Módulo 2. Escenarios básicos en Make e integración con IA

CÓRDOBA CREA FUTURO

Lección 1 de 5

1. Consideraciones iniciales al crear un escenario desde cero

El primer paso para crear un escenario efectivo en Make es la planificación del proceso que se desea automatizar. Antes de abrir la herramienta, es fundamental identificar claramente las tareas o pasos individuales que componen ese proceso y el orden lógico en que deben ocurrir (Make, 2025a). Una vez definido el flujo, se traducen esos pasos a módulos dentro del escenario en Make, respetando la secuencia adecuada. En resumen, una automatización exitosa comienza entendiendo qué se va a automatizar: qué aplicaciones intervienen y qué acciones realizará cada una (Make, 2025a). Por ejemplo, si queremos notificar al equipo de ventas cuando Marketing agrega un prospecto en una hoja de cálculo, debemos identificar las aplicaciones (Google Sheets y Slack) y las acciones: el disparador será “nueva fila añadida en Google Sheets” y la acción será “enviar un mensaje en Slack” (Make, 2025a). Con esos pasos claros, podemos montarlos fácilmente en Make.

Dividir el proceso en pasos ayuda a reflejarlo como un conjunto de módulos conectados. Supongamos que deseamos automatizar la gestión de nuevos proyectos a partir del email: podemos diseñar un escenario que, cada cierto tiempo, revise Gmail en busca de correos marcados como “Nuevo Proyecto”; luego, por cada correo, cree automáticamente una tarjeta en Trello con los detalles del proyecto, y, finalmente, registre la información en una fila de Google Sheets (Lamphere, 2024). De esta manera, tres aplicaciones (Gmail, Trello, Google Sheets) trabajan integradas: Gmail aporta el evento inicial, Trello organiza la tarea y Sheets conserva un registro. Otro ejemplo sencillo: al recibir un correo en Gmail con un adjunto, subir ese archivo a Google Drive y luego enviar una notificación por Slack. En todos los casos, la clave es descomponer el flujo: identificar el módulo disparador (ej. “*Watch emails*” de Gmail para detectar un correo entrante) y los módulos de acción subsecuentes (ej. “Upload a File” en Drive, “*Post Message*” en Slack, etc.), conectándolos en el orden correcto. Esta metodología de pensar en pasos y traducirlos a módulos asegura que el escenario en Make refleje fielmente el proceso real que queremos automatizar (Make, 2025a).

En la etapa de diseño, también conviene empezar con flujos básicos de 2 o 3 módulos para familiarizarse con Make. Por

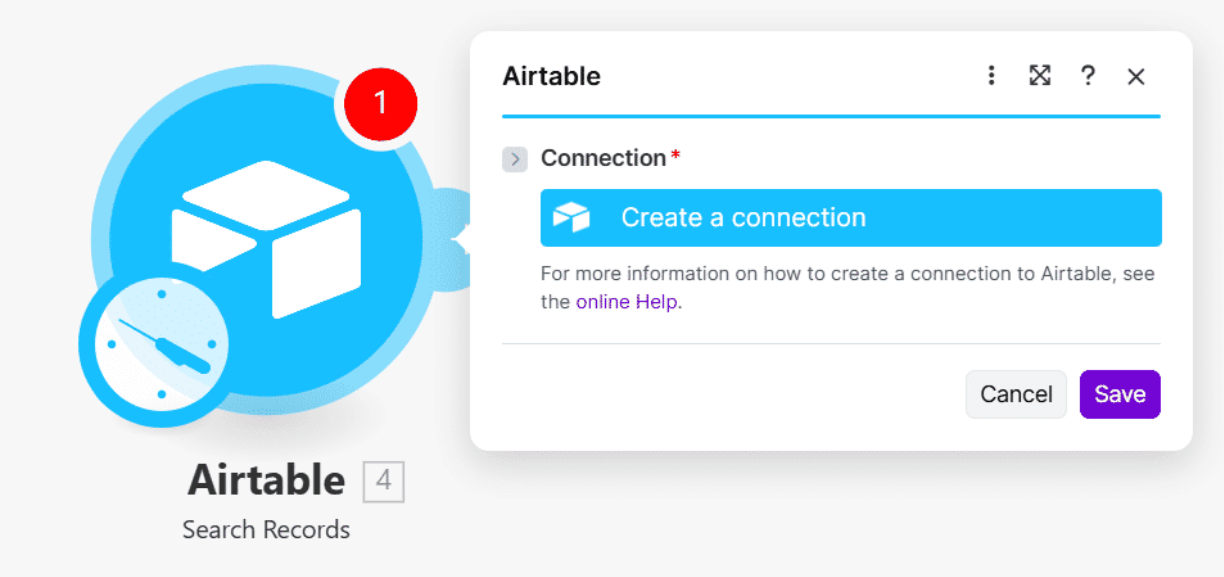
ejemplo, un escenario simple podría ser: “Cuando recibo un nuevo email en Gmail, tomar los datos y agregarlos a una hoja de Google Sheets”, o “Si se crea una nueva tarea en Trello, enviar una alerta por Gmail”. Estas automatizaciones iniciales permiten validar que sabemos conectar las aplicaciones y mapear la información entre ellas. A medida que ganemos confianza, podemos ampliar el escenario con más pasos o condiciones. Lo importante es recordar que cada escenario es un flujo de trabajo autónomo compuesto por módulos encadenados, donde cada módulo realiza una tarea específica y pasa la posta de datos al siguiente. Con una buena planificación previa, configurar el escenario se vuelve mucho más sencillo y libre de errores.

