Vibe Engineering: Aplicando la ingeniería al Vibe Coding

Alejandro Fernández Camello

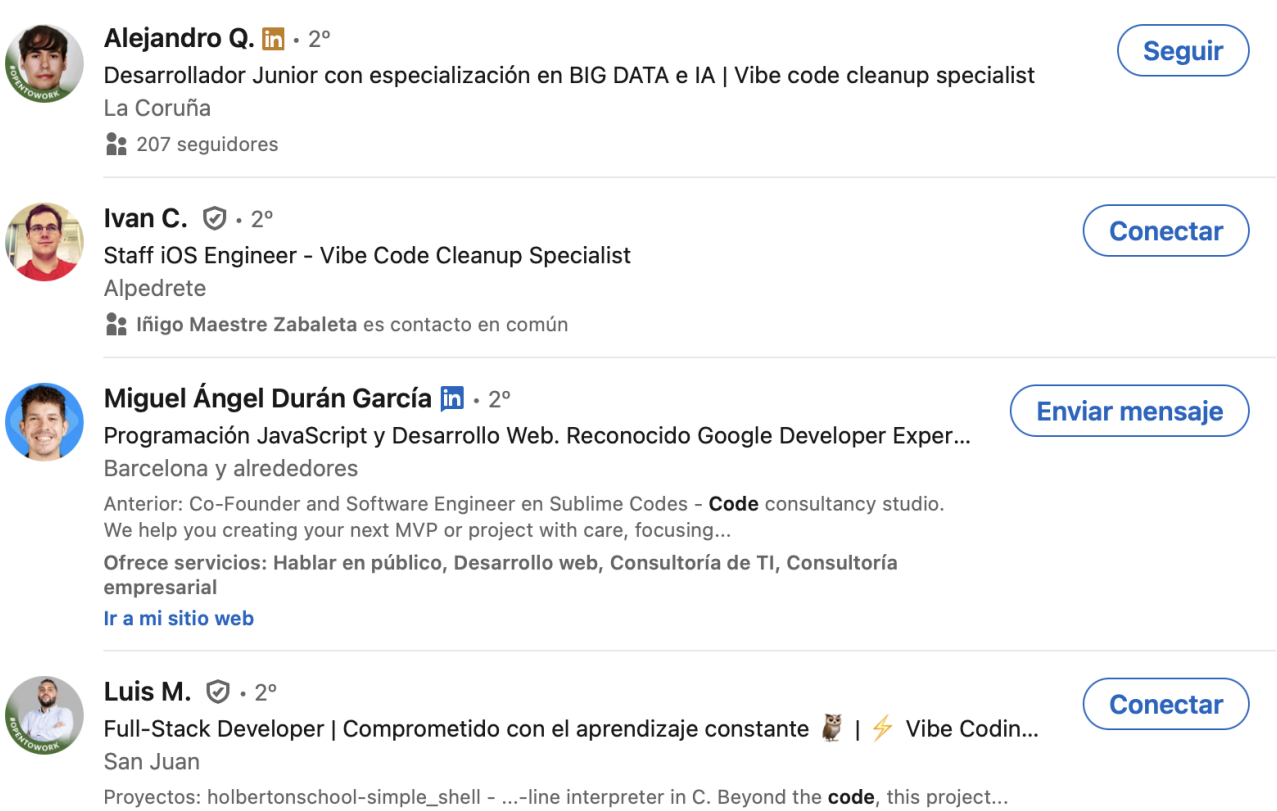
16 de octubre de 2025

Presentación

Alejandro Fernández Camello

CTO ZeroChats & Doctorando en Agentes de IA UCM

¿Qué es el Vibe Coding?



Alejandro Q. • 2º
Desarrollador Junior con especialización en BIG DATA e IA | Vibe code cleanup specialist
La Coruña
207 seguidores
Seguir

Ivan C. • 2º
Staff iOS Engineer - Vibe Code Cleanup Specialist
Alpedrete
Iñigo Maestre Zabaleta es contacto en común
Conectar

Miguel Ángel Durán García • 2º
Programación JavaScript y Desarrollo Web. Reconocido Google Developer Exper...
Barcelona y alrededores
Anterior: Co-Founder and Software Engineer en Sublime Codes - **Code** consultancy studio. We help you creating your next MVP or project with care, focusing...
Ofrece servicios: Hablar en público, Desarrollo web, Consultoría de TI, Consultoría empresarial
Ir a mi sitio web
Enviar mensaje

Luis M. • 2º
Full-Stack Developer | Comprometido con el aprendizaje constante | ⚡ Vibe Codin...
San Juan
Proyectos: holbertonschool-simple_shell - ...-line interpreter in C. Beyond the **code**, this project...
Conectar

