Módulo 1. Agentes de inteligencia artificial y automatización no-code

CÓRDOBA CREA FUTURO

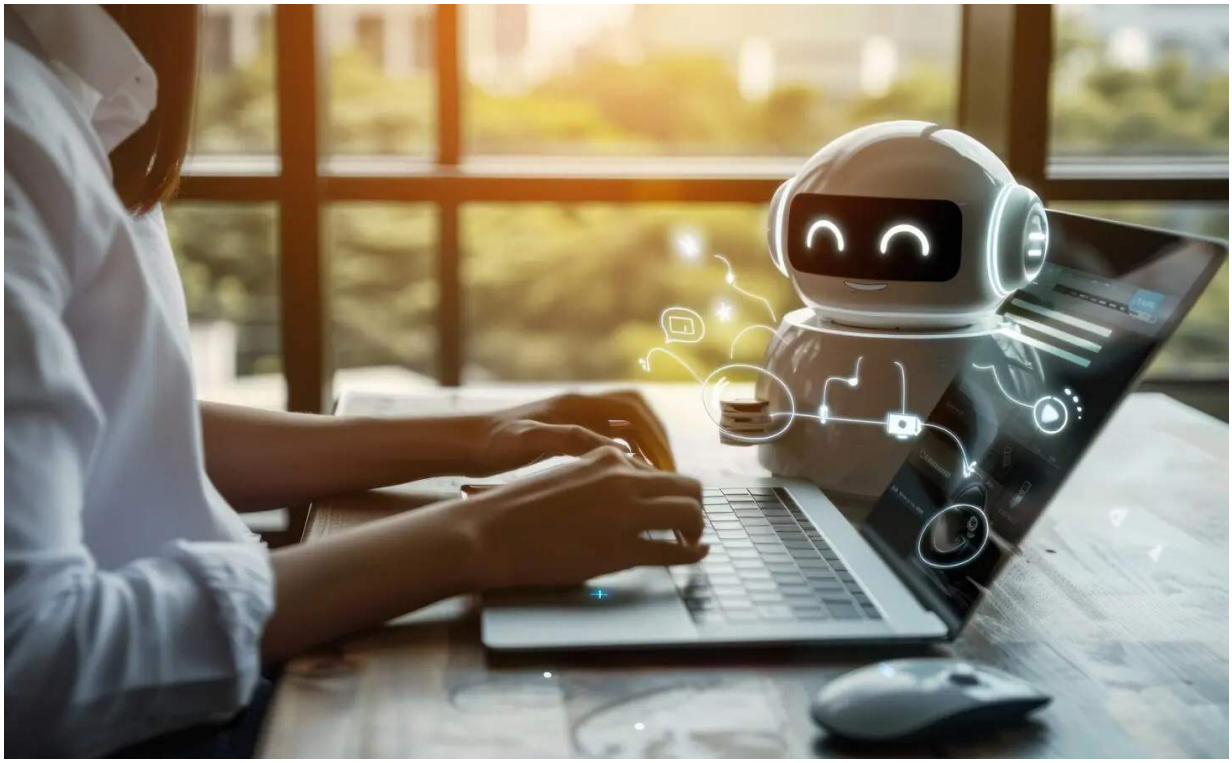
1. Introducción a los agentes de inteligencia artificial

Un agente de inteligencia artificial es típicamente definido como un sistema de *software* autónomo capaz de percibir su entorno, procesar información y actuar de forma orientada a objetivos específicos (Manzanera, Abarca e Iñigo, 2025; Amazon Web Services, s.f.).

Es fundamental entender la diferencia de programas tradicionales con secuencias fijas, ya que, un agente de IA puede tomar decisiones racionales basadas en sus percepciones y datos recibidos, eligiendo independientemente las acciones más apropiadas para cumplir las metas que se le hayan predefinido. En otras palabras, los humanos establecen los objetivos generales, pero el agente de IA decide por sí mismo cómo lograr dichos objetivos, ajustando su comportamiento según las circunstancias.

Así como se diferencia de los programas tradicionales, también podemos encontrar distinciones con los chats con IA (ChatGPT, Gemini) y los asistentes como los GPT que vimos anteriormente. Como ya sabemos, estos sí pueden tomar decisiones en base a la información, con el fin de elaborar un texto en lenguaje natural como respuesta al usuario. La diferencia principal, entonces, está en la capacidad de ejecutar una acción en el entorno. Las respuestas normales de ChatGPT no son más que texto; sin embargo, un agente de inteligencia artificial puede llevar a cabo acciones e influir en el entorno, lo que le permite ejecutar procesos enteros de forma autónoma. En dicho caso, en vez de copiar el texto generado por GPT para un correo y enviarlo manualmente, el agente interpretaría el destinatario y lo enviaría automáticamente.

Figura 1: Agentes de IA



Fuente: [imagen sin título sobre agentes de IA], (s. f.). https://bit.ly/4bzWJD3

CARACTERÍSTICAS FUNDAMENTALES DE UN AGENTE DE IA | EJEMPLOS ACTUALES DE AGENTES DE IA | BENEFICIOS EN CONTEXTOS PROFESIONALES

Los agentes inteligentes poseen una serie de capacidades clave que les permiten operar de manera autónoma en entornos complejos. Entre las fundamentales se incluyen:

- **Percepción del entorno:** habilidad para captar información del medio en el que operan, ya sea mediante sensores físicos o a través de entradas de datos y señales digitales. Esta percepción proporciona al agente datos sobre el estado del entorno o del problema a resolver (Manzanera, Abarca e Iñigo, 2025). Por ejemplo, un agente puede "leer" entradas de usuario, detectar objetos vía cámara o recibir eventos de *software* como *input*.

- **Razonamiento y toma de decisiones autónoma:** capacidad de analizar la información percibida, evaluar opciones y decidir cursos de acción de forma racional. Un agente de IA suele emplear algoritmos para inferir la mejor acción siguiente que se acerque al cumplimiento de sus objetivos (Manzanera, Abarca e Iñigo, 2025). Este proceso implica evaluar distintas posibilidades conforme a las reglas u objetivos definidos, y seleccionar la acción óptima dado el contexto. A diferencia de sistemas rígidos, los agentes pueden adaptarse a situaciones dinámicas y tomar decisiones no preprogramadas explícitamente.
- **Capacidad de actuación:** facultad de ejecutar acciones que influyen en el entorno o en otros sistemas, en función de las decisiones tomadas. Esto abarca desde acciones físicas (mover un mecanismo, enviar una señal) hasta acciones en entornos digitales (enviar un mensaje, actualizar una base de datos) (Manzanera, Abarca e Iñigo, 2025). La acción es el medio por el cual el agente implementa sus decisiones y produce un efecto tangible orientado a la meta trazada.

Estas tres características trabajan en conjunto como ciclo básico de funcionamiento de un agente inteligente. Adicionalmente, los agentes más avanzados suelen incorporar mecanismos de aprendizaje y adaptación. Gracias a técnicas de aprendizaje automático, un agente puede mejorar su desempeño con la experiencia, ajustando su comportamiento a partir de resultados previos (retroalimentación) (Google Cloud, 2025). Esta capacidad de autorrefinamiento diferencia a los sistemas de IA avanzados, ya que les permite evolucionar y optimizar sus funciones con el tiempo.

