GUÍA DE NOTEBOOKLM PARA DEVELOPERS

Prepara el contexto antes de programar con IA

BIG school

Usa NotebookLM para entender funcionalidades, detectar lo que falta y llegar a tu agente con mejores instrucciones.

Una aclaración antes de empezar

Si entras en NotebookLM y ves el nombre **Gemini Notebook**, no te has equivocado de herramienta. Google cambió su nombre en julio de 2026 y ahora está más integrado dentro del ecosistema de Gemini.

En esta guía voy a seguir llamándolo NotebookLM, que seguramente es como todavía lo conoces y así tampoco nos complicamos más de la cuenta.

Dicho esto, vamos a lo importante: cómo puede ayudarte NotebookLM antes de empezar a desarrollar con IA.

¿Qué vas a conseguir con esta guía?

Trabajar con IA te permite avanzar muchísimo más rápido a la hora de desarrollar software. El problema es que esa velocidad también puede jugar en tu contra si empiezas a generar código antes de entender realmente qué quieres construir.

Imagina que te llega una tarea aparentemente sencilla: añadir recuperación de contraseña a una aplicación.

Ahora imagina que abres tu agente de desarrollo y escribes algo como:

"Implementa un sistema de recuperación de contraseña."

¿Podría hacerlo? Seguramente.

El problema es que todavía quedan bastantes decisiones abiertas.
Antes deberías responder preguntas como estas:

¿Cuánto tiempo debe ser válido el enlace?

¿Puede utilizarse más de una vez?

¿Qué ocurre si el correo no existe?

¿Hay requisitos específicos de seguridad?

¿Cómo funciona actualmente la autenticación?

¿Existe ya algún servicio encargado de enviar emails?

Parece obvio: si tú no tienes claras esas respuestas, tampoco puedes esperar que una IA tome siempre las decisiones correctas por ti.

Y aquí es donde NotebookLM te puede resultar especialmente útil.

Lo importante es que tengas esto claro: no vas a utilizar NotebookLM para programar ni necesitas aprender todas sus funciones.

En esta guía quiero enseñarte un único flujo de trabajo:

Utilizar NotebookLM para reunir el contexto de una tarea, entender qué sabes, detectar qué te falta y llegar mucho mejor preparado al momento de desarrollar con IA.

¿Dónde encaja NotebookLM dentro del proceso de desarrollo con IA?

Si utilizas IA para desarrollar, no necesitas convertir NotebookLM en otro editor de código ni en otro agente. Yo lo colocaría un paso antes: en el momento de entender qué quieres construir y qué contexto necesita la IA.

Entender el contexto: Qué tienes que construir, cómo funciona el proyecto y qué información necesitas antes de tocar nada.

Detectar lo que falta: Como restricciones, contradicciones, casos que todavía no están definidos o decisiones que deberías aclarar antes de desarrollar.

Preparar mejores instrucciones: Ordenar ese contexto para que, cuando pases la tarea a tu agente, no tenga que rellenar los huecos por su cuenta.

El flujo

Fíjate en el orden: no estamos utilizando NotebookLM para hacer el trabajo del agente, sino para llegar al agente con el trabajo mucho mejor definido.

No se trata de utilizar la IA para todo, sino de saber qué herramienta utilizar en cada parte del proceso.

La IA puede encargarse de ejecutar el trabajo Revisar lo que ha hecho y decidir si está bien sigue siendo cosa tuya.

1. Crea un Notebook para el problema que quieres resolver

Lo primero es crear un nuevo *Notebook*. Podrías llamarlo simplemente "Proyecto ecommerce", "Programación" o "Documentación". Estarías empezando a meter ruido desde el primer momento.

