

DAVMAC SOLUTIONS

Guía práctica de desarrollo asistido por IA

VIBE CODE

*Guía completa de desarrollo asistido por inteligencia artificial: de la idea al
producto en producción*

David Maceiras

DAVMAC Solutions

Índice

PARTE I — Fundamentos del Vibe Coding.....	6
Capítulo 1. Qué es el Vibe Coding.....	7
Capítulo 2. Historia y Evolución de la Programación Asistida por IA.....	11
Capítulo 3. El Ecosistema de Herramientas del Vibe Coder.....	13
Capítulo 4. Mentalidad y Flujo de Trabajo.....	15
PARTE II — Del Prompt al Prototipo.....	18
Capítulo 5. Prompts Efectivos: el Arte de Comunicarse con la IA.....	19
Capítulo 6. De la Idea al MVP: Proceso Paso a Paso.....	22
Capítulo 7. Caso Práctico: Aplicación de Gestión de Tareas.....	24
PARTE III — Arquitectura y Buenas Prácticas.....	27
Capítulo 8. Estructurar Proyectos para Trabajar Bien con IA.....	28
Capítulo 9. Patrones de Diseño que Siguen Siendo Relevantes.....	30
Capítulo 10. Testing en la Era del Vibe Coding.....	31
Capítulo 11. Seguridad y Revisión de Código Generado.....	33
Capítulo 12. Revisión de Código con IA como Segunda Opinión.....	34
Capítulo 13. Deuda Técnica en Proyectos Vibe-Coded.....	35
PARTE IV — Automatizaciones e Integraciones.....	36
Capítulo 14. Diseño de Flujos Automatizados.....	37
Capítulo 15. Trabajar con APIs y Webhooks.....	38
Capítulo 16. Scraping y Extracción de Datos.....	39
Capítulo 17. Tareas Programadas y Orquestación.....	40
Capítulo 18. Caso Práctico: Automatización de un Proceso de Negocio.....	41
PARTE V — Aplicaciones Web Full-Stack con IA.....	43
Capítulo 19. Frontend Asistido por IA.....	44
Capítulo 20. Backend y Bases de Datos con IA.....	45
Capítulo 21. Despliegue y Puesta en Producción.....	46
Capítulo 22. Rendimiento y Optimización.....	47
Capítulo 23. Caso Práctico: Aplicación Full-Stack Completa.....	48
PARTE VI — Mantenimiento, Iteración y Escalado.....	50
Capítulo 24. Mantener un Producto Vibe-Coded en el Tiempo.....	51
Capítulo 25. Iteración sin Degradar el Proyecto.....	52
Capítulo 26. Monitorización y Observabilidad.....	53
Capítulo 27. Cuándo Escalar y Cómo Prepararse.....	54
Capítulo 28. Cuándo Incorporar a un Experto Humano.....	55
PARTE VII — Catálogo de Patrones de Prompting.....	57
Capítulo 29. Patrón: Especificación por Ejemplos.....	58
Capítulo 30. Patrón: Restricción Negativa Explícita.....	59

Capítulo 31. Patrón: Contexto de Proyecto Adjunto.....	60
Capítulo 32. Patrón: Rol y Audiencia.....	61
Capítulo 33. Patrón: Iteración Acotada sobre un Fallo.....	62
Capítulo 34. Patrón: Comparación de Alternativas.....	63
Capítulo 35. Patrón: Descomposición de Tarea Compleja.....	64
Capítulo 36. Patrón: Verificación Cruzada.....	65
Capítulo 37. Patrón: Formato de Salida Estructurado.....	66
Capítulo 38. Patrón: Referencia a un Ejemplo Existente en el Proyecto.....	67
Capítulo 39. Patrón: Límite de Alcance Explícito.....	68
Capítulo 40. Patrón: Petición de Casos Límite Explícitos.....	69