Hoy en día encontramos dos tipos de agentes inteligentes en diversos contextos profesionales y cotidianos, tales como:

- **Sistemas de conducción autónoma y robótica inteligente:** vehículos autodirigidos y robots equipados con IA que perciben su entorno físico mediante sensores, toman decisiones en tiempo real para navegar de forma segura, y actúan controlando aceleración, dirección u otras funciones mecánicas. Un ejemplo es un coche autónomo que detecta obstáculos en la vía y decide maniobrar para evitarlos conforme a las normas de tráfico (Amazon Web Services, s. f.). Aquí se manifiestan las instancias de percibir información del entorno, tomar decisiones y ejecutar acciones. De modo similar, robots industriales o *drones* inteligentes perciben condiciones de su entorno de trabajo y actúan adaptativamente.
- **Gestores autónomos de procesos empresariales:** agentes que supervisan y coordinan flujos de trabajo complejos dentro de una organización. Estos sistemas pueden integrar múltiples aplicaciones (un CRM, un ERP y una plataforma de *marketing*) recopilando datos de cada una, tomando decisiones para orquestar el proceso (p. ej., detectar cuellos de botella, reasignar tareas) y ejecutando acciones como disparar notificaciones o actualizar registros (Manzanera, Abarca e Iñigo, 2025). Su objetivo es optimizar procesos de negocio de principio a fin sin intervención humana continua.

En este último tipo de agentes nos centraremos en este curso por su accesibilidad económica y funcional.

CARACTERÍSTICAS FUNDAMENTALES DE UN AGENTE DE IA	EJEMPLOS ACTUALES DE AGENTES DE IA	BENEFICIOS EN CONTEXTOS PROFESIONALES

La incorporación de agentes de IA en entornos empresariales y productivos conlleva una serie de ventajas significativas:

- **Incremento de la productividad:** los agentes de IA pueden encargarse de tareas repetitivas, operativas o de gran volumen de datos sin cansancio ni errores humanos, liberando a los equipos para enfocarse en actividades de mayor valor añadido. Al delegar labores rutinarias al agente, las organizaciones aceleran la consecución de objetivos y mejoran la eficiencia global de los procesos (Amazon Web Services, s. f.).
- **Reducción de costos operativos:** automatizar procesos mediante agentes inteligentes disminuye costes asociados a ineficiencias, errores manuales y retrabajos. Los agentes siguen modelos consistentes y pueden adaptarse a entornos cambiantes, lo cual evita gastos innecesarios (Amazon Web Services, s. f.).
- **Toma de decisiones más informada:** los agentes avanzados pueden analizar enormes cantidades de datos en tiempo real apoyándose en técnicas de machine learning. Esto permite extraer patrones, tendencias y predicciones con mucha rapidez, sirviendo de apoyo a la toma de decisiones empresariales (Amazon Web Services, s. f.).

- **Escalabilidad:** un agente puede atender grandes volúmenes de solicitudes, eventos o datos en paralelo, manteniendo consistencia en la ejecución y habilitando que un proceso funcione 24/7. Esta capacidad es especialmente relevante cuando una organización necesita pasar de pruebas puntuales a despliegues en producción, donde importan la gestión, la observabilidad, el control de costos y la posibilidad de escalar la infraestructura (para picos de demanda en campañas, soporte masivo o flujos de *back-office*).
- **Personalización:** un beneficio distintivo de los agentes de IA es la posibilidad de personalizarlos en cuanto a comportamiento y lógica de ejecución, es decir, definir **cómo** llevar a cabo su trabajo. Esta personalización no se limita a la interfaz o al contenido generado, sino que abarca la configuración de objetivos, prioridades, reglas de decisión, fuentes de datos, herramientas permitidas y secuencias de acción. En la práctica, un mismo agente puede comportarse de manera diferente según el contexto organizacional: por ejemplo, ejecutar un flujo con mayor nivel de validación, seguir políticas internas específicas, aplicar criterios conservadores o agresivos, o delegar ciertas decisiones a intervención humana. Este enfoque convierte al agente en una pieza configurable del proceso, alineada con la lógica del negocio y no en una automatización rígida.

CONTINUAR

2. Herramientas no-code para automatización

El término "*no-code*" (sin código) se refiere a un enfoque de desarrollo de *software* en el cual es posible construir aplicaciones y automatizaciones sin necesidad de programar, empleando interfaces gráficas y componentes predefinidos. Es como construir con bloques de LEGO en lugar de crear los ladrillos desde cero: el usuario diseña visualmente la lógica de lo que desea lograr, mientras la plataforma *no-code* se encarga de generar por detrás el código y gestionar la infraestructura necesaria (Google Cloud, s. f.). La plataforma *no-code* traduce ese diseño visual en una aplicación real operativa, manejando automáticamente aspectos técnicos complejos (código, servidores, bases de datos) en segundo plano (Google Cloud, s. f.).

Por eso a la actividad de automatizar con herramientas *no-code* le solemos decir "**orquestar**", y no "programar", ya que

aquella programación sucede justamente en segundo plano a partir de nuestro diseño inicial del flujo.

Figura 2: API

Fuente: [imagen sin título sobre API], (s. f.).

El movimiento *no-code* ha cobrado gran relevancia en el mundo empresarial porque democratiza la creación de soluciones tecnológicas. Permite que personal no técnico (como nosotros) puedan desarrollar por sí mismos aplicaciones, integraciones y automatizaciones para resolver problemas de negocio, sin depender

exclusivamente de personal técnico (Google Cloud, s. f.; Código Media, s. f.). Esto resulta crítico ante la creciente demanda de *software* en las empresas y la limitada disponibilidad de programadores expertos. Las plataformas sin código proporcionan una vía para acelerar la transformación digital aprovechando el conocimiento de quienes mejor entienden los procesos (los usuarios finales), reduciendo la brecha entre necesidades de negocio y soluciones tecnológicas.

En esencia, las plataformas *no-code* de automatización (también llamadas de "integración" u "orquestación") permiten conectar distintas aplicaciones entre sí y definir reglas para que esas tareas se ejecuten automáticamente, ahorrando tiempo y minimizando errores.

En cualquier empresa existen multitud de tareas repetitivas o integraciones entre sistemas que pueden beneficiarse de la automatización, desde enviar correos de bienvenida a nuevos clientes, publicar contenido en varias redes sociales a la vez, hasta consolidar datos de diferentes fuentes en un reporte único.

- **Baja barrera de entrada:** las plataformas *no-code* suelen ser intuitivas, con interfaces de "arrastrar y soltar" y plantillas preconfiguradas. Esto hace posible que personas con conocimientos informáticos generales (pero sin saber programar) puedan usarlas (Código Media, s. f.). La democratización de la automatización implica que áreas como marketing, operaciones o finanzas pueden crear sus propios flujos automatizados sin esperar al desarrollo de *software* formal.
- **Mayor velocidad de desarrollo:** al reutilizar componentes ya hechos y eliminar la codificación manual, se reduce drásticamente el tiempo necesario para pasar de una idea a una solución funcional. Las empresas pueden responder con agilidad a demandas cambiantes del mercado, implementando procesos automatizados en días o incluso horas, en lugar de las semanas o meses que tomaría un desarrollo tradicional (Código Media, s. f.). Esto acelera la innovación y la capacidad de iterar soluciones rápidamente.
- **Costos más bajos:** el desarrollo sin código tiende a ser más económico, ya que ahorra horas de programación especializada. Además, muchas herramientas ofrecen modelos de precios escalables (incluso planes gratuitos limitados), por lo que implementar una automatización inicial puede no requerir gran inversión. Al necesitar menos recursos de desarrollo, las empresas obtienen ahorros significativos en la creación de soluciones digitales (Código Media, s. f.).
- **Inclusividad y autonomía de las áreas de negocio:** con *no-code*, profesionales de distintas áreas (marketing, ventas, RR. HH., etc.) pueden involucrarse directamente en la creación de las

soluciones que necesitan. Esto fomenta equipos multidisciplinares y reduce la dependencia total del departamento de TI para cada requerimiento (Código Media, s. f.). El personal que mejor conoce un proceso puede automatizarlo por sí mismo, logrando exactamente lo que necesita y ajustándolo cuantas veces haga falta.