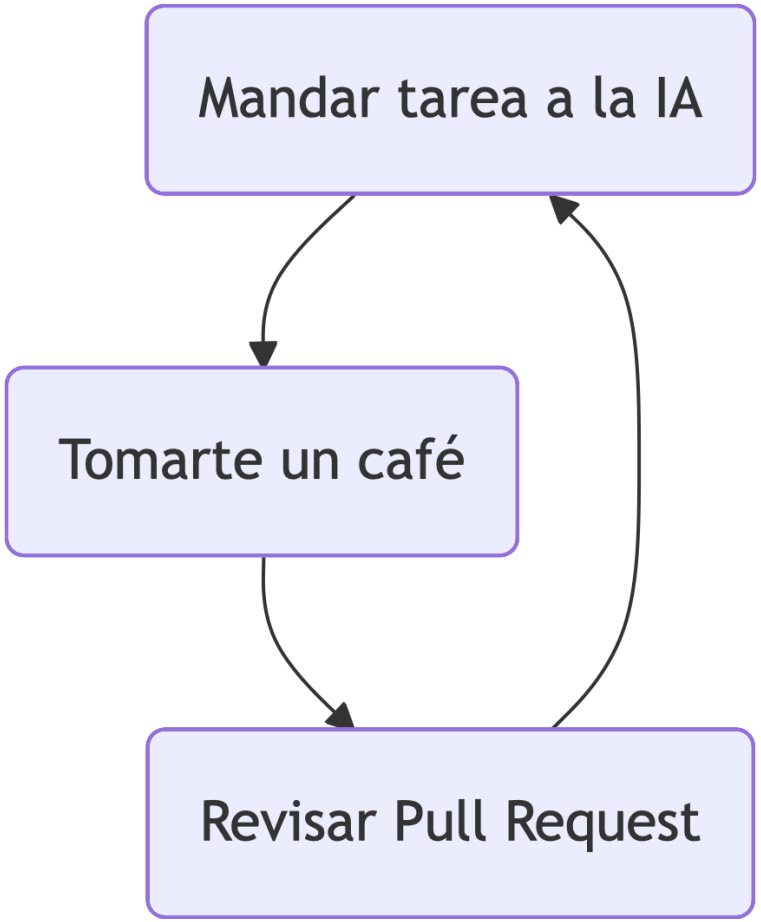
Vibe Engineering

Vibe Engineering = Vibe Coding + Ingeniería de Software

Índice

1. Introducción al Vibe Engineering
2. ¿Dónde aplicarlo?
3. Context Engineering
4. Testing
5. Conclusiones

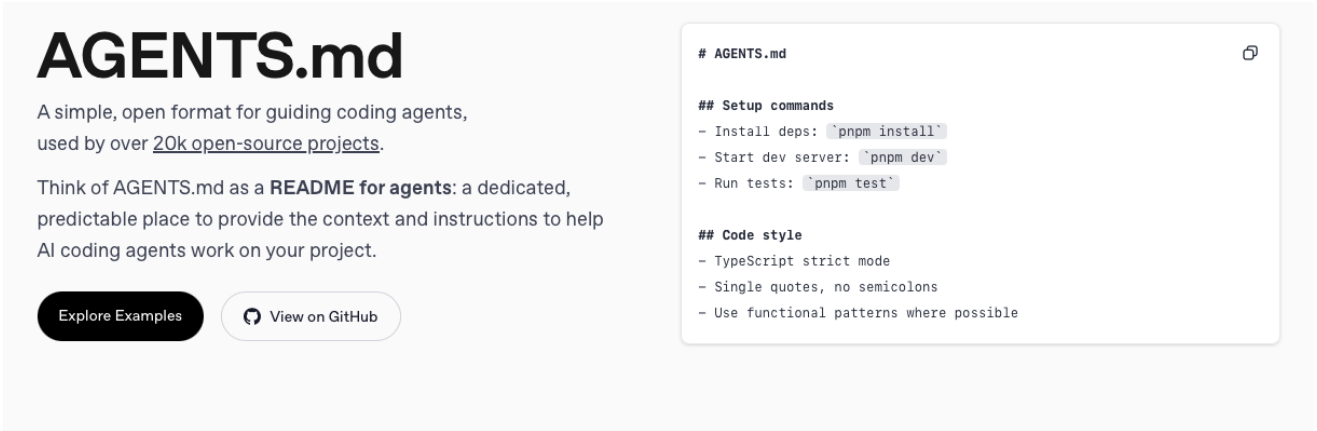
Flujo de trabajo



Configuración inicial

- Proyecto de fin de semana
- Proyecto nuevo a largo plazo
- Proyecto en progreso

El documento maestro



Why AGENTS.md?

