GUÍA GRATIS · IA + PRODUCTO

Guía de Vibecoding

Mejores prácticas para construir con IA sin que todo se rompa a la mitad.

Por Benjamin Ramos

Product · AI · Builder
Basado en Costa Rica

Antes de empezar

El vibecoding es programar de la mano de la inteligencia artificial. Le decís qué querés y la IA escribe el código por vos. Suena mágico y en parte lo es. En cinco minutos podés tener una página que se ve bien.

El problema aparece después.

Le pedís un ajuste. Después otro. Después una cosa nueva. Y de repente el proyecto es un nudo que nadie entiende, que se rompe cada vez que lo tocás, y que ni la propia IA puede arreglar sin dañar otra cosa. A eso le decimos espagueti: código enredado donde todo depende de todo y nada se puede cambiar en paz.

La buena noticia es que esto no pasa por usar IA. Pasa por usarla sin método. Acá te muestro el método que yo uso para construir web apps de verdad sin terminar con ese nudo.

La idea de fondo es simple. La IA es como un desarrollador junior brillante y rapidísimo, pero sin criterio propio. Va a hacer exactamente lo que le pidás, aunque lo que le pidás sea una mala idea. El criterio lo tenés que poner vos. Este documento es sobre cómo ponerlo.

Los tres errores que garantizan espagueti

Antes del método, mirá los tres errores que veo casi siempre. Si evitás solo estos tres, ya vas ganando.

Error 1: pedir toda la app de un solo tiro.

"Hacéme una plataforma que haga esto, esto y esto." La IA lo intenta, arma algo enorme de una vez, y vos no entendés ni la mitad de lo que quedó. Cuando algo falle, no vas a saber dónde tocar.

Error 2: decirle que sí a todo.

La IA propone algo, vos decís "sí", propone otra cosa, "sí" otra vez. Sin darte cuenta le entregaste todas las decisiones a algo que no conoce tu objetivo. Vos sos quien manda.

Error 3: no revisar lo que genera.

Aceptás el código sin leerlo ni probarlo, seguís adelante, y acumulás cinco features encima de una base que quizás ya estaba mal desde la segunda.

Los tres tienen la misma raíz. Ir rápido sin plan. El método que sigue ataca justo eso.

El método, paso a paso

01 Planeá antes de tocar una línea de código

Esta es la parte que casi nadie hace y la que más rinde. Antes de abrirle la boca a la IA, escribí en simple qué vas a construir.

No necesitás un documento de cien páginas. Necesitás responder tres cosas:

- **Qué es.** En dos o tres líneas, qué hace esta app o web y qué problema resuelve.
- **Para quién es.** Quién la va a usar y qué espera lograr.
- **Qué NO es.** El límite. Lo que a propósito dejás por fuera de esta primera versión.

Ese último punto es el más subestimado. Definir lo que no vas a hacer es lo que evita que el proyecto crezca sin control.

> **El porqué:** cuando vos tenés claro el objetivo, la IA deja de adivinar. Cada decisión que tome se puede medir contra ese objetivo. Sin plan, cada prompt es una improvisación y las improvisaciones se acumulan hasta volverse espagueti.

02 Armá el contexto antes de pedir features

La IA es tan buena como el contexto que le das. Si arrancás en frío, va a inventar sus propios criterios y cada vez van a ser distintos.

Antes de construir, dejale por escrito las reglas del juego. Yo armo una guía de diseño y una de contexto técnico. Algo así:

- **Guía de diseño.** Colores, tipografías, espaciados, cómo se ven los botones, las tarjetas, los estados. Esto hace que todo lo que genere se vea consistente en lugar de un Frankenstein de estilos distintos.
- **Contexto técnico.** Con qué stack vas a trabajar, cómo se nombran las cosas, qué convenciones seguir. Reglas claras para que no cambie de criterio a mitad de camino.

Muchas herramientas de vibecoding dejan guardar este contexto en un archivo que la IA lee siempre. Usalo. Es la diferencia entre corregir lo mismo veinte veces y que lo haga bien desde el principio.

> **El porqué:** sin contexto compartido, cada respuesta parte de cero. Con contexto, la IA tiene una memoria de cómo se hacen las cosas en tu proyecto y deja de reinventarlas.

03 Partí el trabajo en épicas e historias

Acá aplico lo que hago en producto todos los días. En lugar de "hacéme la app", parto el proyecto en pedazos manejables.

- **Épicas.** Los bloques grandes. Por ejemplo: registro de usuarios, panel principal, pagos.
- **Historias.** Dentro de cada bloque, las piezas pequeñas y concretas. Por ejemplo, dentro de registro: "un usuario puede crear cuenta con correo y contraseña", "un usuario puede recuperar su contraseña".

La regla es que cada historia sea lo bastante pequeña como para construirla y probarla sola.

> **El porqué:** un pedazo pequeño se entiende, se prueba y se arregla. Un pedazo enorme no. Cuando trabajás por historias, cada avance queda firme antes de pasar al siguiente, y si algo se rompe sabés exactamente dónde buscar.

04 Construí de a una historia a la vez

Con las historias definidas, trabajá una sola por vez. Le pedís a la IA esa historia, nada más. La revisás, la probás, la dejás funcionando, y recién ahí seguís con la próxima.

Sé que es tentador pedir cinco cosas juntas para ir más rápido. Es justo ahí donde empieza el nudo. Cinco cosas juntas es cinco veces más difícil de revisar y de arreglar.