Cabe mencionar que, si bien las herramientas *no-code* ofrecen enormes ventajas, también presentan algunas **limitaciones**. Por ejemplo, suelen estar acotadas a las capacidades que provee la plataforma: si se requiere una funcionalidad muy específica fuera de lo común, puede no ser posible implementarla sin código. Asimismo, existe cierta dependencia del proveedor: migrar un flujo creado en una plataforma *no-code* a otra plataforma o al código tradicional puede resultar complejo. También puede haber restricciones de personalización avanzada y, en algunos casos, dificultades para escalar un desarrollo a nivel de gran empresa si el volumen de datos o transacciones crece mucho (Código Media, s. f.). No obstante, las plataformas *no-code* evolucionan constantemente ampliando sus capacidades, y muchas permiten extensiones mediante código (*low-code*) para cubrir huecos, por lo que en la práctica se han vuelto suficientemente robustas para una gran variedad de aplicaciones empresariales.

EN EL ENTORNO LABORAL ACTUAL, ESTE TIPO DE HERRAMIENTAS ES IMPORTANTE POR VARIAS RAZONES:

PLATAFORMAS LÍDERES DE AUTOMATIZACIÓN NO-CODE: ZAPIER, MAKE Y N8N

MAKE: POTENCIA Y FLEXIBILIDAD CON ENFOQUE VISUAL

Existen numerosas herramientas *no-code* enfocadas en la automatización e integración de aplicaciones. Entre las más populares y ampliamente utilizadas destacan Zapier, Make

(anteriormente Integromat) y n8n, cada una con un enfoque y público objetivo ligeramente distinto. A continuación presentamos y comparamos estas tres plataformas, resaltando sus características, fortalezas y posibles limitaciones:

Zapier: automatización sencilla y catálogo masivo de integraciones

Zapier es probablemente la plataforma pionera y más conocida en el ámbito de la automatización *no-code*. Su principal fortaleza radica en la facilidad de uso: está diseñada para que cualquier persona pueda crear flujos de trabajo simples en cuestión de minutos mediante una interfaz web muy amigable (ActivDev, 2025). En Zapier, estos flujos se denominan "Zaps", y típicamente conectan dos o más aplicaciones en una secuencia lineal de pasos. La plataforma guía al usuario paso a paso en la configuración de cada Zap, ofreciendo menús desplegables y campos predefinidos para seleccionar *triggers* (disparadores) y acciones, lo que elimina prácticamente cualquier complejidad técnica.

Otra gran ventaja de Zapier es su enorme ecosistema de integraciones disponibles: actualmente ofrece conectores para casi cualquier aplicación popular que uno pueda imaginar. Su directorio anuncia más de 8.000 aplicaciones compatibles listas para "enchufar" en flujos de trabajo (Román, 2025), la cifra más alta entre las tres plataformas comparadas. Esto significa que es muy probable que Zapier ya incluya las herramientas específicas (incluso de nicho) que una empresa utiliza, desde CRM hasta hojas de cálculo, redes sociales o servicios especializados. La amplia cobertura de integraciones hace de Zapier una opción muy versátil cuando se requieren automatizaciones entre aplicaciones diversas, sin tener que recurrir a programación.

Sin embargo, la otra cara de la sencillez de Zapier son sus limitaciones en escenarios más complejos. Los flujos en Zapier tienden a ser secuenciales y menos adaptables a lógicas ramificadas o transformaciones de datos elaboradas. Aunque Zapier ha incorporado funciones como filtros y rutas condicionales, su capacidad de implementar lógica compleja es menor comparada con Make o n8n, que ofrecen un control más granular. Adicionalmente, Zapier opera bajo un modelo de precios basado en tareas: cada "acción" ejecutada (por ejemplo, enviar un correo, añadir un registro) consume una tarea de la cuota mensual. Si una empresa automatiza muchas acciones o tiene flujos con gran volumen, los costos pueden escalar rápidamente, ya que los planes de Zapier con mayores límites de tareas se vuelven significativamente más costosos (Paseur, 2024). Este esquema puede encarecer la plataforma "a lo grande", siendo Zapier muy asequible para prototipos pequeños, pero menos económico para automatizaciones a escala masiva.

Figura 3: Zapier

Fuente: [imagen sin título sobre Zapier], (s. f.). https://bit.ly/4r9cR3s

Make se distingue por ofrecer un equilibrio logrado entre potencia técnica y usabilidad visual. Su propuesta se dirige a usuarios y equipos que necesitan automatizaciones más complejas, con múltiples pasos, transformaciones de datos y condiciones, pero desean lograrlas a través de una interfaz gráfica intuitiva en lugar de escribir código. En Make, las automatizaciones se denominan "escenarios" y se construyen en un lienzo visual (canvas) conectando bloques llamados módulos (cada módulo realiza una operación). En Make uno puede ramificar el flujo, agregar módulos intermedios de lógica, repetir operaciones, etc., con mucha libertad, similar a dibujar un diagrama de flujo de procesos.

La interfaz visual de Make es uno de sus puntos fuertes: permite arrastrar y soltar módulos en el canvas y conectarlos para diseñar el flujo de datos (Román, 2025). Cada módulo representa una acción o consulta de alguna aplicación (p. ej., "buscar filas en Google Sheets" o "enviar mensaje por Slack") o bien una operación lógica (como un filtro, un *router* condicional, un bucle, etc.). En términos generales, Make es muy potente en manipulación de datos y lógica granular (aspectos en los que Zapier es más limitado), lo que le permite abordar procesos complejos de forma elegante (Román, 2025). Esta versatilidad viene con una curva de aprendizaje algo mayor: los usuarios novatos pueden encontrar la interfaz de Make un poco abrumadora al inicio, dado que expone más opciones y requiere entender conceptos como el flujo de datos entre módulos, el manejo de variables (*bundles*), etc. (ActivDev, 2025). Sin embargo, una vez superada la fase inicial y adquirida cierta experiencia, Make ofrece un control sin precedentes y una flexibilidad difícil de igualar en el ámbito *no-code* (ActivDev, 2025).

En cuanto a integraciones, Make soporta de fábrica una amplísima lista de aplicaciones, en el orden de +2.000 *apps* conectables directamente mediante módulos predefinidos (Paseur, 2024). Si bien su catálogo es algo menor que el de Zapier, cubre la mayoría de herramientas populares en negocios (Google Workspace, Microsoft 365, Salesforce, HubSpot, Slack, etc.) y muchas otras.

Otro aspecto notable es el modelo de precios de Make, el cual suele resultar más rentable para cargas de trabajo medias/altas. Make cobra con base en operaciones ejecutadas (cada ejecución de un módulo cuenta como una operación) en lugar de por tareas discretas como Zapier (ActivDev, 2025). En la práctica, esto significa que Make suele ser más asequible que Zapier para automatizaciones de complejidad o volumen medio, ya que brinda más "acciones" posibles por el mismo precio.

Figura 4. Make

Fuente: [imagen sin título sobre make], (s. f.). https://bit.ly/4qoKSLY

n8n: potencia de código abierto y control *self-host* para usuarios técnicos

Figura 5. N8n

Fuente: [imagen sin título sobre n8n], (s. f.). https://bit.ly/4a1oPWD

n8n destaca por su filosofía de código abierto ("*fair-code*"). A diferencia de Zapier y Make, que son servicios en la nube cerrados, n8n permite descargar su *software* y ejecutarlo en servidores propios de la empresa (*self-hosting*), además de ofrecer una instancia en la nube pública. Esto le da a los usuarios un control total sobre los datos y la infraestructura de sus automatizaciones, una ventaja importante para organizaciones con altos requerimientos de seguridad, privacidad o personalización, que desean mantener todo "en casa" (Román, 2025). El enfoque *open source* también significa que la comunidad de desarrolladores puede extender la plataforma creando nuevos conectores (nodos) o funciones y compartiéndolos, lo cual ha nutrido un ecosistema muy activo en torno a n8n.

