LOCAL LLM CODER

IA que programa en tu máquina. Y comprueba lo que entrega.

Asistente de programación que corre entero en tu máquina. Compila lo que escribe, lo ejecuta y te dice qué comprobó y qué no.

El código no sale de tu red · Se enchufa a VS Code · Sin suscripción por cabeza

EL PROBLEMA

La IA escribe el código.
Los fallos los encuentras tú.

Compila ≠ funciona

Un programa puede compilar, ejecutarse, terminar con código 0 y no haber hecho nada de lo que se le pidió. Para cualquier comprobación normal, eso es un éxito.

El código sale de casa

Cada pregunta a una IA de nube manda tu código —o el de tu cliente, bajo NDA— a un servidor ajeno.

Se paga por cabeza

Contratas tres programadores y la factura sube. Todos los meses, para siempre.

QUÉ ES

Una plataforma de IA para equipos de desarrollo, dentro de su máquina.

No sale de tu red

Ni a OpenAI, ni a Anthropic, ni a ningún servidor. Los proveedores de nube existen como opción y vienen apagados de serie.

Comprueba el trabajo

Compila con el compilador de verdad, comprueba que los archivos encajan y ejecuta el programa. Si falla, lo arregla y lo vuelve a comprobar.

Se enchufa a tu editor

API compatible con OpenAI. Funciona con Continue en VS Code apuntando a tu máquina, con la configuración lista para pegar.

LA DIFERENCIA

Qué pasa antes de que te llegue nada

01

Sintaxis de verdad

Cada archivo por el compilador real del lenguaje. No es el modelo opinando.

02

Compilación conjunta

En Java, C# y Kotlin los archivos comparten espacio de nombres. Se compilan juntos.

03

Encaje

Que lo que un archivo usa de otro exista de verdad: el script, el id, la firma.

04

Ejecución en jaula

Se ejecuta de verdad, con la red cortada. Doce lenguajes. El informe trae lo que imprimió.

05

Reparación

Lo que falla vuelve al modelo con el diagnóstico exacto, y se comprueba otra vez.

Y el informe dice qué NO se pudo comprobar. Si en esa máquina no está el compilador de Go, lo dice, en vez de dar por bueno lo que nadie ha mirado.

CASOS REALES

Lo que un modelo hace cuando no sabe hacer el trabajo: fingirlo

Lo que entregó el modelo	Qué hizo la plataforma
Una copia falsa de una clase «para que compile», con los números a mano	No compila junto → se lo devuelve
Un único archivo llamado ruta/del/archivo.csv, con Java dentro	Rechaza el plan antes de escribir nada
Caza su excepción, imprime «Error» y termina con código 0	Detecta que lo único que escribió es un fallo
Abre datos.csv y no lo incluye en la entrega	Le pide el archivo con datos de ejemplo
Un juego que pinta y no escucha el teclado	Lo señala: no cuenta como terminado

Cada fila es una entrega real, no un supuesto

MEDIDO, NO PROMETIDO

El mismo encargo, cinco veces

Programa en Java que lee un CSV y suma por mes

5/5
compilan, ejecutan y dan resultado

60 s
mediana por encargo

Un Tetris jugable con el teclado, en una web

5/5
arrancan y pintan

2
reparaciones automáticas en total

Las cuatro entregas anteriores del mismo encargo, antes de estas comprobaciones, fallaron las cuatro. Cada una por un motivo distinto.

EJEMPLO · SOBRE EL CÓDIGO DEL CLIENTE

Pregúntale por el repositorio. Contesta con el archivo y la línea.

```
> ¿Dónde se valida el token de sesión?

El token se valida en validar_token, en auth.py: comprueba la firma y la caducidad, y devuelve el usuario
o None.

[2] demostracion/src/auth.py:L24-39

La llama servidor.py antes de ejecutar cualquier ruta privada.

[1] demostracion/src/servidor.py:L35-45
```

No dice «en algún sitio de auth». Da el archivo y la línea, con una **regla explícita para no inventárselos** — y un clic abre ese archivo en el visor. Detrás: tabla de símbolos y grafo de llamadas de todo el repositorio, 60+ lenguajes, calculados sin gastar un token.

EJEMPLO · EL QUE ENTRA NUEVO

El onboarding se genera solo, desde el código de verdad

La lección que más vale —**cómo viaja una petición por dentro**— sale del grafo de llamadas. Cada paso es una llamada que existe, con su archivo:

```
1. Manejador.do_GET · servidor.py:49
2. despachar · servidor.py
3. validar_token · auth.py
4. usuario_por_id · db.py
5. conexion · db.py
```

Seis lecciones derivadas del repositorio, no un curso genérico: qué es, el mapa, el recorrido, las convenciones, la deuda y lo que nadie documentó.

No se le pide al modelo. El esqueleto sale de hechos; la prosa va marcada como borrador y la aprueba un sénior.

Cada lección abre un tutor con el repositorio delante.

```
Dos semanas de un sénior por cada persona que entra → media hora de revisión.
```

EJEMPLO · NO TE FÍES DEL MODELO, NI DE NOSOTROS

La herramienta se mide a sí misma, y no le echa la culpa al modelo

Dice en qué se va el tiempo

Una espera de «211 s» no dice nada. La plataforma la reparte: cola, cargar el modelo, leer el contexto. Con ese dato a la vista, un ajuste llevó la lectura de **44 a 108 tokens/s** en la misma máquina.

No acusa de lo que no pasó

Un modelo de nube que no llegó a contestar salía con «0/100, se inventa 4/4»: la peor nota, por un fallo de conexión. Ahora sale como **error con su motivo**, no como nota.

