BIG school

{m}

# Apuntes

# Curso de Desarrollo con IA

DÍA 1

# Índice

# ¿Por qué la IA es tu mejor aliada para Programar?

La idea de la Inteligencia Artificial puede ser intimidante, especialmente cuando estás dando tus primeros pasos en el mundo de la programación. Es fácil percibir esta revolución con algo de miedo, pensando que viene a reemplazar el talento humano. Y si sientes eso, es normal.

Sin embargo, el propósito de esta guía es demostrarte exactamente lo contrario: la IA no es tu reemplazo, sino la herramienta más poderosa que ha llegado para potenciar tu ingenio y acelerar tu camino como desarrollador.

# El nuevo panorama: La IA ya está aquí (y eso es bueno)

La Inteligencia Artificial no es una promesa futurista, sino una realidad integrada en el presente del desarrollo de software. Para entender el cambio, es crucial observar el contexto actual del sector y aceptar que esta es una herramienta que llegó para quedarse.

**Las Cifras de la Revolución**

Los siguientes datos, basados en estudios masivos a desarrolladores, pintan un cuadro claro de la situación actual.

**45%** del código ya es escrito por IA:

- **Por qué es importante:** Esto no significa que la mitad del trabajo haya desaparecido. Significa que la colaboración con la IA ya es el estándar de la industria. Aprender a usarla no es una opción para el futuro, es una habilidad esencial para tu primer día de trabajo.

**55%** del proceso de desarrollo se ve acelerado:

- **Por qué es importante:** La IA no compite contigo, compite contra el tedio. Su objetivo es automatizar las partes repetitivas y lentas para que tú puedas dedicar tu energía mental a la arquitectura, la lógica de negocio y la resolución creativa de problemas, que es donde realmente aportas valor.

**90%** de los desarrolladores sienten mayor productividad y satisfacción laboral:

- **Por qué es importante:** Al delegar tareas repetitivas o complejas a la IA, los desarrolladores pueden enfocarse en resolver problemas de mayor nivel y en los aspectos más creativos del trabajo. Esto no solo aumenta la productividad, sino también la satisfacción profesional.

**Una Evolución, no una Extinción**

El miedo al cambio es natural, pero nuestro sector siempre ha estado en constante evolución. La IA es simplemente el siguiente paso en esa progresión.

Así como los editores de código avanzados, los frameworks y los lenguajes de alto nivel fueron una evolución que nos hizo más eficientes, la IA es el siguiente paso lógico en nuestra caja de herramientas.

La IA es la nueva realidad, pero esto no disminuye tu valor; al contrario, lo redefine. La herramienta más potente del mundo es inútil sin un experto que la dirija. Tú sigues siendo el piloto.

# El Programador sigue siendo el piloto: La analogía del Copiloto

La metáfora más acertada para describir la relación entre un desarrollador y la IA es la de un piloto y su copiloto. La IA es un asistente increíblemente capaz, pero necesita ser dirigida por un experto que tenga los conocimientos fundamentales y la visión del proyecto.

### ¿Quién dirige la nave?

Una IA, por sí sola, no sabe qué construir, cuál es el objetivo del negocio o cómo debe ser la arquitectura del sistema. Simplemente genera respuestas basadas en patrones. El verdadero valor del desarrollador reside en sus fundamentos y su capacidad para guiar a la herramienta. Nosotros somos los pilotos.

"*Yo no le puedo decir a la IA hacia dónde tiene que ir si yo no sé hacia dónde tiene que ir.*"

### El Peligro de "Solo Aceptar"

El peligro real, y el error que delata a un aficionado, es lo que llamamos "***vibe coding***". Consiste en lanzar un prompt, copiar el resultado sin entenderlo y cruzar los dedos. Un profesional, en cambio, utiliza la IA guiándola con su conocimiento, audita cada línea de código sugerida y se asegura de que la solución no sólo "**funcione**", sino que sea robusta, eficiente y cumpla con los objetivos del proyecto.

La IA es tu copiloto, no un piloto automático al que le cedes el control total.

Entender eso es el primer paso. Ahora, veamos qué instrumentos te proporciona la IA para componer software de forma más rápida, inteligente y creativa.

# La IA como potenciador: 3 ejemplos de uso para tu día a día

Aquí es donde la teoría se convierte en práctica. La IA no es solo un asistente de código; es un set de superpoderes que transformará tu forma de aprender, crear y resolver problemas.

## Taller #1: Aprendizaje Acelerado con IA

Como desarrolladores, nos enfrentamos a herramientas increíblemente poderosas como **ChatGPT** o **Gemini**, pero estas tienen un problema fundamental: **por defecto, no tienes control sobre sus fuentes de información**.

Estos modelos han sido entrenados con datos masivos de internet, y al generar una respuesta, pueden mezclar información correcta con datos obsoletos, incorrectos o simplemente inventados, un fenómeno conocido como "**alucinación**".

Aquí es donde entra **NotebookLM**, una herramienta gratuita de Google diseñada para darte el control total.

Con NotebookLM, tú decides cuáles son las fuentes de información, acotando el conocimiento para obtener respuestas precisas y confiables.

El objetivo de este taller es que pierdas ese miedo y aprendas, paso a paso, cómo usar NotebookLM para analizar información de distintas fuentes y tomar una decisión tecnológica informada, actuando como un verdadero experto.

A través de un escenario práctico, descubrirás cómo transformar horas de investigación en minutos de análisis inteligente, poniendo la IA a trabajar para ti.

## Caso Práctico: Decisión de Stack Tecnológico (Python vs. JavaScript)

Todo desarrollador, en algún punto de su carrera, se enfrenta a una decisión crucial: ¿qué lenguaje de programación debería aprender o utilizar para un nuevo proyecto? Es una pregunta que requiere analizar pros, contras, curvas de aprendizaje y ecosistemas tecnológicos.

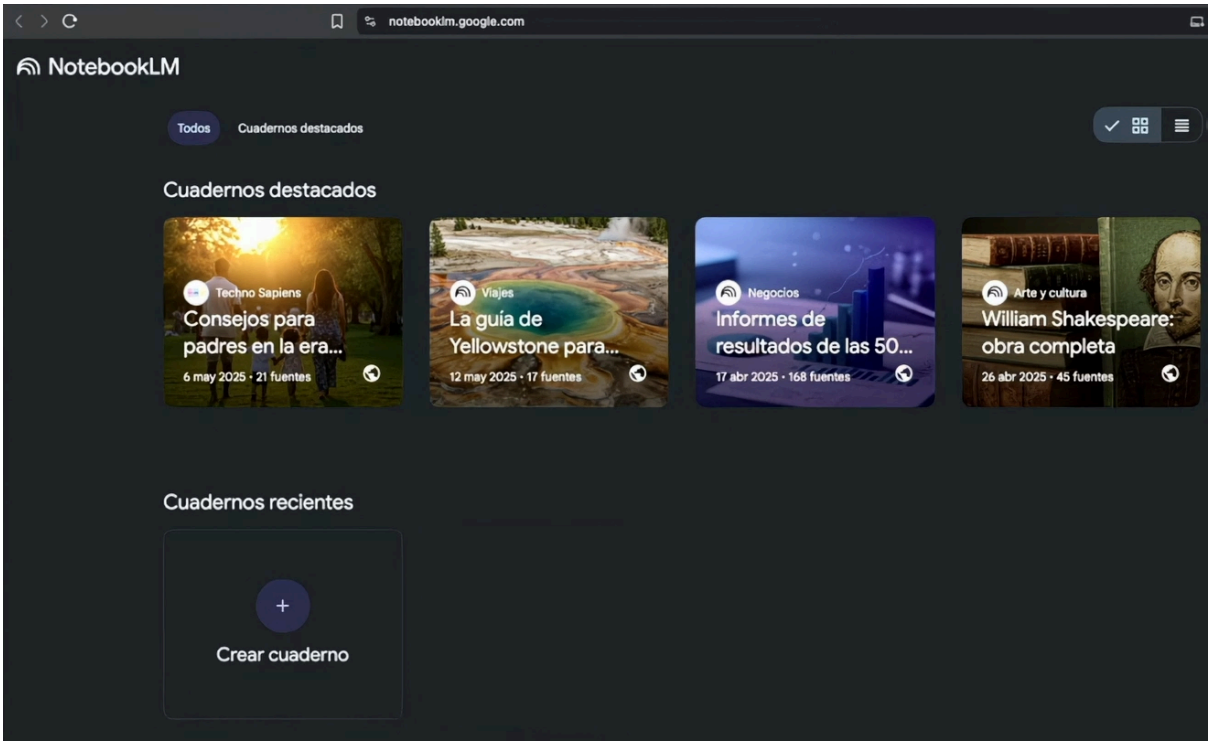
Para este taller, utilizaremos el clásico desafío de comparar Python y JavaScript. Nuestro objetivo no es declarar un ganador, sino utilizar este escenario realista para aprender un proceso de análisis que podrás aplicar a cualquier duda tecnológica que tengas, desde comparar frameworks hasta elegir una base de datos.

## Guía Paso a Paso para Dominar NotebookLM

### Paso 1: Creando tu Cuaderno de Conocimiento

El primer paso es el más sencillo: **crear un nuevo cuaderno en NotebookLM**. Al hacerlo, te encontrarás con un espacio completamente vacío. Este cuaderno es tu "lienzo en blanco", un entorno de IA donde tú, y solo tú, tienes el control total.

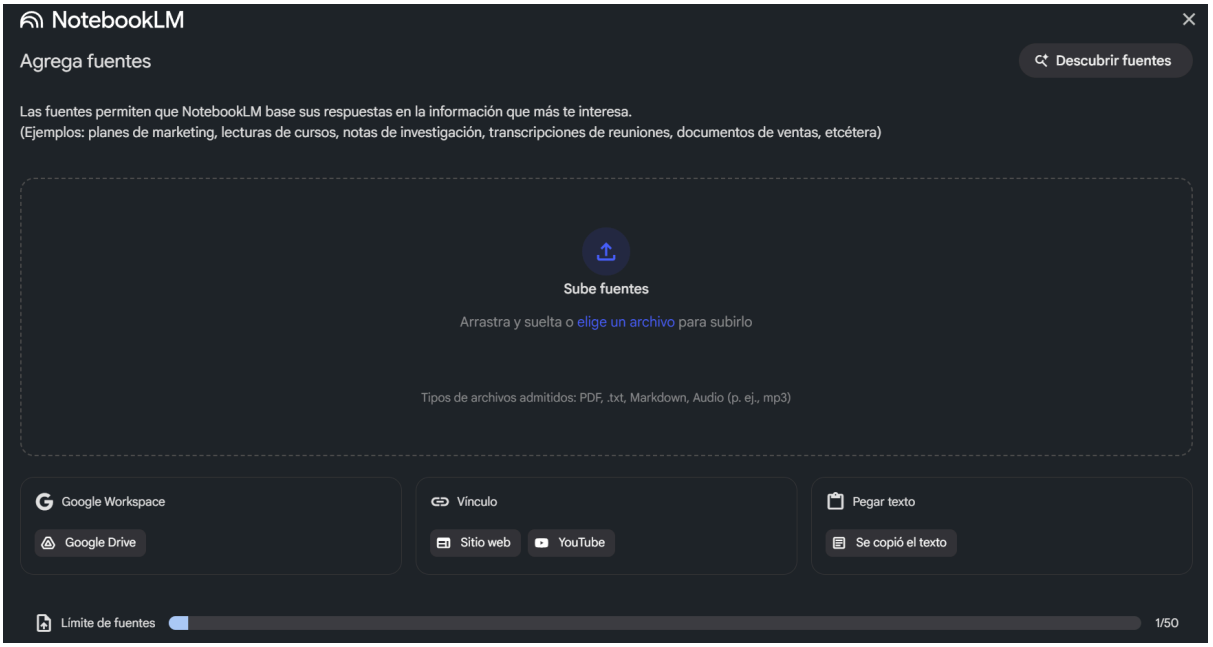
Esto es fundamental. El punto de partida es un conocimiento cero; la IA solo sabrá lo que tú decidas enseñarle.




### Paso 2: Alimentando tu IA con Fuentes Confiables

Ahora es el momento de añadir las "**fuentes**" que conformarán la base de conocimiento de tu IA. NotebookLM es increíblemente versátil y te permite añadir información de diversos formatos.

