

API keys y credenciales en n8n: por qué nunca debes pegar tu key directo en el nodo

Casi todo el que empieza en n8n comete el mismo error una vez: abre un nodo, ve el campo de autenticación y pega su API key ahí mismo, en texto plano. Funciona. El workflow corre. El problema aparece después, cuando exportas ese workflow para un cliente, lo subes a un repositorio o simplemente le tomas una captura de pantalla a tu pantalla para pedir ayuda. Esta guía explica qué es en realidad una API key, qué es una credencial en n8n, por qué son cosas distintas, y cómo dejar tus workflows listos para exportar sin regalar tus accesos.

¿Qué es una API key y qué es una credencial?

Una **API key** es una contraseña que identifica a tu cuenta frente a un servicio externo: OpenAI, Google, Slack, tu CRM. Con ella, ese servicio sabe que eres tú quien está pidiendo algo (enviar un correo, generar texto, leer una hoja de cálculo) y te deja pasar. Si esa key cae en las manos equivocadas, quien la tenga puede usar el servicio a tu nombre y a tu costo.

Una **credencial** en n8n es distinta a la key en sí. Es un objeto separado que n8n guarda en su propio almacén, cifrado, fuera del workflow. Tú creas la credencial una sola vez, le pegas la key ahí, y desde ese momento cada nodo que necesite ese servicio simplemente selecciona la credencial por su nombre. El nodo nunca contiene la key, solo una referencia a dónde vive.

LA DIFERENCIA CLAVE

La API key es el secreto. La credencial es la caja fuerte donde vive ese secreto. Cuando exportas un workflow, n8n exporta el workflow y el nombre de la credencial que usa, pero no el contenido de la caja fuerte. Eso es intencional.

Por qué no debes poner tu API key directamente en el nodo

Algunos nodos (como el nodo HTTP Request) te permiten meter la key a mano en un header o en el body, sin pasar por el sistema de credenciales. Y hay un caso donde no te queda de otra: el nodo **Code (JavaScript)** no tiene un campo de credenciales como los demás nodos, así que si tu código necesita llamar a una API externa, la tentación es escribir la key directo ahí, en texto plano, dentro del script. En cualquiera de los dos casos, se acumulan tres problemas reales:

- 1 **Se exporta con el workflow.** Si metes la key directo en un parámetro del nodo, esa key queda escrita en el JSON del workflow. El día que exportes ese workflow, lo subas a Git, o se lo mandes a alguien, tu key viaja con él, visible para cualquiera que abra el archivo.
- 2 **No la puedes reutilizar.** Si usas el mismo servicio en cinco nodos distintos, tendrías que pegar la misma key cinco veces. El día que la rotes (algo que deberías hacer de forma periódica), tienes que ir nodo por nodo buscándola.
- 3 **No hay control de acceso.** Las credenciales en n8n se pueden compartir con miembros específicos del equipo o proyectos concretos, sin que esas personas vean el valor real de la key. Si la pegas directo en el nodo, cualquiera con acceso de edición al workflow ve el secreto completo.

¿Para qué vas a exportar tu workflow?

Exportar un workflow en n8n genera un archivo JSON con toda la lógica: nodos, conexiones, parámetros, y las referencias a las credenciales usadas (nunca su valor). Vale la pena tener claro para qué lo estás exportando, porque cambia lo que necesitas preparar antes:

Para un cliente. Le entregas el workflow para que lo importe en su propia instancia de n8n. Como las credenciales no viajan con el archivo, el cliente va a necesitar crear sus propias credenciales del mismo tipo (incluso si le pusiste el mismo nombre) para que el workflow funcione. Prepárale una nota corta con qué credenciales necesita crear y de qué servicio.

Para enseñar a alguien más. Compartes el workflow como ejemplo o plantilla. Aquí el hecho de que las keys no se exporten es una ventaja: puedes compartir la lógica completa sin exponer ningún acceso propio, y quien lo importe conecta sus propias credenciales para probarlo.

