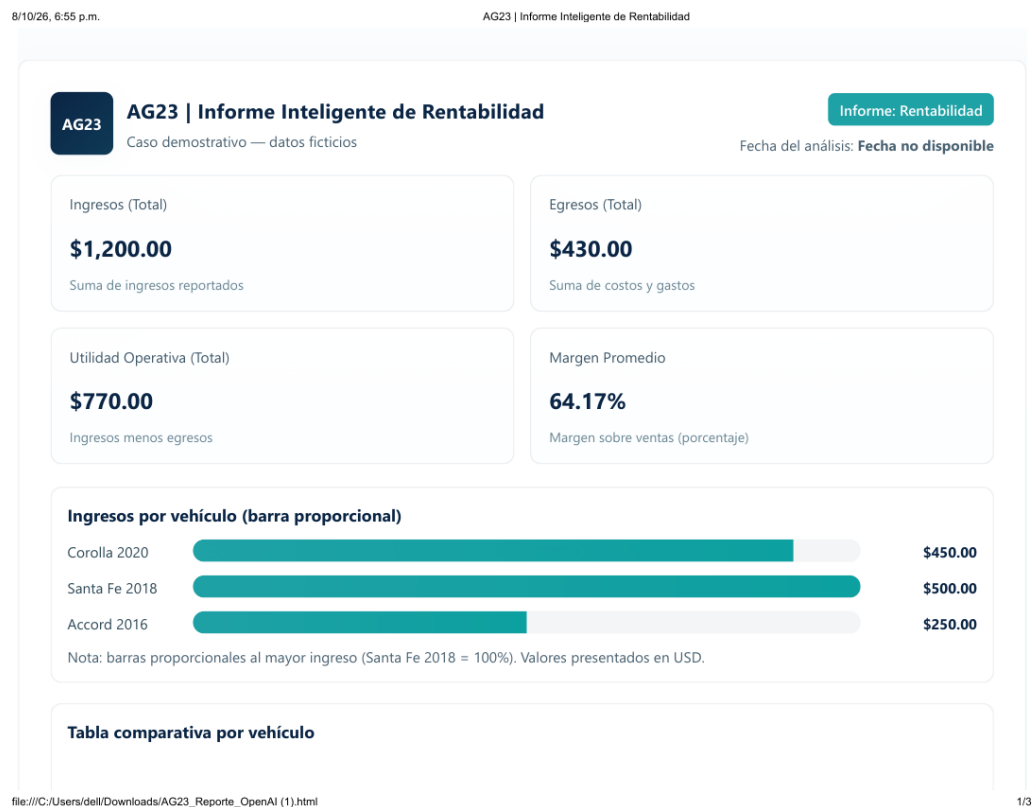


Figura 2. Captura real de la página 1 del reporte PDF de AG23.

Evidencia PDF · Página 2 de 3

Tabla financiera, conciliación y calidad de los datos

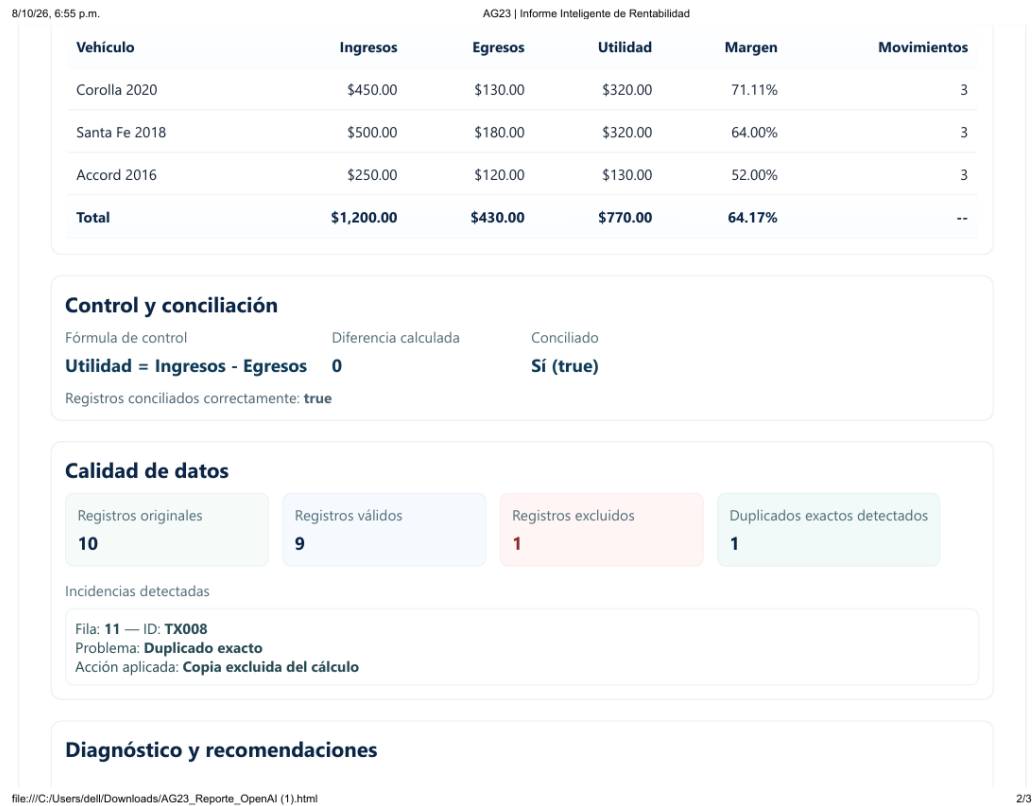


Figura 3. Captura real de la página 2 del reporte PDF de AG23.