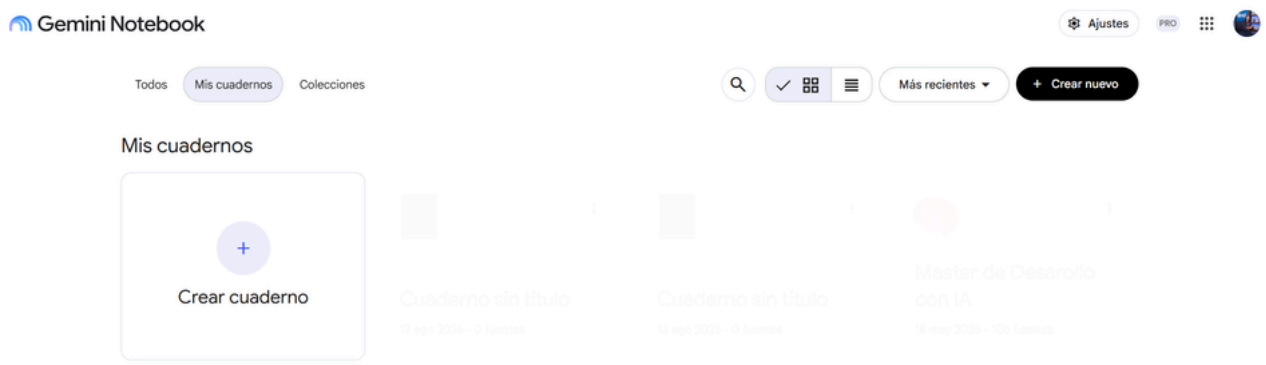
Si lo que quieres resolver es una **funcionalidad concreta**, centra también el Notebook en ella. Por ejemplo:

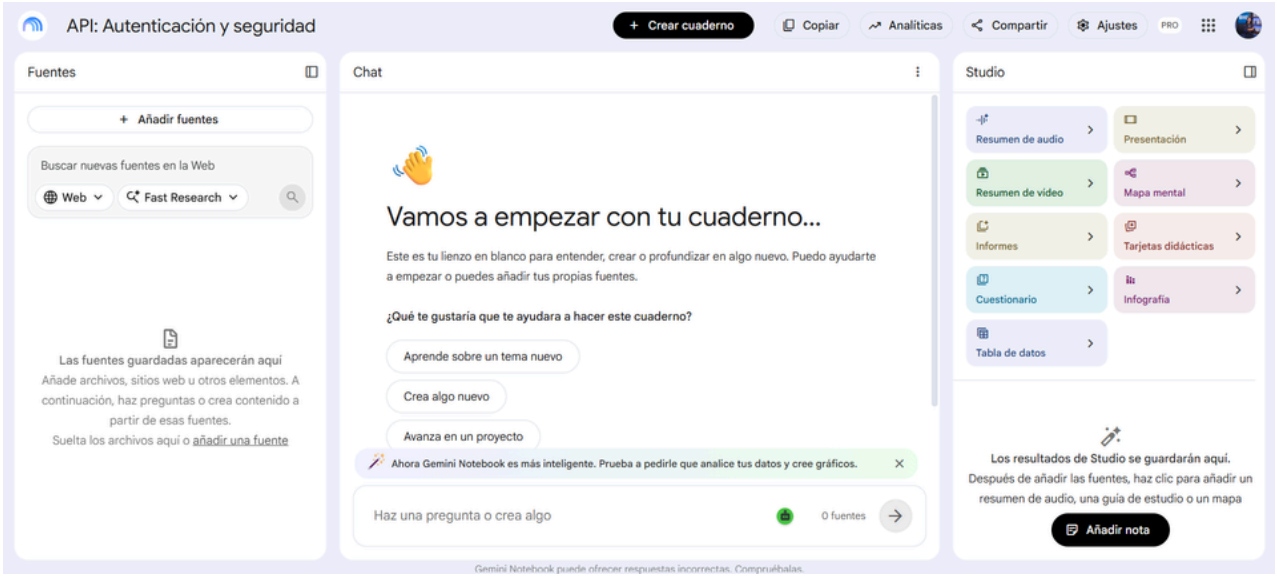
"Proyecto — Recuperación de contraseña"

Piensa en el Notebook como un espacio de investigación para responder una pregunta concreta. Cuanto mejor acotes esa pregunta, más sencillo será decidir qué información necesitas introducir después.

En este ejemplo, la idea es entender qué debe hacer la recuperación de contraseña, cómo encaja en la aplicación actual y qué condiciones tienes que respetar antes de implementarla.

No necesitas conocer todo el proyecto para empezar. Necesitas conocer la parte que afecta al problema que quieres resolver.





2. No metas más contexto. Mete mejor contexto

Una vez creado el Notebook aparece la siguiente duda lógica:

> *"Vale, ¿y ahora qué meto aquí?"*

NotebookLM permite trabajar con diferentes tipos de documentos y fuentes. Hacer una colección enorme de archivos no es el objetivo. Si metes treinta documentos "por si acaso", probablemente acabes trasladando a NotebookLM el mismo caos que ya tenías repartido entre carpetas y pestañas.

Para este ejemplo, podríamos pensar en empezar con cinco fuentes.