Para tu futuro. Guardas el workflow como respaldo o versión en un repositorio. Si alguna vez tienes que restaurarlo en otra instancia (una migración, un nuevo servidor), vas a tener que volver a crear las credenciales ahí. Vale la pena guardar, en un lugar separado y seguro, una lista de qué credenciales usa cada workflow para no tener que reconstruir esa lógica de memoria.

UN EJEMPLO TÍPICO

Armas un workflow para un cliente en tu propia cuenta de n8n. Cuando el proyecto termina, no le vas a dar acceso a tu instancia: exportas el workflow como JSON y se lo mandas para que lo importe en la suya. Ese es el momento en el que se nota si dejaste todo bien armado con credenciales, porque ninguna de tus keys va a viajar en ese archivo.

Guía: cómo crear una credencial en n8n

1. **Haz clic en el botón desplegable de la esquina superior derecha** y elige **Crear credencial**.
2. **Select an app or service to connect to:** busca y selecciona el servicio al que te quieres conectar (OpenAI, Gmail, Google Sheets, el que sea).
3. **Completa los campos** que te pide ese servicio: tu API key, o los datos de OAuth2 si es ese el método de autenticación. Nunca los pegues en el nodo directamente, solo aquí.
4. Ponle un **nombre claro** a la credencial (ej. "OpenAI - Cuenta principal") para que sea fácil de identificar cuando la reutilices en otros nodos.
5. Guarda. n8n suele probar la conexión automáticamente; si falla, te avisa antes de que el workflow corra en producción.
6. Desde ese momento, cualquier nodo que use ese mismo servicio puede seleccionar esa credencial ya guardada, sin volver a pegar la key.

UN DETALLE QUE SE PASA POR ALTO

Las credenciales viven a nivel de tu instancia (o de un proyecto, si usas ese modelo), no a nivel de un workflow individual. Eso significa que borrar un workflow no borra la credencial, y que una credencial mal nombrada puede terminar usada por error en un workflow equivocado. Nómbralas con cuidado.

Bonus: variables para el nodo Code (y por qué necesitas el plan Pro)

Como vimos arriba, el nodo Code no tiene un selector de credenciales. Si tu script en JavaScript necesita autenticarse contra una API, no puedes simplemente elegir una credencial de la lista, así que la solución no es escribir la key en el código: es crear una **variable** en n8n y llamarla desde ahí.

Las variables son un almacén de valores reutilizables (URLs, IDs, nombres de entorno, y también keys cuando el nodo que las necesita no soporta credenciales) que defines una sola vez desde el panel de variables. Se llaman desde el nodo Code o desde cualquier expresión con la sintaxis:

```
$vars.NOMBRE_DE_TU_VARIABLE
```

Así tu código nunca contiene el valor real, solo la referencia. Si compartes el workflow o el script con alguien, la key sigue sin viajar con él, igual que pasaría con una credencial.

El detalle importante: las **variables personalizadas (custom variables)** son una función de los **planes Pro y Enterprise** de n8n, tanto en la nube como en self-hosted. Si estás en un plan Community o Starter, no vas a tener esa opción disponible en el panel de variables, y vas a necesitar otra vía para no hardcodear la key en el nodo Code (por ejemplo, variables de entorno del servidor si tú mismo hosteas tu instancia).

RESUMEN RÁPIDO

Secreto de autenticación en un nodo normal → credencial. Secreto que necesita el nodo Code (que no soporta credenciales) → variable (si tienes plan Pro/Enterprise). Nunca escrito directo en el nodo, en ninguno de los dos casos.

¿Quieres recibir consejos gratuitos sobre IA y automatización?

Comparto trucos, recursos e ideas para usar la inteligencia artificial en el trabajo y en tu vida diaria. Sígueme para no perderte ninguno.

[operativa_ia](#)