En términos funcionales, n8n es similar a Make en que ofrece un editor visual de flujo con nodos conectables para construir automatizaciones ramificadas. Sin embargo, su orientación es claramente más "*developer-friendly*" o dirigida a perfiles técnicos. La interfaz de n8n, si bien moderna y limpia, asume que el usuario entiende conceptos de programación y API; por ejemplo, es común tener que escribir pequeñas expresiones de JavaScript para ciertas transformaciones, o gestionar manualmente autenticaciones complejas. Desde el inicio, la herramienta expone mucha flexibilidad, pero espera que "sepas lo que estás haciendo" (Replogle, 2025). Esto puede hacer que la curva de aprendizaje sea empinada para usuarios no técnicos, quienes podrían sentirse perdidos ante tantas posibilidades sin guía paso a paso. En contrapartida, para un desarrollador o ingeniero, n8n brinda una plataforma altamente personalizable y extensible: puedes insertar fragmentos de código en nodos especiales, crear integraciones propias y, en general, moldear la automatización casi sin límites (Román, 2025). Queda claro que, así como su uso es más complejo, su capacidad y escalabilidad es mucho mayor, permitiendo desarrollar automatizaciones más avanzadas.

El número de integraciones disponibles en n8n ha crecido rápidamente gracias a su comunidad. Actualmente, ofrece más de 1.100 nodos integrados que cubren numerosas

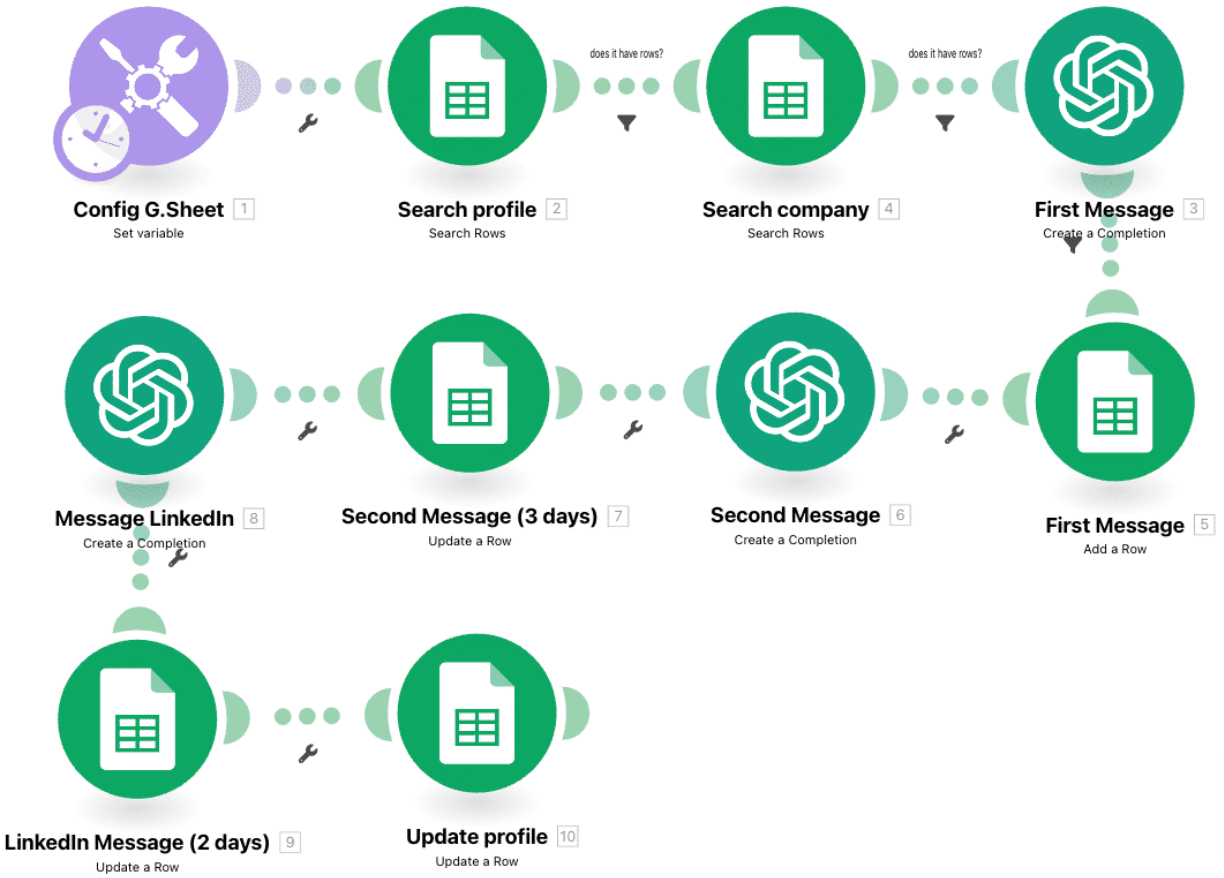
aplicaciones y servicios (muchos de ellos aportados por la comunidad) (Román, 2025). Esta cifra es menor que las miles de Zapier o Make, pero es engañosa: al ser *open source*, n8n permite que tú mismo desarrolles un nuevo conector si lo necesitas y no existe, o que uses nodos genéricos para llamar API externas. En cierto sentido, n8n prioriza la flexibilidad técnica sobre la cantidad de integraciones prediseñadas, aunque su biblioteca por defecto ya es muy amplia y cubre usos comunes.

En cuanto a despliegue y costos, n8n ofrece un modelo singular: la versión autohospedada es gratuita (bajo licencia *fair-code*, gratuita para uso propio), lo que significa que una empresa puede instalar n8n en sus servidores y ejecutar tantos flujos como quiera sin coste de licenciamiento. Obviamente, esto implica asumir el mantenimiento del servidor, actualizaciones, etc., por cuenta propia. Para quienes prefieren no ocuparse de la infraestructura, n8n dispone de un servicio *cloud* gestionado con planes de pago. A diferencia de Zapier/Make, n8n cobra según ejecuciones completas de flujo (cada vez que un flujo se dispara, sin importar cuántos pasos tenga) en lugar de por acción/paso individual (Román, 2025). Este modelo de precio por ejecución puede ser muy conveniente cuando los flujos son largos con muchos pasos, ya que el costo no crece con cada acción adicional.

Integración de agentes de IA en flujos automatizados *no-code*

Como ya habrán imaginado, todas estas herramientas *no-code* no son por sí mismas herramientas de inteligencia artificial, sino que uno las configura u orquesta para que utilicen IA cuando lo necesite el flujo. Es por eso que un aspecto especialmente interesante es cómo las plataformas anteriores permiten incorporar capacidades de inteligencia artificial dentro de los flujos de automatización. Así como en los flujos podemos conectar distintas aplicaciones, algunas de las mismas pueden ser los modelos de IA que se usan generalmente (GPT, Claude, Gemini, etc.). De esta forma, podemos integrar IA en los flujos para que tome decisiones y ejecute acciones en otras aplicaciones, dándole la capacidad a estos modelos para convertirse en agentes de IA.

Figura 6. Integración de agentes de IA en flujos automatizados



Fuente: Make Community, 2023, https://bit.ly/46CuMqL

En pocas palabras, esto convierte una automatización tradicional (que sigue reglas predefinidas) en una automatización "inteligente" capaz de manejar entradas no estructuradas, tomar decisiones complejas o generar contenido nuevo. Algunas capacidades típicas que se logran al sumar agentes de IA en un flujo *no-code* son:

- **Análisis avanzado de datos no estructurados:** un agente de IA puede

examinar textos, imágenes o audio dentro del flujo. Por ejemplo, analizar comentarios de clientes para determinar si son positivos o negativos, extraer entidades clave de un correo electrónico, o identificar objetos en una imagen adjunta. Cosas que serían de mucha dificultad si no se utilizase IA. Luego el flujo puede bifurcarse o actuar según ese análisis (p. ej., escalar un caso si el sentimiento es negativo, o derivar una imagen a cierto catálogo según lo que contenga). Esto permite automatizar tareas de clasificación o filtrado que antes requerían intervención humana por su complejidad semántica.