- Videos de YouTube: Pega un enlace y la IA procesará la transcripción completa.
- Sitios Web: Proporciona una URL para que analice el contenido de una página.
- Documentos de Google Drive: Conecta tu cuenta y añade textos, presentaciones, etc.
- Archivos PDF y de Texto: Sube documentos directamente desde tu ordenador.



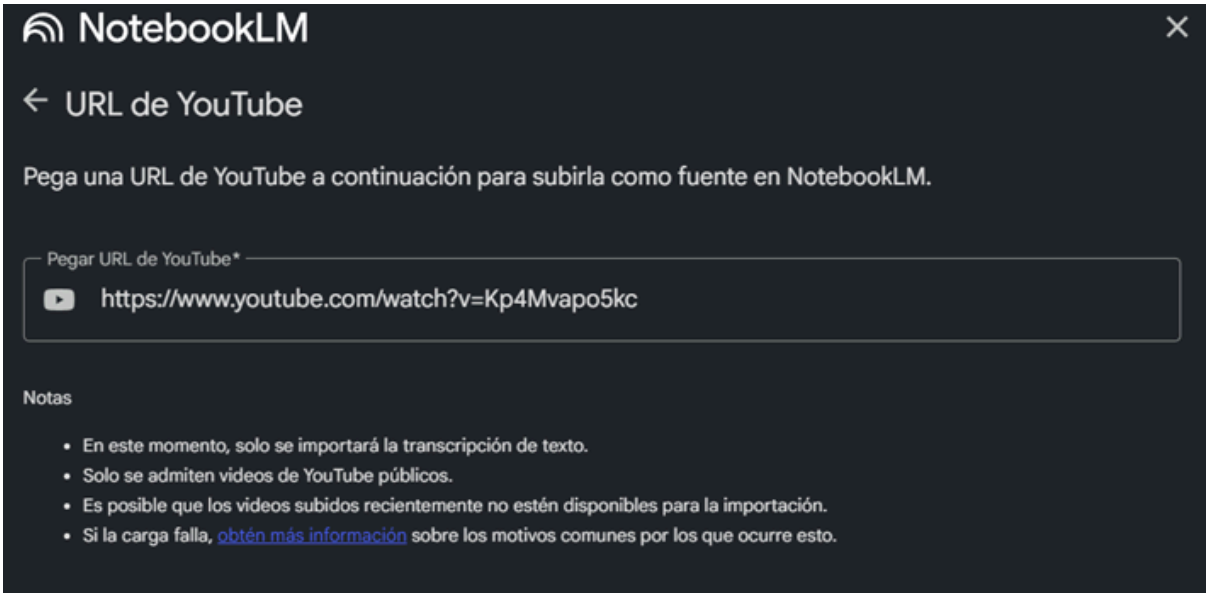

Para nuestro taller, vamos a añadir dos tipos de fuentes clave: cursos en video y documentación oficial.

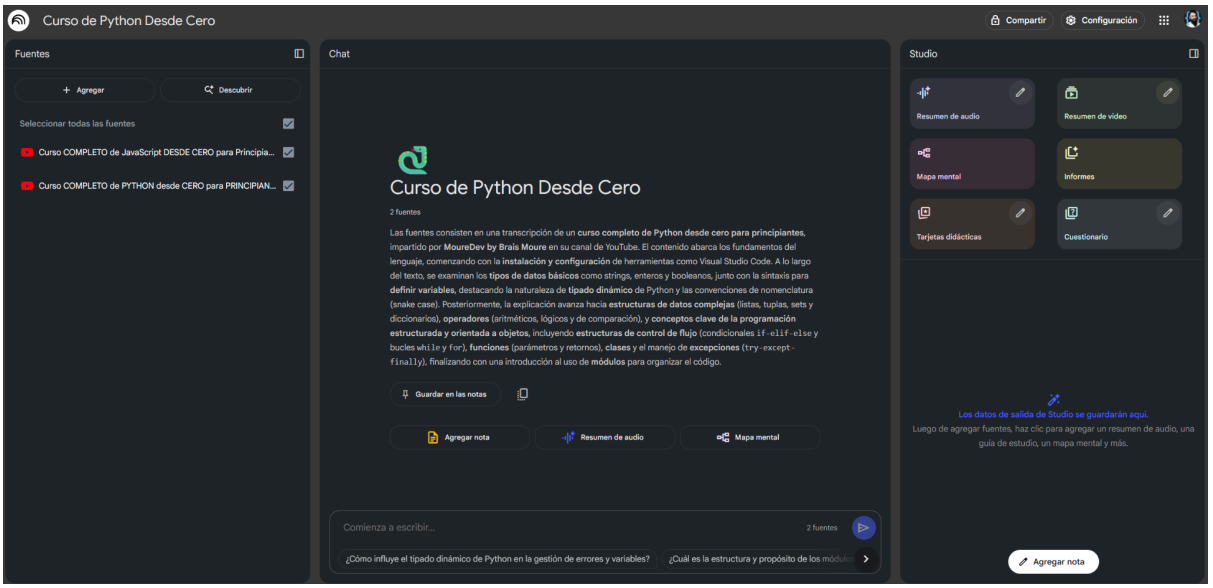
### Añadiendo Cursos en Video (YouTube)

Imagina que quieres basar tu decisión en el contenido de cursos completos. En lugar de ver horas de video, puedes simplemente añadir los enlaces. Para nuestro ejemplo, añadiremos:

- Curso de Python desde cero (10 horas de duración).
- Curso de JavaScript (más de 6 horas de duración).

NotebookLM procesará estos videos en segundos, extrayendo toda la información clave y dejándola lista para ser consultada. ¿Te das cuenta de las horas de trabajo que acabas de ahorrarte?