CONTINUAR

2. Conexiones en los módulos: accediendo a cuentas y servicios

Un concepto clave al construir escenarios es el de conexión. Las conexiones en Make representan la autorización para que Make acceda a nuestras cuentas en otras aplicaciones. Para que un módulo pueda ejecutar una acción en un servicio externo (leer un correo de Gmail o agregar una fila en Sheets), primero debemos conectar nuestra cuenta de esa aplicación a Make (Kommo, 2022). En la práctica, esto significa otorgar permisos a Make para usar la API de dicha aplicación en nuestro nombre. Al intentar configurar un módulo por primera vez, Make nos pedirá que creemos una conexión: por ejemplo, al añadir un módulo de Google Sheets, tendremos que iniciar sesión con nuestra cuenta de Google y autorizar a Make (vía OAuth) para que acceda a la aplicación. Del mismo modo, si usamos un módulo de Trello, es necesario proveer las credenciales de Trello (ya sea mediante OAuth o API *key/token*, según el método que esa aplicación utilice).

Figura 1: Airtable

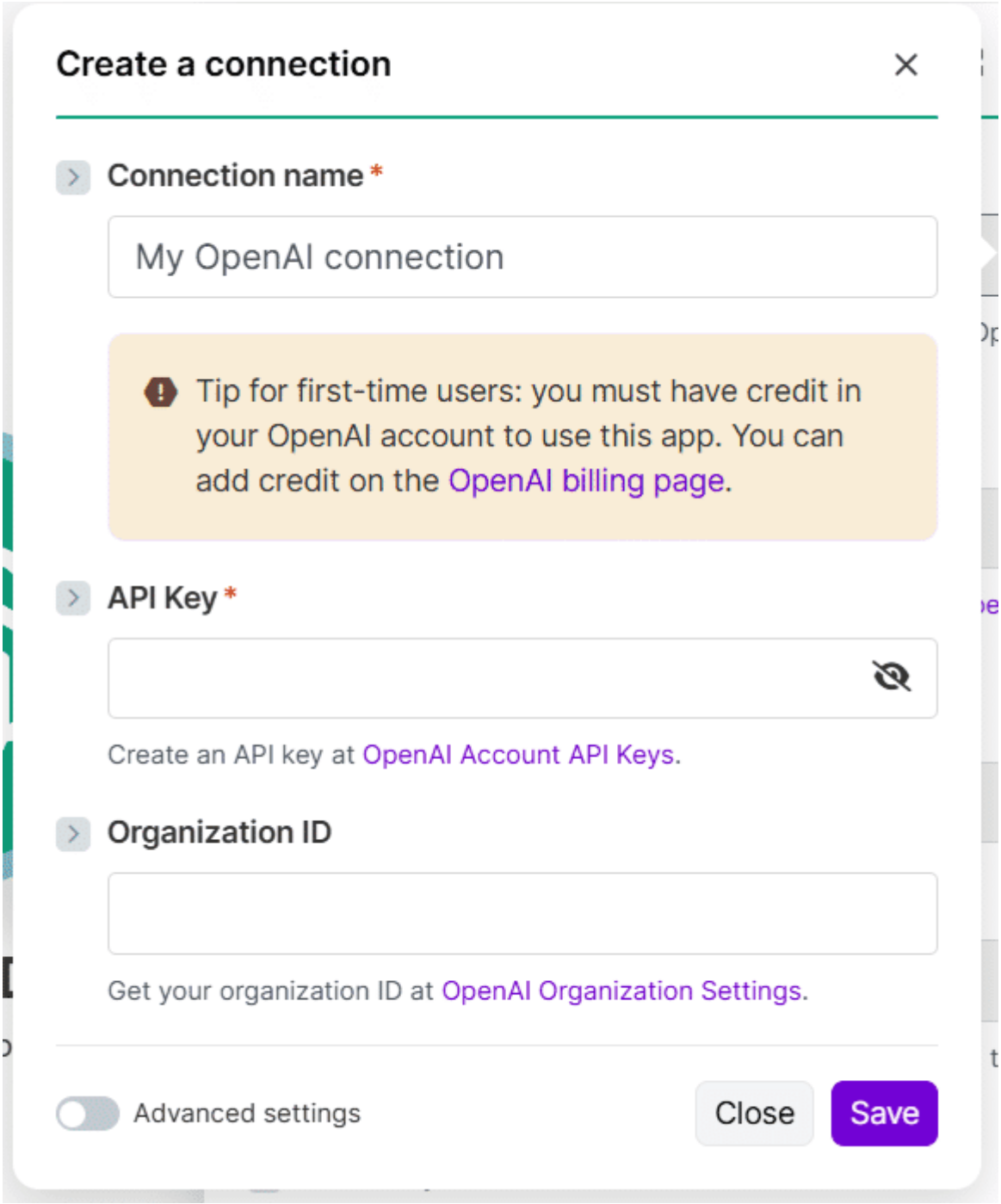


Fuente: elaboración propia.