1. La especificación de la funcionalidad, para saber qué comportamiento se espera y qué requisitos existen.

2. El README o la documentación general del proyecto, para entender tecnologías, estructura y configuración.

3. La documentación de arquitectura o las decisiones técnicas, para conocer dónde encaja la funcionalidad y qué patrones debes respetar.

4. La documentación oficial del servicio de autenticación, si el proyecto utiliza uno, para conocer sus capacidades y limitaciones reales.

5. Los requisitos de seguridad, si existen, para no dejar fuera condiciones importantes durante la implementación.

Fíjate en que cada fuente tiene una función. No estás añadiendo documentos porque NotebookLM pueda leerlos, sino porque contienen información que necesitas para tomar una decisión.

Esta es una regla que también puedes llevarte a cualquier otra herramienta de IA: **Más contexto no siempre significa mejor contexto.**

Antes de añadir una fuente, pregúntate qué necesitas entender y si ese documento puede ayudarte realmente a responderlo.

3. Haz estas 5 preguntas antes de empezar a programar

Ya tienes la información reunida. Ahora sí puedes empezar a trabajar con ella. Y no, la primera pregunta no va a ser “Resúmeme todo”.

Eso te puede dar una visión general, aún así la idea es sacar algo más útil: información que después te permita desarrollar con criterio.

Pregunta 1. ¿Qué tenemos que construir exactamente?

Utiliza una instrucción como esta:

> *"Basándote únicamente en las fuentes de este Notebook, explícame qué comportamiento debe tener esta funcionalidad para considerarse terminada. Indica qué fuente respalda cada requisito importante."*

Con esto buscas transformar documentación que puede estar repartida en diferentes lugares en una visión bastante más clara del objetivo.

Todavía no estás decidiendo cómo programarlo. Primero necesitas asegurarte de que entiendes qué debe ocurrir.

Pregunta 2. ¿Qué restricciones debes respetar?

Ahora puedes preguntar:

> *"Identifica las restricciones técnicas, de arquitectura, seguridad, compatibilidad o negocio que podrían condicionar la implementación de esta funcionalidad. Indica de qué fuente sale cada una."*

Esta pregunta es importante porque una solución puede funcionar y seguir siendo una mala solución para tu proyecto.

Quizás en vuestro proyecto ya tenéis un servicio encargado de enviar notificaciones. Puede que toda la autenticación dependa de un proveedor externo o que en vuestro equipo tengáis una convención de arquitectura que impida acceder a la base de datos desde cierto módulo.

Si no conoces esas decisiones, **un agente puede acabar construyendo algo técnicamente válido que no encaja con el resto del sistema**.

Pregunta 3. ¿Dónde encaja dentro del proyecto?

Vamos bajando al terreno real:

> *"Según la arquitectura y la documentación disponibles, ¿qué componentes del sistema podrían verse afectados por esta funcionalidad? Señala también qué decisiones técnicas existentes deberíamos respetar."*

Aquí NotebookLM te puede ayudar a conectar información que aparece en diferentes documentos.

No le estás pidiendo que decida la arquitectura por ti. Lo que buscas es comprender mejor la que ya existe antes de empezar a modificarla.

Pregunta 4. ¿Qué te falta por saber?

Esta es probablemente una de las preguntas más importantes de todo el proceso:

> *"Busca información que todavía no esté definida: ambigüedades, contradicciones entre fuentes o decisiones que deberíamos aclarar antes de comenzar a implementar. No inventes una respuesta cuando las fuentes no permitan resolverla."*

Muchas veces utilizamos la IA únicamente para buscar respuestas cuando también puede ayudarnos a descubrir preguntas que todavía no habíamos planteado.

Imagina que una especificación dice que el enlace de recuperación caduca a los 30 minutos, mientras que otro documento habla de 15. O simplemente no existe ninguna indicación sobre qué debe ocurrir con las sesiones abiertas después de cambiar la contraseña.

Encontrar esa duda antes de escribir código es bastante más útil que descubrirla cuando ya tienes media funcionalidad implementada.

Pregunta 5. ¿Cómo sabes que está terminado?

Por último:

> *"Extrae de las fuentes los criterios de aceptación, comportamientos esperados y casos límite que deberíamos comprobar antes de considerar terminada la funcionalidad."*

Esto te obliga a definir qué significa realmente "funciona".
Después puedes utilizar esos mismos criterios para revisar el trabajo de tu agente y comprobar que no solo ha generado código, sino que ha resuelto el problema que le habías encargado.