PARTE VIII — Catálogo de Casos de Estudio.....70

Capítulo 41. Monitor de Precios de la Competencia.....	71
Capítulo 42. Generador de Leads B2B por Nicho.....	73
Capítulo 43. Herramienta de Reventa: Monitorización de Oportunidades.....	74
Capítulo 44. Vigilancia de Disponibilidad en Alquileres Vacacionales.....	75
Capítulo 45. Chatbot Interno de Soporte de Primer Nivel.....	76
Capítulo 46. Dashboard Interno de Métricas de Ventas.....	77
Capítulo 47. Automatización de Onboarding de Clientes.....	78
Capítulo 48. Sistema de Alertas de Cumplimiento Normativo.....	79
Capítulo 49. Migración de un Sistema Legacy.....	80
Capítulo 50. Aplicación Móvil Híbrida para Gestión de Turnos.....	82
Capítulo 51. Plataforma de Contenido con Generación Asistida.....	83
Capítulo 52. Herramienta Interna de Análisis de Contratos.....	84

PARTE IX — Recetario de Automatizaciones Rápidas para Agencias y Pymes.....85

Capítulo 53. Resumen Automático de Reseñas de Clientes.....	86
Capítulo 54. Generación de Facturas desde Pedidos Confirmados.....	87
Capítulo 55. Alertas de Stock Bajo con Sugerencia de Pedido.....	88
Capítulo 56. Sincronización de Calendario entre Reservas y Google Calendar.....	89
Capítulo 57. Resumen Diario de Actividad para el Equipo.....	90
Capítulo 58. Clasificación Automática de Emails Entrantes.....	91
Capítulo 59. Generación de Contenido para Redes Sociales a partir del Blog.....	92
Capítulo 60. Detección de Duplicados en la Base de Datos de Clientes.....	93
Capítulo 61. Recordatorios Automáticos de Pagos Pendientes.....	94
Capítulo 62. Generación de Informes de Rendimiento de Campañas.....	95
Capítulo 63. Bot de Slack para Consultar Métricas Internas.....	96
Capítulo 64. Validación Automática de Formularios de Solicitud.....	97
Capítulo 65. Sincronización de Inventario entre Tienda Física y Online.....	98
Capítulo 66. Generación Automática de Contratos a partir de Plantillas.....	99
Capítulo 67. Monitor de Menciones de Marca en Redes Sociales.....	100
Capítulo 68. Automatización de Copias de Seguridad con Verificación.....	101
Capítulo 69. Resumen Ejecutivo Automático para Dirección.....	102
Capítulo 70. Alertas de Cambios en Webs de Proveedores.....	103

Capítulo 71. Automatización de Encuestas de Satisfacción Post-Servicio.....	104
Capítulo 72. Consolidación de Gastos para Cierre Contable Mensual.....	105
PARTE X — Preguntas Frecuentes.....	106
Capítulo 73. Preguntas sobre Aprendizaje y Preparación.....	107
Capítulo 74. Preguntas sobre Herramientas.....	108
Capítulo 75. Preguntas sobre Calidad y Confianza.....	109
Capítulo 76. Preguntas sobre el Futuro del Oficio.....	110
PARTE XI — Glosario de Términos.....	111
Capítulo 77. Términos de Vibe Coding y Herramientas.....	112
Capítulo 78. Términos de Desarrollo de Software.....	113
Capítulo 79. Términos de Calidad y Seguridad.....	114
Capítulo 80. Términos de Producto y Negocio.....	115
PARTE XII — Casos Reales de Automatización en Pymes, Paso a Paso.....	116
Capítulo 81. Panadería: Pedido Automático de Materias Primas a Proveedores.....	117
Capítulo 82. Clínica Dental: Reducción de Ausencias con Recordatorios Automáticos.....	119
Capítulo 83. Inmobiliaria: Cualificación y Enrutamiento Automático de Leads.....	121
Capítulo 84. Gestoría: Recopilación Automática de Documentación Trimestral.....	123
Capítulo 85. Tienda de Ropa Online: Automatización del Proceso de Devoluciones.....	125
Capítulo 86. Taller Mecánico: Seguimiento de Reparaciones y Comunicación con el Cliente.....	127
Capítulo 87. Peluquería: Lista de Espera Automática para Huecos Cancelados.....	129
Capítulo 88. Empresa de Reparto Local: Optimización y Comunicación de Rutas.....	131
PARTE XIII — Casos Reales para Autónomos y Freelancers, Paso a Paso....	133
Capítulo 89. Diseñador Gráfico Freelance: Propuestas y Contratos Automáticos.....	134
Capítulo 90. Coach / Consultor: Agenda de Sesiones y Facturación Recurrente.....	136
Capítulo 91. Fotógrafo Freelance: Entrega de Álbumes y Cobro de Anticipos.....	138
Capítulo 92. Traductor Freelance: Presupuestos Automáticos y Seguimiento de Plazos.....	140
Capítulo 93. Desarrollador Freelance: Registro de Horas y Facturación por Proyecto...	142
Capítulo 94. Redactor / Copywriter Freelance: Pipeline de Contenido Multi-Cliente.....	144
Capítulo 95. Community Manager Freelance: Programación Multi-Cliente.....	146
Capítulo 96. Asesor Financiero Autónomo: Seguimiento de Leads y Propuestas.....	148