Cada aplicación tiene su propio método de autenticación, y Make se adapta a ello. En muchos casos, la conexión es sencilla: basta con hacer clic en “Agregar conexión”, ingresar nuestras credenciales o iniciar sesión y otorgar permisos (esto ocurre con servicios como Google, Facebook, Microsoft, etc., que usan OAuth) (Make, 2025b). Otras aplicaciones requieren datos específicos: por ejemplo, puede que debamos introducir una API Key y quizá un secreto o token de acceso manualmente (Make, 2025b). Casos típicos son Trello, ciertos CRM, OpenAI y otros modelos de IA donde en

su panel de desarrollador obtenemos un API Key y un Token de usuario para copiar en Make. En aplicaciones más orientadas a empresas, puede pedirse también un Client ID/Secret de OAuth propio. En resumen, cada integración puede requerir un tipo distinto de credencial, pero Make nos guía mostrando exactamente qué datos necesita. Conviene revisar la documentación de Make o de la app externa para saber qué tipo de conexión usar y evitar errores de autenticación (Make, 2025b).

Figura 2: Crear una conexión



Fuente: elaboración propia.

Crear estas conexiones es un paso obligatorio la primera vez que usamos un módulo de esa app. Una vez guardada la conexión (se le suele asignar un nombre, p. ej. "Gmail – Personal" o "Trello – Cuenta Trabajo"), podemos reutilizarla en cualquier escenario y módulo que la requiera (Make, 2025b). Todas las conexiones creadas quedan listadas en la sección Conexiones de Make, donde es posible administrarlas: editar si, por ejemplo, cambió una contraseña o caducó un token, verificar si siguen funcionando o eliminarlas si ya no las usamos (Make, 2025b). En ese panel también se suele indicar qué módulos o escenarios están usando cada conexión, para gestionar el impacto de algún cambio. Un buen hábito es dar nombres descriptivos a las conexiones (p. ej., "Google Drive – Empresa" vs. "Google Drive – Personal"), sobre todo si trabajamos en equipo, para que quede claro qué credenciales se están aplicando. Otra consideración es que, como se explicó, los *tokens* y API *keys* permiten a Make acceder a nuestras cuentas, por lo que es de suma importancia que no compartamos estas claves y las mantengamos seguras para evitar posibles vulneraciones.