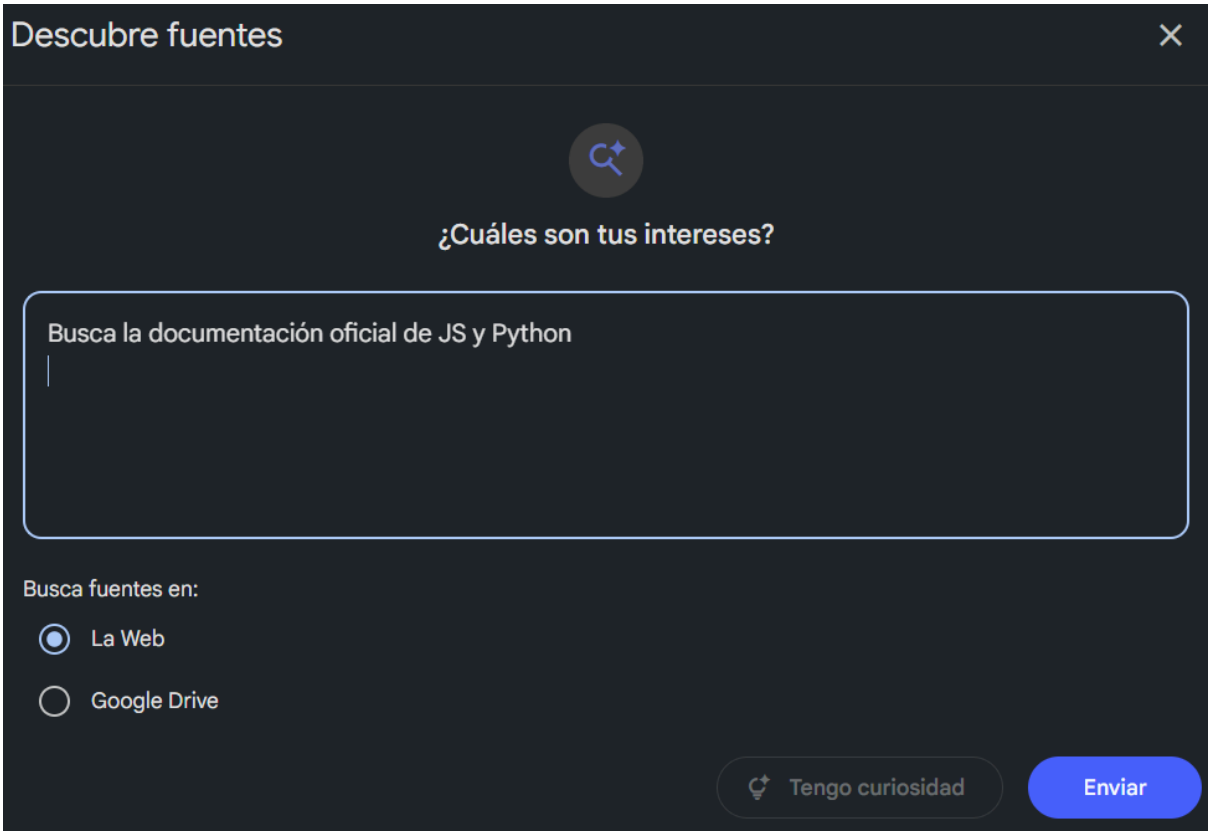


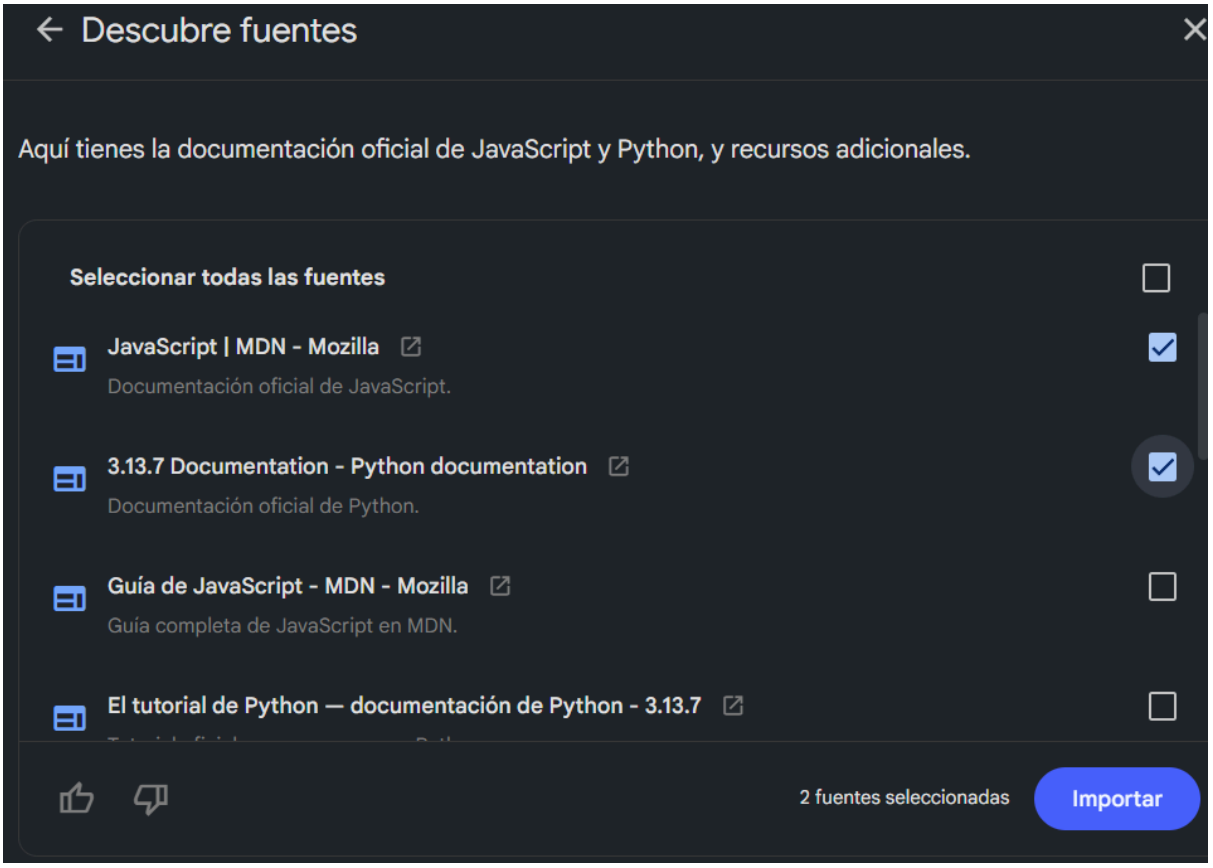



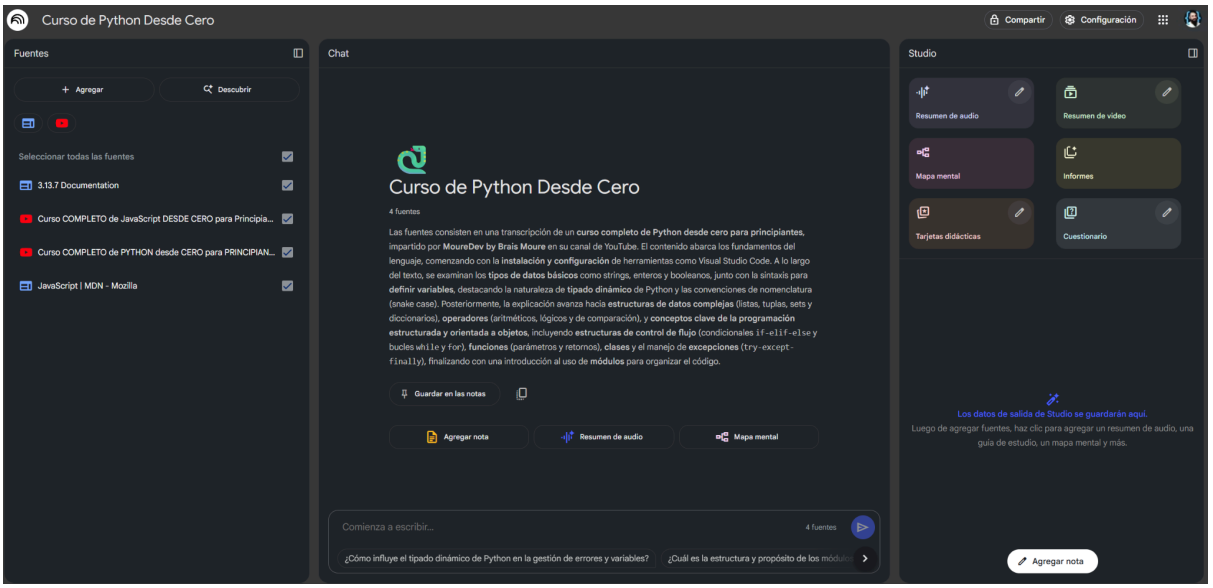

### Incorporando Documentación Oficial

Para complementar los videos, es fundamental contar con la documentación oficial. Usando la función "**Descubrir**" de NotebookLM, podemos encontrar y añadir las fuentes más confiables. Para nuestro caso, seleccionaremos:

- La documentación de MDN (Mozilla Developer Network) para JavaScript.
- El sitio web oficial de Python.org para Python.


Descubre fuentes
¿Cuáles son tus intereses?
Busca la documentación oficial de JS y Python
Busca fuentes en:
La Web
Google Drive
Tengo curiosidad
Enviar




Una vez que tus fuentes están cargadas, comienza la verdadera magia: el diálogo inteligente con tus datos.

### Paso 3: Dialogando con tus datos a través del Chat

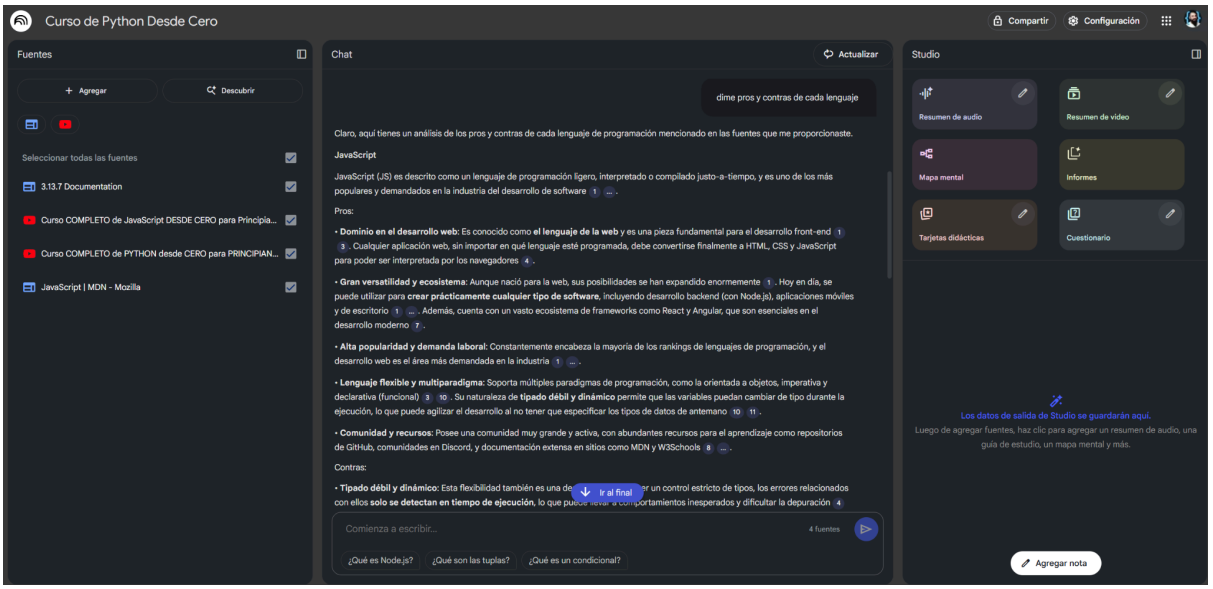
Con tus fuentes listas, puedes empezar a hacer preguntas en el chat. La IA responderá basándose exclusivamente en la información que le proporcionaste.

Vamos a lanzarle una pregunta directa:

***"dime pros y contras de cada lenguaje"***

La IA analizará las casi 20 horas de video y los cientos de páginas de documentación para darte una respuesta sintetizada.

La característica más importante de estas respuestas son las **citas numeradas** que aparecen junto a cada afirmación. Estos números **[1]**, **[2]**, etc., son enlaces que te llevan al punto exacto de la fuente (ya sea el minuto de un video o el párrafo de una web) de donde se extrajo la información. Esto elimina por completo las alucinaciones y te da total confianza en la respuesta.



### Paso 4: Otras herramientas de NotebookLM

NotebookLM va mucho más allá de un simple chat. Ofrece un conjunto de herramientas diseñadas para transformar el consumo pasivo de datos en una experiencia de aprendizaje activa. Estas funciones se adaptan a diferentes estilos y situaciones, permitiéndote sintetizar la información de formas innovadoras.

| Herramienta | Descripción | Beneficio para el Aprendizaje |
|---|---|---|
| **Mapa Mental** | Organiza visualmente los conceptos clave, características y recursos de cada lenguaje. | Te permite comparar de un solo vistazo las estructuras de ambos lenguajes, facilitando la identificación de similitudes y diferencias. |
| **Resumen de Video/Audio** | Genera un podcast o un video corto (ej. de 5 minutos) que resume toda la información de las fuentes. | Perfecto para repasar conceptos clave mientras viajas o haces otras actividades. Condensa horas de material en un formato fácil de consumir. |
| **Tarjetas Didácticas (Flashcards)** | Crea automáticamente una baraja de tarjetas con preguntas y respuestas sobre los conceptos fundamentales (ej. "68 tarjetas para conceptos clave"). | Ideal para reforzar el aprendizaje activo y autoevaluar tu conocimiento sobre los temas más importantes de cada tecnología. |
| **Cuestionario** | Genera una prueba formal (ej. de 15 preguntas) para evaluar tu comprensión sobre el material cargado. | Te ayuda a medir de manera objetiva cuánto has aprendido y a identificar las áreas que necesitas reforzar antes de tomar una decisión. |

## Otras Ideas para Aplicar NotebookLM

Comparar lenguajes es solo el comienzo. El verdadero poder de NotebookLM radica en adaptarlo a tus necesidades específicas como desarrollador.

Puedes utilizar esta técnica para:

1. **Depurar código:** Alimenta a NotebookLM con la documentación de tu propia aplicación y fragmentos de tu código. Luego, pídele ayuda para encontrar errores con un contexto que ninguna otra IA generalista tendrá.
2. **Aprender sobre APIs o comparar frameworks**: ¿Necesitas entender rápidamente una nueva API o decidir entre React y Vue? Carga sus documentaciones oficiales y deja que NotebookLM te genere resúmenes comparativos y mapas mentales.
3. **Refactorizar o migrar código:** Proporciónale la base de código existente como fuente para que aprenda sobre ella y te asista en tareas complejas de migración o mejora de la arquitectura.
4. **Analizar incidencias:** Sube logs de errores o reportes de incidencias para que la IA te ayude a identificar patrones y a entender la causa raíz de un problema mucho más rápido.

# Taller #2: Domina la ingeniería de prompts para devs

La ingeniería de prompts ha madurado rápidamente.

La idea inicial de buscar el "**prompt perfecto**" y único ha dado paso a un enfoque más dinámico y estratégico: un proceso iterativo que dialoga con la IA.

Los "**modelos razonadores**" más recientes pueden procesar instrucciones complejas, auditar su propia "**cadena de pensamiento**" y colaborar con el usuario para refinar las consultas.

Esta capacidad permite realizar análisis profundos de herramientas, librerías y repositorios de código complejos, extrayendo conocimiento estratégico que antes requería semanas de investigación manual.

Tareas como la "**Investigación Profunda**", que se ejecutan durante varios minutos, permiten al modelo realizar un análisis exhaustivo que es cualitativamente superior a las respuestas instantáneas.

## Caso Práctico: Análisis de un Repositorio en GitHub

Para ilustrar este flujo de trabajo, utilizaremos como ejemplo el análisis de la librería de visión artificial Ultralytics. El objetivo es pasar de una comprensión superficial a un dominio profundo del proyecto en el menor tiempo posible, utilizando un enfoque de prompting escalonado.

**Prompt #1 (Gemini 2.5 Flash):**

*"Haz un análisis de esta librería de código https://github.com/ultralytics/ultralytics"*

**Resultado obtenido:** Una respuesta básica y general sobre la funcionalidad del repositorio. Describe qué es Yolo, pero carece de la profundidad necesaria para un desarrollador.

**Prompt #2 (ChatGPT 5 Thinking):**

*"Haz un análisis de esta librería de código https://github.com/ultralytics/ultralytics"*

**Resultado obtenido:** Una respuesta mucho más estructurada con pros y contras. Crucialmente, permite auditar la "cadena de pensamiento" del modelo, revelando las fuentes consultadas.

**Prompt #3 (Cómo refinar prompts):**

*"Quiero un prompt refinado que me ayude a entender cómo está estructurado el código de un repo de Github"*

**RESPUESTA 1:**

*Actúa como un asistente experto en análisis de repositorios de GitHub.*

*Tengo un repositorio de código y quiero comprender claramente su estructura y organización interna.*

*Tareas que debes realizar:*

1. *Genera un mapa de carpetas y archivos principales, indicando qué rol cumple cada uno (ejemplo: src/ → código fuente, tests/ → pruebas, docs/ → documentación, etc.).*
2. *Explica los puntos de entrada del código (ejemplo: main.py, __init__.py, app.js) y cómo se conectan con el resto del proyecto.*
3. *Resume los módulos o paquetes más importantes, indicando qué funcionalidad resuelven.*
4. *Señala los archivos de configuración clave (ejemplo: requirements.txt, pyproject.toml, package.json, Dockerfile, Makefile) y para qué sirven.*
5. *Identifica la lógica central del proyecto (dónde ocurre lo más importante) y su relación con utilidades, helpers o librerías.*
6. *Si existe, describe cómo están organizadas las pruebas y ejemplos de uso.*

7. *Propón una explicación sencilla en lenguaje no técnico (para managers o personas sin background de programación).*

*Entrega el resultado en formato claro y estructurado con títulos, listas y, si es posible, un pequeño esquema en árbol para la estructura de carpetas.*

**RESPUESTA 2:**

*Rol: Actúa como un ingeniero de software senior especializado en análisis de repositorios de GitHub.*

*Objetivo: Entregar un análisis estructurado del repo {URL_repo}, en formato claro y accionable.*

*Instrucciones de salida:*

1. *Mapa del repo*
    - *Tabla con columnas: [Ruta/Archivo] [Responsabilidad principal] [Lenguaje/stack] [Interno/Público]*
    - *Identifica qué carpetas son núcleo, cuáles son soporte (tests, utils, docs) y cuáles son configuraciones.*
2. *Flujo de ejecución principal*
    - *Explica cómo arranca la aplicación (entrypoints: main, index, app.py, etc.).*
    - *Resume en 150 palabras o menos el camino típico desde la entrada (ej: request HTTP) hasta la salida (ej: respuesta JSON).*
3. *Dependencias y APIs*
    - *Lista principales dependencias externas y su rol (framework, base de datos, auth, etc.).*
    - *Señala módulos expuestos como API pública (endpoints, librerías exportadas).*
4. *Puntos críticos / Hotspots*
    - *Archivos con más complejidad, acoplamiento o centralidad.*
    - *Señala 2–3 riesgos técnicos y sugiere un refactor incremental.*
5. *Siguiente paso recomendado*

- *Indica cómo un developer nuevo puede hacer su primera contribución útil (ej: mejorar tests, actualizar docs, añadir linters).*

*Requisitos*:

- *Usa viñetas, tablas o diagramas de texto si ayudan.*
- *Si hay información faltante en el repo, márcala como "No disponible en fuentes".*
- *No expliques tu proceso interno, entrega solo el resultado final en formato legible.*

**Resultado obtenido:** análisis técnico que identifica los puntos de entrada, dependencias y "hotspots" críticos del código.