PARTE I

Fundamentos del Vibe Coding

Antes de escribir una sola línea de código asistido por inteligencia artificial, conviene entender qué es realmente el vibe coding, de dónde viene y por qué ha cambiado la forma en que se construye software. Esta primera parte sienta las bases: la definición del término, su evolución histórica, el ecosistema de herramientas disponible hoy y la mentalidad que separa a quien usa la IA como una calculadora de quien la usa como un colaborador real.

A lo largo de estos cuatro capítulos combinaremos teoría con ejemplos concretos, de forma que al terminar esta parte no solo entiendas los conceptos, sino que puedas reconocerlos la primera vez que te sientes a trabajar con una herramienta de IA.

Capítulo 1. Qué es el Vibe Coding

El término "vibe coding" fue popularizado en 2025 para describir una forma de programar en la que el desarrollador describe en lenguaje natural lo que quiere construir, y un modelo de inteligencia artificial genera, ejecuta y ajusta el código correspondiente, mientras la persona supervisa el resultado y corrige el rumbo mediante nuevas instrucciones. No se trata de escribir código a mano línea por línea, sino de mantener una conversación iterativa con un sistema capaz de escribir, probar y modificar software por sí mismo.

La palabra "vibe" (sensación, ambiente) no es casual: describe un modo de trabajo más fluido e intuitivo que la programación tradicional, donde el desarrollador se mantiene en un estado de flujo alto, avanzando por aproximaciones sucesivas en lugar de planificar cada detalle de antemano. Esto no significa improvisar sin criterio: significa mover el foco de atención de "cómo se escribe cada línea" a "qué debe hacer el sistema y cómo se verifica que lo hace bien".

Vibe coding no es lo mismo que autocompletado

Es habitual confundir el vibe coding con el autocompletado inteligente que ofrecen herramientas como GitHub Copilot desde 2021. La diferencia es de grado y de alcance. El autocompletado sugiere la siguiente línea o función mientras el desarrollador sigue escribiendo el resto a mano. El vibe coding, en cambio, delega unidades de trabajo completas —una función, un módulo, una funcionalidad entera— a un agente que puede leer el proyecto, escribir varios archivos, ejecutar comandos, ver los errores resultantes y corregirlos sin que el humano intervenga en cada paso.

Esta distinción importa porque cambia el rol del programador. En el autocompletado, el humano sigue siendo el autor principal del código y la IA es una ayuda puntual. En el vibe coding, el humano actúa más como director de producto y revisor de calidad: define el objetivo, evalúa el resultado, decide qué se acepta y qué se corrige, y la IA asume la autoría material de gran parte del código.

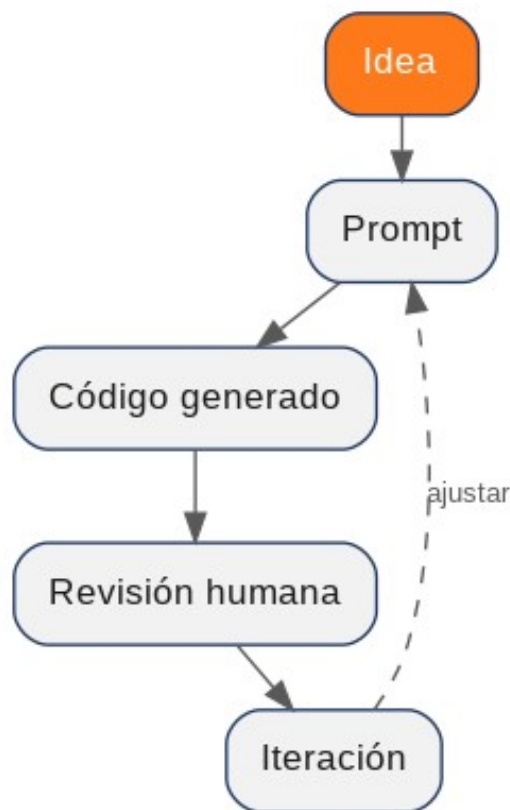


Figura 1.1 — El ciclo básico del vibe coding: de la idea a la iteración continua.