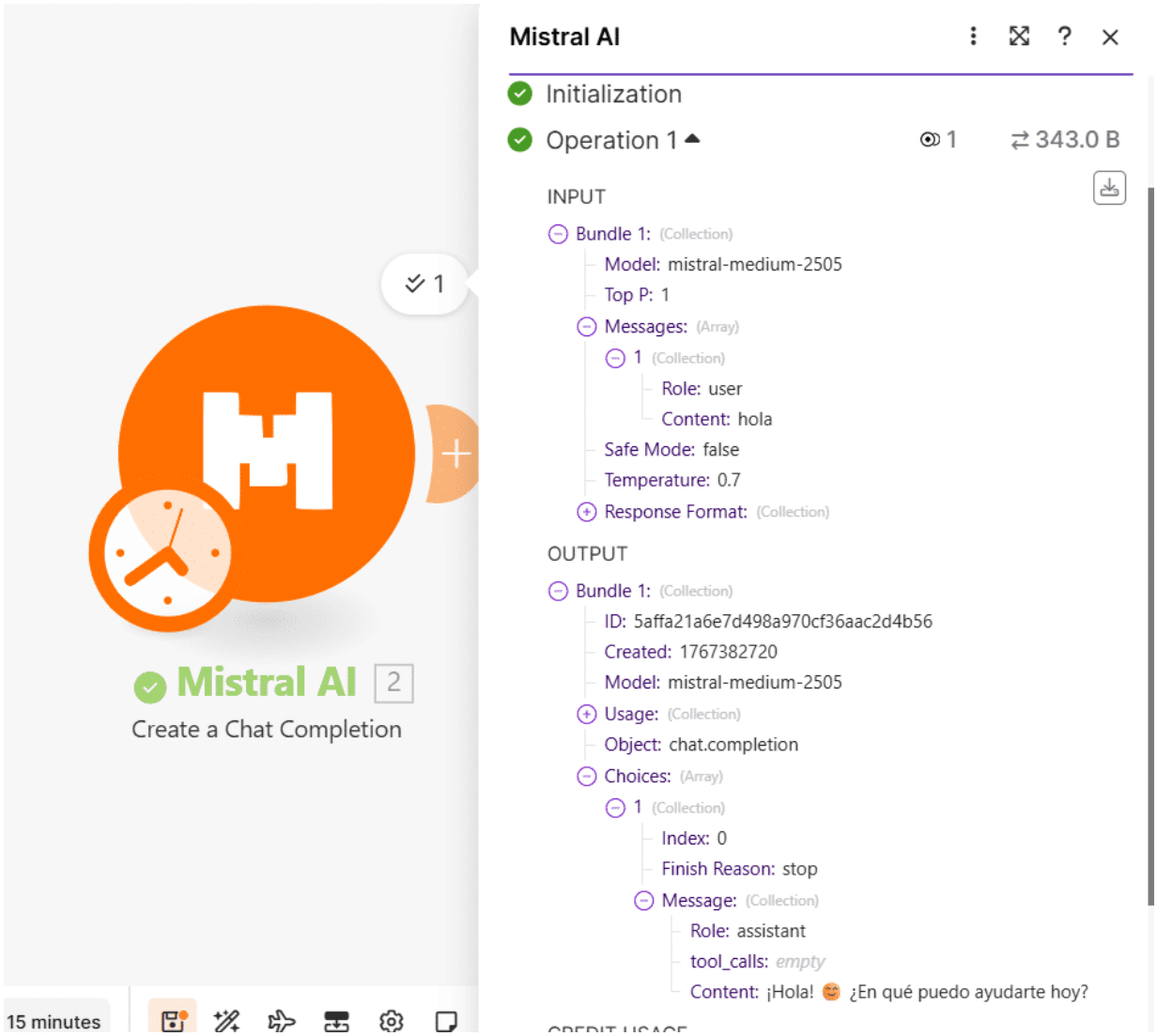
CONTINUAR

3. Mapeo de variables y manejo de datos en escenarios

Un aspecto importante al construir escenarios es entender cómo fluyen los datos de un módulo a otro y cómo se representan esos datos. Cada módulo, al ejecutarse, genera una salida (output) en forma de paquete de datos que Make llama *bundle* (Flow Automation School, 2025). Un *bundle* es básicamente un objeto con campos estructurados en formato clave-valor (JSON) que contiene la información procesada en ese paso (Flow Automation School, 2025). Podemos imaginarlo como una fila o registro con múltiples columnas: por ejemplo, si el módulo es "*Watch emails*" de Gmail, el *bundle* de salida podría contener campos como Remitente, Asunto, Contenido del correo, Fecha de recepción, etc. (cada uno con su valor correspondiente). Ese paquete de datos viaja por los siguientes módulos, de modo que puedan usar la información en sus propias acciones (Hernández, 2025). Cada vez que un módulo se ejecuta en Make, genera un *bundle* con la información resultante, disponible para el resto del escenario (Flow Automation

School, 2025). Para ver estos datos, debemos hacer clic en la burbuja con un número que aparece arriba de cada módulo una vez que ejecutamos el escenario. Comprender esto es crucial: el *bundle* contiene el conjunto de datos resultantes de una operación, y puede ser utilizado en módulos siguientes a través del mapeo de variables.

Figura 3. Mistral IA

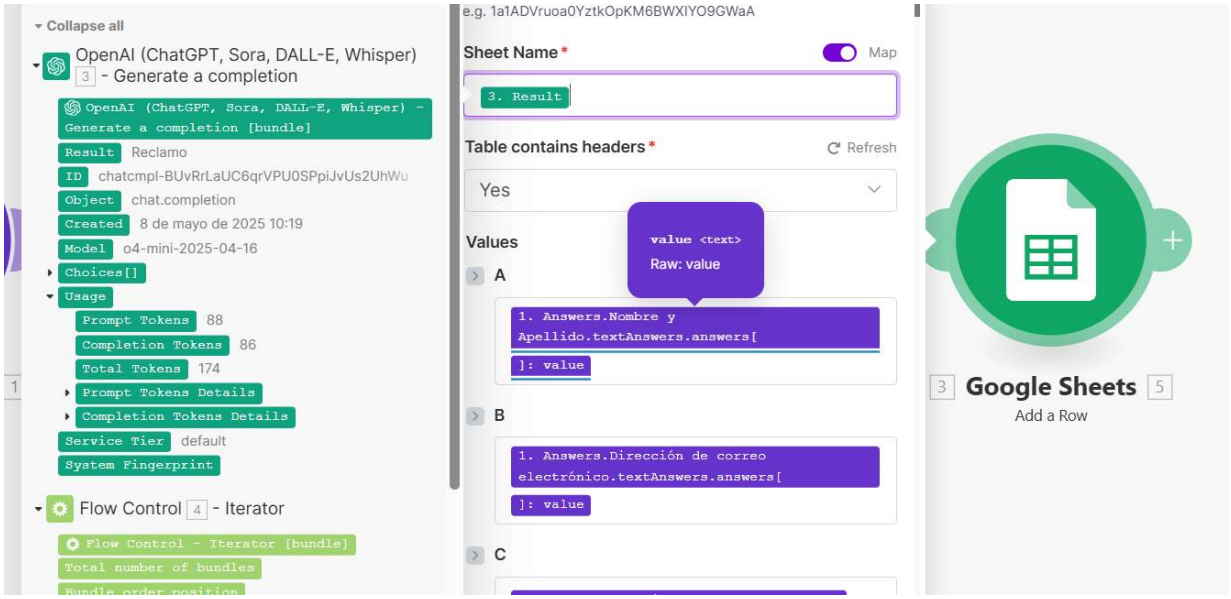


Fuente: elaboración propia.