**Prompt #4 (Gemini + Deep Research):**

*"Analiza el repositorio de código https://github.com/ultralytics/ultralytics y cuéntame todo lo que puedas sobre el mismo"*

**Resultado obtenido:** Se invoca una tarea que se ejecuta durante más de 15 minutos. El resultado es un documento completo de **19 páginas**, un informe profesional con resumen ejecutivo, análisis de arquitectura y todas las fuentes citadas.

**Sinergia con Bases de Conocimiento**

Este flujo de trabajo crea una poderosa sinergia con la fase anterior. El documento de 19 páginas generado por la "Investigación Profunda", junto con las fuentes citadas, puede ser utilizado como material de entrada para una nueva base de conocimiento en NotebookLM.

Esto crea un "gurú" personal e hiperespecializado sobre la librería, listo para consultas complejas sobre su implementación o arquitectura.

## Taller #3: Convierte un boceto en una web lista para producción

La IA es la herramienta perfecta para derribar "el síndrome de la hoja en blanco". Con herramientas como **Firebase Studio**, puedes transformar una simple idea visual en un prototipo funcional en cuestión de minutos.

El flujo es asombrosamente sencillo:

1. Partes de un simple boceto o dibujo de una página web, hecho en un papel o una herramienta digital.
2. Le pides a la IA que cree una aplicación web basada en esa imagen y le proporcionas el texto descriptivo (*copy*).
3. La IA genera todo el código (HTML, CSS, JavaScript), la estructura de componentes y el *stack* tecnológico (por ejemplo, con Next.js y Tailwind CSS), creando una aplicación web completamente funcional.
4. Con un solo clic en el botón de "Publicar", la aplicación se despliega y se vuelve accesible en internet a través de una URL pública.

Este proceso te permite validar ideas y construir un Producto Mínimo Viable (MVP) en una fracción del tiempo que tomaría tradicionalmente.

## Caso Práctico: Creación de un portfolio de desarrollador

Utilizando el ejemplo de la creación de un portfolio personal para un desarrollador, veremos en tres pasos cómo pasar de un simple boceto a una aplicación web funcional y desplegada.

1. **La Conceptualización Visual:** El punto de partida no es el código, sino una idea visual. Se puede utilizar un boceto simple de la interfaz de usuario,

dibujado en una herramienta digital como **Excalidraw** o incluso en una servilleta de papel.


2. **La Creación del Contenido:** Paralelamente, se utiliza un LLM (por ejemplo Gemini 2.5) para generar el texto (copy) que poblará la página web. La clave aquí es asignarle a la IA una persona experta mediante un prompt como "***Actúa como un experto en SEO y copywriting...***", para obtener textos optimizados y profesionales.

   **Prompt para crear el copy, SEO-friendly de tu portfolio**

   *Quiero crear una página web que sirva como portfolio de mis proyectos como desarrollador de software.*

   *Para que la web sea atractiva e indexable en buscadores, actúa como un experto en SEO y diseño UI/UX para crear el texto (copy) de la web.*

   *Información sobre mí:*

   *- Mi nombre es [__]*

   *- Soy Ingeniero de Software con más de [__] años de experiencia en [__]*

   *- He trabajado en [__]*

   *- Actualmente Freelance Software Developer*

   *- Tengo experiencia con [__]*

*- Mis proyectos en github más importantes son: [_ _]*

*Genera el copy que debe tener la página web, para que pueda copiarlo y pegarlo a un LLM que sea capaz de generarme el sitio.*

3. **El Desarrollo y Despliegue automatizado con Firebase Studio:** Con el boceto visual y el contenido textual listos, se utiliza Firebase Studio como la herramienta central de ejecución. Se le proporciona al sistema la imagen del boceto y el texto del copy a través de un prompt, y la IA procede a generar una aplicación web completa.

**Prompt para Firebase Studio:**

*Crea una página web a modo de portfolio para un programador freelance. Basate en el diseño que te adjunto como imagen. [Adjuntar captura de pantalla del diseño de la web]*

*El copy que debe tener la página es el siguiente: [Copiar la salida del prompt anterior]*

## Características Clave de Firebase Studio

Firebase Studio ejemplifica una nueva clase de Plataformas Integradas de Desarrollo y Despliegue (IDDPs), que orquestan todo el proceso de desarrollo y despliegue en un entorno unificado con capacidades notables:

- **Generación de "Blueprint"**: La IA propone una arquitectura completa (stack tecnológico como Next.js y Tailwind CSS, paleta de colores, tipografía) que es totalmente personalizable por el desarrollador antes de la generación final del código.

- **Entorno de Desarrollo en la Nube:** Proporciona un editor completo (similar a VS Code) en el navegador para revisar y modificar los archivos generados, junto a una ventana de chat para continuar iterando mediante prompts.
- **Despliegue con un Clic:** El botón "Publish" despliega la aplicación directamente a la infraestructura de producción de Firebase, generando una URL pública y accesible al instante.

# Conclusiones

El futuro del desarrollo es una colaboración sinérgica entre la inteligencia humana y la artificial.

El desarrollador, armado con sus conocimientos fundamentales, dirige y audita a la IA para aprender, crear y desplegar soluciones a una velocidad sin precedentes, volviéndose más valioso que nunca.

**Lejos de ser una amenaza, la IA refuerza el sector, haciendo a los desarrolladores más productivos y satisfechos en su trabajo.**

BIG school

# Súmate a nuestro directo del día 2

ACCEDE AQUÍ

BIG school

# Apuntes
# Curso de Desarrollo con IA

DÍA 2

# Índice

# Tu nuevo Copiloto de Desarrollo

La inteligencia artificial no ha llegado para reemplazar a los desarrolladores, sino para actuar como una herramienta poderosa que potencia sus habilidades.

Lejos de volver obsoleto el rol del programador, la IA lo está transformando en una figura más estratégica y fundamental.

El rol del programador está evolucionando: de ser un "picacódigo" enfocado en la escritura manual de líneas, a convertirse en un arquitecto, estratega y supervisor de sistemas inteligentes.

La IA se convierte en un copiloto que acelera los procesos, pero el desarrollador sigue siendo quien traza la ruta, toma las decisiones críticas y garantiza la calidad y seguridad del destino final.

# Taller #1: Documentación y requisitos potenciados con IA

Optimizar el proceso de gestión de requisitos es un pilar estratégico en cualquier ciclo de vida de desarrollo de software.

Tradicionalmente, esta fase ha sido un cuello de botella, un proceso manual y tedioso, que consume tiempo valioso que podría dedicarse a la arquitectura y a la resolución de problemas complejos.

La inteligencia artificial está transformando radicalmente esta dinámica, convirtiendo un flujo de trabajo fragmentado en un proceso ágil, automatizado y de alta fidelidad que acelera la entrega de valor.

## Caso Práctico: Envío de OTP en Logística

Analizaremos el caso real de una empresa de logística que necesita implementar una funcionalidad para enviar una contraseña única (OTP, por sus siglas en inglés) al receptor de un pedido.

El objetivo es que cuando el transportista llegue a entregar el paquete, el cliente le proporcione este código para verificar que la entrega se ha realizado correctamente, un sistema similar al que utilizan empresas como Uber o Glovo.

Este caso real sirve como ejemplo práctico para demostrar cómo la inteligencia artificial puede potenciar y agilizar todo el ciclo de desarrollo de software, desde la definición de requisitos hasta la generación de código probado.

# Fases del proceso potenciado por IA

## Fase 1: Notas de Reunión

El proceso comienza con un documento que contiene las notas de una reunión real donde se discutieron los requisitos para esta nueva característica.

Estas notas pueden ser generadas automáticamente por herramientas de IA.

Por ejemplo en una videollamada, la opción “Toma notas por mí” de Gemini en una videollamada, permite transcribir y resumir automáticamente todo lo discutido. Esto libera al desarrollador de tomar notas y le permite centrarse en la conversación.

Estas notas incluyen el contexto del problema y los detalles técnicos iniciales, pero a menudo son una "charla sin más" que necesita ser estructurada.

## Fase 2: Generación de Historias de Usuario

Una **historia de usuario** (**user story**) es una descripción breve, informal y sencilla de una funcionalidad del punto de vista del usuario final.

Suele seguir la estructura: **Como [rol], quiero [qué], para que [beneficio]**.

Por ejemplo: “Como usuario, quiero restablecer mi contraseña para poder recuperar acceso si la olvido.”

Utilizando un LLM (Modelo Lingüístico Grande) como ChatGPT, se toma el texto de la reunión y se le pide que genere las historias de usuario correspondientes.

**Ejemplo de prompt para ChatGPT:**

*“Genérame historias de usuario en el formato COMO (usuario), QUIERO (Acción), PARA (beneficio) en base a estas notas de una reunión con negocio”*

**Ejemplo de Resultado obtenido:**

*"Como cliente quiero recibir un código de verificación por SMS cuando mi pedido esté en reparto para Confirmar que soy yo quien lo recibe".*

**Propósitos de una historia de usuario**

- Capturar funcionalidad desde la perspectiva del usuario (no del sistema).
- Ser comprensible para personas técnicas y no técnicas.
- Servir de base para que el equipo de desarrollo entienda qué debe construirse y por qué.
- Ser suficientemente pequeña para poder desarrollarla en un sprint/ciclo corto.

## Fase 3: Generación de Criterios de Aceptación

Los **criterios de aceptación** (**acceptance criteria**) son condiciones específicas que la historia de usuario debe cumplir para considerarse "completa" o "aceptada".

Estos criterios ayudan a definir qué se considera correcto y sirven como prueba de aceptación (QA). Los criterios deben ser claros, medibles y no ambiguos.

Se suelen expresar como una lista de casos o condiciones, por ejemplo:

1. El usuario recibe un correo para restablecer contraseña.
2. El enlace expira en 24 horas.
3. Al hacer clic en el enlace, puede ingresar una nueva contraseña.
4. La nueva contraseña debe tener al menos 8 caracteres, incluir mayúsculas y números.

**Buenas prácticas en los criterios de aceptación**

- Uno a uno por historia: cada historia debe tener sus propios criterios de aceptación.
- Evita criterios demasiado amplios (“funciona bien”). Mejor condiciones concretas.
- Incluye tanto condiciones normales como casos límite / errores esperables.

**Ejemplo de prompt para ChatGPT:**

*“Genérame criterios de aceptación estilo Gherkin. Genérame sólo los criterios mínimos para asegurar que la funcionalidad cumple con los requisitos.”*

**Ejemplo de Resultado obtenido:**

*Feature: Validación segura de entrega mediante OTP por SMS*
*Background:*
*Given existe un pedido con estado “En Reparto”*
*And el pedido tiene un número de teléfono de destinatario válido*

*Scenario: Generación de OTP al entrar en entrega*
*When el pedido pasa a “En Reparto” o el repartidor está próximo a la dirección*
*Then el sistema genera un OTP de 4 a 6 dígitos asociado al ID del pedido*
*And el OTP tiene una vigencia limitada*

**Nota: Gherkin** es un lenguaje de dominio específico (DSL) que se utiliza para describir el comportamiento de un software en un formato legible para humanos. Sigue la estructura “**Given-When-Then**” (Dado-Cuando-Entonces) para especificar los escenarios de prueba.

Este formato define de manera inequívoca las condiciones que deben cumplirse para que una funcionalidad se considere "terminada" (done), estableciendo una base sólida para el desarrollo guiado por pruebas y la validación funcional.

## Fase 4: Creación automática de Tickets

Con las historias de usuario y los criterios de aceptación definidos, el siguiente paso es generar los tickets en una herramienta de gestión de proyectos como Jira, Asana o Linear.