Los cuatro pilares del vibe coding

Para que este modo de trabajo funcione de forma fiable —y no se convierta en una sucesión de intentos aleatorios— conviene apoyarse en cuatro pilares:

- Comunicación precisa: cuanto más claro sea el objetivo que se transmite a la IA, menos iteraciones hacen falta para llegar a un resultado útil. Esto se desarrolla en profundidad en el Capítulo 5.
- Verificación constante: cada entrega de la IA se prueba, se ejecuta y se revisa antes de darla por buena. La confianza ciega en el código generado es la causa más común de fallos en producción.
- Contexto acumulado: los proyectos que funcionan bien con IA mantienen documentación y estructura que permiten a cualquier sesión nueva de trabajo "ponerse al día" rápidamente, sin depender de que el humano recuerde todo el histórico.
- Responsabilidad humana: quien aprueba un cambio es responsable de él, tanto si lo escribió una persona como si lo escribió un modelo de IA. El vibe coding no traslada la responsabilidad profesional, solo cambia quién teclea.

□ Idea clave

El vibe coding no elimina la necesidad de saber programar: la reduce en el día a día,

pero la hace imprescindible en los momentos de revisión, arquitectura y decisión. Cuanto mejor entiendas lo que la IA genera, mejor sabrás cuándo confiar en ello y cuándo no.

Quién practica vibe coding hoy

Aunque el término nació asociado a founders solitarios construyendo productos de fin de semana, en 2026 el vibe coding se practica en contextos muy distintos: equipos de producto que prototipan funcionalidades antes de comprometer recursos de ingeniería, agencias de desarrollo que reducen los tiempos de entrega a sus clientes, departamentos internos que automatizan procesos sin depender de un equipo técnico saturado, y desarrolladores experimentados que usan la IA para eliminar el trabajo repetitivo y concentrarse en las decisiones que de verdad requieren su criterio.

Lo que comparten todos estos casos es un cambio de cuello de botella: antes, la limitación para construir software era la disponibilidad de programadores; ahora, cada vez más, la limitación es la claridad con la que se define lo que hay que construir y la disciplina para revisarlo bien.

Un primer ejemplo de principio a fin

Para hacer tangible todo lo anterior, veamos un ejemplo mínimo. Supongamos que queremos una función que valide direcciones de email antes de guardarlas en una base de datos. En programación tradicional, escribiríamos la función nosotros mismos. En vibe coding, el proceso es distinto:

- El desarrollador describe el objetivo: "necesito una función en Python que valide direcciones de email, rechace formatos incorrectos y devuelva un mensaje de error claro en cada caso".
- La IA genera una primera versión de la función, junto con un pequeño conjunto de pruebas.
- El desarrollador ejecuta las pruebas y observa que un caso límite —una dirección con un punto justo antes de la arroba— no se maneja bien.
- El desarrollador comunica ese caso concreto a la IA, que ajusta la función y añade una prueba para ese caso específico.
- El desarrollador revisa el resultado final, lo entiende, y solo entonces lo integra en el proyecto.

```
def validar_email(direccion: str) -> tuple[bool, str]:
    """Valida una dirección de email básica.
    Devuelve (True, "") si es válida, o (False, motivo) si no lo es.
    """
    import re
    patron = r"^[a-zA-Z0-9_.-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-]+$"
    if not direccion or "@" not in direccion:
        return False, "Falta el símbolo @"
    if ".." in direccion or direccion.startswith("."):

```

```
        return False, "Formato de puntos inválido"
    if not re.match(patron, direccion):
        return False, "Formato de email no reconocido"
    return True, ""
```

Este ejemplo, deliberadamente sencillo, contiene ya los cinco pasos que se repiten en proyectos mucho más grandes: definir, generar, probar, corregir y revisar. La única diferencia entre este caso y uno complejo es la cantidad de ciclos necesarios, no la naturaleza del proceso.