¿Qué es mapear variables? Es seleccionar campos de la salida de un módulo previo para rellenar campos de entrada de un módulo siguiente. En Make, cuando configuramos un módulo, la mayoría de sus campos pueden recibir valores dinámicos provenientes de módulos anteriores (además de texto fijo que ingresemos). Al hacer clic en el recuadro de un campo, se abre el panel de mapeo, donde vemos la lista de

datos disponibles de módulos previos: allí simplemente elegimos cuál campo (o variable) queremos insertar (Dorouie, 2024). Por ejemplo, si en un escenario primero obtenemos una fila de Google Sheets (con campos como “Nombre”, “Email”, “Empresa”), y luego tenemos un módulo de Gmail “*Send email*”, podemos mapear el campo “Para:” con la variable Email proveniente del *bundle* de Sheets; el campo “Nombre” podemos insertarlo en el cuerpo del mensaje como parte del saludo, etc. Así, el email que se enviará tomará los datos directamente del *bundle* anterior. Esta forma de trabajar input-output permite encadenar la información sin tener que programar: Make se encarga de transportar cada valor del módulo A al módulo B según le indiquemos gráficamente.

Figura 4. Google Sheets



Fuente: elaboración propia.

Es importante destacar que cada campo en un bundle tiene un tipo de dato específico (también llamado formato); puede ser texto, número, fecha, booleano (sí/no), etc. (Consultoria Web, s. f.). Make muestra iconos o abreviaturas que indican el tipo esperado de cada campo cuando hacemos el mapeo. El texto se refiere a cualquier cadena de caracteres (letras, números, símbolos) – por ejemplo, nombres, descripciones o *emails* (Make, 2025c). Número abarca valores numéricos (enteros, decimales) utilizados para cantidades, importes, contadores, etc. (Make, 2025c). Fecha/Hora es un tipo especial para representar fechas (o fechas con hora) y suele manejarse en formatos específicos (Make utiliza por defecto ISO 8601 internamente, aunque muestra la fecha en formato legible según la zona horaria configurada) (Make, 2025c).

Booleano es un valor binario verdadero/falso (en la interfaz suele aparecer como opción Yes/No o true/false) útil para campos de verificación o condiciones lógicas (Make, 2025c). También hay tipos de dato más avanzados: por ejemplo, *array* (arreglo o lista de elementos) y Collection (colección u objeto). Un array es simplemente una lista ordenada de ítems, donde cada ítem puede ser un valor simple o incluso un objeto con varios campos (Flow Automation School, 2025). Por ejemplo, un *array* podría ser la lista de líneas de una factura o múltiples respuestas obtenidas de una API en un solo bundle. Una *collection* se refiere a un objeto estructurado dentro del *bundle,* es decir, un conjunto de subcampos clave-valor agrupados (en la práctica, el *bundle* entero ya es una colección de datos). En Make, a veces un campo de salida puede ser una *collection* (objeto JSON) o un array de objetos; en escenarios complejos debemos manejar estos con iteradores o agregadores, pero en escenarios básicos normalmente trabajamos con los campos simples directamente.

Los tipos de datos importan porque Make valida que mapear una variable tenga sentido en el campo destino. Si intentamos mapear un texto donde se espera un número, obtendríamos un error de tipo (o el escenario podría fallar) (Make, 2025c). Por ello, es clave asegurarse de que cada campo reciba el tipo correcto (Consultoría Web, s. f.). La

plataforma ayuda en cierta medida: por ejemplo, si mapeamos un booleano a un campo de texto, Make lo convertirá automáticamente a “true” o “false” (texto) (Make, 2025c); o si un número va a un texto, lo convertirá a su representación numérica en texto. Sin embargo, no siempre la conversión automática funciona o es deseable; por ejemplo, un texto “100 unidades” no podrá convertirse a número porque incluye letras, causando un error (Make, 2025c). La buena práctica es prestar atención a los iconos de tipo al mapear y, de ser necesario, usar funciones de conversión o formateo (Make ofrece fórmulas para transformar datos, p. ej., convertir texto a fecha, formatear fechas, etc.).