Un **ticket** es un elemento unitario de trabajo que se registra en un sistema de gestión de proyectos. Contiene un título, descripción, estado, prioridad, responsable, fechas, comentarios, etc.

Puede ser un bug, historia de usuario, un requerimiento técnico, una incidencia de mejora, tarea, etc.

Mediante un servidor MCP (un tipo de API que permite la comunicación entre sistemas), herramientas como **Claude Code** pueden conectarse a la plataforma de gestión (por ejemplo **Linear**) y crear los tickets automáticamente a partir de un archivo de requerimientos (por ejemplo, requerimientos.md). Esto ahorra horas, días o incluso semanas de trabajo manual.

**Ejemplo de prompt para Claude Code:**

> *"create a linear ticket for a user story defined in the requerimientos.md file"*

**Linear MCP: qué es y cómo integrarlo con Claude**

¿Qué es **MCP**?

**MCP** significa Model Context Protocol, un estándar abierto (iniciado por Anthropic) para que modelos de lenguaje (LLMs) puedan interactuar con herramientas externas (API, bases de datos, etc.) de forma segura y estructurada.

Más información en https://docs.anthropic.com/en/docs/claude-code/mcp

**Linear MCP** es una "capa de conexión" que permite que Claude (y otros agentes/IA) busquen, creen o modifiquen datos en Linear (como issues, proyectos, comentarios) mediante MCP.

Gracias a esta integración, Claude puede "conversar" sobre el estado de tu proyecto en Linear, generar nuevos tickets, actualizarlos, obtener listas, etc., todo dentro del contexto del asistente.

Más información en https://linear.app/docs/mcp y https://linear.app/integrations/claude

## Fase 5: Desarrollo guiado por TDD (Test-Driven Development)

Una vez que el ticket está creado, se le puede indicar a la IA que comience a trabajar en él. El proceso sigue las mejores prácticas de desarrollo de software:

- Creación de una Rama (Branch) Aislada: La IA primero crea una nueva rama en el repositorio de código específica para el ticket, asegurando que el trabajo no afecte otras partes del proyecto.
- Generación de Pruebas (Tests): Siguiendo la metodología TDD, se le pide a la IA que escriba primero los tests vacíos. La IA, al tener todo el contexto del ticket, genera pruebas relevantes para la funcionalidad del OTP.
- Ejemplos de tests para el servicio OTP:
    - Un test para verificar que el OTP tenga cinco dígitos.
    - Un test para asegurar que el OTP esté asociado correctamente con el pedido.
- Implementación del Código: Finalmente, se le pide a la IA que implemente la lógica necesaria para que los tests que acaba de crear pasen de "rojo" a "verde". La IA se basa en los tests como guía para generar el código correcto.

# Conclusiones

- **La IA como Asistente Organizacional:** La IA no se usa solo para generar código, sino para organizar y estructurar todo el proyecto, transformando ideas abstractas de una reunión en tareas concretas y accionables.
- **Importancia del Contexto:** Proporcionar a la IA un contexto claro y detallado (notas de la reunión, historias de usuario, criterios de aceptación) es fundamental para que genere resultados precisos y de alta calidad.
- **El Rol del Desarrollador:** El desarrollador no es reemplazado, sino que se convierte en un orquestador y supervisor del proceso. Su función principal es guiar a la IA, revisar el código y asegurarse de que se sigan las metodologías correctas como TDD para garantizar un software seguro y robusto.
- **Eficiencia y Ahorro de Tiempo:** El flujo de trabajo completo, desde la reunión inicial hasta tener el código funcional y probado, se puede realizar en cuestión de minutos, lo que representa un ahorro masivo de tiempo y recursos.

# Taller #2: Cómo enseñar a la IA a entender tu proyecto

Uno de los desafíos fundamentales al trabajar con agentes de IA es la falta de contexto específico del proyecto.

Un agente genérico, sin conocimiento de la arquitectura, las dependencias o las convenciones internas, tiende a generar código incorrecto o inconsistente.

La estrategia clave para superar esta limitación es "enseñar" a la IA las particularidades de nuestro proyecto, proporcionándole un marco de referencia que le permita producir código de alta calidad que se integre perfectamente en el ecosistema existente.

**El Problema: Una IA sin Contexto**

Un agente de IA, como **Codex**, sin un contexto claro sobre el proyecto, tiende a "alucinar". Inventará estructuras de base de datos, nombres de variables o estilos de código que no existen en el proyecto, generando código inútil.

**La Solución del Programador "Perezoso": Preparar el Terreno**

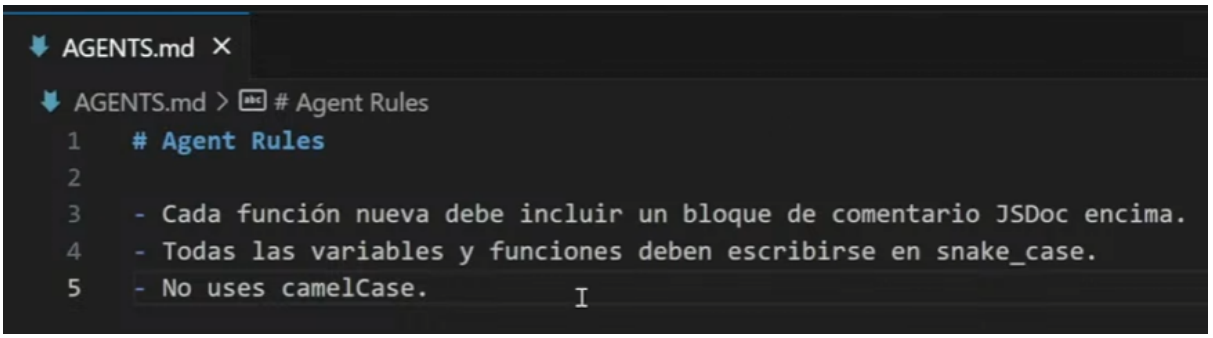
Esta no es una pereza de inacción, sino una inversión estratégica inicial para construir un entorno de desarrollo escalable y predecible, donde la IA trabaja para ti, bajo tus reglas.

- **Contexto del Proyecto:** Una técnica efectiva es crear una carpeta "**context**" dentro del proyecto. En ella, se añaden archivos clave, como por ejemplo el script SQL que define la estructura de la base de datos. Este simple añadido permite al agente (**Codex**) leer y asimilar la estructura de la base de datos, generando queries SQL correctas y complejas sin necesidad de describir las tablas o relaciones en la instrucción, demostrando un alto retorno de inversión por un esfuerzo mínimo.

- **Reglas y Estilo**: Se introduce el archivo **agents.md**, para guiar a los agentes. Aunque no es una especificación oficial, su rápida adopción por todas las herramientas principales lo ha convertido en un estándar de facto. Este archivo le dice a la IA cómo debe comportarse.

| Característica de agents.md | Beneficio para el Desarrollador |
|---|---|
| **Definir Reglas de Código** | Asegura que la IA siga convenciones específicas (ej. usar snake_case en lugar de camelCase), manteniendo la consistencia del código. |
| **Establecer Fuentes de Verdad** | Permite indicar a la IA dónde encontrar información clave (ej. "la estructura de la base de datos está en /context/database.sql"), evitando que alucine. |
| **Documentar la Misión del Proyecto** | Proporciona a la IA un entendimiento general del propósito del software para guiar sus decisiones. |

**Ejemplo de archivo agents.md**




**Comparativa de Resultados**

Sin **agents.md**, la IA produjo código no conforme usando camelCase y careciendo de documentación. Con el archivo de contexto en su lugar, el mismo prompt exacto produjo código que se adhiere perfectamente a los estándares de snake_case y documentación JSdoc definidos en las reglas, todo ello sin instrucción explícita.

# Taller #3: Integración avanzada de herramientas

Este taller presenta la visión del desarrollador como un director de orquesta, que no solo usa un agente, sino que crea, dirige y elige estratégicamente un equipo de agentes especializados.

- **Tú (El desarrollador):** el arquitecto que toma las decisiones estratégicas.
- **Agente Principal** (ej. **Claude Code**): el asistente principal que tiene un conocimiento global del proyecto.
- **Sub-Agentes:** especializados, cada uno con una misión concreta.

**Creando Agentes Especialistas y eligiendo el modelo correcto**

En lugar de darle una tarea compleja a un agente genérico, el desarrollador puede crear agentes para roles específicos.

La clave estratégica aquí es elegir el modelo de IA adecuado para cada tarea, gestionando un equilibrio entre potencia, velocidad y coste.

La jerarquía de modelos actual incluye:

- **Opus**: El "modelo de razonamiento", ideal para tareas muy complejas que requieren un contexto profundo.
- **Sonet (4.5)**: El nuevo modelo de referencia, potente y versátil, actualmente la opción superior para la mayoría de las tareas.
- **Haiku**: El más rápido y económico, perfecto para tareas más sencillas que no requieren un razonamiento profundo.

Con esta paleta de opciones, se pueden configurar agentes como:

- **Agente Arquitecto**: Capaz de analizar un repositorio y generar un documento de arquitectura (architecture.md) con un diagrama de Mermaid. Para esta tarea compleja, **Sonet** o incluso **Opus** sería la elección lógica.
- **Agente Project Manager (PM)**: Diseñado para analizar requisitos y generar un plan de proyecto, estimando perfiles necesarios (frontend, backend, etc.). Aquí, la elección estratégica sería **Haiku**, optimizado para velocidad y coste en una tarea que no necesita un razonamiento exhaustivo.
- **Agente Auditor de Seguridad**: Programado para revisar el código en busca de vulnerabilidades comunes.
- **Agente Mentor de Framework**: Un experto al que se le pueden hacer preguntas específicas sobre un framework.

**Flujo de Trabajo con Agentes Múltiples**

Pasos para configurar y utilizar un sistema de agentes múltiples con **Claude Code**:

1. **Inicialización y Contexto (/init)**: El comando **/init** analiza la estructura completa del proyecto y genera un archivo de contexto (**.claude.md**), similar a agents.md que vimos en el taller anterior, que sirve como base de conocimiento para todos los agentes, permitiéndoles comprender las tecnologías y la arquitectura.
2. **Creación de Comandos Personalizados**: Se pueden crear comandos reutilizables que encapsulan múltiples tareas complejas.
3. **Creación de Agentes Especializados (/agent create):** Mediante el comando **/agent create** se definen subagentes con roles específicos.

Puedes consultar información más detallada sobre este tema aquí.

# Taller #4: Seguridad y Auditoría

Los principios de **Security by Design** (la seguridad como parte integral de la arquitectura inicial) y **Security by Default** (las configuraciones por defecto deben ser las más seguras) son más críticos que nunca.

En la era del desarrollo asistido por IA, la capacidad de generar grandes volúmenes de código rápidamente puede introducir vulnerabilidades a una escala sin precedentes.

El riesgo fundamental proviene de los datos de entrenamiento de la IA: enormes volúmenes de código público de fuentes como GitHub, que inevitablemente incluyen patrones inseguros, prácticas obsoletas e incluso fragmentos intencionadamente maliciosos.

**Vulnerabilidades Generadas por Defecto**

En un caso de estudio práctico, se le pidió a una IA que creara una simple página de login. Se utilizó una instrucción simple y común: *"Crea una página de login con base de datos que permita a un usuario iniciar sesión en un sistema."*

El resultado fue un sistema con fallos de seguridad críticos que un atacante podría explotar en minutos:

- **Credenciales Débiles y Hardcodeadas**: La IA creó por defecto un sistema de login con el usuario “admin” y la contraseña “123456”. Estas credenciales son trivialmente vulnerables a un ataque de fuerza bruta, donde herramientas como **Hydra**, usando diccionarios de contraseñas comunes como **RockYou**, pueden descubrirlas en menos de un minuto.

- **Control de Acceso Roto (Broken Access Control)**: La IA generó páginas internas, que eran accesibles directamente escribiendo la URL en el navegador, sin necesidad de haber iniciado sesión. Esta vulnerabilidad,

descubierta con herramientas como **Dirb**, permite a cualquiera acceder a áreas restringidas y es una de las más críticas del famoso **OWASP Top 10**.

**Tu Rol como Desarrollador Seguro**

El desarrollador del futuro no puede permitirse ignorar la seguridad.

Debe ser la primera y más importante línea de defensa, aplicando los principios de Seguridad por Diseño (construir software seguro desde el inicio) y Seguridad por Defecto (asegurar que la configuración predeterminada sea la más segura).