4. No te quedes con la respuesta y comprueba de dónde sale

Cuando trabajas con documentación técnica: las respuestas pueden llevarte de vuelta a las fuentes utilizadas.
Imagina que después de estas preguntas obtienes esta conclusión:

"El enlace de recuperación debe caducar después de 30 minutos."

Si ese dato va a condicionar el desarrollo, yo no me quedaría simplemente con la respuesta. Abriría la cita y comprobaría el documento original.

Puede que sean realmente 30 minutos. También puede ocurrir que la documentación del proveedor recomiende 30, mientras que vuestro proyecto haya definido 15. O que esa cifra corresponda a otro tipo de token.

NotebookLM te puede ahorrar bastante tiempo encontrando la información. Eso no significa que debas dejar de validarla.
No hace falta comprobar absolutamente cada frase; sí aquellas que afectan a decisiones importantes de arquitectura, seguridad, compatibilidad, configuración o comportamiento.

Quédate con este proceso:

1. Pregunta → 2. Localiza → 3. Comprueba → 4. Decide

La herramienta te ayuda con las tres primeras partes. La decisión sigue siendo tuya.

5. Ahora sí: prepara el contexto para tu agente

A estas alturas ya deberías tener bastante más claro qué quieres construir, qué partes del proyecto están implicadas, qué restricciones existen, qué casos debes contemplar y qué dudas todavía necesitan una respuesta. Ahora tiene mucho más sentido pasar a un agente de desarrollo.

Antes puedes utilizar NotebookLM una última vez para ordenar toda esa información:

Prompt Final

"Basándote únicamente en las fuentes que hemos revisado, prepara una ficha de contexto para implementar esta funcionalidad con un agente de desarrollo.

Incluye:

- *Objetivo de la funcionalidad;*
- *Comportamiento esperado;*
- *Componentes del proyecto que podrían verse afectados;*
- *Restricciones técnicas y de arquitectura;*
- *Decisiones existentes que debemos respetar;*
- *Requisitos de seguridad, si existen;*
- *Casos límite;*
- *Criterios de aceptación;*
- *Cuestiones que todavía no estén resueltas.*

No escribas código. No inventes información que no aparezca en las fuentes y señala cualquier punto que necesitemos aclarar antes de desarrollar."

Este resultado no sustituye la documentación del proyecto ni debería convertirse automáticamente en una especificación definitiva. Es una forma de **reunir el contexto que acabas de investigar para tenerlo más accesible cuando empieces a trabajar con tu agente**.

Y aquí está la diferencia importante.

Si al principio hubieras escrito simplemente:

"Implementa un sistema de recuperación de contraseña."

Estarías dejando una cantidad enorme de decisiones en manos de la IA.
Ahora puedes llegar al agente sabiendo qué quieres conseguir, qué reglas debe respetar, qué casos tiene que cubrir y qué cosas todavía no estás preparado para delegar.

Quédate con esta idea final

NotebookLM tiene muchas más funciones y puedes utilizarlo para muchísimas otras cosas. No necesitas conocer toda la herramienta para empezar a sacarle partido como desarrollador.

Para este flujo basta con algo mucho más sencillo:

Crear un Notebook para un problema concreto, añadir únicamente las fuentes que necesitas, hacer buenas preguntas y comprobar la información importante antes de utilizarla.

La idea no es que NotebookLM piense por ti.

La idea es que llegues al desarrollo con un contexto mucho mejor preparado para que, cuando llegue el momento de delegar trabajo a una IA, sigas siendo tú quien entiende el problema y decide qué resultado da por bueno.

Porque generar código cada vez es más fácil.

Ahora bien, saber qué código necesitas generar y por qué sigue siendo tu trabajo.

Ahora llévate esta forma de trabajar al desarrollo con IA

Este proceso conecta directamente con lo que vamos a trabajar en el **Curso de Desarrollo con IA: El Nuevo Programador**.

En la primera jornada veremos precisamente cómo pasar de una idea o una necesidad a unas instrucciones que la IA pueda ejecutar, entendiendo **qué contexto necesita, qué límites debemos establecer y cómo evaluar su propuesta** antes de empezar a generar código.

ACTIVA LA CAMPANITA EN YOUTUBE

PARA QUE TE AVISE CUANDO COMIENCE LA CLASE #1 DEL CURSO.

ACCEDE AQUÍ A LA PRIMERA CLASE

BIG school