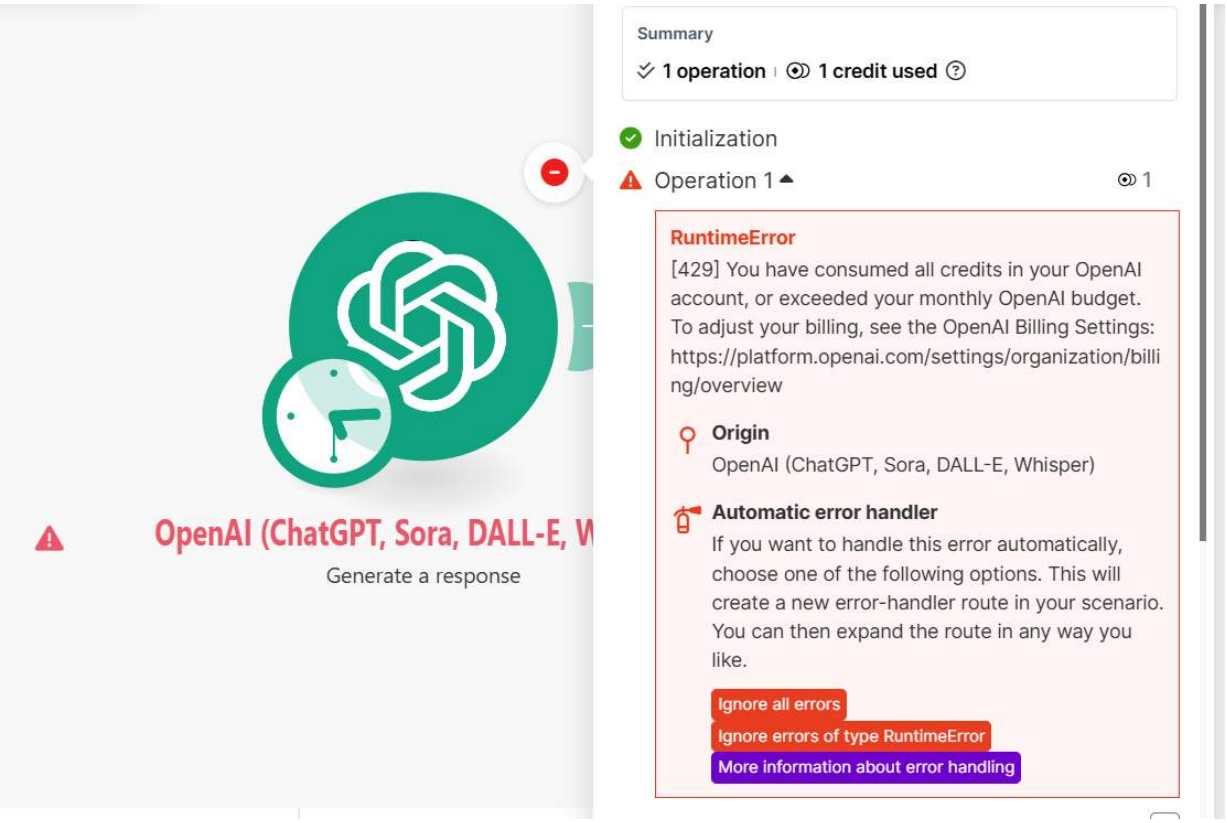
En resumen, en cada módulo debemos configurar qué datos ingresan (ya sea escritos manualmente o mapeados desde módulos previos) y conocer qué datos produce de salida. Dominar el mapeo de variables nos permite encadenar módulos de manera que el output de uno alimente el input del siguiente, logrando automatizaciones coherentes y libres de errores de datos.

Cabe mencionar también los *bundles* múltiples: si un módulo genera varios *bundles* en una operación (por ejemplo, leer 10 filas de una hoja genera 10 *bundles,* uno por fila), Make normalmente procesará cada *bundle* en iteración a través de los siguientes módulos. En otras palabras, los módulos siguientes se ejecutarán una vez por cada *bundle* entrante. Esto ocurre automáticamente si el módulo es iterador (p. ej., Search Rows de Google Sheets devuelve varios resultados), si no lo es, debemos utilizar el módulo Iterator que provee Make, pero eso lo veremos más adelante. Por eso a veces hablamos de primer *bundle*, segundo *bundle,* etc.; en escenarios simples quizá siempre haya un solo *bundle* a la vez, pero es bueno saber que puede haber múltiples *bundles* fluyendo simultáneamente si un módulo devuelve lista de resultados. Para empezar, con entender la lógica de *bundle* (paquete de datos) e ítem (campo individual) y cómo mapearlos correctamente, estaremos en condiciones de construir la mayoría de automatizaciones comunes.

En la práctica, al trabajar con Make es frecuente encontrarse con errores cuando el escenario está incompleto o mal configurado, o cuando la conexión con una aplicación externa no fue autorizada correctamente o quedó desactualizada (por ejemplo, por credenciales vencidas o permisos insuficientes). Este tipo de fallos suele manifestarse recién al momento de ejecutar el escenario, ya sea en una

prueba manual (*Run once*) o durante una ejecución automática. Cuando ocurre un error, Make lo señala visualmente colocando un cartel de advertencia sobre el módulo en el que se produjo el problema; al abrir ese aviso, el sistema muestra una explicación breve del tipo de error (por ejemplo, un problema de autenticación, una validación de datos o un campo requerido faltante). Esta retroalimentación es clave para aprender a depurar escenarios, pero en casos reales puede ser limitada: muchas veces el mensaje indica “qué falló” de manera general, aunque no siempre deja claro “por qué falló” o cuál fue la causa raíz.

Figura 5: Open IA



Fuente: elaboración propia.

Por esa razón, una buena práctica es incorporar asistencia con IA como parte del flujo de trabajo, especialmente cuando se está aprendiendo o cuando el escenario involucra varios módulos y mapeos complejos. ChatGPT (u otros modelos de IA) puede ayudar a interpretar mensajes de error, proponer hipótesis sobre el origen del problema y sugerir pasos concretos para solucionarlo (por ejemplo, revisar un campo obligatorio, validar el tipo de dato que se está mapeando, reconectar una cuenta o verificar permisos en la aplicación externa).

Para aprovechar esta asistencia, conviene copiar el mensaje de error completo, indicar qué módulo falló, describir brevemente qué se esperaba que ocurriera y compartir cómo está configurado el mapeo en ese punto del escenario. Incluso existen en la GPT *store*, GPT especialistas en Make, que contienen información sobre la documentación y son mucho más eficaces a la hora de asistir detalladamente sobre la herramienta.