> **El porqué:** cada pieza que dejás funcionando es terreno ganado. Si la próxima se rompe, tenés una base sólida a la cual volver. Ir de a poco no es ir lento, es ir sin retroceder.

05 Pedile criterio, no obediencia

Este es el corazón del asunto y el error más común del que hablo en el video. No le digas que sí a todo. Poné a la IA a pensar con vos.

Antes de construir algo, pedile que critique tu plan. Cosas como:

```
> Decime qué problemas le ves a este enfoque antes de programarlo.
> Dame dos maneras distintas de resolver esto y las ventajas de cada una.
> Qué se podría romper más adelante si lo hago así.
```

Una IA que solo obedece te construye rápido lo que le pidás, aunque sea un error. Una IA a la que le exigís criterio te avisa antes de meterte en el hueco.

> **El porqué:** la IA sabe de patrones y de problemas comunes que quizás vos no viste. Pero solo los saca si se los pedís. Tratala como a un par que puede opinar, no como a una máquina de decir que sí.

06 Revisá y entendé lo que genera

Nunca aceptes código que no entendés. No hace falta que seas experto. Hace falta que sepas, en simple, qué hace cada pieza.

Si algo no lo entendés, no lo dejes pasar. Pedile a la IA que te lo explique como si le hablaras a alguien sin base técnica. Si la explicación no cierra, ahí hay un problema escondido esperando.

> **El porqué:** si no entendés lo que tenés, no lo vas a poder arreglar ni mejorar. Dependés cien por ciento de la IA. Y el objetivo es que vos mandés, no que la herramienta te maneje a vos.

07 Probá cada pieza antes de seguir

Después de cada historia, probála de verdad. Que funcione, que no rompa lo anterior, que haga lo que tenía que hacer.

Probar de a poco parece que te frena. En realidad te ahorra las horas más frustrantes: esas en las que algo falla, no sabés desde cuándo, y tenés que revisar cinco features para encontrar el culpable.

> **El porqué:** un error encontrado al toque se arregla en minutos. El mismo error encontrado cinco pasos después se vuelve una cacería. Probar seguido es lo más barato que vas a hacer.

08 Guardá cada versión que funcione

Cada vez que dejes algo andando, guardalo. La mayoría de estas herramientas tiene control de versiones, o podés usar Git directo. La idea es tener puntos de retorno.

Así, cuando la IA rompa algo y no logres arreglarlo, no perdés todo. Volvés a la última versión que funcionaba y arrancás desde ahí.

> **El porqué:** esta es tu red de seguridad. Es lo que hace que un error sea un tropezón y no un desastre. Sin puntos de retorno, un cambio malo te puede costar todo el trabajo.

Cuando igual se rompe (porque va a pasar)

Aunque hagás todo bien, algo se va a romper. Es parte del proceso. Lo que separa un tropezón de un espagueti es cómo reaccionás.

No parchés sobre el parche. El reflejo es pedirle a la IA que arregle el error de inmediato. Muchas veces ese arreglo rápido tapa el síntoma y crea dos problemas nuevos. Frená.

Primero entendé la causa. Pedile a la IA que diagnostique antes de arreglar: "explicame por qué está fallando esto antes de tocarlo". Entender la raíz vale más que veinte intentos a ciegas.

Si se enredó feo, volvé atrás. Para eso guardaste las versiones. A veces volver al último punto que funcionaba y rehacer con más cuidado es más rápido que desenredar el nudo.

Checklist rápido

Antes de arrancar cualquier proyecto de vibecoding, repasá esto:

- 01 Escribí qué es, para quién es, y qué NO es.
- 02 Armá tu guía de diseño y tu contexto técnico.
- 03 Partí el proyecto en épicas y esas en historias pequeñas.
- 04 Construí de a una historia por vez.
- 05 Pedile a la IA que critique tu plan antes de programarlo.
- 06 Leé y entendé lo que genera. Si no lo entendés, pedí que te lo expliquen.
- 07 Probá cada pieza antes de seguir.
- 08 Guardá cada versión que funcione.

Cierre

El vibecoding no es magia y tampoco es un atajo para saltarse el pensar. Es una forma nueva y poderosa de construir, pero sigue necesitando lo mismo de siempre: un plan, criterio, y orden.

La IA te da la velocidad. El criterio lo ponés vos. Cuando juntás las dos cosas, dejás de pelear con tu propio proyecto y empezás a construir cosas que aguantan, que crecen, y que entendés de punta a punta.

Ahora te toca probarlo. Agarrá una idea pequeña, seguí estos pasos, y fijate la diferencia por vos mismo.

¿Te atascaste? Hablémoslo

Podés seguir esta guía al pie de la letra y aun así toparte con algo que no sabés cómo destrabar. Pasa. Para eso están las Office Hours.

Es una hora 1:1 conmigo. Vos traés tu problema, salís con un plan. Si estás construyendo algo con IA y te quedaste atascado, o querés una segunda opinión antes de invertir tiempo o plata, esta hora es para eso.

Sin teoría de relleno. Compartís pantalla, vamos a tu caso real, y te quedás con un resumen de próximos pasos y la grabación. Los primeros diez minutos son la prueba: si sentís que no te sirvo, cortamos y te devuelvo el 100%. El riesgo lo asumo yo.

Reservá tu hora — $80 `/office-hours`