- **Generación automática de contenidos o respuestas:** usando modelos generativos de IA, un flujo puede crear contenido nuevo sobre la marcha. Por ejemplo, ante una consulta compleja de un cliente, un agente de IA integrado podría redactar una respuesta apropiada y el sistema la envía directamente, simulando la interacción de un agente humano (Zapier, s. f.). O bien, generar resúmenes de informes extensos para enviarlos por email, componer descripciones de productos a partir de especificaciones, traducir texto, etc.

- **Interacciones conversacionales automatizadas:** integrando agentes de IA conversacionales, se logran flujos donde la automatización puede dialogar con usuarios o clientes. Por ejemplo, un *chatbot* con IA puede conectarse vía una herramienta *no-code* al CRM y a una base de conocimiento: cuando un cliente hace una pregunta en el chat, el flujo pasa la pregunta al agente de IA, este consulta la base de conocimiento para formular la respuesta, y la devuelve al cliente (Román, 2025). Todo el proceso es orquestado por la plataforma *no-code* (que maneja mensajes entrantes, llama al modelo y entrega la salida). Estos *chatbots* integrados pueden atender soporte nivel 1 de forma escalable, calificar leads conversacionalmente, realizar encuestas interactivas, etc., sin intervención humana directa.

Para finalizar, concluimos que las plataformas *no-code* actuales están preparadas para hibridar automatización con inteligencia artificial, lo que abre un abanico enorme de posibilidades. Integrar agentes de IA en los flujos permite manejar información de manera más inteligente y autónoma, ejecutando acciones condicionadas por análisis avanzados que antes requerían el juicio humano. Esto lleva la

automatización a un nuevo nivel (automatización cognitiva), donde los procesos no solo siguen reglas estáticas, sino que también “entienden” y “deciden” en contextos complejos. Para los estudiantes y profesionales que nos leen, dominar estas integraciones significa estar a la vanguardia en la creación de soluciones empresariales, combinando lo mejor de dos mundos: la eficiencia de la automatización y la inteligencia adaptable de la IA.

CONTINUAR

3. Introducción detallada a Make

En esta sección final nos enfocaremos en la plataforma Make (que será la que enseñaremos en este curso debido a las ventajas expuestas anteriormente), brindando una guía detallada de sus conceptos fundamentales, su funcionamiento práctico y los elementos clave de su interfaz. El objetivo es dotar al lector de un entendimiento sólido de cómo usar Make para construir automatizaciones profesionales, apoyándonos en ejemplos y referencias técnicas oficiales de la herramienta.

Make es una plataforma en línea de automatización visual que permite conectar aplicaciones y servicios sin programar, mediante la construcción de flujos de trabajo denominados escenarios (Consultevo, s. f.). En lugar de escribir código, el usuario diseña diagramas de flujo en un editor gráfico, encadenando módulos que representan acciones o intercambios de datos entre aplicaciones. Make se encarga

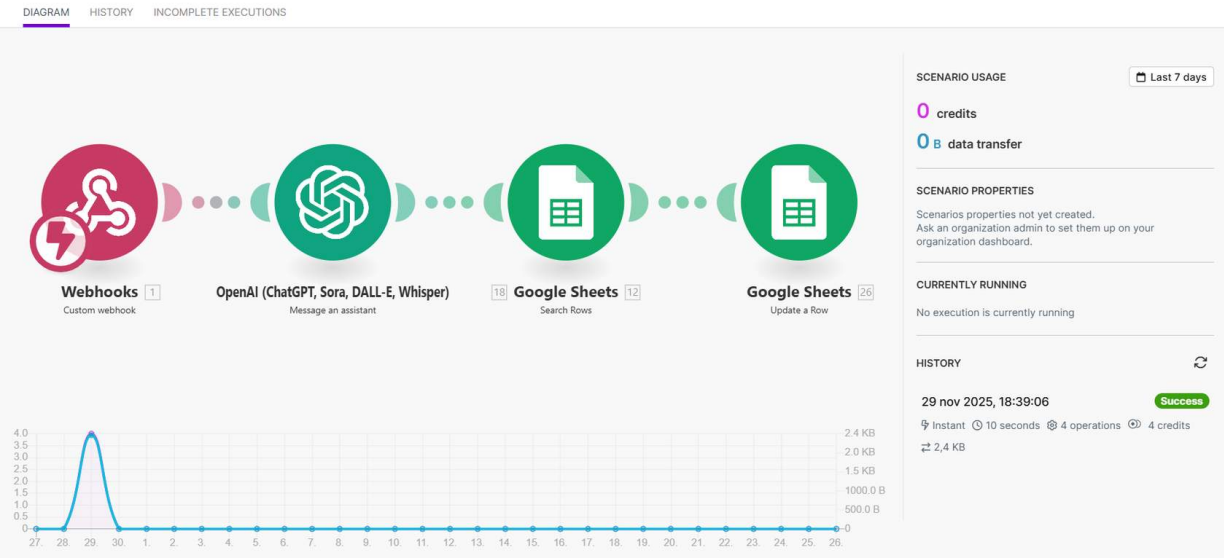
de ejecutar la lógica definida, ya sea en intervalos programados, en tiempo real ante ciertos eventos, o bajo demanda, moviendo datos de una etapa a otra y automatizando tareas repetitivas de forma confiable (Consultevo, s. f.).

Conceptos fundamentales: escenario, módulo, operación y API

Un escenario en Make es la unidad básica de automatización: esencialmente, el flujo de trabajo completo que el usuario diseña en el lienzo visual (Make, 2025). Un escenario define qué aplicaciones se conectan, cómo fluye la información entre ellas, bajo qué condiciones y con qué secuencia. Se puede pensar en un escenario como la descripción formal de un proceso a automatizar: incluye las acciones a ejecutar, el orden, las condiciones lógicas y la forma en que inicia. Por ejemplo, un escenario podría representar “Cuando se agregue una fila nueva a la hoja de cálculo X, tomar esos datos y enviar una notificación en Slack, luego crear un registro en el CRM Y”. En Make, cada cuenta de usuario puede crear múltiples escenarios para automatizar distintos procesos, y existen opciones para organizarlos, documentarlos e incluso plantillas para escenarios comunes. Como límite general, Make permite

hasta cierto tamaño en cada escenario (ej.: 2 MB de definición) y un usuario puede crear cientos de escenarios en su espacio de trabajo (Make, 2025), lo que en la práctica no suele ser restrictivo.

Figura 7: Escenarios en Make



Fuente: elaboración propia.

Formalizando la definición, un escenario es una serie de módulos conectados que indican cómo deben transferirse y transformarse datos entre aplicaciones/servicios dentro de Make (2025). Es decir,

especifica origen de los datos, pasos intermedios de procesamiento y destino final, en un flujo autónomo.

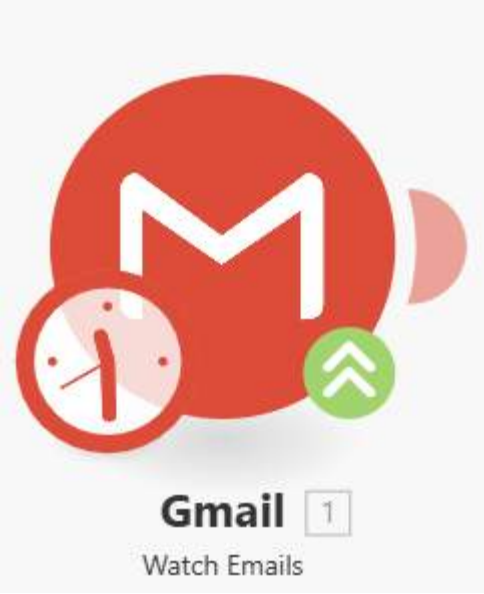
Módulos

Los módulos son los bloques individuales o pasos dentro de un escenario. Cada módulo realiza una operación específica: puede ser una acción en una aplicación (ej.: “Enviar un correo con Gmail”), una búsqueda o consulta (ej.: “Buscar un contacto en Salesforce”), un paso lógico (ej.: “Filtrar resultados que cumplan cierta condición”) o una herramienta auxiliar (ej.: “Esperar 1 hora”, “Dividir texto”, etc.). Los módulos son los componentes que el usuario arrastra al canvas y configura uno por uno para construir el escenario (Consultevo, s. f.). En Make, los módulos están asociados a aplicaciones o funcionalidades: por ejemplo, existe un conjunto de módulos de Google Sheets (añadir fila, buscar filas, actualizar celda, etc.), un conjunto de módulos de Slack (enviar mensaje, buscar canal, etc.), módulos de control de flujo (routers, iteradores) y así sucesivamente.