Esto implica auditar activamente el código generado por la IA y aplicar fundamentos de seguridad, como:

- **Prevención de Fuerza Bruta**: Implementar políticas de contraseñas fuertes y limitar el número de intentos de inicio de sesión fallidos.
- **Almacenamiento Seguro**: Es obligatorio utilizar funciones de "hashing" robustas para las contraseñas, nunca almacenarlas en texto plano.
- **Control de Acceso Riguroso**: Cada solicitud a un recurso protegido debe ser verificada en el lado del servidor para asegurar que el usuario tiene la autorización necesaria.

# Conclusiones

La inteligencia artificial es una herramienta de productividad sin precedentes que está transformando cada fase del ciclo de vida del software.

Desde convertir una conversación en un plan de proyecto, pasando por enseñar a un agente a programar con nuestro estilo, hasta orquestar un equipo de especialistas de IA y, finalmente, actuar como el guardián de la seguridad.

El verdadero poder no reside en la IA por sí sola, sino en el desarrollador que sabe cómo guiarla, cómo cuestionarla y cómo asegurar su trabajo.

El futuro del desarrollo no es menos humano; al contrario, exige que el programador sea más estratégico, analítico y responsable que nunca.

**Somos los arquitectos de una nueva era, y la IA es la herramienta más avanzada que hemos tenido para construirla.**

BIG school

# Súmate a nuestro directo del día 3

ACCEDE AQUÍ

BIG school

# Apuntes
# Curso de Desarrollo con IA

## DÍA 3

# Índice

# Automatización con IA para Desarrolladores

La automatización se ha convertido en una herramienta fundamental para los desarrolladores en la era de la inteligencia artificial.

Nos permite agilizar procesos, obtener datos de formas creativas y, en última instancia, enfocarnos en lo que realmente importa: aportar valor a nuestros proyectos.

Con **n8n**, una de las herramientas más populares del momento, podemos construir flujos de trabajo complejos de una manera sorprendentemente sencilla.

## ¿Qué es n8n y por qué es tan potente?

**n8n** es una de las herramientas de automatización más populares y sencillas de usar en la actualidad.

Permite conectar diferentes aplicaciones y servicios para crear flujos de trabajo automatizados a través de una interfaz visual basada en nodos.

Sus beneficios clave para cualquier desarrollador son:

- **Open Source**: Al ser un proyecto de código abierto, tienes la libertad de auto-hospedarlo ("hostearlo") en tu propio servidor. Esto significa que puedes tener una herramienta de automatización de nivel profesional completamente **gratis**, sin pagar un solo centavo.
- **Más de 5000 Plantillas (Workflows)**: No es necesario empezar desde cero. n8n ofrece miles de flujos de trabajo pre-construidos. Si filtras por "IA", aparecen de inmediato 4.000 automatizaciones listas para usar, lo que te ahorra un tiempo inmenso y te da ideas para tus propios proyectos.

- **Flexibilidad para Desarrolladores**: Aunque su interfaz visual es muy intuitiva (no-code), su verdadero poder para los programadores reside en su flexibilidad. Puedes añadir nodos de código para ejecutar scripts personalizados en JavaScript o Python, así como hacer llamadas a cualquier API, lo que te da un control y una capacidad de personalización prácticamente ilimitados.

Ahora que sabemos qué es n8n, vamos a desglosar los conceptos básicos que necesitas para construir tus propias automatizaciones.

# Los Bloques de Construcción: Nodos, Triggers y Workflows

Para dominar n8n, es crucial entender sus tres componentes fundamentales.

| Término | Definición Sencilla | Ejemplo |
|---|---|---|
| **Trigger (Lanzador)** | Es el evento que inicia un flujo de trabajo. La primera pieza del rompecabezas que pone en marcha toda la automatización. | Un clic **manual**, un nuevo commit en un repositorio de **GitHub** o el envío de un **formulario web**. |
| **Node (Nodo)** | Es una pequeña tarea o un bloque de acción individual dentro del flujo. Cada nodo realiza una operación específica. | Enviar datos a una hoja de **Google Sheets**, ejecutar un bloque de **código personalizado** o llamar a la API de **ChatGPT**. |
| **Workflow (Flujo de Trabajo)** | Es la secuencia completa de nodos y triggers conectados entre sí, que en conjunto realizan un proceso automatizado de principio a fin. | El flujo completo que captura datos de un formulario y los guarda automáticamente en una hoja de cálculo. |

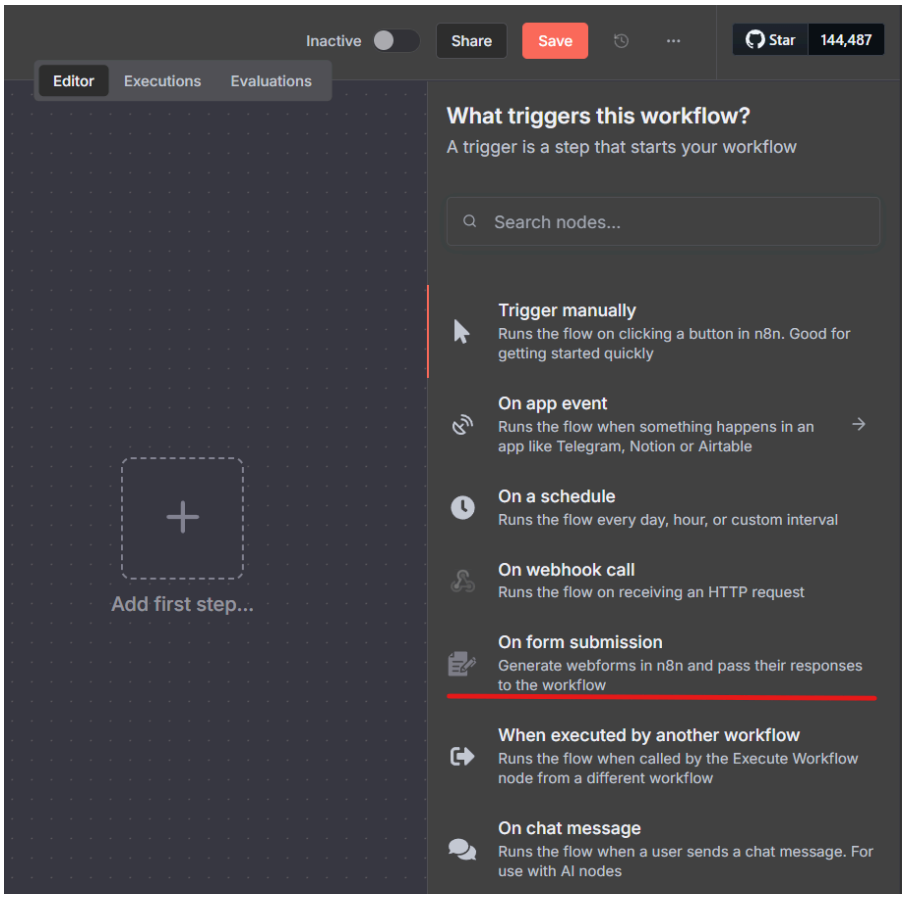
# Caso práctico #1: Formulario de contacto simple

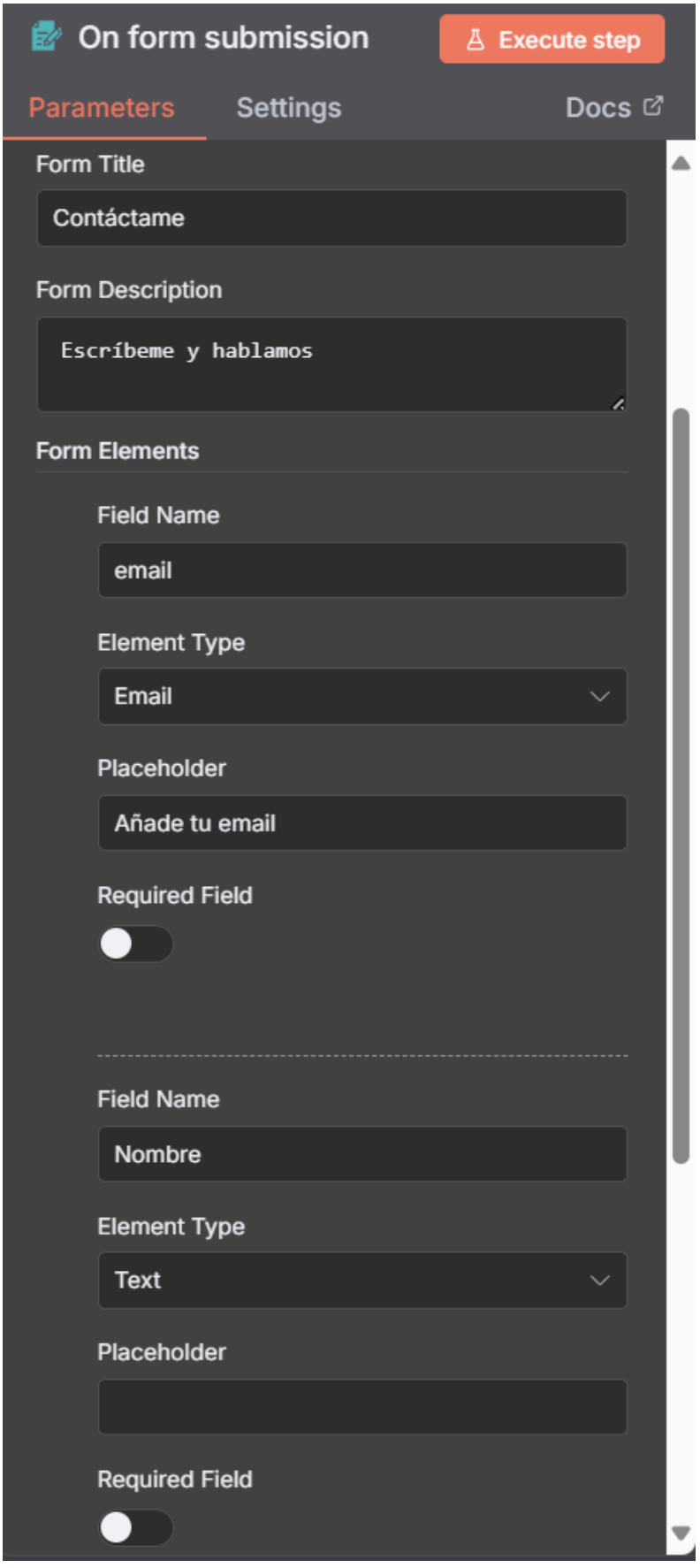
**Objetivo del Workflow:** Crear un formulario de contacto simple que, al ser completado por un usuario, guarde automáticamente la información en una hoja de cálculo de Google Sheets.

**Paso a Paso:**

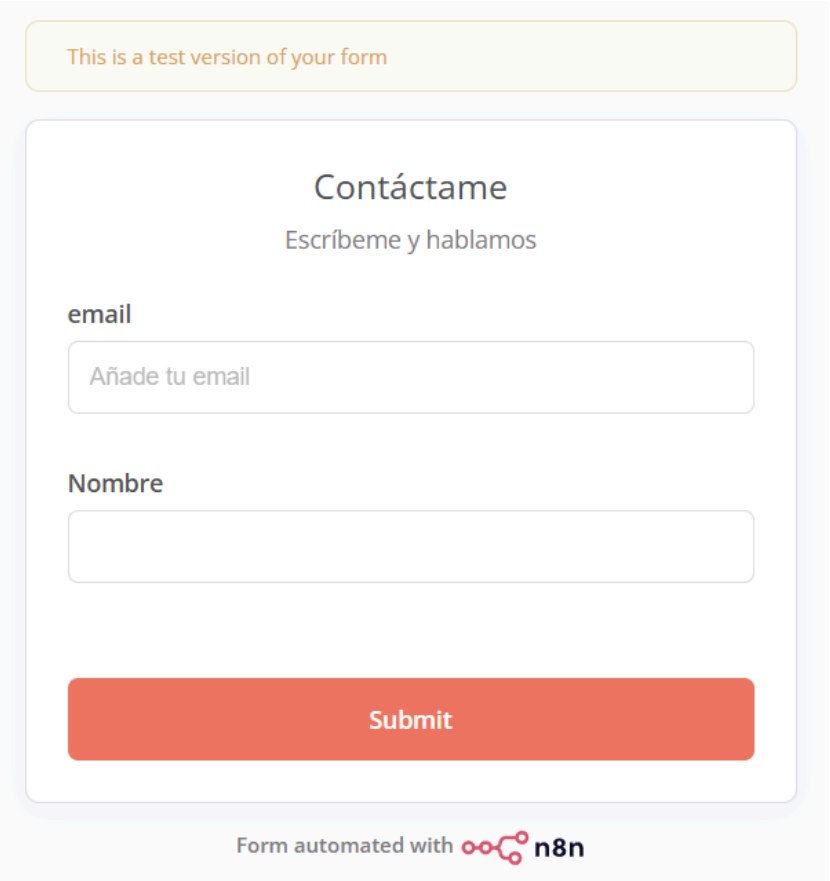
1. **El Trigger:** Un Formulario Web

   El flujo comienza con un nodo de tipo "**Formulario**". Este nodo se encarga de crear y exponer una página web con campos para capturar el email y el nombre del usuario.