Un informe que acusa tiene que ser más fiable que uno que elogia, porque nadie comprueba una acusación que suena técnica. **62 bancos de pruebas**, y cada uno rompe el producto a propósito para confirmar que salta.

EN EL SITIO DONDE YA TRABAJAN

Se enchufa a VS Code

API compatible con OpenAI: /v1/chat/completions con streaming y /v1/models.

Funciona con **Continue en VS Code**, con Zed y con cualquier herramienta que hable ese protocolo.

La configuración se genera **lista para pegar**, con el anfitrión correcto para entrar desde otro ordenador.

Una clave por aplicación, con su cuota y su ámbito: una clave para un repositorio no ve los demás.

```
apiBase: http://192.168.1.40:8100/v1
apiKey: cdr_9yjk…
model: qwen3-coder:30b

# y si la clave no vale, el error dice
# POR QUÉ: si es del ejemplo, si es de
# OpenAI, si no hay ninguna, o si está
# revocada. Cuatro causas, cuatro
# respuestas distintas.
```

PARA UN EQUIPO, NO PARA UNA PERSONA

Lo que hace falta para instalarlo en una empresa de verdad

Gestión

Cuentas con roles • Cuotas por desarrollador • Registro de auditoría • Permisos por carpeta • Panel de uso

Trabajo

Proyectos con contexto compartido • Espacio de trabajo con historial y vuelta atrás • Cuaderno de decisiones • Glosario corporativo

Salida

PDF, Word y Excel • SARIF, CSV y HTML • .zip de código con avisos • Push a Jira y GitHub Issues

Mantenimiento

Copias diarias • Actualización remota con un botón • Telemetría de soporte opcional • Licencia por empresa

Solo Python 3.10+. Ni un pip install. Servidor en macOS, Linux o Windows; los desarrolladores entran desde cualquier navegador, también el móvil. Y marca blanca: nombre, logo, colores y modelo se cambian en un archivo.

PRIVACIDAD, DICHA CON PRECISIÓN

Qué sale de la máquina: nada

El código se queda

No viaja a OpenAI, ni a Anthropic, ni a ningún servidor. El modelo corre en la máquina que controla el cliente.

La nube, avisada

Los proveedores externos vienen apagados. Si el admin enciende uno, la plataforma lo dice en pantalla y marca esas conversaciones. También los modelos «cloud» de Ollama, que parecen locales y no lo son.

Caza de secretos

Doce patrones de alta confianza —claves de AWS, tokens de GitHub, claves de OpenAI, de Slack— para que una clave pegada en el chat no llegue al modelo.

No entrena con tu código

No hay ajuste fino. Hay búsqueda sobre tu repositorio, en tu máquina. Tu código no mejora el modelo de nadie.

Y la plataforma nunca escribe en el repositorio del cliente. Git entra en solo lectura. Antes de descomprimir una entrega encima, un CAMBIOS.diff dice exactamente qué machacaría.

DÓNDE ENCAJA

Frente a lo que ya existe

	Local LLM Coder	Tabnine	GitHub Copilot	Ollama + Continue
Corre sin internet	Sí	Sí (Enterprise)	No	Sí
Compila lo que entrega	Sí	No	No	No
Lo ejecuta y lo mide	Sí	No	No	No
Repara solo lo que falla	Sí	No	No	No
Multiusuario, cuotas, auditoría	Sí	Sí	Sí	No
Precio	por empresa	39–59 $/usuario/mes	19–39 $/puesto/mes	gratis
Quién lo mantiene	nosotros	ellos	ellos	tú

Precios públicos de noviembre de 2026

ACADEMIAS Y CENTROS DE FP · DAM / DAW

El alumno usa IA.
El profesor no ve el proceso.

El diario es la evidencia

Cada encargo deja qué planificó, qué decidió que nadie le dijo, qué comprobación falló y cómo se arregló. El profesor no evalúa un resultado pegado: evalúa el camino. Material para FP Dual y FCT.

Un curso de su propio repo

La ruta de entrada generada es un temario derivado del código con el que trabaja la clase, con progreso por alumno y tutor sobre el código real.

Cabe en un examen

Una IA de nube es un canal abierto al exterior y no cabe en una prueba. Ésta corre en el aula, con auditoría y cuotas, y una licencia por aula — no treinta suscripciones.

Un ciclo de DAM o DAW cuesta al alumno entre 5.900 € y 8.500 €.
Una licencia se paga con un alumno más matriculado. Uno.

LO QUE NO HACE

Un producto que solo cuenta virtudes no se cree. Esto es lo que no hace.

No sustituye al programador. Trabaja con él. El prompt lo dice literalmente.

Que compile no quiere decir que haga lo que se pidió. El informe lo dice con esas palabras: si las cifras son correctas, eso lo mira una persona.

La jaula de Linux no encierra el disco. En macOS sí. Está escrito en la documentación, con el nombre del mecanismo y su límite.

La calidad depende del modelo instalado. La plataforma comprueba y repara; no convierte un modelo pequeño en uno grande.

LO QUE HACE FALTA

Memoria

16 GB mínimo • **32 GB para programar** • 64 GB para un modelo grande

Sistema

macOS, Linux o Windows
Python 3.10+ y Ollama

Instalación

Ni un pip install
Arranca sola al encender

Red

Ninguna para funcionar
LAN para que entren los demás

Pruébalo con tu repositorio.

Media hora, en tu máquina, con tu código. Si no te convence el informe que te entrega, no hay más conversación.

+34 652 947 524 · github.com/ericmugnoz