Evidencia PDF · Página 3 de 3

Diagnóstico, recomendaciones y advertencia de uso

8/10/26, 6:55 p.m.

AG23 | Informe Inteligente de Rentabilidad

Diagnóstico (basado únicamente en los datos proporcionados):

- Se reportaron 10 registros originales; 9 fueron válidos y 1 fue excluido por ser un duplicado exacto (fila 11, ID TX008).
- La conciliación muestra diferencia 0 y estado conciliado = true; las cifras de ingresos, egresos y utilidad están consistentes con la fórmula indicada.
- El vehículo con mayor ingreso reportado es **Santa Fe 2018** (\$500.00) y el mayor margen relativo reportado es **Corolla 2020** (71.11%). El menor margen corresponde a **Accord 2016** (52.00%).

Recomendaciones operativas y de calidad de datos (fundamentadas en las incidencias detectadas):

- Implementar controles automáticos de duplicidad en el punto de captura (hash de campos clave) para reducir exclusiones manuales.
- Registrar un campo de auditoría que guarde motivo y acción aplicada por incidencia (ej. "Duplicado exacto — copia excluida") para seguimiento histórico.
- Establecer validaciones previas a la conciliación para asegurar que el total de ingresos y egresos provenga de registros válidos; mantener alertas cuando existan exclusiones.
- Monitorear vehículos con margen menor (en este conjunto, Accord 2016 con 52.00%) para identificar costos incrementales o ingresos faltantes.
- Documentar el proceso de conciliación y conservar la traza que respalde la diferencia de 0 y el estado conciliado.

Este informe es un caso demostrativo con datos ficticios. No usar para decisiones operativas sin validar fuentes originales.

Generado por: AG23 — Informe Inteligente de Rentabilidad

file:///C:/Users/dell/Downloads/AG23_Reporte_OpenAI (1).html

3/3

Figura 4. Captura real de la página 3 del reporte PDF de AG23.

Decisiones técnicas y utilidad empresarial

La arquitectura combina lectura de XLSX desde Google Drive, transformación y agregación en JavaScript, intercambio de datos JSON, interpretación con OpenAI y presentación mediante HTML/CSS. El control de duplicados y la conciliación favorecen reportes más consistentes para seguimiento operativo. La obtención del PDF se comprobó mediante una conversión manual en Chrome.

Para operar esta solución en otra empresa deben documentarse el flujo exportado y las reglas de cálculo efectivamente utilizadas, configurar accesos con permisos mínimos y probar los formatos de archivo reales. Las capturas y los resultados de muestra no sustituyen las pruebas de aceptación con información del negocio.

Ventaja de la generación programada: formato reproducible y control estable de HTML/CSS. Ventaja del enfoque asistido por OpenAI: rapidez para experimentar con presentaciones. Limitación del enfoque generativo: debe comprobarse que los importes, cálculos, contenido HTML y organización se mantengan correctos entre ejecuciones.

Mantenimiento y límites de la versión 2.0

La descarga de prueba se configuró temporalmente con un archivo Excel fijo. Antes de publicar una versión automática es necesario restaurar la descarga por ID del disparador (`{{ $json.id }}`), conectar la salida deseada a Google Drive, revisar los nodos no utilizados y repetir una prueba end-to-end con un archivo nuevo.

En esta etapa se decidió dejar la conversión PDF como una acción manual sencilla. Por tanto, este documento presenta el HTML automático dentro del flujo y el PDF como producto obtenido mediante Chrome; no afirma que ambos formatos se generen automáticamente en n8n.

Detalle editorial: la captura del PDF presenta encabezados y pies añadidos por Chrome (fecha, ruta local y número de página). Para una entrega pública pueden ocultarse desmarcando «Encabezados y pies de página», y conviene afinar los saltos de página del HTML.

Aplicación empresarial y requisitos de implementación

Aplicación propuesta: recepción periódica de archivos de ingresos y egresos, análisis de rentabilidad por unidad, detección de registros duplicados y entrega de indicadores para apoyar decisiones de operación. Los reportes pueden ayudar a identificar unidades de menor margen, revisar costos y priorizar validaciones de datos.

Para implementar la solución con un cliente se requiere acordar el formato de los archivos, definir reglas financieras y responsables de validación, revisar permisos y confidencialidad, establecer una frecuencia de ejecución y comprobar los resultados con casos representativos. No se presupone compatibilidad universal ni funcionamiento productivo sin estas pruebas.

Documentación para la entrega: exportación del workflow sin credenciales, código JavaScript utilizado, instrucciones de configuración, datos ficticios de prueba, tablas de conciliación, reportes HTML/PDF y procedimientos de atención de errores. Antes de compartir materiales deben ocultarse correos, identificadores y rutas privadas.

Alcance de esta edición: documento técnico en español del caso AG23. Su contenido puede traducirse posteriormente para proyectos en otros mercados, conservando las mismas condiciones de prueba y las limitaciones identificadas.

Producto final del proyecto

Se desarrolló un proceso de análisis de rentabilidad vehicular con limpieza y validación de movimientos en JavaScript, interpretación con OpenAI y presentación de resultados. El sistema original generó un TXT mediante una ejecución automática confirmada; además se probaron dos diseños de HTML, uno guardado en Google Drive, y se obtuvo un PDF de tres páginas mediante Chrome. El resultado ofrece una base técnica para adaptar el análisis periódico de rentabilidad a necesidades empresariales, sujeto a las validaciones e integraciones señaladas.