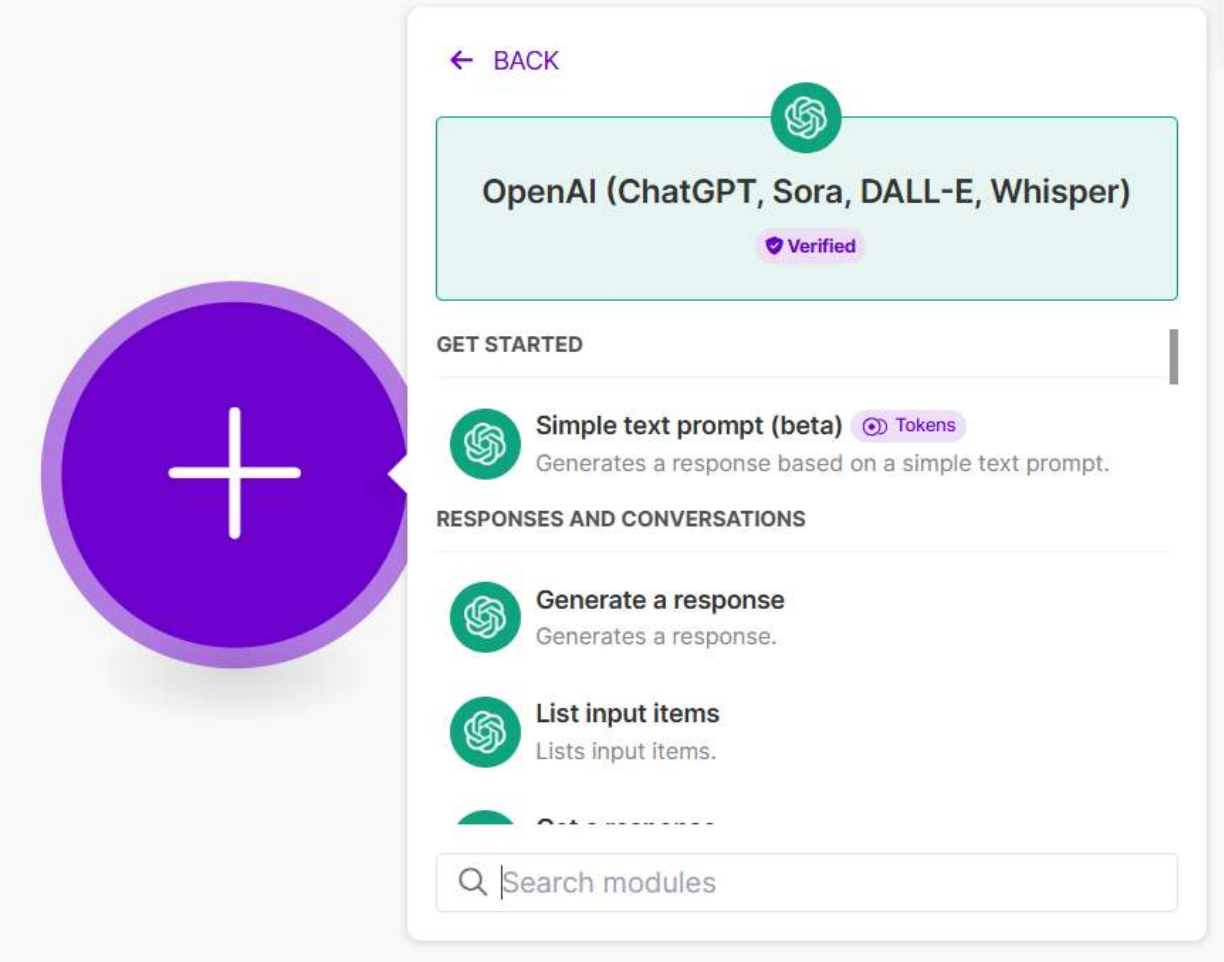
CONTINUAR

4. Conexión de Make con OpenAI: integrando inteligencia artificial en tus escenarios

Una de las características más potentes de las herramientas de automatización *no-code* es la posibilidad de integrar servicios de inteligencia artificial mediante módulos específicos o llamadas a API. En particular, Make ofrece una integración nativa con OpenAI, lo que permite incorporar capacidades de IA generativa en los flujos de trabajo. Esta integración se materializa en forma de módulos de OpenAI dentro de Make, que podemos agregar a nuestros escenarios como cualquier otro módulo. Al crear un nuevo escenario, si buscamos "OpenAI" en la lista de aplicaciones, veremos varios módulos disponibles, por ejemplo: "*Generate a completion*" (Generar un mensaje de texto al modelo), "*Message an asistant*" (hablar con un asistente), módulos de imágenes como "*Generate an image*" (crear imágenes a partir de texto) o "*Edit an image*" (edición de imágenes con IA), módulos de audio como "Transcribe audio" (usar Whisper para pasar audio a texto) o "*Text to speech*" (generar audio a partir de texto), y también "*Moderation*" (evaluar si un

texto/imagen tiene contenido inapropiado), entre otros. En esencia, casi todas las funciones que ofrece la API de OpenAI tienen un módulo correspondiente en Make.

Figura 6: Open IA 2



Fuente: elaboración propia.