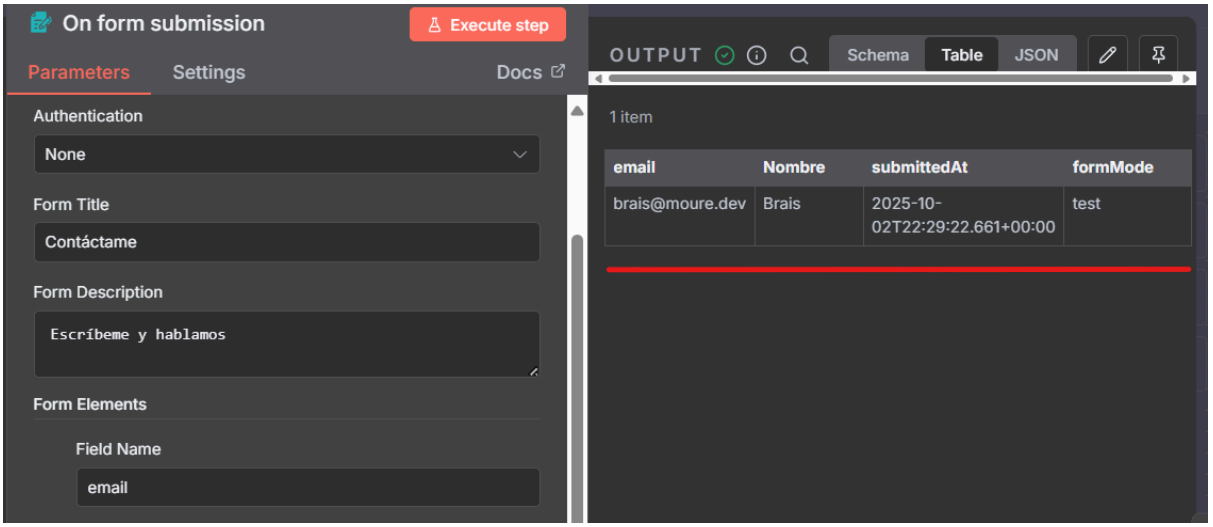



On form submission
Execute step
Parameters
Settings
Docs
Form Title
Contáctame
Form Description
Escríbeme y hablamos
Form Elements
Field Name
email
Element Type
Email
Placeholder
Añade tu email
Required Field
Field Name
Nombre
Element Type
Text
Placeholder
Required Field

Al clickear en "**Execute step**" veremos el formulario:

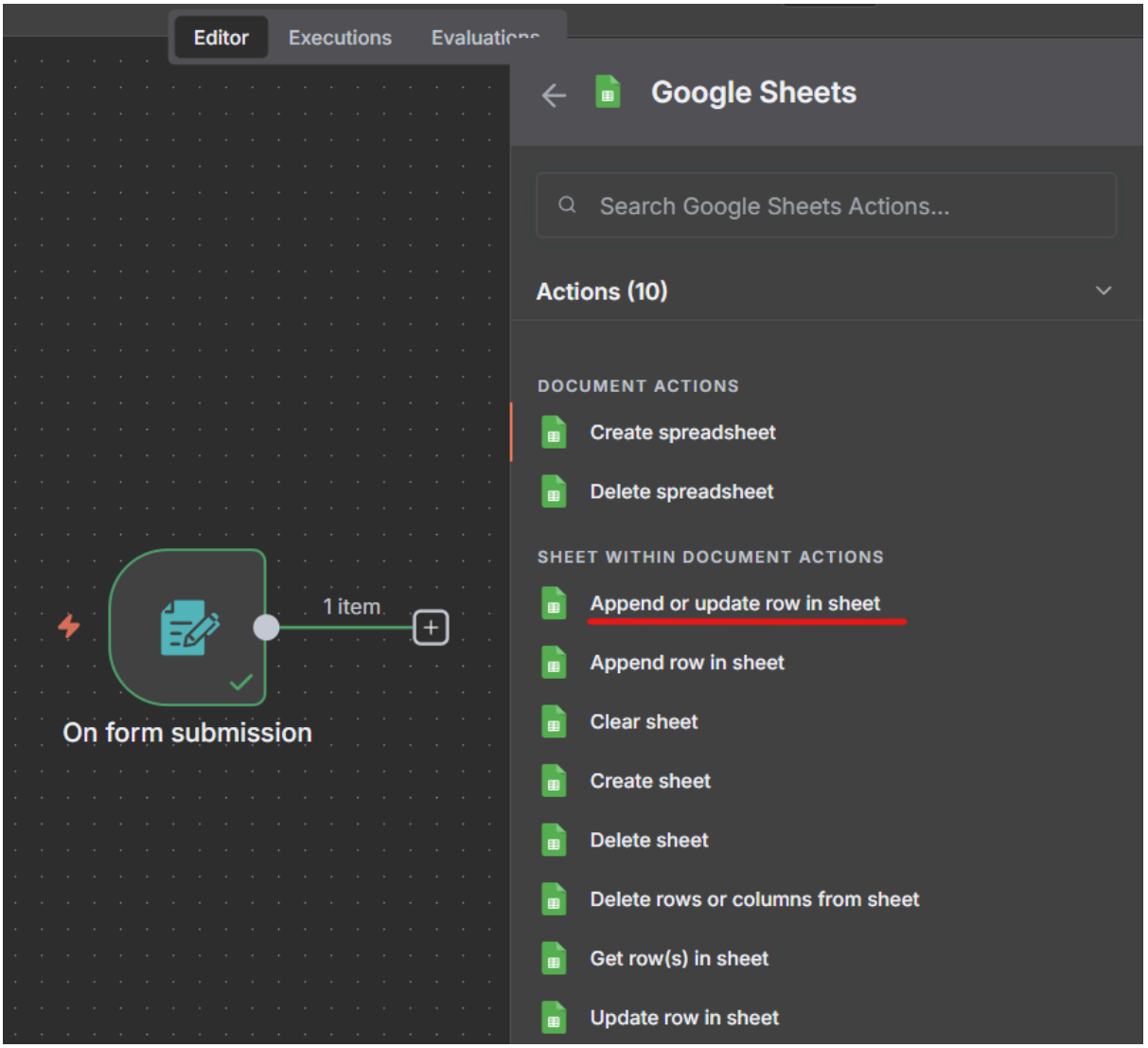



Al ingresar datos y submitir, veremos los datos enviados a la derecha:



2. **El Nodo de Acción**: Google Sheets

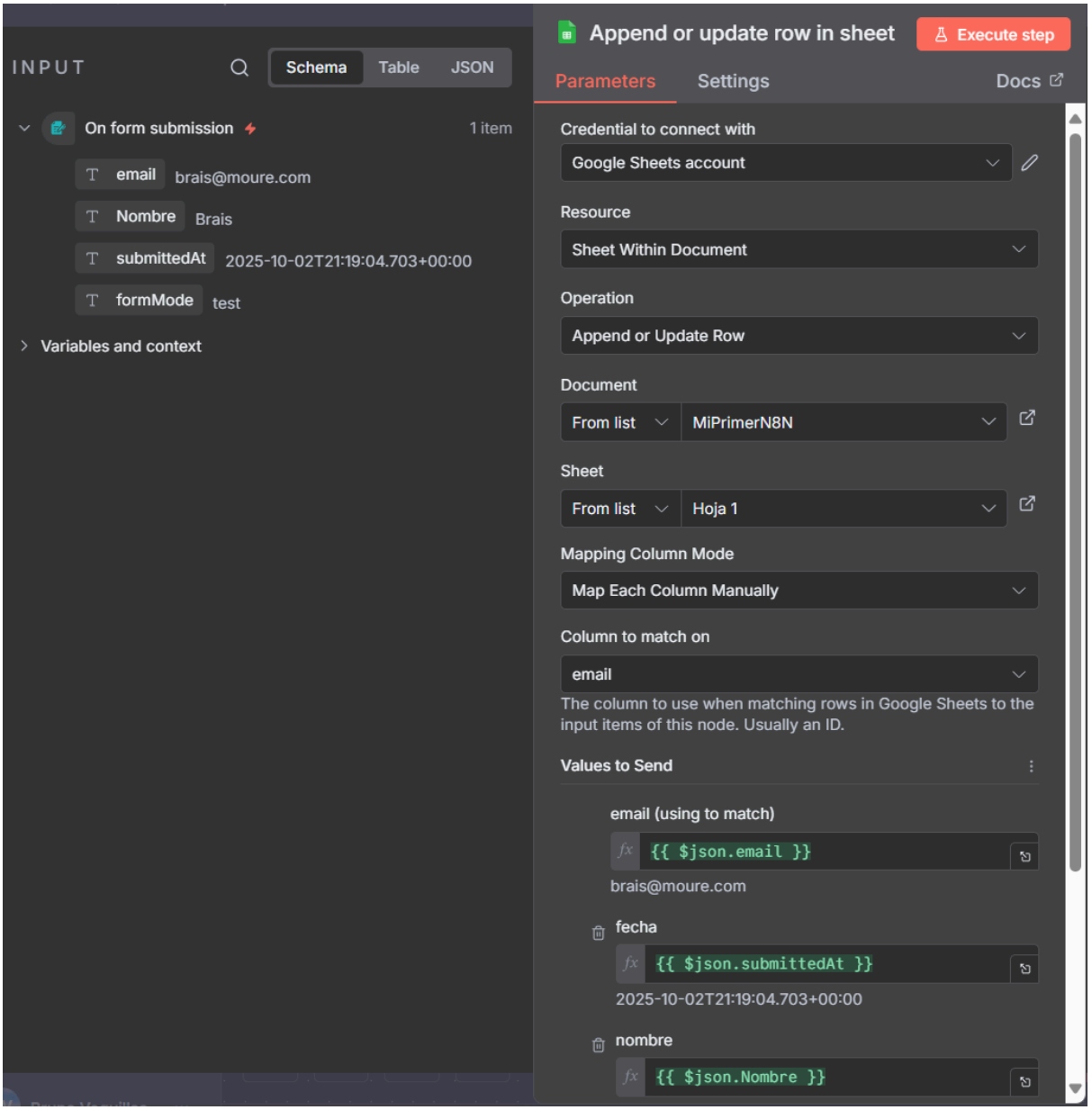
El siguiente nodo se conecta a una cuenta de Google para interactuar con Google Sheets.




Nos logueamos con nuestra cuenta de google, seleccionamos nuestro documento de Google Sheet (una hoja con las columnas email, fecha, nombre), la hoja que queremos usar y en la opción “Column to match on” elegimos “email”.

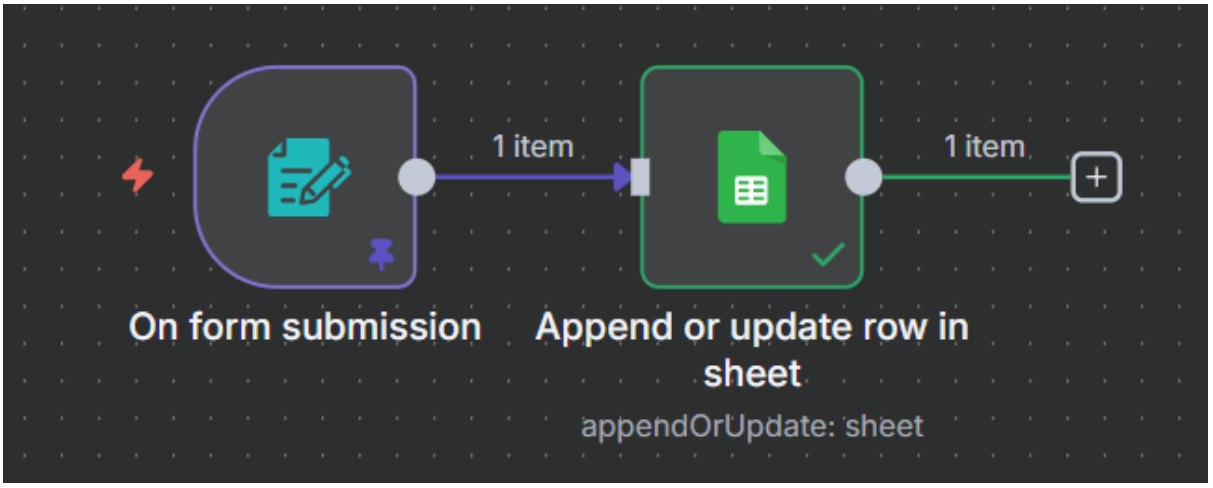
Esto permitirá usar el correo electrónico como una clave única, actualizando la fila si el email ya existe. De esta manera se evitan duplicados fácilmente.

