

PLAYBOOK

Manual de Vibe Coding

Cómo prototipar tu sistema con Lovable, Gemini Canvas, Google AI Studio y Claude Artifacts. Cuándo usar cada herramienta, cómo escribir prompts editoriales y cómo iterar con criterio.

Curso	Marketing Systems Lab
Programa	Aplicaciones de Inteligencia Artificial en Procesos de Mercadeo
Instructor	André Paredes Vega
Institución	Panamerican Business School

Qué es Vibe Coding

Vibe Coding es construir software guiando con intención y juicio en lugar de especificar línea por línea. No es 'codear sin saber qué quieres'; es exactamente lo contrario. Es saber tan bien qué quieres que puedes dirigir al modelo con frases cortas y reconocer al instante cuándo se desvía. La diferencia entre Vibe Coding y prompt-and-pray está en el criterio editorial.

Regla cero. Antes de abrir cualquier herramienta, escribe en una frase qué hipótesis quieres probar con el prototipo. Si no puedes escribirla, no estás listo para prototipar; estás listo para procrastinar.

Cuándo usar cada herramienta

Herramienta	Mejor para	Evitar cuando
Lovable	Apps full-stack con UI, datos y auth. MVPs presentables.	Quieres iterar solo una pieza visual.
Gemini Canvas	Documentos, presentaciones y artefactos visuales rápidos.	Necesitas lógica de backend.
Google AI Studio	Probar modelos, comparar prompts, prototipar APIs.	Necesitas una UI lista para usuarios.
Claude Artifacts	Componentes interactivos, demos de una sola pantalla.	Necesitas personalización o usuarios reales.

Prompts ejecutables vs prompts editoriales

Un **prompt ejecutable** describe qué hacer ('crea un formulario con tres campos'). Un **prompt editorial** describe qué importa ('este formulario debe sentirse íntimo, no transaccional; el primer campo debe ser una pregunta, no un dato'). El primero produce trabajo; el segundo produce diferenciación. Necesitas ambos, pero el orden importa: editorial primero, ejecutable después.

Loop de iteración con criterio

- **1. Hipótesis.** ¿Qué quieres demostrar con esta iteración? Una frase.
- **2. Cambio mínimo.** ¿Cuál es el cambio más pequeño que prueba la hipótesis?
- **3. Generación.** Pide el cambio con un prompt editorial seguido de uno ejecutable.
- **4. Lectura.** Antes de aceptar, léelo como si fueras el usuario, no como autor.
- **5. Decisión.** Quedarse, deshacer o ramificar. Documenta por qué.

Errores frecuentes

- **Generar antes de decidir.** Te quedas con el primer output porque ya lo viste.
- **Iterar sin hipótesis.** Cada cambio se siente productivo pero el sistema no mejora.
- **Confundir herramienta con criterio.** Cambiar de modelo no resuelve un problema de pensamiento.
- **No leer el output como usuario.** Te enamoras del proceso y dejas de ver el resultado.
- **Saltar la documentación.** Si no anotas por qué tomaste una decisión, la vas a repetir.

Checklist antes de mostrar el prototipo

- El camino feliz funciona end-to-end sin intervención manual.
- Hay al menos un detalle editorial visible (microcopy, jerarquía, animación intencional).
- Tienes un plan B si la demo en vivo falla (video, screenshots o flujo alterno).
- Puedes explicar en una frase qué decisión del sistema demuestra el prototipo.
- Hay una pregunta abierta lista para el feedback del público.

Recordatorio final. El prototipo no es el entregable; el aprendizaje sobre el sistema lo es. Un prototipo feo que enseña algo vale más que uno bonito que solo demuestra.