Ahora bien, para utilizar estos módulos de OpenAI en Make, es necesario conectar nuestra cuenta de OpenAI a Make, de modo similar a como conectamos cualquier otra aplicación. OpenAI no usa OAuth para terceros, sino que se basa en API Keys. Por ello, la primera vez que agreguemos un módulo de OpenAI en Make, nos pedirá crear una conexión introduciendo nuestra API Key privada de OpenAI (y opcionalmente el ID de organización). En la práctica, los pasos son: (1) disponer de una cuenta en OpenAI (si no, registrarse en platform.openai.com), (2) en Make, agregar, por ejemplo, el módulo "Generate a completion" y clicar en "Agregar nueva conexión", (3) nombrar la conexión (ej. "OpenAI Personal") y luego Make mostrará campos para pegar la API Key y Org ID (de Brito Fontes, 2024). Junto a estos campos habrá un enlace directo a OpenAI para obtener dichas credenciales; al hacer clic, se abre la página de API Keys de OpenAI. Ahí generamos una nueva clave secreta (una cadena de caracteres alfanumérica), la copiamos y pegamos en Make (de Brito Fontes, 2024). Guardamos la conexión y listo: Make ya puede comunicarse con la API de OpenAI usando nuestros datos de acceso.

Es importante mencionar el tema de los costos. A diferencia de muchas *apps* estándar, OpenAI no tiene un plan "gratuito" ilimitado; su API funciona bajo un modelo de pago por uso, medido típicamente en *tokens* consumidos (fragmentos de texto). Esto implica que, para utilizar los modelos se deben tener créditos en la cuenta de OpenAI. De hecho, tras generar nuestra API Key, si intentamos usarla sin saldo, obtendríamos error; primero hay que cargar crédito en la cuenta (Sharma, 2024). En general, se debe registrar una tarjeta de crédito en OpenAI y comprar créditos prepagos para habilitar el uso de la API (Sharma, 2024; de Brito Fontes, 2024). Make no cobra extra por integrar OpenAI, pero obviamente el uso de los módulos de IA sí genera costos en la cuenta de OpenAI. Por eso, antes de usar intensivamente estos módulos, es recomendable configurar límites de gasto y alertas en la plataforma de OpenAI (de Brito Fontes, 2024).

Vale aclarar que OpenAI no es la única opción de IA. Make puede conectarse con cualquier otra API de IA como Gemini, Claude, etc. De hecho, Mistral AI proporciona su API de forma gratuita para el uso de sus modelos de forma limitada. Por lo que puede ser de utilidad para conectarlo con Make sin

incurrir en gastos, al menos en los primeros usos de la herramienta. La forma de conexión es similar a la de OpenAI, obteniendo una API key en la plataforma de Mistral AI. Como todas las guías para conectar las aplicaciones, esta se encuentra en la documentación de Make: https://apps.make.com/mistral-ai.

Nota aclaratoria sobre uso de IA

Este material fue asistido con herramientas de IA generativa para tareas de borrador, síntesis, reescritura y apoyo en la organización de contenidos. Cada sección fue revisada, editada y validada por el equipo humano, que verificó la precisión conceptual, la coherencia pedagógica y las fuentes citadas. Se invita a contrastar con las referencias bibliográficas incluidas y la documentación oficial. Dado que los modelos de IA evolucionan con rapidez, ciertas especificaciones técnicas podrían actualizarse; este texto refleja el estado del conocimiento al momento de su elaboración.

CONTINUAR

Referencias

Consultoria Web. (s. f.). Empezando con Make: Conceptos básicos. Consultoria Web. https://blog.consultoriaweb.cl/tutorial-automatizacion-make/

DataCamp. (2023, 18 de octubre). A beginner's guide to the OpenAI API: Hands-on tutorial and best practices. DataCamp. https://www.datacamp.com/tutorial/guide-to-openai-api-on-tutorial-best-practices

de Brito Fontes, F. (2024, 25 de julio). Cómo conectar el chat de OpenAI con Make. Consultoria Web. https://blog.consultoriaweb.cl/conectar-openai-a-make/

Dorouie, P. (2024, 6 de febrero). Understanding bundles, arrays & collections in Make.com. The AI Automators. https://www.theaiautomators.com/understanding-bundles-arrays-and-collections-in-makecom/

Flow Automation School. (2025, 10 de marzo). Como usar los Iterators y Aggregators en Make. Flow Automation School. https://letsflowas.com/blog/como-usar-los-iterators-y-aggregators-en-make/

Hernández, A. (2025, 26 de mayo). Guía básica de marketing automation con Make. ZeroMoment Marketing. https://zeromoment.marketing/blog/marketing-digital/marketing-automation-con-make/

Kommo. (2022, 14 de septiembre). Make (antes Integromat): ¿Cómo se utiliza? Kommo. https://www.kommo.com/es/recursos/integraciones/make-antes-integromat-como-se-utiliza/

Lamphere, D. (2024, 6 de junio). Automating project management with Make.com: Email to Trello and Google Sheets. ThinkBot Agency. https://thinkbot.agency/blog/automate-gmail-trello-google-sheets-make-com

Make. (2025a). Step 1. Plan your scenario. Make Help Center. https://help.make.com/step-1-plan-your-scenario

Make. (2025b). Connect an application. Make Help Center. https://help.make.com/connect-an-application

Make. (2025c). Item data types. Make Help Center. https://help.make.com/item-data-types

Sharma, A. (2024, 19 de octubre). How to generate your own OpenAI API key and add credits? Analytics Vidhya. https://www.analyticsvidhya.com/blog/2024/10/openai-api-key-and-add-credits/

CONTINUAR