Podemos clasificar los módulos según su rol en el escenario (Consultevo, s. f.):

- Algunos módulos actúan como disparadores (*triggers*), generalmente colocados al inicio del escenario, encargados de “esperar” un evento o nueva información. Por ejemplo, el módulo *Watch new rows* de Google Sheets se queda vigilando una hoja en busca de nuevas filas; cuando detecta alguna, inicia la ejecución del escenario aportando esos datos. Estos módulos suelen llevar un símbolo de reloj o rayo.

Figura 8: Gmail Watch Emails



Fuente: Make Community, 2025, https://bit.ly/4krH1fO

- La mayoría de los módulos son acciones en aplicaciones: realizan una tarea concreta como crear, enviar, actualizar o borrar algo en un servicio. Ejemplos: *Create a Trello Card, Send an Email via Gmail, Update a record in Airtable*. Son los pasos que efectúan cambios o desencadenan efectos externos.

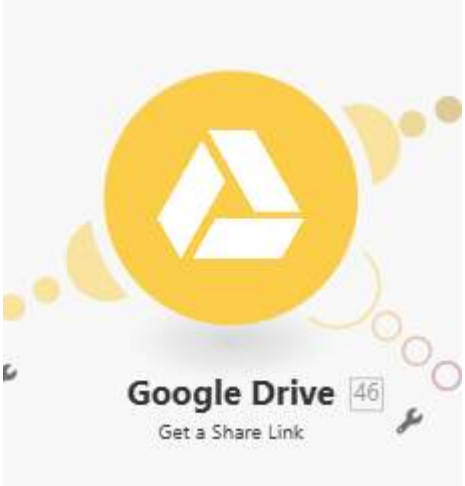
Figura 9: Open IA

Fuente: Make Community, 2025, https://bit.ly/4krH1fO

- Otros módulos son búsquedas o consultas: recuperan información de algún lugar para usarla en el flujo. Por ejemplo, *Search for a contact by email en HubSpot, Get a file de*

Google Drive, etc. Su función es traer datos adicionales que el escenario necesita para tomar decisiones o compilar resultados.

Figura 10: Google Drive



Fuente: Make Community, 2025, https://bit.ly/4krH1fO

- También hay módulos de transformación y lógica interna: son propios de Make para manipular datos o controlar el flujo. Aquí entran los transformadores (como *Text aggregator* para juntar varios textos, *Iterator* para descomponer una lista en elementos individuales), los *routers* (que dividen el flujo en múltiples caminos paralelos o condicionales) y módulos utilitarios como

Sleep (pausa), *Math* (cálculos), etc. Estos no interactúan con aplicaciones externas sino que operan sobre los datos dentro del escenario, dando flexibilidad lógica.

Figura 11. *Router*

Fuente: Make Community, 2025, https://bit.ly/4krH1fO

- Por último, los *webhooks* merecen mención: Make permite crear módulos de tipo *webhook* que sirven como puntos de entrada/salida HTTP. Un *webhook* de entrada puede iniciar un escenario cuando un sistema externo hace una petición HTTP a una URL proporcionada por Make (ej.: un servicio externo llama a ese *webhook* enviando un JSON, lo que alimenta el escenario). Un *webhook* de salida permite

que el escenario realice una petición HTTP a un *endpoint* externo. En la práctica, los *webhooks* funcionan como conectores universales: con ellos se puede integrar casi cualquier sistema web aunque no haya módulo específico, o exponer la funcionalidad de un escenario de Make a otros sistemas. Más adelante veremos la utilidad de los mismos.

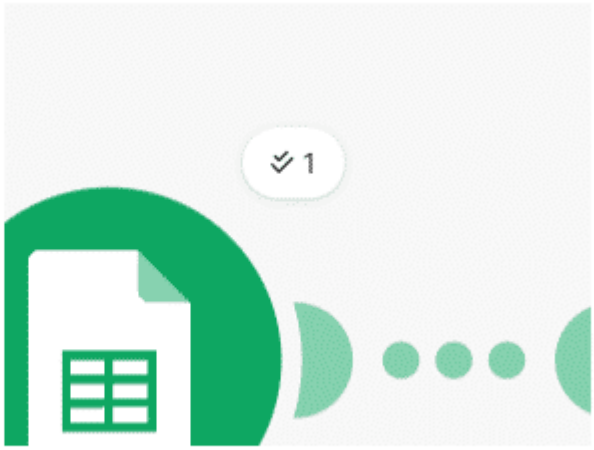
En la interfaz, cada módulo se configura haciendo clic en él y llenando sus campos: por ejemplo, seleccionar la cuenta o conexión a usar (previa autenticación OAuth o con API *key*), mapear qué datos entran (tomados de módulos previos) en los campos requeridos por la acción y, opcionalmente, definir opciones avanzadas o filtros. Una vez configurados, los módulos aparecen como burbujas conectadas por líneas que indican el flujo de datos. Es importante planificar qué módulos serán necesarios y en qué orden; conviene pensar en términos de "¿qué necesito que ocurra paso a paso para lograr el objetivo?", y luego traducir cada paso a un módulo en Make (Consultevo, s. f.).

Operación —

En la terminología de Make, una operación se refiere a la ejecución individual de un módulo para procesar un conjunto de datos (Make, 2025). Cada vez que un módulo corre y maneja un “paquete” de datos (bundle), eso cuenta como una operación. Las operaciones son la unidad de medida del trabajo realizado por un escenario: sirven tanto para efectos de facturación (recordemos que los planes imponen límites mensuales de operaciones) como para análisis del rendimiento. Por ejemplo, si un escenario tiene un módulo “Enviar email” y en una ejecución envía 5 emails (porque quizá venían 5 registros por procesar), eso contabiliza 5 operaciones (una por cada email enviado) (Make, 2025). Algunos módulos especiales de búsqueda cuentan siempre como una sola operación por ejecución (sin importar cuántos ítems devuelvan), esto suele indicarlo la documentación.

Make muestra en tiempo real las operaciones consumidas por cada módulo durante la ejecución: sobre cada módulo aparece un pequeño icono con un número que indica cuántas operaciones y cuántos créditos (otra métrica de Make) se gastaron en él (Make, 2025). Al hacer clic en esas burbujas de operaciones, se pueden inspeccionar los detalles (bundles procesados, datos, etc.), útil para *debugging*.

Figura 12: *Debugging*



Fuente: Make Community, 2025, https://bit.ly/4krH1fO

API —

El término API (*Application Programming Interface*) es muy relevante en Make, ya que la plataforma en el fondo se conecta con aplicaciones externas a través de sus API. Por definición, una API es un conjunto de reglas o protocolos que permiten que dos aplicaciones se comuniquen e intercambien datos y funciones (Goodwin, s. f.). En nuestro contexto, las integraciones de Make con servicios como Gmail, Slack, Dropbox, etc., funcionan consumiendo las API públicas que esos servicios ofrecen. Cuando configuramos un módulo en Make (ej. “Crear Evento en Google Calendar”) y lo ejecutamos, Make internamente hace una llamada API al servidor de Google con los datos proporcionados, recibe la respuesta y la presenta como output del módulo. Todo esto de forma transparente al usuario, pero es importante entenderlo porque explica algunas cosas; por ejemplo, si una app externa tiene limitaciones o permisos en su API, afectarán a lo que Make puede hacer.