README.md files are for humans: quick starts, project descriptions, and contribution guidelines.

AGENTS.md complements this by containing the extra, sometimes detailed context coding agents need: build steps, tests, and conventions that might clutter a README or aren't relevant to human contributors.

We intentionally kept it separate to:

- **Give agents a clear, predictable place for instructions.**
- **Keep READMEs concise and focused on human contributors.**
- **Provide precise, agent-focused guidance that complements existing README and docs.**

Rather than introducing another proprietary file, we chose a name and format that could work for anyone. If you're building or using coding agents and find this helpful, feel free to adopt it.

¿Qué debe incluir el documento maestro?

- Persona
- Estilo y criterios de calidad del código
- Qué comandos ejecutar para formatear, compilar, etc.
- Toda la información relevante sobre el proyecto

Índice

1 Introducción al Vibe Engineering

2 ¿Dónde aplicarlo?

3 Context Engineering

4 Testing

5 Conclusiones

Vibe Coding es como una herramienta

Hay que saber cuándo aplicarlo y cuándo no

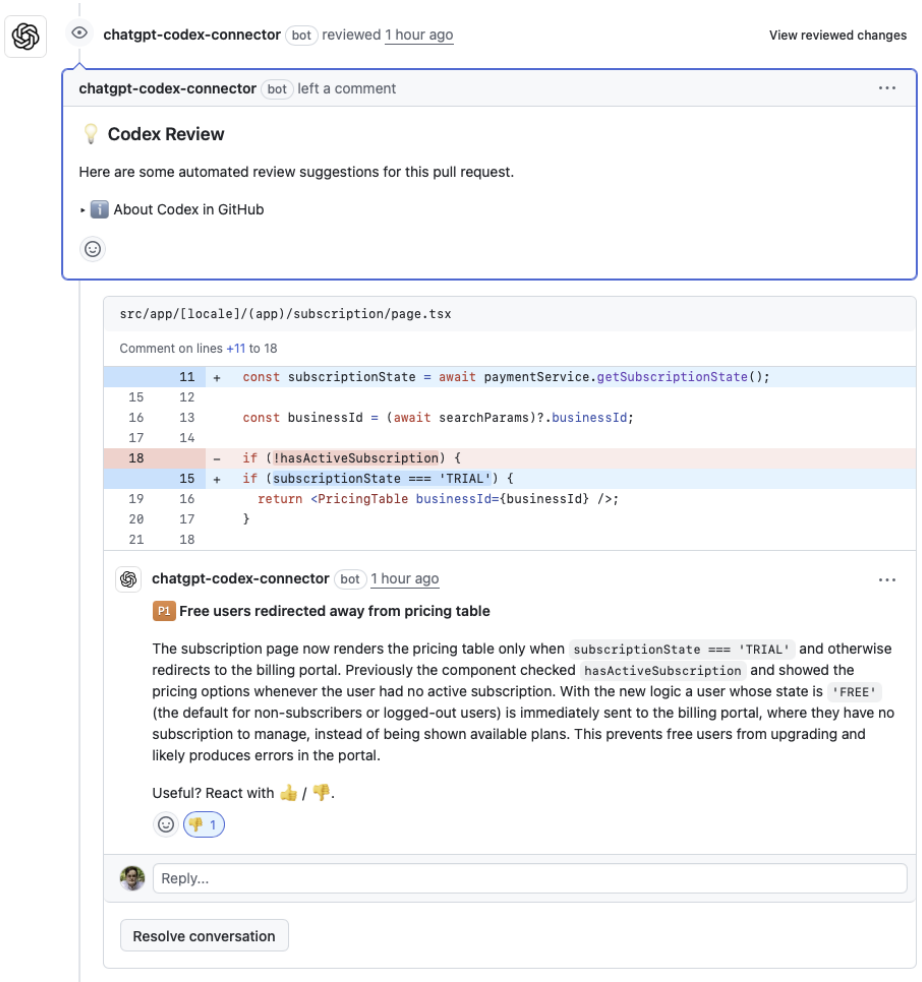
Cuándo no usarlo

- Lógica de negocio
- Modelos de datos
- Interfaces

Cuándo usarlo

- Tareas mecánicas simples
- Refactorizaciones
- Solucionar bugs
- Implementación de las interfaces
- ¿Pull requests?

Pull requests



Recomendación

En general, cuanto más grande sea la base de código más fácil será aplicar el Vibe Engineering

Índice

1. Introducción al Vibe Engineering
2. ¿Dónde aplicarlo?
3. Context Engineering
4. Testing
5. Conclusiones

¿Qué es Context Engineering?

Consiste en establecer la información a la que tendrán acceso los agentes de IA

¿Qué puede formar parte del contexto?