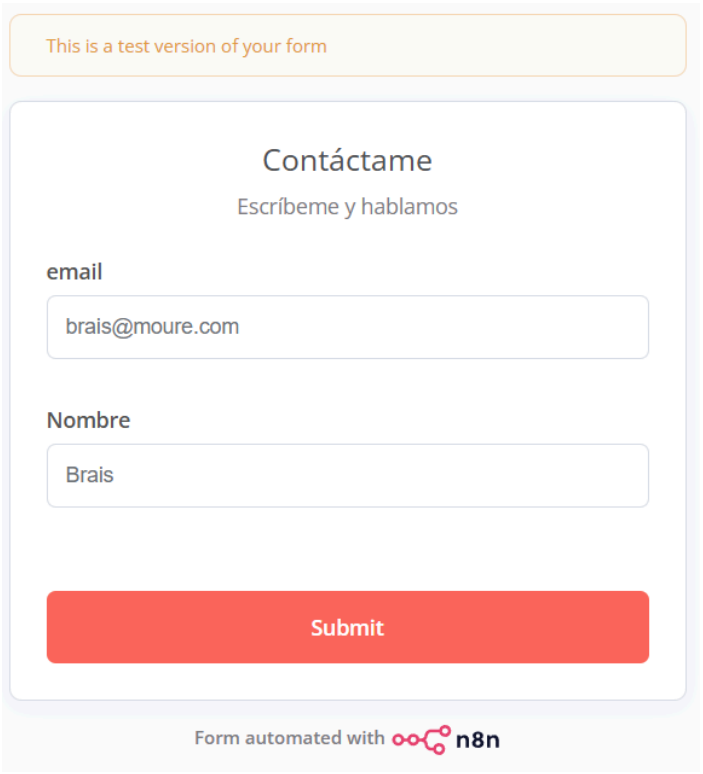
En “Values to Send” arrastramos los campos de nuestro INPUT de la izquierda a cada uno de los campos que queremos completar en el google sheet.

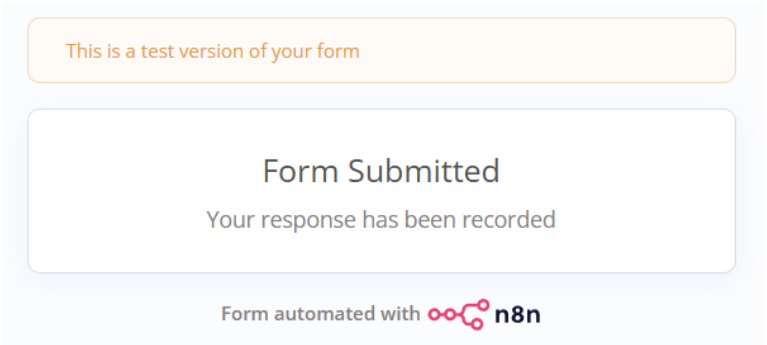


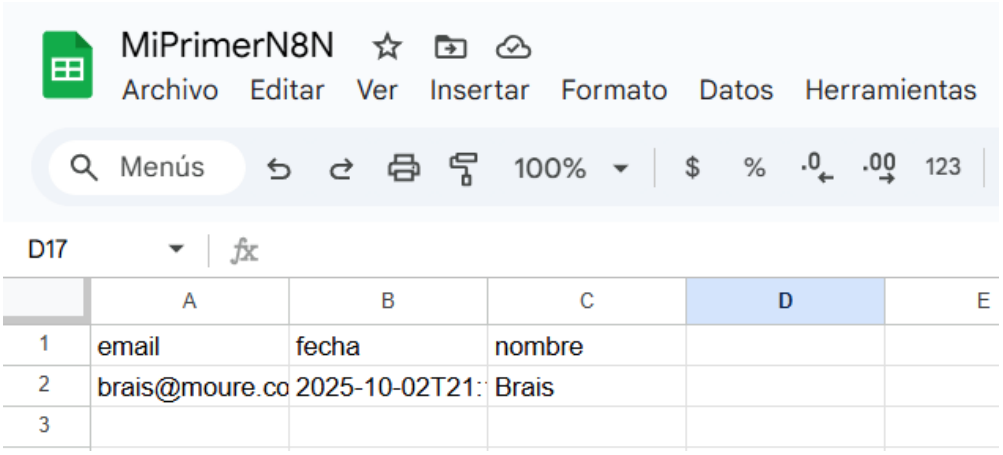

3. **El Resultado**

   Al ejecutar el workflow, los datos enviados desde el formulario (nombre, email y fecha de envío) se añaden o actualizan automáticamente en la hoja de cálculo seleccionada, creando una “base de datos” de contactos sin intervención manual.

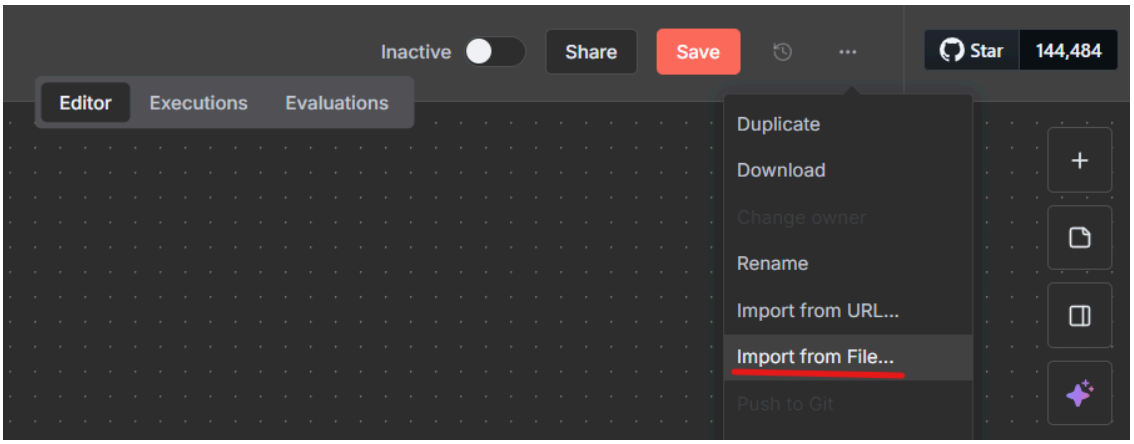

1 item
1 item
On form submission
Append or update row in sheet
appendOrUpdate: sheet


This is a test version of your form
Contáctame
Escríbeme y hablamos
email
brais@moure.com
Nombre
Brais
Submit
Form automated with n8n


This is a test version of your form
Form Submitted
Your response has been recorded
Form automated with n8n



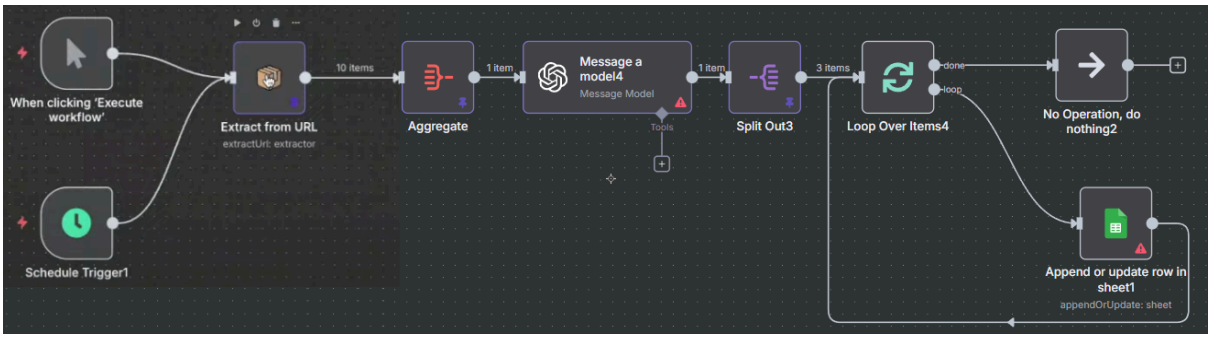

Puedes descargar el archivo JSON aquí e importar el workflow a tu espacio de trabajo.




**Recuerda que debes loguearte con tu cuenta de google en los parámetros del nodo de Google Sheets (opción "Credential to connect with"), y tener el documento de Google Sheets creado en Google Drive para que funcione correctamente.**

# Caso práctico #2: Web Scraping con IA

**Objetivo del Workflow:** Monitorear la web de MoureDev para detectar automáticamente cuándo se publican nuevos cursos de Python.




**Componentes Clave del Flujo con IA:**

- **Trigger Programado (Schedule)**: En lugar de un trigger manual, se utiliza un nodo de programación (schedule). Esto permite que el flujo se ejecute automáticamente, por ejemplo, "todos los días a primera hora".
- **Nodo ScrapeAI**: Este potente nodo permite extraer información de una página web (scraping) utilizando un prompt de lenguaje natural, sin necesidad de programar un scraper tradicional. La IA extraerá el título, la descripción y la URL de los cursos, especificando inteligentemente que la URL está en el botón “Ver el curso". Esto demuestra que la IA entiende el contexto de la página.
- **Nodo Aggregate**: toma elementos separados, o partes de ellos, para agruparlos en elementos individuales.
- **Nodo ChatGPT**: Una vez extraída la información de todos los cursos, se envía a ChatGPT para ser procesada. La instrucción fue clara: "recibirás una Array en formato JSON... y me vas a seleccionar solo los que tengan Python". ChatGPT actúa como un filtro inteligente que procesa los datos brutos y devuelve únicamente la información relevante.

- **Nodo Split Out:** Separa un único elemento de datos que contiene una lista en varios elementos.
- **Nodo Loop Over Item**: recorre todos los elementos.
- **Nodo Google Sheets**: Guarda cada elemento en el documento de Google Sheets seleccionado.

Este flujo podría extenderse fácilmente para, por ejemplo, enviar una notificación por email cada vez que se detecte un nuevo curso.

Puedes descargar el archivo JSON aquí e importar el workflow a tu espacio de trabajo. En los nodos podrás ver en detalle los prompts utilizados y configuraciones necesarias de cada uno de ellos.

# Una herramienta clave para el desarrollo: La Función "Pin"

"**Pinear**" o fijar los datos en un nodo es una funcionalidad extremadamente útil durante el desarrollo.

Consiste en guardar los datos de salida de la ejecución de un nodo. Una vez "pineados", esos datos se reutilizarán en ejecuciones posteriores sin necesidad de volver a ejecutar ese nodo ni los anteriores.

Esta función simplifica radicalmente el proceso de desarrollo y depuración.

Imagina un flujo de trabajo larguísimo, de hasta 500 nodos. Si necesitas probar un cambio en el nodo 499, no tienes que volver a ejecutar los 498 nodos anteriores. Simplemente "pineas" la salida del nodo 498 y puedes probar tus cambios en el siguiente con un solo clic.

Esto ahorra una cantidad enorme de tiempo, reduce el consumo de tokens de API y simplifica enormemente la depuración de partes específicas de tu automatización.

# Conclusiones

Herramientas como **n8n**, combinadas con el poder de la **IA**, han democratizado la automatización.

Permiten a los desarrolladores, desde juniors hasta seniors, automatizar procesos complejos, obtener datos de formas que antes requerían un gran esfuerzo y, lo más importante, centrarse en construir aplicaciones que aporten valor.

**El límite lo pone tu imaginación. Así que, es hora de "inventar, cacharrear y darle duro".**

BIG school

# Conviértete en Desarrollador con IA

MATRÍCULATE AQUÍ