Make facilita el uso de API sin código al preconfigurar la lógica de llamada en módulos amigables. No obstante, ofrece también

opciones más directas: existe un módulo genérico *HTTP Request* donde el usuario puede especificar manualmente una URL de API, método (GET, POST, etc.), encabezados y cuerpo, para conectar con cualquier servicio web aunque Make no tenga un conector específico. Esto es extremadamente útil para extender las integraciones. Básicamente, con el módulo HTTP (y los webhooks mencionados), Make puede interactuar con cualquier API REST de terceros, lo que permite integrar servicios no soportados de fábrica siempre que ofrezcan API (Goodwin, s. f.).

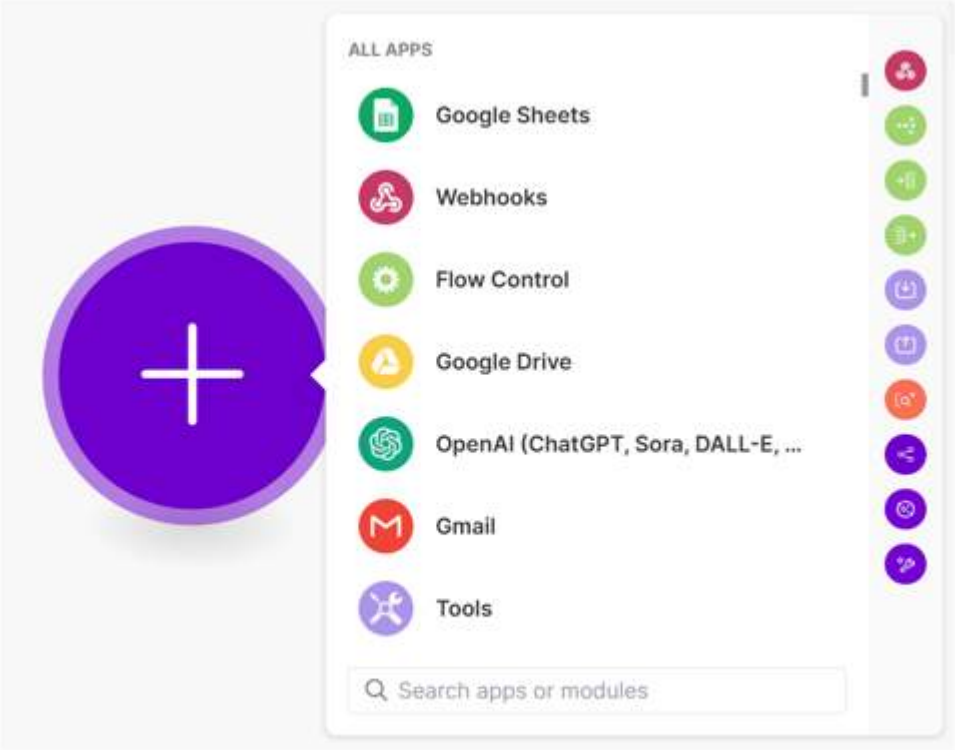
Interfaz y funcionalidades de Make: módulos, plantillas, depuración y configuración

La interfaz de Make, accesible vía navegador web, se compone de varias secciones y menús importantes:

- ***Dashboard* (inicio):** aquí vemos un resumen de nuestros escenarios, el estado (encendido/apagado), últimas ejecuciones y estadísticas básicas (operaciones consumidas, etc.). Desde el *dashboard* se pueden crear escenarios nuevos, organizar en carpetas, y acceder a documentación y comunidad. Es el punto de partida para navegar nuestros proyectos de automatización.
- **Editor de escenarios:** es el entorno visual ya descrito donde diseñamos el flujo. Consta de la barra superior con opciones (nombre de escenario, botón *Run once*, encendido, opciones de escenario, etc.), el canvas central donde aparecen los módulos conectados, y un panel derecho que cambia de contenido según contexto (por ejemplo, al editar un módulo muestra las opciones de configuración; al inspeccionar datos muestra la estructura de *bundles*). También hay un panel izquierdo desplegable con una biblioteca de módulos y aplicaciones disponibles para arrastrar al canvas.

- **Menú de *apps* y módulos:** Make ofrece un buscador de aplicaciones integradas. Al agregar un módulo nuevo en un escenario, aparece una ventana modal con categorías y un buscador para encontrar la *app* o funcionalidad que necesitamos. Están ordenadas alfabéticamente y por tipo. Dado el extenso número de integraciones, esta búsqueda es fundamental para ubicar rápidamente lo que buscamos.

Figura 13: Menú de *apps* y módulos

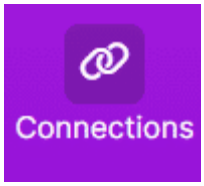


Fuente: elaboración propia.

- **Pantalla de conexiones:** cuando manejamos autenticaciones, Make tiene una sección en el menú izquierdo donde gestionamos las

conexiones a cuentas de las distintas aplicaciones. Allí podemos agregar nuevas credenciales, ver las existentes, revocarlas, etc. Cada conexión guarda tokens OAuth/API keys de acceso a nuestras herramientas.

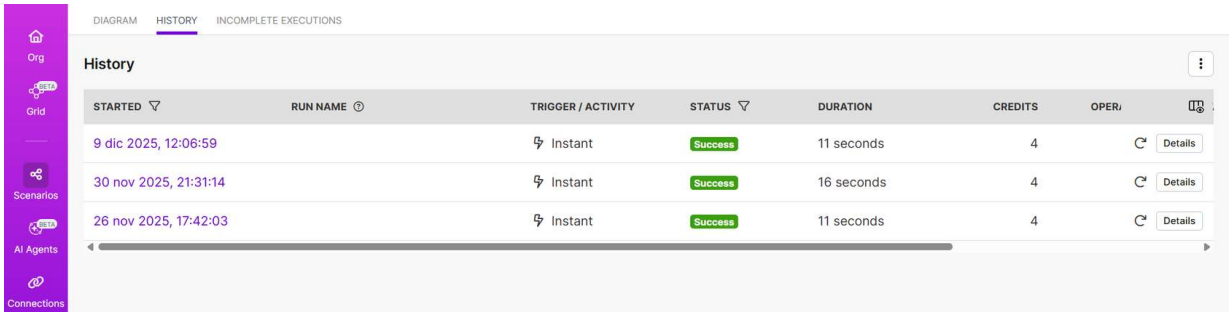
Figura 14: Connections



Fuente: Make Community, 2025, https://bit.ly/4krH1fO

- **Historial de ejecuciones:** accesible desde la vista de escenario, muestra las ejecuciones pasadas con detalles.