- Documento maestro
- Archivos de código
- Estructura de archivos
- Salida de la terminal
- Interfaz gráfica de la aplicación

¿Por qué es tan importante?

- Muchas veces cuando un agente falla en una tarea es por falta de información
- Asegurándonos de que tenga la información necesaria podremos evitar la mayoría de errores

Índice

1. Introducción al Vibe Engineering
2. ¿Dónde aplicarlo?
3. Context Engineering
4. Testing
5. Conclusiones

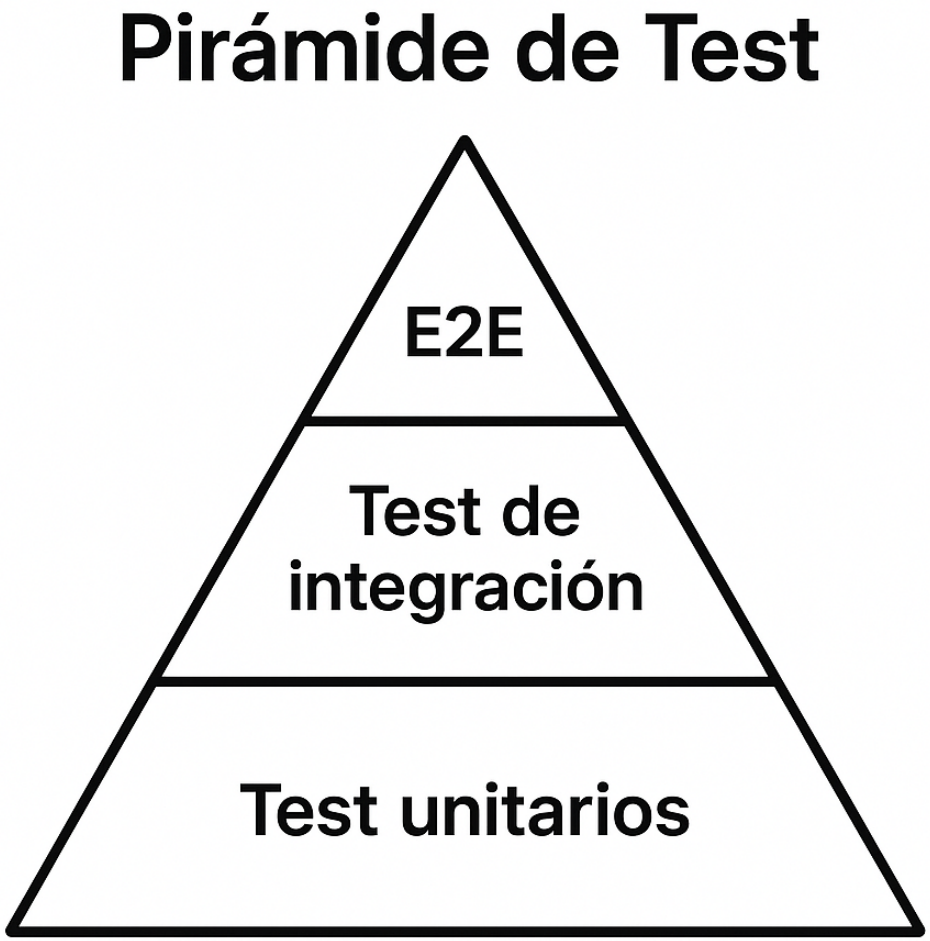
¿Por qué el testing es tan importante?

Hay un gran peligro de regresiones

¿Por qué el testing es tan importante?

- Hay un gran peligro de regresiones
- El agente necesitará tests para comprobar que va por el buen camino

La pirámide de los tests



Cambios en la estrategia de testing

- Mayor cantidad de tests (más fáciles de hacer y mayor peligro de regresión)
- Especialmente incrementar test unitarios (rápidos de ejecutar y fáciles de implementar para los agentes)
- Priorizar en los tests de integración los que impliquen usar LLMs
- Mantener tests e2e en los *happy paths*

Vibe Engineering con TDD

- TDD (Test Driven Development) a grandes rasgos consiste en hacer los tests antes que el código
- Desarrollo lento (gran número de iteraciones en añadir tests y modificar código)
- Puede ser un proceso ideal para los agentes

Índice

1. Introducción al Vibe Engineering
2. ¿Dónde aplicarlo?
3. Context Engineering
4. Testing
5. Conclusiones

Conclusiones

- El Vibe Coding ha llegado para quedarse
- Usando Vibe Engineering podemos lograr un código de mayor calidad en menor tiempo
- Se reduce enormemente el coste de refactorizar y añadir nuevos tests
- Los desarrolladores pasáremos a un rol más de *product manager*

Vayámonos al código

Plantilla Next.js

Plantilla Python