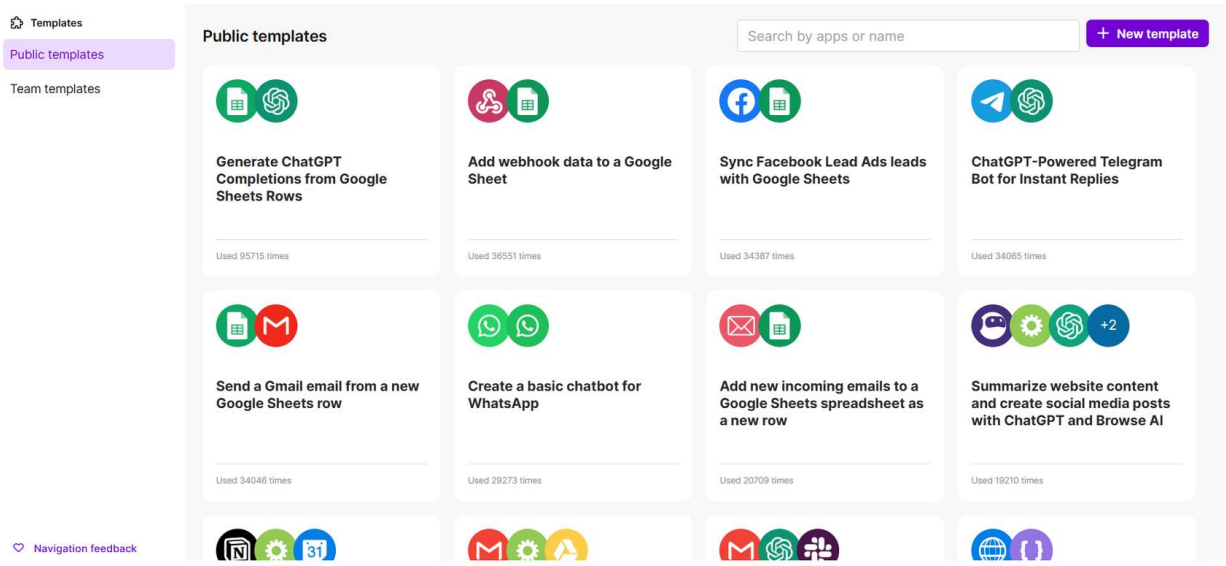
Figura 15. Historial de ejecuciones



Fuente: elaboración propia.

- **Sección de plantillas (*templates*):** Make proporciona una biblioteca de plantillas predefinidas de escenarios para casos de uso comunes (God of prompt, 2025). Al acceder a "Templates" (desde el dashboard o la web de Make), podemos buscar por palabra clave y encontraremos flujos ya armados que podemos importar con un clic (God of prompt, 2025). Por ejemplo: "Enviar respuestas de Google Forms a Google Sheets" o "Notificaciones de Gmail a Slack". Son un recurso didáctico excelente para aprender cómo construir flujos y una forma rápida de desplegar automatizaciones sin partir de cero. Además, son personalizables: al importar una plantilla, esta se puede modificar libremente para adaptarla a las necesidades específicas (God of prompt, 2025). Make mantiene esta librería actualizada con aportes de su equipo y de la comunidad; aprovecharla puede ahorrarnos tiempo e inspirar soluciones.

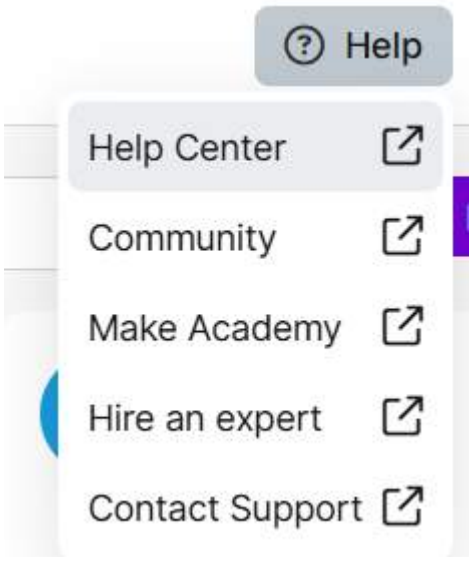
Figura 16: Sección de plantillas



Fuente: elaboración propia.

- **Make Academy y documentación:** Make ofrece una academia en línea y un centro de ayuda con documentación técnica muy completa. Desde la propia plataforma se enlaza a la Make Academy, donde hay cursos interactivos gratuitos y guías de mejores prácticas. Y en el Help Center se encuentra referencia detallada de cada módulo, concepto y función de Make. Estas fuentes oficiales son cruciales para profundizar en aspectos específicos a medida que se vuelve usuario avanzado de la plataforma.

Figura 17: Make Academy y documentación



Fuente: Make Community, 2025, https://bit.ly/4krH1fO

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

[Imagen sin título sobre agentes de IA], (s. f.). https://www.channelpartner.es/negocios/especiales-negocios/que-son-los-agentes-inteligentes-de-ia-o-agentes-de-ia/

[Imagen sin título sobre make], (s. f.). https://1000marcas.net/make-com-logo/

[Imagen sin título sobre n8n], (s. f.). https://en.wikipedia.org/wiki/N8n#/media/File:N8n-logo-new.svg

[Imagen sin título sobre Zapier], (s. f.). https://es.wikipedia.org/wiki/Zapier#/media/Archivo:Zapier-logo.png

ActivDev. (2025, March 16). No-code tools: Make.com, n8n or Zapier for your business? https://www.activdev.com/en/no-

code-tools-makecom-zapier-n8n-comparison/

Amazon Web Services. (s. f.). ¿Qué son los agentes de inteligencia artificial?: explicación de los agentes de inteligencia artificial. Recuperado el 26 de diciembre de 2025, de https://aws.amazon.com/es/what-is/ai-agents/

Código Media. (s. f.). No code: qué es y por qué es tan relevante para el mundo empresarial. Recuperado el 26 de diciembre de 2025, de https://codigomedia.com/no-code-que-es/

Consultevo. (s. f.). Getting Started With Make.com. Recuperado el 26 de diciembre de 2025, de https://consultevo.com/getting-started-with-make-com-3/

God of Prompt. (2025, May 13). Business Process Automation with Make.com: A Step-by-Step Guide. https://www.godofprompt.ai/blog/what-is-make-com

Goodwin, M. (s. f.). ¿Qué es una API (interfaz de programación de aplicaciones)? IBM. Recuperado el 26 de diciembre de 2025, de https://www.ibm.com/es-es/think/topics/api

Google Cloud. (2025, 4 de diciembre). ¿Qué son los agentes de IA? Definición, ejemplos y tipos. https://cloud.google.com/discover/what-are-ai-agents?hl=es

Google Cloud. (s. f.). ¿Qué es el no-code? Guía sobre el desarrollo sin código. Recuperado el 26 de diciembre de 2025, de https://cloud.google.com/discover/what-is-no-code?hl=es

Make Community (2023). Automated, and intelligent prospecting with Make and GPT-4. https://community.make.com/t/automated-and-intelligent-prospecting-with-make-and-gpt-4/9697

Make Community (2025). Instant, Automatic Gmail Trigger? https://community.make.com/t/instant-automatic-gmail-trigger/86904

Make. (2025, 27 de agosto). Operations. Make Help Center. https://help.make.com/operations

Make. (2025, 3 de octubre). Scenario settings. Make Help Center. https://help.make.com/scenario-settings

Make. (2025, 4 de noviembre). Create your first scenario. Make Help Center. https://help.make.com/create-your-first-

scenario

Manzanera, J., Abarca, A., y Iñigo, G. (2025, 3 de enero). Agentes inteligentes: Qué son y cuál será su rol en 2025. WIRED. https://es.wired.com/articulos/2025-el-ano-de-los-agentes-inteligentes

Parseur. (2024, October 4). Zapier vs Make vs n8n - Which Automation Tool Is Best? https://parseur.com/blog/zapier-n8n-make

Replogle, N. (2025). n8n vs. Make: Which is best for your organization? [2026]. Zapier. https://zapier.com/blog/n8n-vs-make/

Román, C. (2025, 14 de mayo). Make vs n8n vs Zapier (2025): guía práctica para elegir bien. Nuclio Digital School. https://nuclio.school/blog/make-vs-n8n-vs-zapier-guia-practica-para-elegir-bien/

Tailored Edge Marketing. (s. f.). Learn To Set Up Your First Make.com Scenario: A Beginner's Guide. Recuperado el 26 de diciembre de 2025, de https://tailorededgemarketing.com/a-beginners-guide-to-setting-up-your-first-make-com-scenario/

Zapier. (s. f.). Zapier: Automate AI Workflows, Agents, and Apps. Recuperado el 26 de diciembre de 2025, de https://zapier.com/

CONTINUAR